UNIVERSITÉ DU QUÉBEC À MONTRÉAL

IBC-GO: UN SYSTÈME D'AGENT ITINÉRANT

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
SERGE GIGUÈRE

AVRIL 2007

Remerciements

En préambule à ce mémoire, je souhaite adresser ici tous mes remerciements aux personnes qui m'ont apporté leur aide et qui ont ainsi contribué à l'élaboration de ce mémoire.

Tout d'abord Monsieur Abdel Obaid, directeur de ce mémoire, pour l'aide et le temps qu'il a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Aussi, j'adresse mes plus sincères remerciements à tous mes proches et amis qui m'ont soutenu et encouragé au cours de la réalisation de ce mémoire.

TABLE DES MATIÈRES

LISTE DES FIGURES

Figure Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ABone	Active Networks backBone
ACM	Association for Computing Machinery
ACMP	Agent Control Message Protocol
AMARRAGE	Architecture Multimédia & Administration Réparties sur un Réseau Actif à Grande Échelle
ANCORS	Adaptable Network COntrol and Reporting System
ANEP	Active Network Encapsulation Protocole
ANETd	Active NETwork deamon
ANS	Active Network Simulator
ANTLR	Another Tool for Language Recognition
ANTS	Active Node Transfer System
API	Application Program Interface
ASIC	Application Specific Integrated Circuit
BBN	Bolt, Beranek and Newman
BD	Base de Donnée
CAML	Categorical Abstract Machine Language
CFIP	Colloque Francophone sur l'Ingénierie des Protocoles
CORBA	Common Object Request Broker Architecture
CPL	Call Processing Language
CPU	Central Processing Unit
CSS	Cascading Style Sheet
DAN	Distributed code cashing Active Network
DARPA	Defense Advanced Research Projects Agency
DHCP	Dynamic Host Configuration Protocol
DOS	Denial-Of-Service
Diffserv	Differentiated services
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard

EJB	Enterprise Java Beans
EUNICE	EUropean Network of universities and companies in Information and Communication Engineering
FIFO	First In First Out
FTP	File Transfert Protocol
GCAP	Global Communication Architecture & Protocols
HABA	Hyper Active Beans component Architecture
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IBC-Go	Itinerary Based Computation and Go
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IOS	Internet Operating System
IP	Internet Protocole
IST	Information Society Technologies
JAAS	Java Authentication and Authorization Service
JAXP	Java API for XML Processing
JCE	Java Cryptography Extension
J2EE	Java 2 platform Enterprise Edition
JNI	Java Native Interface
JSP	Java Server Page
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
MAF	Multicast Actif Fiable
MAPS	Mobility Architecture for Programmable Services
MIT	Massachusetts Institute of Technologie
MOM	Message Oriented Middleware
MMUSIC	Multiparty Multimedia Session Control
MPEG	Moving Pictures Expert Group
MVC	Model-View-Controller

NS	Network Simulator
ORB	Object Request Broker
OS	Operating System
OSI	Open Systems Interconnexion
PIRST-ON	Programmable, Intermediate, Resilient, Self-configuring, Transparent Overlay Networks
PKI	Public Key Infrastructure
PLAN	Programming Language for Active Networks
QCM	Query Certificate Manager
QoS	Quality of Service
RAP	Routeur Assistant Protocol
RNRT	Réseau National de Recherche en Télécommunications
RPC	Remote Procedure Call
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
RTTs	Round-Trip Time
SAPF	Simple Active Packet Format
SARA	Simple Active Router Assistant
SDK	Software Development Kit
SIGCOMM	Special Interest Group on data COMMunications
SIP	Session Initiation Protocol
SLRP	Service Layer Routing Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SRI	Stanford Research Institute
SSL	Secure Socket Layer
TCL	Tool Command Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ITU	International Telecommunication Union

ITU-T	ITU Telecommunication Standardization Sector
WWW	World Wide Web
XML	Extensible Markup Language

RÉSUMÉ

La situation, les problèmes et les nouvelles tendances liés aux réseaux de télécommunications font en sorte qu'il est nécessaire de revoir la façon de développer des systèmes distribués. On peut actuellement constater: qu'il y a une omniprésence des réseaux, que ces réseaux sont de plus en plus intelligents, que ceux-ci ont de la difficulté à suivre l'évolution des standards, qu'il y a une augmentation de la quantité d'information transmise, que les connexions en périphérie d'un réseau comme l'Internet sont limitées, etc. Des recherches récentes relatives au code mobile, sous la forme d'agents mobiles et de réseaux actifs, offrent de nouvelles possibilités intéressantes pour le développement des systèmes distribués. La fusion de ces deux secteurs de recherche est susceptible d'amener de nouveaux mécanismes pour construire des systèmes distribués plus adaptés à la situation actuelle des réseaux de télécommunications. Ce mémoire présente IBC-Go (*Itinerary Based Computation and Go*), un système d'agents mobiles, basé sur une approche itinéraire, qui intègre les technologies de réseau actif. Ce système permet le développement de systèmes distribués qui tiennent davantage compte de la situation, des problèmes et des nouvelles tendances d'aujourd'hui; comparativement aux systèmes traditionnels. Pour montrer les possibilités du système IBC-Go et pour démontrer les avantages amenés par le système IBC-Go, plus particulièrement de démontrer le fait que le dit système amène une diminution des besoins en communication entre le nœud d'origine et les autres nœuds; 3 scénarios mettant en application le système IBC-Go ont été présentés. La présentation de chacun de ces scénarios a également été accompagnée de résultats numériqués qui ont permis de chiffrer les gains amenés par le système IBC-Go. La présentation des scénarios et des résultats numériqués a permis de répondre l'objectif qui était de démontrer que le système développé est davantage en mesure de solutionner les problèmes soulevés dans la problématique que les systèmes actuels, et plus particulièrement de diminuer les besoins en communication entre le nœud d'origine et les autres nœuds.

MOTS-CLES: Code mobile, agent mobile, agent itinérant, réseau actif, ANTS, IBC-Go.

CHAPITRE I

INTRODUCTION

La situation, les problèmes et les nouvelles tendances liés aux réseaux de télécommunications font en sorte qu'il est nécessaire de revoir la façon de développer des systèmes distribués. On peut actuellement constater: qu'il y a une omniprésence des réseaux, que ces réseaux sont de plus en plus intelligents, que ceux-ci ont de la difficulté à suivre l'évolution des standards, qu'il y a une augmentation de la quantité d'information transmise, que les connexions en périphérie d'un réseau comme l'Internet sont limitées, etc.

Des recherches récentes relatives au code mobile, sous la forme d'agents mobiles et de réseaux actifs, offrent de nouvelles possibilités intéressantes pour le développement des systèmes distribués. La fusion de ces deux secteurs de recherche est susceptible d'amener de nouveaux mécanismes pour construire des systèmes distribués plus adaptés à la situation actuelle des réseaux de télécommunications.

Ce mémoire présente IBC-Go (*Itinerary Based Computation and Go*), un système d'agents mobiles, basé sur une approche itinéraire, qui intègre les technologies de réseau actif. Ce système permet le développement de systèmes distribués qui tiennent davantage compte de la situation, des problèmes et des nouvelles tendances d'aujourd'hui; comparativement aux systèmes traditionnels.

Comme son nom l'identique, IBC-Go met un accent particulier sur le concept d'itinéraire. Avec celui-ci, un agent peut visiter plusieurs nœuds, sans avoir à communiquer avec son expéditeur original; ce qui contribue à faire diminuer le besoin en communication entre le nœud d'origine et les autres nœuds (là où les connexions sont généralement plus lentes et instables). En plus de l'emploi d'itinéraire, IBC-Go inclut d'autres mécanismes pour diminuer les besoins en communication et en ressources, comme la séparation de la

transmission du code de l'agent en tant que tel et l'emploi d'une mobilité faible pour les agents. Le premier mécanisme diminue les besoins en communication en diminuant le poids des agents, alors que le second mécanisme diminue les besoins en ressources en ne transportant pas les états des agents lors de la migration de ceux-ci.

Aussi, IBC-Go utilise un langage spécifique, appelé AL (*Agent Language*), pour le développement des agents mobiles. Le fait d'utiliser un langage spécifique pour le développement des agents mobiles permet de ne pas restreindre la programmation des agents au langage utilisé pour le développement du système sous-jacent et ne nécessite pas de comprendre les détails du fonctionnement dudit système [PEI 2003].

Enfin, étant donné les problèmes de sécurité inhérents à l'utilisation de technologies de code mobile, une attention particulière a été portée à la sécurité d'IBC-Go. Cela a mené à la conception de différents mécanismes de sécurité; comme l'inclusion d'un champ d'autorisation et de signatures électroniques.

1.1 Motivation

La principale motivation, dans ce travail de maîtrise, est de développer un système de code mobile plus apte à répondre aux besoins des réseaux d'aujourd'hui que les réseaux traditionnels comme l'Internet. Plus précisément, ce travail doit mener à la création d'un système d'agent mobile incluant une architecture basée sur un modèle de réseau actif.

En plus du développement d'IBC-Go, ce travail entend faire la démonstration que ce dernier est en effet plus apte à répondre aux besoins des réseaux d'aujourd'hui que les réseaux traditionnels comme l'Internet. Plus précisément, ce travail attend démontrer: que le système développé est plus à même de supporter le développement de certains types d'applications et que celui-ci diminue le besoin en communication entre le nœud d'origine et les autres nœuds (là où les connexions sont généralement plus lentes et instables);

comparativement aux systèmes traditionnels (client-serveur, évaluation à distance, code sur demande, etc.). Il aurait également été possible de démontrer d'autres avantages amenés par le système développé, comme la possibilité de déployer des standards plus rapidement et celle de répondre plus adéquatement aux nouvelles tendances, mais cela n'aurait pas tenu compte des limites de temps associées à ce travail.

Enfin, une autre motivation de ce travail est de développer un système qui inclut certaines qualités jugées essentielles. Ces qualités sont notamment: la portabilité, la sécurité et la facilité d'utilisation.

1.2 Problématique

Ici seront brièvement présentés, la situation, les problèmes et les nouvelles tendances liés aux réseaux de télécommunications. Relativement à la situation et aux problèmes liés aux réseaux de télécommunications, il sera notamment question de l'omniprésence et de l'évolution des réseaux, de l'évolution et du déploiement des standards, de l'augmentation de la quantité d'information transmise, et finalement, des limites des connexions en périphérie. Relativement aux nouvelles tendances liées aux réseaux de télécommunications, il sera question de l'augmentation des PDAs (*Personal Digital Assistant*) et de l'informatique nomade. Suite à cela, une solution possible, pour solutionner les problèmes rencontrés et répondre aux nouveaux besoins, sera présentée.

1.2.1 Omniprésence et évolution des réseaux

Aujourd'hui, les réseaux de télécommunications sont de plus en plus omniprésents. La plupart des organisations sont maintenant équipées d'un réseau local d'ordinateurs, en plus d'un circuit téléphonique connecté à un réseau public, et ces réseaux sont de plus en plus intereliés avec le réseau Internet; de même que les ordinateurs personnels que l'on retrouve

dans les maisons. Avec l'avènement de nouvelles technologies sans-fils, comme Bluetooth et la téléphonie de 3^{e} génération comme UMTS (*Universal Mobile Telecommunications System*), il est à prévoir que l'omniprésence des réseaux de télécommunications va aller en s'accroissant.

Jusqu'à un passé récent, les réseaux de télécommunications assuraient la fourniture d'un nombre limité de services essentiels comme la téléphonie de base et la transmission de données. Cette situation a évolué par suite de l'innovation technologique rapide et des nouvelles conditions du marché. Dans le nouveau contexte concurrentiel, les opérateurs sont, en effet, conduits à diversifier leur offre de services auprès de la clientèle. La structure classique des réseaux de télécommunications, avec des nœuds indépendants imposant des limitations dans l'offre de services, a évolué vers une structure dans laquelle le traitement de fonctions spécifiques est confié à des entités spécialisées commandant les nœuds adaptés en conséquence [TEN 1996]. Ce nouveau type de réseau est communément appelé: « réseau intelligent ».

1.2.2 Évolution et déploiement des standards

Au lieu de rester figés dans le temps, les standards, notamment ceux associés au *World Wide Web*, évoluent. Le *World Wide Web* a popularisé l'Internet, grâce à sa puissance et à sa simplicité et le consortium Web (W3C), de même que plusieurs autres groupes, travaillent à l'évolution des standards qui constituent le *World Wide Web*.

Malheureusement, l'évolution des standards dans le domaine des réseaux de télécommunications n'est pas automatiquement suivie d'une implémentation. Cela est notamment dû au fardeau que représentent leurs implantations sur les équipements d'un réseau. Il faut en effet que tous les équipements d'un réseau supportent un nouveau standard pour que celui-ci puisse être exploitable; ce qui est pratiquement impensable pour le réseau Internet [TAB 2002]. Si on prend exemple sur IPv6 et diffserv, ces derniers existent depuis

déjà quelques années, mais n'ont pu être implantés à la grandeur du réseau Internet. Il faut, en l'occurrence, adapter les réseaux de télécommunications pour que les nouveaux standards puissent être déployés plus rapidement.

1.2.3 Augmentation de la quantité d'information transmise

L'Internet, en tant que réseau des réseaux, est devenu largement accepté comme un médium très important pour tout type d'échange d'information. Le nombre d'individus et de compagnies qui offrent des services sur l'Internet augmente constamment. Alors qu'au départ, l'Internet offrait un nombre limité de types de services, comme le courrier électronique, nous pouvons maintenant voir que celui-ci offre de plus en plus de types de services; dont notamment des services de diffusion audio/vidéo. Cette augmentation du nombre d'individus et de compagnies qui offrent des services et l'apparition de nouveaux types de services, comme les services de diffusion audio/vidéo, amènent inévitablement une augmentation de la quantité d'information transmise via le réseau Internet.

1.2.4 Limites des connexions en périphérie du réseau

Le réseau Internet peut être divisé en deux zones: le centre et la périphérie. Au centre, la bande passante des connexions est très grande (en comparaison avec la périphérie) et ces dernières sont la plupart du temps stables; ce qui n'est pas le cas en périphérie. Tout porte à croire que ces différences entre le centre et la périphérie du réseau Internet ne vont pas s'amoindrir, mais vont au contraire s'accentuer. Il est en effet à prévoir que la bande passante du centre va considérablement augmenter afin de pouvoir répondre à la demande grandissante pour la transmission de grandes quantités de données; ce qui ne sera pas le cas pour la périphérie [BRA 2005]. Pour contourner ces limites, les systèmes distribués auraient ainsi avantage à limiter les échanges de données véhiculées par les connexions en périphérie.

1.2.5 Nouvelles tendances du réseau Internet

Depuis sa création, de nouvelles tendances ont vu le jour sur le réseau Internet. Deux de ces tendances seront ici décrites: l'augmentation des PDAs et l'informatique nomade.

Depuis quelques années, nous voyons apparaître de plus en plus d'appareils sans-fils, comme les PDAs: des appareils portables qui combinent ordinateur et téléphone et qui ont la capacité de se brancher à un réseau comme l'Internet [WEB 2005]. Selon un sondage entrepris au près 6 544 adultes, incluant 3 304 utilisateurs actifs d'Internet, 44% des utilisateurs d'Internet accédait à l'Internet via une connexion sans-fils[1] [SIA 2005].

Avec ce type d'appareil, la plupart des prémisses qui influencent la définition et l'évolution des systèmes distribués ne sont plus valides. Ces prémisses incluent: des communications rapides et fiables, des appareils robustes et riches en ressources, de même que des appareils situés à des locations fixes. Cela dit, les utilisateurs veulent avoir un environnement semblable à celui que l'on retrouve sur un appareil situé à une location fixe et muni de ressources riches et d'une connexion fiable et rapide. En conséquence, cet environnement doit, autant que possible, inclure les mêmes applications et possibilités offertes par les appareils situés à une location fixe [BRA 2005].

En plus de l'augmentation du nombre d'appareils sans-fils pouvant se brancher au réseau Internet, une autre tendance suscite actuellement beaucoup d'intérêts: l'informatique nomade. L'informatique nomade fait référence au fait que les utilisateurs se déplacent d'un endroit à un autre tout en travaillant. Par moment, un utilisateur peut être branché à un système via un ordinateur de bureau branché au réseau local d'une entreprise, alors que plus tard, le même utilisateur sera branché au même système via un PDA. Néanmoins, l'utilisateur

[1] Le sondage a été mené au près d'adultes se trouvant dans 12 centres urbains situés dans des pays comme le Canada, les États Unis et la France.

veut une synchronisation des appareils de sorte qu'il puisse continuer à travailler tout en changeant d'appareil [BRA 2005].

1.2.6 Solution possible

Pour solutionner les problèmes rencontrés et répondre aux nouveaux besoins, les systèmes distribués actuels doivent s'adapter. Plusieurs technologies ont la possibilité de solutionner les problèmes rencontrés et de répondre aux nouveaux besoins qui sont posés. Cela dit, ce n'est pas parce qu'une technologie a la possibilité de solutionner un problème qu'elle est pour autant la plus appropriée. Nous ne développons plus de gros systèmes en programmant en assembleur, même si cela demeure possible. Aujourd'hui, nous utilisons les technologies orientées objets, parce qu'elles sont plus proches des concepts et des besoins des clients, de même que de notre façon de penser durant le développement des systèmes [BRA 2005].

De même que les technologies orientées objets se sont démontrées plus appropriées pour le développement de gros systèmes, il est possible de croire que les nouvelles technologies relatives au code mobile soient plus appropriées pour le développement des systèmes distribués actuels et futurs. Des recherches récentes relatives au code mobile, sous la forme d'agents mobiles et de réseaux actifs, offrent de nouvelles possibilités intéressantes pour le développement des systèmes distribués. La fusion de ces deux secteurs de recherche est susceptible d'amener de nouveaux mécanismes pour construire des systèmes distribués plus adaptés à la situation actuelle des réseaux de télécommunications [COL 1998].

1.3 Objectifs

Les objectifs suivants sont poursuivis pour ce travail de maîtrise:

- développer un système d'agent mobile incluant une architecture basée sur un modèle de réseau actif;
- développer un système portable, sécuritaire et facile d'utilisation;
- démontrer que le système développé est davantage en mesure de solutionner les problèmes soulevés dans la problématique que les systèmes actuels.

L'objectif principal du présent travail est de développer un système d'agent mobile incluant une architecture basée sur un modèle de réseau actif. Avec ce système, il sera possible de créer et d'exécuter des agents mobiles sur les serveurs, postes de travail, PDA, routeurs et autres appareils reliés au réseau Internet.

Il est également attendu que le système développé inclut certaines qualités jugées essentielles, qui sont: la portabilité, la sécurité et la facilité d'utilisation. Par rapport à la portabilité, l'utilité du système développé sera proportionnelle à la possibilité qu'il puisse être installé sur les divers appareils reliés au réseau Internet. En conséquence, celui-ci devra pouvoir être installé, autant que possible, sur les dits appareils. Par rapport à la sécurité, il est important que le système développé puisse répondre aux besoins en sécurité des différents usagers. En conséquence, le système développé devra inclure des mécanismes pour répondre à ces besoins. Enfin, pour pouvoir être utilisé par des utilisateurs autres que des experts en informatique, le système développé devra être facile d'utilisation. Celui-ci devra paraître le plus simple possible pour les utilisateurs; masquant ainsi la complexité de celui-ci.

Enfin, le présent travail a également pour objectif de démontrer que le système développé est davantage en mesure de solutionner les problèmes soulevés dans la problématique que les systèmes actuels. Il est attendu que ce travail mène à des démonstrations de mises en application du système développé et à des mesures montrant les avantages amenés par le système développé par rapport aux systèmes actuels.

1.4 Méthodologie

Pour atteindre l'objectif principal, une revue de la littérature et des technologies de réseaux actifs sera faite. Le but de cette revue de la littérature est de donner des définitions claires des concepts fondamentaux utilisés dans ce mémoire et d'exposer les particularités des réseaux actifs et des agents mobiles. La revue des technologies de réseaux actifs, pour sa part, a pour but de mener à l'adoption d'un prototype de réseau actif qui servira de base pour le développement du présent système. Le choix du prototype à adopter se fera en fonction des possibilités offertes par celui-ci; principalement en ce qui a trait à la possibilité d'y implémenter un système d'agent mobile.

Pour la conception du présent système d'agent mobile, ce travail entend reprendre IBC-Go, un système initialement conçu par Abdel Obaid et Anna Cavali et décrit dans un article interne intitulé: « *IBC-Go: An itinerary-based mobile computing system.* ». La conception du système IBC-Go constitue un bon point de départ pour la conception du présent système et sera, au besoin, ajustée.

Relativement aux qualités jugées essentielles, le choix de l'environnement, dans lequel sera développé le système, se fera en respectant la portabilité qui en découlera. En conséquence, l'environnement choisi devra être compatible avec les divers appareils reliés au réseau Internet. Par rapport à la sécurité, la revue de la littérature inclura une section relative aux problèmes liés à la sécurité des agents mobiles. En lien avec cette revue, la conception du présent système sera ajustée pour répondre aux problèmes qui y sont soulevés. Enfin, par rapport à la facilité d'utilisation, le système travaillera le plus possible en arrière plan; laissant une interface utilisateur aussi simple que possible. Cela pourra se concrétiser par une abstraction, de la programmation des agents, des mécanismes de sécurité, etc.

Lors du développement du système, un cycle de développement en spirale, incluant une série de prototypes, est prévu. Cela permet d'obtenir un prototype fonctionnel et testable rapidement; ce qui est approprié, compte tenu des limites de temps et que le système à

développer constituera lui-même un prototype. Pour tester le système, celui-ci sera installé et exécuté sur les machines les plus diverses que possible. Cela contribuera à garantir sa compatibilité avec le plus d'appareils possibles.

Finalement, pour démontrer que le système développé est davantage en mesure de solutionner les problèmes soulevés dans la problématique que les systèmes actuels, des mises en application du système développé seront décrites et des mesures relatives à ces mises en application seront présentées. Ces dernières seront issues de résultats numériqués ou de simulations (avec *ns-2*).

1.5 Organisation du mémoire

En plus de l'introduction, ce mémoire inclut 5 chapitres:

- le deuxième chapitre contient la revue de la littérature et des technologies de réseaux actifs;
- le troisième chapitre présente la conception du système IBC-Go;
- le quatrième chapitre décrit la mise en œuvre du système IBC-Go;
- le cinquième chapitre est une courte conclusion qui résume ce mémoire et expose quelques réflexions.

À la suite de ces chapitres, les références ayant servi à la rédaction de ce mémoire, de même que des annexes, sont présentées. Ces dernières contiennent: la description du langage AL et les fonctions incluses avec ce langage, la présentation du paquet IBC-Go avec la taille de chacune de ces composantes, et finalement, le code ayant servi à l'implémentation du système IBC-Go.

CHAPITRE II

REVUE DE LA LITTÉRATURE

Le présent chapitre présente une revue de la littérature relative aux technologies de réseaux actifs et d'agents mobiles. Cela commence par la présentation des concepts fondamentaux et des liens existant entre ces concepts et le présent travail, pour ensuite présenter en détails ce qu'est un réseau actif et un agent mobile. Suite à cela, une série de prototypes de réseaux actifs sera présentée et analysée. Cette analyse mènera à l'adoption d'un prototype de réseau actif pour supporter le développement du présent système.

2.1 Les définitions des concepts fondamentaux

Trois concepts fondamentaux sont utilisés dans ce mémoire et sont en l'occurrence ici présentés, à savoir: code mobile, réseau actif et agent mobile.

a. Code mobile

Dans [WIK 2004], on définit le concept de code mobile comme suit:

> «*In computer science, mobile code is a general term for any executable software program that is sent via some computer network from one computer to another to be executed at the destination.*
>
> *Mobile code technology essentially encompasses three different paradigms: remote evaluation, code on demand, and mobile agents.*»

Dans [BRA 2005], la définition suivante du concept de code mobile est offerte:

> «*Mobile code is a technique in which code is transferred from the computer system that stores the code files to the computer system that will execute the code.*»

De ces deux définitions, il ressort qu'un code mobile est un code exécutable qui est transmis d'un ordinateur à un autre pour ensuite être exécuté. La première définition présente également le concept d'agent mobile comme étant un des paradigmes du concept de code mobile. Les deux autres paradigmes étant l'évaluation à distance et le code sur demande. Dans «*Mobile Agents: Basic Concepts, Mobility Models & the Tracy Toolkits*», le concept d'agent mobile est, dans le même sens, présenté comme étant un des paradigmes du concept de code mobile; même si cela ne figure pas dans la définition précédente.

Le concept de code mobile sera ici défini comme étant un code exécutable qui est transmis d'un ordinateur à un autre pour ensuite être exécuté. Comme il sera possible de le voir avec la définition d'un agent mobile, ce dernier peut être effectivement vu comme étant un paradigme du concept de code mobile.

b. Réseau actif

Dans [WET 1999], la définition suivante du concept de réseau actif est offerte:

> «*Active networks are a novel approach to network architecture in which customized programs are executed within the network.*»

Dans [SCH 2000], la définition suivante du même concept est offerte:

> «*Active Networks is a framework within which users inject programs contained in messages into a network capable of performing computations and manipulations on behalf of the user.*»

Finalement, dans [DEC 1998], la définition suivante est donnée:

«Active networks are packet-switched networks in which packets can contain code fragments that are executed on the intermediary nodes.»

Des deux premières définitions, il ressort qu'un réseau actif est une architecture, ou une plate-forme, dans laquelle des programmes, traitements ou manipulations, sont exécutés à l'intérieur du réseau. La deuxième définition va plus loin en précisant que les programmes sont contenus dans des messages issus des utilisateurs du réseau. Finalement, la troisième définition reprend l'idée de la fusion programme-message en indiquant que les paquets d'un réseau actif contiennent des fragments de code qui sont exécutés sur les nœuds intermédiaires; en l'occurrence, le réseau.

Pour les besoins du présent travail, un réseau actif sera défini comme étant un réseau dans lesquel les paquets peuvent contenir des fragments de code étant exécutés sur les nœuds actifs (routeur, commutateur, poste de travail, etc.).

c. Agent mobile

Dans [WIK 2005], on définit le concept d'agent mobile comme suit :

«In computer science, a mobile agent is a piece of computer software that is able to migrate (move) from one computer to another autonomously and continue its execution on the destination computer.

Mobile agents are a specific form of mobile code. However, in contrast to the Remote evaluation and Code on demand paradigms, mobile agents are active in that they may choose to migrate between computers at any time during their execution. This makes them a powerful tool for implementing distributed applications in a computer network.

Besides belonging the family of mobile code paradigms, mobile agents are also a specific form of software agents.»

Dans [BRA 2005], deux définitions du concept d'agent mobile sont notamment données:

> «*A mobile agent is a program that can migrate from a starting host to many other hosts in a network of heterogeneous computer systems and fulfill a task specified by its owner. It works autonomously and communicates with other agents and host systems. During the self-initiated migration, the agent carries all its code and data, and in some systems it also carries some kind of execution state.*»

> «*Mobile agents refer to self-contained and identifiable computer programs, bundled with their code, data, and execution state, that can move within a heterogeneous network of computer systems. They can suspend their execution on an arbitrary point and transport themselves to another computer system. During the migration the agent is transmitted completely, that is, as a set of code, data, and execution state. At the destination computer system, an agent's execution is resumed at exactly the point where it was suspended before.*»

Il ressort de ces trois définitions qu'un agent mobile est un programme, ou logiciel, informatique ayant la possibilité de migrer d'un ordinateur à un autre de façon autonome pour ensuite accomplir une tâche ou poursuivre une exécution. Par rapport à la première définition, les deux autres définitions mettent également l'accent sur l'aspect hétérogène des ordinateurs qui transitent les agents mobiles. Par rapport toujours à la première définition, les deux autres définitions précisent également qu'un agent mobile transporte du code, des données et possiblement l'état de son exécution. La troisième définition met par ailleurs davantage de l'avant l'importance du transport de l'état d'exécution que la deuxième définition en ajoutant que l'exécution se poursuit sur l'ordinateur destinataire là où elle a été interrompue sur l'ordinateur précédent. Enfin la deuxième définition présente le fait que les agents peuvent communiquer entre eux et avec les ordinateurs hôtes.

Pour le présent travail, la définition du concept d'agent mobile suivante sera donnée: un agent mobile est un programme informatique ayant la possibilité de migrer d'un ordinateur à un autre de façon autonome pour ensuite accomplir une tâche ou poursuivre une exécution. Cette définition n'indique pas comment un agent mobile suit son exécution d'un ordinateur à un autre. Un agent peut reprendre son exécution depuis le début à chaque nœud visité ou peut poursuivre son exécution là où elle s'est terminée au nœud précédent.

2.2 Les liens entre les concepts fondamentaux et le présent travail

Le présent travail concerne la mise en oeuvre un système d'agents mobiles basée sur un modèle de réseau actif. C'est une plate-forme d'agents mobiles en ce sens quelle permet de construire et d'exécuter des agents mobiles, des programmes informatiques ayant la possibilité de migrer d'un ordinateur à un autre de façon autonome pour ensuite accomplir une tâche ou poursuivre une exécution. Le système développé est basé sur un modèle de réseau actif en ce sens que les agents mobiles du système sont transportés dans des paquets qui peuvent être exécutés sur les nœuds actifs (routeur, commutateur, poste de travail, etc.).

Le lien entre les agents mobiles du système et le réseau actif est présenté à la figure 2.1. Cette dernière montre un poste de travail qui envoie un agent mobile vers un serveur, en passant par un routeur. Le poste de travail, le routeur et le serveur sont des nœuds actifs; ce qui implique que l'agent mobile sera exécuté sur chacun de ces trois appareils.

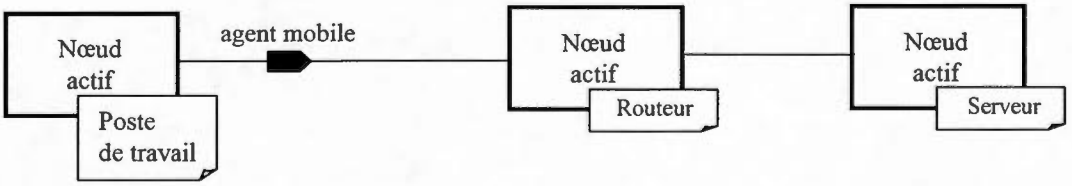

Figure 2.1: Lien entre les agents mobiles d'IBC-Go et le réseau actif

2.3 Les réseaux actifs

Le concept de réseau actif a émergé en 1994 des discussions entre les chercheurs de DARPA (*Defense Advanced Research Projects Agency*) sur les directions futures des systèmes réseaux [TEN 1997]. Les recherches sur les réseaux actifs ont mené à l'élaboration de trois objectifs spécifiques: l'introduction d'un code portable pour les logiciels réseaux, le

traitement des flux de données spécifiques à l'intérieur du réseau et finalement, la reconfiguration dynamique et efficace du réseau. Cette dernière pouvant être faite par un administrateur de réseau pour l'ensemble des flux de données ou par un utilisateur spécifique pour son propre flux de données [BER 2000].

Les sections suivantes montrent les caractéristiques propres au réseau actif. Cela comprendra: les types d'approches, l'architecture d'un nœud actif, les applications actives, l'environnement d'exécution, le système d'exploitation d'un nœud et les protocoles ANEP et SAPF.

2.3.1 Les types d'approches

Il y a différentes approches possibles par lesquelles un réseau peut être amené à être actif. Dans *A Survey of Active Network Research*, Tennenhouse et al. (1997) ont fait référence à deux: l'approche *commutateur programmable* (supportée par l'IEEE) et l'approche *capsule* (supportée par DARPA).

Avec l'approche *commutateur programmable*, également appelé *approche discrète*, le format actuel des paquets est conservé en intégrant un mécanisme discret qui se charge de télécharger les programmes. Un utilisateur envoie dans un premier temps ses *routines* de traitement et ensuite ses paquets de données [TEN 1997]. La séparation entre l'exécution et le chargement du programme peut être valable quand le chargement doit être contrôlé avec soin ou quand les programmes sont particulièrement volumineux. Cette approche est utilisée, par exemple, dans les réseaux intelligents standardisés par le ITU-T (*ITU* (*International Telecommunication Union*) *Telecommunication Standardization Sector*). Avec l'Internet, le chargement des programmes peut être restreint à l'opérateur d'un routeur qui est muni d'un accès avec lequel il peut charger dynamiquement les programmes. Des mesures de sécurité comme l'authentification de l'opérateur et la vérification du code des programmes pourraient être appliquées au chargement des programmes [TEN 1996].

Avec l'approche *capsule*, également appelée *approche intégrée*, les paquets des réseaux actuels sont remplacés par des programmes miniatures qui sont exécutés à chacun des nœuds rencontrés sur son chemin [TEN 1997]. Comme il sera possible de le voir avec la revue des prototypes de réseaux actifs, la plupart des systèmes de réseaux actifs utilisent une approche qui est un mélange de l'approche discrète et de l'approche intégrée.

2.3.2 L'architecture d'un nœud actif

L'architecture d'un nœud actif comprenant 3 composantes: les applications actives (AA), l'environnement d'exécution (EE) et le système d'exploitation du nœud; comme illustré à la figure 2.2 [BER 2000].

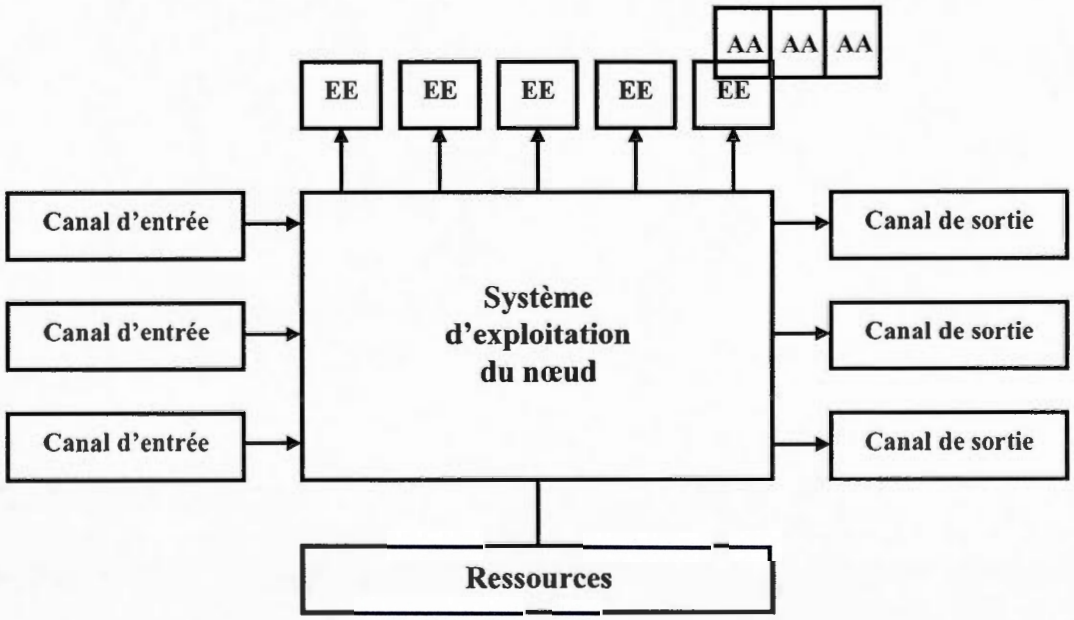

Figure 2.2: Architecture d'un nœud actif

Comme on peut le voir avec la figure 2.2, une ou des applications actives sont exécutées dans un environnement d'exécution et un ou des environnements d'exécution sont exécutés dans le système d'exploitation du nœud. Ce dernier inclut une série de canaux d'entrée et de sortie qui permet aux paquets d'entrer et de sortir du nœud actif.

2.3.3 Les applications actives

Les applications actives accomplissent les fonctions de communication en utilisant une combinaison d'acheminement de paquets et de traitement à l'intérieur des nœuds du réseau. Elles sont généralement écrites dans un langage restreint qui permet la portabilité et la sécurité, comme Java, et peuvent opérer sur le plan des données, de la gestion ou du contrôle. Les composantes d'une application active peuvent migrer d'un nœud à un autre et une application active peut créer un état de nœud persistant [BER 2000].

La mobilité peut se faire à différentes couches de représentation du programme. À un niveau plus élevé, un programme peut être exprimé dans langage interprété comme TCL (*Tool Command Language*). À un niveau intermédiaire, un programme peut être encodé dans un langage pré-compilé comme Java [TEN 1997]. Finalement, un programme peut être transformé dans un format binaire comme Omniware [TEN 1997], un outil développé par Colusa Software qui permet aux développeurs de logiciel de prendre des composantes écrites en C et C++ et de créer des composantes indépendantes du client [UMB 2003].

Une approche possible pour permettre l'interopérabilité entre les nœuds pourrait également être d'adopter un langage intermédiaire inter-nœud avec un mécanisme de traduction à la sortie et à l'entrée des nœuds. Les utilisateurs du réseau pourraient ainsi utiliser le langage désiré, si un mécanisme de traduction existe pour ce langage [TEN 1997].

2.3.4 L'environnement d'exécution

Une application active est exécutée dans un environnement d'exécution. Ce dernier est la partie stable du logiciel du nœud [BER 2000]. Il constitue une architecture virtuelle de réseau. C'est lui qui mémorise le code des services actifs et réalise les traitements adaptés aux paquets et aux services demandés (chaque paquet est associé à un traitement correspondant au service).

L'environnement d'exécution permet la création de services (interface de programmation), la gestion des services utilisés et le partage des ressources allouées entre les services [HAE 2002]. Il implémente généralement un ensemble de librairies qui permettent aux applications actives d'accéder aux ressources du nœud et du réseau et forme la première ligne de défense contre les violations de sécurité et la quantité de ressource retenue par une même application.

Un environnement d'exécution peut également supporter les entrée/sortie du réseau dans un plan de donnée virtuel ou dans le plan de donnée IP natif [BER 2000]. Équipé d'une interface correspondante, il peut créer les canaux qui lui sont reliés et décider quels flux doivent être acheminés directement (sans traitement).

Un même nœud peut supporter plus d'un environnement d'exécution. L'opérateur d'un réseau actif peut configurer le système d'exploitation d'un nœud par l'intermédiaire d'un environnement de gestion. Cela peut lui permettre notamment de gérer le partage des ressources entre les environnements d'exécution [HAE 2002].

2.3.5 Le système d'exploitation d'un nœud

Les environnements d'exécution d'un nœud exécutent un environnement créé par le système d'exploitation du nœud. Le système d'exploitation du nœud alloue les ressources du

réseau (canaux, protocoles de liaison) et du système (CPU (*Control Processing Unit*), mémoires) aux environnements d'exécution et offre une protection des ressources entre eux. Il doit supporter les entrée/sortie du réseau et offrir quelques mécanismes de communication entre les environnements [BER 2000]. Ainsi, c'est le système d'exploitation du nœud qui identifie les paquets entrants qui doivent être envoyés à l'environnement d'exécution (les paquets actifs) et ceux qui doivent être envoyés directement vers un canal de sortie (les autres).

Le système d'exploitation identifie les différents paquets selon deux méthodes de classement. La première méthode utilise des critères spécifiques à chaque environnement d'exécution (format du paquet, contenu de certains champs, etc.); alors que la seconde méthode utilise un des protocoles suivants au niveau du système d'exploitation: ANEP (*Active Network Encapsulation Protocole*) et SAPF (*Simple Active Packet Format*) [HAE 2002].

2.3.6 Le protocole ANEP

ANEP est un protocole indépendant des technologies utilisées. Il peut se placer au-dessus de Ethernet, IP, UDP, etc. Il comprend notamment un identificateur d'environnement d'exécution (Type ID) et des options comme les certificats de sécurité, l'adressage et le *checksum*. La figure 2.3 présente le format d'un paquet ANEP. Le champ *Drapeaux* identifie si le paquet doit être acheminé ou retiré par défaut.

La figure 2.4 montre le format des options ANEP. Le champ *Drapeaux* est sur 2 bits; le premier étant réservé à l'environnement d'exécution et le second n'étant pas couramment utilisé. Parmi les types d'options possibles, on retrouve: l'identificateur de la source, l'identificateur de la destination, l'*integrity checksum* [DRA 1997].

Version	Drapeaux	Type ID
Longueur de l'entête		Longueur du paquet
Options		
Charge (paquet actif)		

Figure 2.3: Format du paquet ANEP

Drapeaux	Type d'option	Longueur de l'option
Charge de l'option		

Figure 2.4: Format des options ANEP

2.3.7 Le protocole SAPF

Tout comme le protocole ANEP, le protocole SAPF est un protocole indépendant des technologies utilisées. Le format de l'en-tête SAPF est réduit au strict minimum: un champ *version* (sur 1 bit) et un champ *sélecteur* (de 63 bits). Les bits de 0 à 47, du champ *sélecteur*, servent à identifier l'environnement d'exécution (équivalent au champ *Type ID* d'ANEP) et les bits de 48 à 50, le type du paquet: statique, émetteur ou récepteur. [DEC 1998]. La figure 2.5 présente le format d'un paquet SAPF.

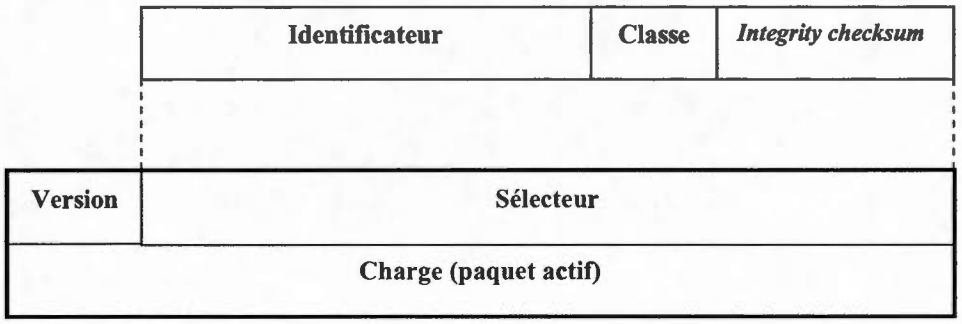

Figure 2.5: Format du paquet SAPF

2.4 Les agents mobiles

C'est en 1994 que James E. White publia un *white paper*, publié en 1996 dans *Software Agents* de [WHI 1996], qui initia la recherche dans le domaine des agents mobiles. Dans ce papier, James E. White introduisait la technologie *Telescript* qui comprenait un environnement et un langage de programmation pour les agents mobiles. Par la suite, d'autres prototypes d'agent mobile ont été développés. La plupart d'entre eux utilisent les nouvelles possibilités offertes par les technologies Java, notamment la portabilité [BRA 2005].

La présente section explique les particularités des agents mobiles. Cela comprendra: la terminologie associée à ce type de logiciel, la structure générale des agents mobiles, l'infrastructure associée à la migration des agents, les langages de programmation utilisés par les systèmes d'agent mobile, la communication entre les agents mobiles, les avantages amenés par les agents mobiles, et finalement, la sécurité des systèmes d'agents mobiles.

2.4.1 La terminologie

Ici seront définis les termes qui sont propres aux systèmes d'agents mobiles. Les définitions de ces termes sont présentées au tableau 2.1. Ces termes se retrouvent dans la plupart des écrits relatifs aux agents mobiles.

Toolkit pour agent mobile	Un *toolkit* pour agent mobile est un produit permettant de construire et d'exécuter des agents mobiles. Il existe actuellement plusieurs *toolkit* pour agent mobile. À titre d'exemple, mentionnons : Aglets de IBM, Voyager de Recursion et Tracy de l'université de Jena.
Agence	Une agence est un environnement d'exécution de lequel les agents mobiles sont exécutés. Cet environnement contrôle l'exécution des agents et offre des fonctionnalités pour la communication, la sécurité et la migration des agents. D'autres auteurs, comme [WHI 1996] et [LAN 1998], utilisent les notions de *place* et de *moteur* pour faire référence à une agence.
Agence expéditrice	Durant le processus de migration, l'agence quittée par l'agent est appelée agence expéditrice.
Agence réceptrice	Durant le processus de migration, l'agence qui reçoit l'agent est appelée agence réceptrice.
Protocole de migration	Durant le processus de migration, l'agence expéditrice et l'agence réceptrice échangent de l'information au sujet de l'agent qui doit migré. Pour cela, un protocole de migration est utilisé par les agences.
Agence d'origine	L'agence d'origine correspond à l'endroit où est créé un agent. Habituellement, un agent retourne à son agence d'origine après avoir complété sa tâche.
Système d'agent mobile	Toutes les agences pouvant échanger des agents mobiles forment un réseau logique appelé système d'agent mobile. Chaque agence du système est accessible via un URL spécifique. Cet URL sert également de nom pour l'agence.
Propriétaire d'agent	Le propriétaire d'un agent correspond à l'utilisateur qui a créé l'agent. Les informations relatives au propriétaire d'un agent peuvent être utilisées par les agences pour l'assignation des droits.

Tableau 2.1: Définitions des termes propres aux systèmes d'agents mobiles

Nom de l'agent	Chaque agent possède un nom spécifique. Ce nom est défini par le propriétaire de l'agent. Le nom de l'agent est nécessaire pour pouvoir identifier sans équivoque un agent à toutes les agences d'un système d'agent mobile.
Serveur de code	En plus d'héberger les agents, les agences peuvent également contenir le code des agents. Les agences qui contiennent le code d'agents sont appelées serveurs de code.

Tableau 2.1 (suite)

2.4.2 La structure

Un agent mobile est constitué de trois composantes: le code, les données et l'état de l'exécution. Le code contient la logique de l'agent et peut être utilisé par plus d'un agent. La transmission du code et de l'agent sont généralement séparées; ce qui permet de diminuer le poids de l'agent. Très souvent, le code n'est transmis qu'une seule fois et emmagasiné dans les agences pour une utilisation future. Dans le cas où le code est séparé, celui-ci doit pouvoir être identifié sans équivoque par toutes les agences d'un système d'agent mobile; comme pour les agents [BRA 2005].

La seconde composante, les données, correspond à la valeur des variables de l'agent [BRA 2005]. Certains auteurs, comme [WHI 1996] et [LAN 1998], appèlent les données: *état de l'objet.*

La troisième composante, l'état d'exécution, correspond à l'information qui est contrôlée par le processeur et le système d'exploitation [BRA 2005]. L'état d'exécution n'est pas toujours transmis avec l'agent mobile; comme il sera possible de le voir à la section suivante.

2.4.3 La migration

Le processus de migration peut être découpé en six étapes. Ces six étapes, inspirées des processus décris par [LAN 1998], sont: l'initialisation, la sérialisation, le transfert, la réception, la désérialisation, l'exécution. L'initialisation démarre lorsque l'agent annonce à l'agence son intention de migrer. Avec cette annonce, l'agence doit interrompre l'exécution de l'agent en vue de la sérialisation de l'agent. À la sérialisation, les données et l'état d'exécution avec une forte mobilité (voir plus loin) de l'agent sont transformées en séquence de bits pour leur transfert. Lorsque du transfert, la séquence de bits issue de la sérialisation est transmise à l'agence réceptrice via un protocole de migration. À la réception de l'agent, l'agence réceptrice vérifie l'identité de l'agent et les droits qui lui sont associés. En fonction de cela, l'agent sera accepté ou non. Après la réception de l'agent, celui-ci est désérialisé en vue de réétablir les données, et l'état d'exécution avec une forte mobilité (voir plus loin), de l'agent transmis. Après la désérialisation, l'agent poursuit son exécution.

Il y a deux approches pour la mobilité des agents: forte et faible. Avec la migration forte, l'état d'exécution de l'agent est capturé et envoyé avec le code vers sa destination. Cet état est réétabli lorsque l'agent atteint sa destination. La migration forte peut être exigeante en terme de traitement et de temps; surtout si la taille des états est importante. Pour plus d'efficacité, la mobilité faible est souvent utilisée. Avec cette dernière, l'état d'exécution de l'agent n'est pas transmis avec l'agent et celui-ci est réinitialisé lorsqu'il arrive à une autre agence. [LEV 2001].

Différents patrons de mobilité sont utilisés par les agents mobiles. La mobilité des agents mobiles peut être caractérisée par une série de destinations visitées par l'agent et par un ordre dans lequel la visite peut se faire. La série de destinations est communément appelée *itinéraire*. L'utilisation d'un itinéraire permet à un agent de visiter plusieurs agences avant de retourner à son agence d'origine; ce qui contribue à faire diminuer le besoin en

communication entre l'agence d'origine et les autres agences. Relativement à l'itinéraire et à l'ordre des destinations, trois types de mobilité peuvent être observés:

- mobilité avec itinéraire statique et ordre statique,
- mobilité avec itinéraire statique avec ordre dynamique,
- mobilité avec itinéraire dynamique [LEV 2001].

2.4.4 Les langages de programmation pour les agents mobiles

Le langage de programmation Java est devenu un standard de fait pour le développement de systèmes d'agents mobiles. Depuis 5 ans, presque tous les systèmes d'agents mobiles sont développés avec le langage Java. La plupart d'entre eux utilisent également le langage Java pour la création des agents mobiles. Cela s'explique en bonne partie dû au fait que le langage Java est portable et qu'il inclut différents dispositifs pour supporter la migration des processus [BRA 2005].

Bien que le langage Java soit utilisé pour le développement des systèmes d'agent mobile, il est possible de faire appel à d'autres langages pour la création des agents mobiles proprement dit. Pour cela, des compilateurs ou des interpréteurs peuvent être inclus dans les agences du système. Ces compilateurs ou interpréteurs permettraient en l'occurrence de faire le lien entre le langage utilisé pour la création des agents mobiles et celui utilisé pour le développement du système. Le fait de séparer la programmation des agents mobiles de la programmation du système permet de ne pas restreindre la programmation des agents au langage Java et ne nécessite pas de comprendre les détails du fonctionnement du système pour la programmation de ces derniers.

2.4.5 La communication

Les agents mobiles peuvent communiquer entre eux; qu'ils se trouvent à une même agence ou à des agences différentes. En général, les échanges entre les agents sont de type *peer-to-peer* ou à diffusion multiple (*multicast*). Différentes techniques peuvent être utilisées pour communiquer avec les agents distants. [BRA 2005] en propose cinq:

- l'approche serveur central, qui emmagasine la location des agents dans une base de données centrale;
- l'approche serveur d'origine, qui emmagasine la location d'un agent à l'agence d'origine;
- l'approche pointeur expéditeur, qui envoie les messages le long des agences visitées par l'agent;
- l'approche à large diffusion (*broadcast*), où les messages sont envoyés en parallèle à toutes les agences;
- l'approche hiérarchique, qui utilise une structure en arbre pour la diffusion des messages.

2.4.6 Les avantages

Les systèmes d'agents mobiles amènent plusieurs avantages par rapport aux systèmes traditionnels. Ici sont présentés quelques-uns de ces avantages; en l'occurrence: l'harmonisation des communications hétérogènes, la réduction de la communication, l'enrichissement des interfaces, la possibilité d'utiliser des applications temporaires, et enfin, la possibilité d'utiliser des données intelligentes (qui contiennent de la logique).

a. Les communications hétérogènes

Reliés ensemble, deux systèmes révèlent rapidement des incompatibilités de langages et de formats de données; incompatibilités qui se multiplient avec l'ajout d'autres systèmes. Les agents mobiles peuvent solutionner ce genre de problèmes en jouant le rôle d'intermédiaire entre les systèmes hétérogènes. La seule exigence est que l'agent puisse être exécuté sur chaque type de système [KNA 1996].

b. La réduction de la communication

Un des avantages clés des agents mobiles est de pouvoir réduire les coûts en communication. En effet, dans un système structuré autour des agents mobiles, un tiers peut se déplacer de l'autre côté d'une connexion pour éviter d'utiliser celle-ci. Ainsi, les tiers peuvent interagir, même si les connexions sont de mauvaises qualités [KNA 1996].

c. L'enrichissement des interfaces

Les agents mobiles permettent la présentation d'interfaces plus sophistiquées. Si on prend pour exemple une interface de réalité virtuelle, cette dernière demanderait normalement trop de traitement et de bande passante pour qu'un serveur puisse la fournir à distance. En envoyant un agent mobile contenant l'interface au client, cela devient davantage possible [KNA 1996].

d. Les applications temporaires

Un agent mobile n'a pas besoin de faire partie d'une application. Il peut former un tout et n'avoir besoin d'aucune communication. Télécharger un tel agent n'est pas différent

de télécharger une application d'un site FTP (*File Transfert Protocol*). L'avantage de l'utilisation des agents mobiles est que ceux-ci sont plus faciles à télécharger et ne nécessitent pas d'installation dans le système hôte [KNA 1996].

e. Les données intelligentes

L'association de codes et de données offre un moyen pour les données de savoir comment se traiter elles-mêmes. Un exemple pour cela est le standard de compression vidéo MPEG 4 (*Moving Pictures Expert Group* 4), où l'algorithme de décompression est empaqueté avec les données [KNA 1996].

2.4.7 La sécurité

Il va de soi que des systèmes d'agents mobiles introduisent différents problèmes de sécurité. Il y a principalement trois catégories de menaces reliées aux technologies d'agents mobiles:

- une attaque d'un agent vers une agence,
- une attaque d'une agence vers un agent,
- une attaque d'un agent vers un agent [RAO 2004].

En ce qui concerne l'attaque d'un agent vers une agence, on peut retrouvrer principalement des DOS (*Denial-Of-Service*) et des attaques de personnification d'agent. Le premier type d'attaque, DOS, est provoqué par le fait que les agents font travailler inutilement et volontairement les agences. Par rapport au travail qui est demandé aux agences, il est également possible que les agents fassent, volontairement ou non, une mauvaise utilisation des ressources; ce qui contribue à rendre les agences moins performantes

et qui peut ultimement mener à un DOS. La deuxième attaque, la personnification, se caractérise par le fait qu'un agent emprunte une identité autre que la sienne. Ce faisant, il peut avoir accès à des services et des informations confidentielles et possiblement modifier ces informations [RAO 2004].

L'autre catégorie d'attaque, l'attaque d'une agence vers un agent, concerne entre autres le fait qu'une agence puisse modifier le code d'un agent. Le code ainsi introduit peut devenir à la fois nuisible pour le créateur de l'agent modifié, les autres agences visitées par l'agent et les autres agents avec lesquels l'agent modifié interagit [RAO 2004]. Ce dernier point constitue l'attaque d'un agent vers un autre agent.

Comme il sera possible de le voir au chapitre suivant, différents mécanismes peuvent être mis en place pour contrer ces attaques.

2.5 Les prototypes

Le présente section présente 4 prototypes de réseaux actifs: *Switchware*, ANTS, AMARRAGE et *Smart Packets*. Une analyse comparative suivra cette présentation. Cette dernière aura pour but de mener à l'adoption d'un prototype de réseau actif qui servira de base pour le développement du présent système. Le choix du prototype à adopter se fera en fonction des possibilités offertes par celui-ci; principalement en ce qui a trait à la possibilité d'y implémenter un système d'agent mobile.

2.5.1 Switchware

À l'université de Pennsylvanie, un projet nommé *SwitchWare* a mené au développement d'une approche commutateur programmable qui permet de télécharger des

modules marqués d'une signature électronique à l'intérieur des nœuds d'un réseau [TEN 1997].

Switchware est basé sur une architecture à trois niveaux (voir figure 2.6). Au niveau supérieur, il y a des programmes contenus dans des capsules et écrits dans un langage fonctionnel, PLAN (*Programming Language for Active Networks*)[2]. Au second niveau, il y a des *extensions actives* constituées de programmes dynamiquement téléchargeables et écrits dans le langage Caml ou Java[3]. Les extensions actives sont utilisées pour offrir des services qui peuvent être invoqués par les capsules. Un contrôle d'accès au système est exercé sous la base de privilèges et de certificats[4]. La collaboration des paquets actifs et des services est appelée *switchlet*. Enfin, le dernier niveau de l'architecture est constitué par l'infrastructure qui supporte l'allocation des ressources [ALE 1998].

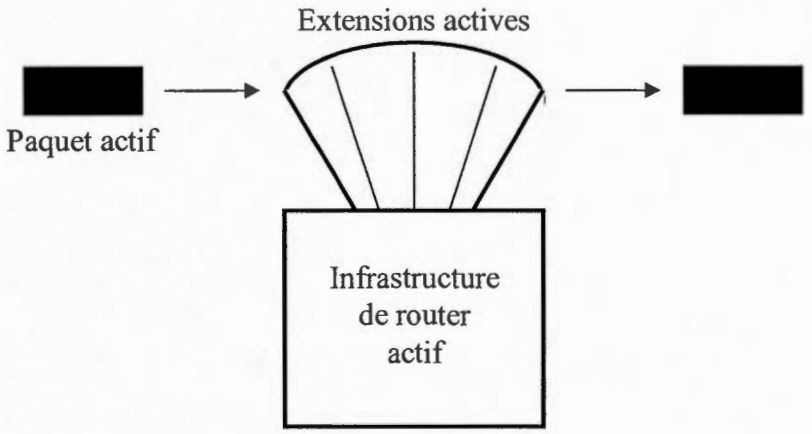

Figure 2.6: Architecture *Switchware* [ALE 1998]

[2] *Switchware* distribue également une architecture qui permet de coder des capsules en Caml. Celle-ci s'appelle ALIEN.

[3] La version Java n'est plus supportée. En conséquence, cette version ne fonctionne pas avec les machines virtuelles supérieures à 1.1.

[4] *Switchware* est pourvu d'un gestionnaire de certificat QCM (*Query Certificate Manager*).

À la base, les capsules[5] sont encapsulées dans des paquets UDP ou TCP (*Transmission Control Protocol*) pour ensuite être envoyées dans des paquets IP. Il est par contre possible d'envoyer directement des capsules sans utiliser UDP, TCP ou IP avec PLANet (seulement disponible en version Caml). PLANet est un Internet actif où tous les paquets sont des capsules PLAN. Les fonctions de base, comme le routage et la résolution d'adresse, sont implémentées dans des extensions actives. PLANet s'exécute directement au-dessus de Ethernet [ALE 1998].

L'utilisation de *Switchware* est relativement simple. On instancie la classe *ARMain* pour activer un nœud actif et une autre classe, *PLANStart*, est utilisée pour installer un service. Par exemple:

```
java PLANStart host installServices
      (["NewService",
      getBlobFromFile("NewService.class")])
```

Cela aura pour effet de charger la classe *NewService* sur *host*. Ce service pourra ensuite être invoqué à partir d'une capsule PLAN [HIC 1997].

Switchware est distribué avec une documentation à la fois simple et complète; ce qui permet de l'exploiter rapidement et efficacement.

2.5.2 ANTS

ANTS est une implémentation de réseau actif qui comprend: un système d'exploitation, un environnement d'exécution et des outils (*toolkit*) pour développer des applications actives [WET 1997]. Il a été développé par l'équipe de Tennenhouse au MIT (*Massachusetts Institute of Technology*). ANTS a été développé en Java, principalement à

[5] Il n'a malheureusement pas été possible de trouver une illustration du format des capsules PLAN.

cause de sa rapidité de développement et de ces possibilités d'adaptation à des environnements hétérogènes [TEN 1997].

ANTS utilise des capsules pour implanter de nouveaux protocoles ou services. Une application ANTS peut obtenir un service de réseau personnalisé en envoyant et en recevant des capsules via un nœud actif. Un nouveau service résulte de l'exécution de l'ensemble des programmes associés aux capsules d'un même protocole. La figure 2.8 présente la structure unissant les capsules aux protocoles. Comme on peut le voir avec cette figure, un protocole est décomposé en groupes de codes pour ensuite être fragmenté en unités de traitement qui sont expédiées dans les capsules [WET 1998]. Un groupe de code représente une unité de transfert. Cela signifie que celui-ci doit être véhiculé dans son entier; même s'il est décomposé en partie pour la transmission. Comme on peut également le voir, un protocole représente une unité de protection. Ainsi, une capsule peut communiquer avec une autre capsule d'un même protocole, mais elle ne peut par contre pas communiquer avec une capsule d'un autre protocole [SPA 2000].

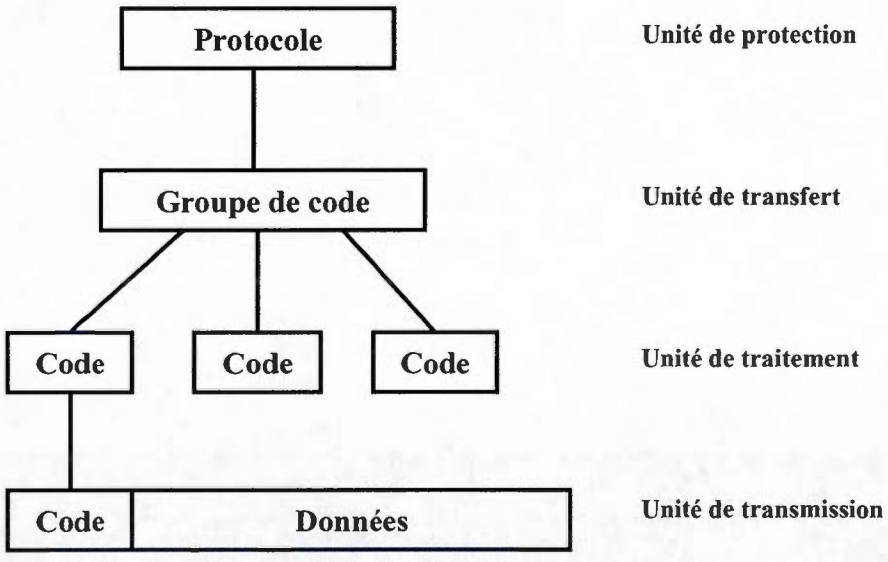

Figure 2.8: Relation entre capsule et protocole [SPA 2000]

Le format de la capsule est montré à la figure 2.9. Le champ *type* identifie le protocole, le groupe code et le programme. Ce dernier contient un numéro qui est le résultat d'une fonction de hachage appliquée à un numéro de protocole, de groupe code et de programme. La capsule ANTS est encapsulé dans un paquet ANEP qui est lui-même encapsulé dans un paquet UDP, pour ensuite être véhiculé dans un paquet IP standard.

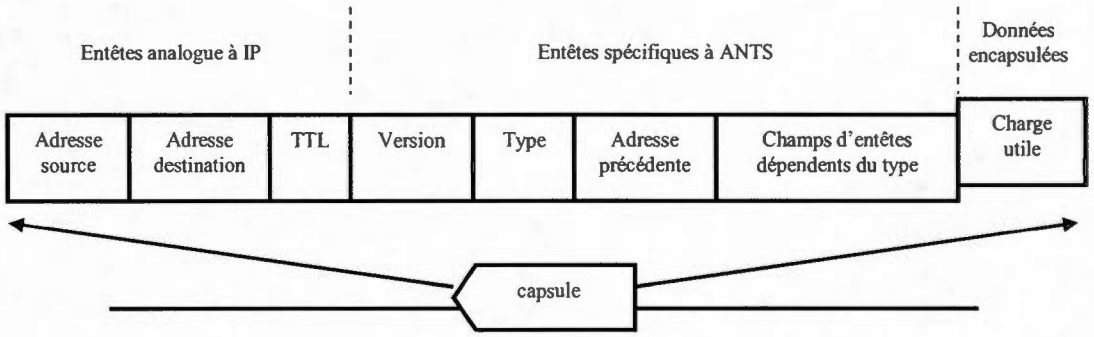

Figure 2.9: Format de la capsule ANTS [WET 1999]

Pour éviter d'avoir à télécharger un même groupe de codes plus d'une fois, celui-ci est emmagasiné sur chacun des nœuds visités. Lorsqu'une capsule fait référence à un groupe de codes qui n'est pas reconnu par un nœud, ce dernier enverra une demande de chargement au nœud actif précédent. À la réception de la demande, le nœud actif précédent répondra par une série de capsules véhiculant les programmes appartenant au groupe de code. Les codes emmagasinés ne peuvent être manipulés que par les capsules appartenant au même protocole que celle qui l'a emmagasiné [SPA 2000].

Tout comme, *Switchware*, ANTS est relativement facile d'utilisation. Un protocole est développé en implémentant la classe *Protocole*; une capsule est développée en implémentant la classe *Capsule*, et une application est développée en implémentant la classe *Application*. Pour démarrer un nœud ANTS, il suffit d'instancier la classe *ConfigurationManager*: Un fichier routage est utilisé comme table de routage [WHI 2001].

2.5.3 AMARRAGE

Dans *AMARRAGE: Déploiement et expérimentation d'un réseau actif à grande échelle*, [ZEB 2002] présente, AMARRAGE (*Architecture Multimédia & Administration Réparties sur un Réseau Actif à Grande Échelle*), un projet qui a débuté en 1999 et qui a donné naissance à l'une des premières plates-formes du RNRT: l'AMARRAGEBone (voir figure 2.10).

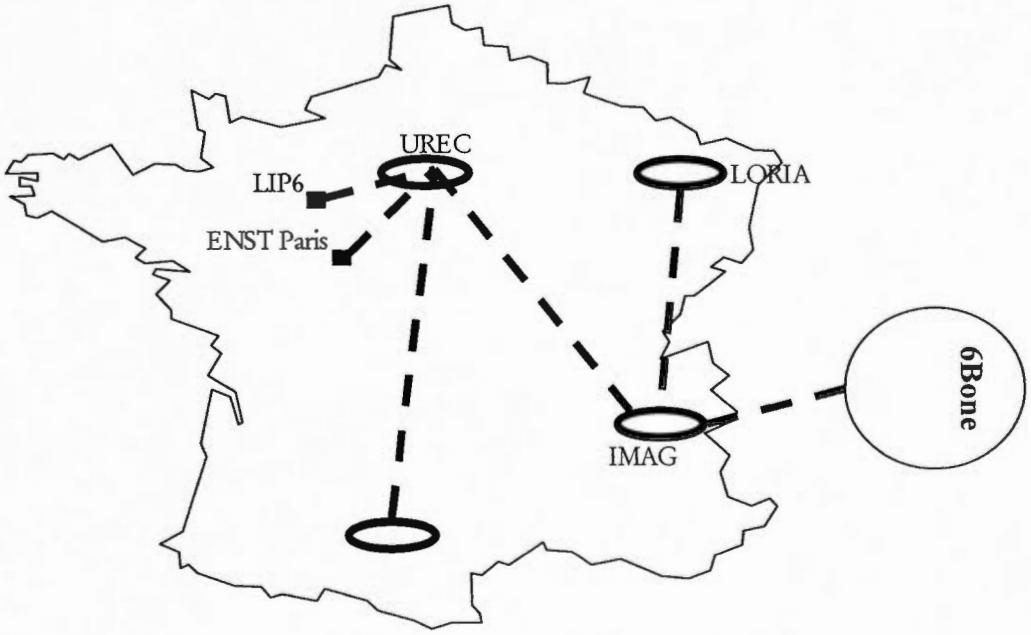

Figure 2.10: Plate-forme AMARRAGE v1 [SPA 2001]

L'architecture d'AMARRAGE définit le nœud actif en conformité avec l'architecture de réseau actif défini par DARPA où les fonctionnalités d'un nœud actif se répartissent en 3 composantes majeures: le système d'exploitation, l'environnement d'exécution et les applications actives. Le système d'exploitation est un noyau Linux (2.4.0) étendu qui intègre à IPv6, le protocole standard d'encapsulation de paquets actifs ANEP.

En plus de cette répartition en 3 composantes, AMARRAGE se structure en trois plans : un plan de contrôle, un plan actif et un plan de transport (voir figure 2.11). Le plan de contrôle regroupe des nœuds spécifiques, de type contrôle et administration, qui prennent en charge le chargement du code et l'administration des nœuds actifs. Le plan actif regroupe les nœuds actifs standards qui offrent une API de base (ANTS) permettant l'accès aux ressources du nœud et la transmission de données. Étant donné que l'API de base est ANTS (légèrement modifié) les possibilités offertes par celle-ci sont essentiellement les mêmes que pour ANTS. Le plan de transport regroupe les nœuds classiques transportant les paquets actifs sans y effectuer de traitement particulier. Cette architecture permet d'offrir un protocole de téléchargement sécurisé, absent dans la distribution originale d'ANTS.

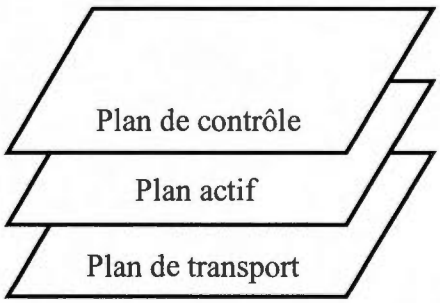

Figure 2.11: Architecture en trois plans [ZEB 2002]

Contrairement au réseau mondial d'expérimentation, Abone, AMARRAGE ne fait pas appel à UDP, et implante des canaux de communication basés sur le protocole IPv6 auquel a été intégré le protocole ANEP. La suppression d'UDP permet de retirer un niveau d'indirection dans les traitements des données et permet aux services actifs de supervision de l'Internet d'opérer directement sur IP. L'environnement d'exécution est une machine virtuelle Java rendue compatible avec IPv6 [DEE 1998] et l'interface de programmation essentiellement celle de ANTS [SPA 2001].

2.5.4 SMART PACKETS

Dans le cadre du programme *DARPA Active Nets*, *Smart Packets* du BBN (*Bolt, Beranek and Newman*)[6] est un projet qui implante une extension aux fonctionnalités de gestion de réseau [DEC 1998]. Il a été conçu pour démontrer que la gestion de réseau est un terrain propice à l'émergence des technologies de réseaux actifs [SCH 2000].

La figure 2.12 présente l'architecture de *Smart Packet*. Un usager ayant écrit un programme génère un paquet *Smart* qui est encapsulé dans une trame ANEP. Celui-ci est ensuite traité par le *daemon* ANEP pour finalement être injecté dans le réseau en mode nœud-à-noeud (*hop-by-hop*). C'est le *daemon* ANEP qui est chargé de la transmission, de la réception et de l'exécution des paquets *Smart*. Le contenu des paquets est écrit en *Sprocket*, un langage de type C[7], et compilé en *Spanner*, un langage de type Assembleur CISC[8]. Le code *Spanner* est ensuite encodé dans un code binaire compact, indépendant des machines, pour finalement être placé dans un paquet *Smart*. Un programme doit être contenu dans un seul paquet *Smart* de 1 Kb et il n'y a pas de persistance d'état à l'intérieur des nœuds [SCH 2000]. Les programmes sont authentifiés avant l'interprétation et l'exécution est limitée en durée [DEC 1999].

---

[6] Ne pas confondre avec le projet *Smart Packets* de l'University du Kansas.

[7] Les pointeurs, les accès aux fichiers et les appels au système ont été enlevés. Des éléments de gestion de réseaux, comme des types intégrés pour les accès MIB et les paquets, ont été ajoutés [SCH 2000].

[8] Le contenu des paquets peut également être directement écrit en *Spanner* [SCH 2000].

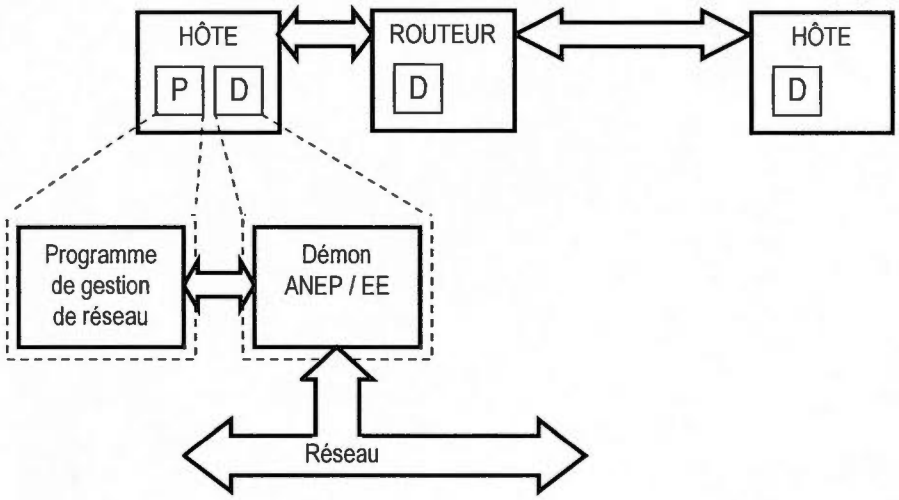

Figure 2.12: Architecture *Smart Packet*

Il est à noter que les trames ANEP sont transportées à l'intérieur des paquets IP ou UDP. *Smart Paquets* utilise l'option *Alerte* des routeurs pour identifier qu'un paquet doit être traité. Si un routeur ne supporte pas les paquets actifs, il ignore cette option et achemine le paquet. Si un routeur supporte les paquets actifs, il examine le message ANEP. S'il supporte les paquets *Smart*, il traite le paquet.

La figure 2.13 présente les formats des paquets *Smart*, ANEP et IP. L'entête des paquets *Smart* a quatre champs : le numéro de la version, le type, le contexte, et le numéro de séquence. Le champ *numéro de version* identifie les mises à jour, alors que le champ *type* identifie le type de paquet : programme, donnée, erreur ou message. Les différents types de paquet transportent respectivement: un programme exécutable, le résultat d'une exécution, un message d'erreur, et finalement, une information qui n'est pas le résultat d'une exécution. Aussi, le champ *contexte* identifie l'origine d'un paquet et le champ *numéro de séquence* contient une valeur qui sert à différencier les messages d'un même contexte [SCH 2000].

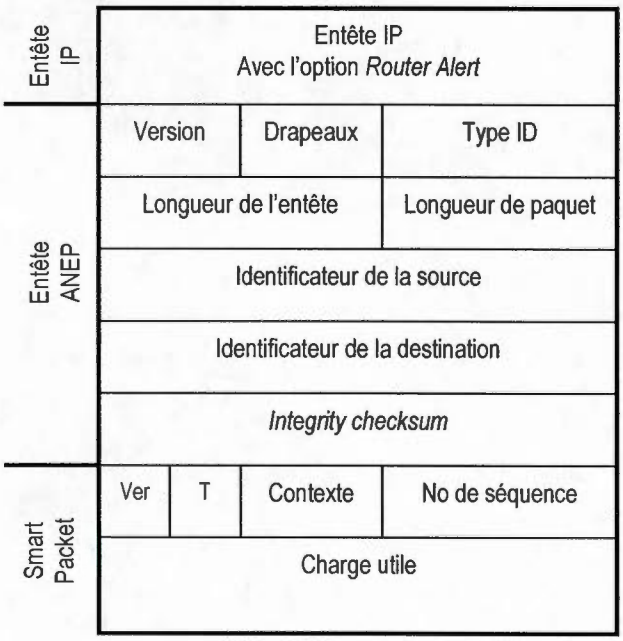

Figure 2.13: Paquet *Smart* avec encapsulation ANEP et IP [SCH 2000]

En terminant, il est important de mentionner que *Smart Packets* est uniquement compatible avec le système d'exploitation FreeBSD.

2.5.5 Étude comparative

Le tableau 2.2 ressort quelques caractéristiques des prototypes de réseaux actifs présentés précédemment. Le choix du prototype, qui servira pour le développement du présent système d'agents mobiles, se basera sur ces caractéristiques.

Prototype	Switchware	ANTS	Amarrage	Smart Packets
Approche	discrète et intégrée	discrète et intégrée	discrète et intégrée	discrète
Langage Environnement d'exécution Paquet	 Caml ou Java Caml et PLAN	 Java Java	 Java Java	 C Sprocket et Spanner
Taille des capsules	illimitée	illimitée	illimitée	1 Kb
Protocole ANEP ou SAPF	---	ANEP	ANEP	ANEP
Sécurité	contrôle d'accès	non	plan de contrôle	non
Documentation	adéquate	adéquate	inexistante	adéquate
Compatibilité	Linux (version Caml), machine virtuelle Java 1.1	machine virtuelle Java 1.2 ou plus récente	Linux (IPv6) et une machine virtuelle Java 1.2 ou plus récente	FreeBSD

Tableau 2.2: Comparaison entre les systèmes de réseau actif

Pour le développement d'un système d'agents mobiles, l'une des qualités qu'il est important (pour ne pas dire essentiel) d'avoir est la portabilité. À cet égard, ANTS ressort comme étant le prototype de réseau actif le plus approprié. Il est le seul à être compatible avec n'importe quelle machine virtuelle Java 1.2 ou plus récente. Le fait qu'ANTS utilise le langage Java, qui est le langage privilégié pour le développement d'un système d'agents mobiles, est également un avantage. D'autres caractéristiques intéressantes d'ANTS sont: l'utilisation d'une approche mixte qui économise la bande passante et qui demeure flexible, la taille des capsules qui est illimitée, la compatibilité avec le protocole ANEP et la documentation adéquate. Le seul inconvénient est le manque de mécanismes de sécurité du système. Par rapport cette dernière, Switchware et Amarrage sont plus intéressants, mais ils sont moins portables et il y a un manque de documentation dans le cas d'Amarrage.

Pour ces raisons, le choix d'ANTS, comme prototype de réseau actif, semble le plus judicieux en ce qui attrait au développement d'un système d'agent mobile. En conséquence, celui-ci sera utilisé pour le développement du présent système.

CHAPITRE III

IBC-GO

Ce chapitre présente le système IBC-Go (*Itinerary Based Computation and Go*). IBC-Go a été initialement conçu par Abdel Obaid et Anna Cavali et a été décrit dans un article interne intitulé: « *IBC-Go: An itinerary-based mobile computing system.* ». Pour répondre aux objectifs du présent travail, la conception d'IBC-Go sera ici reprise et complétée avec l'ajout d'un langage spécifique (AL) pour le développement des agents mobiles et de mécanismes de sécurité pour répondre aux problèmes liés à ce type de système.

Dans ce chapitre, il sera notamment question des caractéristiques suivantes du système IBC-Go: le modèle fonctionnel, la migration, la communication, le protocole, le langage pour la programmation des agents, l'architecture des agences, et finalement, la sécurité.

3.1 Le modèle fonctionnel

D'un point de vue fonctionnel, IBC-Go est divisé en quatre plans: *Itinéraire*, *Agent*, *Communication* et *Sécurité* (voir figure 3.1). Le plan Itinéraire permet de créer et de modifier des itinéraires pour les agents, alors que le plan Agent permet de créer et d'activer des agents mobiles. Comme il a été mentionné précédemment, les itinéraires permettent aux agents de contrôler l'accès aux agences visitées. Le troisième plan, le plan Communication, permet notamment aux agents de communiquer entre eux. Le mécanisme pour supporter cette communication est décrit à la section 3.3. Finalement, le quatrième plan, le plan Sécurité, offre différents mécanismes pour sécuriser le système. La sécurité du système IBC-Go est décrite en détails à la section 3.7.

La figure 3.1 montre le lien entre les quatre plans et les services réseaux. Ces derniers font appel aux services offerts par le système qui supporte IBC-Go; en l'occurrence, un système de réseau actif.

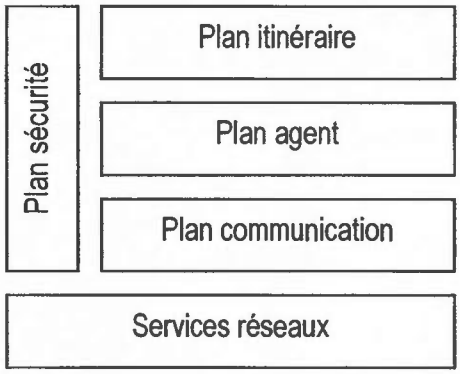

Figure 3.1: Modèle fonctionnel de IBC-Go

3.2 La migration

Comme son nom l'identique, IBC-Go est un système basé sur le concept d'itinéraire. Cela signifie qu'IBC-Go fait appel à des itinéraires pour permettre à ces agents de visiter plusieurs agences avant de retourner à leurs agences d'origine; ce qui contribue à faire diminuer le besoin en communication entre les agences d'origine et les autres agences. Pour offrir plus de flexibilité, IBC-Go offre une mobilité avec itinéraire dynamique aux agents mobiles IBC-Go. Cela permet à ces derniers de modifier leurs itinéraires en cours de route; ce qui contribue à rendre le processus de migration plus flexible.

IBC-Go est un système d'agent à mobilité faible. Cela implique que les états des agents ne sont pas transportés avec eux et fait en sorte que les agents mobiles nécessitent moins de traitement; ce qui, en fin de compte contribue à rendre le système plus efficace.

Enfin, la migration du code des agents est séparée de la migration des agents proprement dits. Il y a deux façons par lesquelles la migration du code peut se faire: en cours

d'exécution et par téléchargement précoce. Avec la première approche, le code est téléchargé et emmagasiné à la demande. Avec la deuxième approche, le code est téléchargé et emmagasiné avant l'arrivée des agents pouvant faire appel à ce code. La première approche offre plus de flexibilité, mais risque de compromettre la performance du système, surtout si les codes à télécharger sont de grandes tailles. La deuxième peut être plus performante, mais rend le système moins flexible. Aucun code ne peut être téléchargé après l'initialisation d'une agence.

3.3 La communication

Pour les communications entre agents, le système IBC-Go utilise l'approche à large diffusion. Cela signifie que les messages sont envoyés en parallèle à toutes les agences. Cette approche a l'avantage d'être facile à implémenter et c'est pourquoi elle a été choisie pour le présent système. L'utilisation de l'approche à large diffusion peut par contre s'avérer moins efficace lorsque le nombre d'agences augmente. Dans le cas où le nombre d'agences deviendrait important, il pourrait être judicieux d'opter pour l'une des autres approches mentionnées dans le chapitre précédent.

3.4 Le protocole

IBC-Go utilise un protocole spécifique pour tous les échanges entre les agences. Ces échanges comprennent la migration des codes et des agents, les messages entre les agents, les messages de gestion, etc. La figure 3.2 présente le format des paquets de ce protocole.

Dans l'entête du paquet, différents champs identifient la composition de celui-ci, comme le champ *Type*, qui détermine si le paquet, représente un agent, contient du code, a une fonction de gestion ou autre. Un autre champ, le champ *Identificateur*, sert à identifier s'il s'agit d'un paquet requête, réponse ou maintenance. Les champs *Origine* et *Destination*,

et possiblement *Créateur*, contiennent des adresses IP correspondant à leur appellation. Enfin, le champ *Autorisations* spécifie les droits associés au paquet (voir la section 3.7).

Chaque code peut contenir plusieurs éléments de code écrit dans des langages différents. Le champ *Type de code* spécifie le langage utilisé et le champ *Adresse de l'interpréteur* indique le site à partir duquel l'interpréteur peut être téléchargé. Si ce dernier n'est pas utilisé (dans le cas d'un interpréteur partagé par tous les sites), l'adresse générique *00000000* est utilisée. L'itinéraire est spécifié avec une séquence d'adresses. Si une des adresses doit être sauvegardée, le bit K (*Keep*) est mis à 1. Des champs *signataire* et *signature* accompagnent les champs associés à l'itinéraire et aux éléments de code. Ces champs permettent d'avoir une signature séparée pour les éléments de code, l'itinéraire et l'agent proprement dit (voir la section 3.7).

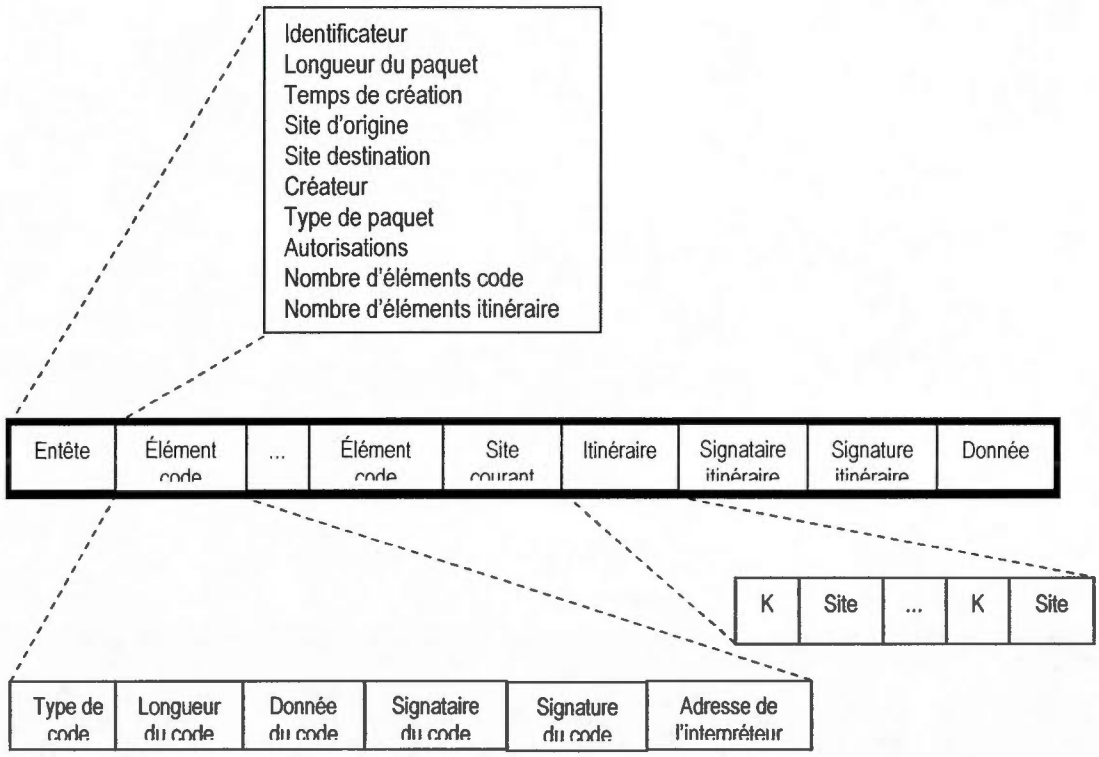

Figure 3.2: Format des paquets IBC-Go

3.5 Le langage pour la programmation des agents

IBC-Go utilise un langage spécifique pour le développement des agents mobiles. À toutes les agences visitées, incluant l'agence d'origine, le code est envoyé à un interpréteur; si les droits relatifs à l'agence visitée sont respectés (voir la section 3.7). Le fait d'ajouter un interpréteur et un langage spécifique pour le développement des agents mobiles permet de ne pas restreindre la programmation des agents au langage ayant servi au développement du système sous-jacent et ne nécessite pas de comprendre les détails du fonctionnement du dit système.

Actuellement, seul le langage AL (*Agent Language*) a été conçu pour le système IBC-Go. Éventuellement, il serait possible de concevoir des interpréteurs pour d'autres langages, comme C, Tcl, etc.. Le langage AL a une syntaxe spécifique et inclut des fonctions destinées à un système d'agents mobiles. Pour montrer de façon plus explicite le langage AL, deux exemples sont ici présentés. Le premier exemple montre le code d'un programme qui fait une série d'additions et de multiplications et qui affiche trois nombres: 3, -1 et 2187. Voici le code du premier l'exemple :

```
// Output is:
//        3
//        -1
//        2187
program math =
        constant one : Integer := 1;
        constant two : Integer := 2;
        var y, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14 : Integer;
begin // math
        x1  := one;
        x2  := two;
        x3  := one;
        x4  := two;
        x5  := one;
        x6  := two;
        x7  := one;
        x8  := two;
        x9  := one;
        x10 := two;
        x11 := one;
        x12 := two;
        x13 := one;
        x14 := two;
```

```
        y := (x1+x2)*((x3+x4)*((x5+x6)*((x7+x8)*((x9+x10)*((x11+x12)*(x13+x14))))));
        put(one + two);
        newLine;
        put(-1);
        newLine;
        put(y);
        newLine;
end math.
```

Comme on peut le voir avec l'exemple précédent un programme en AL commence par l'instruction *program* suivie du nom du programme et de l'opérateur '=' et se termine par l'instruction *end* suivie du nom du programme et d'un point. Un programme en AL est composé de deux sections: la section déclaration et la section logique. La section déclaration se trouve avant l'instruction *begin*. C'est là que sont déclarée les constantes, les variables et les procédures. Dans l'exemple qui précède, les constantes entières (*integer*) *one* et *two* sont déclarées et initialisées à 1 et 2 respectivement. Les variables entières *y*, *x1*, *x2*, *x3*, *x4*, *x5*, *x6*, *x7*, *x8*, *x9*, *x10*, *x11*, *x12*, *x13* et *x14* y sont également déclarées.

La logique du programme se trouve à la section logique, qui commence après l'instruction *begin*. Dans l'exemple qui précède, les variables *x1* à *x14* sont initialisées à *one* (1) pour les variables *x* impaires et à *two* pour les variables *x* paires. La variable *y* pour sa part est initialisée avec le résultat (2187) d'une expression comprenant les variables *x1* à *x14*. Le programme affiche ensuite à l'écran la somme (3) de la constante *one* et *two*, le nombre –1 et la valeur de la variable *y* (2187). Chaque nombre apparaît sur une nouvelle ligne grâce aux instructions *newLine*.

Enfin, comme il est également possible de le constater, les lignes de commentaire commencent par les symbôles '//' et les lignes d'instruction se terminent par le symbôle ';'.

Le deuxième exemple présente un programme en AL qui utilise des fonctions associées aux agents mobiles. Le programme commence par la déclaration de deux procédures: *Code1* et *Code2*. La première procédure affiche à l'écran le message « procedure Code1 », alors que la deuxième procédure affiche à l'écran le message « procedure Code2 »

et indique à l'agent de poursuivre son itinéraire. Dans la logique du programme, les codes *Code1* et *Code2* sont créés avec les codes inclus dans les procédures *Code1* et *Code2* respectivement. L'agent *NewAgent* est ensuite créé avec les codes *Code1* et *Code2* et le message « message1 » pour la charge utile. L'itinéraire, incluant les agences 18.31.12.3, 18.31.12.4 et 18.31.12.1, est par la suite attribué à l'agent *NewAgent*.

Suite à cela, une copie de l'agent *NewAgent*, *18.31.12.1.NewAgent2*, est créée avec la fonction *copyAgent*. Deux autres copies de l'agent *NewAgent*, *18.31.12.1.NewAgent3* et *18.31.12.1.NewAgent4*, sont ensuite créées avec la fonction *spawnAgent*. Finalement, l'agent *18.31.12.1.NewAgent3* est activé avec la fonction *activateAgent*. Voici le code du deuxième exemple:

```
// Creation et la copie de codes et agents
program createCopy =
        procedure Code1 =
        begin
                put("procedure Code1");
                newLine;
        end Code1;
        procedure Code2 =
        begin
                put("procedure Code2");
                newLine;
                followIt();
        end Code2;
begin // createCopy
        createCode("Code1", Code1);
        createCode("Code2", Code2);
        createAgent("NewAgent", {"Code1", "Code2"}, "message1");
        It(NewAgent) := ({"0", "18.31.12.3", "0", "18.31.12.4", "0", "18.31.12.1"});
        copyAgent("18.31.12.1. NewAgent2", "NewAgent");
        spawnAgent({"18.31.12.1. NewAgent3","18.31.12.1. NewAgent4"},
                "18.31.12.1. NewAgent2");
        activateAgent("18.31.12.1. NewAgent3", false);
end createCopy.
```

Les fonctions qui se trouvent dans le code précédent appellent des fonctions de l'environnement d'exécution de l'agence. Cela simplifie le code des agents mobiles et, comme il a été dit précédemment, n'oblige pas les programmeurs à comprendre la complexité du système sous-jacent. D'autres fonctions, comme des fonctions relatives aux fichiers, aux

bases de données et aux communications entre agents, sont fournies avec le langage AL. Le langage AL et les fonctions spécifiques aux agents mobiles sont présentées en détail en annexe.

3.6 L'architecture des agences

La figure 3.3 présente l'architecture des agences IBC-Go. Comme on peut le voir avec cette figure, les agences IBC-Go comprennent différents modules; comme le gestionnaire des communications qui est responsable des échanges entre les agences. Toutes les transmissions et réceptions d'agents, de codes et de messages sont transigées par le gestionnaire des communications.

À l'arrivé d'un agent IBC-Go, le gestionnaire des communications transmettra ce dernier au module vérificateur pour que celui-ci vérifie la signature de l'agent et de l'itinéraire, lise le champ d'autorisations de l'agent et accorde les privilèges en conséquence (voir la section 3.7). Le module vérificateur se charge également de vérifier si les codes à exécuter sont emmagasinés dans le répertoire d'agents. Ce dernier sert en effet à emmagasiner les agents et les codes reçus. Si un des codes à exécuter ne se trouve pas dans le répertoire d'agents, le module vérificateur se charge de faire une demande de code à l'agence qui a envoyé l'agent, via le gestionnaire des communications. À la réception du code, le module vérificateur vérifie la signature du code (voir la section 3.7) et emmagasine ce dernier dans le répertoire d'agents. Lorsque tous les codes à exécuter se trouvent dans le répertoire d'agents, le module chargeur chargera les codes dans l'environnement d'exécution pour que ceux-ci soient exécutés. Tout au long de l'exécution, l'environnement d'exécution échangera avec le module vérificateur pour vérifier que l'agent est autorisé à exécuter chacune des commandes.

Une base de donnée *Trace* garde la trace des agents, des messages et des itinéraires, de même que les informations relatives aux utilisateurs d'IBC-Go. Aussi, un module ACMP

(*Agent Control Message Protocol*) est utilisé pour gérer les messages de gestion qui permettent de contrôler les agents et une interface graphique permet aux utilisateurs de créer et de gérer les codes et agents IBC-Go.

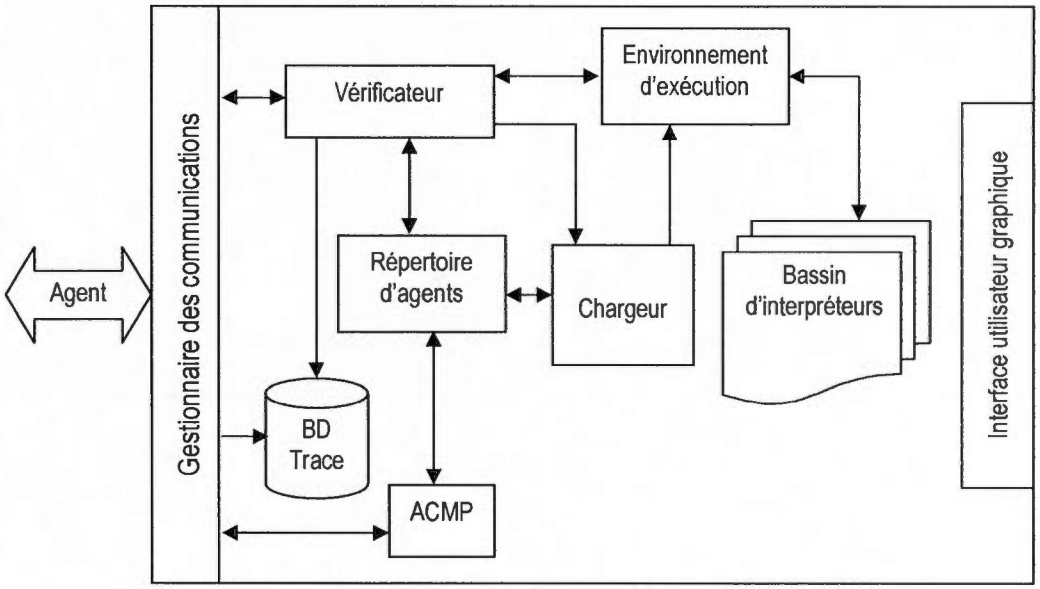

Figure 3.3: Modèle architectural des agences IBC-Go

3.7 La sécurité

Au chapitre précédent, trois catégories de menaces reliées aux technologies d'agents mobiles ont été exposées. Les catégories de menaces sont:

- une attaque d'un agent vers une agence,
- une attaque d'une agence vers un agent,
- une attaque d'un agent vers un autre agent.

Dans la première catégorie d'attaque, il y avait la menace d'une mauvaise utilisation des ressources des agences et la possibilité qu'un agent change son identité. Pour contrer la

menace d'une mauvaise utilisation des ressources, ces dernières doivent être protégées. Avec le système IBC-Go, les accès aux ressources des agences sont contrôlés par les agences; via le module vérificateur. Pour permettre aux agents mobiles d'avoir accès aux ressources des agences, un champ d'autorisation est inclus avec le protocole IBC-Go (voir la figure 3.4). Le tableau 3.1 présente la signification des différents bits étant inclus dans le champ d'autorisation.

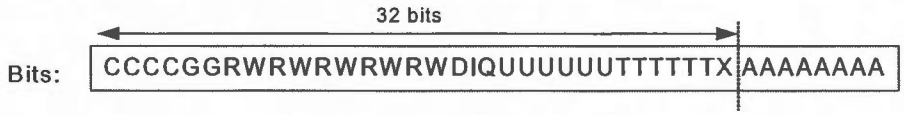

Figure 3.4: Champ d'autorisation

CCCC	Niveau de confiance : 0000-1111
GG	Accéder à tout (11), à rien (00), accès partiel (10)
WRWRWRWR	E/S fichiers, *sockets*, processus, agents
D	Accès aux *drivers*
I	Affichage sur l'interface graphique
Q	QoS permis
UUUUUU	Limite CPU
TTTTTT	TTL limite
X	Réservé
AAAAAAAA	Bits d'identification (signature digitale)

Tableau 3.1: Champ d'autorisation

Avec les bits G, il sera possible de permettre aux agents d'utiliser toutes les opérations du système (jumpTo, meetAgent, ...), aucune ou certaines d'entre elles. Il y aura également un niveau de confiance (les bits C) associé à l'agent. En fonction de celui-ci, il sera possible de traiter l'agent en conséquence. Également, d'autres bits permettront d'octroyer les accès en fonction des agents (les autres bits, à l'exception des bits A).

Finalement, le champ Créateur et le champ Autorisation devront être signés[9] (les bits A) par une personne de confiance pour assurer la validité du champ. Cette signature permettra de plus d'éviter la possibilité qu'un agent change son identité.

Relativement aux catégories de menaces reliées aux technologies d'agents mobiles, les deux autres catégories se rapportaient essentiellement à la possibilité de modifier le contenu d'un agent. Pour protéger l'intégrité des agents, la portion de ces derniers, qui est propre au créateur et qui est statique, devra être signée. En l'occurrence, cette portion inclut le ou les codes et l'itinéraire. Les noms des signataires et les signatures pour les codes et l'itinéraire sont transmis avec le protocole IBC-Go (voir la figure 3.2). En fonction des agences et des créateurs, le ou les codes et l'itinéraire pourront être signés par le créateur ou devront être signés par une personne de confiance. Cela dépendra de la confiance des agences envers le créateur.

En plus des menaces propres aux systèmes d'agents mobiles, d'autre menaces demeurent et sont propres au système IBC-Go. Ces dernières touchent plus particulièrement à l'accès au système et aux échanges entre les utilisateurs et le système. En ce qui concerne l'accès au système, celui-ci doit être contrôlé pour éviter que des individus non autorisés accèdent au système et en perturbe le bon fonctionnement. Pour cela, un nom d'utilisateur et un mot-de-passe devront être fournis pour accéder aux fonctionnalités du système IBC-Go. Ces derniers seront emmagasinés aux différentes agences du système et les mots de passe y sont encryptés. En ce qui concerne les échanges entre les utilisateurs et le système, il est en effet possible que les données soient interceptées ou modifiées. Pour éviter cela, ces échanges devront être encryptés.

---

[9] Les signatures utiliseront l'algorithme DSA (*Digital Signature Algorithm*).

CHAPITRE IV

MISE EN ŒUVRE

Le présent chapitre décrit l'implémentation du système IBC-Go. On commence par le choix de l'environnement et du langage de développement pour ensuite présenter l'architecture logicielle du système IBC-Go. Suite à cela, l'implémentation relative aux interfaces graphiques, à l'interpréteur du langage AL, au gestionnaire des communications IBC-Go (incluant le protocole et le paquet IBC-Go), à l'environnement d'exécution, aux communications, et finalement, à la sécurité du système IBC-Go est présentée.

Après la présentation de l'implémentation des composantes précédentes, une technique particulière appelée *assistant routeur*, utilisée pour intégrer des équipements réseaux, comme les routeurs actuellement utilisés, sera présentée. Ce dernière présentation s'accompagnera de celle relative à l'implémentation de la dite technique.

4.1 L'environnement et langage

L'environnement dans lequel est exécuté le système IBC-Go est une machine virtuelle Java 1.4.2. En l'occurrence, le système IBC-Go a été développé en utilisant le langage Java. L'environnement et le langage Java ont été choisis compte tenu de leur portabilité et de leur omniprésence dans le domaine des agents mobiles.

4.2 L'architecture logicielle

Respectant une approche orientée objet, le système IBC-Go est composé d'un ensemble de classes. La figure 4.1 présente, les principales classes du système et les liens qui les unissent avec l'interface graphique. Comme on le voir à la figure 4.1, l'ensemble des liens qui unissent les classes sont de type instanciation; à l'exception du lien avec l'interface graphique qui établit une communication via le protocole HTTP et une connexion de type *socket*.

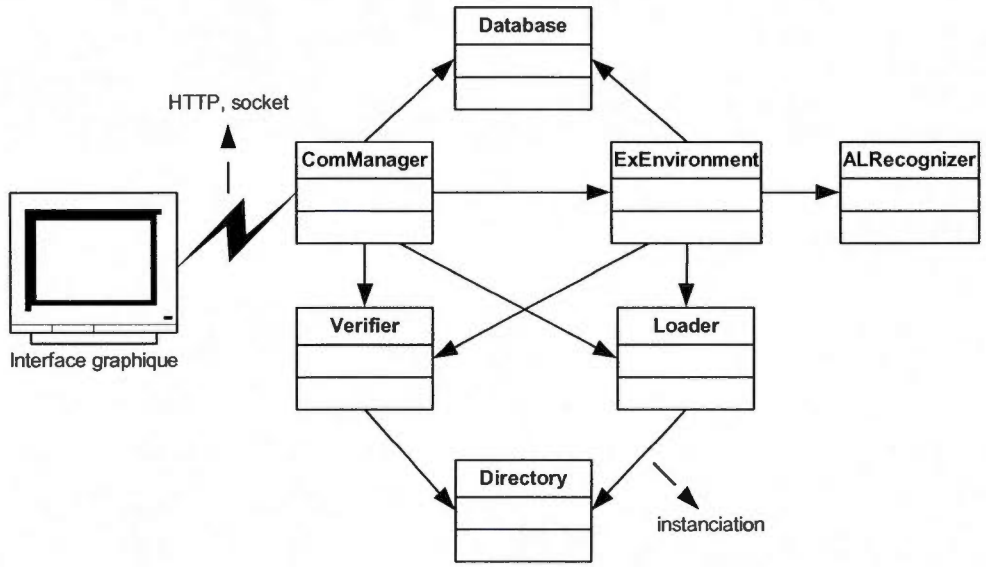

Figure 4.1: Architecture logicielle du système IBC-Go

D'autres classes, comme *IBCProtcol* et *IBCPacket*, ne figurent pas dans la figure 4.1 et seront mentionnées plus loin. Ces classes ne figurent pas dans la figure 4.1 pour conserver la simplicité de celle-ci.

4.3 Les interfaces utilisateurs

Pour créer et activer des agents mobiles, un utilisateur doit tout d'abord interagir avec une agence IBC-Go. Il y a deux façons par lesquelles un utilisateur peut interagir avec une agence IBC-Go: avec des fichiers textes spécifiques ou avec l'interface graphique. Ces deux façons d'interagir sont décrites ci-dessous.

4.3.1 Les fichiers textes

L'interaction avec des fichiers textes se fait via les fichiers de configuration qui sont lus au démarrage de l'agence. Plus précisément, un fichier de type XML comprenant le nom de l'agent, la description du code (avec une référence aux fichiers contenant le code et la signature[10]), le ou les messages inclus dans la charge utile et l'itinéraire (avec une référence au fichier signature) peut être lu lors du démarrage de l'agence. Voici un exemple de ce fichier:

```
<?xml version="1.0" encoding="utf-8"?>

<ibcgo:agent   xmlns:ibcgo="http://www.uqam.ca/ibcgo"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation=" http://www.uqam.ca/ibcgo
                                    http://66.131.176.10:8080/ibcgo.xsd"
               identifier="18.31.12.1.Test"
               >
               <code
                              id="test"
                              name="test.al"
                              signer="GSerge"
                              signature="test.sig"
                              interpretor="18.31.12.1:3332"
                              />
               <payload><message></message></payload>
               <itinerary signer="GSerge" signature="testIti.sig">
                              <k>0</k><site>18.31.12.2</site>
                              <k>0</k><site>18.31.12.3</site>
```

[10] Un petit utilitaire, implémenté avec la classe *Hancock*, a été conçu pour permettre aux utilisateurs de créer les fichiers contenant les signatures.

```
                    <k>0</k><site>18.31.12.4</site>
                    <k>0</k><site>18.31.12.1</site>
            </itinerary>
</ibcgo:agent>
```

Ce mode d'utilisation a été utilisé en cours de développement pour faire des tests et peut être utilisé pour créer et activer des agents au démarrage. Pour interagir avec l'agence après le démarrage de celle-ci, l'interface graphique doit être utilisée.

4.3.2 L'interface graphique

L'interface graphique du système IBC-Go est la principale façon d'interagir avec les agences IBC-Go. Avec celle-ci, il est possible d'effectuer l'ensemble des opérations relatives aux agences IBC-Go, dont la création et l'activation des agents mobiles. La figure 4.2 montre l'interface graphique du système IBC-Go. Elle est composée de 5 sections:

- un menu Commande qui permet de naviguer à travers les différentes fenêtres de travail;
- un menu Agent qui permet de voir la liste des agents qui résident à l'agence;
- un menu Code qui permet de voir la liste des codes qui résident à l'agence;
- une fenêtre de travail, qui permet aux usagers d'ajouter ou d'enlever des codes et agents, d'activer un agent, etc;
- une console qui affiche les messages transmis par un agent.

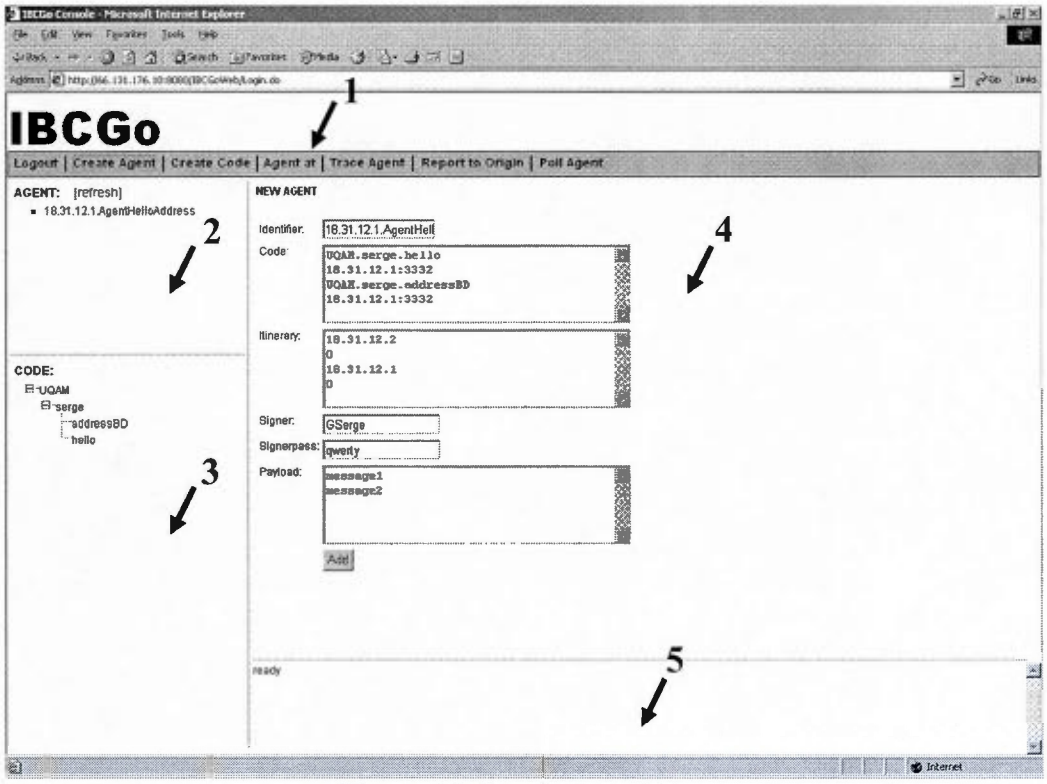

Figure 4.2: Interface graphique

L'interface graphique est une application Web qui, d'un point de vue structurel, est séparée du reste du système IBC-Go. Cette séparation fait en sorte que l'interface graphique et le serveur Web associé peuvent être installés sur une machine autre que celle utilisée pour héberger le reste du système IBC-Go. Le fait également que l'interface graphique est une application Web implique qu'il n'y a pas d'installation à faire sur les postes clients. N'importe quel poste muni d'un navigateur Web peut accéder à l'interface graphique du système IBC-Go.

Pour implémenter l'interface graphique, différentes technologies Web ont été utilisées notamment Struts et une applet. L'appellation *Struts* fait référence à un support pour l'architecture. C'est une plate-forme qui permet de construire une application Web en respectant le patron de conception MVC (*Model View Controller*). Le patron MVC est

largement reconnu et figure parmi les modèles de conception les mieux développés et les plus matures. Il permet de faire une séparation modulaire des fonctions ce qui permet de faciliter l'entretien de l'application [HUS 2003]. Une applet a également été construite pour recevoir les messages en provenance d'une agence IBC-Go. Cette dernière établit une connexion de type *socket* avec l'agence IBC-Go.

Outre l'utilisation de Struts et d'une applet, l'interface graphique a été implémentée avec d'autres technologies Web comme les *JavaBeans*, les *custom tags*, JSP, JavaScript, CSS et HTML.

4.4 L'interpréteur pour le langage AL

La programmation d'un interpréteur est une tâche relativement longue et complexe. Cela nécessite notamment de programmer un analyseur lexical et syntaxique. Heureusement, il existe des outils pour simplifier la tâche reliée à la programmation d'interpréteur. L'un des outils les plus populaires pour construire un interpréteur est ANTLR (*Another Tool for Language Recognition*) [ANT 2004].

ANTLR a été utilisé pour implémenter l'interpréteur pour le langage AL. ANTLR est un outil qui permet de construire des interpréteurs, compilateurs et traducteurs en générant des analyseurs lexical et syntaxique en Java (la classe ALRecognizer de la figure 4.1), C++ ou C#.

Avec ANTLR, les analyseurs lexical et syntaxique sont définis avec un langage spécifique. Pour l'analyseur lexical, des composantes, appelés *jetons* sont définies et associées à un ou des caractères spécifiques. Il est par exemple possible de définir le jeton *DIGIT* et de l'associer aux caractères de 0 à 9 de la façon suivante: DIGIT : '0'..'9';. Les jetons ainsi définis sont par la suite utilisés par l'analyseur syntaxique.

Pour la définition de l'analyseur syntaxique, des instructions sont définies et associées à un code source correspondant; en l'occurrence Java. Il est par exemple possible de définir l'instruction *newLine* et de l'associer au code Java qui exécute un saut de ligne de la façon suivante: "newLine" (LPAREN RPAREN)? SEMI { System.out.println(); }.

4.5 Le gestionnaire des communications, le protocole et le paquet IBC-Go

ANTS a été utilisé pour le développement du gestionnaire des communications, du protocole et du paquet IBC-Go, incluant l'agent mobile IBC-Go. Le gestionnaire des communications (voir la figure 3.3) est implémenté avec la classe *ComManager* (voir la figure 4.1) qui est en fait une implémentation de la classe ANTS *Application* (voir la section 2.6.2). En ce qui a trait au protocole et à paquet IBC-Go, ces derniers sont respectivement une extension des classes ANTS *Protocol* et *DataCapsule*.

La classe *ComManager* est la première classe IBC-Go qui est exécutée lorsqu'une agence est démarrée. Avec l'exécution de cette classe, le protocole IBC-Go (*IBCProtocol*) est enregistré à l'agence. Ce dernier identifie simplement le paquet IBC-Go (*IBCPacket*) qui inclut l'agent mobile IBC-Go comme étant inclus dans le protocole IBC-Go. L'enregistrement du protocole IBC-Go incluant le paquet IBC-Go, fait en sorte que les paquets IBC-Go sont reconnus et acceptés par une agence IBC-Go.

La classe *ComManager* est utilisée pour tous les échanges entre les agences. Pour l'envoi de paquet IBC-Go, une méthode particulière (*sendPacket*) de la classe *ComManager* est utilisée. Cette dernière se charge de créer le paquet à envoyer et d'initialiser les champs de celui-ci en fonction des paramètres qui lui sont transmis, pour ensuite indiquer à ANTS d'envoyer le paquet. Ce dernier se chargera de sérialiser le paquet et de l'envoyer.

Lorsqu'un paquet arrive à une agence, il est référé à la classe *ComManager* via ANTS. Lorsque la classe *ComManager* reçoit un paquet, la vérification de son intégrité est

effectuée. Pour cela, la validité des signatures relatives au champ d'autorisation et à l'itinéraire est vérifiée en collaboration avec la classe *Verifier*. Dans l'éventualité où l'intégrité du paquet est respectée et que celui-ci représente un agent IBC-Go, la classe *Verifier* vérifiera si les codes qui doivent être exécutés, se trouvent dans le répertoire d'agents de l'agence. Si des codes sont manquants, la classe *Verifier* fera une demande de code, pour les codes manquants, à l'agence qui a envoyé l'agent via la classe *ComManager*.

4.6 L'environnement d'exécution

Lorsqu'un agent arrive à une agence et que tous les codes à exécuter se trouvent dans le répertoire d'agents, le module chargeur, implémenté avec la classe *Loader* (voir la figure 4.1) charge les codes à exécuter dans l'environnement d'exécution pour que ceux-ci soient exécutés.

L'environnement d'exécution est implémenté avec la classe *ExEnvironment* (voir la figure 4.1). Lorsque cette classe reçoit un code à exécuter, celui-ci est interprété via l'interpréteur approprié; en l'occurrence, la classe *ALRecognizer* (voir la figure 4.1). Comme il a été présenté au chapitre 3, les codes AL comprennent des fonctions propres aux agents mobiles. Ces fonctions font référence à des méthodes de la classe *ExEnvironment* qui contiennent du code Java réalisant l'exécution de ces fonctions. Par exemple, la fonction *followIt()* du langage AL fait référence à la méthode *follow* de la classe *ExEnvironment* pour l'exécution de la fonction.

4.7 Les communications

Comme il a été mentionné au chapitre précédent, IBC-Go utilise une approche à large diffusion pour la communication entre agents. Avec l'Internet, il est nécessaire de définir un groupe d'agences pour savoir à quelles agences doivent être envoyés les messages. Ce groupe

d'agences est défini dans un fichier de type XML qui est lu lors de l'initialisation de l'agence. Voici un exemple de ce fichier:

```
<?xml version="1.0" encoding="utf-8"?>

<ibcgo:init xmlns:ibcgo="http://www.uqam.ca/ibcgoInit"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.uqam.ca/ibcgoInit
                       http://66.131.176.10:8080/ibcgoInit.xsd"
   >

      <owner>18.31.12.1:3332</owner>
      <credentials>11111111111111111111111111111110</credentials>
      <signer>GSerge</signer>
      <signature>credentials.sig</signature>

      <group>
            <site>18.31.12.1</site>
            <site>18.31.12.2</site>
            <site>18.31.12.3</site>
            <site>18.31.12.4</site>
      </group>
</ibcgo:init>
```

Comme on peut le voir, ce fichier définit notamment un groupe incluant les sites (agences): 18.31.12.1, 18.31.12.2, 18.31.12.3 et 18.31.12.4. Il définit également d'autres composantes comme *owner* et *credentials*, qui seront utilisées par les mécanismes de sécurité présentés à la section 4.8.

Tous les messages reçus par une agence sont emmagasinés dans des mémoires tampons. Ces mémoires, gérées par la classe *ComManager*, contiennent le nom des expéditeurs, le nom des destinataires et le contenu des messages. Les agents doivent utiliser une fonction spécifique, *{from Agent}?Message*, pour pouvoir saisir les messages qui leur sont destinés.

4.8 Sécurité

Comme il a été présenté au chapitre précédent, IBC-Go comprend différents mécanismes de sécurité pour contrer les menaces reliées aux technologies d'agents mobiles. L'un de ces mécanismes est l'inclusion d'un champ d'autorisation dans le protocole IBC-Go. Celui-ci est véhiculé avec les paquets IBC-Go, implémenté avec la classe *IBCPacket* (voir la figure 4.1), et est signé avec le champ Créateur (également véhiculé avec les paquets IBC-Go). Lorsqu'un agent est créé, à l'initialisation ou avec l'interface graphique (voir la section 4.3), l'agence va prendre les valeurs relatives au créateur et au champ d'autorisation dans un fichier de type XML spécifique à l'agence[11]. Un exemple de ce fichier est présenté à la section 4.7. Comme on peut le voir avec cet exemple, les valeurs relatives au créateur et au champ d'autorisation sont contenues sous les *tags owner* et *credentials*. En plus des valeurs précédentes, le nom du signataire et du fichier, contenant la signature des champs Créateur et Autorisation, sont également fournis. Le fichier contenant la signature est créé à l'aide d'une commande qui prend la forme suivante:

```
java IBCGo.SignFactory scredits
KEYSTORE STOREPASS SIGNER SIGNERPASS IBCGOXML SIGNFILE
```

Cette dernière utilise la classe *SignFactory* et fournit entre autres les informations relatives au fichier *keystore*, qui contient les clés. La classe *SignFactory* fait appel à la librairie JCE 1.2.2 (*Java Cryptography Extension*), en plus des librairies fournies avec le SDK 1.4.2.

Les valeurs, incluses dans le fichier de type XML, sont utilisées uniquement lorsqu'un agent est créé par un utilisateur. Lorsqu'un agent est créé avec la fonction

[11] Actuellement, les autorisations relatives aux agents IBC-Go sont les mêmes pour tous les agents créés à une même une agence. Cela a pour avantage que les utilisateurs d'une agence peuvent être créés par un administrateur de cette agence. Il serait possible d'imaginer un système où les autorisations sont reliées davantage aux utilisateurs qu'aux agences. Dans un tel système, il serait nécessaire que les utilisateurs soient créés par une autorité indépendante de l'agence, pour garder l'indépendance avec celle-ci.

createAgent, une valeur par défaut est fournie relativement aux champs Créateur et Autorisation et la signature est générée automatiquement par le système IBC-Go. Cela est nécessaire, compte tenu qu'il faut le mot de passe du signataire pour faire une signature électronique et que ce mot de passe doit demeurer confidentiel. Bien entendu, cela fait en sorte que les agents créés avec la fonction *createAgent* sont potentiellement moins sécuritaires que les agents créés par les utilisateurs.

En plus du champ d'autorisation, IBC-Go inclut également la signature des codes et de l'itinéraire. De même que pour le signataire et la signature reliés aux champs Créateur et Autorisation, les signataires et les signatures associés aux codes et à l'itinéraire sont inclus dans le protocole IBC-Go; et en l'occurrence dans les paquets IBC-Go. Une commande, similaire à la commande utilisée pour signer le nom du créateur et la valeur de l'autorisation, peut être utilisée pour signer les codes et l'itinéraire. Lorsque l'interface graphique est utilisée pour créer un agent, les informations fournies avec cette commande sont fournies via l'interface. La commande, de même que l'interface graphique, utilise la classe *SignFactory* pour faire les signatures.

Enfin, des mécanismes de sécurité sont inclus avec le système IBC-Go pour sécuriser l'accès au système et les échanges entre les utilisateurs et le système. Pour sécuriser l'accès au système, un nom d'usager et un mot de passe doivent être fournis à une agence IBC-Go via l'interface graphique du système. Cela permet aux utilisateurs d'accéder à une agence IBC-Go. Les noms des utilisateurs et les mots de passe sont contenus dans une base de données et un utilitaire (implémenté avec la classe *User*) permet d'entrer un utilisateur dans la base de donnée.

Pour sécuriser les échanges entre les utilisateurs et le système, le protocole SSL est utilisé quand la communication est établie via le protocole HTTP et le protocole Diffie-Hellman est utilisé dans le cas où la communication utilise une connexion de type *socket* (applet). Les connexions de type *socket* s'établissent entre la classe *ClientConnexion* (applet) et la classe *ComConnexion* (agence IBC-Go).

4.9 L'intégration des routeurs traditionnels

Comme il a été mentionné au chapitre 2, IBC-Go utilise un modèle de réseau actif, ANTS pour pouvoir exécuter des agents mobiles sur n'importe quel nœud actif (poste de travail, serveur, routeur, etc.). Pour qu'un nœud soit actif, il faut en l'occurrence que le système IBC-Go y soit installé. Bien que cela ne pose pas vraiment de problème pour les postes de travail et les serveurs, il en va autrement des équipements réseaux comme les routeurs.

Pour faire en sorte qu'un routeur traditionnel puisse devenir un nœud actif (c'est-à-dire qu'il soit une agence IBC-Go), l'approche *assistant routeur* a été retenue [LAR 2001]. Avec cette approche, une machine assistante (Linux par exemple) est reliée à un des ports des routeurs. Lorsqu'un paquet incluant l'option *router alert* arrive à un routeur; celui-ci est redirigé vers la machine assistante pour être traité (*Slow path*). Les paquets n'incluant pas l'option *router alert* sont pour leur part acheminés sans redirection (*Fast path*) [KAT 1997]. L'approche, *assistant routeur* offre plusieurs avantages comme le fait de pouvoir utiliser les routeurs actuels et celui de séparer le travail de routage actif du travail de routage passif [LAR 2002]. Pour supporter cette approche, les routeurs actuels doivent être configurés en conséquence. Le code suivant montre un exemple de configuration pour un routeur Cisco [DOO 2003]:

```
enable
configure terminal
ip access-list extended mylist
permit ip any any option router-alert
end
show access-lists
route-map mymap
match ip address mylist
set ip next hop 192.168.1.100
```

ANTS a été modifié pour pouvoir envoyer des paquets incluant l'option *router alert*. Plus particulièrement, la classe *UDPChannel* de ANTS a été modifiée pour pouvoir interagir avec une classe en C++ via JNI (*Java Native Interface*). Le code de la classe C++ et de la classe *UDPChannel* est présenté en annexe.

CHAPITRE V

TESTS ET RÉSULTATS

Ce chapitre présente les tests qui ont été faits pour démontrer le bon fonctionnement du système IBC-Go et des scénarios de mise en application du système. La présentation des scénarios a pour but de démontrer les avantages amenés par le système IBC-Go et plus particulièrement de démontrer le fait qu'il amène une diminution des besoins en communication entre le nœud d'origine et les autres nœuds (là où les connexions sont généralement plus lentes et instables).

Pour appuyer les démonstrations des avantages amenés par le système IBC-Go, des résultats numériqués accompagneront chacun des scénarios présentés. Ces dernières permettront de chiffrer les gains amenés par le système IBC-Go.

5.1 Les tests

Une série de tests ayant pour but de tester exclusivement l'interpréteur, a été faite. Ces tests avaient pour but de tester les différents aspects de l'interpréteur comme les procédures, les boucles, etc. Un de ces tests est présenté ci-dessous.

```
program sample8 =
        constant arraySize : Integer := 10;
        type Table = array[1..arraySize] of Integer;
        var a : Table;
        var i : Integer;
        var x : Integer;
        var yes : Boolean;
        procedure search(searchValue : Integer; var found : Boolean; var index : Integer) =
                var limit : Integer := arraySize;
        begin // search
                found := FALSE;
                index := 1;
                while index <= arraySize and not found loop
```

```
                    if a[index] = searchValue then
                        found := TRUE;
                        yes := TRUE;
                        i := index;
                    else
                        index := index + 1;
                    end if;
                end loop;
        end search;
begin // sample8
        i := 1;
        while i <= arraySize loop // input table
                get(a[i]);
                i := i + 1;
        end loop;
        loop
                get(x);
                exit when x = 0;
                search(x, yes, i);
                put(x);
                put(": ");
                if yes then
                        put(i);
                end if;
                yes := FALSE;
                newLine;
        end loop;
end sample8.
```

Comme on peut le voir, cet exemple inclut plusieurs composantes de l'interpréteur, comme une procédure, des boucles et des conditions. Lorsqu'il est exécuté, il saisit 10 nombres entrés par un utilisateur suivis d'un nombre qui correspond au nombre qui sera recherché parmi les 10 premiers nombres entrés. Si le nombre recherché est présent parmi les 10 premiers nombres entrés, le programme affiche le nombre recherché suivi de la position du nombre dans la liste des 10 premiers nombres. Sinon, il affiche le nombre tout simplement. Le nombre 0 peut être entré pour quitter le programme. La figure 5.1 présente le résultat de l'exécution du programme.

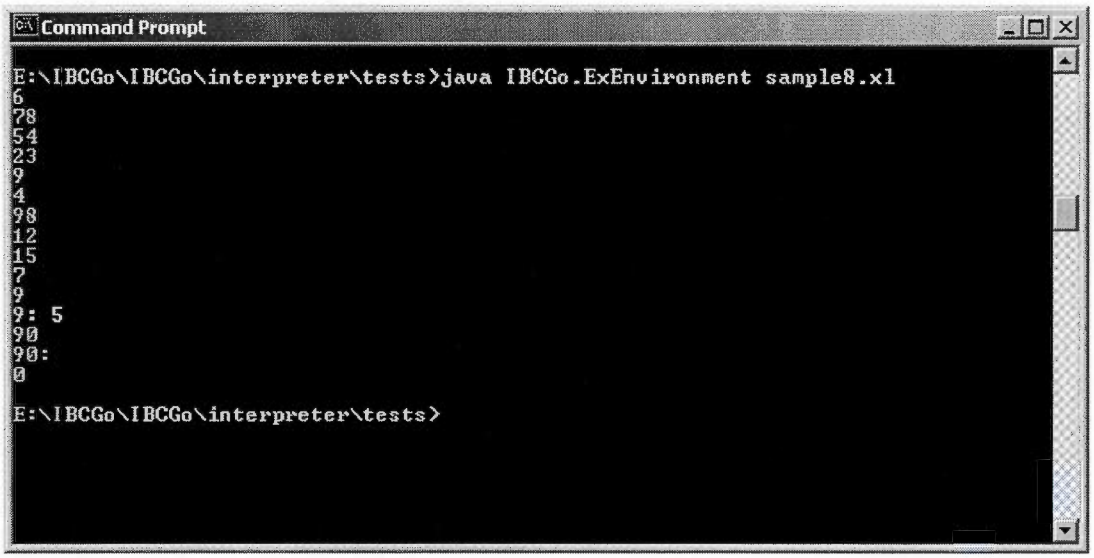

Figure 5.1: Résultat d'un test pour l'interpréteur

Comme on peut le voir à la figure 5.1, les nombres 6, 78, 54, 23, 9, 4, 98, 12, 15 et 7 sont les dix premiers nombres entrés et le nombre 9 est entré comme premier nombre à rechercher. Suite à cela, l'interpréteur affiche '9: 5', ce qui signifie que le nombre 9 est en cinquième position dans la liste; ce qui est exact. Le nombre 90 est ensuite entré comme nombre à rechercher et l'interpréteur affiche par la suite '90:', ce qui signifie que le nombre 90 n'est pas dans la liste; ce qui est aussi exact. Enfin, le nombre 0 est entré et le programme arrête son exécution comme prévu.

Suite aux tests sur l'interpréteur, d'autres tests ont été faits pour tester le système IBC-Go dans son ensemble. La figure 5.2 présente la topologie utilisée pour les tests du système. On peut constater que les agences utilisés sont très différentes; ce qui permet de tester la portabilité du système IBC-Go.

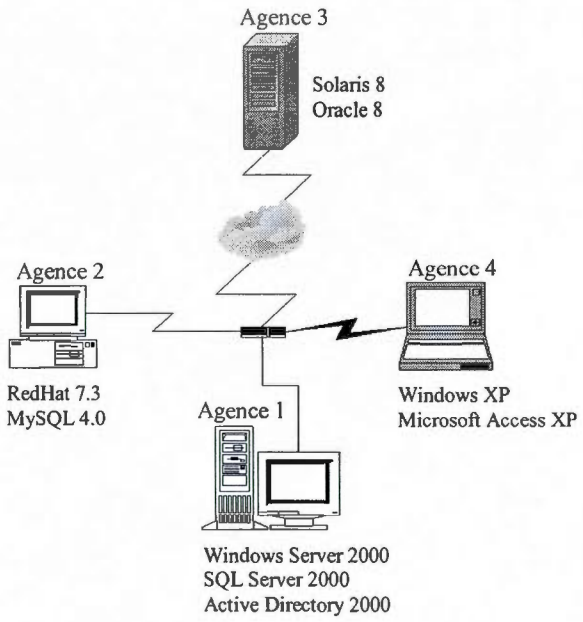

Figure 5.2: Topologie pour les tests

Plusieurs tests ont été faits avec la topologie précédente afin de tester l'ensemble des fonctions d'IBC-Go. Quatre tests sont ici présentés. Le premier est un simple test qui utilise un agent mobile affichant le message "Hello World!". Il est activé par l'agence 1 et est envoyé successivement aux agences 2, 3, 4 et 1. Ce test permet notamment de tester les fonctions de base des agents mobiles. Le code suivant présente le code de ce test.

```
// Hello World!
Program test1 =
begin // test1
        put("Hello World!");
        newLine;
        followIt();
end test1.
```

La figure 5.3 présente les résultats du test. Outre les messages *trace*, les trois fenêtres *telnet*, relatives aux agences 2, 3 et 4, et la console de l'interface graphique, relative à

l'agence 1, affichent le message "Hello World!"; ce qui démontre que le système IBC-Go a passé le premier test.

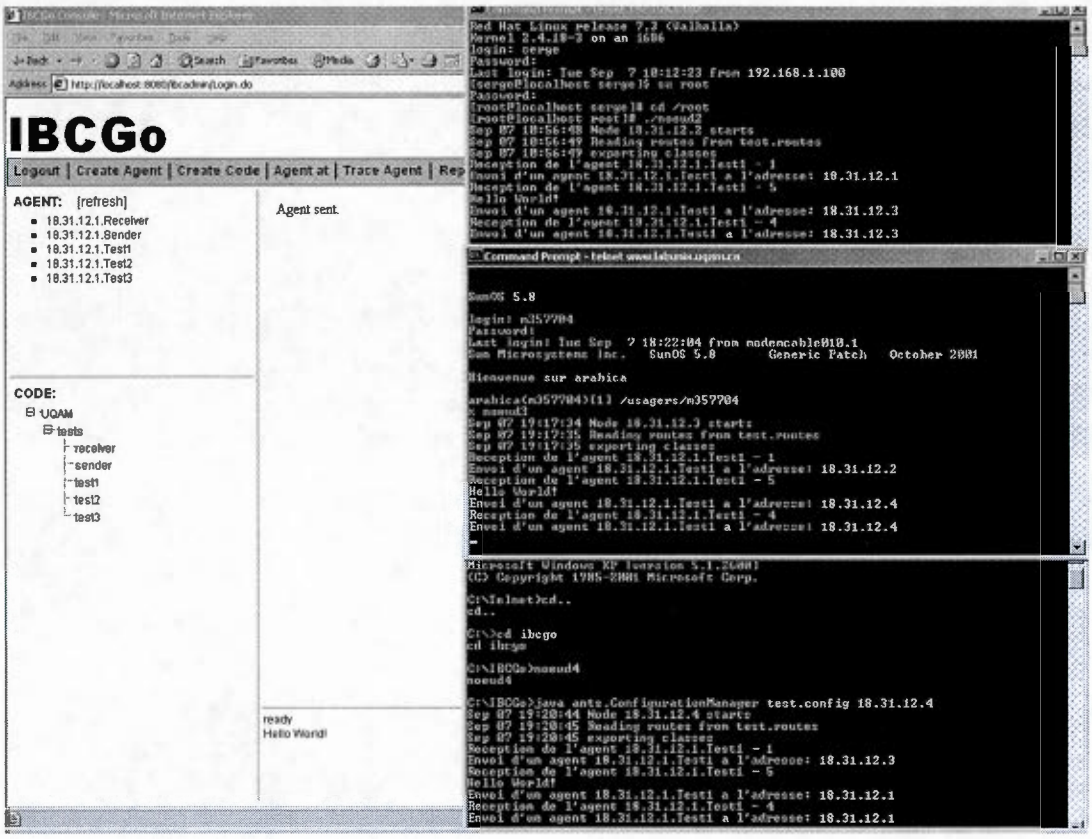

Figure 5.3: Résultat du test 1 sur le système IBC-Go

Le deuxième test utilise un agent mobile qui saisit l'adresse de l'agence qu'il l'héberge et l'emmagasine dans la base de données de l'agence sous la forme, "Je suis au nœud *adresse*". L'agent mobile récupère ensuite l'adresse (de la base de données) et l'affiche à l'écran. Ce test permet notamment de tester les fonctions reliées à une base de données. Le code suivant présente le code de ce test.

```
// Test la fonction get Address et les functions BD
program test2 =
```

```
        var address, mBD : String;
begin // test2
        getAddress(address);
        put(address);
        newLine;
        putMessage(test, "Je suis au noeud "+address);
        getMessage(mBD, test);
        put(mBD);
        newLine;
        followIt();
end test2.
```

La figure 5.4 présente les résultats du présent test. Outre les messages *trace*, les trois fenêtres *telnet* et la console de l'interface graphique affichent le message "Je suis au nœud ", suivi de l'adresse de l'agence; ce qui démontre que le système IBC-Go a passé le deuxième test.

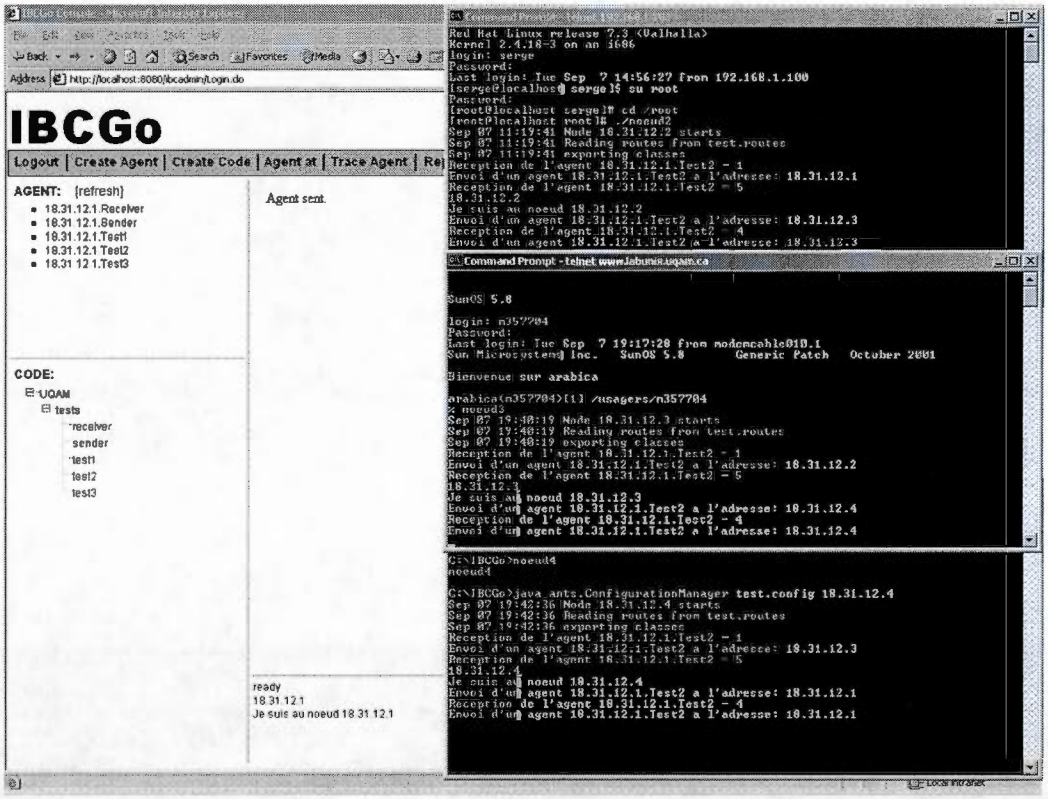

Figure 5.4: Résultat du test 2 sur le système IBC-Go

Le troisième test utilise un agent mobile qui inclut deux procédures, de même que des fonctions de création et de copies d'agents. Ce test crée les codes *Code1* et *Code2* avec le code de ces procédures *Code1* et *Code2* respectivement pour ensuite créer l'agent *NewAgent* avec les codes *Code1* et *Code2* et l'itinéraire: *18.31.12.3*, *18.31.12.4*, *18.31.12.1* (avec des bits K à 0). La procédure *Code1* affiche le message "procedure Code1" alors que la procédure *Code2* affiche le message "procedure Code2". Après avoir créé l'agent *NewAgent*, le troisième test copie ce dernier sous le nom *18.31.12.1.Test32* avec la fonction *copyAgent*, pour ensuite copier l'agent *18.31.12.1.Test32* sous les noms *18.31.12.1.Test33* et *18.31.12.1.Test34* avec la fonction *spawnAgent*. Finalement, le troisième test active *18.31.12.1.Test33*. Ce test a permis notamment de tester les fonctions de création et de copie d'agent. Le code suivant présente le code de ce test.

```
// Test la creation et la copie de codes et agents
program test3 =
        procedure Code1 =
        begin
                put("procedure Code1");
                newLine;
        end Code1;

        procedure Code2 =
        begin
                put("procedure Code2");
                newLine;
                followIt();
        end Code2;

begin // test3
        createCode("Code1", Code1);
        createCode("Code2", Code2);
        createAgent("NewAgent", {"Code1", "Code2"}, "message1");
        It(NewAgent) := ({"0", "18.31.12.3", "0", "18.31.12.4", "0", "18.31.12.1"});
        copyAgent("18.31.12.1.Test32", "NewAgent");
        spawnAgent({"18.31.12.1.Test34","18.31.12.1.Test33"}, "18.31.12.1.Test32");
        activateAgent("18.31.12.1.Test33", false);
end test3.
```

La figure 5.5 présente les résultats du troisième test. Outre les messages *trace*, les trois fenêtres *telnet* et la console de l'interface graphique affiche les messages "procedure

Code1" et "procedure Code2"; ce qui démontre que le système IBC-Go a passé le troisième test.

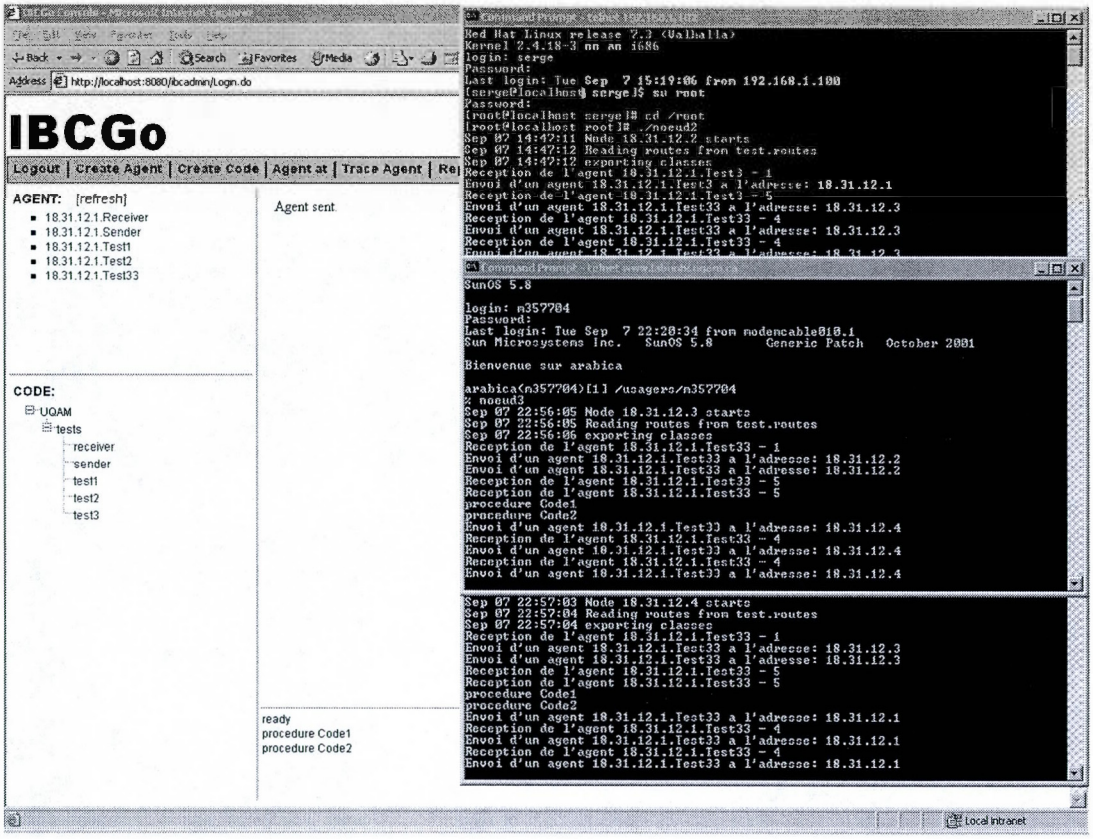

Figure 5.5: Résultat du test 3 sur le système IBC-Go

Le dernier test qui est présenté inclut deux codes. Le premier code, *sender*, envoi le message "Hello World!" à l'agent *18.31.12.1.Receiver*. Le deuxième code, *receiver*, attend le message de l'agent *18.31.12.1.Sender* et l'affiche. Suite à cela, l'agent *18.31.12.1.Receiver* va à la rencontre de l'agent *18.31.12.1.Sender*. Ce test permet notamment de tester les fonctions relatives aux interactions entre les agents mobiles. Le code suivant présente le code de ce test.

```
// Test l'envoi de message
program sender =
        var message : String;
```

```
// Test la reception et la rencontre
program receiver =
        var message : String;
```

```
begin // sender
message := "Hello World!";
        {"18.31.12.1.Receiver"}!message;
end sender.
```

```
begin // receiver
        {"18.31.12.1.Sender"}?message;
        put(message);
        newLine;
        meetAgent("18.31.12.1.Sender");
end receiver.
```

La figure 5.6 présente les résultats du test. Outre les messages *trace*, qui permettent entre autres de voir que l'agent *18.31.12.1.Receiver* va à la rencontre de l'agent *18.31.12.1.Sender*, les trois fenêtres *telnet* montrent le message "Hello World!"; ce qui démontre que le système IBC-Go a passé le dernier test.

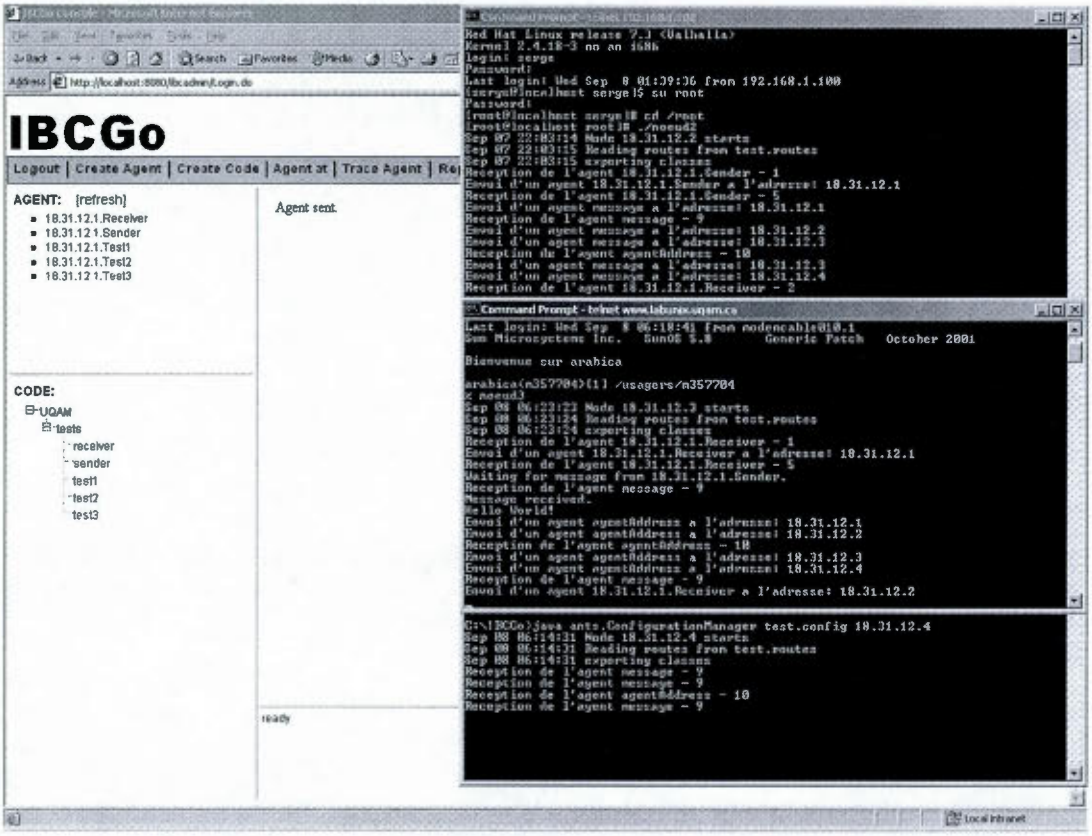

Figure 5.6: Résultat du test 4 sur le système IBC-Go

5.2 Les résultats

Pour démontrer les possibilités d'IBC-Go, 3 scénarios mettant en application le système IBC-Go sont ici présentés. Ces scénarios touchent plus particulièrement les domaines du commerce électronique, de la diffusion audio/vidéo et de la gestion des réseaux.

Pour chaque scénario présenté, des résultats numériqués relatifs à la diminution des besoins en communication entre le nœud d'origine et les autres nœuds, seront faites. Ces dernières permettront de chiffrer les gains amenés par le système IBC-Go.

5.2.1 Le commerce électronique

Le premier scénario de mise en application du système IBC-Go concerne le domaine du commerce électronique. Plus particulièrement, une application faisant appel à des agents mobiles IBC-Go pour vendre et acheter des produits en ligne sera présentée.

L'application de vente et achat en ligne offre des services semblables à ceux offerts par *eBay*. Avec celle-ci, des usagers peuvent afficher des produits à vendre pouvant être achetés par d'autres usagers. Ces produits sont affichés pour une durée limitée et, lorsque l'affichage d'un produit prend fin, l'usager ayant offert le plus d'argent est celui-ci qui achète le produit. Ainsi, un usager voulant acheter un produit doit envoyer une série de requêtes (accompagnées de réponses de la part d'un serveur) pour trouver un produit à acheter, suivre le processus de vente et ajuster son offre d'achat au besoin. Ces opérations nécessitent le maintien d'une connexion entre l'usager et un serveur et peut s'avérer exigeant pour l'usager; particulièrement quand celui-ci veut acheter un produit convoité.

a. Vente et achat en ligne avec IBC-Go

Avec IBC-Go, il serait possible d'utiliser des agents mobiles pour trouver un produit à acheter, suivre le processus de vente et ajuster l'offre d'achat au besoin. Cela ferait en sorte qu'un usager aurait uniquement à être en ligne lors de l'envoi d'un agent mobile et lors de la réception des résultats d'une vente potentielle. De plus, un même agent pourrait visiter plus d'un site pour trouver les meilleurs prix ou pour surveiller la présence de la vente d'un produit difficile à trouver (voir la figure 5.7). Avec le réseau Internet actuel, il est nécessaire d'aller voir régulièrement plus d'un site pour ne pas manquer une bonne occasion d'achat.

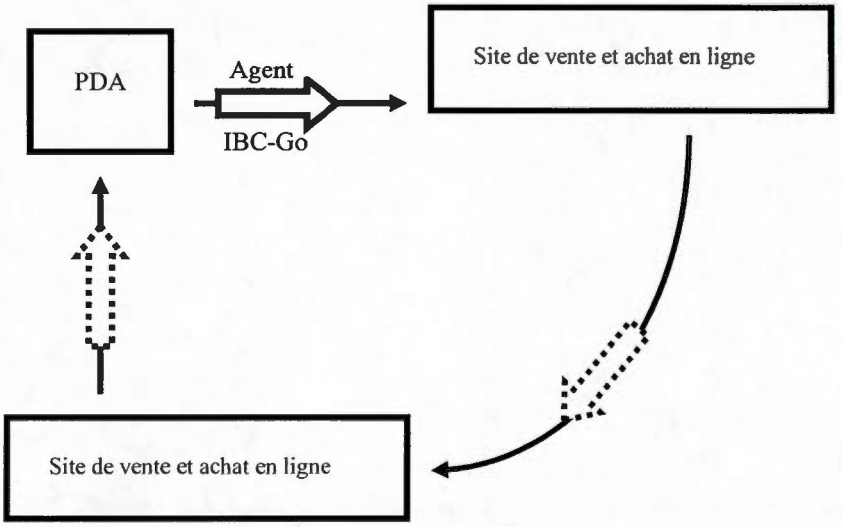

Figure 5.7: Achat en ligne avec IBC-Go

b. Avantages amenés par le système IBC-Go

Tel qu'il a été mentionné précédemment, l'un des avantages amené par le système IBC-Go est la diminution des besoins en communication entre un usager et l'application d'achat et vente en ligne. En effet, il est possible avec le système IBC-Go de réduire le nombre d'échanges de la part d'un usager à un échange (une requête et une réponse); et cela,

peu importe le nombre de sites de vente et le nombre d'items recherchés. Sans l'utilisation du système IBC-Go, il faut au minimum 3 échanges (recherche de l'item, offre d'achat et surveillance du processus de vente) pour chacun des items recherchés et chacun des sites de ventes.

Cela dit, ce n'est pas parce que le nombre d'échanges a diminué que les besoins en communication entre un usager et l'application d'achat et vente en ligne diminuent. Pour vraiment voir les exigences en communication, il est important de voir la quantité de données qui est échangée au lieu du nombre d'échanges. Pour cela, les quantités de donnée échangées avec et sans le système IBC-Go ont été comparées. Pour faire cette comparaison, la taille des paquets IBC-Go a été établie à *2 896* bits + (64 bits x s) + *d* bits (voir l'annexe C), où *s* correspond au nombre de sites visités et *d* correspond à la quantité de données véhiculées. En ce qui concerne la taille des paquets véhiculés sans le système IBC-Go, elle a été établie à *d* (la quantité de données véhiculées tout simplement).

La figure 5.8 montre la quantité de données échangées par un utilisateur qui désire acheter de 1 à 10 items, avec et sans le système IBC-Go, auprès de 1 et 5 sites. Pour ces scénarios, la quantité de données transmises pour les requêtes de l'utilisateur a été établie à 352*i*, où *i* représente le nombre d'items recherchés. 352 bits correspondent à une description de l'item recherché, avec une moyenne de 20 caractères, et le prix maximal à payer pour cet item. La quantité de données transmises pour les réponses du système a été établie à 800 000 bits; ce qui correspond à la taille minimale d'une page transmise par un site comme eBay. Enfin, les scénarios de la figure 5.8 supposent qu'une seule requête et une seule réponse sont transmises pour l'utilisateur, avec le système IBC-Go, et que 3^{12} requêtes et 3 réponses sont transmises pour l'utilisateur pour chaque item et pour chaque site, quand le système IBC-Go n'est pas utilisé.

[12] Le nombre de 3 correspond à un minimum, compte tenu de ce qui a été mentionné précédemment.

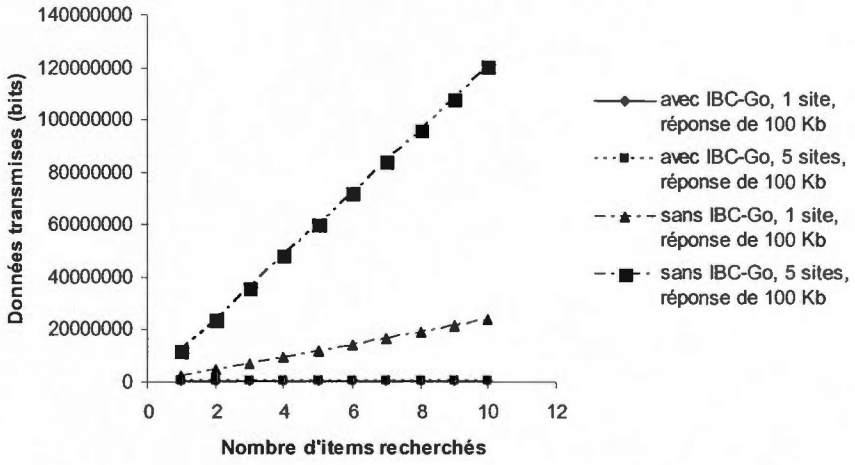

Figure 5.8 : Scénarios incluant des réponses d'une taille de 800 000 bits

Comme on peut le voir avec la figure 5.8, la quantité de données transmises est petite avec l'utilisation du système IBC-Go; comparativement à la quantité de données transmises sans le système IBC-Go. On peut également y voir que la différence des quantités de données transmises s'accroît quand le nombre d'items recherchés augmente, de même que quand le nombre de sites augmente.

Étant donnée que le nombre de réponses est toujours supérieur sans l'utilisation du système IBC-Go, il est évident que plus la taille de la réponse est grande, plus les différences de quantités de données transmises observées seront grandes. Pour vérifier s'il serait possible de modifier ce qui a été observé à la figure 5.8 en diminuant la taille de la réponse, les scénarios précédant ont été repris en établissant la taille des réponses transmises à 1000 bits (ce qui correspond à la transmission d'un peu plus de 100 caractères). La figure 5.9 présente le résultat de ces scénarios.

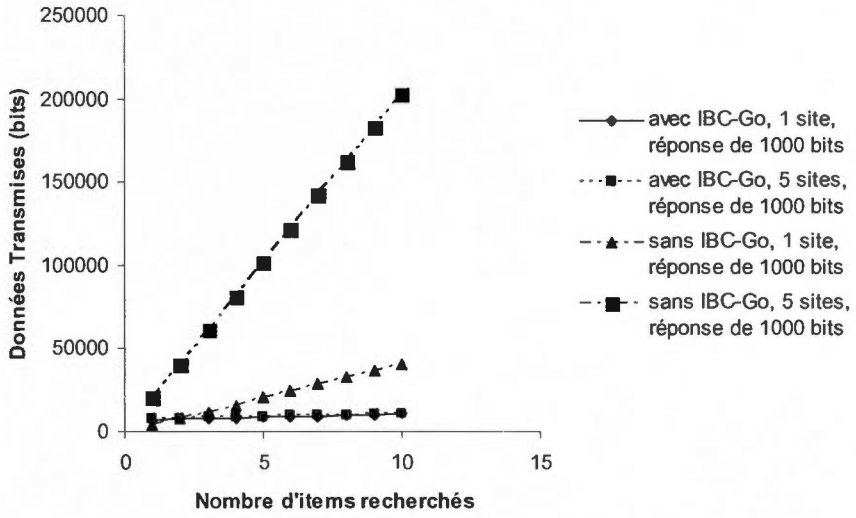

Figure 5.9 : Scénarios incluant des réponses d'une taille de 1000 bits

Comme on peut le voir avec la figure 5.9, les différences de quantités de données transmises observées à la figure 5.8 demeurent; à l'exception du cas où l'utilisateur recherche moins de 2 items auprès d'un seul site.

5.2.2 La diffusion audio/vidéo

Avec le réseau Internet actuel, la transmission audio/vidéo peut s'avérer très exigeante en terme de bande passante et de traitement de la part du serveur (dans le paradigme client/serveur). Il est en effet fréquent qu'une radio ou une télévision Internet soit inaccessible à cause d'une surcharge du serveur ou du manque de bande passante.

Ce problème résulte principalement du fait qu'un diffuseur voulant diffuser sur le réseau Internet doit multiplier l'envoi du contenu à diffuser par son nombre d'auditeurs. Ainsi, un diffuseur ayant 10 000 auditeurs doit envoyer 10 000 fois le contenu qu'il veut

diffuser; ce qui peut s'avérer coûteux en terme de bande passante et de traitement de la part du serveur.

a. Diffusion audio/vidéo avec IBC-Go

Avec IBC-Go, il serait possible de créer un arbre de diffusion pour la transmission audio/vidéo. Pour ce faire, un serveur pourrait envoyer un agent mobile contenant le contenu à diffuser, la liste des adresses des auditeurs, de même que des instructions pour indiquer aux routeurs de faire la multiplication des agents et des envois, afin que tous les auditeurs reçoivent le contenu. La figure 5.10 présente la diffusion audio/vidéo avec le système IBC-Go.

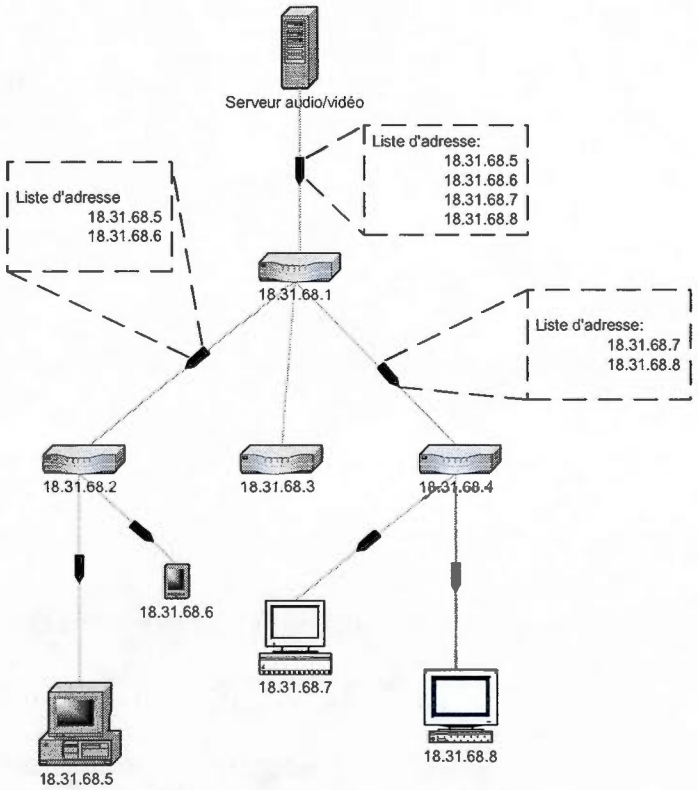

Figure 5.10: Diffusion audio/vidéo avec IBC-Go

Comme on peut le voir avec à la figure 5.10, un serveur audio/vidéo envoie un agent IBC-Go, incluant un contenu audio/vidéo et une liste d'adresses d'auditeurs (de même qu'une série d'instructions) au routeur situé à l'adresse 18.31.68.1. À la réception de l'agent, ce dernier crée deux agents contenant le contenu audio/vidéo de l'agent reçu et une partie de la liste d'adresses (de même que les instructions de l'agent reçu) et envoie les agents créés aux routeurs situés aux adresses 18.31.68.2 et 18.31.68.4. Ces derniers font un travail similaire au routeur précédent et envoient des agents aux auditeurs du serveur audio/vidéo. Il résulte, de ce processus que le serveur audio/vidéo n'a qu'à faire le traitement et l'envoi d'un seul contenu audio/vidéo; indépendamment du nombre d'auditeurs.

b. Avantages amenés par le système IBC-Go

Comme il a été mentionné précédemment, le fait de faire la diffusion audio/vidéo avec le système IBC-Go diminue les besoins en traitement pour le serveur audio/vidéo et en bande passante pour la connexion reliée à ce dernier. Étant donné que l'objectif de ce mémoire est de démontrer la diminution des besoins en bande passante (entre le nœud d'origine et les autres nœuds) amenée par le système IBC-Go; les résultats qui vont suivre porteront exclusivement sur cet aspect.

Pour démontrer la diminution des besoins en bande passante entre le nœud d'origine et les autres nœuds amenée par le système IBC-Go, la bande passante utilisée par la connexion reliée au serveur audio/vidéo (l'origine) pour la diffusion audio/vidéo sera calculée avec et sans l'utilisation du système IBC-Go. En ne tenant pas compte des mécanismes sous-jacents au système (comme IP ou RTSP), il possible de calculer la bande passante utilisée en multipliant le débit de diffusion par le nombre d'utilisateurs, lorsque le système IBC-Go n'est pas utilisé. Lorsque ce dernier est utilisé, la bande passante utilisée pour la transmission des données reliées aux entêtes des paquets IBC-Go, de même que celles

reliées à la liste des adresses des auditeurs, sera ajoutée à la bande passante utilisée pour la diffusion audio/vidéo en tant que telle. En considérant qu'un paquet IBC-Go contient le contenu d'une trame audio/vidéo, il est possible de calculer la bande passante associée aux entêtes IBC-Go en multipliant la taille de l'entête (2 896 bits + 64 bits (pour l'adresse de la première destination) = 2 960) par le fps (*frame per second*).

Ainsi, la bande passante utilisée pour des diffusions de 28Kbps (qui correspond à une radio Internet conventionnelle) et de 300 Kbps (qui correspond à une télévision Internet conventionnelle) pour un nombre d'auditeurs allant de 1 à 10 000 a été calculée, lorsque le système IBC-Go n'était pas utilisé. Ces paramètres ont été repris pour le calcul de la bande passante utilisée avec l'utilisation du système IBC-Go en ajoutant un fps égale à 5 et 25 respectivement. La figure 5.11 montre le résultat de ces calculs.

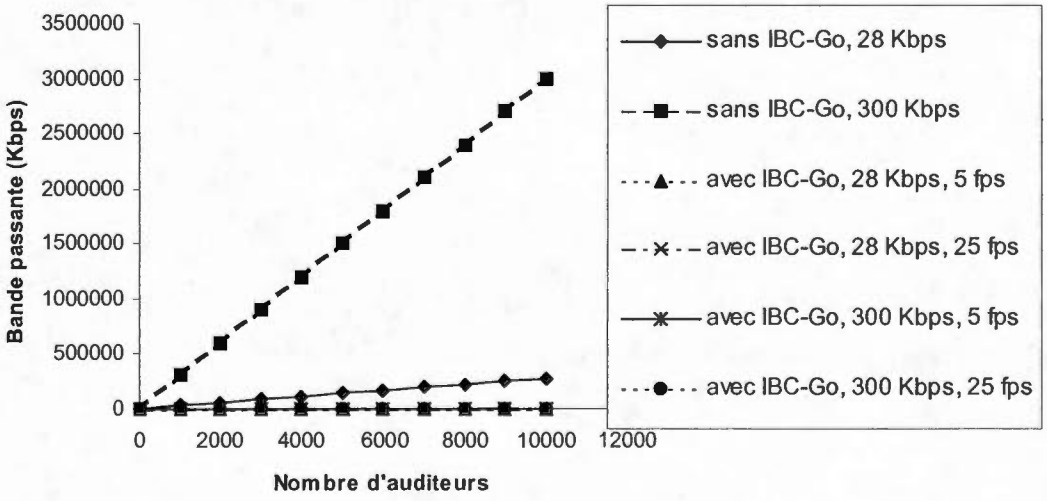

Figure 5.11: Bande passante utilisé pour la diffusion audio/vidéo

Comme on peut le voir à la figure 5.11, la bande passante utilisée (par la connexion reliée au serveur) pour la diffusion audio/vidéo augmente grandement lorsque le système IBC-Go n'est pas utilisé; particulièrement lorsque le débit de diffusion est plus élevé (300

Kbps). Lorsque le système IBC-Go est utilisé, la bande passante utilisée augmente très peu en fonction du nombre d'auditeurs et du débit utilisé; ceci démontre que le système IBC-Go diminue les besoins en bande passante pour la connexion relié au serveur audio/vidéo.

5.2.3 La gestion des réseaux

La gestion de réseau peut se résumer aux opérations qui touchent au maintien du réseau opérationnel ainsi qu'à la surveillance et au contrôle des équipements du réseau. La gestion des équipements du réseau peut se faire localement (sur place) ou à distance. Lorsque le réseau est très étendu, incluant un grand nombre d'équipements, la gestion locale des équipements du réseau devient impossible. Dans ce cas, il est nécessaire de gérer les équipements du réseau à distance [ADV 2003]. Avec ce type de gestion, un ordinateur central communique avec les divers appareils du réseau pour obtenir de l'information les concernant. Lorsqu'un problème survient, divers outils, comme Telnet ou VNC (*Virtual Network Computing*) ou des outils de gestion de réseaux, peuvent être utilisés. Pour communiquer entre eux, l'ordinateur central et les divers équipements du réseau peuvent utiliser un protocole comme SNMP (*Simple Network Management Protocol*) ou autres.

Avec la croissance de l'Internet et des réseaux en général l'architecture de gestion centralisée découlant de la gestion des réseaux à distance devient de plus en plus inefficace. Cela résulte du fait que l'ordinateur central en vient à consommer une bonne quantité de la bande passante disponible pour accomplir ces différentes tâches [TIM 2003].

a. Gestion des réseaux avec IBC-Go

IBC-Go peut être utilisé pour diminuer les besoins en communication entre le gestionnaire et les autres nœuds et possiblement, des besoins en communication pour le réseau en général. En effet, il est possible avec IBC-Go, d'envoyer un agent mobile incluant

un itinéraire correspondant aux différents équipements du réseau pour collecter de l'information sur les états des nœuds et pour effectuer des opérations au besoin. De cette façon, le gestionnaire n'envoie qu'une seule requête qui est véhiculé à chacun des nœuds, pour ensuite revenir à son origine (voir la figure 5.12).

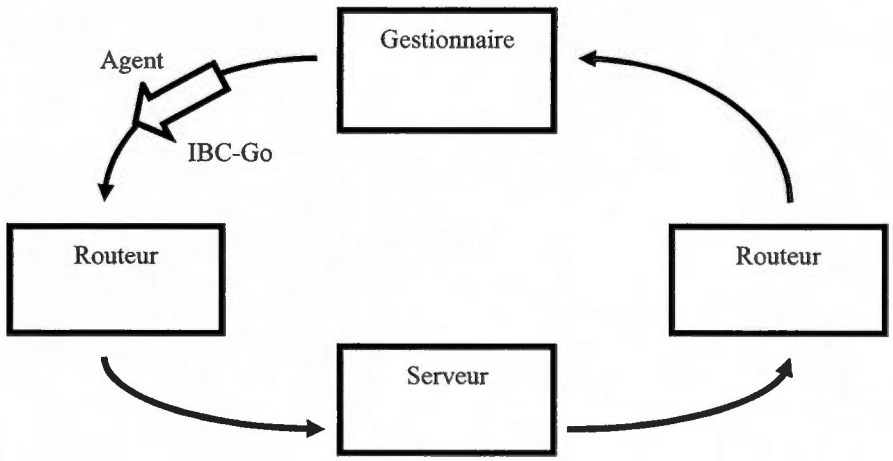

Figure 5.12: Gestion des réseaux avec agents mobiles

b. Avantages amenés par le système IBC-Go

Comme il a été mentionné précédemment, la gestion des réseaux avec IBC-Go permet diminuer les besoins en communication entre le gestionnaire et les autres nœuds.

Pour démontrer la diminution des besoins en communication (entre le gestionnaire et les autres nœuds) amenée par le système IBC-Go, la quantité d'information véhiculée par une connexion reliée au gestionnaire sera calculée avec et sans le système IBC-Go. En ne tenant pas compte des mécanismes sous-jacents au système, en prenant pour acquis que chaque nœud reçoit une seule requête du gestionnaire et renvoie une seule réponse et établissant la taille d'une requête ou d'une réponse à 2 400 bits (qui correspond à la taille approximative d'un paquet SNMP) il possible de calculer la quantité d'information véhiculée comme étant 2 x 2 400 x nombre de nœuds lorsque le système IBC-Go n'est pas utilisé. Lorsque ce dernier

est utilisé, nous pouvons calculer la quantité de donnée véhiculée en additionnant la taille de la requête avec la taille de la réponse. En établissant que la taille des données transportées comme étant également à 2 400 bits (la taille d'un paquet SNMP), nous pouvons calculer la taille de la requête avec la formule suivante: 2 896 + 64 x nombre de nœuds + 2 400. La taille de la réponse peut être établie de la même manière, à l'exception de la taille des données transportées. Dans ce cas, les données transportées par cette réponse contiendraient un rapport sur le réseau qui serait de taille variable. Pour le besoin du présent calcul, des valeurs de 100 Kb et 1 000 Kb ont été utilisées pour la taille des rapports. La figure 5.13 présente les résultats du calcul des données véhiculées par une connexion reliée au gestionnaire pour un nombre de nœuds allant de 1 à 10 000.

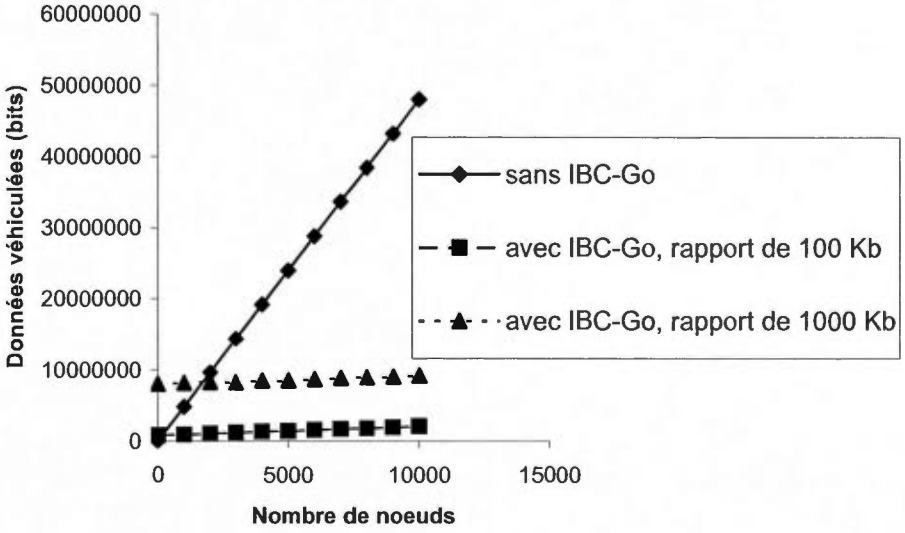

Figure 5.13: Données véhiculées via le gestionnaire pour la gestion des réseaux

Comme on peut le voir avec la figure 5.13, la quantité de données véhiculées par une connexion reliée au gestionnaire augmente très peu en fonction du nombre de nœuds du réseau en utilisant le système IBC-Go; comparativement au fait de ne pas utiliser le système IBC-Go. Cela demeure vrai indépendamment de la taille du rapport sur l'état du réseau.

Même si l'objectif de ce mémoire est uniquement de démontrer que le système IBC-Go diminue le besoin en communication entre le nœud d'origine et les autres nœuds, la quantité de données véhiculées sur l'ensemble du réseau a de plus été calculées pour voir si le système IBC-Go amène une diminution à ce niveau-là; ce qui permettrait de répondre à la problématique de la gestion des réseaux décrite précédemment. La quantité de données véhiculées par l'ensemble du réseau correspond à la taille de la requête et de la réponse multipliée par le nombre de nœuds, lorsque le système IBC-Go n'est pas utilisé. Lorsque ce dernier est utilisé, la quantité de données véhiculées correspond à la taille de la requête, multipliée par le nombre de nœuds plus 1 (le retour au gestionnaire). La figure 5.14 montre les résultats de ces calculs.

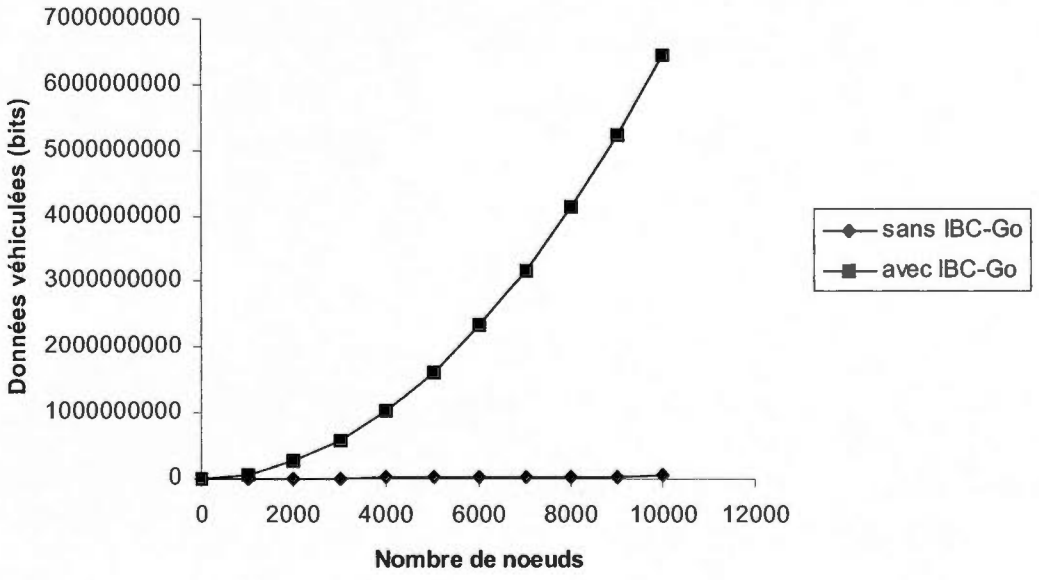

Figure 5.14: Données véhiculées par l'ensemble du réseau pour la gestion des réseaux

Comme on peut le voir avec la figure 5.14, la quantité de données véhiculées par l'ensemble du réseau augmente considérablement avec l'utilisation du système IBC-Go. Cela s'explique par le fait que la taille des requêtes, avec le système IBC-Go, est supérieure à la somme de la taille de la requête et de la réponse, sans l'utilisation d'IBC-Go. Si on soustrait

le poids de l'itinéraire à la taille des requêtes du système IBC-Go, on peut voir que la quantité de données véhiculées est semblable avec ou sans l'utilisation du système IBC-Go (voir la figure 5.15).

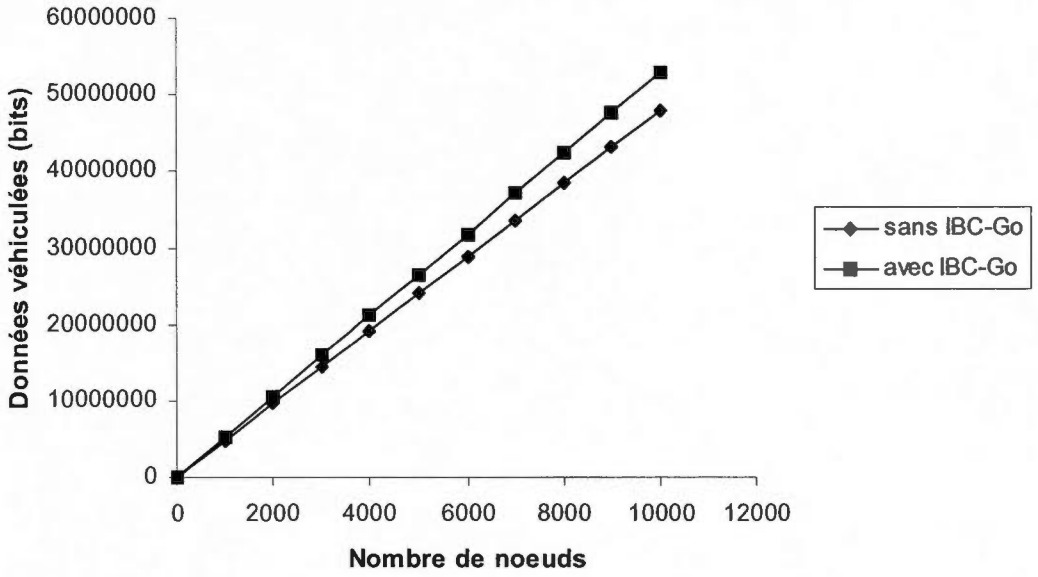

Figure 5.15: Données véhiculées moins le poids de l'itinéraire

Il serait possible, dans ce cas-ci, de placer l'itinéraire dans le code pour qu'il ne soit pas véhiculé avec les agents mobiles. Cela ferait en sorte que la quantité des données véhiculées par l'ensemble du réseau serait du même ordre, avec ou sans l'utilisation du système IBC-Go.

Comme on peut le voir avec la figure 5.15, le système IBC-Go ne diminue pas les besoins en communication pour l'ensemble du réseau, lorsqu'une seule requête et réponse est faite pour chaque nœud du réseau. Il en va autrement si plus d'une requête (suivi d'une réponse) est faite pour chaque nœud du réseau. Dans ce cas, le nombre des échanges augmenterait lorsque le système IBC-Go ne serait pas utilisé; ce qui ne serait pas le cas avec

l'utilisation du système IBC-Go. La figure 5.16 montre la quantité de données véhiculées par l'ensemble du réseau lorsqu'il y a plus d'un échange pour chaque nœud du réseau.

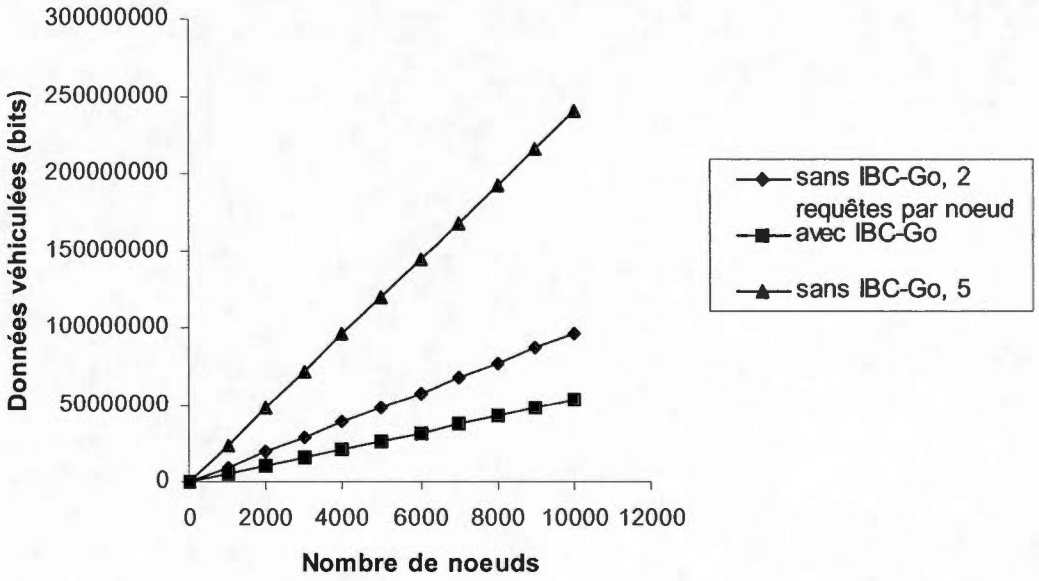

Figure 5.16: Données véhiculées lorsqu'il y a plus d'un échange par nœud

Comme on peut le voir avec la figure 5.16, la quantité de données véhiculées par l'ensemble du réseau est supérieure, sans l'utilisation d'IBC-Go, lorsqu'il y a plus d'un échange pour chaque nœud du réseau; comparativement au fait d'utiliser le système IBC-Go.

CHAPITRE VI

CONCLUSION

La popularité grandissante de réseau comme l'Internet amène de nouvelles tendances dont l'augmentation des PDAs et l'informatique nomade. Ces nouvelles tendances amènent un accroissement de connexions lentes et instables en périphérie du réseau et de nouvelles exigences de la part des utilisateurs du réseau. Pour répondre aux besoins suscités par ces nouvelles tendances; les systèmes distribués actuels doivent s'adapter.

Des recherches récentes relatives au code mobile, sous la forme d'agents mobiles et de réseaux actifs, offrent de nouvelles possibilités intéressantes pour le développement des systèmes distribués. La fusion de ces deux secteurs de recherches est susceptible d'amener de nouveaux mécanismes pour construire des systèmes distribués plus gros, plus robustes et plus malléables.

Dans ce contexte, le système IBC-Go a été conçu et développé. IBC-Go est un système d'agents mobiles, basé sur une approche itinéraire, qui intègre les technologies de réseau actif. Ce système permet de répondre aux exigences de divers types d'applications en diminuant les besoins liés aux connexions en périphérie du réseau; plus particulièrement en diminuant les besoins de transmission entre un nœud qui crée et envoie des agents mobiles (un appareil manipulé par un utilisateur, comme un PDA, un serveur, etc.) et les autres nœuds du réseau (routeurs, serveurs multiples, postes de travail multiples, etc.).

En plus des mécanismes inhérents aux systèmes d'agents mobiles et de l'apport inhérent aux réseaux actifs, le système IBC-Go inclut divers mécanismes pour effectivement réduire les besoins en communications, incluant les besoins de transmission entre un nœud d'origine et les autres nœuds. Ces mécanismes sont notamment: l'emploi d'itinéraire, la séparation de la transmission du code de l'agent de l'agent en tant que tel et l'emploi d'une

mobilité faible pour les agents. Alors que le premier mécanisme permet à un agent de visiter plusieurs nœuds, sans avoir à communiquer avec son expéditeur original, les deux autres mécanismes permettent plus particulièrement de diminuer le poids des agents; ce qui se traduit par une diminution de données transmises.

Aussi, le système IBC-Go inclut un langage spécifique pour la programmation des agents mobiles et des mécanismes de sécurité pour rendre le système plus sécure. Le fait d'utiliser un langage spécifique pour le développement des agents mobiles permet de ne pas restreindre la programmation des agents au langage utilisé pour le développement du système sous-jacent et ne nécessite pas de comprendre les détails du fonctionnement du dit système; ce qui fait en sorte que le système IBC-Go peut être utilisé par des utilisateurs autres que des experts en informatique. L'inclusion d'un langage spécifique et de mécanismes de sécurité a contribué à répondre à l'un des objectifs de ce mémoire qui est développer un système sécure et facile d'utilisation.

Après avoir fait une revue des technologies de réseaux actifs, le prototype de réseau actif ANTS été choisi pour supporter le développement du système IBC-Go. ANTS été choisi en bonne partie dû à la portabilité inhérente à ce prototype. En ce qui attrait à l'environnement et au langage, Java a été choisi pour le développement du système IBC-Go. L'environnement et le langage Java ont été choisis, compte tenu de leur portabilité et de leur omniprésence dans le domaine des agents mobiles; ce qui a contribué à répondre à l'objectif de développer un système portable. Pour le développement des différentes composantes du système IBC-Go, différents choix et stratégies ont été mis de l'avant, comme le développement d'une interface graphique Web, l'emploi d'ANTLR pour le développement de l'interpréteur pour le langage AL et l'adoption de l'approche *assistant routeur* pour l'intégration des routeurs actuellement utilisés.

Pour tester le système IBC-Go, différents tests ont été réalisés avec 4 nœuds différents; ce qui a permis de vérifier le bon fonctionnement et la portabilité du système.

Enfin, pour montrer les possibilités du système IBC-Go et pour démontrer les avantages amenés par le système IBC-Go, plus particulièrement de démontrer le fait que le dit système amène une diminution des besoins en communication entre le nœud d'origine et les autres nœuds; 3 scénarios mettant en application le système IBC-Go ont été présentés. La présentation de chacun de ces scénarios a également été accompagnée de résultats numériqués qui ont permis de chiffrer les gains amenés par le système IBC-Go. La présentation des scénarios et des résultats numériqués a permis de répondre l'objectif qui était de démontrer que le système développé est davantage en mesure de solutionner les problèmes soulevés dans la problématique que les systèmes actuels, et plus particulièrement de diminuer les besoins en communication entre le nœud d'origine et les autres nœuds.

Ainsi, les 3 objectifs de ce travail de maîtrise qui étaient:

- de développer un système d'agent mobile incluant une architecture basée sur un modèle de réseau actif;
- de développer un système portable, sécure et facile d'utilisation;
- de démontrer que le système développé est davantage en mesure de solutionner les problèmes soulevés dans la problématique que les systèmes actuels;

ont été réalisés. Cela dit, il aurait été intéressant de démontrer davantage qu'IBC-Go amène des gains par rapport aux systèmes actuels. En effet, les résultats qui ont été présentés se sont essentiellement attardées à démontrer le fait que le système IBC-Go amène une diminution les besoins en communication entre le nœud d'origine et les autres nœuds, compte tenu du temps accordés à ce travail; mais le système amène également d'autres gains, comme l'enrichissement des interfaces et la réduction des besoins en communication pour l'ensemble du réseau.

Aussi, il aurait été intéressant d'analyser les performances du système en terme de temps d'exécution et de consommation des ressources. Cette analyse aurait possiblement

révélé des gains ou des pertes, dépendamment de la portion du système ou du réseau qui aurait été analysée. Toujours en lien avec la performance, le système IBC-Go, conçu pour diminuer les besoins en communication, n'a pas été conçu pour être le plus performant possible. Il serait possible, en ce sens, d'améliorer les performances du système IBC-Go en compilant le code des agents et en incluant davantage de classes écrites dans un langage plus performant comme C++. Dans ce dernier cas, il faudrait par contre être prudent de ne pas rendre le système moins portable.

ANNEXE A

LE LANGAGE AL

Le système IBC-Go utilise un langage particulier pour programmer ces agents mobiles: AL (*Agent Language*). AL[13] distingue les lettres minuscules des lettres majuscules. Le nombre d'espaces n'est pas considéré (1 espace = *n* espaces) et les fins de ligne sont prises en compte. Aussi, les commentaires commencent par deux barres obliques vers l'avant (//) et se terminent à la fin de la ligne; comme en C++.

a. Identificateurs

Les identificateurs commencent par une lettre et contiennent des lettres et des chiffres. Ils doivent être contenus sur une ligne et seul les 100 premiers caractères sont pris en compte.

b. Mots réservés

Les mots clés suivants sont réservés dans AL:

and	array	Boolean	begin	Char
constant	else	elsif	end	exit
function	if	Integer	loop	mod
not	of	or	out	procedure
program	record	return	String	then
type	var	when	while	

[13] Le langage AL est inspiré du tutorial ANTLR de Scott Stanchfield.

c. Types de données

AL connaît les types Boolean, Char, Integer et String.

d. Type Boolean

Une donnée de type Boolean est utilisée pour afficher les valeurs logiques. Il peut prendre uniquement les deux valeurs prédéfinies, soit *true* et *false*, respectivement vrai et faux.

e. Type Char

Une donnée de type Char (de l'anglais *character*) est utilisée pour mémoriser un caractère du jeu en usage sur l'ordinateur.

f. Type Integer

Une donnée de type Integer est utilisée pour mémoriser un nombre entier compris entre – 2 147 483 648 et 2 147 483 647.

g. Type String

Une donnée de type String est utilisée pour mémoriser une chaîne de caractères.

h. Générateurs de types

AL permet de générer trois types: enumeration, array et record. Le type enumeration est utilisé pour définir une collection de noms de constantes entières. Par exemple:

```
Type Saison = <PRINTEMPS, ETE, AUTOMNE, HIVER>;
('PRINTEMPS' = 0, 'ETE' = 1, 'AUTOMNE' = 2 et 'HIVER' = 3)
```

Le type array est utilisé pour définir un tableau à une dimension. Par exemple:

```
type Table = array[1..10] of Integer;
```

Le type record est utilisé pour définir une liste de composantes, incluant leurs noms et types respectifs. Par exemple:

```
type Date =
        record
                day : Integer;
                month : Integer;
                year : Integer;
        end record;
```

i. Constantes

Les constantes sont introduites avec une déclaration de la forme suivante:

```
constant ID, ID, ..., ID : nomType := expression;
```

nomType est un identificateur qui identifie le type de donnée et est égale à: Boolean, Char, Integer ou Strin'. Une affectation doit obligatoirement se faire avec: *:= expression*. Par exemple:

```
constant maxIndex : Integer := 100;
```

j. Variables

Les variables sont introduites avec une déclaration de la forme suivante:

```
var ID, ID, ..., ID : nomType := expression;
```

nomType est un identificateur qui identifie le type de donnée et est égale à: Boolean, Char, Integer ou String. Une valeur est optionnellement affectée avec: *:= expression*. Par exemple:

```
var i : Integer := 1;
var b1, b2 : Boolean;
```

k. Opérateurs

Les opérateurs, en ordre de précédence, sont:

Négation booléenne	not
Opérateurs de signe	+ -
Opérateurs multiplicatifs	* / mod
Opérateurs d'additions	+ -
Opérateurs relationnels	= /= < <= > >=
Opérateurs logiques	and or

l. Assignations

L'assignation se fait avec l'opérateur :=. Par exemple:

```
i := 2 * 5;
text := "bonjour";
a := 'a';
```

m. Instructions conditionnelles

L'instruction *if* prend pour paramètre une expression de type Boolean. Cette expression est évaluée et produit un résultat valant *true* ou *false*. Si le résultat est *true*, l'instruction ou le bloc entre *then* et *elsif*, *else* ou *end if* est exécuté. Si le résultat est *false*, l'instruction ou le bloc entre *then* et *elsif*, *else* ou *end if* est ignoré. L'instruction *elsif* est identique à l'instruction *if*, à l'exception que celle-ci est exécutée seulement quand l'expression de l'instruction *if*, ou *elsif*, précédente est fausse. Une instruction *if* peut inclure plusieurs instructions *elsif*. Enfin, si une instruction *else* est incluse dans une instruction *if*, le bloc entre *else* et *end if* est exécuté quand les expressions des instructions *if* et *elsif* précédentes sont fausses. Par exemple:

```
if x > MIN then
        MAX := x;
elsif x < MIN then
        MIN := x;
else
        x := x + 1;
end if;
```

n. Boucles

En AL, une boucle peut être définie de deux façons, avec et sans instruction *while*. Quand il n'y a pas d'instruction *while*, le bloc entre *loop* et *end loop* sont exécuté indéfiniment. Pour sortir de cette boucle, il faut inclure une instruction *exit* dans le bloc. L'instruction *exit* prend pour paramètre une expression de type Boolean. Cette expression est évaluée et produit un résultat valant *true* ou *false*. La boucle se termine quand le résultat est égal à *true*. Par exemple:

```
x := 1;
loop
        x := x + 1;
        exit when x > 10;
end loop;
```

Quand il y a une instruction *while*, c'est l'expression incluse avec l'instruction *while* qui détermine la fin de la boucle. La boucle se termine quand l'expression de l'instruction *while* est égale à *false*. Par exemple:

```
x := 1;
while x <= 10 loop
        x := x + 1;
end loop;
```

o. Procédures

AL permet d'écrire des procédures. Ces procédures peuvent accepter des paramètres, de types Boolean, Char, Integer ou String, et peuvent retourner une valeur, de types Boolean, Char, Integer ou String. Les paramètres sont passés exclusivement par valeur. Par exemple:

```
procedure Salutation (var name : String) =
        constant BONJOUR : String := "Bonjour ";
begin
        put(BONJOUR + name);
        newLine;
        return true;
end Salutation;
```

Pour exécuter une procédure, l'instruction *call* est utilisée. Par exemple:

```
Boolean ok := call Salutation("toi");
```

ANNEXE B

LES FONCTIONS

IBC-Go inclut une série de fonctions permettant aux agents mobiles d'accomplir leur tâche. Cela inclut des fonctions relatives à la création et la destruction de code et d'agent, de communication entre agents, de gestion d'agents, etc. L'ensemble des fonctions est présenté dans les tableaux suivants.

FONCTION	DESCRIPTION	EXEMPLE
get(VARIABLE)	Saisi un message, entré par un utilisateur, et le place dans la variable VARIABLE.	Var message : String; get(message);
put(MESSAGE)	Affiche le message MESSAGE à l'écran. Cela n'inclut pas de saut de ligne.	Put("Bonjour");
newLine	Exécution d'un saut de ligne.	

Fonctions d'entrées/sorties à l'écran.

FONCTION	**DESCRIPTION**	**EXEMPLE**
Get(VARIABLE, FILE)	Met le contenu du fichier FILE dans la variable VARIABLE. FILE inclut le chemin conduisant au fichier.	Var contenu : String; get(contenu, "C:\message.txt");
put(FILE, CONTENT)	Met CONTENT dans le fichier FILE. FILE inclut le chemin conduisant au fichier.	Get("C:\message.txt", "Bonjour");

* Ce chemin peut prendre la forme d'un chemin Java (i.e. répertoire//fichier), Unix/Linux (i.e. répertoire/fichier) ou DOS/Windows (i.e. disque:\répertoire\fichier).

Fonctions de lecture et d'écriture de fichier.

FONCTION	DESCRIPTION	EXEMPLE
putMessage (NAME, MESSAGE)	Emmagasine le message MESSAGE nommé NAME à l'agence où se trouve l'agent.	putMessage (message1, "Bonjour");
getMessage (VARIABLE, NAME)	Prend le message nommé NAME de l'agence où se trouve l'agent et le place dans la variable VARIABLE.	var message : String; getMessage (message, message1);
getAddress(VARIABLE)	Met l'adresse de l'agence dans la variable VARIABLE,	var address : String; getAddress(address);
agentAt(AGENCE)	Permet d'obtenir la liste des agents associés à l'agence AGENCE. Cette liste est remise sous la forme: "agent1;agent2;...". Si aucun agent n'est présent à l'agence AGENCE, "null" sera retourné.	put(agentAt("18.31.12.1"));

Fonctions relatives à une agence.

FONCTION	DESCRIPTION	EXEMPLE
getPayload(VARIABLE)	Saisit la charge utile et la place dans la variable VARIABLE. La charge utile est composée d'un message texte.	Var payload : String; getPayload(payload);
putPayload(MESSAGE)	Met le message MESSAGE dans la charge utile.	PutPayload("Bonjour");

Fonctions de lecture et d'écriture de la charge utile.

FONCTION	DESCRIPTION	EXEMPLE
createCode (NAME, PROCEDURE)	Saisit le code contenu dans la procédure PROCEDURE (qui contient aucun paramètre et qui ne retourne pas de valeur) et le place dans le répertoire associé à l'agence sous le nom NAME.	Procedure Code = begin put("procedure Code"); newLine; end Code; ... createCode(« Code1 », Code);
destroyCode(NAME)	Élimine le code nommé NAME du répertoire associé à l'agence.	DestroyCode(« Code1 »);

Fonctions de création et de destruction de code.

FONCTION	DESCRIPTION	EXEMPLE
createAgent(NAME, {CODE1,CODE2,...}, PAYLOAD)	Créé un agent nommé NAME, incluant les codes CODE1, CODE2, ... et la charge utile PAYLOAD et le place dans le répertoire associé à l'agence sous le nom NAME. Les codes inclus doivent avoir été créés précédemment et être présents dans le répertoire associé à l'agence. Si l'agent est déjà existant, il sera remplacé par celui-ci. Les champs *Créateur* et *Crédits*, de l'agent qui créé l'agent, sont transmis à l'agent créé.	createAgent("NewA", "Code1","");
destroyAgent(NAME)	Élimine l'agent nommé NAME du répertoire associé à l'agence.	destroyAgent("NewA");
copyAgent (NAME2, NAME1)	Copie l'agent nommé NAME1 et le place dans le répertoire associé à l'agence sous le nom NAME2. L'agent NAME1 doit se trouver dans le répertoire associé à l'agence.	copyAgent ("NewA2", "NewA");
spawnAgent({NAME2, NAME3,...}, NAME1)	Copie l'agent nommé NAME1 et le place dans le répertoire associé à l'agence sous le nom NAME2, NAME3, ... L'agent NAME1 doit se trouver dans le répertoire associé à l'agence.	spawnAgent({"NewA2", "NewA3"}, "NewA");
activateAgent (NAME, KEEP)	Envoie l'agent nommé NAME à la première agence de son itinéraire pour qu'il y soit exécuté. Lorsque KEEP égale 'false', les codes associés à l'agent sont téléchargés; même s'ils sont déjà existants aux sites visités. KEEP doit être mis à 'false' quand un code associé à l'agent a été modifié. Lorsque KEEP égale 'true', un code associé à l'agent n'est pas téléchargé, lorsqu'il existe déjà un code ayant le même nom.	activateAgent ("NewA", false);
origin(AGENT)	Permet d'obtenir l'agence d'origine d'un agent.	origin("NewA");

Fonctions relatives aux agents.

FONCTION	DESCRIPTION	EXEMPLE
It(AGENT) := **({K1, S1, K2, S2, ...})**	Associe l'itinéraire incluant les sites S1, S2, ..., et les K (*Keep*) correspondant, à l'agent AGENT. L'agent doit avoir été créé et être présent dans le répertoire associé à l'agence.	It(A1):={{0,"18.31.12.2",0,"18.31.12.3"});
followIt()	Indique à l'agent d'aller à la prochaine agence de son itinéraire. Le code se trouvant après cette fonction ne sera pas exécuté. Cette fonction n'a aucun effet, si l'agent se trouve à la dernière agence de son itinéraire.	
jumpTo(X)	Indique à l'agent d'aller à la *n* ième agence de son itinéraire; les agences de l'itinéraire étant numérotées de 1 à	jumpTo(3);
meetAgent(AGENT)	Indique à l'agent d'aller à l'agence où se trouve l'agent AGENT en suivant son itinéraire.	meetAgent("NewA");
borrowIt(AGENT)	Indique à l'agent d'emprunter l'itinéraire de l'agent AGENT comme s'il s'agissait de son propre itinéraire.	borrowIt("NewA");
meetAgent (AGENT, itAGENT)	Indique à l'agent d'aller à l'agence où se trouve l'agent AGENT en suivant l'itinéraire de l'agent itAGENT.	meetAgent ("NewA", "NewA2");
resetIt()	Remet l'itinéraire de l'agent à son état initial.	

Fonctions relatives à l'itinéraire d'un agent.

FONCTION	DESCRIPTION	EXEMPLE
{toAGENT}!MESSAGE	Envoie le message MESSAGE à l'agent toAGENT.	{« 18.31.12.3.Receiver »}! « salut »;
{toAGENT, toAGENT2, ...}!MESSAGE	Envoie le message MESSAGE aux agents, toAGENT, toAGENT2, ...	{« NewA », « NewA2 »}! »salut »;
{fromAGENT}? MESSAGE	Reçoit un message de l'agent fromAGENT et le place dans la variable MESSAGE.	Var message : String; {« NewA »}?message;
{}!!MESSAGE	Diffuse un message à tous les agents se trouvant sur la même agence que l'agent qui envoie le message. Comme pour la fonction '{AGENT}!MESSAGE', ces agents doivent utiliser la fonction '{AGENT}?MESSAGE' pour recevoir le message.	{}!! »Bonjour »;

Fonctions de communication.

FONCTION	DESCRIPTION	EXEMPLE
pollAgent (AGENT, itAGENT)	Permet à un agent de sonder l'agent AGENT via l'itinéraire de l'agent itAGENT.	pollAgent ("Agent2", "Agent2");
respondTo (AGENT, itAGENT)	Permet à un agent de répondre au sondage de l'agent AGENT via l'itinéraire de l'agent itAGENT.	respondTo ("Agent1", "Agent2");
traceAgent (AGENT, MESSAGE)	Permet d'envoyer une demande de compte rendu à l'agent AGENT. L'agent précédent renvoie: son itinéraire, sa charge utile, son emplacement, etc.	traceAgent ("Agent1", "Bonjour");
traceAgent(AGENT, MESSAGE, itAGENT)	Permet d'envoyer un message à un message à l'agent AGENT via l'itinéraire de l'agent itAGENT.	traceAgent("Agent1", "Agent2", "Bonjour");
reportTo (origin(AGENT))	Indique à un agent de se reporter à son agence d'origine.	reportTo(origin("Agent1"));

Fonctions de gestion.

ANNEXE C

LA TAILLE DES PAQUETS IBC-Go

Le tableau qui va suivre présente la taille du paquet IBC-Go et de ces composantes. La présentation de ces tailles sert essentiellement de référence pour les résultatss présentées au chapitre 5.

Composantes	Taille en bits
Paquet IBC-Go	2 896 + 64s* + d*
Entête ANTS	288
Version	32
MD5	128
Adresse source	32
Adresse destination	32
Adresse précédente	32
TTL	32
Entête IBC-Go (excluant les codes et l'itinéraire)	1136
Identificateur	160 (10 car. approx.)
Longueur du paquet	32
Temps de création	64
Site d'origine	32
Site de destination	32
Créateur	160 (10 car. approx.)
Type de paquet	32
Autorisation	32
Signature de l'autorisation	368
Signataire de l'autorisation	160 (10 car. approx.)
Nombre de codes	32
Nombre d'itinéraires	32
Iitinéraire	528 + 64s
Signature de l'itinéraire	368
Signataire de l'itinéraire	160 (10 car. approx.)
K	32 (par site)
Site	32 (par site)
Code	944
Type de code	32
Longueur du code	32
Donnée du code	320** (20 car. approx.)
Signature du code	368
Signataire du code	160 (10 car. approx.)
Adresse de l'interpréteur	32

* s: nombre de sites
d: donnée

** Le code n'est pas véhiculé avec les agents. Seul le nom du code-ci y est véhiculé. Aussi, un seul code est ici prévu, ce qui facilite les calculs lors des résultats présentés. Il est raisonnable d'utiliser un seul code pour les démonstrations, compte tenu qu'il est toujours possible d'inclure l'ensemble de la logique d'un agent mobile à l'intérieur d'un seul code.

ANNEXE D

LE CODE DU SYSTÈME IBC-Go

La présente section présente le code qui a servi à l'implémetation du système IBC-Go. Celui-ci est découpé en 4 groupes: l'interface graphique, l'interpréteur pour le langage AL, le reste du système IBC-Go et les modifications apportées au système ANTS.

a. L'interface graphique

```
import java.applet.Applet;
import java.awt.*;
import java.io.*;
import java.net.*;

public class Main extends Applet {

        TextArea monitor;
        ClientConnexion clientConnexion = null;

        //Lorsque l'applet est instanciée suite
        public void init() {
                monitor = new TextArea(5, 20);
                monitor.setEditable(false);
                monitor.setBackground(Color.white);
                GridBagLayout gridBag = new GridBagLayout();
                setLayout(gridBag);
                GridBagConstraints c = new GridBagConstraints();
                c.fill = GridBagConstraints.BOTH;
                c.weightx = 1.0;
                c.weighty = 1.0;
                gridBag.setConstraints(monitor, c);
                add(monitor);

                clientConnexion = new ClientConnexion(getParameter("address"),
                Integer.parseInt(getParameter("port")), getParameter("user"), getParameter("pass"),
                getParameter("addressants"), monitor, getAppletContext());
                clientConnexion.start();
                }


        //Pour faire un graphique
        public void paint(Graphics Element) {
```

```
            }

        //Lorsque l'applet est quittée
        public void destroy() {}

        private Color stringToColor(String paramValue) {
                int red;
                int green;
                int blue;

                red = (Integer.decode("0x" + paramValue.substring(0,2))).intValue();
                green = (Integer.decode("0x" + paramValue.substring(2,4))).intValue();
                blue = (Integer.decode("0x" + paramValue.substring(4,6))).intValue();

                return new Color(red,green,blue);
                }

        public void send(String s) {
                clientConnexion.send(s);
                }

        }


import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;

class ClientConnexion extends java.lang.Thread {

        private String host = null;
        private int port = 0;
        private String user = null;
        private String pass = null;
        private String ants = null;
        private TextArea monitor = null;
        private AppletContext appletContext = null;
        private SecretKey key = null;
        private Cipher cipher = null;
        private DataOutputStream out = null;


        ClientConnexion(String host, int port, String user, String pass, String ants, TextArea monitor,
            AppletContext appletContext) {
                this.host = host;
                this.port = port;
                this.user = user;
                this.pass = pass;
                this.ants = ants;
                this.monitor = monitor;
                this.appletContext = appletContext;
```

```
        }

public void run() {
        try {
                Socket socket = new Socket(host, port);
                DataInputStream in =
                  new DataInputStream(new
                BufferedInputStream(socket.getInputStream()));
                out = new DataOutputStream(socket.getOutputStream());

                Security.addProvider(new com.sun.crypto.provider.SunJCE());
                KeyPairGenerator kpg = KeyPairGenerator.getInstance("DH");
                kpg.initialize(Skip.sDHParameterSpec);
                KeyPair keypair = kpg.genKeyPair();

                byte[] keyBytes = keypair.getPublic().getEncoded();
                out.writeInt(keyBytes.length);
                out.write(keyBytes);

                keyBytes = new byte[in.readInt()];
                in.readFully(keyBytes);
                KeyFactory kf = KeyFactory.getInstance("DH");
                X509EncodedKeySpec x509Spec = new X509EncodedKeySpec(keyBytes);
                PublicKey theirPublicKey = kf.generatePublic(x509Spec);
                KeyAgreement ka = KeyAgreement.getInstance("DH");
                ka.init(keypair.getPrivate());
                ka.doPhase(theirPublicKey, true);
                byte[] secret = ka.generateSecret();

                SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
                DESKeySpec desSpec = new DESKeySpec(secret);
                key = skf.generateSecret(desSpec);
                cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

                cipher.init(Cipher.ENCRYPT_MODE, key);
                byte[] stringBytes = user.getBytes("UTF8");
                byte[] raw = cipher.doFinal(stringBytes);
                out.writeInt(raw.length);
                out.write(raw);
                stringBytes = pass.getBytes("UTF8");
                raw = cipher.doFinal(stringBytes);
                out.writeInt(raw.length);
                out.write(raw);

                Cipher cipherListen = Cipher.getInstance("DES/ECB/PKCS5Padding");
                cipherListen.init(Cipher.DECRYPT_MODE, key);
                while(true) {
                        raw = new byte[in.readInt()];
                        in.readFully(raw);
                        stringBytes = cipherListen.doFinal(raw);
                        String message = new String(stringBytes, "UTF8");
                        System.out.println(message);
                        if(message.equals("____refresh")) {
                                try {
```

```
                                                    URL pageURL = new ↓
URL("http://192.168.1.100:8080/IBCGoWeb/agent_refresh.jsp?"+ants);

        appletContext.showDocument(pageURL,"agent");
                                                    }
                                                catch(MalformedURLException mue) {}
                                                }
                                        else
                                                monitor.append(message);
                                        }
                                }
                        catch (Exception e) {}
                        }

        public void send(String s) {
                try {
                        cipher.init(Cipher.ENCRYPT_MODE, key);
                        byte[] stringBytes = s.getBytes("UTF8");
                        byte[] raw = cipher.doFinal(stringBytes);
                        out.writeInt(raw.length);
                        out.write(raw);
                        }
                catch(Exception e) {}
                }
        }


import java.math.BigInteger;
import javax.crypto.spec.*;

public class Skip {
        // SKIP's 1024 DH parameters
        private static final String skip1024String =
          "F488FD584E49DBCD20B49DE49107366B336C380D451D0F7C88B31C7C5B2D8EF6" +
          "F3C923C043F0A55B188D8EBB558CB85D38D334FD7C175743A31D186CDE33212C" +
          "B52AFF3CE1B1294018118D7C84A70A72D686C40319C807297ACA950CD9969FAB" +
          "D00A509B0246D3083D66A45D419F9C7CBD894B221926BAABA25EC355E92F78C7";

        // Modulus
        private static final BigInteger skip1024Modulus = new BigInteger(skip1024String, 16);

        // Base
        private static final BigInteger skip1024Base = BigInteger.valueOf(2);

        public static final DHParameterSpec sDHParameterSpec =
                new DHParameterSpec(skip1024Modulus, skip1024Base);
        }


package ActionLib;

import FormLib.AgentAddForm;
import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;
import java.io.*;

public class AgentAddAction extends Action {

        private static int fromString(String name) {
                StringTokenizer st = new StringTokenizer(name, ".");

                int[] b = new int[4];
                try {
                        for (int i = 0; i < 4; i++)
                        b[i] = Integer.parseInt(st.nextToken());
                        }
                catch (Exception e) {}

                return ((b[0]<<24) | (b[1]<<16) | (b[2]<<8) | b[3]);
                }

        public ActionForward execute(
                                        ActionMapping mapping,
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
          throws IOException, ServletException {

                AgentAddForm agentAddForm = (AgentAddForm)form;

                Agent agent = new Agent();
                agent.setIdentifier(agentAddForm.getIdentifier());
                agent.setPTime(DateTime.getLong());
                agent.setType(1);
                agent.setCredentials(0);

                Vector vectorCodes = new Vector();
                int nbCode = 0;
                String element;
                for(
                        StringTokenizer stringtokenizer =
                          new StringTokenizer(agentAddForm.getCode(),
                          new Character((char)13).toString(), false);
                        stringtokenizer.hasMoreTokens();
                        vectorCodes.addElement(element.trim())
                        ) {
                                element = stringtokenizer.nextToken();
                                nbCode++;
                                }
                nbCode = nbCode / 2;
```

```
int i = 0;
agent.setCodeType(nbCode);
agent.setCodeLength(nbCode);
agent.setCodeData(nbCode);
agent.setCodeSigner(nbCode);
agent.setCodeSignature(nbCode);
agent.setInterpreterAddress(nbCode);
Enumeration enumerationCodes = vectorCodes.elements();
while(enumerationCodes.hasMoreElements()) {
        element = (String)enumerationCodes.nextElement();
        long interpreter;
        String interpreterString = (String)enumerationCodes.nextElement();
        interpreter = (fromString(interpreterString.substring(0,
          interpreterString.lastIndexOf(":")))*100000)+

Integer.parseInt(interpreterString.substring(interpreterString.lastIndexOf(":")+1));
        agent.setInterpreterAddress(i, interpreter);
        if(!element.equals("")) {
                agent.setCodeType(i, 0);
                agent.setCodeData(i, element.getBytes("UTF8"));
                agent.setCodeSigner(i, agentAddForm.getSigner());
                i++;
                }
          else nbCode--;
        }
agent.setNbCode(nbCode);

Vector vectorItineraries = new Vector();
int nbItinerary = 0;
for(
        StringTokenizer stringtokenizer = new
        StringTokenizer(agentAddForm.getItinerary(),
        new Character((char)13).toString(), false);
        stringtokenizer.hasMoreTokens();
        vectorItineraries.addElement(element.trim())
        ) {
                element = stringtokenizer.nextToken();
                nbItinerary++;
                }
nbItinerary = nbItinerary / 2;

i = 0;
String itineraries = "";
agent.setK(nbItinerary);
agent.setSite(nbItinerary);
Enumeration enumerationItineraries = vectorItineraries.elements();
while(enumerationItineraries.hasMoreElements()) {
        element = (String)enumerationItineraries.nextElement();
        if(!element.equals("")) {
                agent.setK(i, Integer.parseInt((String)
                enumerationItineraries.nextElement()));
                agent.setSite(i, fromString(element));
                itineraries += element;
                i++;
                }
          else nbItinerary--;
```

```
                }
        agent.setItiSigner(agentAddForm.getSigner()+":"+agentAddForm.getSignerpass());
        agent.setNbItinerary(nbItinerary);

        Vector vectorStrings = new Vector();
        String string;
        for(
                StringTokenizer stringtokenizer = new
                StringTokenizer(agentAddForm.getPayload(), "\n", false);
                stringtokenizer.hasMoreTokens();
                vectorStrings.addElement(string)
                )
                        string = stringtokenizer.nextToken();
        String[] stringArray = new String[vectorStrings.size()];
        Enumeration enumerationStrings = vectorStrings.elements();
        i = 0;
        while(enumerationStrings.hasMoreElements())
                stringArray[i] = (String)enumerationStrings.nextElement();
        agent.setPayload(stringArray);

        Verifier verifier = new Verifier();
        boolean ok = verifier.createAgent(agentAddForm.getAddress(),
                        agentAddForm.getIdentifier(), agent);

        String out = agentAddForm.getIdentifier() + " added.<br>";

        request.setAttribute("content", out);
        return (mapping.findForward(new String("target")));
        }

}
```

```
package ActionLib;

import FormLib.AgentContentForm;
import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class AgentContentAction extends Action {

        public ActionForward execute(
                                        ActionMapping mapping,
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
          throws IOException, ServletException {
                String out = "";
```

```
AgentContentForm agentContentForm = (AgentContentForm)form;
String address = agentContentForm.getAddress();
String agentId = agentContentForm.getAgent();

request.setAttribute("paramRemove", "address="+address+"&agent="+agentId);
request.setAttribute("paramSend", "send:"+agentId);

//price = getQuote(symbol);
Directory dir = new Directory();
Agent agent = dir.getAgentContent(address, agentId);

out = "<table>";
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}>
        <b>Identifier:</b> </font></td>";
out += "<td><font STYLE={font-family:Arial;font-size:12px;}>" +
        agent.getIdentifier() + "</font></td></tr>";
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}>
        <b>Time of creation:</b> </font></td>";
out += "<td><font STYLE={font-family:Arial;font-size:12px;}>" +
        DateTime.longToString(agent.getPTime()) + "</font></td></tr>";
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}>
        <b>Origin:</b> </font></td>";
out += "<td><font STYLE={font-family:Arial;font-size:12px;}>" +
        NodeToString(agent.getOrigin()) + "</font></td></tr>";
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}>
        <b>Destination:</b> </font></td>";
out += "<td><font STYLE={font-family:Arial;font-size:12px;}>" +
        NodeToString(agent.getDestination()) + "</font></td></tr>";
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}>
        <b>Creator:</b> </font></td>";
out += "<td><font STYLE={font-family:Arial;font-size:12px;}>" +
        agent.getCreator() + "</font></td></tr>";
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}><b>Type:</b>
        </font></td>";
out += "<td><font STYLE={font-family:Arial;font-size:12px;}>" +
        Integer.toString(agent.getType()) + "</font></td></tr>";
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}><b>Credentials:</b>
        </font></td>";
out += "<td><font STYLE={font-family:Arial;font-size:12px;}>" +
        Credentials.toString(agent.getCredentials()) + " ("+agent.getCreSigner()+")
        </font></td></tr>";
for(int i=1;i<=agent.getNbCode();i++) {
        try {
                out += "<tr><td><font STYLE={font-family:Arial;font-
                        size:12px;}><b>Code "+Integer.toString(i)+":</b>
                        </font></td><td>
                        <font STYLE={font-family:Arial;font-size:12px;}>";
                out += new String(agent.getCodeData(i-1), "UTF8") + "
                        ("+Integer.toString(agent.getCodeType(i-1))+",
                        "+Integer.toString(agent.getCodeLength(i-1))+",
                        "+agent.getCodeSigner(i-1)+")</font></td></tr>";
                }
        catch(Exception e) {}
        }
out += "<tr><td><font STYLE={font-family:Arial;font-size:12px;}><b>Itinerary:</b>
```

```
                    </font></td><td><font STYLE={font-family:Arial;font-size:12px;}>[";
                for(int i=1;i<=agent.getNbItinerary();i++) {
                    out += "("+Integer.toString(agent.getK(i-1))+", "+
                        NodeToString(agent.getSite(i-1))+")";
                    if(i<agent.getNbItinerary()) out += ", ";
                    }
                if(agent.getItiSigner().indexOf(":") > 0)
                    out += ",
                        "+agent.getItiSigner().substring(0,agent.getItiSigner().indexOf(":"))+
                        "]</font></td></tr>";
                else
                    out += ", "+agent.getItiSigner()+"]</font></td></tr>";
                out += "</table>";

                request.setAttribute("content", out);
                return (mapping.findForward(new String("target")));
                }

        public static String NodeToString(int a) {
                String dotted = Integer.toString(a & 0xff);
                dotted = Integer.toString((a >> 8) & 0xff) + "." + dotted;
                dotted = Integer.toString((a >> 16) & 0xff) + "." + dotted;
                dotted = Integer.toString((a >> 24) & 0xff) + "." + dotted;
                if(dotted.equals("0.0.0.0")) dotted = "";
                return dotted;
                }

        }


package ActionLib;

import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;

public class AgentNewAction extends Action {

        public ActionForward execute(
                                        ActionMapping mapping,
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
          throws IOException, ServletException {

                return (mapping.findForward(new String("target")));
                }
```

```
        }


package ActionLib;

import FormLib.AgentRemoveForm;
import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;

public class AgentRemoveAction extends Action {

        public ActionForward execute(
                                        ActionMapping mapping,
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
          throws IOException, ServletException {
                AgentRemoveForm agentRemoveForm = (AgentRemoveForm)form;
                Verifier verifier = new Verifier();
                boolean ok =
                   verifier.removeAgent(agentRemoveForm.getAddress(),
                agentRemoveForm.getAgent());
                String out = agentRemoveForm.getAgent() + " removed.<br>";
                request.setAttribute("content", out);
                return (mapping.findForward(new String("target")));
                }
        }


package ActionLib;

import FormLib.CodeAddForm;
import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;

public class CodeAddAction extends Action {

        public ActionForward execute(
                                        ActionMapping mapping,
```

```
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
        throws IOException, ServletException {
                CodeAddForm codeAddForm = (CodeAddForm)form;
                Verifier verifier = new Verifier();
                int ok = verifier.createCode("code", codeAddForm.getAddress(),
                        codeAddForm.getName(), codeAddForm.getCode(), false);
                String out = codeAddForm.getName() + " added.<br>";
                request.setAttribute("content", out);
                return (mapping.findForward(new String("target")));
                }
        }


package ActionLib;

import FormLib.CodeContentForm;
import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;

public class CodeContentAction extends Action {

        public ActionForward execute(
                                ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response
                                )
        throws IOException, ServletException {
                String out = "";

                CodeContentForm codeContentForm = (CodeContentForm)form;
                String port = codeContentForm.getAddress();
                String codeId = codeContentForm.getCode();
                codeId = codeId.replace('-', '=');

                Directory dir = new Directory();
                CodeUnit code = dir.getCode(port, codeId);

                try {
                        out = new String(code.getCode(), "UTF8");
                        }
                catch(Exception e) {}

                //out = out.replaceAll(Character.toString((char)13),"<br>");
                //out = out.replaceAll(Character.toString((char)9),"   ");
```

```
                out = replaceAll(out, new Character((char)13).toString(), "<br>");
                out = replaceAll(out, new Character((char)9).toString(), "   ");

                request.setAttribute("content", out);
                request.setAttribute("code", codeContentForm.getCode());
                return (mapping.findForward(new String("target")));
                }

        // Websphere n'accepte pas replaceAll
        private String replaceAll(String s1, String s2, String s3) {
                Vector vectorLines = new Vector();
                String line;
                for(
                        StringTokenizer stringtokenizer = new StringTokenizer(s1, s2, false);
                        stringtokenizer.hasMoreTokens();
                        vectorLines.addElement(line+s3)
                        )
                        line = stringtokenizer.nextToken();
                String out = "";
                Enumeration enumerationLines = vectorLines.elements();
                while(enumerationLines.hasMoreElements())
                        out += (String)enumerationLines.nextElement();
                return out;
                }
        }


package ActionLib;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;

public class CodeNewAction extends Action {

        public ActionForward execute(
                                        ActionMapping mapping,
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
          throws IOException, ServletException {
                return (mapping.findForward(new String("target")));
                }
        }


package ActionLib;

import FormLib.CodeRemoveForm;
```

```
import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;

public class CodeRemoveAction extends Action {

        public ActionForward execute(
                                        ActionMapping mapping,
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
          throws IOException, ServletException {

                CodeRemoveForm codeRemoveForm = (CodeRemoveForm)form;
                String code = codeRemoveForm.getCode();

                Vector vectorObjects = new Vector();
                String element;
                for(
                                StringTokenizer stringtokenizer=new StringTokenizer(code,",",
                                        false);
                                stringtokenizer.hasMoreTokens();
                                vectorObjects.addElement(element)
                                )
                                        element = stringtokenizer.nextToken();

                code = "";
                Enumeration enumerationObjects = vectorObjects.elements();
                while(enumerationObjects.hasMoreElements()) {
                        element = (String)enumerationObjects.nextElement();
                        if(code.equals(""))
                                code = element.substring(element.indexOf('-')+1);
                         else
                                code = element.substring(element.indexOf('-')+1) + "." + code;
                        }

                Verifier verifier = new Verifier();
                boolean ok = verifier.removeCode(codeRemoveForm.getAddress(), code);

                String out = code + " removed.<br>";

                request.setAttribute("content", out);
                return (mapping.findForward(new String("target")));
                }
        }


package ActionLib;
```

```
import FormLib.LoginForm;
import IBCGo.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import java.util.*;
import java.io.*;
import java.security.*;

public class LoginAction extends Action {

        private static int fromString(String name) {
                StringTokenizer st = new StringTokenizer(name, ".");

                int[] b = new int[4];
                try {
                        for (int i = 0; i < 4; i++)
                        b[i] = Integer.parseInt(st.nextToken());
                        }
                catch (Exception e) {}

                return ((b[0]<<24) | (b[1]<<16) | (b[2]<<8) | b[3]);
                }


        public ActionForward execute(
                                        ActionMapping mapping,
                                        ActionForm form,
                                        HttpServletRequest request,
                                        HttpServletResponse response
                                        )
          throws IOException, ServletException {

                LoginForm loginForm = (LoginForm)form;

                HttpSession session = request.getSession();
                session.setAttribute("user", loginForm.getUser());
                session.setAttribute("pass", loginForm.getPass());
                session.setAttribute("addressip", loginForm.getAddressip());
                session.setAttribute("addressants", loginForm.getAddressants());
                session.setAttribute("port", loginForm.getPort());

                String out = "";

                request.setAttribute("content", out);
                return (mapping.findForward(new String("target")));
                }
        }
```

```
package FormLib;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class AgentAddForm extends ActionForm {

        // Getters & Setters
        private String address = null;
        private String identifier = null;
        private String credentials = null;
        private String code = null;
        private String itinerary = null;
        private String signer = null;
        private String signerpass = null;
        private String keystore = null;
        private String storepass = null;
        private String payload = null;

        public String getAddress() { return this.address; }
        public void setAddress(String address) { this.address = address; }
        public String getIdentifier() { return this.identifier; }
        public void setIdentifier(String identifier) { this.identifier = identifier; }
        public String getCredentials() { return this.credentials; }
        public void setCredentials(String credentials) { this.credentials = credentials; }
        public String getCode() { return this.code; }
        public void setCode(String code) { this.code = code; }
        public String getItinerary() { return this.itinerary; }
        public void setItinerary(String itinerary) { this.itinerary = itinerary; }
        public String getSigner() { return this.signer; }
        public void setSigner(String signer) { this.signer = signer; }
        public String getSignerpass() { return this.signerpass; }
        public void setSignerpass(String signerpass) { this.signerpass = signerpass; }
        public String getKeystore() { return this.keystore; }
        public void setKeystore(String keystore) { this.keystore = keystore; }
        public String getStorepass() { return this.storepass; }
        public void setStorepass(String storepass) { this.storepass = storepass; }
        public String getPayload() { return this.payload; }
        public void setPayload(String payload) { this.payload = payload; }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.address = null;
                this.identifier = null;
                this.credentials = null;
                this.code = null;
                this.itinerary = null;
                this.signer = null;
                this.signerpass = null;
                this.keystore = null;
                this.storepass = null;
                this.payload = null;
                }
}
```

```
package FormLib;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class AgentContentForm extends ActionForm {
        // Getters & Setters
        private String address = null;
        private String agent = null;

        public String getAddress() { return this.address; }
        public void setAddress(String address) { this.address = address; }
        public String getAgent() { return this.agent; }
        public void setAgent(String agent) { this.agent = agent; }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.address = null;
                this.agent = null;
                }
}


package FormLib;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class AgentRemoveForm extends ActionForm {
        // Getters & Setters
        private String address = null;
        private String agent = null;

        public String getAddress() { return this.address; }
        public void setAddress(String address) { this.address = address; }
        public String getAgent() { return this.agent; }
        public void setAgent(String agent) { this.agent = agent; }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.address = null;
                this.agent = null;
                }
}


package FormLib;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class CodeAddForm extends ActionForm {
        // Getters & Setters
        private String address = null;
```

```
        private String name = null;
        private String code = null;

        public String getAddress() { return this.address; }
        public void setAddress(String address) { this.address = address; }
        public String getName() { return this.name; }
        public void setName(String name) { this.name = name; }
        public String getCode() { return this.code; }
        public void setCode(String code) { this.code = code; }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.address = null;
                this.name = null;
                this.code = null;
                }

}


package FormLib;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class CodeContentForm extends ActionForm {
        // Getters & Setters
        private String address = null;
        private String code = null;

        public String getAddress() { return this.address; }
        public void setAddress(String address) { this.address = address; }
        public String getCode() { return this.code; }
        public void setCode(String agent) { this.code = agent; }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.address = null;
                this.code = null;
                }
}


package FormLib;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class CodeRemoveForm extends ActionForm {
        // Getters & Setters
        private String address = null;
        private String code = null;

        public String getAddress() { return this.address; }
        public void setAddress(String address) { this.address = address; }
        public String getCode() { return this.code; }
```

```
        public void setCode(String agent) { this.code = agent; }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.address = null;
                this.code = null;
                }
}


package FormLib;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class LoginForm extends ActionForm {
        // Getters & Setters
        private String user = null;
        private String pass = null;
        private String addressip = null;
        private String addressants = null;
        private String port = null;

        public String getUser() { return this.user; }
        public void setUser(String user) { this.user = user; }
        public String getPass() { return this.pass; }
        public void setPass(String pass) { this.pass = pass; }
        public String getAddressip() { return this.addressip; }
        public void setAddressip(String address) { this.addressip = address; }
        public String getAddressants() { return this.addressants; }
        public void setAddressants(String address) { this.addressants = address; }
        public String getPort() { return this.port; }
        public void setPort(String port) { this.port = port; }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.user = null;
                this.pass = null;
                this.addressip = null;
                this.addressants = null;
                this.port = null;
                }
}


package TagLib;

import IBCGo.*;
import java.io.IOException;
import java.io.PrintStream;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.PageContext;
import javax.servlet.http.HttpSession;
import javax.servlet.jsp.tagext.Tag;
import javax.servlet.jsp.tagext.TagSupport;
import java.util.*;
```

```
public class AgentTag extends TagSupport {

        /* Properties */
        private String address = null;

        /* Getter and Setter Methods */
        public String getAddress () { return this.address; }
        public void setAddress (String address) { this.address = address; }

        public int doStartTag () {
                try {
                        JspWriter out = pageContext.getOut();

                        HttpSession session = pageContext.getSession();
                        this.address = (String)session.getAttribute("addressants");

                        Directory dir = new Directory();
                        Vector agents = dir.getAgentList(this.address);
                        Enumeration enumerationAgents = agents.elements();
                        while(enumerationAgents.hasMoreElements()) {
                                String name = (String)enumerationAgents.nextElement();
                                out.print("<LI><A HREF=AgentContent.do?address="+
                                        this.address +
                                        "&agent=" + name + " target=body>" + name +
                                "</A></LI>");
                                }
                        out.print("</UL>");
                        }
                catch(Exception e) {}

                return SKIP_BODY;
                }
        }


package TagLib;

import IBCGo.*;
import java.io.IOException;
import java.io.PrintStream;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.PageContext;
import javax.servlet.http.HttpSession;
import javax.servlet.jsp.tagext.Tag;
import javax.servlet.jsp.tagext.TagSupport;
import java.util.*;

public class CodeTag extends TagSupport {

        /* Properties */
        private String address = null;

        /* Getter and Setter Methods */
        public String getAddress () { return this.address; }
        public void setAddress (String address) { this.address = address; }
```

```
public int doStartTag () {
        try {
                JspWriter out = pageContext.getOut();

                HttpSession session = pageContext.getSession();
                this.address = (String)session.getAttribute("addressants");

                out.print("<script type=text/javascript>");
                out.print("var Tree = new Array;");

                Directory dir = new Directory();
                Vector codes = dir.getCodeList(this.address);
                Enumeration enumerationCodes = codes.elements();
                int node = 0;
                int[] parent = new int[100];
                int pastLevel = 0;
                Vector path = new Vector();
                while(enumerationCodes.hasMoreElements()) {
                        String element = (String)enumerationCodes.nextElement();
                        String type = element.substring(0, element.indexOf('='));
                        String name = element.substring(element.indexOf('=')+1);
                        int level =
                          Integer.parseInt((String)enumerationCodes.nextElement());

                        node += 1;

                        for(int i=level; i<pastLevel; i++) path.remove(path.lastElement());

                        if(type.equals("OU")) {
                                // nodeId | parentNodeId | nodeName | nodeUrl
                                out.print("Tree["+(node-1)+"] =
                                        '"+node+"|"+parent[level]+"|"+name+"|#';");
                                path.addElement(name);
                                parent[level+1] = node;
                                }
                          else {
                                String pathString = "";
                                Enumeration enumerationPath = path.elements();
                                while(enumerationPath.hasMoreElements())
                                        pathString = ", OU-

                                        "+(String)enumerationPath.nextElement()+
                                                pathString;
                                pathString = "CN-"+name + pathString;
                                out.print("Tree["+(node-1)+"] =
                                        '"+node+"|"+parent[level]+"|"+name+
                                        "|CodeContent.do?address="+this.address+
                                        "&code="+pathString+"';");
                                }
                        pastLevel = level;
                        }

                out.print("</script>");
                }
        catch(Exception e) {}
```

```
                return SKIP_BODY;
                }
        }


a               { color: #00FF; }
a:hover  { color: #CCCCCC; text-decoration: none; }
a:visited { text-decoration: none; }
a:link          { text-decoration: none; }
UL              { margin-bottom: 0px; margin-top: 0px; margin-left: 1em; list-style-type: square; }
LI              { margin-bottom: 0px; margin-top: 0px; margin-left: 1em; list-style-type: square; }
.tree           {
                position: absolute;
                top: 17px;
                left: 8px;
                font-family: Arial, Verdana, Geneva, Helvetica, sans-serif;
                font-size: 12px;
                padding: 10px;
                white-space: nowrap;
                }
.tree img       { border: 0px; height: 18px; vertical-align: text-bottom; }
.tree a    { color: #00FF; text-decoration: none; }
.tree a:hover   { color: #CCCCCC; text-decoration: none; }
.tree a:visited { text-decoration: none; }
.tree a:link    { text-decoration: none; }


var nodes = new Array();;
var openNodes = new Array();
var icons = new Array(8);

function preloadIcons() {
        icons[0] = new Image();
        icons[0].src = "images/tree/plus.gif";
        icons[1] = new Image();
        icons[1].src = "images/tree/plusbottom.gif";
        icons[2] = new Image();
        icons[2].src = "images/tree/plustop.gif";
        icons[3] = new Image();
        icons[3].src = "images/tree/plusalone.gif";
        icons[4] = new Image();
        icons[4].src = "images/tree/minus.gif";
        icons[5] = new Image();
        icons[5].src = "images/tree/minusbottom.gif";
        icons[6] = new Image();
        icons[6].src = "images/tree/minustop.gif";
        icons[7] = new Image();
        icons[7].src = "images/tree/minusalone.gif";
        }

function createTree(arrName) {
        nodes = arrName;
        if (nodes.length > 0) {
                preloadIcons();
                var recursedNodes = new Array();
                addNode(0, recursedNodes);
```

```
                }
        }

function getArrayId(node) {
        for (i=0; i<nodes.length; i++) {
                var nodeValues = nodes[i].split("|");
                if (nodeValues[0]==node) return i;
                }
        }

function setOpenNodes(openNode) {
        for (i=0; i<nodes.length; i++) {
                var nodeValues = nodes[i].split("|");
                if (nodeValues[0]==openNode) {
                        openNodes.push(nodeValues[0]);
                        setOpenNodes(nodeValues[1]);
                        }
                }
        }

function isNodeOpen(node) {
        for (i=0; i<openNodes.length; i++)
                if (openNodes[i]==node) return true;
        return false;
        }

function hasChildNode(parentNode) {
        for (i=0; i< nodes.length; i++) {
                var nodeValues = nodes[i].split("|");
                if (nodeValues[1]==parentNode) return true;
                }
        return false;
        }

function lastSibling (node, parentNode) {
        var lastChild = 0;
        for (i=0; i< nodes.length; i++) {
                var nodeValues = nodes[i].split("|");
                if (nodeValues[1]==parentNode) lastChild = nodeValues[0];
                }
        if (lastChild == node) return true;
        return false;
        }

function firstNode (node, parentNode) {
        if(parentNode>0) return false;
        for (i=0; i<nodes.length; i++) {
                var nodeValues = nodes[i].split("|");
                if (nodeValues[1] == parentNode)
                        if (nodeValues[0]==node)
                                return true;
                        else
                                return false;
                }
        return false;
        }
```

```
function addNode(parentNode, recursedNodes) {
        for (var i = 0; i < nodes.length; i++) {
                var nodeValues = nodes[i].split("|");
                if (nodeValues[1] == parentNode) {

                        var fr = firstNode(nodeValues[0], nodeValues[1]);
                        var ls = lastSibling(nodeValues[0], nodeValues[1]);
                        var hcn = hasChildNode(nodeValues[0]);
                        var ino = isNodeOpen(nodeValues[0]);

                        // Write out line & empty icons
                        for (g=0; g<recursedNodes.length; g++) {
                                if (recursedNodes[g] == 1)
                                  document.write
                                    ("<img src=\"images/tree/line.gif\" align=\"absbottom\" alt=\"\"
                                />");
                                else
                                  document.write
                                    ("<img src=\"images/tree/empty.gif\" align=\"absbottom\" alt=\"\"
                                />");
                                }

                        // put in array line & empty icons
                        if (ls)
                        recursedNodes.push(0);
                           else
                        recursedNodes.push(1);

                        // Write out join icons
                        if (hcn) {
                        if (fr && ls) {
                                document.write
                                  ("<a href=\"javascript: oc(" + nodeValues[0] + ", 0);\">
                                  <img id=\"join" + nodeValues[0] + "\" src=\"images/tree/");
                                if (ino)
                                        document.write("minus");
                                else
                                        document.write("plus");
                                        document.write
                                          ("alone.gif\" align=\"absbottom\"
                                          alt=\"Open/Close node\" /></a>");
                                }
                        else if (fr) {
                                document.write
                                  ("<a href=\"javascript: oc(" + nodeValues[0] + ", 1);\">
                                  <img id=\"join" + nodeValues[0] + "\" src=\"images/tree/");
                                if (ino)
                                        document.write("minus");
                                else
                                        document.write("plus");
                                        document.write
                                          ("top.gif\" align=\"absbottom\"
                                          alt=\"Open/Close node\" /></a>");
                                }
                        else if (ls) {
```

```
                        document.write
                          ("<a href=\"javascript: oc(" + nodeValues[0] + ", 2);\">
                          <img id=\"join" + nodeValues[0] + "\" src=\"images/tree/");
                        if (ino)
                                document.write("minus");
                        else
                                document.write("plus");
                                document.write
                                  ("bottom.gif\" align=\"absbottom\"
                                  alt=\"Open/Close node\" /></a>");
                        }
                else {
                        document.write
                          ("<a href=\"javascript: oc(" + nodeValues[0] + ", 3);\">
                          <img id=\"join" + nodeValues[0] + "\" src=\"images/tree/");
                        if (ino)
                                document.write("minus");
                        else
                                document.write("plus");
                                document.write
                                  (".gif\" align=\"absbottom\" alt=\"Open/Close node\"
                        /></a>");
                        }
                }
        else {
                if (fr && ls)
                        document.write
                          ("<img src=\"images/tree/empty.gif\" align=\"absbottom\" alt=\"\"
                />");
                else if (fr)
                        document.write
                          ("<img src=\"images/tree/jointop.gif\" align=\"absbottom\" alt=\"\"
                />");
                else if (ls)
                        document.write
                          ("<img src=\"images/tree/joinbottom.gif\"
                          align=\"absbottom\" alt=\"\" />");
                else
                        document.write
                          ("<img src=\"images/tree/join.gif\" align=\"absbottom\" alt=\"\" />");
                }

if(nodeValues[3] != "#")
        document.write("<a href=\"" + nodeValues[3] + "\" target=body
          onmouseover=\"window.status='" + nodeValues[2] + "';return true;\"
          onmouseout=\"window.status=' ';return true;\">");

document.write(nodeValues[2]);

if(nodeValues[3] != "#") document.write("</a>");

        document.write("<br />");

                // If node has children write out divs and go deeper
                if (hcn) {
                        document.write("<div id=\"div" + nodeValues[0] + "\"");
```

```
                                        if (!ino) document.write(" style=\"display: none;\"");
                                        document.write(">");
                                        addNode(nodeValues[0], recursedNodes);
                                        document.write("</div>");
                                        }

                                // remove last line or empty icon
                                recursedNodes.pop();
                                }
                        }
        }

function oc(node, pos) {
        var theDiv = document.getElementById("div" + node);
        var theJoin = document.getElementById("join" + node);

        if (theDiv.style.display == 'none') {
                if (pos==0) theJoin.src = icons[7].src;
                else if (pos==1) theJoin.src = icons[6].src;
                else if (pos==2) theJoin.src = icons[5].src;
                else theJoin.src = icons[4].src;
                theDiv.style.display = '';
                }
        else {
                if (pos==0) theJoin.src = icons[3].src;
                else if (pos==1) theJoin.src = icons[2].src;
                else if (pos==2) theJoin.src = icons[1].src;
                else theJoin.src = icons[0].src;
                theDiv.style.display = 'none';
                }
        }


if(!Array.prototype.push) {
        function array_push() {
                for(var i=0;i<arguments.length;i++) this[this.length]=arguments[i];
                return this.length;
                }
        Array.prototype.push = array_push;
        }

if(!Array.prototype.pop) {
        function array_pop() {
                lastElement = this[this.length-1];
                this.length = Math.max(this.length-1,0);
                return lastElement;
                }
        Array.prototype.pop = array_pop;
        }


<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/ibcgo.tld" prefix="ibcgo" %>

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
```

```
<html>

 <head>

  <link rel="stylesheet" type="text/css" href="styles/main.css">
 </head>

 <body link="blue" vlink="blue" alink="blue">

  <font style="{position:absolute;top:4px;left:0px;font-family:Arial;font-size:14px;}">
        <b>  AGENT:</b>
          
        <a href="javascript:window.location.reload()">[refresh]</a>
  </font>

  <%-- Inialisation de la composante Web --%>
  <%-- Version Bean
   <% String address = (String)session.getAttribute("addressants"); %>
   <jsp:useBean id="agent" class="IBCGo.IBCWeb">
    <jsp:setProperty name="agent" property="port" value="<%=address%>" />
   </jsp:useBean>
  --%>

  <font style="{position:absolute;top:24px;left:14px;font-family:Arial;font-size:12px;}">

   <%-- Saisie de la liste d'agents --%>
   <%-- Version Bean
    <jsp:getProperty name="agent" property="agentList" />
   --%>

   <%-- Version Tag --%>
   <ibcgo:agentList />

  </font>

 </body>

</html>


<html>

 <head>

 </head>

 <body topmargin="2" leftmargin="5">

  <%-- Saisie de la liste d'agents --%>
  <%-- Version Bean
  <% String port = (String)session.getAttribute("port"); %>
  <jsp:useBean id="agent" class="IBCGo.IBCWeb">
   <jsp:setProperty name="agent" property="port" value="<%=port%>" />
   <jsp:setProperty name="agent" property="agent" value='<%=request.getParameter("agent")%>' />
  </jsp:useBean>
```

```
--%>
<font STYLE="{font-family:Arial;font-size:12px;}">
 <%-- Version Bean
 <jsp:getProperty name="agent" property="agentContent" />
 --%>
 <%-- Version Struts --%>
 <%= request.getAttribute("content")%>
</font>

<br>

        <script language="javascript">
        function pausecomp(Amount) {
                d = new Date() //today's date
                while (1) {
                        mill=new Date() // Date Now
                        diff = mill-d //difference in milliseconds
                        if( diff > Amount ) {break;}
                        }
                }

        function send() {
                parent.console.console.send("<%=(String)request.getAttribute("paramSend")%>;"+
                                document.form.keep.checked);
                document.write("Agent sent.");
                }
        </script>

<form name="form">
 <input type="button" value="Remove"
        onClick="javascript:location.href('RemoveAgent.do?<%=request.getAttribute("paramRemove")
        %>');
  "><br><br>
 <input type="button" value="Activate" onClick="javascript:send();">
 <input type="checkbox" name="keep" checked>Keep</input>
</form>

</body>

</html>


<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

<html:html locale="true">

<head>
 <html:base/>
</head>

<body topmargin="3" leftmargin="5">

 <font STYLE="{font-family:sans-serif;font-size:12px;}">
  <b>NEW AGENT</b>
```

```
  </font>

  <% String address = (String)session.getAttribute("addressants"); %>

  <html:form action="/AddAgent.do">
   <html:hidden property="address" value="<%=address%>" />
   <table>
    <tr>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Identifier: </font></td>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:text property="identifier"
size="15"
                maxlength="30"/></font></td>
    </tr>
    <tr valign="top">
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Code: </font></td>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:textarea property="code"
cols="40"
                rows="5"/></font></td>
    </tr>
    <tr valign="top">
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Itinerary: </font></td>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:textarea property="itinerary"
                cols="40" rows="5"/></font></td>
    </tr>
    <tr valign="top">
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Signer: </font></td>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:text property="signer" size="16"
                maxlength="16"/></font></td>
    </tr>
    <tr valign="top">
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Signerpass: </font></td>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:text property="signerpass"
size="16"
                maxlength="16"/></font></td>
    </tr>
    <tr valign="top">
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Payload: </font></td>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:textarea property="payload"
                cols="40" rows="5"/></font></td>
    </tr>
    <tr>
     <td></td>
     <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:submit property="submit"
                value="Add"/></font></td>
    </tr>
   </table>
  </html:form>

 </body>

</html:html>


<html>
```

```
 <head>

  <link rel="stylesheet" type="text/css" href="styles/main.css">

 </head>

 <body link="blue" vlink="blue" alink="blue">

  <font style="{position:absolute;top:4px;left:0px;font-family:Arial;font-size:14px;}">
        <b>  AGENT:</b>
          
        <a href="javascript:window.location.reload()">[refresh]</a>
  </font>

  <font style="{position:absolute;top:24px;left:14px;font-family:Arial;font-size:12px;}">
<%--
        <jsp:useBean id="directory" class="IBCGo.Directory">
        <jsp:setProperty name="directory" property="address"
              value="<%=(String)request.getQueryString()%>" />
        </jsp:useBean>

        <jsp:getProperty name="directory" property="agentList" />
--%>
  </font>

 </body>

</html>


<html>

 <head>

 </head>

 <body topmargin="3" leftmargin="5">

  <font STYLE="{font-family:sans-serif;font-size:12px;}">
   <%= request.getAttribute("content")%>
  </font>

  <%-- Mise a jour du frame 'agent' --%>
  <script language="javascript">
   parent.agent.location = "agent.jsp";
  </script>

 </body>

</html>


<html>

 <head>
```

```
</head>

<body topmargin="2" leftmargin="7">

 <br />

 <font style="{font-family:Arial;font-size:12px;}">
        <ul>
                <li>
                IBC-Go (Itinerary Based Computation and Go) is a mobile agent system, based on an
                itinerary approach which integrates
                active network technologies. This way, IBC-Go exceeds the extension of the
                client/server model to venture in the heart
                of the network.
                </li>

                <br /><br />

                <li>
                To be able to integrate IBC-Go with the equipments of current networks, the assistant
                router approach has been used.
                This one offers several advantages, such as enabling the use the current routers and
                separating
                active routing from passive routing work.
                </li>

                <br /><br />

                <li>
                To make IBC-Go is easy to use, the agents are coded in an independent
                language from that of the platform implementation. It offers more flexibility for the
                user, besides not obliging him to
                understand all the details of implementation.
                </li>
        </ul>
 </font>

</body>

</html>


<html>
 <head></head>
 <body>
  <!-- Ligne verticale jaune -->
  <img src="images/FillYellow.gif" width="2" height="99%" border="0"
     style="{position:absolute;top:0px;left:0px;visibility:visible;z-index:1;}">
 </body>
</html>


<html>
 <head> </head>
 <body>
  <!-- Ligne horizontale jaune -->
```

```
    <img src="images/FillYellow.gif" width="100%" height="2" border="0"
        style="{position:absolute;top:0px;left:0px;visibility:visible;z-index:1;}">
  </body>
</html>


<html>
  <head></head>
  <body>
    <!-- Ligne horizontale jaune -->
    <img src="images/FillYellow.gif" width="99%" height="2" border="0"
        style="{position:absolute;top:0px;left:4px;visibility:visible;z-index:1;}">
  </body>
</html>


<%@ taglib uri="/WEB-INF/ibcgo.tld" prefix="ibcgo" %>

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">

<html>

  <head>
    <link rel="stylesheet" type="text/css" href="styles/main.css">
  </head>

  <body link="blue" vlink="blue" alink="blue">

    <script language=Javascript src=scripts/tree.js></script>

    <font style="{position:absolute;top:8px;left:0px;font-family:Arial;font-
          size:14px;}"><B>  CODE:</B></font>

          <%-- N'accepte pas!?!!: --%>
            <%-- <ibcgo:codeList port="=port"/> --%>
    <ibcgo:codeList />

    <div class="tree">
      <script type="text/javascript">
            createTree(Tree);
      </script>
    </div>

  </body>

</html>


<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

<html:html locale="true">

  <head>
    <html:base/>
  </head>
```

```
<body topmargin="3" leftmargin="5">

  <% String address = (String)session.getAttribute("addressants"); %>
  <% String code = (String)request.getAttribute("code"); %>

  <font STYLE="{font-family:sans-serif;font-size:12px;}">
   <%--
   <form action="/AddCode2.do">

    <hidden property="port" value="<%=port%>" />
    <hidden property="name" value="<%=code%>" />

    <table>
     <tr valign="top">
      <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Code: </font></td>
      <td>
       <font STYLE="{font-family:sans-serif;font-size:12px;}">
        <textarea property="code" cols="40" rows="10" value="<%=
              request.getAttribute("content")%>"/>
       </font>
      </td>
     </tr>
     <tr>
      <td></td>
      <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:submit property="submit"
              value="Modify"/></font></td>
     </tr>
    </table>

   </form>
   --%>

   <%= request.getAttribute("content")%>

  </font>

  <br>

  <input type="button" value="Remove"

onClick="javascript:location.href('RemoveCode.do?address=<%=address%>&code=<%=code%>');">
 </body>

</html:html>


<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

<html:html locale="true">

 <head>
  <html:base/>
 </head>
```

```
<body topmargin="3" leftmargin="5">

  <font STYLE="{font-family:sans-serif;font-size:12px;}">
    <b>NEW CODE</b>
  </font>

  <% String address = (String)session.getAttribute("addressants"); %>

  <html:form action="/AddCode.do">
    <html:hidden property="address" value="<%=address%>" />
    <table>
      <tr>
        <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Name: </font></td>
        <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:text property="name" size="16"
            /></font></td>
      </tr>
      <tr valign="top">
        <td><font STYLE="{font-family:sans-serif;font-size:12px;}">Code: </font></td>
        <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:textarea property="code"
            cols="40"
            rows="10"/></font></td>
      </tr>
      <tr>
        <td></td>
        <td><font STYLE="{font-family:sans-serif;font-size:12px;}"><html:submit property="submit"
            value="Add"/></font></td>
      </tr>
    </table>
  </html:form>

 </body>

</html:html>


<html>
 <head></head>
 <body topmargin="3" leftmargin="5">
   <font STYLE="{font-family:sans-serif;font-size:12px;}">
     <%= request.getAttribute("content")%>
   </font>

   <%-- Mise a jour du frame 'code' --%>
   <script language="javascript">
     parent.code.location = "code.jsp";
   </script>
 </body>
</html>


<%-- Comment utiliser une applet signee avec WSAD? --%>
<%-- Necessaire pour la console et les fonctions agent --%>


<%@page import="IBCGo.authentification.RdbmsLoginModule" %>
```

```
<%@page import="IBCGo.authentification.PassiveCallbackHandler" %>
<%@page import="javax.security.auth.login.LoginContext, java.util.*" %>
<%@page import="java.io.*" %>

<html>

 <head>

 </head>

 <body topmargin="0" leftmargin="5">

  <font style="{font-size:12px;}">
    <%
    //System.setProperty("java.security.auth.login.config","E://Program
        Files//Eclipse//workspace//ibcgo//IBCGo//db//ODBC.config");

          try {
            String user = (String)session.getAttribute("user");
            String pass = (String)session.getAttribute("pass");
            String address = (String)session.getAttribute("addressip");
            String addressants = (String)session.getAttribute("addressants");
            String port = (String)session.getAttribute("port");
            //PassiveCallbackHandler consoleCallbackHandler = new PassiveCallbackHandler(user,
                  pass);
            //LoginContext loginContext = new LoginContext("IBCGo", consoleCallbackHandler);
            //loginContext.login();
     %>
      <applet codebase="applets" archive="connexion.jar" code="Main" name="console"
        width="101.5%"
        height="104%" style="{position:absolute;top:-2;left:-2;z-index:3;}">
                  <param name="user" value="<%=user%>" />
                  <param name="pass" value="<%=pass%>" />
                  <param name="address" value="<%=address%>" />
                  <param name="addressants" value="<%=addressants%>" />
                  <param name="port" value="<%=port%>" />
      </applet>
     <%
           //loginContext.logout();
           }
          catch (Exception e) {}
     %>
  </font>

 </body>

</html>


<html>

 <head>

        <STYLE>
        <!--
        a:hover  {color:#FFFFFF; text-decoration:none}
```

```
        a:visited {text-decoration:none}
        a:link    {text-decoration:none}
        -->
        </style>

</head>

<body link="#000000" vlink="#000000" alink="#000000">

 <!-- Cursors -->
 <!-- auto,crosshair,default,hand,move,e-resize,ne-resize,nw-resize,n-resize,se-resize,sw-resize,
              s-resize,w-resize,text, wait,help -->

 <!-- Scanlines -->
 <script language="javascript">
 /*
  for (i=0;i<=20;i++)
   {
   document.write(
                  "<IMG" +
                  " ID = LigneHorizontale" + i +
                  " SRC = images/FillBlack.gif" +
                  " WIDTH = 100.7%" +
                  " HEIGHT = 1" +
                  " STYLE='" +
                  "     {" +
                  "     position : absolute;" +
                  "     top : " + (i*3) + "px;" +
                  "     left : -9px;" +
                  "     visibility : visible;" +
                  "     z-index : 3;" +
                  "     }'" +
                  ">"
                  )
   }
  */
 </script>

 <!-- Logo -->
 <!--
 <img src="images/Logo.gif" height="50" border="0"
  style="{position:absolute;top:5px;left:8px;visibility:visible;z-index:2;}">
 -->
 <font style="{position:absolute;top:1px;left:1px;font-family:Arial black;font-size:50px;}">IBCGo</font>

 <!-- Yellow box -->
 <img src='images/FillYellow.gif' width='99.80%' height='27' border='1'
  style="{position:absolute;top:63px;left:1px;visibility:visible;z-index:2;}">

 <!-- Menu -->
 <font
  id="Logout"
  onMouseOver="document.all.Logout.style.color='#FFFFFF';"
  onMouseOut="document.all.Logout.style.color='#000000';"
  style="{position:absolute;top:68px;left:8px;font-family:Arial;font-
        size:14px;color:#000000;visibility:visible;z-index:3;cursor:hand;}">
```

```
    <b><a href="index.jsp" target="_top">Logout</a></b>
</font>
<font
 id="CreateAgent"
 onMouseOver="document.all.CreateAgent.style.color='#FFFFFF';"
 onMouseOut="document.all.CreateAgent.style.color='#000000';"
 style="{position:absolute;top:68px;left:74px;font-family:Arial;font-
        size:14px;color:#000000;visibility:visible;z-index:3;cursor:hand;}">
   <b><a href="NewAgent.do" target="body">Create Agent</a></b>
</font>
<font
 id="CreateCode"
 onMouseOver="document.all.CreateCode.style.color='#FFFFFF';"
 onMouseOut="document.all.CreateCode.style.color='#000000';"
 style="{position:absolute;top:68px;left:182px;font-family:Arial;font-
        size:14px;color:#000000;visibility:visible;z-index:3;cursor:hand;}">
   <b><a href="NewCode.do" target="body">Create Code</a></b>
</font>
<font
 id="AgentAt"
 onMouseOver="document.all.AgentAt.style.color='#FFFFFF';"
 onMouseOut="document.all.AgentAt.style.color='#000000';"
 style="{position:absolute;top:68px;left:284px;font-family:Arial;font-
        size:14px;color:#000000;visibility:visible;z-index:3;cursor:hand;}">
   <b>Agent at</b>
</font>
<font
 id="TraceAgent"
 onMouseOver="document.all.TraceAgent.style.color='#FFFFFF';"
 onMouseOut="document.all.TraceAgent.style.color='#000000';"
 style="{position:absolute;top:68px;left:358px;font-family:Arial;font-
        size:14px;color:#000000;visibility:visible;z-index:3;cursor:hand;}">
   <b>Trace Agent</b>
</font>
<font
 id="ReportOrigin"
 onMouseOver="document.all.ReportOrigin.style.color='#FFFFFF';"
 onMouseOut="document.all.ReportOrigin.style.color='#000000';"
 style="{position:absolute;top:68px;left:460px;font-family:Arial;font-
        size:14px;color:#000000;visibility:visible;z-index:3;cursor:hand;}">
   <b>Report to Origin</b>
</font>
<font
 id="PollAgent"
 onMouseOver="document.all.PollAgent.style.color='#FFFFFF';"
 onMouseOut="document.all.PollAgent.style.color='#000000';"
 style="{position:absolute;top:68px;left:590px;font-family:Arial;font-
        size:14px;color:#000000;visibility:visible;z-index:3;cursor:hand;}">
   <b>Poll Agent</b>
</font>

<!-- Vertical black lines -->
<img src="images/FillBlack.gif" width="2" height="18" border="0"
 style="{position:absolute;top:68px;left:65px;visibility:visible;z-index:2;}">
<img src="images/FillBlack.gif" width="2" height="18" border="0"
 style="{position:absolute;top:68px;left:173px;visibility:visible;z-index:2;}">
```

```
  <img src="images/FillBlack.gif" width="2" height="18" border="0"
    style={position:absolute;top:68px;left:276px;visibility:visible;z-index:2;}">
  <img src="images/FillBlack.gif" width="2" height="18" border="0"
    style={position:absolute;top:68px;left:349px;visibility:visible;z-index:2;}">
  <img src="images/FillBlack.gif" width="2" height="18" border="0"
    style={position:absolute;top:68px;left:450px;visibility:visible;z-index:2;}">
  <img src="images/FillBlack.gif" width="2" height="18" border="0"
    style={position:absolute;top:68px;left:580px;visibility:visible;z-index:2;}">

 </body>

</html>


<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>

<html:html locale="true">

 <style>
        body
                {
                scrollbar-3dlight-color              : #FFFFFF;
                scrollbar-shadow-color               : #FFFFFF;
                scrollbar-face-color                 : #FFFFFF;
                scrollbar-darkshadow-color           : #FFFFFF;
                scrollbar-highlight-color    : #FFFFFF;
                scrollbar-track-color                : #FFFFFF;
                }
 </style>

 <title>IBCGo (Itinerary Based Computation and Go)</title>

 <head>
  <html:base/>
 </head>

 <body topmargin="3" leftmargin="5">

  <html:form action="/Login.do" target="_parent">
        <div style="{position:absolute;top:36%;left:40%;z-index:3;}">
                <img src="images/FillYellow.gif" width="246" height="170" border="1">
        </div>
        <div style="{position:absolute;top:36%;left:40%;z-index:5;}">
                        <table border=0>
                        <tr>
                                <td><font style="{font-family:Arial;font-size:12px;}">ID:</font></td>
                                <td><font style="{font-family:Arial;font-size:12px;}"><html:text
                                        property="user"/></font></td>
                        </tr>
                        <tr>
                                <td><font style="{font-family:Arial;font-
                                        size:12px;}">Password:</font></td>
                                <td><font style="{font-family:Arial;font-size:12px;}"><html:text
                                        property="pass"/></font></td>
```

```
                        </tr>
                        <tr>
                                <td><font style="{font-family:Arial;font-size:12px;}">IP
                                        Address:</font></td>
                                <td><font style="{font-family:Arial;font-size:12px;}"><html:text
                                        property="addressip"/></font></td>
                        </tr>
                        <tr>
                                <td><font style="{font-family:Arial;font-size:12px;}">ANTS
                                        Address:</font></td>
                                <td><font style="{font-family:Arial;font-size:12px;}"><html:text
                                        property="addressants"/></font></td>
                        </tr>
                        <tr>
                                <td><font style="{font-family:Arial;font-
                                        size:12px;}">Port:</font></td>
                                <td><font style="{font-family:Arial;font-size:12px;}"><html:text
                                        property="port"/></font></td>
                        </tr>
                        <tr>
                                <td></td>
                                <td><font style="{font-family:Arial;font-size:12px;}"><html:submit
                                        property="submit" value="Login"/></font></td>
                        </tr>
                        </table>
                </div>
  </html:form>

 </body>

</html:html>


<html>

 <head>
   <title>IBCGo Console</title>
 </head>

 <frameset rows="92,*" frameborder="1" border="1" resize="yes">
   <frame name="header" title="Cadre d'en-tête" src="header.html" marginwidth="0" marginheight="0"
         scrolling="no" resize="no" />
   <frameset cols="23%,2,*" framespacing="3" resize="no">
     <frameset rows="30%,2,*" framespacing="0" frameborder="0" border="0" resize="no">
       <frame name="agent" title="Cadre des agents" src="agent.jsp" marginwidth="0" marginheight="0"
           resize="no" />
             <frame name="border3" src="border3.html" marginwidth="0" marginheight="0"
                   scrolling="no"
           resize="no" />
       <frame name="code" title="Cadre des codes" src="code.jsp" marginwidth="0" marginheight="0"
           resize="no" />
     </frameset>
     <frame name="border1" src="border1.html" marginwidth="0" marginheight="0" scrolling="no"
           resize="no" />
     <frameset rows="*,2,100" framespacing="2" border="1" resize="no">
             <frame name="body" title="Cadre de zone de travail" SRC="body.html" marginwidth="15"
```

```
                marginheight="10" resize="no" />
            <frame name="border2" src="border2.html" marginwidth="0" marginheight="0" />
    <frame name="console" title="Cadre de la zone Etat" src="console.jsp" resize="yes"
        scrolling="no"
        resize="no" />
   </frameset>
  </frameset>
 </frameset>

 <noframes>
  You must use a browser that supports frames for the IBCGo Console.
 </noframes>

</html>
```

b. L'interpréteur pour le langage AL

```
header {
        package IBCGo.interpreter;
        }

{
import java.io.*;
import java.io.Reader;
import java.util.*;
import IBCGo.*;
import java.security.*;
import ants.Xdr;
import ants.ByteArray;
import com.adventnet.snmp.beans.*;
}

class XLRecognizer extends Parser;
options {
        defaultErrorHandler = true;
        }

        {
        private static ExEnvironment master = null;
        private IBCPacket agent = null;
        private static String DB = null;
        private Vector contentType = new Vector();
        private Vector contentName = new Vector();
        private Vector contentValue = new Vector();
        private String returnParam = "";
        private boolean exit = false;
        private boolean finish = true;
        private String programName = "";
        private String procedureNameParam = "";

        // Getters & Setters
```

```
public static ExEnvironment getMaster() { return master; }
public static void setMaster(ExEnvironment ee) { master = ee; }
public IBCPacket getAgent() { return agent; }
public void setAgent(IBCPacket i) { agent = i; }
public static String getDB() { return DB; }
public static void setDB(String i) { DB = i; }
public Vector getContentType() { return contentType; }
public void setContentType(Vector v) { contentType = v; }
public Vector getContentName() { return contentName; }
public void setContentName(Vector v) { contentName = v; }
public Vector getContentValue() { return contentValue; }
public void setContentValue(Vector v) { contentValue = v; }
public String getReturnParam() { return returnParam; }
public void setReturnParam(String s) { returnParam = s; }
public boolean getExit() { return exit; }
public void setExit(boolean b) { exit = b; }
public boolean getFinish() { return finish; }
public void setFinish(boolean b) { finish = b; }
public void setProgramName(String s) { programName = s; }

public static void parseCode(String s) {
        XLRecognizer parser = null;
        try {
                XLLexer lexer = new XLLexer(new StringReader(s));
                parser = new XLRecognizer(lexer);
                parser.program();
                }
        catch (Exception e) {}

        }

public static IBCPacket parseCode(IBCPacket agent, String s) {
        XLRecognizer parser = null;
        try {
                XLLexer lexer = new XLLexer(new StringReader(s));
                parser = new XLRecognizer(lexer);
                parser.setAgent(agent);
                parser.program();
                }
        catch (Exception e) {}

        return parser.getAgent();
        }

private static boolean isNumber(String s) {
        try {
                Double.parseDouble(s);
                return true;
                }
        catch(Exception e) { return false; }
        }

private static int payloadLength(Vector content) {
        int length = Xdr.INT;
        Enumeration enumerationContent = content.elements();
        while(enumerationContent.hasMoreElements())
```

```
                length += Xdr.STRING((String)enumerationContent.nextElement());
        return length;
        }

private static int fromString(String name) {
        StringTokenizer st = new StringTokenizer(name, ".");

        int[] b = new int[4];
        try {
                for (int i = 0; i < 4; i++)
                b[i] = Integer.parseInt(st.nextToken());
                }
        catch (Exception e) {}

        return ((b[0]<<24) | (b[1]<<16) | (b[2]<<8) | b[3]);
        }

private static String toString(int a) {
        String dotted = Integer.toString(a & 0xff);
        dotted = Integer.toString((a >> 8) & 0xff) + "." + dotted;
        dotted = Integer.toString((a >> 16) & 0xff) + "." + dotted;
        dotted = Integer.toString((a >> 24) & 0xff) + "." + dotted;
        return dotted;
        }

private String validation(String type, String value) {
        if(type.equals("Integer") && !isNumber(value)) value = "0";
        if(type.equals("Boolean") && !value.equals("false") && !value.equals("true"))
                value = "false";
        if(type.equals("Char")) value = value.substring(0,0);

        return value;
        }

private void multiAffect(Vector identifiers, String type, String value) {
        if(!type.equals("Boolean") && !type.equals("Char") && !type.equals("Integer") &&
          !type.equals("String") && !type.equals("enumeration") && !type.equals("record")) {
                String newType =
                        (String)contentValue.elementAt(contentName.lastIndexOf((String)"
                        _type_"+type));
                String newTypeName = newType.substring(0, newType.indexOf(":"));
                if(newTypeName.equals("enumeration")) {
                        Vector nameVector = new Vector();
                        String nameContent = newType.substring(newType.indexOf(":")+1);
                        String element;
                        for(
                                StringTokenizer stringtokenizer =
                                new StringTokenizer(nameContent, ";", false);
                                stringtokenizer.hasMoreTokens();
                                nameVector.addElement(element)
                                )
                                element = stringtokenizer.nextToken();
                        Enumeration enumeration = identifiers.elements();
                        while(enumeration.hasMoreElements()) {
                                String identifier = (String)enumeration.nextElement();
                                int i = 0;
```

```
                    Enumeration enumerationName =
                      nameVector.elements();
                    while(enumerationName.hasMoreElements()) {
                            element =
                             (String)enumerationName.nextElement();
                            Vector newVector = new Vector();
                            newVector.addElement("_enumeration_" +
                                    identifier +
                                     "_" + element);
                            multiAffect(newVector, "Integer",
                                    Integer.toString(i++));
                            }
                    }
            }
    else if(newTypeName.equals("array")) {
            type =
                    (String)contentType.elementAt(
                    contentName.lastIndexOf((String)"_type_"+type));
            String range = newType.substring(newType.lastIndexOf(":")+1);
            int low = Integer.parseInt(range.substring(0, range.lastIndexOf("-
                    ")));
            int high = Integer.parseInt(range.substring(range.lastIndexOf("-
                    ")+1));
            Enumeration enumeration = identifiers.elements();
            while(enumeration.hasMoreElements()) {
                    Vector newVector = new Vector();
                    String element = (String)enumeration.nextElement();
                    for(int i=low; i<=high; i++)
                            newVector.addElement("_array_" + element + "_"
                                    +
                                     Integer.toString(i));
                    value = validation(type, value);
                    multiAffect(newVector, type, value);
                    }
            }
    else if(newTypeName.equals("record")) {
            Vector variableRecord = new Vector();
            String newTypeContent =
                    newType.substring(newType.indexOf(":")+1);
            String element;
            for(
                    StringTokenizer stringtokenizer = new
                    StringTokenizer(newTypeContent, ";", false);
                    stringtokenizer.hasMoreTokens();
                    variableRecord.addElement(element)
                    )
                            element = stringtokenizer.nextToken();
            Enumeration enumeration = identifiers.elements();
            while(enumeration.hasMoreElements()) {
                    String identifier = (String)enumeration.nextElement();
                    Enumeration enumerationRecord =
                            variableRecord.elements();
                    while(enumerationRecord.hasMoreElements()) {
                            element =
                              (String)enumerationRecord.nextElement();
                            Vector newVector = new Vector();
```

```
newVector.addElement("_record_"+identifier+"_"+
                                                element.substring(0,
                                                element.indexOf(":")));
                                        value =
                                                validation(element.substring(
                                                element.indexOf(":")+1), value);
                                        multiAffect(newVector,
                                          element.substring(element.indexOf(":")+1),
                                value);
                                        }
                                }
                        }
                }
        else {
                Enumeration enumeration = identifiers.elements();
                while(enumeration.hasMoreElements()) {
                        String element = (String)enumeration.nextElement();
                        if(!element.substring(0,1).equals("_")) value = validation(type,
                                value);
                        if(element.length()>10)
                                if(element.substring(0,10).equals("_constant_")) value =
                                        validation(type, value);
                        contentName.addElement(element);
                        contentType.addElement(type);
                        contentValue.addElement(value);
                        }
                }
        }

private void execute(String statement) {
        try {
                XLLexer lexer = new XLLexer(new StringReader(statement));
                XLRecognizer parser2 = new XLRecognizer(lexer);
                parser2.setAgent(getAgent());
                parser2.setContentType(getContentType());
                parser2.setContentName(getContentName());
                parser2.setContentValue(getContentValue());
                parser2.setReturnParam(getReturnParam());
                parser2.setExit(getExit());
                parser2.setFinish(getFinish());
                parser2.statementList();
                setAgent(parser2.getAgent());
                setContentType(parser2.getContentType());
                setContentName(parser2.getContentName());
                setContentValue(parser2.getContentValue());
                setReturnParam(parser2.getReturnParam());
                setExit(parser2.getExit());
                setFinish(parser2.getFinish());
                }
        catch (Exception e) {}
        }

private void executeProgram(String statement) {
        try {
```

```
                        XLLexer lexer = new XLLexer(new StringReader(statement));
                        XLRecognizer parser2 = new XLRecognizer(lexer);
                        parser2.setAgent(getAgent());
                        parser2.setContentType(getContentType());
                        parser2.setContentName(getContentName());
                        parser2.setContentValue(getContentValue());
                        parser2.setReturnParam(getReturnParam());
                        parser2.setExit(getExit());
                        parser2.setFinish(getFinish());
                        parser2.program();
                        setAgent(parser2.getAgent());
                        setContentType(parser2.getContentType());
                        setContentName(parser2.getContentName());
                        setContentValue(parser2.getContentValue());
                        setReturnParam(parser2.getReturnParam());
                        setExit(parser2.getExit());
                        setFinish(parser2.getFinish());
                        }
                catch (Exception e) {}
                }

        }
```

```
program
  : "program" name:IDENT
                {
                programName = name.getText();
                Vector vector = new Vector();
                vector.addElement("_constant_TRUE");
                multiAffect(vector, "Boolean", "true");
                vector = new Vector();
                vector.addElement("_constant_FALSE");
                multiAffect(vector, "Boolean", "false");
                }
    EQUALS
    subprogramBody
    DOT
  ;

subprogramBody
  : (basicDecl)*
    (procedureDecl)*
    "begin"
    statementList
    "end" IDENT
  ;

basicDecl
  : varDecl
  | constDecl
  | typeDecl
```

```
 ;

typeName returns [String type = "";]
 : t:IDENT          { type=t.getText(); }
 | "Integer"        { type="Integer"; }
 | "Boolean"        { type="Boolean"; }
 | "String"         { type="String"; }
 | "Char"           { type="Char"; }
 ;

varDecl
                    {
                    Vector identifiers = new Vector();
                    String type = "";
                    String value = "";
                    }
 : "var" identifiers=identList COLON type=typeName (BECOMES value=constantValue)?SEMI
                    { multiAffect(identifiers, type, value); }
 ;

constDecl           {
                    Vector identifiers = new Vector();
                    String type = "";
                    String value = "";
                    }
 : "constant" identifiers=identList COLON type=typeName BECOMES value=constantValue SEMI
                    {
                    Vector idenModif = new Vector();
                    Enumeration enumeration = identifiers.elements();
                    while(enumeration.hasMoreElements())
                            idenModif.addElement("_constant_"+
                                    (String)enumeration.nextElement());
                    multiAffect(idenModif, type, value);
                    }
 ;

identList returns [Vector identifiers = new Vector();]
 : first:IDENT      { identifiers.addElement(first.getText()); }
   (
     COMMA
     others:IDENT
                    { identifiers.addElement(others.getText()); }
   )*
 ;

constantValue returns [String value="";]
                    { String identifier = ""; }
 : t:INT            { value = t.getText(); }
 | u:STRING         { value = u.getText(); }
 | v:CHAR           { value = u.getText(); }
 | identifier=variableReference
                    { value =
        (String)contentValue.elementAt(contentName.lastIndexOf((String)identifier)); }
 ;

typeDecl            {
```

```
                Vector identifiers = new Vector();
                String type;
                Vector types = new Vector();
                String x = "";
                String x2= "";
                String record = "";
                }
: "type" identifier:IDENT EQUALS
 (
  LT
  first:IDENT   { types.addElement(first.getText()); }
  (
   COMMA
   others:IDENT
                { types.addElement(others.getText()); }
  )*
  GT            {
                Enumeration enumerationTypes = types.elements();
                String enumeration = "enumeration:";
                while(enumerationTypes.hasMoreElements())
                        enumeration += (String)enumerationTypes.nextElement() + ";";
                identifiers.addElement("_type_"+identifier.getText());
                multiAffect(identifiers, "enumeration", enumeration);
                }
 | "array" LBRACKET x=constantValue DOTDOT x2=constantValue RBRACKET "of" type=typeName
                {
                identifiers.addElement("_type_"+identifier.getText());
                multiAffect(identifiers, type, "array:"+x+"-"+x2);
                }
 | "record"     { record = "record:"; }
  (
        identifiers=identList
        COLON
        types=typeList
                {
                Enumeration enumerationIdentifier = identifiers.elements();
                Enumeration enumerationType = types.elements();
                type = (String)enumerationType.nextElement();
                while(enumerationIdentifier.hasMoreElements())
                        record += (String)enumerationIdentifier.nextElement() + ":" + type + ";";
                }
        SEMI
  )+ "end" "record"
                {
                Vector vectorRecord = new Vector();
                vectorRecord.addElement("_type_"+identifier.getText());
                multiAffect(vectorRecord, "record", record);
                }
 )
 SEMI
;

typeList returns [Vector types = new Vector();]
                {
                String first = "";
                String others = "";
```

```
                }
 : first=typeName
                { types.addElement(first); }
  (
    COMMA
    others=typeName
                { types.addElement(others); }
  )*
 ;

procedureDecl
                {
                String parameterName = "";
                Vector parameters = new Vector();
                }
 : "procedure" procedureName:IDENT
                { procedureNameParam = procedureName.getText(); }
  (parameterName=formalParameters)? EQUALS
                {
                String element;
                for(
                        StringTokenizer stringtokenizer =
                                new StringTokenizer(parameterName, ";", false);
                        stringtokenizer.hasMoreTokens();
                        parameters.addElement(element)
                        )
                                element = stringtokenizer.nextToken();
                }
  // subprogramBody
                {
                contentName.addElement("_proc_" + procedureName.getText());
                contentType.addElement("Procedure");
                String content = "program " + procedureName.getText() + " =\r\n";
                while(!LT(1).getText().equals("end") ||
        !LT(2).getText().equals(procedureName.getText()))
                {
                        String item = LT(1).getText();
                        Enumeration enumeration = parameters.elements();
                        while(enumeration.hasMoreElements()) {
                                String parameter = (String)enumeration.nextElement();
                                if(LT(1).getType()!=21 && item.equals(parameter))
                                        item = "_param_"+ procedureNameParam + "_" + item;
                                }
                        if(LT(1).getType()==21) content += "\"";
                        if(item.equals("begin") || LT(1).getText().equals(";"))
                                content += item + "\r\n";
                        else if(LT(1).getType()==21)
                                content += item + "\"";
                        else
                                content += item + " ";
                        consume();
                }
                content += LT(1).getText() + " ";
                content += LT(2).getText() + ".\r\n";
                consume();
                consume();
```

```
                    contentValue.addElement(content);
                    procedureNameParam = "";
                    parameterName = "";
                    }
  SEMI
 ;

formalParameters returns [String parameterName = "";]
                    { String identifier = ""; }
 : LPAREN identifier=parameterSpec
                    { parameterName = identifier; }
  (
   SEMI identifier=parameterSpec
                    { parameterName += ";" + identifier; }
  )* RPAREN
 ;

parameterSpec returns [String identifier = "";]
                    { String type = ""; }
 : ("var")? identifier2:IDENT COLON type=typeName
                    {
                    identifier = identifier2.getText();
                    Vector idenModif = new Vector();
                    idenModif.addElement((String)"_param_"+ procedureNameParam + "_" + identifier);
                    multiAffect(idenModif, type, "");
                    }
 ;

variableReference returns [String value = "";]
                    {
                    boolean array = false;
                    String index = "";
                    boolean member = false;
                    }
 : identifier:IDENT
  (
   LBRACKET index=expression RBRACKET
                    { array = true; }
  | DOT identifier2:IDENT
                    { member = true; }
  )*
                    {
                    if(!array && !member) {
                            if(contentName.contains((String)"_constant_"+identifier.getText()))
                                    value = "_constant_"+identifier.getText();
                            if(contentName.contains((String)identifier.getText()))
                                    value = identifier.getText();
                            }
                    else if (!member)
                            value = "_array_"+identifier.getText()+"_"+index;
                    else {
                            if(contentName.contains((String)"_enumeration_"+
                              identifier.getText()+"_"+identifier2.getText()))
                                    value =
"_enumeration_"+identifier.getText()+"_"+identifier2.getText();
                            if(contentName.contains((String)"_record_"+identifier.getText()+
```

```
                        "_"+identifier2.getText()))
                                value = "_record_"+identifier.getText()+"_"+identifier2.getText();
                        }
                }
  ;

statementList
  : statement statementList
  | (endStatement) => endStatement statementList
  | // nothing
  ;

// (IDENT (LPAREN|SEMI)) => result=procedureCallStatement doit etre en premier
statement       { String result; }
  : (IDENT LPAREN) => result=procedureCallStatement
  | assignmentStatement
  | returnStatement
  | ifStatement
  | loopStatement
  | ioStatement
  | fileStatement
  | dbStatement
  | payloadStatement
  | nodeStatement
  | codeStatement
  | result=agentStatement
  | itiStatement
  | comStatement
  | controlStatement
  | whenStatement
  | exitStatement
  ;

assignmentStatement
                { String identifier, value; }
  : identifier=variableReference BECOMES value=expression SEMI
                {
                int index = contentName.lastIndexOf((String)identifier);
                value = validation((String)contentType.elementAt(index), value);
                contentValue.set(index, (String)value);
                }
  ;

procedureCallStatement returns [String value=""]
                { String result = ""; }
  : ("call")? procedure:IDENT (result=actualParameters)? SEMI
                {
                Vector parameters = new Vector();
                String element;
                for(
                        StringTokenizer stringtokenizer = new StringTokenizer(result, ";", false);
                        stringtokenizer.hasMoreTokens();
                        parameters.addElement(element)
                        )
                                element = stringtokenizer.nextToken();
                                int index = contentName.indexOf((String)"_proc_"+
```

```
                                        procedure.getText()) - parameters.size();
                                    Enumeration enumeration = parameters.elements();
                                    while(enumeration.hasMoreElements()) {
                                        String parameter = (String)enumeration.nextElement();
                                        contentValue.set(index++, (String)parameter);
                                        }
                                    index = contentName.lastIndexOf((String)"_proc_" +
                                        procedure.getText());
                                    String code = (String)contentValue.elementAt(index);
                                    executeProgram(code);
                                    value = returnParam;
                                    returnParam = "";
                                    index = contentName.indexOf((String)"_proc_"+
                                        procedure.getText()) - parameters.size();
                                    enumeration = parameters.elements();
                                    while(enumeration.hasMoreElements()) {
                                        String parameter = (String)enumeration.nextElement();
                                        contentValue.set(index++, (String)"");
                                        }
                    }
  | "agentAt" (result=actualParameters)?
                    { value = master.agentAt(result); }
  | "origin" (result=actualParameters)?
                    { value = master.agentOrigin(this.agent.getIdentifier(), result); }
  | "snmpGet" LPAREN result=expression RPAREN SEMI
                    {
                    SnmpTarget target = new SnmpTarget();
                    target.setTargetHost("localhost");
                    target.setObjectID(result);
                    value = target.snmpGet();
                    }
  ;

actualParameters returns [String value=""]
                    { String result; }
  :
    LPAREN
    (
      result=expression
                    { value = result; }
      (
        COMMA
        result=expression
                    { value += ";" + result; }
      )*
    )?
    RPAREN
  ;

returnStatement
                    { String value = ""; }
  : "return" value=expression SEMI
                    { returnParam = value; }
  ;

ifStatement
```

```
 : "if" ifPart
   "end" "if"        { finish = true; }
   SEMI
 ;

ifPart               { String result; }
 : result=expression "then"
                     {
                     if(result=="false") {
                              while(
                                (!LT(1).getText().equals("elsif") && !LT(1).getText().equals("else")) &&
                                !(LT(1).getText().equals("end") && LT(2).getText().equals("if"))
                                )
                                       consume();
                              }
                     else {
                              String statement = "";
                              while(
                                (!LT(1).getText().equals("elsif") && !LT(1).getText().equals("else")) &&
                                !(LT(1).getText().equals("end") && LT(2).getText().equals("if"))
                                ) {
                                       if(LT(1).getType()==21) statement += "\"";
                                       statement += LT(1).getText();
                                       if(LT(1).getType()==21)
                                                statement += "\"";
                                       else
                                                statement +=  " ";
                                                consume();
                                       }
                              statement += "end";
                              if(finish) execute(statement);
                              finish = false;
                              }
                     }
   (
      "elsif" ifPart  {
                     if(result=="true")
                     while(!(LT(1).getText().equals("end") && LT(2).getText().equals("if"))) consume();
                     }
     | "else"        {
                     if(result=="true")
                              while(!(LT(1).getText().equals("end") && LT(2).getText().equals("if")))
consume();
                     else {
                              String statement = "";
                              while(!(LT(1).getText().equals("end") && LT(2).getText().equals("if"))) {
                                       if(LT(1).getType()==21) statement += "\"";
                                                statement += LT(1).getText();
                                       if(LT(1).getType()==21)
                                                statement += "\"";
                                       else
                                                statement +=  " ";
                                       consume();
                                       }
                              statement += "end";
                              if(finish) execute(statement);
```

```
                        }
                }
  )?
 ;

loopStatement   { String condition = ""; }
 : (
    "while"     {
                while(!LT(1).getText().equals("loop")) {
                        if(LT(1).getType()==21) condition += "\"";
                                condition += LT(1).getText();
                                if(LT(1).getType()==21)
                                        condition += "\"";
                                else
                                        condition +=  " ";
                                consume();
                        }
                }
  )? "loop"     {
                String statement = "";
                int loopCount = 1;
                while(loopCount > 0) {
                        while(!(LT(1).getText().equals("end") && LT(2).getText().equals("loop"))) {
                                if(LT(1).getText().equals("loop")) loopCount++;
                                if(LT(1).getType()==21) statement += "\"";
                                statement += LT(1).getText();
                                if(LT(1).getText().equals("begin") || LT(1).getText().equals(";"))
                                        statement += "\r\n";
                                if(LT(1).getType()==21)
                                        statement += "\"";
                                else
                                        statement +=  " ";
                                consume();
                                }
                        loopCount--;
                        if(loopCount>0) {
                                statement += "end loop;\r\n";
                                consume();
                                consume();
                                consume();
                                }
                        }
                statement += "end.";

                if(condition.equals("")) {
                        while(!exit) execute(statement);
                        }
                 else {
                        condition = "executeWhen " + condition + " end";
                        execute(condition);
                        while(!exit) {
                                execute(statement);
                                execute(condition);
                                }
                        }
                exit = false;
```

```
                    }
  "end" "loop" SEMI
 ;
whenStatement   { String result; }
 : "executeWhen" result=expression
                    { if(result.equals("false")) exit = true; }
 ;

exitStatement       { String result; }
 : "exit" "when" result=expression SEMI
                    {
                    if(result.equals("true")) exit = true;
                    if(exit) while(!(LT(1).getText().equals("end") && LT(2).getText().equals(".")))
consume();
                    }
 ;

endStatement
 : "end" SEMI
 ;

ioStatement         { String identifier, string; }
 : "put" LPAREN string=expression RPAREN SEMI
                    {
                    System.out.print(string);
                    master.print(string);
                    }
 | "get" LPAREN identifier=variableReference RPAREN SEMI
                    {
                    try {
                      BufferedReader data = new BufferedReader(new InputStreamReader(System.in));
                      String message = data.readLine();
                      contentValue.set(contentName.lastIndexOf((String)identifier), (String)message);
                      }
                    catch(Exception e) {}
                    }
 | "newLine" (LPAREN RPAREN)? SEMI
                    {
                    System.out.println();
                    master.println("");
                    }
 | "skipLine" (LPAREN RPAREN)? SEMI
                    {
                    System.out.println();
                    master.println("");
                    }
 ;

fileStatement       {
                    String file = "";
                    String identifier = "";
                    String content = "";
                    }
 : "putFile" LPAREN file=expression COMMA content=expression RPAREN SEMI
                    {
                    try {
```

```
                        file = file.replace((char)92,'/');
                        file = file.replaceAll("/","//");
                        FileWriter message = new FileWriter(file);
                        BufferedWriter buffer = new BufferedWriter(message);
                        buffer.write(content);
                        buffer.close();
                        message.close();
                        }
                catch(Exception e) {}
                }
 | "getFile" LPAREN identifier=variableReference COMMA file=expression RPAREN SEMI
                {
                try {
                        file = file.replace((char)92,'/');
                        file = file.replaceAll("/","//");
                        FileReader message = new FileReader(file);
                        BufferedReader buffer = new BufferedReader(message);
                        String c;
                        while((c=buffer.readLine()) != null)
                                content += c;
                        contentValue.set(contentName.lastIndexOf((String)identifier),
(String)content);
                        buffer.close();
                        message.close();
                        }
                catch(Exception e) {}
                }
 ;

dbStatement     { String identifier, message, id; }
 : "putMessage" LPAREN id=typeName COMMA message=expression RPAREN SEMI
                {
                Database database = new Database(DB);
                database.createMessage(id, message);
                }
 | "getMessage" LPAREN identifier=variableReference COMMA id=typeName RPAREN SEMI
                {
                Database database = new Database(DB);
                String result = database.readMessage(id);
                contentValue.set(contentName.lastIndexOf((String)identifier), (String)result);
                }
 ;

payloadStatement
                { String identifier, message; }
 : "putPayload" LPAREN message=expression RPAREN SEMI
                {
                String[] temp = new String[1];
                temp[0] = message;
                agent.setPayload(temp);
                }
 | "getPayload" LPAREN identifier=variableReference RPAREN SEMI
                {
                String result = agent.getPayload();
                contentValue.set(contentName.lastIndexOf((String)identifier), (String)result);
                }
```

```
;

nodeStatement   { String identifier; }
 : "getAddress" LPAREN identifier=variableReference RPAREN SEMI
                {
                String result = agent.getCurrentSite();
                contentValue.set(contentName.lastIndexOf((String)identifier), (String)result);
                }
 ;

agentStatement returns [String value=""]
                {
                String name = "";
                String name2 = "";
                Vector gcode = null;
                String code = "";
                String signer = "";
                String signerpass = "";
                String payload = "";
                Vector gagent = null;
                }
 : "createAgent" LPAREN name=expression COMMA
  LACC
  code=expression
                {
                gcode = new Vector();
                gcode.addElement(code);
                }
  (
   COMMA
   code=expression
                { gcode.addElement(code); }
  )*
  RACC
  COMMA payload=expression RPAREN SEMI
                {
                Signature signature = null;
                KeyStore keystore = null;
                try {
                        signature = Signature.getInstance("DSA");
                        keystore = KeyStore.getInstance("JKS");
                        keystore.load(new FileInputStream(master.getMaster().getKeystore()),
                                master.getMaster().getStorepass().toCharArray());
                        signature.initSign((PrivateKey)keystore.getKey(master.getMaster().
                                getPublicUser(),
                        master.getMaster().getPublicPass().toCharArray()));
                        }
                catch(Exception e) {}

                IBCPacket ibcpacket = new IBCPacket();
                ibcpacket.setIdentifier(name);
                ibcpacket.setPTime(DateTime.getLong());
                ibcpacket.setCreator(master.getMaster().getCredentials().getCreator());
                ibcpacket.setCredentials(master.getMaster().getCredentials().getCredentials());
                ibcpacket.setCreSignature(master.getMaster().getCredentials().getSignature());
                ibcpacket.setCreSigner(master.getMaster().getCredentials().getSigner());
```

```
                    ibcpacket.setNbCode(gcode.size());
                    ibcpacket.setCodeType(gcode.size());
                    ibcpacket.setCodeLength(gcode.size());
                    ibcpacket.setCodeData(gcode.size());
                    ibcpacket.setCodeSigner(gcode.size());
                    ibcpacket.setCodeSignature(gcode.size());
                    ibcpacket.setInterpreterAddress(gcode.size());
                    Enumeration gcodeContent = gcode.elements();
                    for(int i=0; i<gcode.size(); i++) {
                            ibcpacket.setCodeType(i, 1);
                            Loader loader = new Loader(agent.getCurrentSite());
                            code = (String)gcodeContent.nextElement();
                            byte[] buffer = loader.saisie(code);
                            ibcpacket.setCodeLength(i, buffer.length);
                            byte[] raw = null;
                            try {
                                    ibcpacket.setCodeData(i, code.getBytes("UTF8"));
                                    ibcpacket.setCodeSigner(i, master.getMaster().getPublicUser());
                                    signature.update(buffer, 0, buffer.length);
                                    raw = signature.sign();
                                    }
                            catch(Exception e) {}
                            ibcpacket.setCodeSignature(i, raw);
                            ibcpacket.setInterpreterAddress(i,
                                    (master.getMaster().thisNode().getAddress()*100000)+
                                    master.getMaster().getAntsPort());
                            }
                    Vector content = new Vector();
                    content.add(payload);
                    ByteArray buf = new ByteArray(payloadLength(content));
                    Xdr xdr = new Xdr(buf,0);
                    xdr.PUT(content.size());
                    Enumeration enumerationContent = content.elements();
                    while(enumerationContent.hasMoreElements())
                    xdr.PUT((String)enumerationContent.nextElement());
                    ibcpacket.setData(buf);
                    master.getMaster().getVerifier().createAgent(agent.getCurrentSite(), name,
ibcpacket);
                    }
  | "destroyAgent" LPAREN name=expression RPAREN SEMI
                    { master.getMaster().getVerifier().removeAgent(agent.getCurrentSite(), name); }
  | "copyAgent" LPAREN name=expression COMMA name2=expression RPAREN SEMI
                    {
                    IBCPacket ibcpacket =
master.getMaster().getVerifier().getAgent(agent.getCurrentSite(),
                            name2);
                    master.getMaster().getVerifier().createAgent(agent.getCurrentSite(), name,
ibcpacket);
                    }
  | "spawnAgent" LPAREN
    LACC
    name=expression
                    {
                    gagent = new Vector();
                    gagent.addElement(name);
                    }
```

```
  (
    COMMA
    name=expression
                    { gagent.addElement(name); }
  )*
  RACC
  COMMA name2=expression RPAREN SEMI
                    {
                    IBCPacket ibcpacket =
                            master.getMaster().getVerifier().getAgent(agent.getCurrentSite(), name2);
                    Enumeration gagentContent = gagent.elements();
                    while(gagentContent.hasMoreElements())
                    master.getMaster().getVerifier().createAgent(agent.getCurrentSite(),
                            (String)gagentContent.nextElement(), ibcpacket);
                    }
| "activateAgent" LPAREN name=expression COMMA keepR:IDENT RPAREN SEMI
                    {
                    IBCPacket ibcpacket =
                            master.getMaster().getVerifier().getAgent(agent.getCurrentSite(), name);
                    Code codep = new Code();
                    codep.setNbCode(ibcpacket.getNbCode());
                    codep.setCodeType(ibcpacket.getCodeType());
                    codep.setCodeLength(ibcpacket.getCodeLength());
                    codep.setCodeData(ibcpacket.getCodeData());
                    codep.setSigner(ibcpacket.getCodeSigner());
                    codep.setSignature(ibcpacket.getCodeSignature());
                    codep.setInterpreterAddress(ibcpacket.getInterpreterAddress());
                    Itinerary itinerary = new Itinerary();
                    itinerary.setNbSite(ibcpacket.getNbItinerary());
                    itinerary.setK(ibcpacket.getK());
                    itinerary.setSite(ibcpacket.getSite());
                    itinerary.setSigner(ibcpacket.getItiSigner());
                    itinerary.setSignature(ibcpacket.getItiSignature());
                    boolean keep = false;
                    if(keepR.getText().equals("true")) keep = true;
                    master.getMaster().sendPacket(name, 1, ibcpacket.getPTime(),
                            master.getMaster().thisNode().getAddress(), ibcpacket.getSite(0),
                            ibcpacket.getSite(ibcpacket.getSite().length-1),
                            master.getMaster().getCredentials(), codep, itinerary, ibcpacket.getData(),
                    keep);
                    }
  ;

itiStatement        {
                    String site = "";
                    String agentID = "";
                    Vector itinerary = null;
                    String signer = "";
                    String signerpass = "";
                    }
  : "It" LPAREN agentID2:IDENT RPAREN BECOMES LPAREN
    LACC
    site=expression
                    {
                    itinerary = new Vector();
                    itinerary.addElement(site);
```

```
                }
  (
    COMMA
    site=expression
                { itinerary.addElement(site); }
  )*
  RACC          {
                IBCPacket ibcpacket =
master.getMaster().getVerifier().getAgent(agent.getCurrentSite(),
                        agentID2.getText());

                Signature signature = null;
                KeyStore keystore = null;
                try {
                        signature = Signature.getInstance("DSA");
                        keystore = KeyStore.getInstance("JKS");
                        keystore.load(new FileInputStream(master.getMaster().getKeystore()),
                                master.getMaster().getStorepass().toCharArray());
                        signature.initSign((PrivateKey)keystore.getKey(master.getMaster().
                                getPublicUser(),
                        master.getMaster().getPublicPass().toCharArray()));
                        }
                catch(Exception e) {}

                int i = 0;
                String itineraries = "";
                ibcpacket.setNbItinerary(itinerary.size()/2);
                ibcpacket.setK(itinerary.size()/2);
                ibcpacket.setSite(itinerary.size()/2);
                Enumeration itineraryContent = itinerary.elements();
                while(itineraryContent.hasMoreElements()) {
                        ibcpacket.setK(i, Integer.parseInt((String)itineraryContent.nextElement()));
                        String element = (String)itineraryContent.nextElement();
                        ibcpacket.setSite(i, fromString(element));
                        itineraries += element;
                        i++;
                        }

                ibcpacket.setItiSigner(master.getMaster().getPublicUser());
                try {
                        byte[] buffer = itineraries.getBytes("UTF8");
                        signature.update(buffer, 0, buffer.length);
                        byte[] raw = signature.sign();
                        ibcpacket.setItiSignature(raw);
                        }
                catch(Exception e) {}

                master.getMaster().getVerifier().createAgent(agent.getCurrentSite(),
agentID2.getText(),
                        ibcpacket);
                }
  RPAREN SEMI
 | "followIt" LPAREN RPAREN SEMI
                {
                master.setFollow(true);
                while(!LT(1).getText().equals("end") || !LT(2).getText().equals(programName))
```

```
                    consume();
                }
| "jumpTo" LPAREN siteID:INT RPAREN SEMI
                {
                master.setFollow(true);
                master.setJump(true);
                master.setDestination(Integer.parseInt(siteID.getText()));
                while(!LT(1).getText().equals("end") || !LT(2).getText().equals(programName))
                    consume();
                }
| "meetAgent" LPAREN agentID=expression
 (
 COMMA site=expression
 )?

 RPAREN SEMI
                {
                if(!site.equals("")) {
                    String statement = "borrowIt(\""+site+"\");";
                    statement += "end";
                    execute(statement);
                    }

                int agentAddress = fromString(master.agentAddress(this.agent.getIdentifier(),
                    agentID));

                int agentNo = agent.getSite().length-1;
                int i = agent.getSite().length-1;
                int siteIti = agent.getSite(i);
                while(siteIti != agentAddress) {
                    i--;
                    siteIti = agent.getSite(i);
                    }
                agentNo = i;

                master.setFollow(true);
                master.setJump(true);
                master.setDestination(agentNo+1);
                while(!LT(1).getText().equals("end") || !LT(2).getText().equals(programName))
                    consume();
                }
| "borrowIt" LPAREN agentID=expression RPAREN SEMI
                {
                String agentItinerary = master.getItinerary(this.agent.getIdentifier(), agentID);
                master.setNewStart(agent.getSite().length+1);
                itinerary = new Vector();

                for(int i=0; i<agent.getSite().length; i++) {
                    itinerary.addElement(Integer.toString(agent.getK(i)));
                    itinerary.addElement(toString(agent.getSite(i)));
                    }
                itinerary.addElement("0");
                itinerary.addElement("0.0.0.0");
                String element;
                for(
                    StringTokenizer stringtokenizer = new StringTokenizer(agentItinerary, "/",
```

```
                        false);
                stringtokenizer.hasMoreTokens();
                itinerary.addElement(element)
                )
                        element = stringtokenizer.nextToken();

        Signature signature = null;
        KeyStore keystore = null;
        try {
                signature = Signature.getInstance("DSA");
                keystore = KeyStore.getInstance("JKS");
                keystore.load(new FileInputStream(master.getMaster().getKeystore()),
                        master.getMaster().getStorepass().toCharArray());
                signature.initSign((PrivateKey)keystore.getKey(master.getMaster().
                        getPublicUser(),
                master.getMaster().getPublicPass().toCharArray()));
                }
        catch(Exception e) {}

        int i = 0;
        String itineraries = "";
        agent.setNbItinerary(itinerary.size()/2);
        agent.setK(itinerary.size()/2);
        agent.setSite(itinerary.size()/2);
        Enumeration itineraryContent = itinerary.elements();
        while(itineraryContent.hasMoreElements()) {
                agent.setK(i, Integer.parseInt((String)itineraryContent.nextElement()));
                element = (String)itineraryContent.nextElement();
                agent.setK(i, 0);
                agent.setSite(i, fromString(element));
                itineraries += element;
                i++;
                }

        agent.setItiSigner(master.getMaster().getPublicUser());
        try {
                byte[] buffer = itineraries.getBytes("UTF8");
                signature.update(buffer, 0, buffer.length);
                byte[] raw = signature.sign();
                agent.setItiSignature(raw);
                }
        catch(Exception e) {}

        master.setDestination(agent.getSite().length-1);
        }
| "resetIt" LPAREN RPAREN SEMI
        {
        int limite = 0;
        int i = 0;
        int siteIti = agent.getSite(i);
        while((siteIti!=fromString("0.0.0.0")) && (i<agent.getSite().length)) {
                i++;
                siteIti = agent.getSite(i);
                }
        limite = i-1;
```

```
                if(limite < agent.getSite().length-1) {
                        master.setNewStart(0);
                        itinerary = new Vector();
                        for(i=0; i<limite+1; i++) {
                                itinerary.addElement(Integer.toString(agent.getK(i)));
                                itinerary.addElement(toString(agent.getSite(i)));
                                }
                        Signature signature = null;
                        KeyStore keystore = null;
                        try {
                                signature = Signature.getInstance("DSA");
                                keystore = KeyStore.getInstance("JKS");
                                keystore.load(new
                                        FileInputStream(master.getMaster().getKeystore()),
                                        master.getMaster().getStorepass().toCharArray());
                                signature.initSign((PrivateKey)keystore.getKey(master.getMaster().
                                        getPublicUser(),
                                        master.getMaster().getPublicPass().toCharArray()));
                                }
                        catch(Exception e) {}

                        i = 0;
                        String itineraries = "";
                        agent.setNbItinerary(itinerary.size()/2);
                        agent.setK(itinerary.size()/2);
                        agent.setSite(itinerary.size()/2);
                        Enumeration itineraryContent = itinerary.elements();
                        while(itineraryContent.hasMoreElements()) {
                                agent.setK(i,
                                        Integer.parseInt((String)itineraryContent.nextElement()));
                                String element = (String)itineraryContent.nextElement();
                                agent.setK(i, 0);
                                agent.setSite(i, fromString(element));
                                itineraries += element;
                                i++;
                                }

                        agent.setItiSigner(master.getMaster().getPublicUser());
                        try {
                                byte[] buffer = itineraries.getBytes("UTF8");
                                signature.update(buffer, 0, buffer.length);
                                byte[] raw = signature.sign();
                                agent.setItiSignature(raw);
                                }
                        catch(Exception e) {}

                        master.setDestination(agent.getSite().length);
                        }
                }
        ;

codeStatement
                { String name = ""; }
        : "createCode" LPAREN name=expression COMMA procedure:IDENT RPAREN SEMI
                {
                String source =
```

```
                    (String)contentValue.elementAt(contentName.lastIndexOf((String)"_proc_"+
                    procedure.getText()));
                master.getMaster().getVerifier().createCode("code", agent.getCurrentSite(),
                    name, source, false);
                }
| "destroyCode" LPAREN name=expression RPAREN SEMI
        { master.getMaster().getVerifier().removeCode(agent.getCurrentSite(), name); }
;

comStatement    {
                String result = "";
                String part = "";
                String part2 = "";
                Vector receivers = new Vector();
                String message = "";
                boolean multicast = false;
                }
: LACC
  (
    part=expression
    (
        COMMA
        part2=expression
                { receivers.addElement(part2); }
    )*
  )?
  RACC
  (
    EXCL
    (
      EXCL      { multicast = true; }
    )?
    message=expression
                {
                if(multicast)
                    master.multicastMessage(agent.getIdentifier(), message);
                else {
                    master.sendMessage(agent.getIdentifier(), part, message);
                    Enumeration enumeration = receivers.elements();
                    while(enumeration.hasMoreElements())
                        master.sendMessage(agent.getIdentifier(),
                            (String)enumeration.nextElement(), message);
                    }
                }
  | INTR variable:IDENT
                {
                result = master.receiveMessage(part, agent.getIdentifier());
                contentValue.set(contentName.lastIndexOf((String)variable.getText()), (String)result);
                }
  )
  SEMI
;

controlStatement
                {
                String agentID = "";
```

```
                String itinerary = "";
                String message = "";
                String signer = "";
                String signerpass = "";
                }
: "pollAgent" LPAREN agentID=expression COMMA itinerary=expression RPAREN SEMI
                {
                String agentItinerary = master.getItinerary(this.agent.getIdentifier(), itinerary);

                Vector itineraryVector = new Vector();
                String element;
                for(
                        StringTokenizer stringtokenizer = new StringTokenizer(agentItinerary, "/",
                                false);
                        stringtokenizer.hasMoreTokens();
                        itineraryVector.addElement(element)
                        )
                                element = stringtokenizer.nextToken();

                Signature signature = null;
                KeyStore keystore = null;
                try {
                        signature = Signature.getInstance("DSA");
                        keystore = KeyStore.getInstance("JKS");
                        keystore.load(new FileInputStream(master.getMaster().getKeystore()),
                                master.getMaster().getStorepass().toCharArray());
                        signature.initSign((PrivateKey)keystore.getKey(
                                master.getMaster().
                                getPublicUser(),
                                master.getMaster().getPublicPass().toCharArray()));
                        }
                catch(Exception e) {}

                Itinerary itineraryClass = new Itinerary();

                int i = 0;
                String itineraries = "";
                itineraryClass.setNbSite(itineraryVector.size()/2);
                itineraryClass.setK(itineraryVector.size()/2);
                itineraryClass.setSite(itineraryVector.size()/2);
                Enumeration itineraryContent = itineraryVector.elements();
                while(itineraryContent.hasMoreElements()) {
                        itineraryClass.setK(i, fromString((String)itineraryContent.nextElement()));
                        element = (String)itineraryContent.nextElement();
                        itineraryClass.setSite(i, fromString(element));
                        itineraries += element;
                        i++;
                        }

                itineraryClass.setSigner(master.getMaster().getPublicUser());
                try {
                        byte[] buffer = itineraries.getBytes("UTF8");
                        signature.update(buffer, 0, buffer.length);
                        byte[] raw = signature.sign();
                        itineraryClass.setSignature(raw);
                        }
```

```
                catch(Exception e) {}

                master.sendIti(this.agent.getIdentifier(), agentID, "poll: null", itineraryClass);
                }
| "respondTo" LPAREN agentID=expression COMMA itinerary=expression RPAREN SEMI
                {
                String agentItinerary = master.getItinerary(this.agent.getIdentifier(), itinerary);

                Vector itineraryVector = new Vector();
                String element;
                for(
                        StringTokenizer stringtokenizer = new StringTokenizer(agentItinerary, "/",
                                false);
                        stringtokenizer.hasMoreTokens();
                        itineraryVector.addElement(element)
                        )
                                element = stringtokenizer.nextToken();

                Signature signature = null;
                KeyStore keystore = null;
                try {
                        signature = Signature.getInstance("DSA");
                        keystore = KeyStore.getInstance("JKS");
                        keystore.load(new FileInputStream(master.getMaster().getKeystore()),
                                master.getMaster().getStorepass().toCharArray());
                        signature.initSign((PrivateKey)keystore.getKey(
                                master.getMaster().getPublicUser(),
                                master.getMaster().getPublicPass().toCharArray()));
                        }
                catch(Exception e) {}

                Itinerary itineraryClass = new Itinerary();
                int i = 0;
                String itineraries = "";
                itineraryClass.setNbSite(itineraryVector.size()/2);
                itineraryClass.setK(itineraryVector.size()/2);
                itineraryClass.setSite(itineraryVector.size()/2);
                Enumeration itineraryContent = itineraryVector.elements();
                while(itineraryContent.hasMoreElements()) {
                        itineraryClass.setK(i, fromString((String)itineraryContent.nextElement()));
                        element = (String)itineraryContent.nextElement();
                        itineraryClass.setSite(i, fromString(element));
                        itineraries += element;
                        i++;
                        }

                itineraryClass.setSigner(master.getMaster().getPublicUser());
                try {
                        byte[] buffer = itineraries.getBytes("UTF8");
                        signature.update(buffer, 0, buffer.length);
                        byte[] raw = signature.sign();
                        itineraryClass.setSignature(raw);
                        }
                catch(Exception e) {}
```

```
                master.respondAgent(this.agent.getIdentifier(), agentID, "respondTo: null",
itineraryClass);
                }
 | "traceAgent"
  LPAREN
  agentID=expression COMMA
  message=expression
  (
   COMMA itinerary=expression
  )?
  RPAREN SEMI
                {
                if(itinerary.equals(""))
                        master.sendMessage(agent.getIdentifier(), agentID, message);
                else {
                        String agentItinerary = master.getItinerary(this.agent.getIdentifier(),
                                itinerary);
                        Vector itineraryVector = new Vector();
                        String element;
                        for(
                         StringTokenizer stringtokenizer = new StringTokenizer(agentItinerary, "/",
                                false);
                         stringtokenizer.hasMoreTokens();
                         itineraryVector.addElement(element)
                         )
                                element = stringtokenizer.nextToken();

                        Signature signature = null;
                        KeyStore keystore = null;
                        try {
                                signature = Signature.getInstance("DSA");
                                keystore = KeyStore.getInstance("JKS");
                                keystore.load(new
                                        FileInputStream(master.getMaster().getKeystore()),
                                        master.getMaster().getStorepass().toCharArray());
                                signature.initSign((PrivateKey)keystore.getKey(master.getMaster().
                                        getPublicUser(),
                                        master.getMaster().getPublicPass().toCharArray()));
                                }
                        catch(Exception e) {}

                        Itinerary itineraryClass = new Itinerary();
                        int i = 0;
                        String itineraries = "";
                        itineraryClass.setNbSite(itineraryVector.size()/2);
                        itineraryClass.setK(itineraryVector.size()/2);
                        itineraryClass.setSite(itineraryVector.size()/2);
                        Enumeration itineraryContent = itineraryVector.elements();
                        while(itineraryContent.hasMoreElements()) {
                          itineraryClass.setK(i, fromString((String)itineraryContent.nextElement()));
                          element = (String)itineraryContent.nextElement();
                          itineraryClass.setSite(i, fromString(element));
                          itineraries += element;
                          i++;
                          }
```

```
                        itineraryClass.setSigner(master.getMaster().getPublicUser());
                        try {
                                byte[] buffer = itineraries.getBytes("UTF8");
                                signature.update(buffer, 0, buffer.length);
                                byte[] raw = signature.sign();
                                itineraryClass.setSignature(raw);
                                }
                        catch(Exception e) {}

                        master.sendIti(this.agent.getIdentifier(), agentID, message, itineraryClass);
                        }
                }
 | "reportTo" LPAREN agentID=expression RPAREN SEMI
                { master.go(agent, fromString(agentID)); }
 ;

primitiveElement returns [String value=""]
                { String result; }
 : (IDENT LPAREN) => result=procedureCallStatement
                { value = result; }
 | result=constantValue
                { value = result; }
 | LPAREN result=expression RPAREN
                { value = result; }
 ;

booleanNegationExpression returns [String value=""]
                { boolean negation = false; }
 : (
    "not"       { if(negation) negation=false; else negation=true; }
   )*
   value=primitiveElement
                {
                if(negation && value.equals("true")) value = "false";
                else if(negation && value.equals("false")) value = "true";
                }
 ;

signExpression returns [String value=""]
                {
                int signe = 1;
                String value2 = "";
                }
 :
   ( MINUS      { signe = -signe; }
   | PLUS
   )*
   value2=booleanNegationExpression
                {
                if(isNumber(value2))
                        value = Integer.toString(signe * Integer.parseInt(value2));
                 else
                        value = value2;
                }
 ;
```

```
multiplyingExpression returns [String value=""]
                { String value2;}
 : value=signExpression
  ( TIMES        value2=signExpression
                { value = Integer.toString(Integer.parseInt(value) * Integer.parseInt(value2)); }
  | DIV   value2=signExpression
                { value = Integer.toString(Integer.parseInt(value) / Integer.parseInt(value2)); }
  | "mod"        value2=signExpression
                { value = Integer.toString(Integer.parseInt(value) % Integer.parseInt(value2)); }
  )*
 ;

addingExpression returns [String value=""]
                { String value2; }
 : value=multiplyingExpression
  ( PLUS         value2=multiplyingExpression
                {
                if(isNumber(value) && isNumber(value2))
                        value = Integer.toString(Integer.parseInt(value) + Integer.parseInt(value2));
                 else
                        value = value + value2;
                }
  | MINUS        value2=multiplyingExpression
                { value = Integer.toString(Integer.parseInt(value) - Integer.parseInt(value2)); }
  )*
 ;

relationalExpression returns [String value=""]
                { String value2; }
 : value=addingExpression
  (
   EQUALS       value2=addingExpression
                {
                if(isNumber(value) && isNumber(value2))
                        if(Integer.parseInt(value)==Integer.parseInt(value2)) value="true";
                        else value="false";
                 else
                        if(value.equals(value2)) value="true"; else value="false";
                }
  | NOT_EQUALS value2=addingExpression
                {
                        if(isNumber(value) && isNumber(value2))
                                if(Integer.parseInt(value)!=Integer.parseInt(value2)) value="true";
                                else value="false";
                        else
                                if(!value.equals(value2)) value="true"; else value="false";
                }
  | GT    value2=addingExpression
                { if(Integer.parseInt(value)>Integer.parseInt(value2)) value="true"; else value="false";
        }
  | GTE  value2=addingExpression
                { if(Integer.parseInt(value)>=Integer.parseInt(value2)) value="true"; else
        value="false"; }
  | LT    value2=addingExpression
                { if(Integer.parseInt(value)<Integer.parseInt(value2)) value="true"; else value="false";
        }
```

```
  | LTE  value2=addingExpression
                { if(Integer.parseInt(value)<=Integer.parseInt(value2)) value="true"; else
        value="false"; }
  )*
 ;

expression returns [String value=""]
                { String value2; }
 : value=relationalExpression
  ( "and"        value2=relationalExpression
                { if(value=="true" && value2=="true") value="true"; else value="false"; }
  | "or"   value2=relationalExpression
                { if(value=="true" || value2=="true") value="true"; else value="false"; }
  )*
 ;


class XLLexer extends Lexer;

options
                {
                charVocabulary = '\0'..'\377';
                testLiterals = false;
                k = 2;
                }

COMMENT
 : "//" (~('\n'|'\r'))*
  {$setType(Token.SKIP);}
 ;

protected DIGIT : '0'..'9'     ;
INT             : (DIGIT)+     ;
CHAR            : '\''! . '\''! ;
STRING
                : '"'!
                  ( '"' '"'!
                  | ~('"'|'\n'|'\r')
                  )*
                  ( '"'!
                  | // nothing -- write error message
                  )
                ;
DOT             : '.'       ;
BECOMES         : ":="      ;
COLON           : ':'                 ;
SEMI            : ';'       ;
COMMA           : ','       ;
EQUALS          : '='       ;
LBRACKET        : '['       ;
RBRACKET        : ']'       ;
DOTDOT          : ".."      ;
LPAREN          : '('       ;
RPAREN          : ')'       ;
NOT_EQUALS      : "/="      ;
```

```
LT      : '<'   ;
LTE             : "<="  ;
GT      : '>'   ;
GTE             : ">="  ;
PLUS            : '+'   ;
MINUS           : '-'   ;
TIMES           : '*'   ;
DIV     : '/'   ;
LACC            : '{'           ;
RACC            : '}'   ;
EXCL            : "!"           ;
INTR            : "?"           ;


WS
 : ( ' '
  | '\t'
  | '\f'

  // handle newlines
  | ( "\r\n"  // DOS
    | '\r'    // Macintosh
    | '\n'    // Unix
    )               { newline(); }
  )               { $setType(Token.SKIP); }
 ;

IDENT
 options {testLiterals=true;}
 : ('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_')*
 ;
```

c. Le système IBC-Go

```
package IBCGo;

import java.io.*;
import java.util.*;
import ants.NodeAddress;
import ants.Xdr;
import ants.ByteArray;
import ants.DataCapsule;

public class Agent implements java.io.Serializable {

        private short port;
        private String identifier;
        private int pLength = 0;
        private long pTime;
        private int origin;
        private int destination;
```

```
private String creator;
private int type;

private int credentials;
private byte[] creSignature;
private String creSigner;
private int nbCode;
private int nbItinerary;
private int[] codeType;

private int[] codeLength;
private byte[][] codeData;
private String[] codeSigner;
private byte[][] codeSignature;
private long[] interpreterAddress;
private String currentSite;
private int[] K;
private int[] site;
private String itiSigner;
private byte[] itiSignature;
private boolean keep;
private byte[] data;

// Getters & Setters
public short getPort() { return port; }
public void setPort(short s) { port = s; }
public String getIdentifier() { return identifier; }
public void setIdentifier(String s) { identifier = s; }
public int getPLength() { return pLength; }
public void setPLength(int i) { pLength = i; }
public long getPTime() { return pTime; }
public void setPTime(long i) { pTime = i; }
public int getOrigin() { return origin; }
public void setOrigin(int i) { origin = i; }
public int getDestination() { return destination; }
public void setDestination(int i) { destination = i; }
public String getCreator() { return creator; }
public void setCreator(String s) { creator = s; }
public int getType() { return type; }
public void setType(int i) { type = i; }
public int getCredentials() { return credentials; }
public void setCredentials(int i) { credentials = i; }
public byte[] getCreSignature() { return creSignature; }
public void setCreSignature(byte[] ba) { creSignature = ba; }
public String getCreSigner() { return creSigner; }
public void setCreSigner(String s) { creSigner = s; }
public int getNbCode() { return nbCode; }
public void setNbCode(int i) { nbCode = i; }
public int getNbItinerary() { return nbItinerary; }
public void setNbItinerary(int i) { nbItinerary = i; }
public int[] getCodeType() { return codeType; }
public int getCodeType(int i) { return codeType[i]; }
public void setCodeType(int[] i) { codeType = i; }
public void setCodeType(int i) { codeType = new int[i]; }
public void setCodeType(int i, int j) { codeType[i] = j; }
public int[] getCodeLength() { return codeLength; }
```

```
public int getCodeLength(int i) { return codeLength[i]; }
public void setCodeLength(int[] i) { codeLength = i; }
public void setCodeLength(int i) { codeLength = new int[i]; }
public void setCodeLength(int i, int j) { codeLength[i] = j; }
public byte[][] getCodeData() { return codeData; }
public byte[] getCodeData(int i) { return codeData[i]; }
public void setCodeData(byte[][] b) { codeData = b; }
public void setCodeData(int i) { codeData = new byte[i][]; }
public void setCodeData(int i, byte[] b) { codeData[i] = b; }
public String[] getCodeSigner() { return codeSigner; }
public String getCodeSigner(int i) { return codeSigner[i]; }
public void setCodeSigner(String[] s) { codeSigner = s; }
public void setCodeSigner(int i) { codeSigner = new String[i]; }
public void setCodeSigner(int i, String s) { codeSigner[i] = s; }
public byte[][] getCodeSignature() { return codeSignature; }
public byte[] getCodeSignature(int i) { return codeSignature[i]; }
public void setCodeSignature(byte[][] b) { codeSignature = b; }
public void setCodeSignature(int i) { codeSignature = new byte[i][]; }
public void setCodeSignature(int i, byte[] b) { codeSignature[i] = b; }
public long[] getInterpreterAddress() { return interpreterAddress; }
public long getInterpreterAddress(int i) { return interpreterAddress[i]; }
public void setInterpreterAddress(long[] l) { interpreterAddress = l; }
public void setInterpreterAddress(int i) { interpreterAddress = new long[i]; }
public void setInterpreterAddress(int i, long l) { interpreterAddress[i] = l; }
public String getCurrentSite() { return currentSite; }
public void setCurrentSite(String s) { currentSite = s; }
public int[] getK() { return K; }
public int getK(int i) { return K[i]; }
public void setK(int[] i) { K = i; }
public void setK(int i) { K = new int[i]; }
public void setK(int i, int j) { K[i] = j; }
public int[] getSite() { return site; }
public int getSite(int i) { return site[i]; }
public void setSite(int[] i) { site = i; }
public void setSite(int i) { site = new int[i]; }
public void setSite(int i, int j) { site[i] = j; }
public String getItiSigner() { return itiSigner; }
public void setItiSigner(String s) { itiSigner = s; }
public byte[] getItiSignature() { return itiSignature; }
public void setItiSignature(byte[] b) { itiSignature = b; }
public boolean getKeep() { return keep; }
public void setKeep(boolean b) { keep = b; }
public byte[] getData() { return data; }
public void setData(byte[] b) { data = b; }

public Agent() {}

public String getPayload() {
        ByteArray buf = new ByteArray(getData());
        Xdr xdr = new Xdr(buf,0);
        int nbLine = xdr.INT();
        String content = "";
        for (int i=0; i<nbLine; i++){
                content += xdr.STRING();
                if(i < nbLine-1) content += "\n";
                }
```

```
                return content;
                }

        public void setPayload(String[] content) {
                ByteArray  buf = new ByteArray(payloadLength(content));
                Xdr xdr = new Xdr(buf,0);
                xdr.PUT(content.length);
                for(int i=0; i<content.length; i++) xdr.PUT(content[i]);
                setData(buf.toBytes());
                }

        private int payloadLength(String[] content) {
                int length = Xdr.INT;
                for(int i=0; i<content.length; i++) length += Xdr.STRING(content[i]);
                return length;
                }
        }


package IBCGo;

import java.io.*;
import java.util.*;
import ants.NodeAddress;

public class Code implements java.io.Serializable {

        private int nbCode = 0;
        private int[] codeType;
        private int[] codeLength;
        private byte[][] codeData;
        private String[] signer;
        private byte[][] signature;
        private long[] interpreterAddress;

        // Getters & Setters
        public int getNbCode() { return nbCode; }
        public void setNbCode(int i) { nbCode = i; }
        public void setCodeType(int i) { codeType = new int[i]; }
        public int[] getCodeType() { return codeType; }
        public void setCodeType(int[] i) { codeType = i; }
        public int getCodeType(int i) { return codeType[i]; }
        public void setCodeType(int i, int j) { codeType[i] = j; }
        public void setCodeLength(int i) { codeLength = new int[i]; }
        public int[] getCodeLength() { return codeLength; }
        public void setCodeLength(int[] i) { codeLength = i; }
        public int getCodeLength(int i) { return codeLength[i]; }
        public void setCodeLength(int i, int j) { codeLength[i] = j; }
        public void setCodeData(int i) { codeData = new byte[i][]; }
        public byte[][] getCodeData() { return codeData; }
        public void setCodeData(byte[][] b) { codeData = b; }
        public byte[] getCodeData(int i) { return codeData[i]; }
        public void setCodeData(int i, byte[] b) { codeData[i] = b; }
        public void setSigner(int i) { signer = new String[i]; }
        public String[] getSigner() { return signer; }
        public String getSigner(int i) { return signer[i]; }
```

```
public void setSigner(String[] s) { signer = s; }
public void setSigner(int i, String s) { signer[i] = s; }
public void setSignature(int i) { signature = new byte[i][]; }
public byte[][] getSignature() { return signature; }
public byte[] getSignature(int i) { return signature[i]; }
public void setSignature(byte[][] b) { signature = b; }
public void setSignature(int i, byte[] b) { signature[i] = b; }
public long[] getInterpreterAddress() { return interpreterAddress; }
public void setInterpreterAddress(long[] l) { interpreterAddress = l; }
public long getInterpreterAddress(int i) { return interpreterAddress[i]; }
public void setInterpreterAddress(int i, long l) { interpreterAddress[i] = l; }

public Code() {}

public static Code fromXML(ComManager master, String fileAgent) {
        Code code = new Code();

        SAXParse saxparse = new SAXParse(fileAgent, "ibcgo:agent:code");
        Vector codes = saxparse.getResultat();
        Enumeration enumerationCode = codes.elements();

        Vector ids = new Vector();
        Vector codeFiles = new Vector();
        Vector signers = new Vector();
        Vector signatures = new Vector();
        Vector interpretors = new Vector();

        int nbCode = 0;
        while(enumerationCode.hasMoreElements()) {
                String param = (String)enumerationCode.nextElement();
                if(param.equals("id")) {
                        ids.addElement((String)enumerationCode.nextElement());
                        nbCode++;
                        }
                else if(param.equals("name"))
                        codeFiles.addElement((String)enumerationCode.nextElement());
                else if(param.equals("signer"))
                        signers.addElement((String)enumerationCode.nextElement());
                else if(param.equals("signature"))
                        signatures.addElement((String)enumerationCode.nextElement());
                else if(param.equals("interpretor"))
                        interpretors.addElement((String)enumerationCode.nextElement());
                else {
                        String garbage = (String)enumerationCode.nextElement();
                   }
                }

        int[] codeLength = new int[nbCode];
        int[] codeType = new int[nbCode];
        String[] codeName = new String[nbCode];
        String[] codeSigner = new String[nbCode];
        String[] codeSignature = new String[nbCode];
        String[] interpreterString = new String[nbCode];

        Enumeration enumerationId = ids.elements();
        Enumeration enumerationFile = codeFiles.elements();
```

```
Enumeration enumerationSigner = signers.elements();
Enumeration enumerationSignature = signatures.elements();
Enumeration enumerationInterpretor = interpretors.elements();

int CodeNo = 0;
byte[][] rawT1 = new byte[nbCode][];
byte[][] rawT2 = new byte[nbCode][];

long[] interpreterAddress = new long[nbCode];

int create = 0;
while(enumerationId.hasMoreElements()) {
        try {
                codeType[CodeNo] = 0;
                codeName[CodeNo] = (String)enumerationId.nextElement();
                rawT1[CodeNo] = codeName[CodeNo].getBytes("UTF8");
                codeSigner[CodeNo] = (String)enumerationSigner.nextElement();
                codeSignature[CodeNo] =
                        (String)enumerationSignature.nextElement();
                FileInputStream sigIn = new FileInputStream(fileAgent.substring(0,
                        fileAgent.lastIndexOf("/")+1)+"//"+codeSignature[CodeNo])
                ;
                rawT2[CodeNo] = new byte[sigIn.available()];
                sigIn.read(rawT2[CodeNo]);
                sigIn.close();
                interpreterString[CodeNo] =
                        (String)enumerationInterpretor.nextElement();
                interpreterAddress[CodeNo] =
                        (fromString(interpreterString[CodeNo].substring(0,
                        interpreterString[CodeNo].lastIndexOf(":")))*100000)+
                        Integer.parseInt(interpreterString[CodeNo].substring
                        (interpreterString[CodeNo].lastIndexOf(":")+1));
                }
catch (Exception e) {}
        String fileCode = fileAgent.substring(0,
          fileAgent.lastIndexOf("/")+1)+"//"+(String)enumerationFile.nextElement();
        create = master.getVerifier().createCode("file",
          NodeAddress.toString(master.thisNode().getAddress()),
codeName[CodeNo],
          fileCode, false);
        codeLength[CodeNo++] = create;
        }

if(create > 0) {
        code.setNbCode(nbCode);
        code.setCodeLength(codeLength);
        code.setCodeType(codeType);
        code.setCodeData(rawT1);
        code.setSigner(codeSigner);
        code.setSignature(rawT2);
        code.setInterpreterAddress(interpreterAddress);
        }

return code;
}
```

```
        private static int fromString(String name) {
                StringTokenizer st = new StringTokenizer(name, ".");

                int[] b = new int[4];
                try {
                        for (int i = 0; i < 4; i++)
                        b[i] = Integer.parseInt(st.nextToken());
                        }
                catch (Exception e) {}

                return ((b[0]<<24) | (b[1]<<16) | (b[2]<<8) | b[3]);
                }

        }
```

```
package IBCGo;

public class CodeUnit implements java.io.Serializable {

        private byte[] code = null;

        // Getters & Setters
        public byte[] getCode() { return code; }
        public void setCode(byte[] b) { code = b; }

        public CodeUnit() {}
        }
```

```
package IBCGo;

import ants.Application;
import ants.Capsule;
import ants.Protocol;
import ants.Node;
import ants.NodeAddress;
import ants.RouteTable;
import ants.Xdr;
import ants.ByteArray;
import utils.KeyArgs;
import java.io.*;
import java.util.Random;
import java.util.*;
import IBCGo.interpreter.*;
import java.security.*;

public class ComManager extends Application implements Runnable, java.io.Serializable {

        private int antsPort = 0;
        private int ibcport = 5000;
        private Database database = null;
        private String home = null;
        private String DB = null;
        private String migration = "fly";
        private String fileGroup = null;
```

```
private String fileAgent = "";
private String fileCredentials = null;
private Credentials credentials = null;
private String keystore = null;
private String storepass = null;
private String publicUser = null;
private String publicPass = null;
private boolean fileKeep = true;
private int[] sitesGroup = null;
private ComManagerControl comManagerControl = null;
private ExEnvironment exEnvironment = null;
private Verifier verifier = null;
private Vector mesSender = new Vector();
private Vector mesReceiver = new Vector();
private Vector mesContent = new Vector();

// Getters & Setters
public int getAntsPort() { return antsPort; }
public void setAntsPort(int i) { antsPort = i; }
public int getIbcport() { return ibcport; }
public void setIbcport(int i) { ibcport = i; }
public Database getDatabase() { return database; }
public void setDatabase(Database d) { database = d; }
public String getHome() { return home; }
public void setHome(String s) { home = s; }
public String getDB() { return DB; }
public void setDB(String s) { DB = s; }
public String getMigration() { return migration; }
public void setMigration(String s) { migration = s; }
public String getFileGroup() { return fileGroup; }
public void setFileGroup(String s) { fileGroup = s; }
public String getFileAgent() { return fileAgent; }
public void setFileAgent(String s) { fileAgent = s; }
public String getFileCredentials() { return fileCredentials; }
public void setFileCredentials(String s) { fileCredentials = s; }
public Credentials getCredentials() { return credentials; }
public void setCredentials(Credentials c) { credentials = c; }
public String getKeystore() { return keystore; }
public void setKeystore(String s) { keystore = s; }
public String getStorepass() { return storepass; }
public void setStorepass(String s) { storepass = s; }
public String getPublicUser() { return publicUser; }
public void setPublicUser(String s) { publicUser = s; }
public String getPublicPass() { return publicPass; }
public void setPublicPass(String s) { publicPass = s; }
public boolean getFileKeep() { return fileKeep; }
public void setFileKeep(boolean b) { fileKeep = b; }
public int[] getGroup() { return sitesGroup; }
public void setGroup(int[] i) { sitesGroup = i; }
public ComManagerControl getComManagerControl() { return comManagerControl; }
public void setComManagerControl(ComManagerControl cmc) { comManagerControl = cmc; }
public Verifier getVerifier() { return verifier; }
public void setVerifier(Verifier v) { verifier = v; }
public ExEnvironment getExEnvironment() { return exEnvironment; }
public void setExEnvironment(ExEnvironment ee) { exEnvironment = ee; }
public Vector getMesSender() { return mesSender; }
```

```
public void setMesSender(Vector ms) { mesSender = ms; }
public Vector getMesReceiver() { return mesReceiver; }
public void setMesReceiver(Vector mr) { mesReceiver = mr; }
public Vector getMesContent() { return mesContent; }
public void setMesContent(Vector mc) { mesContent = mc; }

public ComManager() throws Exception {}

public void sendPacket(String identifier, int type, long tCreation, int origin, int receiver,
                       int destination, Credentials credentials, Code code, Itinerary itinerary,
                       ByteArray buf, boolean keep) {
        IBCPacket ibcpacket = new IBCPacket(getPort(), getPort(), receiver, buf);
        //                    (Port source, Port destination, Adresse desination, payload, ...)

        ibcpacket.setPort(getPort());
        ibcpacket.setIdentifier(identifier);
        ibcpacket.setPTime(tCreation);
        ibcpacket.setOrigin(origin);
        ibcpacket.setDestination(destination);
        ibcpacket.setType(type);
        ibcpacket.setKeep(keep);

        if(type<6) {
                ibcpacket.setCreator(credentials.getCreator());
                ibcpacket.setCredentials(credentials.getCredentials());
                ibcpacket.setCreSignature(credentials.getSignature());
                ibcpacket.setCreSigner(credentials.getSigner());

                if(type!=3) {
                        ibcpacket.setNbCode(code.getNbCode());
                        ibcpacket.setCodeType(code.getCodeType());
                        ibcpacket.setCodeLength(code.getCodeLength());

                        ibcpacket.setCodeData(code.getCodeData());
                        ibcpacket.setCodeSigner(code.getSigner());
                        ibcpacket.setCodeSignature(code.getSignature());
                        ibcpacket.setInterpreterAddress(code.getInterpreterAddress());
                        }

                ibcpacket.setNbItinerary(itinerary.getNbSite());
                ibcpacket.setK(itinerary.getK());

                ibcpacket.setSite(itinerary.getSite());
                ibcpacket.setItiSigner(itinerary.getSigner());
                ibcpacket.setItiSignature(itinerary.getSignature());
                }
            else {
                ibcpacket.setCreator("null");
                ibcpacket.setCredentials(0);
                try {
                        ibcpacket.setCreSignature("null".getBytes("UTF8"));
                        }
                catch(Exception e) {}
                ibcpacket.setCreSigner("null");
                ibcpacket.setNbCode(0);
                ibcpacket.setNbItinerary(0);
```

```
                ibcpacket.setItiSigner("null");
                try {
                        ibcpacket.setItiSignature("null".getBytes("UTF8"));
                        }
                catch(Exception e) {}
                }

        // Type 1 or 2: Agent quit
        if(type<3) {
                // Retrait de l'agent dans le repertoire
                boolean remove =
                 verifier.removeAgent(NodeAddress.toString(thisNode().getAddress()),
        identifier);
                }
        ibcpacket.setCurrentSite(NodeAddress.toString(thisNode().getAddress()));
        send(ibcpacket);

        // Trace
        System.out.println("Envoi d'un agent " + identifier + " a l'adresse: " +
                NodeAddress.toString(receiver));
        database.createTrace("Envoi d"un agent " + identifier + " a l"adresse: " +
                NodeAddress.toString(receiver));
        }


public boolean parseBool(String s) {
if (s.equals("true")) return true;
return false;
        }

// Saisie les arguments incluent dans le fichier config (ANTS)
public void setArgs(KeyArgs k) throws Exception {
        k.merge(defaults);
        try {
                for (int i = 0; i < k.length(); i++) {
                        if (k.key(i).equals("-home")) {
                                home = k.arg(i);
                                k.strike(i);
                                }
                        else if (k.key(i).equals("-ibcport")) {
                                ibcport = Integer.parseInt(k.arg(i));
                                k.strike(i);
                                }
                        else if (k.key(i).equals("-DB")) {
                                DB = k.arg(i);
                                k.strike(i);
                                }
                        else if (k.key(i).equals("-migration")) {
                                migration = k.arg(i);
                                k.strike(i);
                                }
                        else if (k.key(i).equals("-fileIBCGo")) {
                                fileGroup = k.arg(i);
                                fileCredentials = k.arg(i);
                                k.strike(i);
                                }
```

```
                else if (k.key(i).equals("-fileAgent")) {
                        fileAgent = k.arg(i);
                        k.strike(i);
                        }
                else if (k.key(i).equals("-keystore")) {
                        keystore = k.arg(i);
                        k.strike(i);
                        }
                else if (k.key(i).equals("-storepass")) {
                        storepass = k.arg(i);
                        k.strike(i);
                        }
                else if (k.key(i).equals("-publicuser")) {
                        publicUser = k.arg(i);
                        k.strike(i);
                        }
                else if (k.key(i).equals("-publicpass")) {
                        publicPass = k.arg(i);
                        k.strike(i);
                        }
                else if (k.key(i).equals("-keep")) {
                        fileKeep = parseBool(k.arg(i));
                        k.strike(i);
                        }
                }
            }
        catch (Exception e) { }
        super.setArgs(k);
        }


// Point de départ pour ANTS -> Run
public void start() throws Exception {
        getNode().register(new IBCProtocol());          // Enregistre le protocole
        new Thread(this).start();                       // Demarre l'application

        }


public void run() {
        antsPort= getPort();

        // Demarre le centre de controle; accepte les commandes
        comManagerControl = new ComManagerControl(this, home, ibcport, DB);
        comManagerControl.start();

        // ...
        database = new Database(DB);

        // Instantie Verifier
        verifier = new Verifier();

        SAXParse saxparse = new SAXParse(fileGroup, "ibcgo:init:group:site");
        Vector site = saxparse.getResultat();
        int nbSite = saxparse.getNbElement();
        Enumeration enumerationSite = site.elements();
```

```
int SiteNo = 0;
sitesGroup = new int[nbSite];
while(enumerationSite.hasMoreElements()) {
        try {
                sitesGroup[SiteNo++] =
                  NodeAddress.fromString((String)enumerationSite.nextElement());
                }
        catch (Exception e) {}
        }

// Creation du contexte initial pour le noeud
boolean create =
  verifier.createAgentContext(NodeAddress.toString(thisNode().getAddress()));



// Saisie des credits associes a l'hote
credentials = Credentials.fromXML(fileCredentials);

try {
        if(!migration.equals("fly")) {
                ByteArray buf = new ByteArray(Xdr.INT);
                Xdr xdr = new Xdr(buf,0);
                xdr.PUT(0);
                sendPacket("pre-loading", 6, DateTime.getLong(),
                  thisNode().getAddress(), NodeAddress.fromString(migration),
                  NodeAddress.fromString(migration), credentials, null, null, buf,
                true);
                }
        }
catch(Exception e) {}

// If agent is defined
if(!fileAgent.equals("")) {
        Code code = Code.fromXML(this, fileAgent);
        Itinerary itinerary = Itinerary.fromXML(fileAgent);
        saxparse = new SAXParse(fileAgent, "ibcgo:agent");
        Vector identifiers = saxparse.getResultat();
        Enumeration enumerationIdentifier = identifiers.elements();
        String identifier = "";
        while(enumerationIdentifier.hasMoreElements()) {
                String element = (String)enumerationIdentifier.nextElement();
                if(element.equals("identifier")) {
                        identifier = (String)enumerationIdentifier.nextElement();
                        }
                else {
                        String garbage =
                                (String)enumerationIdentifier.nextElement();
                        }
                }

        saxparse = new SAXParse(fileAgent, "ibcgo:agent:payload:message");
        Vector content = saxparse.getResultat();
        ByteArray buf = new ByteArray(payloadLength(content));
        Xdr xdr = new Xdr(buf,0);
        xdr.PUT(content.size());
```

```
                Enumeration enumerationContent = content.elements();
                while(enumerationContent.hasMoreElements())
                        xdr.PUT((String)enumerationContent.nextElement());

                sendPacket(identifier, 1, DateTime.getLong(), thisNode().getAddress(),
                        itinerary.getSite(0), itinerary.getSite(itinerary.getSite().length-1),
                        credentials, code, itinerary, buf, fileKeep);
                }
        }


private int payloadLength(Vector content) {
        int length = Xdr.INT;
        Enumeration enumerationContent = content.elements();
        while(enumerationContent.hasMoreElements())
                length += Xdr.STRING((String)enumerationContent.nextElement());
        return length;
        }


// ANTS Reception
synchronized public void receive(Capsule unknownCap) {

        if (unknownCap instanceof IBCPacket) { // Verification du hache de la capsule
                IBCPacket ibcpacket = (IBCPacket)unknownCap;

                boolean ok = true;

                if(ibcpacket.getType() < 4) {
                        // Verification de la signature du champ d'autorisation
                        ok = verifier.verifySign(keystore, "qwerty",
                                ibcpacket.getCreSigner(),
                                ibcpacket.getCreator()+
                                Credentials.toString(ibcpacket.getCredentials()),
                                ibcpacket.getCreSignature());

                        // Verification de la signature de l'itineraire
                        if(ok) {
                                String iti = "";
                                for(int i=0; i<ibcpacket.getNbItinerary(); i++) iti +=
                                  NodeAddress.toString(ibcpacket.getSite(i));
                                ok = verifier.verifySign(keystore, "qwerty",
                                  ibcpacket.getItiSigner(), iti, ibcpacket.getItiSignature());
                                }
                        }

                // If OK
                if(ok) {
                        // Trace
                        System.out.println("Reception de l'agent " + ibcpacket.getIdentifier()
                                +
                                " - " + Integer.toString(ibcpacket.getType()));
                        database.createTrace("Reception de l"agent " +
                                ibcpacket.getIdentifier()
                                + " - " + Integer.toString(ibcpacket.getType()));
```

```
                switch(ibcpacket.getType()) {

case 1: // New agent reception
        // Emmagasiner l'agent dans le repertoire
          boolean create =
                verifier.createAgent(NodeAddress.toString
                (thisNode().getAddress()), ibcpacket.getIdentifier(), ibcpacket);
          //comManagerControl.print("____refresh");

          // Pour chaque section code
          boolean complet = true;
          for(int i=0; i<ibcpacket.getNbCode(); i++) {

            try {
                // Verifier si le code existe avec Verifier
                boolean exist =
                verifier.existCode(NodeAddress.toString(thisNode().
                getAddress()), new String(ibcpacket.getCodeData(i),"UTF8"));

                // Remplace le code si keep=false
                if(exist && !ibcpacket.getKeep() && migration.equals("fly")) {

                verifier.removeCode(NodeAddress.toString(thisNode().
                        getAddress()), new String(ibcpacket.getCodeData(i),
                        "UTF8"));
                        exist = false;
                        }

                // Demande pour le code manquant
                if(!exist) {
                  complet = false;
                  if(!migration.equals("fly"))
                                System.out.println("Non-existent code: "+
                                new String(ibcpacket.getCodeData(i), "UTF8"));
                  else {
                        Credentials credentials = new Credentials();
                        credentials.setCreator(ibcpacket.getCreator());

                        credentials.setCredentials(ibcpacket.getCredentials());
                        credentials.setSigner(ibcpacket.getCreSigner());


                        credentials.setSignature(ibcpacket.getCreSignature());
                                Code code = new Code();
                        code.setNbCode(1);
                        code.setCodeType(new int[1]);
                        code.setCodeType(0, 0);
                        code.setCodeLength(new int[1]);
                        code.setCodeLength(0, ibcpacket.getCodeLength(i));
                        code.setCodeData(new byte[1][]);
                        code.setCodeData(0, ibcpacket.getCodeData(i));
                        code.setSigner(new String[1]);
                        code.setSigner(0, ibcpacket.getCodeSigner(i));
                        code.setSignature(new byte[1][]);
                        code.setSignature(0, ibcpacket.getCodeSignature(i));
                        code.setInterpreterAddress(new long[1]);
```

```
                        code.setInterpreterAddress(0,
                                ibcpacket.getInterpreterAddress(i));

                        Itinerary itinerary = new Itinerary();
                        itinerary.setNbSite(ibcpacket.getNbItinerary());
                        itinerary.setK(ibcpacket.getK());
                        itinerary.setSite(ibcpacket.getSite());
                        itinerary.setSigner(ibcpacket.getItiSigner());
                        itinerary.setSignature(ibcpacket.getItiSignature());

                        sendPacket(ibcpacket.getIdentifier(), 4,
                                ibcpacket.getPTime(),
                          ibcpacket.getOrigin(),
                          NodeAddress.fromString(ibcpacket.getPreviousSite()),
                          NodeAddress.fromString(ibcpacket.getPreviousSite()),
                          credentials, code, itinerary, ibcpacket.getData(),
                          ibcpacket.getKeep());
                        }
                    }
                  }
                  catch(Exception e) {}
                  }

                // If complet -> execute agent
                if(complet) {
                        exEnvironment = new ExEnvironment();
                        exEnvironment.setMaster(this);
                        exEnvironment.setAgent(ibcpacket);
                        exEnvironment.start();
                        }
                break;

        case 2: // Follow agent without compute it
                exEnvironment = new ExEnvironment();
                exEnvironment.setMaster(this);
                exEnvironment.setAgent(ibcpacket);
                exEnvironment.follow(ibcpacket);
                break;

        case 3: // Follow message without compute it
                exEnvironment = new ExEnvironment();
                exEnvironment.setMaster(this);
                exEnvironment.setAgent(ibcpacket);
                exEnvironment.follow(ibcpacket);
                break;

        case 4: // Code demand
                // Take the demand code
                try {
                 Loader loader = new
                        Loader(NodeAddress.toString(thisNode().getAddress()));
                 byte[] codeData = loader.saisie(new String(ibcpacket.getCodeData(0),
                        "UTF8"));
                 Credentials credentials = new Credentials();
                 credentials.setCreator(ibcpacket.getCreator());
                 credentials.setCredentials(ibcpacket.getCredentials());
```

```
            credentials.setSigner(ibcpacket.getCreSigner());
            credentials.setSignature(ibcpacket.getCreSignature());

            Code code = new Code();
            code.setNbCode(2);
            code.setCodeType(new int[2]);
            code.setCodeType(0, 0);
            code.setCodeType(1, 1);
            code.setCodeLength(new int[2]);
            code.setCodeLength(0, ibcpacket.getCodeLength(0));
            code.setCodeLength(1, ibcpacket.getCodeLength(0));
            code.setCodeData(new byte[2][]);
            code.setCodeData(0, ibcpacket.getCodeData(0));
            code.setCodeData(1, codeData);
            code.setSigner(new String[2]);
            code.setSigner(0, ibcpacket.getCodeSigner(0));
            code.setSigner(1, ibcpacket.getCodeSigner(0));
            code.setSignature(new byte[2][]);
            code.setSignature(0, ibcpacket.getCodeSignature(0));
            code.setSignature(1, ibcpacket.getCodeSignature(0));
            code.setInterpreterAddress(new long[2]);
            code.setInterpreterAddress(0, ibcpacket.getInterpreterAddress(0));
            code.setInterpreterAddress(1, ibcpacket.getInterpreterAddress(0));

            Itinerary itinerary = new Itinerary();
            itinerary.setNbSite(ibcpacket.getNbItinerary());
            itinerary.setK(ibcpacket.getK());
            itinerary.setSite(ibcpacket.getSite());
            itinerary.setSigner(ibcpacket.getItiSigner());
            itinerary.setSignature(ibcpacket.getItiSignature());

            sendPacket(ibcpacket.getIdentifier(), 5, ibcpacket.getPTime(),
              ibcpacket.getOrigin(),
              NodeAddress.fromString(ibcpacket.getPreviousSite()),
              NodeAddress.fromString(ibcpacket.getPreviousSite()), credentials, code,
              itinerary, ibcpacket.getData(), ibcpacket.getKeep());
            }
            catch(Exception e) {}
            break;

case 5: // Code reception
            try {
            // Verification de la signature du code

            ok = verifier.verifySign(keystore, "qwerty", ibcpacket.getCodeSigner(0),
                    new String(ibcpacket.getCodeData(1), "UTF8"),
                    ibcpacket.getCodeSignature(0));

            if(ok) {
              // Emmagasine le code dans le repertoire
              int createCode = verifier.createCode("code",
                    NodeAddress.toString(thisNode().getAddress()),
                    new String(ibcpacket.getCodeData(0), "UTF8"),
                    new String(ibcpacket.getCodeData(1), "UTF8"),
                    ibcpacket.getKeep());
```

```
            if(!ibcpacket.getIdentifier().equals("pre-loading")) {
                // Saisie l'agent dans le repertoire
                IBCPacket agent =
                        verifier.getAgent(NodeAddress.toString(thisNode().
                        getAddress()), ibcpacket.getIdentifier());

                // Verifier si le code est complet
                complet = true;
                for(int i=0; i<agent.getNbCode(); i++) {
                  boolean exist =
                        verifier.existCode(NodeAddress.toString(thisNode().
                        getAddress()), new String(agent.getCodeData(i), "UTF8"));
                  if(!exist) complet = false;
                  }

                // If complet -> execute agent
                if(complet) {
                        exEnvironment = new ExEnvironment();
                        exEnvironment.setMaster(this);
                        exEnvironment.setAgent(agent);
                        exEnvironment.start();
                        }
                }
              }
            }
            catch(Exception e) {}
            break;

case 6: // Pre-loading
            Directory directory = new Directory();
            Vector codes =
                    directory.getCodeList(NodeAddress.toString(thisNode().getAddress
            ()));

            Loader loader = new
                    Loader(NodeAddress.toString(thisNode().getAddress()));

            Enumeration enumerationCode = codes.elements();
            while(enumerationCode.hasMoreElements()) {
            try {
              String name = (String)enumerationCode.nextElement();
              name = name.substring(name.lastIndexOf("=")+1);
              byte[] codeData = loader.saisie(name);
              String interpreterString =
                      NodeAddress.toString(thisNode().getAddress())+":"+
                      Integer.toString(getPort());

              Credentials credentials = new Credentials();
              credentials.setCreator(ibcpacket.getCreator());
              credentials.setCredentials(ibcpacket.getCredentials());
              credentials.setSigner(ibcpacket.getCreSigner());
              credentials.setSignature(ibcpacket.getCreSignature());

              Signature signature = null;
              KeyStore keystore2 = null;
              try {
```

```
    signature = Signature.getInstance("DSA");
    keystore2 = KeyStore.getInstance("JKS");
    keystore2.load(new FileInputStream(getKeystore()),
          getStorepass().toCharArray());
    signature.initSign((PrivateKey)keystore2.getKey(publicUser,
         publicPass.toCharArray()));
    }
  catch(Exception e) {}

  Code code = new Code();
  code.setNbCode(2);
  code.setCodeType(new int[2]);
  code.setCodeType(0, 0);
  code.setCodeType(1, 1);
  code.setCodeLength(new int[2]);
  code.setCodeLength(0, codeData.length);
  code.setCodeLength(1, codeData.length);
  code.setCodeData(new byte[2][]);
  code.setCodeData(0, name.getBytes("UTF8"));
  code.setCodeData(1, codeData);
  code.setSigner(new String[2]);
  code.setSigner(0, publicUser);
  code.setSigner(1, publicUser);
  code.setSignature(new byte[2][]);

  byte[] raw = null;
  try {
            signature.update(codeData, 0, codeData.length);
    raw = signature.sign();
    }
  catch(Exception e) {}

  code.setSignature(0, raw);
  code.setSignature(1, raw);
  code.setInterpreterAddress(new long[2]);
  code.setInterpreterAddress(0,
        (NodeAddress.fromString(interpreterString.substring(0,
        interpreterString.lastIndexOf(":")))*100000)+Integer.parseInt(interpr
eter
        String.substring(interpreterString.lastIndexOf(":")+1)));
  code.setInterpreterAddress(1,
        (NodeAddress.fromString(interpreterString.substring(0,
        interpreterString.lastIndexOf(":")))*100000)+Integer.parseInt(interpr
eter
        String.substring(interpreterString.lastIndexOf(":")+1)));

  Itinerary itinerary = new Itinerary();
  itinerary.setNbSite(0);
  itinerary.setSigner(publicUser);
  try {
    byte[] buffer = "".getBytes("UTF8");
    signature.update(buffer, 0, buffer.length);
    raw = signature.sign();
    itinerary.setSignature(raw);
    }
  catch(Exception e) {}
```

```
            sendPacket("pre-loading", 5, ibcpacket.getPTime(), ibcpacket.getOrigin(),
                    NodeAddress.fromString(ibcpacket.getPreviousSite()),
                    NodeAddress.fromString(ibcpacket.getPreviousSite()), credentials,
                    code, itinerary, ibcpacket.getData(), ibcpacket.getKeep());
            }
          catch(Exception e) {}
          break;

case 7: // agentAt
          String response = "";
          directory = new Directory();
          Vector sites =
                    directory.getAgentList(NodeAddress.toString(thisNode().getAddres
          s()));
          if(sites.isEmpty()) response = "null";
          Enumeration enumerationSites = sites.elements();
          while(enumerationSites.hasMoreElements()) {
            if(response.equals(""))
              response = (String)enumerationSites.nextElement();
            else
              response += ";" + (String)enumerationSites.nextElement();
            }
          ByteArray buf = new ByteArray(Xdr.INT + Xdr.STRING(response));
          Xdr xdrTemp = new Xdr(buf,0);
          xdrTemp.PUT(1);
          xdrTemp.PUT(response);
          try {
            sendPacket("agentAtResponse", 8, ibcpacket.getPTime(),
            thisNode().getAddress(), ibcpacket.getOrigin(), ibcpacket.getOrigin(), null,
          null,
            null, buf, true);
            }
          catch(Exception e) {}
          break;

case 8: // agentAtResponse
          exEnvironment.setAgentList(ibcpacket.getPayload());
          break;

case 9: // message
          String payload = ibcpacket.getPayload();
          Vector message = new Vector();
          String element;
          for(
            StringTokenizer stringtokenizer = new StringTokenizer(payload, "\n",
                    false);
            stringtokenizer.hasMoreTokens();
            message.addElement(element)
            )
                    element = stringtokenizer.nextToken();
          Enumeration enumerationEleMessage = message.elements();
          mesSender.add((String)enumerationEleMessage.nextElement());
          mesReceiver.add((String)enumerationEleMessage.nextElement());
          mesContent.add((String)enumerationEleMessage.nextElement());
          break;
```

```
case 10: // agentOrigin & agentAddress & agentItinerary
        String here = "";
        payload = ibcpacket.getPayload();
        message = new Vector();
        for(
          StringTokenizer stringtokenizer = new StringTokenizer(payload, "\n",
                false);
          stringtokenizer.hasMoreTokens();
          message.addElement(element)
          )
                element = stringtokenizer.nextToken();
        enumerationEleMessage = message.elements();
        String sender = (String)enumerationEleMessage.nextElement();
        String receiver = (String)enumerationEleMessage.nextElement();
        String content = (String)enumerationEleMessage.nextElement();

        directory = new Directory();
        sites =
          directory.getAgentList(NodeAddress.toString(thisNode().getAddress()));
        enumerationSites = sites.elements();
        while(enumerationSites.hasMoreElements()) {
          element = (String)enumerationSites.nextElement();
          if(element.equals(receiver)) {
            if(content.equals("origin")) {
                here += "originResponse:";
                IBCPacket agent =
                        verifier.getAgent(NodeAddress.toString(thisNode().
                        getAddress()), element);
                here += NodeAddress.toString(agent.getOrigin());
                }
            else if(content.equals("address")) {
                here =

                "addressResponse:"+NodeAddress.toString(thisNode().getAddress
                ());
                }
            else {
                here += "itineraryResponse:";
                IBCPacket agent =
                  verifier.getAgent(NodeAddress.toString(thisNode().getAddress()),
                  element);
                for(int i=0;i<agent.getSite().length;i++) {
                  here += Integer.toString(agent.getK(i)) + "/" +
                  NodeAddress.toString(agent.getSite(i));
                if(i!=agent.getSite().length-1) here += "/";
                }
            }
          }
        }

        if(!here.equals("")) {
          buf = new ByteArray(Xdr.INT + Xdr.STRING(receiver) +
                Xdr.STRING(sender) +
                Xdr.STRING(here));
          xdrTemp = new Xdr(buf,0);
```

```
                                xdrTemp.PUT(3);
                                xdrTemp.PUT(receiver);
                                xdrTemp.PUT(sender);
                                xdrTemp.PUT(here);
                                try {
                                  sendPacket("message", 9, ibcpacket.getPTime(),
                                        thisNode().getAddress(),
                                        ibcpacket.getOrigin(), ibcpacket.getOrigin(), null, null, null, buf,
                              true);
                                  }
                                catch(Exception e) {}
                                }

                              }
                        }
                  }
            }
      }
```

```
package IBCGo;

import java.io.*;
import java.net.*;
import java.util.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import IBCGo.authentification.RdbmsLoginModule;
import IBCGo.authentification.PassiveCallbackHandler;
import javax.security.auth.login.LoginContext;
import ants.NodeAddress;
import ants.Xdr;
import ants.ByteArray;

public class ComManagerControl extends Thread implements java.io.Serializable {
        private ServerSocket serverSocket;                  // Interne
        private ComConnexion comConnexion = null;           // Interne
        private ComManager master = null;
        private String home = null;
        private int ibcport = 0;
        private String DB = null;
        private boolean actif = false;

        // Getters & Setters
        public ComManager getMaster() { return master; }
        public void setMaster(ComManager cm) { master = cm; }
        public String getHome() { return home; }
        public void setHome(String s) { home = s; }
        public int getIBCPort() { return ibcport; }
        public void setIBCPort(int i) { ibcport = i; }
        public String getDB() { return DB; }
        public void setDB(String s) { DB = s; }
        public boolean getActif() { return actif; }
        public void setActif(boolean b) { actif = b; }
```

```
public ComManagerControl() {}

public ComManagerControl(ComManager master, String home, int ibcport, String DB) {
        this.master = master;
        this.home = home;
        this.ibcport = ibcport;
        this.DB = DB;
        }

// Reception
public void run() {
        try {
                this.serverSocket = new ServerSocket(ibcport);
                actif = true;
                while (actif) {
                        comConnexion = new ComConnexion(this.home, this.DB, this,
                                this.serverSocket.accept());
                        comConnexion.start();
                        System.out.println("nouvelle connexion");
                        }
                serverSocket.close();
                }
        catch(IOException ex) {}
        }

// Envoi d'un message au client
public void print(String s) {
        if(comConnexion != null) {
                try {
                        DataOutputStream out = comConnexion.getOut();
                        Cipher cipher = comConnexion.getCipher();
                        cipher.init(Cipher.ENCRYPT_MODE, comConnexion.getKey());
                        byte[] stringBytes = s.getBytes("UTF8");
                        byte[] raw = cipher.doFinal(stringBytes);
                        out.writeInt(raw.length);
                        out.write(raw);
                        }
                catch(Exception e) {}
                }
        }

// Envoi d'un saut de ligne au client (encrypte)
public void println(String s) {
        if(comConnexion != null) {
                try {
                        s = "\r\n";
                        DataOutputStream out = comConnexion.getOut();
                        Cipher cipher = comConnexion.getCipher();
                        cipher.init(Cipher.ENCRYPT_MODE, comConnexion.getKey());
                        byte[] stringBytes = s.getBytes("UTF8");
                        byte[] raw = cipher.doFinal(stringBytes);
                        out.writeInt(raw.length);
                        out.write(raw);
                        }
                catch(Exception e) {}
```

```
                    }
                }
            }

class ComConnexion extends Thread implements java.io.Serializable {
        private ComManagerControl master = null;   // Interne
        private Socket clientSocket = null;        // Interne
        private String home = null;
        private String DB = null;
        private DataInputStream in = null;
        private DataOutputStream out = null;
        private Cipher cipher = null;
        private SecretKey key = null;
        private boolean actif = false;

        // Getters & Setters
        public String getHome() { return home; }
        public void setHomet(String s) { home = s; }
        public String getDB() { return DB; }
        public void setDB(String s) { DB = s; }
        public DataInputStream getIn() { return in; }
        public void setIn(DataInputStream dis) { in = dis; }
        public DataOutputStream getOut() { return out; }
        public void setOut(DataOutputStream dos) { out = dos; }
        public Cipher getCipher() { return cipher; }
        public void setCipher(Cipher c) { cipher = c; }
        public SecretKey getKey() { return key; }
        public void setKey(SecretKey sk) { key = sk; }
        public boolean getActif() { return actif; }
        public void setActif(boolean b) { actif = b; }

        public ComConnexion() {}

        public ComConnexion (String home, String DB, ComManagerControl master, Socket
          clientSocket)
          throws SocketException {
                this.home = home;
                this.DB = DB;
                this.master = master;
                this.clientSocket = clientSocket;
                setPriority(NORM_PRIORITY - 1);
                }

        // Etablissement de la connexion avec le client
        public void run() {
                try {
                  in = new DataInputStream(new
                        BufferedInputStream(clientSocket.getInputStream()));
                  OutputStream outs = clientSocket.getOutputStream();
                  out = new DataOutputStream(outs);

                  StringTokenizer stringtokenizer = new StringTokenizer(DB, ",", false);
                  String type = stringtokenizer.nextToken();
                  System.setProperty("java.security.auth.login.config",
                        home+"//IBCGo//db//"+type+".config");
```

```
// Generation d'une paire de cle privee/publique pour le protcole Diffie-Hellmann
// Utilise 'Skip' pour obtenir g et n (1024 bits)
Security.addProvider(new com.sun.crypto.provider.SunJCE());
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DH");
kpg.initialize(Skip.sDHParameterSpec);
KeyPair keypair = kpg.genKeyPair();

// Recoit la cle publique du client
byte[] keyBytes = new byte[in.readInt()];
in.readFully(keyBytes);

KeyFactory kf = KeyFactory.getInstance("DH");
X509EncodedKeySpec x509Spec = new X509EncodedKeySpec(keyBytes);
PublicKey theirPublicKey = kf.generatePublic(x509Spec);

// Envoie sa cle publique au client
keyBytes = keypair.getPublic().getEncoded();
out.writeInt(keyBytes.length);
out.write(keyBytes);

// Generation de le secret partage
KeyAgreement ka = KeyAgreement.getInstance("DH");
ka.init(keypair.getPrivate());
ka.doPhase(theirPublicKey, true);
byte[] secret = ka.generateSecret();

// Encryption avec le secret partage
SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
DESKeySpec desSpec = new DESKeySpec(secret);
key = skf.generateSecret(desSpec);

// Construction d'un cipher en mode ECB
cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

// Reception de l'identite et du mot-de-passe du client (encryptes)
cipher.init(Cipher.DECRYPT_MODE, key);
byte[] raw = new byte[in.readInt()];
in.readFully(raw);
byte[] stringBytes = cipher.doFinal(raw);
String user = new String(stringBytes, "UTF8");
raw = new byte[in.readInt()];
in.readFully(raw);
stringBytes = cipher.doFinal(raw);
String pass = new String(stringBytes, "UTF8");

// Validation de l'identite et du mot-de-passe du client (JAAS)
PassiveCallbackHandler consoleCallbackHandler =
        new PassiveCallbackHandler(user, pass);
LoginContext loginContext = new LoginContext("IBCGo", consoleCallbackHandler);
loginContext.login();
loginContext.logout();

// OK
master.print("ready");
master.println("");
actif = true;
```

```
            Listen();
            }
          catch (Exception e) { System.out.println(e); }
          }

private void Listen() {
  try {
    Cipher cipherListen = Cipher.getInstance("DES/ECB/PKCS5Padding");

    while (actif) {
          cipherListen.init(Cipher.DECRYPT_MODE, key);
          byte[] raw = new byte[in.readInt()];
          in.readFully(raw);
          byte[] stringBytes = cipherListen.doFinal(raw);
          String message = new String(stringBytes, "UTF8");
          if(message.substring(0, message.indexOf(":")).equals("send")) {
                    String object = message.substring(message.indexOf(":")+1);
                    String identifier = object.substring(0, object.indexOf(";"));
                    boolean keep = true;
                    if(object.substring(object.indexOf(";")+1).equals("false")) keep = false;

                    IBCPacket agent =
                              master.getMaster().getVerifier().getAgent(NodeAddress.toString
                              (master.getMaster().thisNode().getAddress()), identifier);

                    Credentials credentials = null;
                    Code code = new Code();
                    Itinerary itinerary = new Itinerary();

                    if(agent.getCredentials() == 0) {
                      Signature signature = null;
                      KeyStore keystore = null;
                      try {
                        signature = Signature.getInstance("DSA");
                        keystore = KeyStore.getInstance("JKS");
                        keystore.load(new FileInputStream(master.getMaster().getKeystore()),
                              master.getMaster().getStorepass().toCharArray());


                        signature.initSign((PrivateKey)keystore.getKey(agent.getItiSigner().
                              substring(0,agent.getItiSigner().indexOf(":")),
                              agent.getItiSigner().substring(agent.getItiSigner().indexOf(":")+1).
                              toCharArray()));
                        }
                      catch(Exception e) {}

                      credentials = master.getMaster().getCredentials();

                      code.setSignature(agent.getNbCode());
                      for(int i=0; i<agent.getNbCode(); i++) {
                        Loader loader = new
                              Loader(NodeAddress.toString(master.getMaster().thisNode().
                              getAddress()));
                        byte[] buffer = loader.saisie(new String(agent.getCodeData(i), "UTF8"));
                        code.setCodeLength(agent.getNbCode());
                        code.setCodeLength(i, buffer.length);
```

```
                    try {
                      signature.update(buffer, 0, buffer.length);
                      raw = signature.sign();
                      }
                    catch(Exception e) {}
                    code.setSignature(i, raw);
                    }
                  String itineraries = "";
                  for(int i=0; i<agent.getNbItinerary(); i++)
                        itineraries += NodeAddress.toString(agent.getSite(i));

                  byte[] buffer = itineraries.getBytes("UTF8");
                  try {
                    signature.update(buffer, 0, buffer.length);
                    raw = signature.sign();
                    }
                  catch(Exception e) {}
                  itinerary.setSignature(raw);
                        itinerary.setSigner(agent.getItiSigner().substring(0,
                      agent.getItiSigner().indexOf(":")));
                  }
                else {
                  credentials = new Credentials();
                  credentials.setCreator(agent.getCreator());
                  credentials.setCredentials(agent.getCredentials());
                  credentials.setSigner(agent.getCreSigner());
                  credentials.setSignature(agent.getCreSignature());
                  code.setCodeLength(agent.getCodeLength());
                  code.setSignature(agent.getCodeSignature());
                  itinerary.setSignature(agent.getItiSignature());
                  itinerary.setSigner(agent.getItiSigner());
                  }

                code.setNbCode(agent.getNbCode());
                code.setCodeType(agent.getCodeType());
                code.setCodeData(agent.getCodeData());
                code.setSigner(agent.getCodeSigner());
                code.setInterpreterAddress(agent.getInterpreterAddress());

                itinerary.setNbSite(agent.getNbItinerary());
                itinerary.setK(agent.getK());
                itinerary.setSite(agent.getSite());

                master.getMaster().sendPacket(agent.getIdentifier(), 1, agent.getPTime(),
                        master.getMaster().thisNode().getAddress(), agent.getSite(0),
                        agent.getSite(itinerary.getSite().length-1), credentials, code,
                itinerary,
                        agent.getData(), keep);

                }
          }
        }
      catch(Exception e) {}
      }

}
```

```
package IBCGo;

import java.io.*;
import java.util.*;

public class Credentials implements java.io.Serializable {

        private String creator = null;
        private int credentials = 0;
        private String signer = null;
        private byte[] signature = null;

        public String getCreator() { return creator; }
        public void setCreator(String s) { creator = s; }
        public int getCredentials() { return credentials; }
        public void setCredentials(int i) { credentials = i; }
        public String getSigner() { return signer; }
        public void setSigner(String s) { signer = s; }
        public byte[] getSignature() { return signature; }
        public void setSignature(byte[] b) { signature = b; }

        public Credentials() {}

        public static void main( String[] args ) throws IOException {
                        System.out.println(Credentials.toString(Integer.parseInt(args[0])));
                }

        public static int fromString(String s) {
                int i = 0;

                for(int j=31; j>=0; j--) {
                        if(s.charAt(j) == '1') i += Math.pow(2,31-j);
                        }

                return i;
                }

        public static String toString(int i) {
                String s = "";

                for(int j=0; j<=31; j++)
                        if(i >= Math.pow(2,30-j)) {
                                i = i - (int)Math.pow(2,30-j);
                                s += 1;
                                }
                          else
                                s += 0;

                return s;
                }

        public static String toString(Byte b) {
                int byte1Int = b.intValue();
```

```
        int bit1Int = byte1Int & 1;
        int bit2Int = (byte1Int>>1) & 1;
        int bit3Int = (byte1Int>>2) & 1;
        int bit4Int = (byte1Int>>3) & 1;
        int bit5Int = (byte1Int>>4) & 1;
        int bit6Int = (byte1Int>>5) & 1;
        int bit7Int = (byte1Int>>6) & 1;
        int bit8Int = (byte1Int>>7) & 1;

        String bit1 = Integer.toString(bit1Int);
        String bit2 = Integer.toString(bit2Int);
        String bit3 = Integer.toString(bit3Int);
        String bit4 = Integer.toString(bit4Int);
        String bit5 = Integer.toString(bit5Int);
        String bit6 = Integer.toString(bit6Int);
        String bit7 = Integer.toString(bit7Int);
        String bit8 = Integer.toString(bit8Int);

        String byte1 = bit8 + bit7 + bit6 + bit5 + bit4 + bit3 + bit2 + bit1;

        return byte1;
        }

public static Credentials fromXML(String fileCredentials) {
        Credentials credentials = new Credentials();

        SAXParse saxparse = new SAXParse(fileCredentials, "ibcgo:init:owner");
        Vector resultat = saxparse.getResultat();
        Enumeration enumerationResultat = resultat.elements();
        credentials.setCreator((String)enumerationResultat.nextElement());

        saxparse = new SAXParse(fileCredentials, "ibcgo:init:credentials");
        resultat = saxparse.getResultat();
        enumerationResultat = resultat.elements();

        credentials.setCredentials(Credentials.fromString
                ((String)enumerationResultat.nextElement()));

        saxparse = new SAXParse(fileCredentials, "ibcgo:init:signer");
        resultat = saxparse.getResultat();
        enumerationResultat = resultat.elements();
        credentials.setSigner((String)enumerationResultat.nextElement());

        saxparse = new SAXParse(fileCredentials, "ibcgo:init:signature");
        resultat = saxparse.getResultat();
        enumerationResultat = resultat.elements();
        String credentialsSignatureFile = (String)enumerationResultat.nextElement();
        byte[] raw = null;
        try {
                FileInputStream sigIn = new FileInputStream(fileCredentials.substring(0,
                        fileCredentials.lastIndexOf("/")+1)+"//"+credentialsSignatureFile);
                raw = new byte[sigIn.available()];
                sigIn.read(raw);
                sigIn.close();
                }
        catch (Exception e) {}
```

```
                credentials.setSignature(raw);

                return credentials;
                }

        }


package IBCGo;

import java.sql.*;
import java.util.*;

public class Database implements java.io.Serializable {

        private String type = null;
        private String name = null;
        private String ID = null;
        private String pass = null;
        private String DBUrl = null;
        private Connexion connexion = null;

        // Getters & Setters
        public String getType() { return type; }
        public void setType(String s) { type = s; }
        public String getName() { return name; }
        public void setName(String s) { name = s; }
        public String getID() { return ID; }
        public void setID(String s) { ID = s; }
        public String getPass() { return pass; }
        public void setPass(String s) { pass = s; }
        public String getDBUrl() { return DBUrl; }
        public void setDBUrl(String s) { DBUrl = s; }

        public Database(String DB) {
                StringTokenizer stringtokenizer = new StringTokenizer(DB, ",", false);
                this.type = stringtokenizer.nextToken();
                this.name = stringtokenizer.nextToken();
                this.ID = stringtokenizer.nextToken().trim();
                this.pass = stringtokenizer.nextToken().trim();

                try {
                        if(this.type.equals("ODBC")) {
                                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                                DBUrl = "jdbc:odbc:"+name;
                                }
                        else if(this.type.equals("SQLServer")) {
                                Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
                                DBUrl =
                                  "jdbc:microsoft:sqlserver://192.168.1.100:1433;DatabaseName="
                                  +name;
                                }
                        else if(this.type.equals("MySQL")) {
                                Class.forName("org.gjt.mm.mysql.Driver");
                                DBUrl = "jdbc:mysql://localhost/"+name;
                                }
```

```
                else if(this.type.equals("Postgres")) {
                        Class.forName("org.postgresql.Driver");
                        DBUrl = "jdbc:postgresql://localhost/"+name.trim();
                        }
                else if(this.type.equals("Oracle")) {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        DBUrl = "jdbc:oracle:thin:@localhost:1521:"+name;
                        }
                }
        catch(Exception e) {}
        }

public static void main(String[] args) {
        Database database = new Database(args[0]+","+args[1]+","+args[2]+","+args[3]);
        database.createDB();
        }

private void open() throws Exception {
        connexion = DriverManager.getConnexion(DBUrl, this.ID, this.pass);
        }

private void close() throws Exception {
        connexion.close();
        }

private void createDB() {
        try {
                this.open();

                // Pas permis sur Arabica
                // statement.executeUpdate("CREATE DATABASE IBCGo");

                // Creation des tables
                PreparedStatement preparedStatement =
                        connexion.prepareStatement
                        ("CREATE TABLE ibcgoMemo (ID char(16), Message
                varchar(255))");
                preparedStatement.executeUpdate();
                preparedStatement =
                        connexion.prepareStatement
                        ("CREATE TABLE ibcgoTrace (Times varchar(50), Event
                        varchar(255))");
                preparedStatement.executeUpdate();
                preparedStatement =
                        connexion.prepareStatement
                        ("CREATE TABLE ibcgoUser (userid varchar(50), password
                        varchar(50), salt varchar(255))");
                preparedStatement.executeUpdate();

                // Creation d'un usager par defaut: admin, pass
                preparedStatement =
                        connexion.prepareStatement
                        ("INSERT INTO ibcgoUser VALUES
                        ('admin','Uty4EJMeIPeqL0mzUQ04BQ==','12345')");
                preparedStatement.executeUpdate();
```

```
                preparedStatement.close();
                this.close();
                }
        catch(Exception e) { System.out.println(e); }
        }

public void createUser(String userid, String password, String salt) {
        try {
                this.open();
                Statement statement = connexion.createStatement();
                statement.executeUpdate
                        ("INSERT INTO [ibcgoUser] (userid, password, salt)
                        VALUES ('"+userid+"', '"+password+"', '"+salt+"')");
                statement.close();
                this.close();
                }
        catch (Exception e) {}
        }

public void createTrace(String event) {
        try {
                this.open();
                Statement statement = connexion.createStatement();
                statement.executeUpdate
                        ("INSERT INTO ibcgoTrace (Times, Event)
                        VALUES ('"+DateTime.getString()+"', '"+event+"')");
                statement.close();
                this.close();
                }
        catch (Exception e) {}
        }

public void createMessage(String id, String message) {
        try {
                this.open();
                Statement statement = connexion.createStatement();
                ResultSet resultset = statement.executeQuery
                        ("SELECT * FROM ibcgoMemo WHERE ID='"+id+"'");
                if (resultset.next()) statement.executeUpdate
                        ("DELETE FROM ibcgoMemo WHERE ID='"+id+"'");
                statement.executeUpdate("INSERT INTO ibcgoMemo (ID, Message)
                        VALUES ('"+id+"', '"+message+"')");
                statement.close();
                this.close();
                }
        catch (Exception e) {}
        }

public String readMessage(String id) {
        String message = "";

        try {
                this.open();
                Statement statement = connexion.createStatement();
                ResultSet resultset = statement.executeQuery
                        ("SELECT * FROM ibcgoMemo WHERE ID='"+id+"'");
```

```
                    if (resultset.next()) message = resultset.getString("Message");
                    statement.close();
                    this.close();
                    }
                catch (Exception e) {}

                return message;
                }

        }


package IBCGo;

import java.util.*;

public class DateTime implements java.io.Serializable {

        public DateTime() {}

        public static void main(String[] args) {
                getLong();
                }

        public static long getLong() {
                Calendar now = Calendar.getInstance();

                int year = now.get(Calendar.YEAR);
                int month = now.get(Calendar.MONTH) + 1;
                int day = now.get(Calendar.DAY_OF_MONTH);
                int hour = now.get(Calendar.HOUR_OF_DAY);
                int minute = now.get(Calendar.MINUTE);
                int second = now.get(Calendar.SECOND);

                long result =
                        ((long)year*100000*100000)+((long)month*100000000)+((long)day*100000
                0);
                result += ((long)hour*10000)+((long)minute*100)+(long)second;

                return result;
                }

        public static String getString() {
                Calendar now = Calendar.getInstance();

                int year = now.get(Calendar.YEAR);
                int month = now.get(Calendar.MONTH) + 1;
                int day = now.get(Calendar.DAY_OF_MONTH);
                int hour = now.get(Calendar.HOUR_OF_DAY);
                int minute = now.get(Calendar.MINUTE);
                int second = now.get(Calendar.SECOND);

                String result = Integer.toString(year)+"-"+Integer.toString(month)+
                        "-"+Integer.toString(day);
                result += " ";
                result +=
```

```
                Integer.toString(hour)+":"+Integer.toString(minute)+":"+Integer.toString(second);

            return result;
            }

        public static String longToString(long l) {

            int year = (int)(l/100000/100000);
            int month = (int)((l/100000000)%100);
            int day = (int)((l/1000000)%100);
            int hour = (int)((l/10000)%100);
            int minute = (int)((l/100)%100);
            int second = (int)(l%100);

            String result = Integer.toString(year)+"-"+Integer.toString(month)+
                    "-"+Integer.toString(day);
            result += " ";
            result +=
              Integer.toString(hour)+":"+Integer.toString(minute)+":"+Integer.toString(second);

            return result;
            }

        }
```

```
package IBCGo;

import javax.naming.*;
import javax.naming.directory.*;
import java.util.*;

public class Directory implements java.io.Serializable {

        private Hashtable environment = new Hashtable(11);
        private String port = null;
        private String address = null;

        public static String origine = ", DC=none, DC=com";                    // Open LDAP
        //public static String origine = ", cn=configuration";                 // Tivoli
        //public static String origine = ", DC=none, DC=com";                  // Active Directory
        public static String root = "cn=Manager, DC=none, DC=com";             // Open LDAP

        //public static String root = "cn=root";                               // Tivoli
        //public static String root = "none\\administrator";                   // Active Directory
        public static String password = "password";
        public static String url = "127.0.0.1";


        // Getters & Setters
        public Hashtable getEnvironment() { return environment; }
        public void setEnvironment(Hashtable e) { environment = e; }
        public String getPort() { return port; }
        public void setPort(String s) { port = s; }
        public String getAddress() { return address; }
        public void setAddress(String s) { address = s; }
```

```
public Directory() {
        environment.put(Context.SECURITY_AUTHENTICATION,"simple");
        environment.put(Context.SECURITY_PRINCIPAL, root);
        environment.put(Context.SECURITY_CREDENTIALS, password);
        environment.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.sun.jndi.ldap.LdapCtxFactory");
        environment.put(Context.PROVIDER_URL, "ldap://"+url);
        }

public static void main(String[] args) {
        Directory dir = new Directory();

        if(args[1].equals("list")) {
                Vector agents = dir.getAgentList(args[0]);
                Enumeration enumerationAgents = agents.elements();
                while(enumerationAgents.hasMoreElements())
                        System.out.println((String)enumerationAgents.nextElement());
                }

        if(args[1].equals("agent")) {
                dir.getAgentContent(args[0], args[2]);
                }
        }


public boolean createContext(String context, String name) {
        boolean create = false;

        boolean exist = this.existContext(context, "OU", name);
        if(!exist) {
                try {
                        create = true;
                        DirContext dirctx  = new InitialDirContext(environment);
                        Attributes attrs = new BasicAttributes(true);
                        Attribute objclass = new BasicAttribute("objectclass");
                        objclass.add("top");
                        objclass.add("organizationalUnit");
                        attrs.put(objclass);
                        String sep = "";
                        if (!context.equals("")) sep = ",";
                        Context result = dirctx.createSubcontext("OU=" + name + ", " +
                                context
                                + sep + "OU=IBCGo"+origine, attrs);
                        dirctx.close();
                        }
                catch (Exception e) {
                        System.out.println(e);
                        }
                }

        return create;
        }

public boolean existContext(String context, String type, String name) {
        boolean exist = false;
```

```
        try {
                DirContext dirctx  = new InitialDirContext(environment);
                Attributes matchAttrs = new BasicAttributes(true);
                matchAttrs.put(new BasicAttribute(type, name));
                String sep = "";
                if (!context.equals("")) sep = ",";
                NamingEnumeration answer = dirctx.search(context + sep +
                        "OU=IBCGo"+origine, matchAttrs);
                if(answer.hasMore()) exist = true;
                dirctx.close();
                }
        catch (Exception e) {}

        return exist;
        }

public boolean isEmpty(String context) {
        boolean empty = false;

        try {
                DirContext dirctx  = new InitialDirContext(environment);
                Attributes matchAttrs = new BasicAttributes(true);
                        // matchAttrs est vide, donc *
                NamingEnumeration answer = dirctx.search(context + ",
                        OU=IBCGo"+origine,
                        matchAttrs);
                if(!answer.hasMore()) empty = true;
                dirctx.close();
                }
        catch (Exception e) {}

        return empty;
        }

public Vector getAgentList(String site) {
        Vector agents = new Vector();

        try {
                DirContext dirctx  = new InitialDirContext(this.environment);
                Attributes matchAttrs = new BasicAttributes(true);
                        // matchAttrs est vide, donc *

                NamingEnumeration answer = dirctx.search("OU=_agent" + "," + "OU=" +
                        site +
                        "," + "OU=IBCGo"+origine, matchAttrs);
                while(answer.hasMore()) {
                        SearchResult sr = (SearchResult)answer.next();

agents.addElement(sr.getName().substring(sr.getName().indexOf('=')+1));
                        }
                dirctx.close();
                }
        catch (Exception e) {}

        return agents;
```

```
        }

public String getAgentList() {
        String sending = "";

        try {
                Directory dir = new Directory();
                Vector agents = dir.getAgentList(address);
                Enumeration enumerationAgents = agents.elements();
                while(enumerationAgents.hasMoreElements()) {
                        String name = (String)enumerationAgents.nextElement();
                        sending += "<LI><A HREF=AgentContent.do?address="+ address
                                +
                                "&agent=" + name + " target=body>" + name +
                        "</A></LI>";
                        }
                sending += "</UL>";
                }
        catch(Exception e) {}

        return sending;
        }

public Agent getAgentContent(String site, String agentName) {
        Agent agent = null;

        try {
                Context ctx = new InitialContext(environment);
                agent = (Agent)ctx.lookup("CN=" + agentName + ", OU=_agent, OU=" + site
                        + ",
                        OU=IBCGo"+origine);
                ctx.close();
                }
        catch (Exception e) {
                System.out.println(e);
                }

        return agent;
        }

public Vector getCodeList(String site) {
        Vector codes = this.getCodeSub("OU=" + site + "," + "OU=IBCGo"+origine, 0);
        return codes;
        }

public Vector getCodeSub(String context, int level) {
        Vector codes = new Vector();

        Vector contexts = this.getContextList(context);

        Enumeration enumerationContexts = contexts.elements();
        while(enumerationContexts.hasMoreElements()) {
                String sr = (String)enumerationContexts.nextElement();
                if(!sr.substring(sr.indexOf('=')+1).equals("_agent")) {
                        codes.addElement(sr);
                        codes.addElement(Integer.toString(level));
```

```
                        if(sr.substring(0, sr.indexOf('=')).equals("OU")) {
                                Vector subs = this.getCodeSub(sr+","+context, (level+1));

                                Enumeration enumerationSubs = subs.elements();
                                while(enumerationSubs.hasMoreElements()) {
                                        codes.addElement
                                          ((String)enumerationSubs.nextElement());

                                        }
                                }
                        }
                }

        return codes;
        }

private Vector getContextList(String context) {
        Vector agents = new Vector();

        try {
                DirContext dirctx  = new InitialDirContext(this.environment);
                Attributes matchAttrs = new BasicAttributes(true);
                        // matchAttrs est vide, donc *

                NamingEnumeration answer = dirctx.search(context, matchAttrs);
                while(answer.hasMore()) {
                        SearchResult sr = (SearchResult)answer.next();
                        agents.addElement(sr.getName());
                        }
                dirctx.close();
                }
        catch (Exception e) {}

        return agents;
        }

public CodeUnit getCode(String site, String codeName) {
        CodeUnit code = null;

        try {
                Context ctx = new InitialContext(environment);
                code = (CodeUnit)ctx.lookup(codeName + ", OU=" + site + ",
                        OU=IBCGo"+origine);
                ctx.close();
                }
        catch (Exception e) {
                System.out.println(e);
                }

        return code;
        }

}

package IBCGo;
```

```
import java.io.*;
import java.util.*;
import ants.NodeAddress;
import ants.ByteArray;
import ants.Xdr;
import IBCGo.interpreter.*;

public class ExEnvironment extends Thread implements java.io.Serializable {

        private ComManager master = null;
        private IBCPacket agent = null;
        private String agentList = "";
        private boolean follow = false;
        private int destination = 0;
        private int newStart = 0;
        private boolean jump = false;
        private boolean alone = false;

        // Getters & Setters
        public ComManager getMaster() { return master; }
        public void setMaster(ComManager cm) { master = cm; }
        public IBCPacket getAgent() { return agent; }
        public void setAgent(IBCPacket ip) { agent = ip; }
        public String getAgentList() { return agentList; }
        public void setAgentList(String s) { agentList = s; }
        public boolean getFollow() { return follow; }
        public void setFollow(boolean b) { follow = b; }
        public int getDestination() { return destination; }
        public void setDestination(int i) { destination = i; }
        public int getNewStart() { return newStart; }
        public void setNewStart(int i) { newStart = i; }
        public boolean getJump() { return jump; }
        public void setJump(boolean b) { jump = b; }
        public boolean getAlone() { return alone; }
        public void setAlone(boolean b) { alone = b; }

        public static void main(String args[]) {
                try {
                        ExEnvironment exEnvironment = new ExEnvironment();
                        exEnvironment.setAlone(true);
                        XLRecognizer.setMaster(exEnvironment);
                        FileInputStream codeData = new FileInputStream(args[0]);
                        byte[] code = new byte[codeData.available()];
                        codeData.read(code);
                        codeData.close();
                        XLRecognizer.parseCode(new String(code, "UTF8"));
                        }
                catch(Exception e) {}
                }


        public void run() {
                this.master = master;

                // Saisie du code a executer
```

```
        Loader loader = new Loader(NodeAddress.toString(master.thisNode().getAddress()));

        // Interpretation du code
        XLRecognizer.setMaster(this);
        XLRecognizer.setDB(master.getDB());
        for(int i=0;i<agent.getNbCode(); i++) {
                try {
                        agent = XLRecognizer.parseCode(agent, new
                                String(loader.saisie(new
                                String(agent.getCodeData(i), "UTF8")), "UTF8"));
                        }
                catch(Exception e) {}
                }

        // Suit l'itineraire
        if(this.follow) follow(agent);
        }


public void follow(IBCPacket agent) {
        // itineraire et destination par defaut
        int type = agent.getType();
        int finalSite = agent.getNbItinerary()-1;
        for(int i=agent.getNbItinerary()-1;i>=0;i--) {
                if(agent.getSite(i) == agent.getDestination()) finalSite = i;
                if(NodeAddress.toString(agent.getSite(i)).equals("0.0.0.0")) {
                        i = -1;
                        finalSite = -1;
                        }
                }


        // itineraire et destination temporaire
        if(destination != 0) {
                if(jump) type = 2;
                finalSite = destination - 1;
                }

        // nextSite = here
        int nextSite = agent.getNbItinerary()-1;
        for(int i=agent.getNbItinerary()-1;i>=0;i--) {
                if(agent.getSite(i) == master.thisNode().getAddress()) nextSite = i;
                if(NodeAddress.toString(agent.getSite(i)).equals("0.0.0.0")) {
                        i = -1;
                        nextSite = -1;
                        }
                }


        // Si pas a la destination suivante
        if(nextSite != finalSite) {
                if(nextSite < finalSite) nextSite++;
                if(nextSite > finalSite) nextSite--;
                if(newStart != 0) nextSite = newStart;
```

```
                    // ...envoi l'agent a la nouvelle destination, sinon...
                    Credentials credentials = new Credentials();
                    credentials.setCreator(agent.getCreator());
                    credentials.setCredentials(agent.getCredentials());
                    credentials.setSigner(agent.getCreSigner());
                    credentials.setSignature(agent.getCreSignature());

                    Code code = new Code();
                    code.setNbCode(agent.getNbCode());
                    code.setCodeType(agent.getCodeType());
                    code.setCodeLength(agent.getCodeLength());
                    code.setCodeData(agent.getCodeData());
                    code.setSigner(agent.getCodeSigner());
                    code.setSignature(agent.getCodeSignature());
                    code.setInterpreterAddress(agent.getInterpreterAddress());

                    Itinerary itinerary = new Itinerary();
                    itinerary.setNbSite(agent.getNbItinerary());
                    itinerary.setK(agent.getK());
                    itinerary.setSite(agent.getSite());
                    itinerary.setSigner(agent.getItiSigner());
                    itinerary.setSignature(agent.getItiSignature());
                    if(!NodeAddress.toString(agent.getSite(nextSite)).equals("0.0.0.0")) {
                            if(nextSite==finalSite && agent.getType()==2) type = 1;
                            if(nextSite==finalSite && agent.getType()==3) type = 9;
                            master.sendPacket(agent.getIdentifier(), type, agent.getPTime(),
                                    agent.getOrigin(), agent.getSite(nextSite),
                                    agent.getSite(finalSite), credentials, code, itinerary,
                                    agent.getData(), agent.getKeep());
                            }
                       else {
                            // ...emmagasine l'agent dans le repertoire
                            boolean create =
                                    master.getVerifier().createAgent(NodeAddress.toString
                                    (master.thisNode().getAddress()), agent.getIdentifier(),
                                    agent);
                            }
                    }
                else {
                    // ...emmagasine l'agent dans le repertoire
                    boolean create =
                                    master.getVerifier().createAgent(NodeAddress.toString
                                    (master.thisNode().getAddress()), agent.getIdentifier(),
                            agent);
                    }

            this.follow = false;
            this.destination = 0;
            }

    public void go(IBCPacket agent, int destination) {
            Credentials credentials = new Credentials();
            credentials.setCreator(agent.getCreator());
            credentials.setCredentials(agent.getCredentials());
            credentials.setSigner(agent.getCreSigner());
            credentials.setSignature(agent.getCreSignature());
```

```
        Code code = new Code();
        code.setNbCode(agent.getNbCode());
        code.setCodeType(agent.getCodeType());
        code.setCodeLength(agent.getCodeLength());
        code.setCodeData(agent.getCodeData());
        code.setSigner(agent.getCodeSigner());
        code.setSignature(agent.getCodeSignature());
        code.setInterpreterAddress(agent.getInterpreterAddress());

        Itinerary itinerary = new Itinerary();
        itinerary.setNbSite(agent.getNbItinerary());
        itinerary.setK(agent.getK());
        itinerary.setSite(agent.getSite());
        itinerary.setSigner(agent.getItiSigner());
        itinerary.setSignature(agent.getItiSignature());
        master.sendPacket(agent.getIdentifier(), 1, agent.getPTime(), agent.getOrigin(),
                agent.getSite(destination), agent.getSite(destination), credentials, code,
        itinerary,
                agent.getData(), agent.getKeep());
        }


public void print(String s) {
        if(!alone) {
                ComManagerControl comManagerControl =
                        master.getComManagerControl();
                comManagerControl.print(s);
                }
        }


public void println(String s) {
        if(!alone) {
                ComManagerControl comManagerControl =
                        master.getComManagerControl();
                comManagerControl.println(s);
                }
        }


public String agentAt(String s) {
        agentList = "";

        try {
                master.sendPacket("agentAt", 7, DateTime.getLong(),
                        master.thisNode().getAddress(), NodeAddress.fromString(s),
                        NodeAddress.fromString(s), null, null, null,
                        new ByteArray(Xdr.INT), true);
                }
        catch(Exception e) {}

        while (agentList.equals("")) {
                // Wait response
```

```
            }

        return agentList;
        }


public void sendMessage(String sender, String receiver, String message) {

        ByteArray buf = new ByteArray(Xdr.INT + Xdr.STRING(sender) +
                Xdr.STRING(receiver) +
                Xdr.STRING(message));
        Xdr xdrTemp = new Xdr(buf,0);
        xdrTemp.PUT(3);
        xdrTemp.PUT(sender);
        xdrTemp.PUT(receiver);
        xdrTemp.PUT(message);

        try {
                int[] sites = master.getGroup();
                for(int i=0; i<sites.length; i++)
                        master.sendPacket("message", 9, DateTime.getLong(),
                                master.thisNode().getAddress(), sites[i], sites[i], null, null,
                        null,
                                buf, true);
                }
        catch(Exception e) {}
        }


public void multicastMessage(String sender, String message) {
        agentList = agentAt(NodeAddress.toString(master.thisNode().getAddress()));

        Vector agents = new Vector();
        String element;
        for(
                StringTokenizer stringtokenizer = new StringTokenizer(agentList, ";", false);
                stringtokenizer.hasMoreTokens();
        agents.addElement(element)
                )
                        element = stringtokenizer.nextToken();

        Vector mesSender = master.getMesSender();
        Vector mesReceiver = master.getMesReceiver();
        Vector mesContent = master.getMesContent();

        Enumeration enumerationAgent = agents.elements();
        while(enumerationAgent.hasMoreElements()) {
                String receiver = (String)enumerationAgent.nextElement();
                mesSender.add(sender);
                mesReceiver.add(receiver);
                mesContent.add(message);
                }
        }


public String receiveMessage(String sender, String receiver) {
```

```
String message = "";

Vector mesSender = master.getMesSender();
Vector mesReceiver = master.getMesReceiver();
Vector mesContent = master.getMesContent();

System.out.println("Waiting for message from " + sender + ".");

while (message.equals("")) {
        Enumeration enumerationMesSender = mesSender.elements();
        Enumeration enumerationMesReceiver = mesReceiver.elements();
        int i = 0;
        while(enumerationMesSender.hasMoreElements()) {
                String senderR = (String)enumerationMesSender.nextElement();
                String receiverR = (String)enumerationMesReceiver.nextElement();
                if(senderR.equals(sender) && receiverR.equals(receiver)) {
                        message = (String)mesContent.elementAt(i);
                        if(message.equals("")) message = "null";
                        mesSender.remove(i);
                        mesReceiver.remove(i);
                        mesContent.remove(i);
                        }
                i++;
                }
        }

System.out.println("Message received.");

return message;
}


public void sendIti(String sender, String receiver, String message, Itinerary itinerary) {

        ByteArray buf = new ByteArray(Xdr.INT + Xdr.STRING(sender) +
                Xdr.STRING(receiver) +
                Xdr.STRING(message));
        Xdr xdrTemp = new Xdr(buf,0);
        xdrTemp.PUT(3);
        xdrTemp.PUT(sender);
        xdrTemp.PUT(receiver);
        xdrTemp.PUT(message);

        int agentAddress = 0;

        try {
                agentAddress = NodeAddress.fromString(agentAddress(sender, receiver));
                }
        catch(Exception e) {}

        int agentNo = itinerary.getSite().length-1;
        int i = itinerary.getSite().length-1;
        int siteIti = itinerary.getSite(i);
        while(siteIti != agentAddress) {
                i--;
                siteIti = itinerary.getSite(i);
```

```
            }
        agentNo = i;

        master.sendPacket("poll", 3, DateTime.getLong(), master.thisNode().getAddress(),
            itinerary.getSite(0), itinerary.getSite(agentNo), null, null, itinerary, buf, true);
        }


public void respondAgent(String receiver, String sender, String response, Itinerary itinerary) {

        String message = "";

        Vector mesSender = master.getMesSender();
        Vector mesReceiver = master.getMesReceiver();
        Vector mesContent = master.getMesContent();

        Enumeration enumerationMesSender = mesSender.elements();
        Enumeration enumerationMesReceiver = mesReceiver.elements();
        int i = 0;
        while(enumerationMesSender.hasMoreElements()) {
            String senderR = (String)enumerationMesSender.nextElement();
            String receiverR = (String)enumerationMesReceiver.nextElement();
            String messageR = (String)mesContent.elementAt(i);

            if(receiverR.equals(receiver) && senderR.equals(sender) &&
              (messageR.substring(0, messageR.lastIndexOf(":")).equals("poll"))) {
                sender = senderR;
                message = messageR.substring(messageR.lastIndexOf(":")+1);
                mesSender.remove(i);
                mesReceiver.remove(i);
                mesContent.remove(i);
                }
            i++;
            }

        if(message.equals("")) {
            ByteArray buf = new ByteArray(Xdr.INT + Xdr.STRING(receiver) +
                Xdr.STRING(sender) + Xdr.STRING(response));
            Xdr xdrTemp = new Xdr(buf,0);
            xdrTemp.PUT(3);
            xdrTemp.PUT(receiver);
            xdrTemp.PUT(sender);
            xdrTemp.PUT(response);

            int agentAddress = 0;
            try {
                agentAddress = NodeAddress.fromString(agentAddress(receiver,
                    sender));
                }
            catch(Exception e) {}

            int agentNo = itinerary.getSite().length-1;
            i = itinerary.getSite().length-1;
            int siteIti = itinerary.getSite(i);
            while(siteIti != agentAddress) {
                i--;
```

```
                    siteIti = itinerary.getSite(i);
                    }
                agentNo = i;

                master.sendPacket("respond", 3, DateTime.getLong(),
                    master.thisNode().getAddress(), itinerary.getSite(0),
                    itinerary.getSite(agentNo), null, null, itinerary, buf, true);
                }
            }


public String agentOrigin(String sender, String receiver) {

        ByteArray buf = new ByteArray(Xdr.INT + Xdr.STRING(sender) +
                Xdr.STRING(receiver) +
                Xdr.STRING("origin"));
        Xdr xdrTemp = new Xdr(buf,0);
        xdrTemp.PUT(3);
        xdrTemp.PUT(sender);
        xdrTemp.PUT(receiver);
        xdrTemp.PUT("origin");

        try {
                int[] sites = master.getGroup();
                for(int i=0; i<sites.length; i++)
                        master.sendPacket("agentAddress", 10, DateTime.getLong(),
                                master.thisNode().getAddress(), sites[i], sites[i], null, null,
                        null,
                                buf, true);
                }
        catch(Exception e) {}

        String message = "";

        Vector mesSender = master.getMesSender();
        Vector mesReceiver = master.getMesReceiver();
        Vector mesContent = master.getMesContent();

        while (message.equals("")) {
                Enumeration enumerationMesSender = mesSender.elements();
                Enumeration enumerationMesReceiver = mesReceiver.elements();
                int i = 0;
                while(enumerationMesSender.hasMoreElements()) {
                        String senderR = (String)enumerationMesSender.nextElement();
                        String receiverR = (String)enumerationMesReceiver.nextElement();

                        if(receiverR.equals(sender) && senderR.equals(receiver)) {
                                String messageR = (String)mesContent.elementAt(i);
                                if(messageR.substring(0,
                                        messageR.lastIndexOf(":")).equals("originRespon
                                se"))
                                        message =

                                        messageR.substring(messageR.lastIndexOf(":")+
                                        1);
                                mesSender.remove(i);
```

```
                                        mesReceiver.remove(i);
                                        mesContent.remove(i);
                                        }
                                i++;
                                }
                        }

                return message;
                }


        public String agentAddress(String sender, String receiver) {

                ByteArray buf = new ByteArray(Xdr.INT + Xdr.STRING(sender) +
                        Xdr.STRING(receiver) +
                        Xdr.STRING("address"));
                Xdr xdrTemp = new Xdr(buf,0);
                xdrTemp.PUT(3);
                xdrTemp.PUT(sender);
                xdrTemp.PUT(receiver);
                xdrTemp.PUT("address");

                try {
                        int[] sites = master.getGroup();
                        for(int i=0; i<sites.length; i++)
                                master.sendPacket("agentAddress", 10, DateTime.getLong(),
                                        master.thisNode().getAddress(), sites[i], sites[i], null, null,
                                null,
                                        buf, true);
                        }
                catch(Exception e) {}

                String message = "";

                Vector mesSender = master.getMesSender();
                Vector mesReceiver = master.getMesReceiver();
                Vector mesContent = master.getMesContent();

                while (message.equals("")) {
                        Enumeration enumerationMesSender = mesSender.elements();
                        Enumeration enumerationMesReceiver = mesReceiver.elements();
                        int i = 0;
                        while(enumerationMesSender.hasMoreElements()) {
                                String senderR = (String)enumerationMesSender.nextElement();
                                String receiverR = (String)enumerationMesReceiver.nextElement();

                                if(receiverR.equals(sender) && senderR.equals(receiver)) {
                                        String messageR = (String)mesContent.elementAt(i);
                                        if(messageR.substring(0,
                                          messageR.lastIndexOf(":")).equals("addressResponse"))
                                                message =

                                                messageR.substring(messageR.lastIndexOf(":")+
                                                1);
                                        mesSender.remove(i);
                                        mesReceiver.remove(i);
```

```
                                        mesContent.remove(i);
                                        }
                                i++;
                                }
                        }

                return message;
                }

        public String getItinerary(String sender, String receiver) {

                ByteArray buf = new ByteArray(Xdr.INT + Xdr.STRING(sender) +
                        Xdr.STRING(receiver) +
                        Xdr.STRING("itinerary"));
                Xdr xdrTemp = new Xdr(buf,0);
                xdrTemp.PUT(3);
                xdrTemp.PUT(sender);
                xdrTemp.PUT(receiver);
                xdrTemp.PUT("itinerary");

                try {
                        int[] sites = master.getGroup();
                        for(int i=0; i<sites.length; i++) {
                                master.sendPacket("agentItinerary", 10, DateTime.getLong(),
                                        master.thisNode().getAddress(), sites[i], sites[i], null, null,
                                null,
                                        buf, true);
                                }
                        }
                catch(Exception e) {}

                String message = "";

                Vector mesSender = master.getMesSender();
                Vector mesReceiver = master.getMesReceiver();
                Vector mesContent = master.getMesContent();

                while (message.equals("")) {
                        Enumeration enumerationMesSender = mesSender.elements();
                        Enumeration enumerationMesReceiver = mesReceiver.elements();
                        int i = 0;
                        while(enumerationMesSender.hasMoreElements()) {
                                String senderR = (String)enumerationMesSender.nextElement();
                                String receiverR = (String)enumerationMesReceiver.nextElement();

                                if(receiverR.equals(sender) && senderR.equals(receiver)) {
                                        String messageR = (String)mesContent.elementAt(i);
                                        if(messageR.substring(0,
                                          messageR.lastIndexOf(":")).equals("itineraryResponse"))
                                                message =

                                        messageR.substring(messageR.lastIndexOf(":")+1);
                                        mesSender.remove(i);
                                        mesReceiver.remove(i);
                                        mesContent.remove(i);
                                        }
```

```
                                        i++;
                                        }
                                }

                        return message;
                        }
                }

package IBCGo;

import java.io.*;
import java.security.*;
import java.util.*;

public class Hancock implements java.io.Serializable {

        public static void main(String[] args) throws Exception {
                String options = args[0];
                String keystorefile = args[1];
                String storepass = args[2];
                String alias = args[3];
                String userpass = args[4];
                String dataFile = args[5];
                String signaturefile = args[6];

                Signature signature = Signature.getInstance("DSA");
                KeyStore keystore = KeyStore.getInstance("JKS");
                keystore.load(new FileInputStream(keystorefile), storepass.toCharArray());
                if (options.equals("scode") || options.equals("scredits")  || options.equals("siti"))
                        signature.initSign((PrivateKey)keystore.getKey(alias,
                                userpass.toCharArray()));
                  else
                        signature.initVerify(keystore.getCertificate(alias).getPublicKey());

                if (options.equals("scredits") || options.equals("vcredits")) {
                        SAXParse saxparse = new SAXParse(dataFile, "ibcgo:init:owner");
                        Vector resultat = saxparse.getResultat();
                        Enumeration enumerationResultat = resultat.elements();
                        String owner = (String)enumerationResultat.nextElement();

                        saxparse = new SAXParse(dataFile, "ibcgo:init:credentials");
                        resultat = saxparse.getResultat();
                        enumerationResultat = resultat.elements();
                        String credentials = (String)enumerationResultat.nextElement();

                        String bufferString = owner + credentials;
                        byte[] buffer = bufferString.getBytes("UTF8");
                        signature.update(buffer, 0, buffer.length);
                        }
                  else if (options.equals("siti") || options.equals("viti")) {
                        SAXParse saxparse = new SAXParse(dataFile, "ibcgo:agent:itinerary:site");
                        Vector resultat = saxparse.getResultat();
                        Enumeration enumerationResultat = resultat.elements();

                        String bufferString = "";
```

```
                        while(enumerationResultat.hasMoreElements())
                                bufferString += (String)enumerationResultat.nextElement();
                        byte[] buffer = bufferString.getBytes("UTF8");
                        signature.update(buffer, 0, buffer.length);
                        }
                else {
                        FileInputStream in = new FileInputStream(dataFile);
                        byte[] buffer = new byte[8192];
                        int length;
                        while ((length = in.read(buffer)) != -1)
                                signature.update(buffer, 0, length);
                        in.close();
                        }

                if (options.equals("scode") || options.equals("scredits") || options.equals("siti")) {
                        FileOutputStream out = new FileOutputStream(signaturefile);
                        byte[] raw = signature.sign();
                        out.write(raw);
                        out.close();
                        }
                else {
                        FileInputStream sigIn = new FileInputStream(signaturefile);
                        byte[] raw = new byte[sigIn.available()];
                        sigIn.read(raw);
                        sigIn.close();
                        if (signature.verify(raw))
                        System.out.println("La signature est bonne.");
                           else
                        System.out.println("La signature est mauvaise.");
                        }
                }

        }


package IBCGo;

// ANTS 1
import ants.*;
// ANTS 2
// import ants.core.*;

import java.util.*;

public class IBCPacket extends DataCapsule implements java.io.Serializable {

        // Le port destination
        private short port;

        // Entete du paquet IBCGo
        private String identifier;                      // Fournit par ComManager
                                                        // Adresse IP-Nom du paquet-numero
                                                        // Sert pour les reponses, la maintenance, ...
        private int pLength = 0;                        // Calculer lors de l'envoi (length())
        private long pTime;                             // Format: Temps    (##/##/##, ##:##:##)
        private int origin;                             // Format ANTS fournit par ComManager
```

```
private int destination;                    // Format ANTS (na)
private String creator;                     // Adresse ANTS fournit par ComManager
                                            // (Format ANTS Source x 100000) + port source
private int type;                           // Fournit par ComManager
                                                    // 1: agent
                                                    // 2: codeQuery
                                                    // 3: code
                                                    // 4: agentAt
                                                    // 5: agentAtResponse
                                                    // 6: message
private int credentials;                    // 32 bits
private byte[] creSignature;
private String creSigner;
private int nbCode;                         // Fournit par l'utilisateur (fichier de configuration
                                            // IBC-Go)
private int nbItinerary;                    // Fournit par l'utilisateur (fichier de configuration
                                            // IBC-Go)
private int[] codeType;                             // 0: Nom de code
                                                    // 1: Code avec le langage par defaut
private int[] codeLength;
private byte[][] codeData;
private String[] codeSigner;
private byte[][] codeSignature;
private long[] interpreterAddress;
private String previousSite = "0.0.0.0";
 private String currentSite = "0.0.0.0";
private int[] K;
private int[] site;
private String itiSigner;
private byte[] itiSignature;
private boolean keep;                       // false: remplacer le code existant

// Getters & Setters
public short getPort() { return port; }
public void setPort(short s) { port = s; }
public String getIdentifier() { return identifier; }
public void setIdentifier(String s) { identifier = s; }
public int getPLength() { return pLength; }
public void setPLength(int i) { pLength = i; }
public long getPTime() { return pTime; }
public void setPTime(long i) { pTime = i; }
public int getOrigin() { return origin; }
public void setOrigin(int i) { origin = i; }
public int getDestination() { return destination; }
public void setDestination(int i) { destination = i; }
public String getCreator() { return creator; }
public void setCreator(String s) { creator = s; }
public int getType() { return type; }
public void setType(int i) { type = i; }
public int getCredentials() { return credentials; }
public void setCredentials(int i) { credentials = i; }
public byte[] getCreSignature() { return creSignature; }
public void setCreSignature(byte[] ba) { creSignature = ba; }
public String getCreSigner() { return creSigner; }
public void setCreSigner(String s) { creSigner = s; }
public int getNbCode() { return nbCode; }
```

```
public void setNbCode(int i) { nbCode = i; }
public int getNbItinerary() { return nbItinerary; }
public void setNbItinerary(int i) { nbItinerary = i; }
public int[] getCodeType() { return codeType; }
public int getCodeType(int i) { return codeType[i]; }
public void setCodeType(int[] i) { codeType = i; }
public void setCodeType(int i) { codeType = new int[i]; }
public void setCodeType(int i, int j) { codeType[i] = j; }
public int[] getCodeLength() { return codeLength; }
public int getCodeLength(int i) { return codeLength[i]; }
public void setCodeLength(int[] i) { codeLength = i; }
public void setCodeLength(int i) { codeLength = new int[i]; }
public void setCodeLength(int i, int j) { codeLength[i] = j; }
public byte[][] getCodeData() { return codeData; }
public byte[] getCodeData(int i) { return codeData[i]; }
public void setCodeData(byte[][] b) { codeData = b; }
public void setCodeData(int i) { codeData = new byte[i][]; }
public void setCodeData(int i, byte[] b) { codeData[i] = b; }
public String[] getCodeSigner() { return codeSigner; }
public String getCodeSigner(int i) { return codeSigner[i]; }
public void setCodeSigner(String[] s) { codeSigner = s; }
public void setCodeSigner(int i) { codeSigner = new String[i]; }
public void setCodeSigner(int i, String s) { codeSigner[i] = s; }
public byte[][] getCodeSignature() { return codeSignature; }
public byte[] getCodeSignature(int i) { return codeSignature[i]; }
public void setCodeSignature(byte[][] b) { codeSignature = b; }
public void setCodeSignature(int i) { codeSignature = new byte[i][]; }
public void setCodeSignature(int i, byte[] b) { codeSignature[i] = b; }
public long[] getInterpreterAddress() { return interpreterAddress; }
public long getInterpreterAddress(int i) { return interpreterAddress[i]; }
public void setInterpreterAddress(long[] l) { interpreterAddress = l; }
public void setInterpreterAddress(int i) { interpreterAddress = new long[i]; }
public void setInterpreterAddress(int i, long l) { interpreterAddress[i] = l; }
public String getPreviousSite() { return previousSite; }
public void setPreviousSite(String s) { previousSite = s; }
public String getCurrentSite() { return currentSite; }
public void setCurrentSite(String s) { currentSite = s; }
public int[] getK() { return K; }
public int getK(int i) { return K[i]; }
public void setK(int[] i) { K = i; }
public void setK(int i) { K = new int[i]; }
public void setK(int i, int j) { K[i] = j; }
public int[] getSite() { return site; }
public int getSite(int i) { return site[i]; }
public void setSite(int[] i) { site = i; }
public void setSite(int i) { site = new int[i]; }
public void setSite(int i, int j) { site[i] = j; }
public String getItiSigner() { return itiSigner; }
public void setItiSigner(String s) { itiSigner = s; }
public byte[] getItiSignature() { return itiSignature; }
public void setItiSignature(byte[] b) { itiSignature = b; }
public boolean getKeep() { return keep; }
public void setKeep(boolean b) { keep = b; }

public IBCPacket() {}
```

```
//Initialisation de la capsule et du protocole (ANTS)
final private static byte[] MID = findMID("IBCGo" + ".IBCPacket");
final private static byte[] PID = findPID("IBCGo" + ".IBCPacket");

// Necessaire pour ANTS
protected byte[] mid() {
        return MID;
        }
protected byte[] pid() {
        return PID;
        }
public IBCPacket(short sa, short da, int na, ByteArray d) {
        super(sa, da, na, d);
        }

// Calcul la longueur de la capsule sérialisée (ANTS)
public int length() {
        ByteArray buf = new ByteArray(Xdr.STRING(identifier));
        ByteArray buf2 = new ByteArray(Xdr.STRING(creator));
        ByteArray buf3 = new ByteArray(creSignature);
        ByteArray buf4 = new ByteArray(Xdr.STRING(creSigner));
        ByteArray buf5 = new ByteArray(Xdr.STRING(previousSite));
        ByteArray buf6 = new ByteArray(Xdr.STRING(currentSite));
        int length =
                super.length() +
                Xdr.SHORT +
                Xdr.BYTEARRAY(buf) +
                Xdr.INT +
                Xdr.LONG +
                Xdr.INT +
                Xdr.INT +
                Xdr.BYTEARRAY(buf2) +
                Xdr.INT +
                Xdr.INT +
                Xdr.BYTEARRAY(buf3) +
                Xdr.BYTEARRAY(buf4) +
                Xdr.INT +
                Xdr.INT +
                codeElementsLength() +
                Xdr.BYTEARRAY(buf5) +
                Xdr.BYTEARRAY(buf6) +
                itineraryElementsLenght()+
                Xdr.BOOLEAN;
        pLength = length;
        return length;
        }

private int codeElementsLength() {
        int length = 0;
        for(int i=0; i<nbCode; i++) {
                ByteArray buf = new ByteArray(codeData[i]);
                ByteArray buf2 = new ByteArray(Xdr.STRING(codeSigner[i]));
                ByteArray buf3 = new ByteArray(codeSignature[i]);
                length +=
                        Xdr.INT +
                        Xdr.INT +
```

```
                                Xdr.BYTEARRAY(buf) +
                                Xdr.BYTEARRAY(buf2) +
                                Xdr.BYTEARRAY(buf3) +
                                Xdr.LONG;
                        }

                return length;
                }

        private int itineraryElementsLenght() {
                int length = 0;
                for(int i=0; i<nbItinerary; i++) {
                        length +=
                                Xdr.INT +
                                Xdr.INT;
                        }
                ByteArray buf = new ByteArray(Xdr.STRING(itiSigner));
                ByteArray buf2 = new ByteArray(itiSignature);
                length +=
                        Xdr.BYTEARRAY(buf) +
                        Xdr.BYTEARRAY(buf2);

                return length;
                }

        // Lors de l'envoi de la capsule (ANTS)
        // Serialisation (Xdr: external data representation)
        public Xdr encode() {
                Xdr xdr = super.encode();
                xdr.PUT(port);
                ByteArray buf = new ByteArray(Xdr.STRING(identifier));
                Xdr xdrTemp = new Xdr(buf,0);
                xdrTemp.PUT(identifier);
                xdr.PUT(buf);
                xdr.PUT(pLength);
                xdr.PUT(pTime);
                xdr.PUT(origin);
                xdr.PUT(destination);
                xdr.PUT(creator);
                xdr.PUT(type);
                xdr.PUT(credentials);
                buf = new ByteArray(creSignature);
                xdr.PUT(buf);
                buf = new ByteArray(Xdr.STRING(creSigner));
                xdrTemp = new Xdr(buf,0);
                xdrTemp.PUT(creSigner);
                xdr.PUT(buf);
                xdr.PUT(nbCode);
                xdr.PUT(nbItinerary);
                for(int i=0; i<nbCode; i++) {
                        xdr.PUT(codeType[i]);
                        xdr.PUT(codeLength[i]);
                        buf = new ByteArray(codeData[i]);
                        xdr.PUT(buf);
                        buf = new ByteArray(Xdr.STRING(codeSigner[i]));
                        xdrTemp = new Xdr(buf,0);
```

```
                xdrTemp.PUT(codeSigner[i]);
                xdr.PUT(buf);
                buf = new ByteArray(codeSignature[i]);
                xdr.PUT(buf);
                xdr.PUT(interpreterAddress[i]);
                }
        buf = new ByteArray(Xdr.STRING(previousSite));
        xdrTemp = new Xdr(buf,0);
        xdrTemp.PUT(previousSite);
        xdr.PUT(buf);
        buf = new ByteArray(Xdr.STRING(currentSite));
        xdrTemp = new Xdr(buf,0);
        xdrTemp.PUT(currentSite);
        xdr.PUT(buf);
        for(int i=0; i<nbItinerary; i++) {
                xdr.PUT(K[i]);
                xdr.PUT(site[i]);
                }
        buf = new ByteArray(Xdr.STRING(itiSigner));
        xdrTemp = new Xdr(buf,0);
        xdrTemp.PUT(itiSigner);
        xdr.PUT(buf);
        buf = new ByteArray(itiSignature);
        xdr.PUT(buf);
        xdr.PUT(keep);
        return xdr;
        }

// Lors de la réception (ANTS)
// Deserialisation
public Xdr decode() {
        Xdr xdr = super.decode();
        port = xdr.SHORT();
        ByteArray buf = xdr.BYTEARRAY();
        Xdr xdrTemp = new Xdr(buf,0);
        identifier = xdrTemp.STRING();
        pLength = xdr.INT();
        pTime = xdr.LONG();
        origin = xdr.INT();
        destination = xdr.INT();
        creator = xdr.STRING();
        type = xdr.INT();
        credentials = xdr.INT();
        creSignature = xdr.BYTEARRAY().toBytes();
        buf = xdr.BYTEARRAY();
        xdrTemp = new Xdr(buf,0);
        creSigner = xdrTemp.STRING();
        nbCode = xdr.INT();
        codeType = new int[nbCode];
        codeLength = new int[nbCode];
        codeData = new byte[nbCode][];
        codeSigner = new String[nbCode];
        codeSignature = new byte[nbCode][];
        interpreterAddress = new long[nbCode];
        nbItinerary = xdr.INT();
        K = new int[nbItinerary];
```

```
		site = new int[nbItinerary];
		for(int i=0; i<nbCode; i++) {
			codeType[i] = xdr.INT();
			codeLength[i] = xdr.INT();
			codeData[i] = xdr.BYTEARRAY().toBytes();
			buf = xdr.BYTEARRAY();
			xdrTemp = new Xdr(buf,0);
			codeSigner[i] = xdrTemp.STRING();
			codeSignature[i] = xdr.BYTEARRAY().toBytes();
			interpreterAddress[i] = xdr.LONG();
			}
		buf = xdr.BYTEARRAY();
		xdrTemp = new Xdr(buf,0);
		previousSite = xdrTemp.STRING();
		buf = xdr.BYTEARRAY();
		xdrTemp = new Xdr(buf,0);
		currentSite = xdrTemp.STRING();
		for(int i=0; i<nbItinerary; i++) {
			K[i] = xdr.INT();
			site[i] = xdr.INT();
			}
		buf = xdr.BYTEARRAY();
		xdrTemp = new Xdr(buf,0);
		itiSigner = xdrTemp.STRING();
		itiSignature = xdr.BYTEARRAY().toBytes();
		keep = xdr.BOOLEAN();
		return xdr;
		}

	public String getPayload() {
		ByteArray buf = getData();
		Xdr xdr = new Xdr(buf,0);
		int nbLine = xdr.INT();
		String content = "";
		for (int i=0; i<nbLine; i++){
			content += xdr.STRING();
			if(i < nbLine-1) content += "\n";
			}
		return content;
		}

	public void setPayload(String[] content) {
		ByteArray  buf = new ByteArray(payloadLength(content));
		Xdr xdr = new Xdr(buf,0);
		xdr.PUT(content.length);
		for(int i=0; i<content.length; i++) xdr.PUT(content[i]);
		setData(buf);
		}

	private int payloadLength(String[] content) {
		int length = Xdr.INT;
		for(int i=0; i<content.length; i++)
			length += Xdr.STRING(content[i]);
		return length;
		}
```

```
        // Traitement (ANTS)
        public boolean evaluate(Node n) {
                int here = n.getAddress();
                previousSite = currentSite;
                currentSite = NodeAddress.toString(here);
         if (here!=getDst()) {
                        return n.routeForNode(this,getDst());
                        }
                  else {
                        return n.deliverToApp(this,port);
                        }
                }

        }
```

```
package IBCGo;

public class IBCProtocol extends ants.Protocol implements java.io.Serializable {

        public IBCProtocol() throws Exception {
                startProtocolDefn();
                        startGroupDefn();
                                addCapsule("IBCGo" + ".IBCPacket");
                        endGroupDefn();
                endProtocolDefn();
                }

        }
```

```
package IBCGo;

import java.io.*;
import java.util.*;
import ants.NodeAddress;

public class Itinerary implements java.io.Serializable {

        private int nbSite;
        private int[] k;
        private int[] site;
        private String signer = null;
        private byte[] signature = null;

        // Getters & Setters
        public int getNbSite() { return nbSite; }
        public void setNbSite(int i) { nbSite = i; }
        public void setK(int i) {k = new int[i];}
        public int[] getK() { return k; }
        public void setK(int[] i) { k = i; }
        public int getK(int i) { return k[i]; }
        public void setK(int i, int j) { k[i] = j; }
        public void setSite(int i) {site = new int[i];}
        public int[] getSite() { return site; }
        public void setSite(int[] i) { site = i; }
```

```
public int getSite(int i) { return site[i]; }
public void setSite(int i, int j) { site[i] = j; }
public String getSigner() { return signer; }
public void setSigner(String s) { signer = s; }
public byte[] getSignature() { return signature; }
public void setSignature(byte[] b) { signature = b; }

public Itinerary() {}

public static Itinerary fromXML(String fileAgent) {
        Itinerary itinerary = new Itinerary();

        SAXParse saxparse = new SAXParse(fileAgent, "ibcgo:agent:itinerary:k");
        Vector k = saxparse.getResultat();
        int nbSite = saxparse.getNbElement();

        Enumeration enumerationK = k.elements();
        int kNo = 0;
        int[] ks = new int[nbSite];
        while(enumerationK.hasMoreElements()) {
                try {
                        ks[kNo++] = Integer.parseInt((String)enumerationK.nextElement());
                        }
                catch (Exception e) {}
                }
        itinerary.setK(ks);

        saxparse = new SAXParse(fileAgent, "ibcgo:agent:itinerary:site");
        Vector site = saxparse.getResultat();
        itinerary.setNbSite(nbSite);
        Enumeration enumerationSite = site.elements();
        int SiteNo = 0;
        int[] sites = new int[nbSite];
        while(enumerationSite.hasMoreElements()) {
                try {
                        sites[SiteNo++] =
                          NodeAddress.fromString((String)enumerationSite.nextElement());
                        }
                catch (Exception e) {}
                }
        itinerary.setSite(sites);

        saxparse = new SAXParse(fileAgent, "ibcgo:agent:itinerary");
        Vector signers = new Vector();
        Vector signatures = new Vector();

        Vector itinaryss = saxparse.getResultat();
        Enumeration enumerationSS = itinaryss.elements();
        String itiSignatureFile = "";
        while(enumerationSS.hasMoreElements()) {
                String param = (String)enumerationSS.nextElement();
                if(param.equals("signer"))
                        itinerary.setSigner((String)enumerationSS.nextElement());
                else if(param.equals("signature")) {
                        itiSignatureFile = (String)enumerationSS.nextElement();
                        }
```

```
                    else {
                        String garbage = (String)enumerationSS.nextElement();
                    }
                }

            byte[] raw = null;
            try {
                FileInputStream sigIn = new FileInputStream(fileAgent.substring(0,
                    fileAgent.lastIndexOf("/")+1)+"//"+itiSignatureFile);
                raw = new byte[sigIn.available()];
                sigIn.read(raw);
                sigIn.close();
                }
            catch (Exception e) {}
            itinerary.setSignature(raw);

            return itinerary;
            }

    }


package IBCGo;

import java.io.*;
import javax.naming.*;
import javax.naming.directory.*;
import java.util.*;

public class Loader implements java.io.Serializable {

    private String site = "";

    // Getters & Setters
    public String getSite() { return site; }
    public void setSite(String s) { site = s; }

    public Loader() {}

    public Loader(String site) {
        this.site = site;
        }

    public byte[] saisie(String s) {
        byte[] codeData = null;

        // Environnement initial
        Directory directory = new Directory();
        Hashtable env = directory.getEnvironment();


            // Separe le nom du fichier en jeton; selon le separateur '.'
            // false: le point n'est pas retenu
            Vector vectorObjects = new Vector();
            vectorObjects.addElement(site);
            String element;
```

```
                    for(
                        StringTokenizer stringtokenizer = new StringTokenizer(s,
                            ".",
                            false);
                        stringtokenizer.hasMoreTokens();
                    vectorObjects.addElement(element)
                        )
                            element = stringtokenizer.nextToken();

                // Construction du contexte specifique
                Enumeration enumerationObjects = vectorObjects.elements();
                String objectGroup = "";
                while(enumerationObjects.hasMoreElements()) {
                    String objectAdd = (String)enumerationObjects.nextElement();
                    if(enumerationObjects.hasMoreElements()) {
                        objectGroup = "OU=" + objectAdd + ", " + objectGroup;
                        }
                    else {
                        objectGroup = "CN=" + objectAdd + ", " + objectGroup;
                        }
                    }

                // Saisie le code dans le directory
                try {
                    Context ctx = new InitialContext(env);
                    CodeUnit codeObject = (CodeUnit)ctx.lookup(objectGroup +
                        "OU=IBCGo"+Directory.origine);
                    ctx.close();
                    codeData = codeObject.getCode();
                    }
                catch (Exception e) {}

            return codeData;
            }

        }


package IBCGo;

import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.*;
import java.util.*;

public class SAXParse extends DefaultHandler implements java.io.Serializable {

        private String element = "";
        private String elementRetenu = "";
        private Vector resultat = new Vector();
        private int nbElement = 0;

        // Getters & Setters
        public String getElementRetenu() { return elementRetenu; }
        public void setElementRetenu(String s) { elementRetenu = s; }
```

```
public Vector getResultat() { return resultat; }
public void setResultat(Vector v) { resultat = v; }
public int getNbElement() { return nbElement; }
public void setNbElement(int i) { nbElement = i; }

public SAXParse() {}

public SAXParse(String file, String s) {
        elementRetenu = s;
        DefaultHandler handler = this;
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
                SAXParser saxParser = factory.newSAXParser();
                saxParser.parse( new File(file), handler );
                }
        catch (ParserConfigurationException ex) {
                System.err.println ("Failed to create SAX parser:" + ex);
                }
        catch (SAXException ex) {
                System.err.println ("SAX parser exceeption:" + ex);
                }
        catch (IOException ex) {
                System.err.println ("IO exeception:" + ex);
                }
        catch (IllegalArgumentException ex) {
                System.err.println ("Invalid file argument" + ex);
                }
        }

public void startDocument() throws SAXException {
        }

public void endDocument() throws SAXException {
        }

public void startElement(String uri, String localName, String qualifiedName, Attributes
  attributes)
  throws SAXException {
        if(element.equals(""))
                element = qualifiedName;
          else
                element += ":" + qualifiedName;

        if (element.equals(elementRetenu)) {
                int len = attributes.getLength();
                for (int i = 0; i < len; i++) {
                resultat.addElement(attributes.getQName(i));
                resultat.addElement(attributes.getValue(i));
                }
                }
        }

public void endElement(String uri, String localName, String qualifiedName) throws
  SAXException {
        if(element.equals(qualifiedName))
                element = "";
```

```
                else
                        element = element.substring(0, element.lastIndexOf(":"));
                }

        public void characters(char[] ch, int start, int length) throws SAXException {
                        if (length > 0)
                                if (element.equals(elementRetenu)) {
                                        if(!new String(ch, start, length).trim().equals("")) {
                                                resultat.addElement(new String(ch, start,
                                                        length));
                                                nbElement++;
                                                }
                                        }
                }

        }
```

```
package IBCGo;

import java.math.BigInteger;
import javax.crypto.spec.*;

public class Skip {
        // n en hexadecimale
        private static final String skip1024String =
          "F488FD584E49DBCD20B49DE49107366B336C380D451D0F7C88B31C7C5B2D8EF6" +
          "F3C923C043F0A55B188D8EBB558CB85D38D334FD7C175743A31D186CDE33212C" +
          "B52AFF3CE1B1294018118D7C84A70A72D686C40319C807297ACA950CD9969FAB" +
          "D00A509B0246D3083D66A45D419F9C7CBD894B221926BAABA25EC355E92F78C7";

        // n --> creation d'un BigInteger a partir d'une representation en base 16
        // skip1024Modulus =
          171718397966129586011229151993178480901904202533705695869569760169920539808075437788747086722975900425740754301098468647941395164593810074170462799608062493021989285837416815548721035874378548121236050948528229416139585571568998066586304075565145536350296006867635076744949977849997684222020336013226588207303
        private static final BigInteger skip1024Modulus = new BigInteger(skip1024String, 16);

        // g = 2
        private static final BigInteger skip1024Base = BigInteger.valueOf(2);

        public static final DHParameterSpec sDHParameterSpec = new
                DHParameterSpec(skip1024Modulus, skip1024Base);
        }
```

```
package IBCGo;

import java.io.*;
import java.security.*;
import sun.misc.*;
import java.sql.*;

public class User {
```

```
public static void main(String[] args) throws Exception {
        String type = args[0];
        String name = args[1];
        String ID = args[2];
        String pass = args[3];
        String DBUrl = null;

        String userid = args[4];
        String password = args[5];
        String salt = args[6];
        String passwordSalt = password + salt;

        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(passwordSalt.getBytes());
        byte[] raw = md.digest();

        BASE64Encoder encoder = new BASE64Encoder();
        String base64 = encoder.encode(raw);

        try {
                if(type.equals("ODBC")) {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        DBUrl = "jdbc:odbc:"+name;
                        }
                else if(type.equals("SQLServer")) {
                        Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
                        DBUrl =

                "jdbc:microsoft:sqlserver://localhost:1443;DatabaseName="+name;
                        }
                else if(type.equals("MySQL")) {
                        Class.forName("org.gjt.mm.mysql.Driver");
                        DBUrl = "jdbc:mysql://localhost/"+name;
                        }
                else if(type.equals("Postgres")) {
                        Class.forName("org.postgresql.Driver");
                        DBUrl = "jdbc:postgresql://localhost/"+name.trim();
                        }
                else if(type.equals("Oracle")) {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        DBUrl = "jdbc:oracle:thin:@localhost:1521:"+name;
                        }

                Connexion connexion = DriverManager.getConnexion(DBUrl, ID, pass);

                PreparedStatement preparedStatement =
                        connexion.prepareStatement("INSERT INTO ibcgoUser VALUES
                        ('"+userid+"','"+base64+"','"+salt+"')");
                preparedStatement.executeUpdate();

                preparedStatement.close();
                connexion.close();
                }
        catch(Exception e) { System.out.println(e); }
```

```
                System.out.println("Usager cree");
                }

}


package IBCGo;

import java.io.*;
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;
import java.security.*;
import ants.ByteArray;

public class Verifier implements java.io.Serializable {

        private Directory directory = new Directory();
        private Hashtable environment = directory.getEnvironment();

        public Verifier() {}

        public boolean createAgentContext(String site) {
                boolean create = directory.createContext("", site);
                create = directory.createContext("OU="+site, "_agent");
                return create;
                }

        public boolean createAgent(String site, String identifier, Agent agent) {
                boolean create = false;

                boolean remove = this.removeAgent(site, identifier);
                try {
                        Context ctx = new InitialContext(environment);
                        ctx.bind("cn=" + identifier + ", OU=_agent, OU=" + site + ",
                                OU=IBCGo"+Directory.origine, agent);
                        ctx.close();
                        }
                catch (Exception e) {
                        System.out.println(e);
                        }

                return create;
                }

        public boolean createAgent(String site, String identifier, IBCPacket ibcpacket) {
                boolean create = false;

                boolean remove = this.removeAgent(site, identifier);
                try {
                        Context ctx = new InitialContext(environment);
                        Agent agent = new Agent();
                        agent.setPort(ibcpacket.getPort());
                        agent.setIdentifier(ibcpacket.getIdentifier());
                        agent.setPLength(ibcpacket.getPLength());
                        agent.setPTime(ibcpacket.getPTime());
```

```
                agent.setOrigin(ibcpacket.getOrigin());
                agent.setDestination(ibcpacket.getDestination());
                agent.setCreator(ibcpacket.getCreator());
                agent.setType(ibcpacket.getType());
                agent.setCredentials(ibcpacket.getCredentials());
                agent.setCreSignature(ibcpacket.getCreSignature());
                agent.setCreSigner(ibcpacket.getCreSigner());
                agent.setNbCode(ibcpacket.getNbCode());
                agent.setNbItinerary(ibcpacket.getNbItinerary());
                agent.setCodeType(ibcpacket.getCodeType());
                agent.setCodeLength(ibcpacket.getCodeLength());
                agent.setCodeData(ibcpacket.getCodeData());
                agent.setCodeSigner(ibcpacket.getCodeSigner());
                agent.setCodeSignature(ibcpacket.getCodeSignature());
                agent.setInterpreterAddress(ibcpacket.getInterpreterAddress());
                agent.setCurrentSite(ibcpacket.getCurrentSite());
                agent.setK(ibcpacket.getK());
                agent.setSite(ibcpacket.getSite());
                agent.setItiSigner(ibcpacket.getItiSigner());
                agent.setItiSignature(ibcpacket.getItiSignature());
                agent.setKeep(ibcpacket.getKeep());
                agent.setData(ibcpacket.getData().toBytes());
                ctx.bind("cn=" + identifier + ", OU=_agent, OU=" + site + ",
                        OU=IBCGo"+Directory.origine, agent);
                ctx.close();
                }
        catch (Exception e) {
                System.out.println(e);
                }

        return create;
        }

public boolean removeAgent(String site, String identifier) {
        boolean remove = false;

        boolean exist = this.existAgent(site, identifier);
        if(exist) {
                // Retrait de l'agent dans le repertoire
                try {
                        Context ctx = new InitialContext(environment);
                        ctx.unbind("cn=" + identifier + ", OU=_agent, OU=" + site + ",
                                OU=IBCGo"+Directory.origine);
                        ctx.close();
                        }
                catch (Exception e) {}
                remove = true;
                }

        return remove;
        }

public boolean existAgent(String site, String identifier) {
        boolean exist = directory.existContext("OU=_agent, OU="+site, "cn", identifier);

        return exist;
```

```
        }

public IBCPacket getAgent(String site, String identifier) {
        IBCPacket ibcpacket = new IBCPacket();

        try {
                Context ctx = new InitialContext(environment);
                Agent agent = (Agent)ctx.lookup("cn=" + identifier + ", OU=_agent, OU=" +
                        site+
                        ", OU=IBCGo"+Directory.origine);
                ibcpacket.setPort(agent.getPort());
                ibcpacket.setIdentifier(agent.getIdentifier());
                ibcpacket.setPLength(agent.getPLength());
                ibcpacket.setPTime(agent.getPTime());
                ibcpacket.setOrigin(agent.getOrigin());
                ibcpacket.setDestination(agent.getDestination());
                ibcpacket.setCreator(agent.getCreator());
                ibcpacket.setType(agent.getType());
                ibcpacket.setCredentials(agent.getCredentials());
                ibcpacket.setCreSignature(agent.getCreSignature());
                ibcpacket.setCreSigner(agent.getCreSigner());
                ibcpacket.setNbCode(agent.getNbCode());
                ibcpacket.setNbItinerary(agent.getNbItinerary());
                ibcpacket.setCodeType(agent.getCodeType());
                ibcpacket.setCodeLength(agent.getCodeLength());
                ibcpacket.setCodeData(agent.getCodeData());
                ibcpacket.setCodeSigner(agent.getCodeSigner());
                ibcpacket.setCodeSignature(agent.getCodeSignature());
                ibcpacket.setInterpreterAddress(agent.getInterpreterAddress());
                ibcpacket.setCurrentSite(agent.getCurrentSite());
                ibcpacket.setK(agent.getK());
                ibcpacket.setSite(agent.getSite());
                ibcpacket.setItiSigner(agent.getItiSigner());
                ibcpacket.setItiSignature(agent.getItiSignature());
                ibcpacket.setKeep(agent.getKeep());
                ibcpacket.setData(new ByteArray(agent.getData()));
                ctx.close();
                }
        catch (Exception e) {
                System.out.println(e);
                }

        return ibcpacket;
        }

// Separe le name du objet en jeton; selon le separateur '.'
// false: le point n'est pas retenu
private Vector vectorizedObject(String name) {
        Vector vectorObjects = new Vector();

        String element;
        for(
                StringTokenizer stringtokenizer = new StringTokenizer(name, ".", false);
                stringtokenizer.hasMoreTokens();
        vectorObjects.addElement(element)
                )
```

```
                element = stringtokenizer.nextToken();

        return vectorObjects;
        }

public boolean createCodeContext(String site, String name) {
        boolean create = false;

        Vector vectorObjects = this.vectorizedObject(name);

        // Evalue les elements un par un, sauf le dernier (l'objet)
        String chemin = "OU=" + site;
        Enumeration enumerationObjects = vectorObjects.elements();
        while(enumerationObjects.hasMoreElements()) {
                String object = (String)enumerationObjects.nextElement();
                if(enumerationObjects.hasMoreElements()) {
                        create = directory.createContext(chemin, object);
                        chemin = "OU=" + object + "," + chemin;
                        }
                }

        return create;
        }

public int createCode(String sourceType, String site, String name, String source, boolean
  keep) {
        int create = 0;

        boolean ok = this.createCodeContext(site, name);

        boolean exist = this.existCode(site, name);

        if(exist && !keep) {
                this.removeCode(site, name);
                exist = false;
                }

        if(!exist) {
                // Separation du path de l'object
                Vector vectorObjects = this.vectorizedObject(name);
                String path = "OU=" + site;
                String object = "";
                Enumeration enumerationObjects = vectorObjects.elements();
                while(enumerationObjects.hasMoreElements()) {
                        object = (String)enumerationObjects.nextElement();
                        if(enumerationObjects.hasMoreElements()) path = "OU="+ object +
                                "," +
                                path;
                        }

                // Saisie du code
                byte[] code = null;
                if(sourceType == "file") {
                        try {
                                FileInputStream codeData = new FileInputStream(source);
                                code = new byte[codeData.available()];
```

```
                                codeData.read(code);
                                codeData.close();
                                create = code.length;
                                }
                        catch (Exception e) {}
                                }
                        else if(sourceType == "code") {
                                try {
                                        code = source.getBytes("UTF8");
                                        }
                                catch (Exception e) {}
                                create = code.length;
                                }
                        try {
                                CodeUnit codeObject = new CodeUnit();
                                codeObject.setCode(code);
                                Context ctx = new InitialContext(environment);
                                ctx.bind("cn=" + object + "," + path + ",
                                        OU=IBCGo"+Directory.origine,
                                        codeObject);
                                ctx.close();
                                }
                        catch (Exception e) {}
                        }

                return create;
                }

        public boolean removeCode(String site, String name) {
                boolean remove = false;

                // Separation du path de l'object
                Vector vectorObjects = this.vectorizedObject(name);
                String path = "OU=" + site;
                String object = "";
                Enumeration enumerationObjects = vectorObjects.elements();
                while(enumerationObjects.hasMoreElements()) {
                        object = (String)enumerationObjects.nextElement();
                        if(enumerationObjects.hasMoreElements()) path = "OU="+ object + "," +
                                path;
                        }

                boolean exist = this.existCode(site, name);
                if(exist) {
                        // Retrait du code dans le repertoire
                        try {
                                Context ctx = new InitialContext(environment);
                                ctx.unbind("cn=" + object + "," + path + ",
                                        OU=IBCGo"+Directory.origine);
                                ctx.close();
                                }
                        catch (Exception e) {}
                        remove = true;
                        }

                // Elimination des contextes vides
```

```
        while(!path.trim().equals("") && directory.isEmpty(path)) {
                if(directory.isEmpty(path)) {
                        try {
                                Context ctx = new InitialContext(environment);
                                ctx.unbind(path + ", OU=IBCGo"+Directory.origine);
                                ctx.close();
                                }
                        catch (Exception e) {}
                        }
                path = path.substring(path.indexOf(',')+1);
                }

        return remove;
        }

public boolean existCode(String site, String name) {
        boolean exist = true;

        // Separation du path de l'object
        Vector vectorObjects = this.vectorizedObject(name);
        String path = "OU=" + site;
        String object = "";
        Enumeration enumerationObjects = vectorObjects.elements();
        while(enumerationObjects.hasMoreElements()) {
                object = (String)enumerationObjects.nextElement();
                if(enumerationObjects.hasMoreElements()) path = "OU="+ object + "," +
                        path;
                }

        exist = directory.existContext(path, "cn", object);

        return exist;
        }

public boolean verifySign(String keystorefile, String storepass, String alias, String in, byte[]
  sigIn) {
        try {
                Signature signature = Signature.getInstance("DSA");
                KeyStore keystore = KeyStore.getInstance("JKS");
                keystore.load(new FileInputStream(keystorefile), storepass.toCharArray());
                signature.initVerify(keystore.getCertificate(alias).getPublicKey());

                byte[] buffer = in.getBytes("UTF8");
                signature.update(buffer, 0, buffer.length);

                if (signature.verify(sigIn)) return true;
                }
        catch(Exception e) {}

        return false;
        }

public void test() {

        }
}
```

d. Les modifications apportées au système ANTS

```
package ants;

import java.net.*;
import java.io.*;

public class UDPChannel extends Channel {

        // Serge Giguere 2004
        // javah ants.UDPChannel
        static { System.loadLibrary("ants_UDPChannel"); }
        public static native void sendAlert(DatagramPacket datagramPacket);

        public static final int BUFSIZE = 1500;
        protected DatagramSocket socket;
        public int getMTU() { return BUFSIZE; }

        public boolean send(byte[] buf, int length, ChannelAddress dst) {
                if (! (dst instanceof UDPChannelAddress)) {
                        Entity.warn("UDPChannel.send() passed non-UDP destination address");
                        return false;
                        }

                UDPChannelAddress a = (UDPChannelAddress)dst;
                DatagramPacket fr = new DatagramPacket(buf, length, a.address, a.port);

                // Serge Giguere 2004
                // The c++ class send the packet with the 'router alert' option
                // try {
                        // socket.send(fr);
                        UDPChannel.sendAlert(fr);

                        if (LOGGING) log(L[7], "transmitted " + fr.getLength() + " bytes");
                //      }
                // catch (IOException e) {
                //      return false;
                //      }
                return true;
                }

  public Capsule receive() throws Exception {

    do {
      /* note: essential that we refresh p because of
            the "ever smaller packets" receive screw */
      DatagramPacket p = new DatagramPacket(rxbuf.buf, rxbuf.len);

      try {
            socket.receive(p);
```

```
            if (p.getLength() <= 0) {
              if (LOGGING) log(L[7], "ignoring 0 byte read");
              continue;
            }

            if (LOGGING) log(L[7], "received " + p.getLength() + " bytes");
            return decode(new Xdr(rxbuf, 0, p.getLength()));

        }
        catch (IOException e) {
            if (LOGGING) log(L[1], "DatagramSocket.receive() caught " + e);
            continue;
        }
      } while (true);
    }


    public UDPChannel(Manager b, String n, UDPChannelAddress src)
      throws IOException {

      super(b, n, src);

      socket = new DatagramSocket(src.port);
      txbuf = new Xdr(BUFSIZE);
      rxbuf = new Xdr(BUFSIZE);
    }

    public UDPChannel(Manager b, String n, String hp)throws Exception {
      this(b, n, new UDPChannelAddress(hp));
    }

}


#include <jni.h>
#include "ants_UDPChannel.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ermo.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>

JNIEXPORT void JNICALL Java_ants_UDPChannel_sendAlert(JNIEnv *env, jclass jthis, jobject
datagramPacket) {
        jclass cls = env->GetObjectClass(datagramPacket);

        // Pour connaitre les signatures
        // javap -s -p java.net.DatagramPacket

        // Address
        jmethodID mid = env->GetMethodID(cls, "getAddress", "()Ljava/net/InetAddress;");
```

```
if (mid == NULL) {
        return; // method not found
        }

jobject inetAddress = env->CallObjectMethod(datagramPacket, mid);
jclass cls2 = env->GetObjectClass(inetAddress);

mid = env->GetMethodID(cls2, "getHostAddress", "()Ljava/lang/String;");
if (mid == NULL) {
        return; // method not found
        }
jstring jaddress = (jstring)env->CallObjectMethod(inetAddress, mid);
jint len = env->GetStringLength(jaddress);
char* address;
address = (char*)malloc(len+1);
env->GetStringUTFRegion(jaddress, 0, len, address);

// Port
mid = env->GetMethodID(cls, "getPort", "()I");
if (mid == NULL) {
        return; // method not found
        }
jint jport = env->CallIntMethod(datagramPacket, mid); // = long

// Data
mid = env->GetMethodID(cls, "getData", "()[B");
if (mid == NULL) {
        return; // method not found
        }
jbyteArray data = (jbyteArray)env->CallObjectMethod(datagramPacket, mid);

// Data length
mid = env->GetMethodID(cls, "getLength", "()I");
if (mid == NULL) {
        return; // method not found
        }
jint dataLength = env->CallIntMethod(datagramPacket, mid);

// Create a socket.
int m_socket;
if ((m_socket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
        printf("socket");
        return;
        }

// Set the router alert option
int value = 1;
int err = setsockopt(m_socket, SOL_IP, IP_ROUTER_ALERT, &value, sizeof(value));
if (err < 0) {
        return;
        }

struct hostent *he;
struct sockaddr_in clientService;
he = gethostbyname(address);
```

```
clientService.sin_family = AF_INET;
clientService.sin_addr = *((struct in_addr *)he->h_addr);
memset(&(clientService.sin_zero), '\0', 8);
clientService.sin_port = htons(jport);

// Sending datagrams
char* sendbuf = 0;
len = env->GetArrayLength(data);
sendbuf = (char*)malloc(len+1);
env->GetByteArrayRegion(data, 0, len, (jbyte*)sendbuf);
int bytesSent = sendto(m_socket, sendbuf, dataLength, 0,
        (struct sockaddr*)&clientService, sizeof(clientService));

// When your application is finished sending datagrams close the socket.
close(m_socket);

return;
}
```

RÉFÉRENCES

[ABO 2004] Abone, http://www.isi.edu/abone/, 2004.

[ADV 2006] AdventNetSNMP, http://www.adventnet.com/, 2006.

[ALE 1998] Alexander, D. S., M. W. Hicks, P. Kakkar, A. D. Keromytis, M. Shaw, J. T. Moore, C. A. Gunter, T. Jim, S. M. Nettles et J. T. Moore. 1998. «The Switchware Active Network Implementation». *ML Workshop*. (Baltimore, septembre 1998). 10 p.

[AMA 2001] AMARRAGE, http://www-rp.lip6.fr/~spathis/private_html/logiciels.htm, 18 juillet 2001.

[ANT 2004] ANTLR, http://www.antlr.org/, avril 2004.

[ANT 2005] ANTS, http://www.cs.washington.edu/research/networking/ants/, novembre 2005.

[BER 2000] Bernard, G. «Apport des agents mobiles à l'exécution répartie». *In Proc. 4ème Ecole d'Informatique des Systèmes Parallèles et Répartis (ISYPAR'00)*, Toulouse, France, Février 1-3, 2000.

[BER 2000] Berson, S., B. Braden, et L. Ricciulli. «*Introduction to the Abone*». juin 2000, http://www.isi.edu/abone/DOCUMENTS/ABoneIntro.pdf., 38 p.

[BRA 2005] Braun P. 2005. *Mobile Agent: Basic Concepts, Mobility Models, & The Tracy Toolkit*. Heidelberg: Morgan Kaufmann, 441 p.

[CAM 1999] Campbell, A. T, H. G. DeMeer, M. E. Kounavis, K. Miki, J. B. Vicente, D. Villela. 1999. «A Survey of Programmable Networks». *ACM SIGCOMM Computer Communication Review*, vol. 29, no 2 (avril), p. 7-23.

[CAS 1990] Case J, M. Fedor, M. Schoffstall, J. Davin. «*Simple Network Management Protocol*». RFC 1157, IETF, mai 1990.

[CHE 1995] Chess, D., B. Grosof, C. Harrisson, D. Levine, C. Parris et G. Tsudik. 1995. «Itinerant Agents for Mobile Computing». *IBM Research Report*. Switzerland. 28 p.

[COL 1998] Coller, M. 1998. «*Mobile Agents and Active Networks: Complementary or Computing Technologies?*». www.eeng.dcu.ie/~bssl/reports/ma_vs_an.pdf, 10 p.

[DEC 1998] Decasper, D. et C. F. Tschudin. «*Simple Active Packet Format (SAPF)*», RFC DRAFT, Active Networks Group, août 1998.

[DEC 1998] Decasper, D & B. Plattner. 1998. *DAN: Distributed Code Cashing for Active Networks: Actes du Proceedings of Infocom '98* (San Francisco, 29 mars-2 avril 1998). Los Alamitos: IEEE Computer Society, 8 p.

[DEC 1998] Decasper, D., G. Parulkar, S. Choi, J. DeHart, W. Tilman & B. Plattner. 1999. «A Scalable, High Performance Active Network Node».*IEEE Network*, vol. 13, no 1 (janvier-février), p. 8-27.

[DEE 1998] Deering, S. et R. Hinden. «*Internet Protocol, Version 6 (IPv6) Specification*». RFC 2460, IETF, décembre 1998.

[DOO 2003] Dooley, K. et I. J. Brown. 2003. *Cisco Cookbook*. Sebastopol: O'Reilly, 800 p.

[DRA 1997] Draden, B., C. A. Gunter, A. W. Jackson, A. D. Keromytis, G. J. Minden et D. Wetherall. «*Active Network Encapsulation Protocol (ANEP)*». RFC DRAFT, Alexender, juillet 1997.

[JUI 1999] Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach et T. Berners-Lee. «Hypertext Transfer Protocol -- HTTP/1.1». RFC 2616, IETF, juin 1999.

[HAE 2002] Haettel, F. 2002. *Les réseaux actifs. Gestion dynamique des services pour les réseaux futur: Actes du 9e Colloque Francophone sur l'Ingénierie des Protocoles* (Montréal, 27-30 mai 2002). Paris : Thalès Communications, 80 p.

[HIC 1997] Hicks, M. W. 1997. «PLAN Service Programmer's Guide». *PLAN documentation.* 8p.

[HOH 1999] Hohl, F. 1999. «*Tutorial: Mobile Agents and Active Networks*», http://www.cs.ait.ac.th/~ca/smartnet99/tutorial6.html.

[HUS 2003] Husted, T., C. Dumoulin, G. Franciscus et D. Winterfeldt. 2003. *Struts in Action: Building web applicationswith the leading Java framework.* Greenwich : Manning, 630 p.

[KAK 1997] Kakkar, P., J. T. Moore et M. W. Hicks. 1997. «The PLAN Tutorial». *PLAN documentation.* 10p.

[KAT 1997] Katz, D. «*IP Router Alert Option*», RFC 2113, IETF, février 1997.

[KIL 2002a] Kilany, R. et A. Sehrouchni. «*ANS-2: A General Purpose Active Network Simulator*», 2002, http://www.fi.usj.edu.lb/prof/Rima_Kilany/ANS-2/active_ns_contrib.htm, 5 p.

[KIL 2002b] Kilany, R. et A. Serhrouchni. 2002. «Une architecture à base de composantes distribuées pour le déploiement de services actifs». Dans *CFIP'2002: Actes du 9e Colloque Francophone sur l'Ingénierie des Protocoles* (Montréal, 27-30 mai 2002), sous la dir. de Abdel Obaid, p. 113-125. Paris: Hermès Science.

[KNA 1996] Knabe, F. 1996. «An overview of mobile agent programming». *Proceedings of the Fifth LOMAPS Workshop on Analysis and Verification of Multiple-Agent Languages*, p. 100 – 115.

[KNU 1998] Knudsen, J. B. 1998. *Java Cryptography.* San Francisco: O'Reilly, 362 pages.

[KOT 2002] Kotz, D., R. Gray, et D. Rus. 2002 «Future directions for mobile-agent research». *Technical Report Technical Report TR2002-415*, Dept. Computer Science, Dartmouth College, http://citeseer.ist.psu.edu/kotz02future.html.

[LAN 1998] Lange, D. B. et M. Oshima. 1998. *Programming and Deploying Java Mobile Agents with Aglets.* Massachusetts : Addison Wesley, 225 p.

[LAR 2001] Larrabeiti, D., M. Calderón, A. Azcorra, M. Urueña. 2001. «SARA: a Simple Active Router-Assistant Architecture», *white paper*, 12 p.

[LAR 2002] Larrabeiti, D., M. Calderón, A. Azcorra, M. Urueña. 2002. *A practical approach to network-based processing : Actes du 4th International Workshop on Active Middleware Services* (Edimbourg, 23 juillet 2002). Los Alamitos: IEEE Computer Society, 8 p.

[LEN 2003] Lennox, Wu et Schulzrinne. «*CPL: A Language for User Control of Internet Telephony Services*». Internet Draft, IETF, août 2003.

[LEV 2001] Levina, T. 2001. «*Mobile Agents*», Expert Systems, décembre 2001.

[LIA] Liang, S. *The Java Native Interface: Programmer's Guide and Specification.* California : Addison-Wesley, 303 p.

[MAR 2002] Marques, P., P. Santos, L. Silva, et J. G. Silva. 2002. «Supporting Disconnected Computing in Mobile Agent Systems», *in Proc. of the 14th IASTED Int. Conference on Parallel and Distributed Computing and Systems*, http://citeseer.ist.psu.edu/marques02supporting.html

[MIL 2004] Mills, A. J. S. «*ANTLR: Reference Manual*», mai 2004, http://www.antlr.org/doc/index.html.

[MOO 1997] Moore, J. T. et M. Hicks. 1997. «PLAN Programmer's Guide». *PLAN documentation.* 15p.

[MOO 1997] Moore, J. T. et M. Hicks. 1997. «A Service Layer Routing Protocol for PLAN». *PLAN documentation.* 5p.

[NET] Network Simulator, http://www.isi.edu/nsnam/ns/.

[OBA 2004] Obaid, A. et S. Giguère. 2004. «IBC-Go». *White paper*, 15 p.

[OBJ 1997] ObjectSpace, Inc. 1997. «ObjectSpace Voyager and Agent Plateforms Comparaison». *ObjectSpace Voyager Core Technology*. septembre 1997. 23 p.

[ODO 2003] O'Doherty, P. «*SIP Specifications and the Java Platforms: The look and feel of SIP!*». 7 janvier 2003, http://www.cs.columbia.edu/sip/Java-SIP-Specifications.pdf.

[PEI 2003] Peine, H. «*An Introduction to Mobile Agent Programming and the Ara System.*» mars 2003, http://www.agent.ai/doc/upload/200303/pein97_2.pdf.

[POS 1980] Postel, J. «*User Datagram Protocol*». RFC 768, IETF, août 1980.

[POS 1981] Postel, J. «*Internet Protocol*». RFC 791, IETF, septembre 1981.

[POS 1985] Postel, J. «*File Transfert Protocol*» RFC 959, IETF, octobre 1985.

[RAO 2004] Rao, R. H. R., S. Hewavitharana et W. J. Choo. 2004. «Security issues in mobile agents and their solutions», *White paper,* http://www.comp.nus.edu.sg/~cs4274/termpapers/0304-I/group5/paper.pdf.

[REC] Recursion Software Inc., http://www.recursionsw.com/.

[REC 2003] Recursion Software, Inc. 2003. «*Voyager ORB Developer's Guide*». http://www.recursionsw.com/products/voyager/voyager.asp.

[SAT 2003] Satoh, I. 2003. «Building Reusable Mobile Agents for Network Management». *IEEE SMC Transactions*, Part C (octobre), p 1-8.

[SCH 1999] Schulzrinne, H. «*Session Initiation Protocol (SIP)*», 2 février 1999, http://www.cs.colombia.edu/~hgs/sip/.

[SCH 2000] Schwartz, B., A. W. Jackson, W. T. Strayer, W. Zhou, D. Rockwell et C. Partridge. 2000. «Smart Packets for Active Networks». *ACM Transactions on Computer Systems*, vol. 18, no 1 (février), p. 1-8.

[SIA 2005] Siam Communications Group, 2005. «*Mobile Internet Usage on the rise.*», http://www.siamcomm.com/article148.html.

[SMA] Smart Packets, http://www.ir.bbn.com/projects/spkts/smtpkts-index.html.

[SME 2001] Smed, J., T. Kaukoranta et H. Hakonen. 2001. «*Aspects of Networking in Multiplayer Computer Games*». http://staff.cs.utu.fi/~jounsmed/papers/AspectsOfMCGs.pdf.

[SPA 2000] Spatihs, P. 2000. «Active Node Transfert System: Architecture & Toolkit». *Tutorial sur ANTS Amarrage*. (Paris, 16-17 mars 2000). 45 p.

[SPA 2001] Spathis, P. et al. 2001. «Plate-forme Amarrage v1 : Enseignements & Perspectives». *Réunion plénière Amarrage, FT R&D, Issy Les Moulineaux*. (Paris, 27-28 février 2001). 17 p.

[STA 2004] Stanchfield, S. «*An ANTLR Tutorial*». Mai 2004, http://javadude.com/articles/antlrtut/.

[SUN 2001a] Sun Microsystems. 2001. «*jarsigner - JAR Signing and Verification Tool*», http://java.sun.com/j2se/1.3/docs/tooldocs/win32/jarsigner.html.

[SUN 2001b] Sun Microsystems. 2001. «*keytool - Key and Certificate Management Tool*», http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html.

[SUN 2004a] Sun Microsystems. 2004. «*Java Authentication and Authorization Service (JAAS) Overview*», http://java.sun.com/products/jaas/overview.html.

[SUN 2004b] Sun Microsystems. 2004. «*Java Cryptography Extension (JCE)*», http://java.sun.com/products/jce/.

[SUS 2000] Susbielle, J.-F. 2000. *Internet multimedia et temps réel*, Paris : Eyrolles, 729 p.

[TAB 2002] Tabery, P. et C. Bachmeir. 2002. *Advanced Network Services using Programmable Networks:Actes du 8th EUNICE Open European Summer School on Adaptable Networks and Teleservices* (Thondheim, Norvège, 2-4 septembre 2002), 7 p.

[TAN 1995] Tanenbaum, A. S. 1995. *Computer Network*. New Jersey : Prentice Hall PTR, 658 p.

[TEN 1996] Tennenhouse, D. L. et D. J. Wetherall. 1996. «Towards an Active Network Architecture». *Computer Communication Review*, vol. 26, no 2 (avril), p. 24-38.

[TEN 1997] Tennenhouse, D. L., J. M. Smith, W. D. Sincoskie, D. J. Wetherall et G. J. Minden. 1997. «A Survey of Active Network Research». *IEEE Communications Magazine*, vol. 35, no 1 (janvier), p. 80-86.

[THE] The SwitchWare Project, http://www.cis.upenn.edu/~switchware/

[THE 2003] The VINT Project. «*The ns Manual (formerly ns Notes and Documentation)*». http://www.isi.edu/nsnam/ns/ns-documentation, 13 décembre 2003.

[TUM 2000] Tu, M. 2000. «*Voyager*». http://www.utdallas.edu/~tumh2000/VY.html.

[UMB 2003] UMBC Agent Web, http://agents.umbc.edu/papers/papers.shtml, 14 août 2003.

[WAH 1997] Wahl, M., T. Howes et S. Kille. «Lightweight Directory Access Protocol (v3)» RFC 2251, IETF, décembre 1997.

[WEB 2005] Webopedia, 2005, http://www.webopedia.com.

[WET 1997] Wetherall, D. J. 1997. «Developing Network Protocols with the ANTS Toolkit». *Design Review*, 14 p.

[WET 1998a] Wetherall, D. J., J. V. Guttag and D. L. Tennenhouse. 1998. «ANTS: A Toolkit for building and Dynamically Deploying Network Protocols». *IEEE OPENARCH'98*, 11 p.

[WET 1998b] Wetherall, D. J. 1998. «Service Introduction in a Active Network». Thèse de doctorat, Boston, Massachusetts Institute of Technology, 157 p.

[WET 1999] Wetherall, D. J. 1999. «Active network vision and reality: lessons from a capsule-based system». *Operating Systems Review*, vol. 33, no 5 (décembre), p. 64-79.

[WHI 2001] Whitaker, A. et Wetherall, D. 2001. «*ANTS: an Active Node Transfer System*». http://www.cs.washington.edu/research/networking/ants.

[WHI 1996] White, J. E. 1996. «Mobile Agents». Dans *Software Agents*, sous la dir. de Jeffrey M. Bradshaw, p. 437-472, Massachusetts: MIT Press.

[WIK 2004] Wikipedia, 2004, http://en.wikipedia.org/wiki/Mobile_code.

[WIK 2005] Wikipedia, 2005, http://en.wikipedia.org/wiki/Mobile_agent.

[WIL 1998] Wilhelm, U. G. et S. Staamann. 1998. «Protecting the Itinerary of Mobile Agents». *Proceedings of the 4th ECOOP Workshop on Mobility: Secure Internet Mobile Computations*.

[ZEB 2002] Zebiane, D., R. Hammi, S. Prométhée, A. Serhrouchni, K. L. Thai et K. Chen. 2002. «AMARRAGE : Déploiement et expérimentation d'un réseau actif à grande échelle». *White Paper*, 15 p.