nne Démocratique et Populaire
Supérieur et de la Recherche Scientifique
Université Larbi Ben Mhidi Oum El Bouaghi
Département d'Informatique

ECOLE DOCTORALE INFORMATIQUE DE L'EST 2007/2008
PÔLE CONSTANTINE

Mémoire pour obtenir le diplôme de

Magister en Informatique

OPTION: GENIE LOGICIEL

Thème

AGENTS MOBILES ADAPTABLES POUR SE PROTEGER DES ENVIRONNEMENTS DESAVANTAGEUX

Par

Nardjes BOUCHEMAL

Composition du Jury:

Mr Azzeddine BILAMI	**Université de Batna**	**Président**
Mr Ramdane MAAMRI	**Université de Constantine**	**Rapporteur**
Mr Okba KAZAR	**Université de Biskra**	**Examinateur**
Mr Salim CHIKHI	**Université de Constantine**	**Examinateur**

2009/2010

J'adresse mes vifs remerciements à Mr Ramdane MAAMRI, Maître de Conférences, Université Mentouri de Constantine pour son soutien constant, ses encouragements continus, sa patience, sa gratitude et sa qualité humaine qu'il m'a manifestés durant la réalisation de ce travail.

Je tien à remercier Mr Zaidi SAHNOUN, Professeur, Université de Constantine, pour ces précieux conseils et ces bonnes orientations.

J'exprime mes remerciements à Mr Azzeddine BILAMI, Maître de Conférence, Université de Batna, d'avoir accepté la présidence de ce jury, ainsi que Mr Okba KAZAR, Maître de Conférence à l'Université de Biskra, et Mr Salim CHIKHI, Maître de Conférence, Université de Constantine, d'avoir accepté de juger ce mémoire.

Je tiens à remercier tous ceux qui m'ont aidée de prés ou de loin pour la réalisation de ce travail.

Je remercie enfin ma famille, particulièrement mes parents, qui m'ont toujours soutenue, ainsi que mon marie Riade.

Chapitre I : TECHNOLOGIE DES AGENTS MOBILES

Chapitre II : SECURITE DES AGENTS MOBILES

Aôheure actuelle, les réseaux informatiques gagnent de plus en plus de popularité et dôimportance, et tendent vers de nouvelles architectures dynamiques à large échelle. Les sites deviennent mobiles, apparaissent et disparaissent fréquemment et les connexions évoluent de façon continue.

Dôun point de vue logiciel, ces évolutions nous mènent à revoir les systèmes et applications réparties qui étaient conçus selon les anciennes structures fixes. Plusieurs schémas dôorganisations sont proposés dans ce sens, on parle du concept client/serveur, le pair à pair ou le schéma de publication/abonnement. Parmi ces propositions, la notion dôagent mobile constitue une abstraction déjà ancienne et présentée parfois comme une approche pouvant apporter des performances meilleures par rapport au schéma de base client/serveur.

Dans le domaine de l'intelligence artificielle, un agent est une entité autonome capable d'agir dans son environnement tout en communiquant avec les autres et en utilisant des compétences et des ressources propres. Un agent agit pour le compte d'un tiers (un autre agent, un utilisateur), offre des services et tend à satisfaire ses objectifs.

Pour devenir mobile, un agent se dote en plus de la propriété de migration [34] lui permettant de se déplacer d'un site à un autre en cours d'exécution pour se rapprocher des données ou des ressources. Il se déplace non seulement avec son code et ses données propres, mais aussi avec son état d'exécution.

Les agents mobiles ont été introduits initialement en 1994 avec lôenvironnement Telescript [103] qui permettait à des processus de choisir eux-mêmes de se déplacer sur les sites dôun réseau afin de travailler localement sur les ressources.

capacité de migration des agents mobiles, tel que l'exécution asynchrone et autonome des tâches en déléguant cette responsabilité au niveau de l'agent plutôt que l'application, la réduction du trafic réseau en limitant les transactions distantes, et la robustesse et la tolérance aux fautes en maintenant l'exécution de l'application même en cas de défaillance dans les supports d'exécution (machine hôte, connexion réseau).

Cependant, les agents mobiles soufrent d'une faiblesse, cause de leur utilisation limitée. Il s'agit de la sécurité. Ce problème est difficile car lôenvironnement visité a un contrôle total sur lôexécution de lôagent mobile, et plusieurs types d'attaques sont faisables.

Il est possible de distinguer trois grandes catégories dôattaques que des hôtes malicieux pourraient mener : lôinspection (espionnage de données et du code), la modification (altération des données et du code) et le rejeu (la réexécution du code) [109]. Lôinspection consiste à examiner le contenu de lôagent, ou le flot dôexécution afin de récupérer des informations sensibles transportées par lôagent mobile. La modification est réalisée en remplaçant certains éléments (donnée ou partie de code) de lôagent mobile dans le but de conduire une attaque. Le remplacement du code incitera lôagent à effectuer des opérations malveillantes sur les futurs hôtes à visiter, tandis que le remplacement des données permet à lôhôte malicieux de manipuler lôagent mobile à son avantage. Le rejeu sôobtient en clonant lôagent puis en exécutant le clone dans plusieurs configurations pour retrouver le savoir de lôagent.

La protection dôun agent mobile à lôencontre dôhôtes malicieux revient donc à protéger principalement son exécution, son intégrité et sa confidentialité [16, 50, 62]. Protéger lôexécution dôun agent mobile signifie quôil faut sôassurer quôil nôest pas à la merci de lôhôte et quôil ne peut pas être détourné vers dôautres destinations, être privé de ressources ou être terminé de façon prématurée.

ge la détection de la modification de son code et de son état dû à une mauvaise exécution par un hôte malicieux. Enfin, protéger la confidentialité revient à cacher le code et lôétat de lôagent mobile vis-à-vis du site qui lôexécute [49, 85, 92].

Problématique

Le concept dôagent mobile implique en principe deux acteurs : le propriétaire de lôagent et le propriétaire de la plate-forme. Chacun veut avoir des garanties que lôautre ne pourra pas lôattaquer. Dôun coté, on a la plate-forme qui ne connaît pas toujours le programme de l'agent mobile quôelle reçoit et quôelle doit exécuter, il faut donc des techniques de protection de la plate-forme. Cela présente le premier axe de recherche dans le domaine de la sécurité des agents mobiles.

Dôun autre coté, le propriétaire de lôagent mobile ne veut pas quôune plate-forme puisse attaquer son agent, en modifiant ses données, son code ou les résultats quôil a obtenus sur dôautres plates-formes. Pour parer à ces attaques, il faut trouver des techniques de protection de lôagent, ça représente un deuxième axe de recherche non bien maîtrisé, et c'est ce qui nous intéresse dans cette thèse.

Plusieurs approches de protection de lôagent mobile ont été proposées. Elles tentent de garantir lôaccès de lôagent mobile à des hôtes dans lesquels il peut avoir toute confiance ou de déceler ceux qui sont malveillants. On peut classer ces approches en deux grandes classes: Approches basées prévention, dont le principe consiste à éviter les attaques, et Approches basées détection, dont le principe consiste à détecter les attaques plutôt que les éviter.

Le travail présenté dans cette thèse a pour objectifs de contribuer au domaine de la protection des agents mobiles contres les attaques menées par les sites malicieux. Nous avons essayé d'étudier les différentes politiques de sécurité de protection des agents mobiles proposées jusqu'à nos jours, pour pouvoir

ée sur le concept d'adaptabilité en plus d'un agent éclaireur.

L'agent éclaireur est un prototype de l'agent mobile. C'est une copie restreinte de lôagent mobile avec un code sans importance ne traduisant aucun savoir faire, et des données non critiques. Le principe consiste à envoyer dôabord cet agent vers lôhôte de destination.

A son retour, l'agent éclaireur est analysé par lôagent mobile afin de détecter dôéventuelles attaques. Si ces attaques sont graves, lôagent mobile refait une copie et change de destination. Si les attaques sont évitables, lôagent mobile choisit un plan dôadaptation puis migre. Si aucune attaque nôest détectée, lôagent mobile migre avec confiance.

Organisation de la thèse

Ce mémoire de thèse est organisé en quatre chapitres: les deux premiers chapitres fixent le contexte de notre travail, en faisant une vision sur le domaine d'étude et sur les travaux existants réalisés pour la protection des agents mobiles. Les deux chapitres suivants constituent notre proposition ainsi que l'implémentation de l'architecture proposée et une expérimentation comparative entre l'approche proposée et deux autres approches. Nous détaillons par la suite le contenu de chacun de ces chapitres:

Chapitre 1

Le premier chapitre aborde dans une première partie les agents et les systèmes multi agents. Dans une deuxième partie, il décrit les aspects relatifs aux agents mobiles et leurs architectures.

Chapitre 2

Le deuxième chapitre concerne la sécurité des agents mobiles. On s'intéresse d'abord à la sécurité des plates-formes, des attaques menées par les agents mobiles et des techniques proposées pour protéger les sites visités contre les

eux. Ensuite, on présente la sécurité des agents mobiles, allant des attaques et risques jusqu'aux différentes approches proposées.

Chapitre 3

Ce chapitre est consacré à la présentation du protocole proposé ainsi que son architecture dans le but de protéger l'agent mobile des sites malveillants. Notre approche est basée sur le concept d'adaptabilité et un agent éclaireur.

Chapitre 4

L'implémentation du système proposé est présentée dans ce chapitre. Nous avons choisit de faire l'implémentation dans la plate-forme JADE (Java Agent DEvelopment framework). La deuxième partie du chapitre est consacrée à une expérimentation, où nous avons fait une comparaison entre l'approche proposée et deus autres techniques.

Conclusion

La conclusion résume les points essentiels de ce travail, fait le bilan de la réalisation et présente les perspectives de recherches suggérées par ce travail.

1 TECHNOLOGIE DES AGENTS MOBILES

La programmation par agents mobiles est un paradigme de programmation pour les applications réparties née de deux domaines différents : les systèmes multi agents et lôintelligence artificielle distribuée. Les agents mobiles sont d'un grand intérêt pour la mise en î uvre d'applications dont les performances varient en fonction de la disponibilité et de la qualité des services et des ressources, ainsi que du volume des données déplacées.

Le but de ce chapitre est de présenter une synthèse des technologies des agents mobiles ainsi que leurs applications novatrices. Dôabord, nous présenterons le concept dôagent et des systèmes multi-agents. Ensuite, nous passerons en revue de quelques applications basées sur les systèmes multi agents. Puis nous présenterons la notion dôagents mobiles, leurs applications dans différents domaines et les différentes plates-formes existantes. Pour terminer, nous discuterons les différents points forts et faibles des agents mobiles.

...ies

En informatique, un agent est l'équivalent d'un robot logiciel. C'est un programme qui accomplit des tâches à la manière d'un automate et en fonction de ce que lui a demandé son auteur.

I.1.1 Définitions

Vu la relative jeunesse du domaine, il nôexiste pas encore un consensus sur la définition dôun agent. Nous présentons dans ce qui suit quelques définitions proposées:

Dôaprès Ferber [32] : On appelle agent une entité physique ou virtuelle

- qui est capable dôagir dans un environnement,
- qui peut communiquer directement avec dôautres agents,
- qui est munie par un ensemble de tendances (sous la forme dôobjectifs individuels ou dôune fonction de satisfaction, voire de survie, quôelle cherche à optimiser),
- qui possède des ressources propres,
- qui est capable de percevoir (mais de manière limitée) son environnement,
- qui ne dispose que dôune représentation partielle de cet environnement (et éventuellement aucune),
- qui possède des compétences et offre des services,
- qui peut éventuellement se reproduire,
- dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications quôelle reçoit.

Jennings et Wooldridge [61] proposent la définition: « *Un agent est un système informatique situé dans un environnement et capable d'actions autonomes afin d'atteindre ses objectifs de conception* ».

De son côté, Perret [78] propose la définition: *« Un agent est une entité active autonome agissant par délégation pour le compte d'un client ».*

u physique à qui est attribuée une certaine mission qu'elle est capable d'accomplir de manière autonome et en coopération avec d'autres agents".

I.1.2 Capacités d'un agent

L'avancement des travaux dans le domaine des systèmes multi agents a conduit les chercheurs à définir non seulement la notion d'agent mais aussi quelques-unes de ses caractéristiques et capacités [81, 89]:

I.1.2.1 Autonomie

Par rapport à un objet, un agent peut prendre des initiatives, peut refuser d'obéir à une requête ou peut se déplacer. L'autonomie permet au concepteur de se concentrer sur une partie humainement appréhendable du logiciel.

Un agent autonome peut opérer sans une intervention directe, humaine ou non. Cela signifie quõil doit avoir un degré de liberté vis-à-vis de son utilisateur. En effet, un agent autonome peut poursuivre un agenda indépendant de celui de son utilisateur. Cela exige lõintégration des aspects dõaction périodique, dõexécution spontanée et dõinitiative, qui permettent à un agent de prendre des décisions et dõentreprendre des actions de préemption ou des actions indépendantes qui sont éventuellement bénéfiques pour lõutilisateur.

I.1.2.2 Réactivité

Les agents peuvent percevoir leurs environnements et répondre, au moment opportun, à des changements qui sõy produisent.

I.1.2.3 Pro-activité

Les agents peuvent dévoiler ou exhiber un comportement dirigé vers un but en prenant des initiatives. Ils peuvent raisonner sur leurs intentions et leurs croyances, et planifier leurs actions en conséquence.

sant des unités fonctionnelles dédiées à des tâches précises, ainsi que les techniques d'apprentissage classiques de l'intelligence artificielle, s'appliquent particulièrement à l'apprentissage d'un agent.

I.1.2.5 Adaptabilité

Un agent possède cette propriété s'il est capable d'utiliser ses connaissances afin de modifier son comportement (augmentation de l'espace disque, détection d'un seuil, etc.).
En plus, il a la capacité de contrôler ses aptitudes selon l'agent ou l'environnement avec qui il interagit. L'adaptabilité des agents nécessite des capacités d'apprentissage, par exemple pour élaborer des cartes cognitives ou pour modifier ses préférences comportementales en fonction de son expérience.
L'adaptation peut également être collective : en utilisant des mécanismes d'évolution, des populations d'agents peuvent apprendre à s'adapter à leur environnement.

I.1.2.6 Aptitude sociale et coopération

Les agents peuvent interagir avec d'autres agents et avec des humains. La coopération Utilisateur-Agent peut prendre la forme d'un contrat de collaboration. L'utilisateur spécifie les actions devant être entreprises pour son compte, et l'agent spécifie ce qu'il peut faire pour fournir les résultats.

On peut comparer ce contrat à une conversation dans laquelle chacune des parties peut poser des questions à l'autre pour s'assurer de sa bonne compréhension.
La coopération inter-agents, quant à elle, est un mécanisme par lequel les agents échangent leurs connaissances, leurs croyances et leurs plans afin de travailler ensemble pour résoudre des problèmes qui dépassent leurs capacités individuelles.

I.1.2.7 Communication

Bien que l'interaction et la communication sont souvent confondues dans la littérature, la communication représente la transmission d'informations entre agents, alors que l'interaction comprend l'action sur le monde, ainsi que la communication entre les agents du système [21].

distinguent dans la littérature : la communication de signaux via lôenvironnement, et la communication directe, qui correspond aux échanges de messages entre les agents.

I.1.2.8 Personnalisation

Lôune des raisons dôêtre dôun agent est de permettre à ses utilisateurs de réaliser certaines tâches de la meilleure manière possible. Puisque ces derniers nôexécutent pas tous les mêmes tâches, et même ceux qui partagent les mêmes tâches le font de façons différentes, un agent doit être éducable (personnalisable). Idéalement, les agents doivent avoir des composantes dôapprentissage et de mémorisation.

I.1.3 Types d'agents

Suivant leurs caractéristiques d'autonomie, organisation, raisonnement, mobilité et apprentissage [61, 81, 89], on peut définir quelques types dôagents:

I.1.3.1 Agent cognitif

L'intelligence artificielle classique sôest concentrée très tôt sur l'expression du comportement délibératif dôun agent rationnel (ou intelligent) en fonction de ses croyances et buts. Ceci a donné lieu aux premières architectures modernes d'agents dits cognitifs.

Un agent cognitif dispose d'une capacité de raisonnement sur une base de connaissance, d'une aptitude à traiter des informations diverses liées au domaine d'application, et d'informations relatives à la gestion des interactions avec les autres agents et l'environnement.

I.1.3.2 Agent réactif

On part du principe suivant: "Dans un système multi-agent, il n'est pas nécessaire que chaque agent sois individuellement intelligent pour parvenir à un comportement globale intelligent". Le principe consiste à introduire des agents peu intelligents mais plus nombreux, dits agents réactifs.

s sont les démons Unix (processus informatiques autonomes capables de se réveiller à certaines heures ou en fonction de certaines conditions). En plus, les virus informatiques en sont des versions déjà plus sophistiquées et notamment douées de la capacité de reproduction et de malfaisance.

Notons quôil est difficile de donner une définition consensuelle du terme agent logiciel. Cela sous-entend souvent aussi une encapsulation en agent (agentification) de programmes ou de bases de données plus traditionnels, de manière à favoriser leur coopération. En tout cas, ce terme sôoppose au terme agent physique, tel un robot.

I.1.3.4 Agent assistant

Pour dépasser les limitations des interfaces homme-machine à manipulation directe (rigidité, complexité, etc.), les agents assistants apportent une adaptation au profil de l'utilisateur et une capacité à anticiper leurs besoins (automatiser certaines tâches, rappeler certaines informations utiles, etc.). Ceci peut se transposer dans le domaine du collectif. Prenons lôexemple dôun rendez-vous entre plusieurs personnes, mis en place par la coopération de leurs agents assistants respectifs pouvant résider sur des agendas électroniques (PDA) ou des ordinateurs.

I.1.3.5 Agent robotique

On peut considérer lôarchitecture logicielle de contrôle dôun robot comme un agent. Cette architecture peut dôailleurs être testée et entraînée en simulation avant d'être mise en oeuvre dans un vrai robot. Cependant, la réalité physique d'un robot apporte des problèmes spécifiques (imprécision de la perception et de l'action, aspects temps réel, évolution du monde) qui rendent particulièrement difficile, mais aussi particulièrement riche, la conception de tels agents robotiques.

On peut aussi envisager des coopérations entre robots, par exemple dans le cadre de la compétition de robots footballeurs (RoboCup) [1], présentée par ses organisateurs comme un nouveau problème étalon de l'intelligence artificielle.

I.1.3.6 Agent mobile

Un agent mobile, par opposition à un agent fixe ou stationnaire, peut se déplacer d'un site à un autre en cours d'exécution pour accéder à des données ou à des ressources. Il se déplace avec son code, ses données propres, et son état d'exécution. L'agent décide lui-

s mouvements. Ainsi, la mobilité est contrôlée par le système d'exécution comme dans le cas de la migration de processus dans les systèmes opératoires.

La première plate-forme d'agents mobiles est Telescript [52], née dans la première moitié des années 90. Elle a depuis donné lieu à de nombreux descendants, extensions du langage Java (Aglets, Voyager, etc.). Nous allons voir ce paradigme en détail un peu plus loin dans ce chapitre.

I.2 Les Systèmes Multi-Agents

I.2.1 Origines

Pour résoudre un problème complexe, il est parfois plus simple de concevoir des programmes autonomes et relativement petits (les agents) en interaction, qu'un seul gros programme monolithique.

Dans le même temps, le génie logiciel a évolué vers des composants de plus en plus autonomes. On peut dire alors que les SMA sont la réunion du génie logiciel et de l'intelligence artificielle distribuée [23, 32] avec un apport très important des systèmes distribués.

En plus, les SMA doivent aussi leur existence à un concept similaire de la nature, comme beaucoup d'inventions humaines. Le cas le plus flagrant est la colonie de fourmis [41, 89]. Lorsqu'un objet (une pierre, par exemple) apparaît sur le chemin habituel de la colonie de fourmis, ses individus doivent trouver le moyen le plus simple (et souvent le plus rapide) de contourner ce nouvel obstacle. Les fourmis n'ont ni l'intelligence, ni la taille nécessaire pour savoir s'il vaut mieux passer à droite ou à gauche. Leur comportement est réactionnel à un événement, et les SMA utilisent ce concept dans le cas d'agents réactifs.

I.2.2 Définition

Un système multi-agent est un système caractérisé par son environnement logiciel et matériel, par ses éléments de base (ses agents) et par les mécanismes d'actions et d'interactions existants entre ces éléments et l'environnement en question [32].

L'environnement d'un système multi-agent est défini par une description détaillée de toutes les ressources matérielles (mémoire, ordinateurs, réseaux, etc.) et logicielles (bases de données, systèmes d'exploitation, base de connaissances, etc.) mises à la disposition du système.

ralement à résoudre ou à modéliser un problème ou
ıtation dûun problème par un système multi-agent consiste à distribuer les tâches entre plusieurs entités autonomes pouvant communiquer et collaborer afin dœxécuter la tâche globale à laquelle le système est dédié [82].

I.2.3 Les cinq problématiques des SMA

On peut relever cinq problématiques principales lors de la création des systèmes multi-agents (SMA) [32].

- D'abord, la problématique de l'action : Comment un ensemble d'agents peut agir de manière simultanée dans un environnement partagé, et comment cet environnement interagit en retour avec les agents ? Les questions sous-jacentes sont entre-autres celles de la représentation de l'environnement par les agents, de la collaboration entre agents et de la planification multi-agent.
- Ensuite la problématique de l'agent et de sa relation au monde, qui est représentée par le modèle cognitif dont dispose l'agent. L'individu d'une société multi-agent doit être capable de mettre en î uvre les actions qui répondent au mieux à ses objectifs. Cette capacité à la décision est liée à un "état mental" qui reflète les perceptions, les représentations, les croyances et un certain nombre de paramètres "psychiques" (désirs, tendances...) de l'agent. La problématique de l'individu et de sa relation au monde couvre aussi la notion d'engagement de l'agent vis-à-vis d'un agent tiers.
- Les systèmes multi-agents passent aussi par l'étude de la nature des interactions, comme source de possibilités d'une part et de contraintes d'autre part. La problématique de l'interaction s'intéresse aux moyens de l'interaction (quel langage ? quel support ?), et à l'analyse et la conception des formes d'interactions entre agents. Les notions de collaboration et coopération (en prenant coopération comme collaboration + coordination d'actions + résolution de conflits) sont ici centrales.
- On peut évoquer ensuite la problématique de l'adaptation en termes d'adaptation individuelle ou apprentissage d'une part et d'adaptation collective ou évolution d'autre part.
- Enfin, il reste la question de la réalisation effective et de l'implémentation des SMA, en structurant notamment les langages de programmation en plusieurs types allant du langage de type L5, ou langage de formalisation et de spécification, au langage de type L1 qui est le langage d'implémentation effective. Entre les deux, on

munication entre agents, de description des lois de
ıtation des connaissances.

I.3 Exemples d'applications

Il y a évidemment plusieurs domaines d'application dus au fait que les architectures basées sur les agents fournissent une manière bien particulière pour aborder des problèmes rapidement [79]. C'est pour cette raison que les agents sont largement utilisés dans les domaines tel que: l'industrie, la communication (y compris télécommunication), l'information, la santé, et bien d'autres domaines.

I.3.1 Nouvelles et Informations

Les premières applications d'agent sur le marché concernaient les agents de surveillance. Ces systèmes parcouraient les données et avertissaient l'usager quand un événement d'importance se produisait. E-Watch, ZDNet et Excite sont quelques systèmes qui offrent ce service de nouvelles et d'informations. Les agents de ce type sont nombreux sur les sites boursiers et les sites de commerce électronique.

I.3.2 Shopping

Frictionless[111] est un système qui permet aux usagers de comparer les prix et les caractéristiques des produits lorsqu'ils font des achats sur Internet. L'usager peut remplir une fiche décrivant son profil, puis choisir un produit et ses caractéristiques. Son agent demandera alors ce produit aux différents marchands, puis dressera la liste des produits satisfaisants, ordonnés selon leur correspondance au profil de l'usager, leur prix et leur politiques de vente (livraison, retour, échange, etc.).

I.3.3 Enchères

AuctionBot [112] est un serveur Internet, situé à l'Université du Michigan, qui permet de tenir des enchères sur n'importe quel produit. Une interface sur le site permet aux usagers de créer et de spécifier les caractéristiques de leur agent. Cet agent peut alors prendre part aux enchères sur le site. Les usagers créent une nouvelle enchère en spécifiant le type de l'enchère, un prix de départ et une méthode de résolution en cas d'égalité.

Lœélaboration dœune bibliothèque virtuelle revient en quelque sorte à reproduire, dans le cadre dœun réseau ou dœun système informatique, le modèle des bibliothèques classiques. Ainsi, une bibliothèque virtuelle est dotée d'agents assistants de recherche dœinformations qui mettent leur savoir-faire au profit du client utilisateur. Ayant accès à un ensemble de ressources dœinformations, ces agents assistants de recherche virtuelle, à lœimage des bibliothécaires, jouent le rôle dœintermédiaire entre lœinformation disponible et les utilisateurs. Leur travail consiste, tout comme dans le monde réel, à faciliter la tâche de recherche dœinformations.

I.3.6 Recherche d'informations

Cœest le plus grand champ dœapplication et dœexpérimentation des agents. Les technologies « Push » et les « Bots » sont deux champs de la recherche dœinformation qui utilisent ce paradigme.

I.3.6.1 Les Bots

Un bot [113] est un agent logiciel automatique ou semi-automatique qui interagit avec des serveurs informatiques. Il se connecte et interagit avec le serveur comme un programme client utilisé par un humain, d'où le terme "bot" qui est la contraction de "robot". On utilise les bots pour remplir des tâches systématiques comme par exemple: corriger des fautes d'orthographe, générer du contenu en suivant un modèle, vérifier la disponibilité de news et les télécharger si besoin.

Encarta est l'encyclopédie multimédia de Microsoft, disponible en DVD ou en ligne. La société Conversagent [114] a créé un bot permettant d'accéder à cette encyclopédie via le client de messagerie Microsoft. Pour en profiter, il suffit d'ajouter un contact (fr.encarta@botmetro.net) au logiciel de messagerie instantanée comme s'il s'agissait d'un contact classique. Une fois le contact ajouté, le robot apparaîtra en ligne sous le pseudo "Encarta® Réponses Instantanées". On peut dès lors interroger Encarta en langage naturel, le robot répondant directement dans la fenêtre de conversation du client de messagerie instantanée. Si nécessaire, une fenêtre annexe s'ouvrira pour afficher du contenu complémentaire.

La technologie Push est en fait un ensemble de technologies utilisées pour envoyer de lôinformation à un usager sans quôil nôen fasse la demande. Les médias publics, telles la radio et la télévision, se basent sur ce principe.

Cette technologie permet aussi à lôusager de choisir lôinformation qui lui sera envoyée, ainsi que le moment et le format de lôenvoi. Elle permet également aux entreprises dôenvoyer de lôinformation à des audiences ciblées. Elle offre notamment la possibilité de naviguer sur le web et facilite la distribution des nouvelles versions de logiciels.

Les agents conviennent naturellement à cette technologie, puisquôun agent peut être envoyé sur un serveur pour filtrer lôinformation sur place. Seule lôinformation pertinente est retournée à lôexpéditeur. Le réseau Pointcast [115] est le produit le plus avancé de la technologie « Push » qui utilise des agents. Il permet aux usagers de personnaliser leurs nouvelles sur le sport, la météo et les valeurs boursières. Côest un utilitaire gratuit qui dispose dôune interface graphique et qui délivre les nouvelles sur lôordinateur personnel de lôusager.

I.4 Plates-formes de développement des SMA

Nous avons essayé de résumer les plates-formes les plus connues des systèmes multi agents:

I.4.1 CORMAS (COmmon Resources Multi-Agent System)

CORMAS [116] est un framework de développement pour les systèmes multi-agents, open-source et basé sur le langage de programmation orientée objet SmallTalk. Spécialisé, il est centré sur des problématiques de recherche en sciences du développement et de négociation entre agents.

I.4.2 JADE (Java Agent DEvelopment)

JADE [117] Cêst un framework de développement pour les systèmes multi-agents, open-source et basé sur le langage Java. Il offre en particulier un support avancé de la norme FIPA-ACL [118], ainsi que des outils de validation syntaxique des messages entre agents basés sur les ontologies.

Madkit [119] est une plate-forme multi-agents modulaire écrite en Java et construite autour du modèle organisationnel Agent/Groupe/Rôle. C'est une plate-forme libre, développée au sein du LIRMM [120].

I.4.4 MAGIQUE

MAGIQUE [121] est une plate-forme pour agents physiquement distribués, écrite en Java et fournissant un modèle de communication original d'appel à la cantonade. Dans MAGIQUE, les compétences sont dissociées des agents.

I.4.5 JAgent

JAgent [122] un framework open source réalisé en Java dont l'objectif est de faciliter le développement et le test de systèmes multi-agents.

I.4.6 SMAS/OMAS

(Simulation of Multi-Agent Asynchronous Systems) [123] est une plate-forme de recherche développée par l'équipe d'intelligence artificielle de l'Université de Technologie de Compiègne, France.

I.4.7 JACK

JACK est un langage de programmation et un environnement de développement pour agents cognitifs, développé par la société Agent Oriented Software [124] comme une extension orientée agent du langage Java.

La mobilité est une propriété non exigée des agents (tous les agents ne sont pas obligatoirement mobiles).
Un agent statique est un agent qui s'exécute seulement dans le système où il commence son exécution. S'il a besoin d'informations non disponibles dans le système, ou a besoin d'interagir avec un agent dans un système différent, il utilise un mécanisme de communication tel que RPC [4, 18].

Par contre, un agent mobile n'est pas lié au système dans lequel il débute son exécution. L'agent mobile est capable de se déplacer d'un hôte à un autre dans le réseau. Il peut transporter son état et son code d'un environnement vers un autre dans le réseau où il poursuit son exécution.

I.5.1 Code mobile et agent mobile

Les interactions de type client/serveur (envoi de messages, appel de procédure à distance, appel de méthode à distance et invocation distante) [22, 80] se basent sur des interactions distantes entre le client et le serveur.

Il existe une autre manière de concevoir les interactions en introduisant la mobilité. Cette mobilité peut être vue comme une variation du modèle clients/serveur classique.

Nous pouvons distinguer les différentes formes existantes de la mobilité en nous basant sur le déplacement d'un ou plusieurs éléments composant le schéma client/serveur (code, ressources et unité d'exécution) [22]. Nous parlons alors de l'envoi du savoir faire, récupération du savoir faire et migration de processus.

Le schéma de mobilité des agents mobiles dérive de deux domaines différents, i.e les agents venant de l'intelligence artificielle avec les systèmes multi-agents et des systèmes distribués avec la migration de processus (déplacement du code, ressources et unités d'exécution) [74, 75].

La mobilité d'agent permet donc de rapprocher client et serveur, et en conséquence de réduire le nombre et le volume des interactions distantes (en les remplaçants par des interactions locales), de spécialiser des serveurs distants ou de déporter la charge de calcul d'un site à un autre.

Dôaprès Briot et Y. Demazeau [21] : Un agent mobile est un agent capable de se déplacer dans son environnement, qui peut être physique (réel ou simulé) ou structurel (niveaux d'exécution par exemple). Un agent mobile dispose donc de mécanismes assurant sa mobilité.

I.5.3 Pourquoi les Agents Mobiles ?

Les agents mobiles présentent plusieurs avantages pour dépasser les limitations du modèle client serveur. En effet, les limitations de la machine cliente, telles que la puissance des processeurs, son débit et la taille de son espace mémoire, peuvent être atténuées si l'agent s'exécute prés des données [82]. En plus, les agents mobiles bénéficient des avantages suivants :

I.5.3.1 Réduction de la charge Réseau

Il est généralement plus avantageux, en terme de performances, de faire voyager du code plutôt que des données. Les serveurs contenants des données procurent un ensemble fixe d'opérations. Un agent mobile peut étendre cet ensemble pour les besoins particuliers d'un traitement.

I.5.3.2 Fiabilité

La vie d'un programme classique est liée à la machine où il s'exécute. Un agent mobile peut se déplacer pour éviter une erreur matérielle ou logicielle, ou tout simplement un arrêt de la machine.

I.5.3.3 Adaptation dynamique du système aux problèmes

Les agents peuvent se répartir eux-mêmes sur des machines du réseau, afin de mieux tenir compte de l'état du système au moment où ils doivent exécuter leurs tâches.

e la tolérance aux pannes

i s'y trouve peut changer de machine pour terminer sa tâche en cours.

I.5.3.5 Encapsulation de protocoles

Dans les systèmes distribués, les protocoles définissent comment les messages et les données sont échangés. Pour modifier un protocole, il faut changer le code sur toutes les machines du système. Les agents encapsulent les protocoles, changer de protocole revient donc à interagir avec un nouvel agent.

I.5.3.6 Autonomie des composants et l'exécution synchrone/asynchrone

Les terminaux mobiles (PDA, ordinateurs portables, téléphones, ...) ne sont pas toujours connectés au réseau. Les systèmes qui nécessitent des connexions permanentes ne sont pas adaptés dans ce cas.

Avec des agents, les terminaux mobiles peuvent se connecter pour vérifier s'ils ont des messages. Un agent peut être envoyé sur le terminal mobile au moment de la connexion et travailler sur ce dernier après avoir interrompu la connexion. Le même agent peut attendre la prochaine connexion pour envoyer des informations sur le réseau signifiant, par exemple, qu'une tâche a été terminée.

I.5.4 Cycle de vie d'un agent mobile

I.5.4.1 Eléments nécessaires pour une plate-forme d'agents mobiles

Les différents éléments devants apparaître dans l'élaboration d'une plate-forme [90], sont schématisés sur la figure 1.1 :

- ***Les agents***

  Ils sont définis par un identifiant unique au sein de la plate-forme et d'un état d'exécution comprenant le code et les données qu'ils transportent.

- ***Les places***

  Une place peut contenir plus d'un agent et elle définit un environnement d'exécution vers lequel un agent sera orienté pour gérer son exécution.

Encore appelées « ports d'accueil » des agents, les agences représentent les environnements où évoluent les agents. Une agence constitue le coeur de gestion de la plate-forme en assurant la délégation de l'exécution aux places qui la composent ainsi que l'administration, le contrôle, le transport effectif et la communication des agents grâce au système d'exploitation sous-jacent.

- ***Les régions***

Il s'agit du regroupement de plusieurs agences, pas nécessairement de même type, appartenant à un même domaine d'expertise, d'activité ou autre. Le but est de faciliter les activités d'administration. On peut associer par exemple une région à des politiques d'accès aux services.

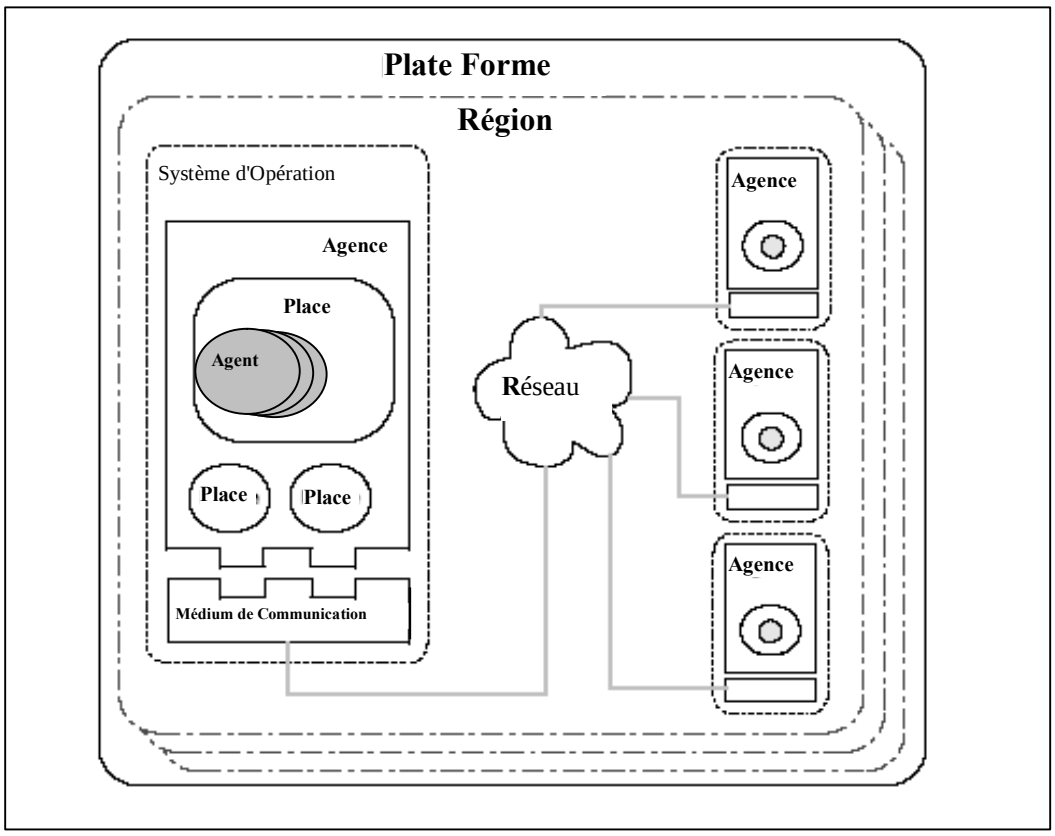


Fig.1.1 Les éléments d'une plate-forme d'agents mobiles

uction de l'agent mobile

a) La création

La création de l'agent mobile se fait en trois étapes :

- ***L'instanciation et l'attribution de l'identificateur***

Le code de la classe agent est chargé puis exécuté, l'objet agent est instancié et la place assigne un identificateur unique à l'agent.

- ***L'initialisation***

L'agent peut s'initialiser en utilisant les arguments d'initialisation fournis par son créateur. Une fois l'initialisation achevée, l'agent est complètement installé dans la place.

- ***L'exécution autonome***

Après l'installation, l'agent peut commencer son exécution. Il est maintenant capable de s'exécuter indépendamment des autres agents dans la même place.

b) La migration

Le processus de transfert peut être initié par l'agent lui même, par un autre agent résidant dans la même place ou par un agent ou système non agent résidant dans une autre place. L'agent est ensuite transféré de sa place courante pour être reçu par la place spécifiée. (Voir Figure 1.2)

La place d'origine et la place destination gèrent le processus de transfert. Quand la place d'origine contacte la place de destination, celle-ci peut soit réaliser le transfert, soit envoyer un message d'échec à la place d'origine. Si la place d'origine ne peut pas contacter la place destination, elle doit retourner un message d'échec à l'agent.

- ***L'envoi de l'agent***

Quand l'agent mobile se prépare pour son voyage, il doit être capable d'identifier sa destination. Si la place est non spécifiée, l'agent s'exécutera dans la place par défaut, sélectionnée par le système d'agent de destination. Une fois la localisation de la destination est établie, l'agent mobile informe le système d'agent local qu'il veut se transférer au système d'agent de destination.

Quand le système d'agent reçoit la requête de transfert, il doit entreprendre les actions suivantes :

Lôagent est dôabord prévenu à propos de son transfert, et un temps lui est alloué pour terminer sa tâche courante. Une fois ce temps écoulé, son thread dôexécution est arrêté.

2) ***Sérialiser l'agent***

Lôagent (composé de son état et de la classe agent) est sérialisé par un moteur. La sérialisation est le processus de création dôune représentation persistante de lôobjet agent qui peut être transportée à travers le réseau. La sérialisation de lôagent peut inclure son état dôexécution.

3) ***Encoder l'agent sérialisé***

Le moteur encode lôagent sérialisé pour le protocole de transport choisit.

4) ***Transférer l'agent***

Le moteur établit une connexion réseau avec lôhôte spécifié et entame le processus de transfert.

- ***Réception de l'agent :***

Avant que le transfert puisse avoir lieu, lôorigine de lôagent doit sôauthentifier auprès du moteur de réception. Lorsque lôauthentification est réussie, le processus de transfert peut alors commencer :

1. Réception de lôagent.
2. Décodage de lôagent.
3. Dé-sérialisation de lôagent : La représentation persistante de lôagent est dé-sérialisée. La classe agent est instanciée et lôétat de lôagent transféré est rétabli.
4. Reprendre lôexécution de lôagent : Lôagent recrée est notifié dans sa place destination. Un nouveau thread dôexécution lui est assigné.

c) La destruction

Ce processus peut être initié par lôagent lui même, par un autre agent ou par un système non-agent résidant ou non dans la même place. Lôagent peut aussi être renvoyé de sa place par le système pour lôune des raisons suivantes :

vie ".

é ou référencé.

- Violation des règles de sécurité.
- Le système est arrêté.

Le processus de destruction est amorcé en deux étapes :

- La préparation : Une chance est donnée à lôagent de finir sa tâche courante avant dôêtre renvoyé.
- Suspension de lôexécution : La place suspend lôexécution de lôagent.

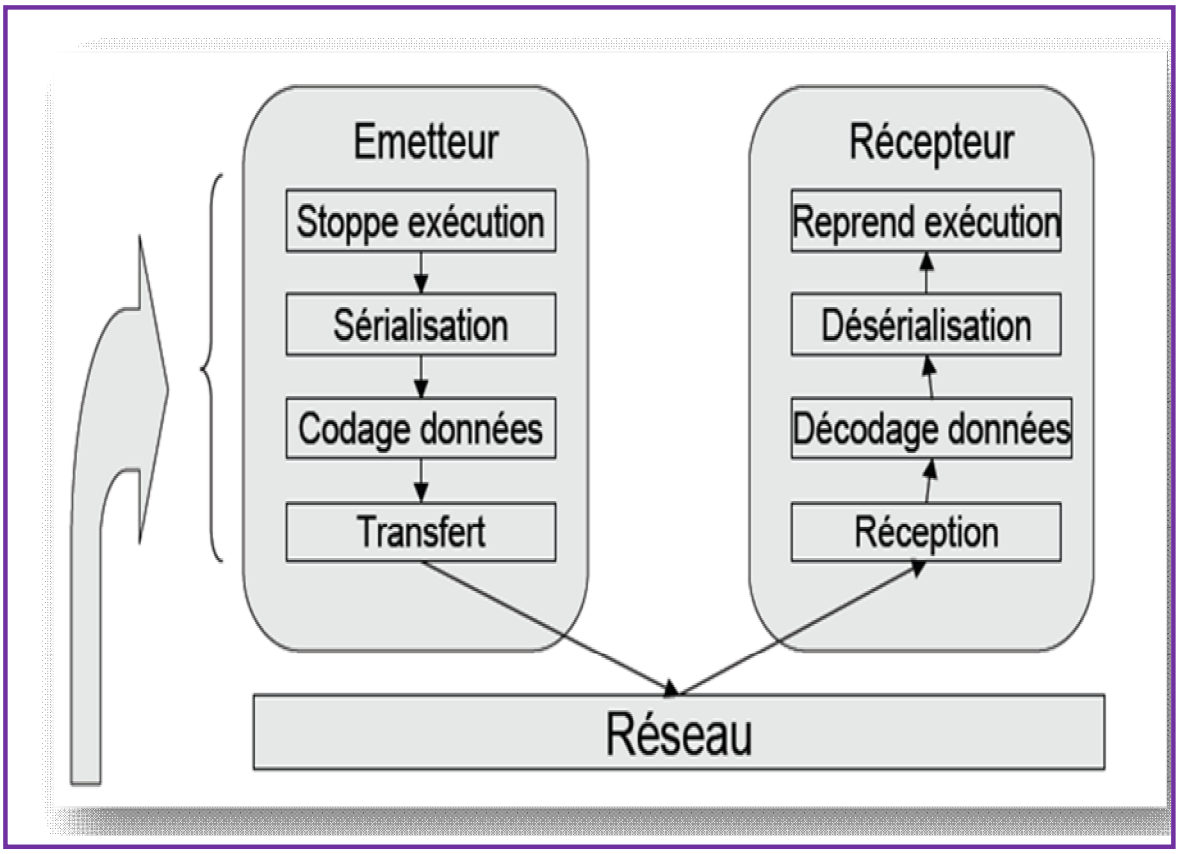


Fig.1.2 Le transfert de l'agent mobile

s très utiles lorsque lôon souhaite mettre en place des explorations de réseau, représenter un utilisateur nomade ou encore lorsque les environnements sont autonomes et décentralisés [72]. Nous allons présenter quelques applications basées sur les agents mobiles [82].

I.5.5.1 Commerce électronique

Les principes dôautonomie et de délégation sont largement employés dans le commerce électronique. Les agents mobiles sont donc naturellement adaptés à ce type dôapplication. Lôagent agit pour le compte dôune personne et applique différentes stratégies pour remplir les services demandés. La mobilité de lôagent lui permet de se déplacer de site en site pour dialoguer et négocier avec les services présents afin de réaliser la meilleure transaction possible.

I.5.5.2 Recherche et filtrage d'informations

Le domaine de recherche et filtrage de l'information représente le plus grand champ dôapplication et dôexpérimentation des agents mobiles. Les premiers systèmes commerciaux utilisant des agents étaient des agents ñmoniteursò. Cô étaient des programmes qui alertaient lôutilisateur quand une information intéressante apparaît. Alexa [125] est une barre dôoutils dôaide gratuite à la navigation. Elle procure des informations statistiques et des liens sur chaque site visité. Elle aide également au magasinage en ligne en vérifiant lôidentité dôun site de commerce à partir des contacts donnés sur la page.

I.5.5.3 Services de télécommunication

La fonctionnalité de mobilité prend toute son importance dans le domaine des télécommunications. Les agents mobiles peuvent y être utilisés pour couvrir toutes les couches des protocoles de communication, de la maintenance de réseau, jusquôaux applications mobiles, en suivant lôusager dans ses déplacements.

Un ñPersonal Communicator Agentò (PCA) est un agent mobile chargé de délivrer un message au destinataire, quel que soit son appareil (téléphone, ordinateur, portable ou téléphone sans fil). Le PCA dôun utilisateur doit pouvoir recevoir les messages et les conduire sans interruption dans des réseaux hétérogènes à lôutilisateur. Par exemple, si le

urgent à un utilisateur est via un téléphone sans fil, essage textuel en message vocal. La mobilité permet à ces agents de suivre lôutilisateur même quand il change de machine.

I.5.5.4 Robotique

Les agents mobiles peuvent aussi servir à aider le comportement de robots physiques. À lôaide de capteurs, les robots basent leurs comportements sur des agents qui traduisent lôinformation obtenue pour définir les actions à suivre (tourner, avancer etc..). Astérix [140] est un exemple de robot basé sur les agents. Il permet un dépistage de haut niveau en utilisant une opinion basée sur un comportement réactionnaire.

I.5.5.5 Surveillance

Les agents surveillants veillent sur des sites ou des thèmes définis par lôusager. Ils peuvent être utilisés pour être automatiquement alertés des dernières actualités dans un domaine donné. Les agents mobiles sont ici adéquats pour voyager dans le réseau et effectuer des tâches périodiques de surveillance.

I.5.5.6 Administration des réseaux

Le domaine de lôadministration des réseaux est également lôobjet de nombreuses recherches. Beaucoup pensent que le futur des réseaux réside dans une plus grande intelligence, pour plus dôadaptabilité et de mobilité et que cette évolution passe par les agents mobiles.

En effet, lôadministration de réseau est par nature asynchrone et distribuée; de plus, il est souvent important dôavoir une vue locale du système pour pouvoir déterminer les causes et conséquences dôun problème. Lôadministrateur doit alors se déplacer vers des machines lointaines qui nécessitent des tâches de maintenance ou de mise à jour. Ces tâches deviennent difficiles avec lôaugmentation du nombre de machines, d'où l'introduction des agents mobiles qui rendent cela moins complexe et plus performant.

ts mobiles

èvement les plates-formes les plus fameuses, conçues pour les agents mobiles [6, 97] :

I.6.1 Telescript

Telescript, développé par General Magic [103], peut être considéré comme la référence dans le domaine. Il est parlant que son inventeur, Jim White, tient un brevet sur la notion même d'agents mobiles. Telescript était un des seuls systèmes supportant le transfert de l'état d'exécution des agents [52].

Toutefois, le système, conçu pour une application industrielle, exigeait des ressources importantes, était plutôt cher, et offrait une fonctionnalité pas encore demandée par le marché. Par conséquent, le système n'existe plus et sa valeur est surtout historique.

I.6.2 Odyssey

Le successeur de Telescript, développé également par General Magic, s'appelle Odyssey [126]. Différent du premier, il est basé sur Java [52]. Il est largement inspiré par Telescript, mais n'a pas exactement la même fonctionnalité. Par exemple, Odyssey ne supporte pas le transfert de l'état d'exécution. Le système n'a toujours pas dépassé la version *bêta*, et ne joue visiblement pas le rôle important que jouait Telescript.

I.6.3 Concordia

Concordia, qui est l'offre d'agents mobiles de Mitsubishi Electric Information Technology Center America [127], est un environnement de développement, d'exécution et de gestion d'agents mobiles. Pour ce qui est des systèmes requis pour son fonctionnement, Concordia est compatible avec Windows95/98/NT et Solaris. Il requiert, aussi, Le JDK (Java Development Kit). Par ailleurs, Concordia est compatible avec tous les réseaux qui utilisent le protocole TCP/IP.

eloppé par ObjectSpace en 1997 [128]. Voyager implémente en Java une plate-forme pour systèmes distribués. Cette plate-forme inclut un ORB (Object Request Brocker) [35, 73] supportant les objets mobiles et les agents autonomes.

I.6.5 Agent TCL

AgentTCL [129] est lôun des premiers projets dôagents mobiles développés par lôuniversité Dartmouth (Angleterre). AgentTCL était basé sur le système Tcl (Tool Command Language) et plus tard il a été étendu pour supporter Java, Scheme et C/C++.

I.6.6 TACOMA

Le projet TACOMA [130] (Tromsô And COrnell Moving Agents) fournit un support dans les systèmes dôexploitation pour la programmation dôagents mobiles. Le modèle ne fait pas de distinction entre un client, un serveur et un agent. Il propose simplement la notion dôagents et des mécanismes de migration et de communication. Un agent TACOMA est un processus exécutant un script Tcl qui est capable de se déplacer dôune manière autonome dôune machine à une autre.

I.6.7 MAP

MAP [131] (**M**obile **A**ssistant **P**rogramming) a été publié en 1995 pour lôaccès nomade aux réseaux dôinformations. Il présente un modèle dôagents mobiles pour la programmation et lôexécution efficace dôactions par délégation dans un réseau de grande envergure pour lôaccès à lôinformation depuis des stations nomades.

I.6.8 JATLite

JATLite [132] (Java Agent Template, Lite) est une plate-forme agents mobiles implémentée en Java. Elle est compatible avec Windows 95, Windows NT, Soloris, UNIX, et elle nécessite le Jdk1.1 ainsi que des navigateurs Internet (Internet Explorer, Netscape).

3] développée par IKV++ (Innovation Know-how Vision++) en 1997 est un environnement dôimplémentation et dôexécution pour agents mobiles.

I.6.10 Aglet

Le système dôIBM Aglet Workbench [134] propose un environnement de programmation dôagents mobiles dans le langage Java qui est issu du modèle dôagents itinérants.

Un Aglet (AGent appLET) est un objet mobile qui dispose de son propre thread de contrôle, auquel on peut envoyer des événements et qui a la possibilité de communiquer par lôintermédiaire de messages.

I.6.11 JADE

JADE (Java Agent DEvelopment Framework), est spécifique à CSELT (Centre Studi E Laboratori Telecomunicazioni) de lôuniversité de Parma (Italie) [11,12].

Cet environnement de développement dôapplications à base dôagents est implémenté entièrement en Java. JADE est conforme aux normes et spécifications de FIPA (Foundation For Intelligent Physical Agents), elle hérite alors son architecture, ses protocoles, ses services et ses mécanismes de transport et de communication.

JADE est compatible avec la plupart des configurations matérielles et logicielles (Windows, Unix). Le seul système nécessaire pour son fonctionnement est la version 1.2 de Java (JRE ou JDK).

Lôagent JADE peut être implémenté en Java ou en Jess [135] (Java Expert System Shell). Ce dernier est un système expert qui permet dôimplémenter des architectures dôagents à base de règles.

Les agents JADE communiquent à travers le langage ACL (Agent Communication Language) de FIPA [118], dans le cas où ces agents sont hétérogènes, ils coopèrent par négociation.

intérêt tout particulier dans les domaines de recherche portant sur les applications réparties. Très rapidement, cette nouvelle méthode de programmation a été évaluée afin de voir ce quêelle pouvait apporter comme caractéristiques propres, et ce quêelle permettait dôaméliorer par rapport aux méthodes de programmation plus classiques [27].

Dans ce qui suit, nous discutons les différents points forts des agents mobiles ainsi que les points faibles, spécialement la sécurité qui est une caractéristique primordiale, et son impuissance peut freiner lôutilisation de cette technologie.

I.7.1 La conception

Du point de vue de la conception, les agents mobiles représentent à la fois une méthode permettant de mieux caractériser certaines applications, mais ils apportent aussi une complexité accrue par rapport au classique client/serveur bien mieux maîtrisé.

Pour commencer, en règle générale, les concepteurs préfèrent avoir une méthode permettant de décrire facilement un comportement réel. Avec la méthode classique, il est très contraignant de décrire des algorithmes dôexploration de réseau, ou bien encore de caractériser les déplacements des utilisateurs nomades [27].

Avec les agents mobiles, les concepteurs disposent dôune méthode qui permet de décrire naturellement ce genre de comportement. Ainsi, on peut facilement mettre en place un déploiement et/ou une maintenance dôapplications sur un réseau, ou encore suivre les utilisateurs dans leurs déplacements.

En plus, les agents possèdent une capacité de traitement spécifique leur permettant de sôadapter à leur environnement. Généralement, dans le modèle client/serveur, le service est caractérisé préalablement par une interface stricte et/ou avec un protocole dôutilisation bien défini. Ainsi, si le client nôa pas une connaissance de la description du service, il ne pourra pas lôutiliser. Par contre, les agents pourront eux sôadapter aux caractéristiques des services afin dôexprimer leurs requêtes.

ande une communication sécurisée, lôagent pourra tre à jour sa pile de communication et dialoguer avec le service. On peut aussi tout à fait imaginer que lôagent pourra sôadapter aux conditions du réseau en se séparant dôune partie de ses fonctionnalités pour sôadapter aux terminaux pauvres en ressources.

I.7.2 Le développement

Sachant que les agents doivent sôadapter aux conditions de lôenvironnement, les développeurs sont confrontés à un éventail de possibilités bien trop large [53]:

I.7.2.1 Manque de standardisation

À l'heure actuelle, il existe un très grand nombre d'intergiciels [138] pour agents mobiles, il est pratiquement difficile de parler de standardisation. En plus, cette difficulté se retrouve aussi dans les interactions entre agents, ainsi il nôexiste pas de langage partagé par toutes les plates-formes. Ceci représente un handicap sérieux, car les agents ont besoin dôexprimer les caractéristiques des services quôils recherchent et surtout dôobtenir des réponses précises sur leur localisation ainsi que sur la manière de les utiliser.

I.7.2.2 Complexité de mise au point des programmes

La plupart des environnements de programmation possèdent des outils permettant de suivre les différents éléments dôune application durant son exécution afin dôen trouver les erreurs. Cependant, ce genre dôoutil sôutilise parfaitement avec des éléments statiques mais est difficilement applicable dès le moment où les éléments se déplacent.

Ceci est particulièrement vrai dans les architectures hybrides et/ou à grande échelle où il est quasiment impossible de surveiller lôintégralité dôun système. Ainsi, il peut sôavérer impossible de récupérer lôétat dôerreur dôun élément se trouvant sur un site déconnecté.

I.7.2.3 Difficulté de teste et de vérification

Le troisième problème important, en relation avec le débogage, est la difficulté de mettre en place une vérification des applications à base dôagents mobiles. La technique la plus utilisée encore de nos jours, bien quôelle ne soit pas totalement satisfaisante, est sans

ode, on va mettre à lôépreuve une application en lui férents points dôentrée dans le but de simuler le comportement utilisateur et de recouvrir une plage de situations la plus large possible. Si le test sôapplique assez naturellement dans des programmes classiques (non répartis), il représente un domaine de recherche à part entière dès quôil sôagit des applications réparties.

Pour vérifier les applications construites sur des agents mobiles, il faut utiliser une méthode permettant de garantir le comportement et les interactions des agents avant leur déploiement. Lôanalyse statique de programme permet de déterminer statiquement, sans exécuter un programme, des propriétés qui seront satisfaites lors de lôexécution.

Notons tout de même que lôanalyse statique a fait ses preuves dans le cadre de la programmation classique mais quôelle constitue toujours un domaine de recherche pour les applications réparties.

I.7.3 La sécurité

Dôun point de vue logiciel, la sécurité consiste à empêcher des accès et/ou des modifications non-autorisés aux éléments dôun système informatique.

Lorsque lôon souhaite avoir un système sûr, on met en place une politique de sécurité qui doit garantir la confidentialité (les données des éléments ne sont pas divulguées aux demandeurs non autorisés) et lôintégrité (les éléments ne sont pas modifiés par des demandeurs non autorisés).

Pour ce qui est des agents mobiles, dôun point de vue de la sécurité, nous sommes face à un problème qui nôest pas encore intégralement résolu. Côest dôailleurs le principal argument avancé pour expliquer la faible utilisation de ce paradigme [24, 88, 101].

En effet, les agents mobiles représentent un nouveau champ dôinvestigation pour le domaine de recherche en sécurité, dôune part dans la protection des sites vis-à-vis des agents malveillants, qui est un domaine bien maîtrisé, et dôautre part dans la protection des agents vis-à-vis des sites malveillants, et côest notre point d'intérêt.

La programmation par agents mobiles est un paradigme de programmation des applications réparties, susceptible de compléter ou de se substituer à d'autres paradigmes plus classiques, tel que le client/serveur. Elle est d'un grand intérêt pour la mise en î uvre d'applications dont les performances varient en fonction de la disponibilité et de la qualité des services et des ressources, ainsi que du volume des données déplacées.

Le concept d'agent mobile facilite en effet la mise en î uvre d'applications dynamiquement adaptables, et il offre un cadre générique pour le développement des applications réparties sur des réseaux de grande taille qui recouvrent des domaines administratifs multiples.

Le but de ce chapitre était de connaître une partie du contexte de notre étude, à savoir les agents mobiles. Nous avons vue leurs avantages à faciliter la programmation distribuée, mais la grande faiblesse de ce paradigme est bien la sécurité.

Le prochain chapitre est consacré à bien investiguer ce domaine et à présenter les différentes solutions existantes.

2 SECURITE DES AGENTS MOBILES

Il existe principalement deux axes de recherches dans le domaine de la sécurité des agents mobiles : d'une part, la protection des hôtes vis-à-vis des agents malveillants et d'autre part la protection des agents vis-à-vis des hôtes malveillants. Le premier axe est bien dominé, tandis que le deuxième ne l'est pas encore.

Dans ce chapitre on s'intéresse à la sécurité des agents mobiles. Nous allons d'abord entamer la protection des plates-formes contre les agents malicieux, commençant par citer les différentes attaques possibles ainsi que les approches proposée pour les éviter. La deuxième partie du chapitre est consacrée à la protection des agents mobiles, qui est notre intérêt. Nous donnerons un exemple pour bien comprendre les différents types d'attaque. Ensuite, nous discuterons les différentes approches existantes pour protéger les agents mobiles contre les attaques des sites visités.

...nes

En général, on distingue quatre types de menaces ou d'attaques [66] : les attaques de l'agent contre les plates-formes, les attaques de la plate-forme contre un agent, les attaques d'un agent contre un autre agent, et d'autres entités contre un agent ou une plate-forme. Ce qui nous intéresse dans cette thèse c'est les attaques Plate-forme Agent, mais pour commencer, nous allons mettre le point sur les attaques Agent/Plate-forme, ainsi que les différentes techniques conçues pour les éviter.

II.1.1 Attaques Agent/Plate-forme

Il existe plusieurs attaques d'un agent contre une plate-forme. Par exemple, un agent hostile peut tenter d'avoir accès à des ressources sans autorisation de la plate-forme, consommer trop de ressources (processeur ou mémoire) ou tenter de se faire passer pour un autre agent [93]. Dans la section suivante, nous présenterons les différentes attaques qu'un agent hostile peut mener contre une plate-forme [56, 91].

II.1.1.1 Mascarade

Une mascarade a lieu quand un agent non autorisé prétend être un autre agent. Elle peut avoir pour but d'obtenir des ressources auxquelles l'attaquant n'a normalement pas accès ou de faire endosser la responsabilité de certaines actions à un autre agent.

II.1.1.2 Déni de service

Il se produit quand un agent consomme trop de ressources (processeur, bande passante des connexions réseau,..). Ces attaques peuvent être intentionnelles ou non (erreur de programmation). Le problème avec l'agent mobile est que le code est écrit en principe en dehors de la plate-forme qui l'exécute et peut contenir du code malveillant.

II.1.1.3 Vol d'information

Quand un agent arrive dans une plate-forme, il faut l'authentifier, puis le soumettre à la politique de sécurité le concernant. Cette pratique permet d'empêcher l'accès d'utilisateurs ou de processus non autorisés à des ressources protégées. En cas d'accès non autorisé, il y'aura un vol d'information.

t le cas où un agent détruit ou modifie des ressources ou des services en les reconfigurant, en les modifiant ou en les effaçant de la mémoire ou du disque. Tous les autres agents sur la plate-forme qui utiliseront ce service ou cette ressource en subiraient les effets.

II.1.1.5 Harcèlement

Un agent peut déranger les usagers par des attaques répétées : un exemple est lôaffichage à répétition dôimages publicitaires non demandées à lôécran de lôhôte sur lequel il sôexécute.

II.1.1.6 Ingénierie sociale

Lôingénierie sociale correspond à la situation où lôagent mobile manipule les utilisateurs ou les hôtes en utilisant la désinformation ou la coercition. Par exemple, un agent mobile peut demander un mot de passe sous la fausse autorité de lôadministrateur du système.

II.2. Techniques de protection des plates-formes

Le concept dôagent mobile pose des problèmes dôun type nouveau. Dans une démarche de reconnaissance des éléments qui distinguent la sécurité des systèmes conventionnels de celle des agents mobiles, quatre suppositions non respectées par les agents mobiles ont été évoquées [76] :

- Lorsquôun programme effectue une action sur un système conventionnel, il est facile dôidentifier la personne qui a effectué cette action ou à qui elle peut être attribuée.
- Tous les logiciels proviennent de sources facilement identifiables et généralement de confiance.
- La plupart des menaces proviennent dôattaquants qui exécutent des programmes dans le but dôobtenir des résultats non autorisés.
- Les programmes ne se transmettent que si les personnes le commandent intentionnellement.

Le but de la protection de la plate-forme est dôempêcher lôagent dôinterférer avec son fonctionnement. En effet, lorsqu'on désire protéger l'hôte contre un code malicieux potentiel, la mobilité du code impose les contraintes suivantes:

authentifiée

vers le réseau à divers manipulations, donc l'hôte doit vérifier l'intégrité du code qu'il vient de recevoir

3 Les actions exécutées par l'agent doivent être limitées par un contrôle d'accès.

La protection des hôtes visités contre les attaques menées par les agents mobiles malveillants est un problème qui est aujourdôhui assez bien maîtrisé, et de nombreuses solutions permettent maintenant de se prémunir contre plusieurs attaques.

La pluparts de ces solutions sont basées sur le modèle de forteresse, selon lequel on cherche à protéger lôhôte grâce à un système fermé qui nôest accessible que par des interfaces bien définies. Dans la section suivante, on présente quelques unes de ces solutions.

II.2.1 Moniteur de référence

Il sôagit dôun programme contrôlant lôaccès de lôagent mobile aux informations, aux ressources et aux services de la plate-forme. Ce moniteur de référence applique un certain nombre de règles, regroupées dans la politique de sécurité de la plate-forme, pour lôaccès de lôagent à ces ressources. On suppose que lôimplémentation du moniteur est toujours appelée (c.à.d. quôil est impossible dôobtenir une ressource sans lôappeler), résistante aux altérations et suffisamment petite pour être analysée et testée [56].

En plus, lôimplémentation des moniteurs de référence emploi un nombre de techniques de sécurité conventionnelles applicables à lôenvironnement de lôagent telles que :

ÅDes mécanismes dôisolation des processus les uns des autres et du processus de contrôle,

ÅDes mécanismes de contrôle dôaccès des ressources de calcul,

ÅDes méthodes cryptographiques pour chiffrer les échanges dôinformations,

ÅDes méthodes cryptographiques pour identifier et authentifier les utilisateurs, lôagent et les plates-formes.

ÅDes mécanismes dôaudit de sécurité des événements pertinents survenant au niveau de la plate-forme agent.

ox) [100] est inspirée du principe des démineurs qui utilisaient un coffre de sable pour faire exploser les engins en toute sécurité. Elle consiste à exécuter le code de l'agent mobile dans un environnement restreint en terme de privilèges et de ressources [36, 37]. Le code est donc exécuté dans une sorte de « carré de sable ». Par exemple, un agent ne pourra pas faire la modification des ressources de la plate-forme, lôaccès au système de fichiers, lôouverture dôune connexion réseau, l'accès à des propriétés ou à des programmes du système local [70]. Les agents ont, en conséquence, des privilèges limités et sont interprétés de manière sécurisée. Ainsi, un hôte peut exécuter un agent suspect dans le bac à sable sans trop se soucier des problèmes de sécurité. On peut utiliser des interpréteurs de code pour mettre en place cette approche.

Comme exemple, nous pouvons citer les applets Java exécutés à lôintérieur dôun navigateur Web. Java utilise cette technique et fournit donc une solution partielle au problème posé par les agents malicieux. Lôinterpréteur de Java comprend trois principaux composants [37, 71, 48]:

- ***ClassLoader :*** Convertit le code distant en structures de données qui peuvent être rajoutées à la hiérarchie de la classe locale.
- ***Verifier :*** Avant que le code distant soit chargé, *Verifier* réalise un ensemble de vérifications afin dôassurer que seulement le code légitime est exécuté.
- ***SecurityManager :*** Enfin, les opérations effectuées par les classes distantes sont contrôlées par le *SecurityManager*.

Parmi les inconvénients de cette technique est quôelle augmente le temps dôexécution dôun code distant légal. Par ailleurs, les applications qui sont exécutées dans cet environnement restreint sont elles mêmes rarement utiles.

II.2.3 Contrôle d'accès et d'autorisation

D'une part, les programmes (dans notre cas les agents mobiles) qui s'exécutent sur un système ont besoin d'accéder à des ressources pour accomplir leurs tâches. Dôautre part, lôenvironnement d'exécution de lôagent mobile est restreint pour des soucis de sécurité. Les ressources dont il a besoin pour accomplir sa tâche doivent donc être rigoureusement contrôlées. La technique du contrôle dôaccès et dôautorisation se rapproche du « carré de sable » [58], car elle se base sur un moniteur de référence qui donne des autorisations à un

re de ressources, en fonction du résultat de son

Pour savoir quelle autorisation donner à quel agent, le moniteur de référence applique des règles de la politique de sécurité. Ces règles touchent toutes les ressources dont un agent peut avoir besoin : accés au réseau, aux disques, etc.

Le problème du carré de sable ne se pose plus, car une application avec des besoins forts ou particuliers en ressources pourra les avoir si la politique de sécurité le permet.

Comme exemple, dans le langage Java, le moniteur de référence est le « security manager ». Il prend en charge tous les accés aux ressources. Par exemple, un agent mobile qui tente de lire un fichier dont il ne détient pas les droits de lecture sera arrété par le moniteur de référence.

II.2.4 Authentification de l'agent

Le problème d'authentification des agents mobiles est similaire à celui qui se pose dans les environnements répartis. En effet, deux buts sont en général poursuivis dans ce contexte : vérifier l'intégrité du code et authentifier ses auteurs et utilisateurs.

L'approche proposée consiste à attribuer une signature électronique à l'agent mobile (signature du code) avec un algorithme cryptographique à clé publique. La signature du code intervient lors de la création d'un agent, où son créateur le signe numériquement afin qu'il puisse s'identifier durant ses déplacements. Cette technique permet d'obtenir une authentification de haut niveau pour les hôtes. Elle assure aussi l'intégrité du code pour l'hôte visité [57, 60, 83].

L'inconvénient de cette méthode est qu'en fait un code signé est soit une garantie pour l'utilisation de toutes les ressources comme il peut ne pas du tout être exécuté.

De plus, cette solution ne résout pas le risque fondamental lié au comportement de l'agent mobile et qui laisse le consommateur vulnérable aux dommages dus au code défectueux ou malicieux provenant d'une source fiable. Autrement dit, la signature du code ne peut révéler ce qu'il peut faire ni garantir que son exécution soit sûre.

II.2.5 Vérification du code

Le but de cette technique est de vérifier si le code d'un agent mobile est un programme valide pour permettre d'obtenir une garantie sur sa sémantique à travers l'analyse de sa structure, ou de son comportement, en fonction d'une politique de sécurité donnée. Si un

ons illégales dans un agent, il empêche (ou refuse) son

Un exemple de refus pourrait être un programme qui exécute une chaine de caractères aléatoire comme un segment de programme. Dans les langages interprétés, cette technique est parfois nommée « interprétation du code inoffensifs ».
Dans Java, il existe un vérificateur de « byte-code » qui effectue une vérification sur les types et fait des vérifications pendant lœexécution.

II.2.6 Estimation de l'état

La technique dœestimation de lœétat (state appraisal) [30] est utilisée pour protéger une plate-forme dœun agent dont lœétat pourrait être modifié par une autre plate-forme défaillante ou ennemie. A ce titre, l'estimation de l'état pourrait aussi apparaître dans les techniques de protection de lœagent, car elle permet de détecter une altération de son état.

Le producteur et/ou lœémetteur de lœagent mobile fournissent les fonctions dœévaluation pour faire une estimation de l'état et constituent une partie du code de lœagent.
Ils appliquent des contraintes dœétat afin de réduire la responsabilité et/ou les coûts de contrôle. Lorsque le producteur et lœémetteur de lœagent mobile le signent, les fonctions d'évaluation sont protégées contre la modification non détectable.

Une plate-forme agent utilise ces fonctions pour vérifier lœétat dœun agent arrivé et déterminer les privilèges que cet agent peut obtenir pour sœexécuter. Ces privilèges se basent sur les résultats des fonctions dœévaluation et sur la politique de sécurité qui est instaurée. En plus de s'assurer qu'un agent n'est pas devenu malveillant pendant son itinéraire, l'évaluation d'état peut également être employée pour désarmer un agent volontairement modifié.

Lœapplication de cette théorie nœest pas clairement définie puisque lœespace état dœun agent peut être assez large. En plus, il n'est pas facile de formuler les propriétés de sécurité appropriées pour l'agent mobile et d'obtenir une fonction d'évaluation d'état qui les garantit [94].

Enfin, plusieurs attaques subtiles demeurent difficiles à détecter car le producteur et l'émetteur de lœagent mobile ne peuvent pas prévoir certaines attaques et ne peuvent donc pas les inclure dans les fonctions d'évaluation et assurer la protection nécessaire [30, 56, 94].

enregistrement authentifiable des plates-formes déjà visitées par un agent [83]. Constituer un historique de lôitinéraire « path history » exige de chaque plate-forme dôy rajouter une entrée signée indiquant son identité et lôidentité de la prochaine plate-forme à visiter. La nouvelle plate-forme se base sur les informations contenues dans lôhistorique de lôitinéraire pour décider de lôexécution ou non de lôagent et des privilèges quôelle doit lui accorder. Afin de prévenir la modification, la signature de la prochaine plate-forme peut déterminer si elle peut se fier aux précédentes plates-formes visitées soit par une révision de la liste des identités fournies ou par une authentification individuelle des signatures de chaque entrée de lôitinéraire pour confirmer lôidentité.

Lôinconvénient de cette technique est quôelle ne prévient pas la plate-forme des comportements malicieux mais exerce un effet d'avertissement puisque lôentrée signée de la plate-forme au niveau de lôitinéraire est non répudiable. En plus, le contrôle de lôitinéraire devient de plus en plus coûteux au fur et à mesure que celui-ci grandit. [25, 56, 76].

II.3 Sécurité des Agents Mobiles

Dans la section précédente, nous avons présenté les différentes attaques que peut mener un agent contre une plate-forme, ainsi que les différentes techniques existantes pour palier à ce problème. Les attaques dôune plate-forme envers un agent sont beaucoup plus graves, car celle-ci a accès à touts les composants de lôagent et peut les modifier [17]. De là, lôobligation de techniques pour détecter, voire empêcher de telles menaces.

II.3.1 Un agent mobile simple et le besoin de protection

Nous pouvons référencer les éléments transportés par un agent mobile et pouvant être cible dôattaque [17, 66, 67] comme suit:

ÅLe Code : ensemble des instructions composant la tâche de lôagent.

ÅLes données statiques : données ne changeant pas durant les déplacements (telles que la signature).

ÅLes données collectées : ensemble des résultats obtenus au cours des déplacements réalisés par lôagent depuis son lancement.

ÅLôétat courant : ensemble de données servant à lôexécution courante de lôagent.

e acheteur dont le rôle est de visiter un certain nombre le prix le plus bas d'un bouquet de fleurs. Nous représentons l'agent par un code source, et un bloc de données (fig.2.1, 2.2). Le code source représente le savoir faire de lôagent, tandis que le bloc de données stocke quelques valeurs initiales ou des valeurs accumulées dans l'itinéraire de l'agent.

Afin de mieux comprendre les conséquences de laisser un agent mobile sans protection, nous allons définir les attaques et les illustrer sur l'agent mobile acheteur.

II.3.1.1 Mascarade

Dans ce contexte, on dira que lôagent mobile est victime dôune mascarade, quand une plate-forme se fait passer pour une autre plate-forme dans le but de tromper lôagent par rapport à sa destination et de lui faire quitter son domaine de sécurité [82, 101, 109]. Dans lôexemple, lôagent mobile peut être cible dôun hôte malicieux qui prétend être « magasin de fleurs ». De cette façon, la plate-forme peut obtenir de lôinformation sensible contenue dans lôagent. Cela fait des dommages à lôagent et à la plate-forme dont on a abusé lôidentité.

II.3.1.2 Déni de service

Lors dôun déni de service, la plate-forme refuse dôeffectuer les services demandés par lôagent [30, 109]. Ainsi, elle peut ignorer les demandes de service, retourner des résultats erronés ou introduire des délais inacceptables pour des tâches critiques. En plus, elle peut nier avoir communiqué avec lôagent mobile.

Dans notre exemple, si une plate-forme ne donne pas lôaccès réseau à lôagent mobile acheteur (Dénier une demande de service), il ne pourra pas se déplacer.

Dôune autre part, *getProviders()* et *getAddress()* sont des appels système. Retourner des résultats incorrects aura des impactes graves sur lôagent.

```
me Host";

float maximumprice = 2000 Da;
good flowers = 10 red roses;
address shoplist[] = empty list;
int shoplistindex = 0;
float bestprice = 200 Da;
address bestshop = empty;
```

Fig 2.1: Agent acheteur (bloque de données)

```
1 public void startAgent(){
2       if (shoplist == null) {
3           shoplist = getTrader().getProvidersOf("BuyFlowers");
4          go(shoplist[1]);
5          break;
6      }
7      if (shoplist[shoplistindex].askprice(flowers) < bestprice){
8           bestprice = shoplist[shoplistindex].askprice(flowers);
9           bestshop = shoplist[shoplistindex];
10     }
11     if (shoplistindex _ (shoplist.length - 1)){
12          buy(bestshop,flowers,wallet);
13         go(home);
14         if (location.getAddress() == home){
15            location.put(wallet);
16        }
17    }
18   go(shoplist[++shoplistindex]);
19 }
```

Fig2.2: Agent acheteur (bloque de code)

II.3.1.3 Altération et manipulation

Un agent se déplace sur plusieurs plates-formes ayants des domaines de sécurité différents. En plus, la plate-forme a accès au code, à lôétat et aux variables de lôagent mobile. En conséquence, il est possible que celle ci modifie le code, lôétat ou les variables [17, 92]: cela constitue une altération de lôagent.

La plate-forme peut aussi modifier la communication de lôagent mobile. Cette attaque serait désastreuse dans le cas de transactions financières.

modifier lôaffectation de la ligne 12 afin que lôagent

II.3.1.4 Vol d'information

Le vol dôinformations est un danger très important, car la plate-forme a accès au code à lôétat et aux variables de lôagent mobile [17, 109]. Il est donc dangereux de faire exécuter des algorithmes propriétaires ou de stocker des informations sensibles (clé privée du propriétaire) dans lôagent. Lôencryptage classique nôest valable ici que pour le code et les données qui ne seront pas utilisés sur cette plate-forme. En outre, même si la plate-forme nôest pas capable dôobtenir directement de lôinformation par lôagent, elle peut en déduire sa destination par les services demandés ou encore à partir des agents avec qui il communique.

Dans lôexemple, la partie exécutable de lôagent mobile englobe la stratégie dôachat qui consiste à acheter suivant un prix donné. Si un site malicieux espionne les lignes de code 8 et 9 ou la variable *bestprice*, il peut modifier sa propre proposition de prix dans le but de gagner lôappel dôoffre de lôagent acheteur.

Le tableau 2.1 résume les attaques présentées ainsi que leurs symptômes.

x		Symptômes
MASCARADE	Mascarade de l'hôte	• Code open source. • Endommagement ou modification de données
DENI DE SERVICE	Déni d'exécution	• Possibilité d'inspecter le flux de contrôle. • Délai d'exécution dépassé.
	Retour de résultats incorrects à l'agent mobile	• Possibilité d'inspecter le flux de contrôle. • Résultas incorrectes.
ALTERATION	Manipulation des données	• Sauvegarde temporaire. • Endommagement ou modification de données.
	Manipulation du flux de contrôle	• Code open source. • Code cassé ou endommagé.
	Manipulation des interactions avec d'autres agents	
	Manipulation du code	• Sauvegarde temporaire. • Modification du code et de l'état de l'agent.
	Exécution incorrecte du code	• Temps d'exécution long. • Code open source.
VOL D'INFORMATION	Espionnage des interactions avec d'autres agents	• Changements du comportement de l'agent. • Résultas erronés.
	Espionnage du code	• Temps d'exécution long. • Sauvegarde temporaire. • Code open source.
	Espionnage de données	• Code open source. • Temps considérable avant la visite du prochain site.
	Espionnage du flux de contrôle	• Détérioration des performances. • Modification de l'agent. • Détermination du prochain site à visiter. • Inspection du flux de contrôle.

Tableau 2.1 Attaques des sites malicieux et leurs symptômes

. ?

de maintenir les caractéristiques principales suivantes [2, 9, 15, 49, 92]

II.3.2.1 La confidentialité

Dans la littérature, la confidentialité semble être la qualité la plus importante dôun système sûr. Assurer la confidentialité des données consiste à faire en sorte que les informations restent secrètes et que seules les personnes autorisées y aient accès. Les utilisateurs du système ont besoin dôavoir la garantie que leurs informations ne risquent pas dôêtre divulguées à des personnes non autorisées.

II.3.2.2 L'intégrité

Outre la confidentialité des informations, lôintégrité évite la corruption, lôaltération et la destruction des données de manière non autorisée. Lôintégrité reste un domaine très large couvrant à la fois les modifications, les moyens de modification et lôaprès-modification.

II.3.2.3 L'authentification

Assurer l'authentification dans le système, consiste à prouver que l'identité des différentes entités doit être vérifiée, et que les informations reçues sont conformes à celles fournies. Il faut ensuite assurer que les deux entités communicantes sont bien ce qu'elles affirment être. De plus, le service d'authentification doit montrer que la connexion ne peut être brouillée par une troisième entité essayant de se faire passer pour un des deux correspondants.

II.3.2.4 La disponibilité

La disponibilité consiste à garantir une présence continue des informations, des ressources et des services (le temps de réponse, la tolérance aux fautes, le contrôle de concurrence, le partage équitable de ressources, etc.). D'une autre façon, c'est prévenir contre les perturbations et les interruptions du fonctionnement d'un système et aussi contre toute utilisation abusive de ses services et ressources.

ormation est transmise entre deux entités, l'émetteur ne doit pas pouvoir nier l'envoi de l'information, et le destinataire l'avoir reçue. Chaque partie possède ainsi la preuve de l'existence de la transaction.

II.3.2.6 Le secret du flux

L'intérêt est d'empêcher tout utilisateur non autorisé d'avoir la possibilité d'analyser les flux des données. Tout accès illégal, même en lecture, à un flux de données permet à l'utilisateur de déduire des informations utiles et qui peuvent, ultérieurement, servir ses intentions malveillantes. La taille des messages échangés, leurs sources et leurs destinations, ainsi que la fréquence des communications entre les utilisateurs, sont des exemples de données à préserver pour prévenir le secret de flux dans le réseau et le rendre plus sûr.

II.4 Les politiques de sécurité

"La politique de sécurité d'un système est l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique" [29].

La politique de sécurité décrit les objectifs de la sécurité du système et identifie les menaces auxquelles le système devra faire face. Il est d'abord utile de faire la distinction entre politique de sécurité et mécanisme de sécurité:

Les politiques de sécurité définissent les autorités et les ressources et spécifient les droits et les règles d'usage.

Les mécanismes (ou approches) de sécurité sont des moyens pour la mise en î uvre d'une politique de sécurité.

Donc, le terme "politique de sécurité" est associé à l'ensemble des propriétés de sécurité que l'on désire appliquer et déployer dans un système ainsi que les dispositifs pour les assurer.

A cet effet, dans le domaine de sécurité des agents mobiles, plusieurs approches de protection des agents mobiles ont vu le jour. Elles tentent de garantir lôaccès de lôagent mobile à des hôtes dans lesquels il peut avoir toute confiance ou de déceler ceux qui sont malveillants.

ches on été proposés, parmi lesquels le classement les divise en deux groupes : approches de sécurité basées sur le matériel et approches de sécurité basées sur le logiciel.

Un autre classement en deux grandes classes: classe d'approches de prévention et classe d'approches de détection.

II.4.1 Approches de protection basées sur la prévention

Le principe de ces approches consiste à éviter les attaques menées par les hôtes contre les agents. Dans la section suivante, nous évoquons quelques unes :

II.4.1.1 Sécurité basée sur le matériel

Lorsqu'il s'agit d'une sécurité basée sur le matériel, il est nécessaire d'avoir un outil matériel. Ce dernier doit être testé par le système avant et pendant l'exécution. Certaines applications réseaux, en particulier Internet, implémentent cette approche pour la sécurisation de leurs logiciels contre le piratage. Il existe plusieurs mécanismes, citons parmi autres:

a- Sécurisation à base d'un environnement d'exécution de confiance

Cette méthode de sécurisation consiste à utiliser un périphérique (coprocesseur) assurant l'exécution de l'agent, ce dernier s'exécute exclusivement à l'intérieur de ce périphérique et ne dialogue avec le site visité qu'à travers une interface sécurisée.

Cette approche a été proposée par Wilhelm [104, 105] et elle est basée sur ce qu'il appel « environnement d'exécution de confiance TPE » (Trusted Processing Envirenment). Le périphérique TPE est fabriqué sous le contrôle d'une autorité de confiance et dispose d'un certificat et d'une politique de sécurité. Dans cette approche les agents sont cryptés par la machine de leur propriétaire et ne seront plus exécutés par le site visité mais par le périphérique TPE. Ainsi, les agents resteront cryptés tout au long de leurs chemins de migration et sur chaque site visité. Quand ils arrivent à leurs sites de destinations ils sont encore cryptés et accèderont au périphérique TPE où ils seront décryptés puis exécutés.

Une telle approche présente une solution très puissante pour sécuriser les agents sur le site d'exécution ainsi qu'à travers le réseau. Toutefois, le périphérique TPE n'est pas facile à fabriquer, ce qui explique son coût élevé. De plus, l'environnement d'exécution de confiance

l'un ordinateur, ce qui réduit l'efficacité d'exécution de
n de plusieurs agents en parallèle reste posé.

b- Sécurisation à base de cartes à puces

Cette solution est proposée pour la première fois par Mana [69]. L'idée consiste à subdiviser le code de l'agent en sections dont certaines seront cryptées par la clé publique [136] d'une carte à puces. Ces sections seront remplacées par des appels procéduraux vers la carte. Sur le site d'exécution, on transmet à la carte, comme arguments, les sections cryptées qui seront décryptées puis exécutées à l'intérieur de la carte (Figure2.3).
La paire de clés est générée à l'intérieur de la carte à puce, la clé publique sera publiée tandis que la clé secrète résidera exclusivement à l'intérieur de la carte et ne sera jamais révélée.

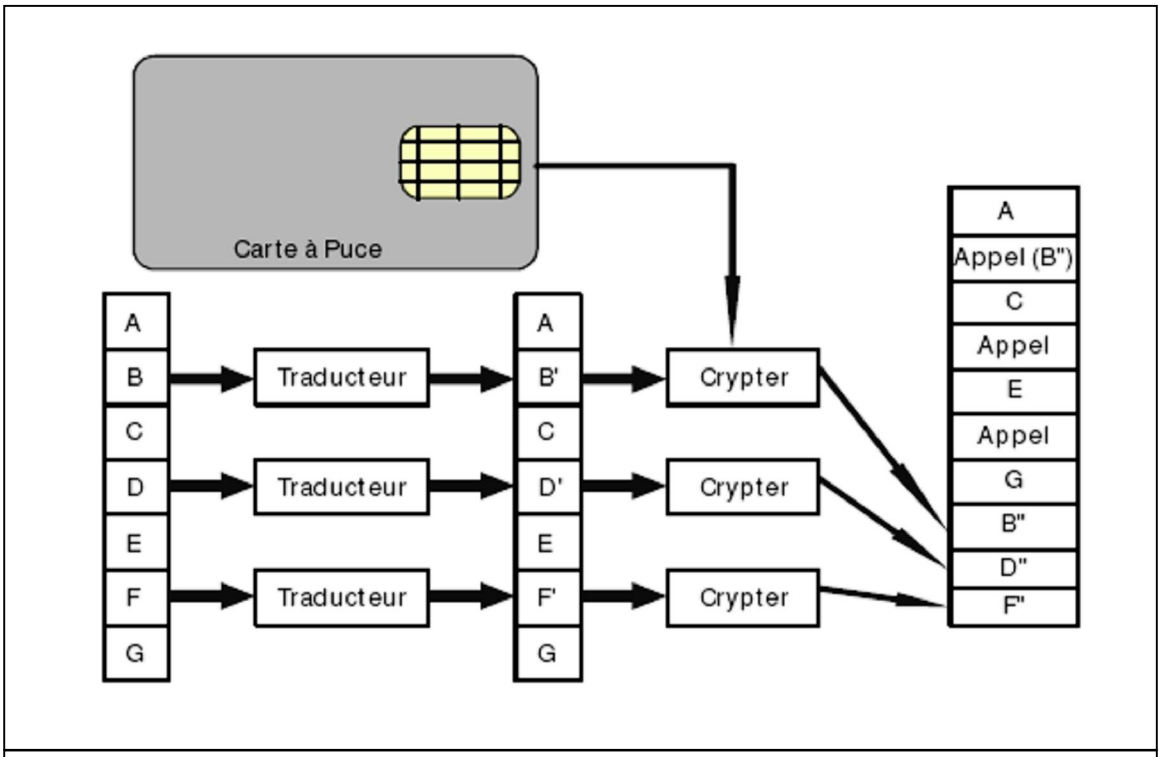


Fig.2.3 Sécurisation à base de cartes à puces

De cette façon, la confidentialité du code est assurée et lôexécution de la totalité de lôagent sur le périphérique de sécurisation est évitée. En cas dôexécution de plusieurs agents sur le site, la carte peut jouer le rôle dôune ressource critique dont lôaccès est géré par les moyens connus.

ôapproche, le code est partiellement caché et le code
se du code clair et la déduction des fonctionnalités du code dissimulé. La déduction des fonctionnalités des sections cachées permet une attaque à la boîte noire [20]. Les sections cachées peuvent être appelées par un code écrit par celui qui attaque afin de réaliser des tâches au nom de lôagent et sur le compte de son propriétaire.
A titre dôexemple, la fonction signature peut être exploitée pour signer un document autre que celui choisit par lôagent.

Les approches basées sur un matériel de confiance sont jugées être parmi les plus puissantes. Cependant, elles présentent un coût trop élevé.

II.4.1.2 Boite noire limitée dans le temps

Appelée aussi technique dôobscurcissement, cette approche était proposée par Fritz Hohl [49] et consiste à produire à partir d'un agent A, un autre agent B analogue au premier dans ses fonctionnalités mais ayant un code difficile à analyser.
Ce code peut être assuré par le fait d'introduire un désordre au niveau du code de l'agent, on peut dire alors que cette approche présente une solution de sécurisation de l'agent par son propre code.

Hohl propose la violation totale des règles nécessaires pour avoir un code lisible. On peut citer par exemple les règles suivantes :

- Ecrire un code modulaire.
- Attribuer des noms significatifs aux variables.
- Utiliser des structures de contrôle qui simplifie le programme.

Ces règles seront respectées au niveau de l'agent source A, et pour passer à l'agent sécurisé B, tout d'abord on crée à partir des variables originaires de nouvelles variables ayant des noms non significatifs et un nombre différent. Comme indiqué dans la figure.2.4, chaque nouvelle variable est composée de fragments de quelques variables originaires.

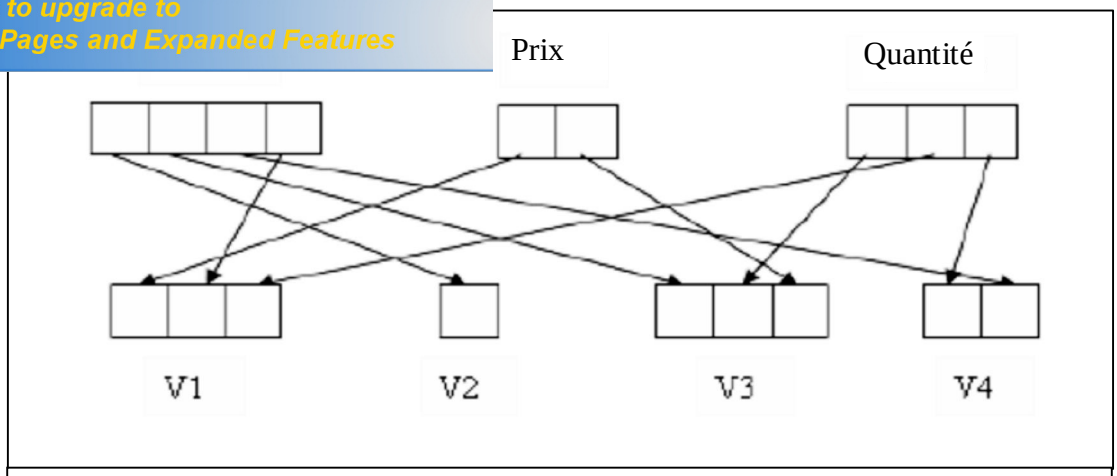


Fig.2.4 Reproduction de variables avec des noms non significatifs

Après l'étape de génération des variables, on procède à la déstructuration du code. Pour cela on élimine les variables locales et on les remplace par des variables globales, on remplace l'appel procédural par le corps des procédures et on utilise la structure de contrôle "GOTO" pour remplacer les autres structures.

Enfin, on pourra insérer des fragments de code mort pour améliorer la protection. Seulement, il faut se méfier des mécanismes de détection de codes morts.
Comme il s'agit d'une sécurisation basée sur le logiciel, ceci nous a permis de réduire le coût élevé des solutions matérielles d'où on a pu retrouver une approche efficace pour sécuriser les agents.

Les limites théoriques des techniques dôobscurcissement ont été étudiées par Barak et al. [5]. Ils ont prouvé quôen général il était impossible de réaliser un obscurcissement complètement sécurisé, et que l'approche souffre du manque de fondement théorique.
De plus, la sécurité de l'agent est temporaire et n'est valide que pour les agents qui transportent des données à courte durée de validité. Autrement, l'agent sera enregistré et analysé lentement afin de déduire les données qu'il transporte.

II.4.1.3 Calcule par fonctions cryptographiques

Cette idée est développée par Sander et Tschudin [92]. L'objectif est analogue à celui de l'approche d'obscurcissement, mais le principe est plus développé, dispose d'un fondement

est pas liée à une durée de vie. Sander et Tschudin

Soit une fonction f matérialisée par un agent A, cette fonction sera cryptée en E(f) qui cache les fonctionnalités et les détails de f. Un programme P(E(f)) sera écrit pour implémenter E(f), ce qui produit un nouvel agent B. L'agent B migre sur un site distant où il sera exécuté sur une donnée x et retourne à son site d'origine qui exécute l'algorithme de décryptage: E (P(E(f)))(x)=$f^{-1}(x)$). Au niveau du site distant, on exécute P(E(f)) (x) qui cache les détails, ce qui assure la sécurité de l'agent.

Voici une illustration du problème :

« Alice a un algorithme qui calcule la fonction f. Bob a une entrée x et veut calculer f(x) pour Alice, mais elle ne veut pas que Bob apprenne quoi que ce soit de f. De plus, Alice et Bob ne doivent pas sœchanger de messages pendant le calcule de f(x) ».

Pour résoudre ce problème, il faut supposer que A puisse crypter *f*, ce qui donne E(*f*). Voici la méthode (fig. 2.5):

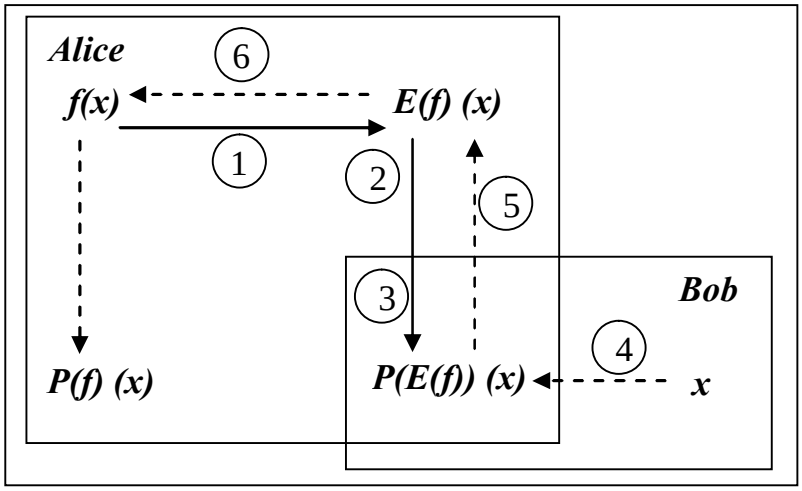


Fig.2.5 Calcule par fonctions cryptographiques

1. Alice crypte *f*.
2. Alice crée un programme P(E(*f*)) qui implémente E(*f*).
3. Alice envoi P(E(*f*)) à Bob.
4. Bob applique P(E(*f*)) à x.
5. Bob envoi P(E(*f*))(x) à Alice.
6. Alice décrypte P(E(*f*))(x) et obtient *f*(x).

ique nôempêche ni la réexécution, ni lôextraction de rak et al. [5] ont démontré que lôobtention dôune justification théorique pour la capacité dôun programme à cacher totalement son information semble imaginaire.

II.4.1.4 Approche de la clé environnementale

Le procédé de la clé environnementale est quelque peu apparenté à la manière dont sont maintenus les mots de passe dans les logiciels d'exploitation modernes (tels que le stockage du hachage d'un mot de passe).

Ce procédé est employé pour déterminer si les tentatives dôidentification sont valides. En effet, ce mécanisme permet à un agent dôexécuter une action spécifique quand une condition environnementale devient vraie [85].

La technique de la génération de la clé environnementale peut être utilisée, par exemple, lorsquôune plate-forme désire communiquer avec une autre plate-forme en lui envoyant un message, et cette dernière ne peut lire ce message que si certaines conditions sont réunies. Ceci peut être réalisé en émettant un agent mobile transportant un message crypté, qui peut contenir des données ou/et du code exécutable. Ni lôagent mobile ne peut prévoir avec précision sa propre exécution, ni la plate-forme qui lôaccueille ne peut deviner la tâche que va exécuter lôagent. Celui-ci attendra au niveau de la plate-forme de réception jusquôà ce quôun certain état environnemental se produise. Dès que la condition environnementale est atteinte, une clé dôactivation est générée afin de déchiffrer le message transporté par lôagent.

Cette technique assure ainsi que ni la plate-forme, ni un observateur extérieur ne peut découvrir le message déclencheur simplement en lisant le code. La condition qui déclenche le mécanisme est généralement cachée soit par une fonction de hachage, soit à travers un mécanise dôencryptage.

Une faiblesse de cette approche est qu'une plate-forme qui contrôle complètement l'agent pourrait simplement modifier l'agent pour imprimer le code exécutable à la réception du déclenchement, au lieu de l'exécuter [56].

Un autre inconvénient est qu'une plate-forme d'agent limite typiquement les possibilités d'un agent pour exécuter le code créé dynamiquement, puisqu'elle le considère en tant quôopération peu sûre.

Cependant, il est possible dôappliquer cette technique en même temps que d'autres mécanismes de protection pour des applications spécifiques sur les plates-formes appropriées :

que de la clé environnementale pour proposer un « agent possède du code encrypté et une fonction qui permet de générer la clé de décryptage en fonction de l'environnement. Si l'environnement n'a pas les conditions pour générer la clé, il est impossible de deviner la fonction de l'agent.

Tschudin [98] a exploité l'idée de la génération de la clé environnementale pour la mort programmée d'un service mobile (le suicide du service mobile) quand on nôen a plus besoin.
Filiol, dans [33], utilise la clé environnementale comme une technique de base pour interdire lôanalyse du code dôun virus, puisque cette approche pousse le propriétaire de lôagent à spécifier un ensemble de contraintes sur lôenvironnement sur lequel lôagent mobile sera exécuté.

Hacini et al dans [47], ont utilisé une clé environnementale pour calculer le degré de confiance dôun hôte pour estimer la confiance que lôagent mobile peut avoir en lôhôte visité.
Le principe est basé sur une interaction dynamique entre lôagent mobile et lôenvironnement visité. Les informations collectées durant cette interaction permettent la génération de la clé environnementale utilisée pour calculer le degré de confiance du site à visiter.

II.4.1.5 Approche du code à la demande

Lôapproche du code à la demande [102] protège lôintégrité de lôagent mobile en utilisant un code de lôagent dynamiquement extensible. Au niveau de cette approche, de nouveaux modules de la fonction de lôagent peuvent être ajoutés et les modules superflus peuvent être supprimés durant lôexécution de lôagent.
Cette approche améliore la confidentialité, réduit le coût de transport et aide à la récupération des agents à la suite dôattaques malicieuses.
Néanmoins, elle exige une communication additionnelle avec lôhôte dôorigine.

II.4.1.6 Approches basées sur l'adaptabilité

L'adaptabilité désigne l'action de s'ajuster et de réagir face aux variations des contraintes de lôenvironnement et des besoins des utilisateurs (voire chapitre 3). Les chercheurs utilisent différentes classifications, la plus utilisée est celle qui la catégorise en adaptation structurelle et adaptation comportementale [39, 40].

Zhi et Zhogwen [110] Présentent un mécanisme dôadaptation dynamique de sécurité où différentes implémentations de sécurités peuvent être remplacées à tour de rôle de façon dynamique. Le but étant lôadaptation des procédures de sécurité telles que les différentes fonctions

Sé) ou asymétriques (RSA, DSA, é) [137] Lôapproche travail dans lequel les agents mobiles peuvent choisir dynamiquement une implémentation de sécurité parmi plusieurs.

La solution de lôadaptation dynamique de sécurité fait que lôagent mobile change lôimplémentation de sécurité selon le changement de lôétat de lôenvironnement dans lequel il sôexécute. Lôadaptation dynamique de la sécurité, selon Zhi et Zhogwen, augmente lôefficacité de lôagent mobile.

S.Hacini et al dans [42, 43, 44, 47] ont introduit l'adaptabilité pour assurer la protection de l'agent mobile. L'architecture proposée est à base dôadaptabilité comportementale et permet à lôagent mobile de changer son comportement suivant la perception des différentes variations de lôenvironnement. Le principe consiste à calculer un degré de confiance suivant lequel lôagent mobile choisit le comportement approprié.

II.4.2 Approches de protection basées sur la détection

Les approches basées sur la détection tentent à déceler les attaques menées contre l'agent mobile par un hôte malicieux.

II.4.2.1 Traces cryptographiques

Cette approche est conçue pour vérifier l'intégrité d'exécution des agents après leur retour. En effet, chaque site visité génère une trace d'exécution de l'agent [99, 100]. Cette trace contient toutes les lignes de code exécutées et toutes les valeurs externes lues par l'agent. Avant la migration de l'agent vers une nouvelle destination, le site calcule une représentation condensée de la trace, la signe et l'accompagne à l'agent mobile lors de sa transmission vers le site suivant.

Après le retour de l'agent, son propriétaire aura une trace de l'exécution de l'agent sur le dernier site et l'adresse du premier site visité (vers lequel il a envoyé l'agent).

Si le propriétaire a un doute envers un site donné, il pourra suivre le chemin de l'agent et avoir la trace d'exécution sur ce site et compare la simulation avec la trace reçue.

Cette approche permet d'assurer la non répudiation et la détection de toute manipulation de l'exécution de l'agent après son retour, mais elle reste détective et n'évite en aucun cas l'utilisation frauduleuse de l'agent. De plus, il faut avoir un doute envers un site particulier pour lancer le mécanisme de trace, sinon l'utilisation systématique de ce mécanisme devient

nitations, telles que la taille de la trace potentiellement
nessages échangés, ou le lancement tardif du processus de vérification.

Des améliorations ont été proposées à cette technique, on cite lôamélioration de Hohl [51] qui propose quôaprès son exécution sur une plate-forme, lôagent doit être émis avec les variables dôétat et les traces précédant et suivant lôexécution. De cette manière, la vérification de la bonne exécution sur la plate-forme précédente peut se faire directement dans la plate-forme suivante et non à la fin du parcours. Par conséquent, la détection d'une attaque se fait juste après l'attaque.
La taille des traces se trouve réduite car on n'échange que les traces d'une exécution d'un hôte et non pas la somme des traces de tous les hôtes précédemment visités. Néanmoins, cette solution nôest pas efficace dans le cas de deux sites hostiles, consécutifs dans lôitinéraire, et collaborent contre lôagent.

Une autre amélioration de la technique de la trace dôexécution a été proposée par Tan et Moreau [95, 96]. Lôidée consiste à assigner la vérification de la trace à une tierce partie de confiance (un serveur de vérification) plutôt quôattendre le propriétaire de lôagent.
Quand un agent mobile migre vers une nouvelle plate-forme, une copie de l'agent est soumise au serveur de vérification correspondant. La plate-forme visitée reçoit l'agent et produit la trace d'exécution associée. Avant la migration de l'agent vers une autre plate-forme, la plate-forme courante envoie la trace au serveur de vérification correspondant. Celui-ci simule l'exécution de l'agent en employant la trace d'exécution correspondante et la copie de l'agent.
Le procédé de simulation est répété pour chaque plate-forme de lôitinéraire par le serveur de vérification correspondant, jusqu'à ce que l'agent revienne de nouveau à sa plate-forme dôorigine. La vérification de la trace dôexécution est forcée et c'est un avantage par rapport à la technique de trace d'exécution originale où le procédé de vérification est déclenché seulement à la suite de résultats suspicieux.

Cependant, cette technique souffre toujours de la limitation de la taille de la trace et de la nécessité de maintenir une taille et un nombre potentiellement grands de notations.
De plus, chaque plate-forme choisit un serveur de vérification et cela pourrait encourager et faciliter une collaboration malveillante possible entre une plate-forme et le serveur.

artiels

Cette technique est utilisée pour encapsuler les résultats de lôexécution dôun agent au niveau de chacune des plates-formes [108]. Lôinformation encapsulée est utilisée plus tard pour vérifier que lôagent nôa pas été attaqué par une plate-forme malicieuse.
Le processus de vérification peut être réalisé lorsque lôagent retourne vers sa plate-forme dôorigine ou bien à des points intermédiaires de son itinéraire.

Afin dôadresser certaines exigences de sécurité telles que lôintégrité, la responsabilité et lôintimité de lôagent, lôencapsulation des résultats partiels utilise différentes primitives cryptographiques telles que lôencryptage, la signature digitale, lôauthentification du code et les fonctions de hachage.

II.4.2.3 Approches basées sur les nœuds de confiance

L'idée est basée sur lôinstallation dôun ensemble dôhôtes de confiance. Ceci peut être fait en encryptant l'agent mobile durant son cheminement entre les hôtes, et en authentifiant lôhôte avant que l'agent mobile ne lui soit transmis. Créer un environnement de confiance dans lequel un agent mobile erre librement sans être menacé par un hôte malveillant, permet de se protéger contre plusieurs types d'attaques [17].

Cependant, cette méthode va à lôencontre de la notion d'agent mobile. Si un agent mobile a un itinéraire préétabli, l'avantage de la vaste quantité de ressources disponibles sur l'Internet, ainsi que la bonne concurrence peut être perdu ou sévèrement engorgé.
Dôun autre côté, en introduisant des nî uds de confiance au sein dôune infrastructure vers laquelle les agents mobiles peuvent, si nécessaire, migrer, il est possible dôéviter que des informations sensibles ne soient émises vers des hôtes non fiables. De plus, il est aisé dôobtenir la trace de certains mauvais comportements de ces hôtes. Lôhôte dôorigine est souvent supposé être un hôte de confiance.

Cette approche ne semble pas avoir été totalement explorée. Elle peut être potentiellement très valable dans des réseaux composés de nî uds mobiles et fixes, offrant aux utilisateurs la possibilité de distribuer les agents mobiles au niveau du réseau fixe en se basant sur les noeuds de confiance afin de traiter lôinformation sensible. Lôagent mobile peut aussi être conçu de façon à ne sôexécuter que sur un hôte jugé de confiance.

plusieurs agents

Dans les approches suivantes, lôagent mobile est protégé par un ou plusieurs agents coopérants :

a. Agents coopérants

Lôidée a été proposée par Schneider et al. [93] et Roth et al [87]. La protection est assurée en faisant partager l'information par plusieurs agents analogues dits clones. La tâche demandée par l'utilisateur est réalisée par plusieurs agents qui coopèrent plutôt que par un seul agent. L'objectif principal est que l'information partagée reste protégée même si plusieurs sites d'exécution collaborent.

Dans un exemple explicatif, Roth [87] définit deux sous-groupes indépendants de sites que l'agent visitera. Ensuite, deux agents seront créés et envoyés chacun à l'un de ces sous groupes. Si un agent retrouve une information pertinente, il l'envoie vers son clone qui la vérifie. Pour avoir plus d'efficacité, l'un des deux agents s'exécute obligatoirement sur un site de confiance (fig. 2.6).

Roth [86, 87] fait plusieurs hypothèses:
La première est qu'aucun hôte sur l'itinéraire d'un agent ne coopère avec un hôte de l'itinéraire de l'autre agent. Il présume aussi l'existence d'un canal de communications. Et enfin, il suppose que les identités sont données sans altération à l'agent.

Cette méthode est intéressante, toutefois elle est entachée dôun certain nombre d'inconvénients [56]:

- Mettre en place le canal de communication à chaque migration est une opération coûteuse.
- Si un agent est tué, le protocole n'est pas capable de savoir lequel des deux hôtes est responsable.
- Si une plate-forme déclare avoir reçu un agent d'une autre plate-forme et que cela n'est pas vrai, le protocole ne peut décider qui est coupable.
- Si un hôte reçoit deux agents qui viennent du même hôte, il est possible d'interchanger les agents qui enregistrent 1'itinéraire des deux autres.
- De plus, la condition de la non coopération dôhôtes malicieux est difficile à prouver ou à vérifier. Par ailleurs, la décomposition des tâches critiques peut ne pas être facile à automatiser, ce qui limite considérablement le déploiement de cette approche.
- Enfin, un agent peut être modifié par un hôte malveillant de façon à ce que lôagent coopérant ne puisse le détecter.

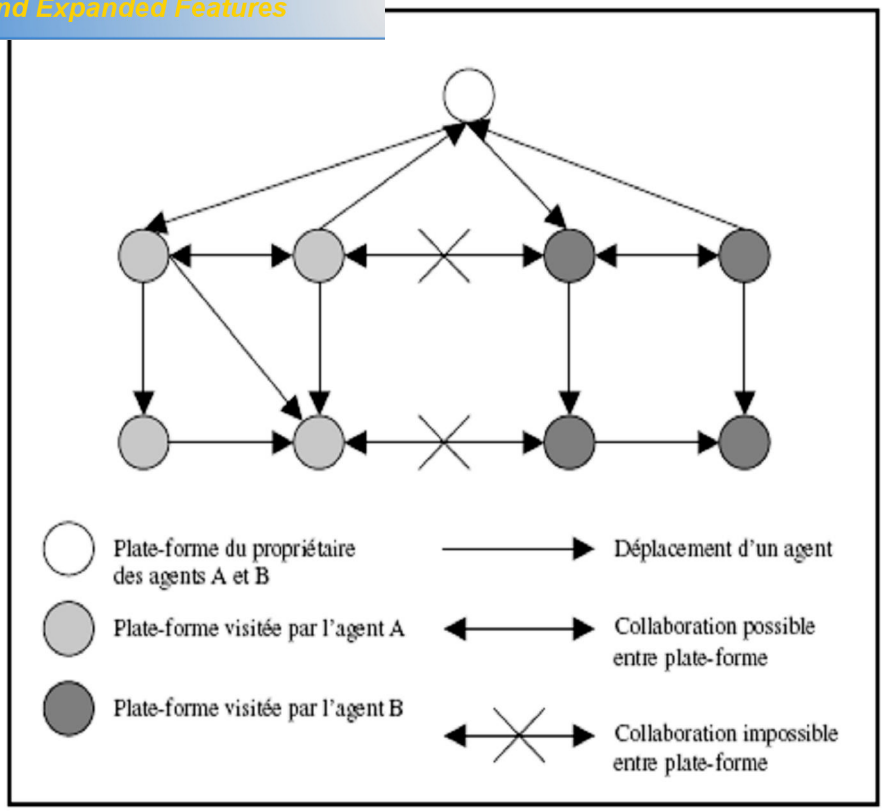


Fig.2.6 Hypothèses de Roth sur les collaborations entre plates-formes

Plusieurs autres approches sont basées sur un ou plusieurs agents en plus de lâagent mobile :

El Rhazi et S.Pierre [28] ont proposé un protocole de sécurité basé sur lâutilisation dâun agent sédentaire dont le rôle est de retenir la partie critique du code de lâagent mobile (la partie qui exprime ses objectifs, son savoir faire). Après son exécution sur lâhôte, lâagent sédentaire le compare avec la copie retenue pour détecter dâéventuelles modifications. Ensuite, un rapport contenant la liste de tous les hôtes malveillants est envoyé à lâhôte dâorigine.

Même si cette approche peut détecter plusieurs types dâattaques, elle n'y peut rien faire, puisque lâagent coopérant ne fait quâenvoyer la liste des sites malveillants au site dâorigine.

Ouardani et S.Pierre [77] ont proposé une approche considérée comme amélioration de [28], basée sur deux agents sédentaires en plus de lâagent mobile. Leur rôle est de suivre lâagent mobile pas à pas en vérifiant son itinéraire et calculer le temps de son exécution afin de détecter différents types dâattaques.

8] utilisent des mécanismes avec un TTP (Trusted Tiers s ce cas cêest un agent coopérants pour enregistrer les informations de lôitinéraire de lôagent mobile directement ou indirectement.
Le problème majeur de cette technique est quôon a besoin dôau moins un TTP, en plus lôagent mobile doit communiquer avec lui sans cesse, et si ce TTP est en panne ou lui-même attaqué lôagent mobile se trouve sans aucune protection. Enfin, il n'est pas évident de trouver un TTP dans un réseau aussi ouvert.

b. Copies d'agent

Dans cette approche, le secret dôune transaction est distribué entre n agents mobiles dupliqués. Ces derniers sont émis vers différents hôtes [9]. Si la tâche à effectuer par lôagent nêest pas affectée par le nombre de fois quêelle est effectuée, plusieurs copies de lôagent peuvent être lancées dans le réseau. Par exemple, la recherche du meilleur prix pour un article est faisable, mais le paiement du produit ne lêest pas. De cette façon, même si quelques copies sont corrompues, les autres vont effectuer la tâche correctement. Pour vérifier quelles copies sont corrompues et lesquelles ne le sont pas, des schémas de vote sont utilisés.
Cette technique de tolérance aux fautes fonctionne avec la supposition que la majorité des plates-formes de migration ne sont pas hostiles.

II.4.2.5 Evaluation d'état

Cette approche est développée par Farmer et al. [30, 31]. Ils définissent un mécanisme permettant à un agent d'évaluer les privilèges dont il dispose sur un site particulier. Ce qui permet au propriétaire de l'agent de limiter les actions que l'agent peut effectuer. L'approche repose sur une fonction protégée qui permet l'évaluation de l'état de l'agent sur chaque site visité. La fonction sera exécutée une fois que l'agent arrive sur un site donné et permettra de vérifier la stabilité de son état. L'existence de telle fonction protège l'agent en détectant les manipulations de son état. La fonction repose sur un calcul complexe à partir d'un ensemble de variables d'état.

Une amélioration a été proposée par Jansen [59] qui consiste à séparer la structure de données, définissant le comportement de l'agent, de son propre code. Cette structure sera définie dans un certificat conforme à la norme X509 [54]. Dans le certificat, on définit les droits et les responsabilités d'un agent sur un site donné.
L'approche protège l'agent contre une utilisation illicite en fournissant une preuve des intentions réelles de son propriétaire mais n'offre aucune protection des données que l'agent transporte.

Les approches à base de fonctions de validation ont été proposées par Blum [19] afin de vérifier la fiabilité d'un code. Ces dernières peuvent être appliquées dans le but de vérifier l'intégrité d'exécution d'un agent mobile sur un site distant. L'approche se base sur la propriété de réductibilité de la fonction qu'implémente l'agent.

II.5 Conclusion

À l'opposé de la protection des hôtes contre les attaques des agents mobiles malicieux, la protection des agents mobiles contre les menaces des sites malveillants est plus complexe, ne dispose pas de solutions sûres, et reste encore aujourd'hui un champ de recherche ouvert.

Nous avons présenté dans ce chapitre les attaques qui menacent un agent mobile sans protection sure. Ensuite, nous avons présenté les approches de protections proposées pour palier à ce problème.

Il existe en effet deux grandes classes d'approches pour la protection: classe d'approches basées sur la prévention, ou les attaques son évitées, et classes d'approches basées sur la détection d'attaques.

Une comparaison entre ces approches (Table 2.2) est établie en s'appuyant sur des critères relatifs à la robustesse de la sécurité des agents et de ceux liés au coût de cette sécurité.

Comme critères relatifs à la sécurité, nous avons examiné les apports de chaque approche en termes de confidentialité d'exécution du code mobile sur chaque site visité ainsi que sa capacité à assurer l'intégrité de l'exécution. Cette intégrité est-elle garantie tout le long de l'itinéraire de l'agent mobile ou est-elle vérifiée seulement après son retour ? Les coûts de la solution de sécurité sont quant à eux examinés selon la taille du code engendrée, le temps d'exécution généré ainsi que les communications additionnelles à travers le réseau en raison des vulnérabilités que ces dernières peuvent impliquer.

En s'appuyant sur l'étude présentée dans ce chapitre, nous présenterons notre contribution pour la protection des agents mobiles dans le prochain chapitre.

		...té	Intégrité d'exécution	Taille du code	Temps d'exécution	Communication additionnelle
Approches de Prévention	**environnement d'exécution de confiance et Cartes à puce**	Forte	Forte	Plus d'instructions (appel de la carte)	Faible (temps d'accès à la carte)	Aucune
	Boite noire limitée dans le temps (obscurcissement)	Forte (pour une courte durée de vie de l'agent)	Forte (pour une courte durée de vie de l'agent)	Acceptable (ajout de code mort)	Acceptable	Aucune
	Calcule par fonctions cryptographiques	Forte	Forte	Acceptable	Acceptable	L'agent retourne à son site d'origine après chaque visite
	la clé environnementale	Forte	Non assurée	Acceptable	Acceptable	Aucune
	Code à la demande	Améliorée	Forte	Non modifié	Non modifié	Nombre de communications additionnelles restreint
Approches de Détection	**Traces cryptographiques**	Non assurée	Vérifiable	Augmentée par la taille de la trace associée	Acceptable	Aucune
	Encapsulation des résultats partiels	Forte	Assurée	Augmentée par les résultats partiels	Acceptable	Aucune
	Noeuds de confiance	Forte	Non assurée	Non modifié	Non modifié	Aucune
	Agents coopérants	Non assurée	Forte	Acceptable	Fortement ralenti	Trop de communications additionnelles
	Copies d'agent	Forte	Non assurée	Non modifié	Non modifié	Aucune
	Approches basées sur l'adaptabilité	Forte	Vérifiable	Non modifié	Non modifié	Aucune
	Evaluation d'état	Non assurée	Partiellement assurée	Acceptable	Acceptable	Aucune
	Fonction de validation	Non assurée	Vérifiable	Acceptable	Acceptable	Aucune

Tab.2.2 Tableau comparatif des différentes approches de protection

3EA CONTRIBUTION

Agent mobile adaptable et Agent Eclaireur

Le problème de sécurité représente le talon d'Achille des agents mobiles. Il constitue un vrai frein à l'utilisation réelle de cette technologie. Nous présentons dans ce chapitre notre contribution pour protéger l'agent mobile contre toute manipulation fraudeuse de la part des sites malveillants.

Le principe consiste à envoyer d'abord un agent prototype, dit éclaireur, vers l'hôte de destination. L'agent éclaireur est une copie restreinte de l'agent mobile avec du code ne traduisant aucun savoir, et des données non importantes.

A son retour, l'éclaireur est analysé par l'agent mobile afin de détecter d'éventuelles attaques. Si ces attaques sont graves, l'agent mobile refait une copie et change de destination. Si les attaques sont évitables, l'agent mobile choisit un plan d'adaptation puis migre. Si aucune attaque n'est détectée, l'agent mobile migre avec confiance.

Nous avons utilisés principalement les concepts suivants : lôadaptabilité, le calcule de la clé environnementale, la signature numérique et lôencryptage. Dans ce qui suit, nous présentons un aperçu de chacune de ces notions.

III.1.1 Adaptabilité

III.1.1.1 Définition

La complexité des nouveaux systèmes d'information et des applications émergentes récentes s'est trouvée considérablement accrue du fait de leur distribution, la grande quantité d'informations qu'ils manipulent, leur aspect coopératif, et leur ouverture. Ces systèmes d'information et ces applications ont les caractéristiques d'un bon domaine d'application des systèmes multi-agents.

Cependant, les connaissances nécessaires à la gestion et au contrôle des systèmes complexes sont très nombreuses, et il est long et difficile de toutes les exprimer en les donnant directement au système. La modélisation de ces systèmes complexes nécessite des agents adaptatifs [10, 39, 68, 84].

Dans la théorie, les systèmes multi-agents sont considérés comme adaptatifs, seul au fait de leur autonomie de prise de décision, hors que cette autonomie est considérée comme une faible adaptabilité.

En réalité, l'adaptabilité désigne l'action de s'ajuster et de réagir face aux variations des contraintes de lôenvironnement et des besoins des utilisateurs [39, 68].

III.1.1.2 Nécessité d'adaptation pour les agents

La mobilité dôagent permet de rapprocher client et serveur, et en conséquence de réduire le nombre et le volume des interactions distantes (en les remplaçants par des interactions locales), de spécialiser des serveurs distants ou de déporter la charge de calcul dôun site à un autre. Une application construite à base dôagents mobiles peut se redéployer dynamiquement suivant un plan pré-établi ou en réaction à une situation particulière, afin par exemple, dôaméliorer la performance ou de satisfaire la tolérance aux pannes, de réduire le trafic sur le réseau, ou de suivre un composant matériel mobile.

Ainsi, la mobilité est un mécanisme important dôadaptation, et les agents mobiles sont un outil à part entière pour lôadaptation des systèmes. Toutefois, lôadaptation au niveau de lôapplication nôest pas suffisante. Nous discuterons les besoins dôadaptation statique des

stème en dehors de leur exécution), puis des besoins cution des agents mobiles et particulièrement lors de leurs déplacements [45].

- Adaptation Statique

Les besoins dôadaptation statique se retrouvent à toutes les échelles du logiciel. En fournissant une architecture souple et capable dôévolution, la maintenance est facilitée et par conséquence, le coût total est réduit.

Ainsi, les agents doivent pouvoir être configurés à froid, cô est-à-dire avant leur exécution, et offrir des mécanismes dôévolution de leur architecture [64].

- Adaptation Dynamique

Prenons lôexemple (fig. 3.1) dôun agent mobile créé sur un ordinateur relié à un réseau local filaire isolé de lôextérieur par des protections adéquates (ex. pare-feu) [40]. Il se déplace ensuite vers un environnement non fiable du point de vue de la sécurité des communications et avec des pertes potentielles du signal réseau (typiquement un portable dans un environnement public relié au réseau par liaison sans fil de type WiFi).

Cet agent peut utiliser à lôorigine des protocoles de communications ayants des propriétés non fonctionnelles de faible sûreté, faible sécurité et forte performance.

Après la mobilité, il serait judicieux, grâce à des mécanismes dôadaptation dynamique, dôutiliser des protocoles dont la sémantique est identique, mais ayant des propriétés non fonctionnelles de forte sûreté et de forte sécurité, éventuellement au détriment de la performance.

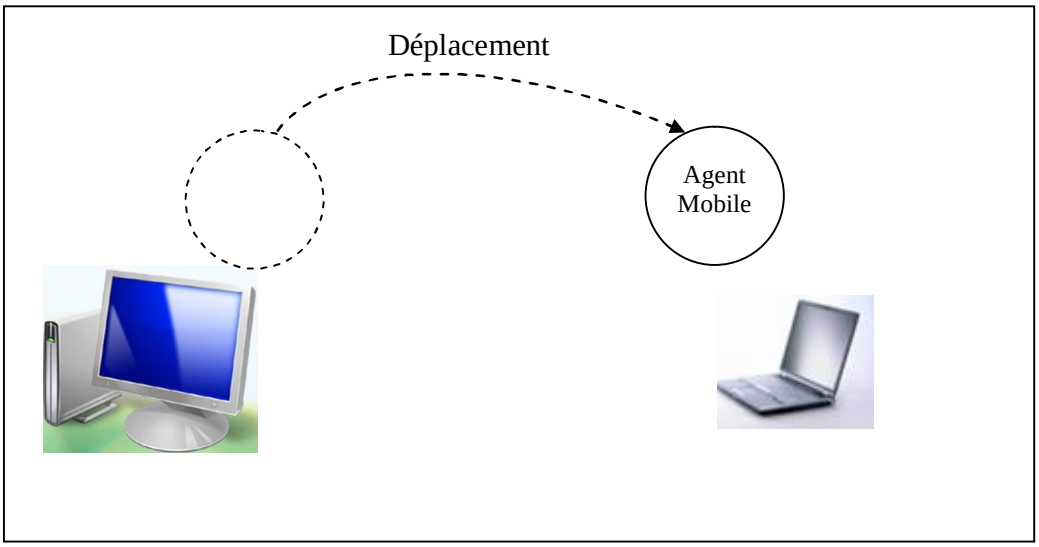


Fig.3.1 Besoin d'adaptation dynamique pour les agents mobiles

mportementale

Lôadaptation structurelle permet d'adapter la structure de l'agent à l'évolution de son environnement. Lôarchitecture dôagent mobile dynamiquement adaptable, à base de microcomposants remplaçables et spécialisables présentée par Leriche et al. [64] est un exemple de ce type dôadaptation. En effet, le concept "composant" est bien adapté pour la conception des systèmes ouverts, complexes et évolutifs tels que les agents mobiles. Il a les avantages dôaugmenter la productivité et baisser les coûts de réalisation des applications qui sôapprêtent à lôassemblage de composants.

Le concept dôadaptabilité est utilisé dans notre cas pour la prise en compte des attaques menées par lôhôte malicieux. Nous proposons d'utiliser lôadaptabilité structurelle, où on effectue des remplacements de composants appropriés à la situation dôattaque.

- Lôadaptation comportementale

Lôadaptation comportementale permet d'adapter le processus de décision de l'agent à l'évolution de son environnement.

Pour protéger les agents mobiles, Hacini et al dans [42, 44, 46] utilisent ce deuxième type d'adaptation, où lôagent mobile change son comportement suivant la perception des différentes variations de lôenvironnement.

III.1.2 Clé environnementale

Le principe, tel proposé dans [85] est basé sur une interaction dynamique entre lôagent mobile et lôenvironnement visité. Les informations collectées durant cette interaction permettent la génération dôune clé, dite clé environnementale, utilisée pour calculer le degré de confiance du site visité.

Dans notre approche, la clé environnementale est calculée dans le but dôidentifier lôhôte visité. En effet, lôagent éclaireur cherche une condition, qui est dans ce cas lôidentité de lôhôte. Une fois cette condition est vérifiée, lôagent éclaireur procède à son exécution.

III.1.3 Signature Numérique

La signature numérique est un mécanisme permettant d'authentifier l'auteur d'un document électronique et de garantir son intégrité, par analogie avec la signature manuscrite d'un

signature numérique doit présenter les propriétés

- Il doit permettre au lecteur d'un document d'identifier la personne ou l'organisme qui a mis sa signature.
- Il doit garantir que le document n'a pas été altéré entre l'instant où l'auteur l'a signé et le moment où le lecteur le consulte.

Pour cela, les conditions suivantes doivent être réunies dans une signature numérique:

- Authentique : L'identité du signataire doit pouvoir être retrouvée de manière certaine.
- Infalsifiable : La signature ne peut pas être falsifiée. Quelqu'un ne peut se faire passer pour un autre.
- Non réutilisable: La signature n'est pas réutilisable. Elle fait partie du document signé et ne peut être déplacée sur un autre document.
- Inaltérable : Un document signé est inaltérable. Une fois qu'il est signé, on ne peut plus le modifier.
- Irrévocable : La personne qui a signé ne peut le nier.

Nous avons utilisé la signature numérique pour permettre d'identifier l'agent mobile et son éclaireur au sein des sites visités, qui doivent eux même avoir une signature numérique pour s'identifier.

III.1.4 Le Chiffrement

Le chiffrement, parfois appelé à tort cryptage, est en cryptographie le procédé grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (dé) chiffrement.

Dans notre travail, le chiffrement permet d'obtenir l'anonymat souhaité pour certaines informations

III.2.1 Origine

Lôidée est inspirée des Explorateurs Militaires. Leur mission principale est dôexplorer le territoire adversaire, afin dôélaborer un rapport sur tous les dangers rencontrés, puis le remettre au chef de troupe.

Si le chef de troupe décide dôaller, il doit choisir une politique pour sôadapter à la nature de lôenvironnement : il demande du renfort, décide dôaller par avion, par bateau ou par route. Le chef de troupe peut ne pas y aller si les explorateurs ne reviennent pas. Notons que ces explorateurs nôont aucune idée sur la mission principale de leur troupe, ils ne font quôéclairer le chemin. Même si par malheur ils sont capturés, ils nôont aucune information importante.

III.2.2 Principe

Le principe consiste donc à envoyer dôabord un agent prototype, dit Eclaireur, vers lôhôte de destination. L'éclaireur est une copie restreinte de lôagent mobile avec du code et des données non critiques. A son retour, il est analysé par lôagent mobile afin de détecter dôéventuelles attaques.

Si ces attaques sont graves (tel que suppression du code ou de données), lôagent mobile refait une copie et change de destination.

Si les attaques sont évitables, lôagent mobile choisit un plan dôadaptation, sôadapte, puis migre.

Si aucune attaque nôest détectée, lôagent mobile migre avec confiance. Donc, au lieu dôexposer directement lôagent mobile aux différentes attaques des sites malicieux, on envoi dôabord un éclaireur pour explorer lôenvironnement. Cela permet de détecter les attaques dés leurs arrivées et ainsi minimiser les dégâts sur lôagent mobile.

La technique proposée permettra de détecter et éviter plusieurs types dôattaques, tel que le déni de service, lôaltération, la répudiation et le kidnapping.

III.2.3 Architecture proposée

Dans cette section nous présenterons lôarchitecture proposée. Nous utilisons deux entités différentes: lôagent mobile (AM) et lôagent éclaireur (AE). Dans ce qui suit, nous détaillons les constituants de chacun (fig.3.2).

alement dôune mémoire, du module manager et dôune interface. Lôagent mobile englobe tout ce qui est important du code et des données critiques (statiques et dynamiques).

1) La mémoire

La mémoire de lôAM contient le code, les données statiques et dynamiques, une copie de l'éclaireur, une bibliothèque dôévénements et une bibliothèque de composants.

- ***Le Code***

Le code exprime le savoir faire de lôagent mobile, et il est constitué de plusieurs composants remplaçables.

- ***Les données statiques***

Ce sont les données qui ne changent pas, tel que lôidentité du créateur, sa signature numérique et les données utilisées pendant lôexécution de lôagent. Les clés de décryptage, pour décrypter le rapport de l'éclaireur, les clés de cryptage des données, des résultats partiels et de lôitinéraire de lôAM, sont aussi considérées comme données statiques.

- ***Les données dynamiques***

Les données dynamiques évoluent à chaque migration. On y trouve la liste des sites malveillants (Black List Hosts, BLH), des sites jugés de confiance (Trust Hosts List, THL), et des sites jugés douteux (Untrusted Hosts List, UHL). En plus, lôitinéraire, les données collectées après chaque migration et les résultats partiels signés par le site et encryptés par lôAM, sont toutes des données dynamiques. Notons que la clé de décryptage est uniquement chez lôhôte dôorigine pour garantir la confidentialité des données.

- ***Copie de l'éclaireur***

LôAM sauvegarde une copie de lôAE (code et données) avant de lôenvoyer vers un site. Le but est de faire la comparaison après le retour pour détecter des modifications du code et des données.

tient un ensemble dôévénements prédéfinis, divisé en trois classes :

-*Classe d'Evénements Permis.* Cette classe contient toutes les actions permises, que lôagent peut faire sans aucun problème.

-*Classe d'Evénements Interdits.* Elle est constituée dôactions non permises pouvant violer les critères de sécurité de l'éclaireur.

-*Classe d'Evénements Douteux.* Cette classe comporte un ensemble dôactions douteuses que lôagent mobile nôest pas sûr quôelles soient de confiance (Ex. Action copier coller).

- ***Bibliothèque de composants***

La bibliothèque de composants est une base de données qui contient des composants remplaçables que lôAM peut utiliser durant son adaptation. Ces composants sont des morceaux de code sélectionnés par lôadaptateur suivant le rapport de lôAE.

2) Interface

Lôinterface entre lôAM et lôAE reçoit les données, le code et le rapport de l'éclaireur pour les envoyer à l'analyse.

3) Manager

Le manager comprend trois modules essentiels:

- ***Coordinateur***

Il est responsable de changer la structure de lôAM afin de lôadapter en cas dôattaques. Il reçoit les composants désignés par lôadaptateur et les place au niveau du code. Il est aussi responsable de changer la destination de lôAE, refaire une copie, ajouter les sites dans la liste noire, liste de doute ou dans la liste de confiance.

- ***Analyseur***

Le rôle de lôanalyseur côest dôanalyser lôAE: code, données et rapport. Pour analyser le code et les données, lôAM les compare avec la copie sauvegardée. Le code et les données statiques ne doivent en aucun cas être modifiés. Les données dynamiques, ou les résultats partiels, doivent être signés par lôhôte visité, sinon celui-ci est jugé malicieux. Quant au

ons produites sur le site, il est analysé en comparant ue dôévénements afin de les classer : actions permises, douteuses ou interdites.

Le but de lôanalyse est de permettre à lôAM de détecter si son éclaireur a été attaqué ou non. Il informe alors lôadaptateur du type dôattaque pour quôil puisse choisir un plan, constitué de composants, à partir de la bibliothèque de composants.

Le coordinateur est chargé de les remplacer au niveau du code pour adapter lôAM à la situation dôattaque.

Notons que lôadaptateur est constitué de règles de la forme :

If Action(i) then Choose ((Composant(i1), Composant(i2),...).

- ***Adaptateur***

Nous nous sommes basés dans cette partie sur les travaux de Hacini et al [47] et Leriche et al [65]. Le rôle de lôadaptateur est de faire adapter lôAM suivant le rapport dôévénements de son éclaireur. Il sélectionne les composants appropriés et les envois au manager qui va changer la structure du code.

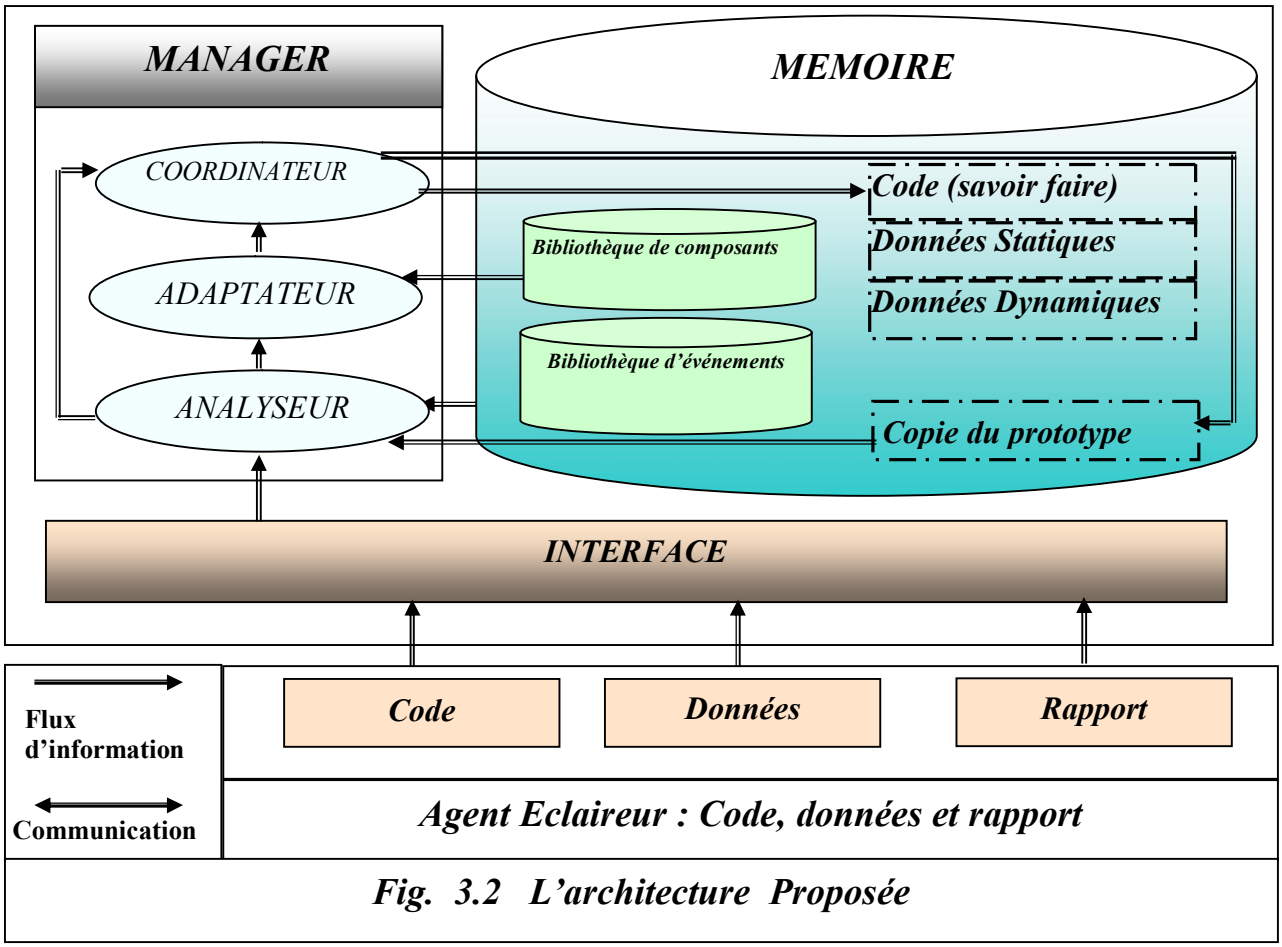


Fig. 3.2 L'architecture Proposée

le, les données et le rapport dôévénements. Le code est une copie de la partie non fonctionnelle du code de lôAM. Les données sont une copie des données non critiques de lôAM (Par exemple, lôAE ne contient pas le numéro de compte dôun client). Le rapport contient toutes les actions réellement produites sur le site visité. Ces actions sont cryptées par une clé connue uniquement par lôAM, dans le but dôéviter quôun hôte malicieux effectue des actions interdites, ensuite essayer de les supprimer du rapport dôévénements.

III.2.4 Etapes du protocole

Les étapes du protocole sont résumées ci-dessous :

1. Au départ, lôagent mobile est sur le site dôorigine H0. Avant de migrer vers un site H1, il crée l'éclaireur et lôenvoi en premier. Lôagent mobile garde une copie de l'éclaireur et calcule le temps dôexécution.

2. Lôagent éclaireur, en arrivant sur la plate forme H1, commence par calculer la clé dôenvironnement. Il réalise certaines actions prédéfinies sur des informations récoltées de H1. Le but de cette clé est de pouvoir identifier lôhôte avant de commencer lôexécution. Si cette clé nôest pas valide, lôagent éclaireur retourne vers son agent mobile qui va lui changer de destination. Si par contre la clé est valide, lôagent éclaireur commence son exécution.

3. Lôagent mobile analyse l'éclaireur. Nous pouvons peut envisager les situations suivantes :

3.1 Lôagent mobile calcule et compare le temps dôexécution avec le temps estimé, si celui-ci est dépassé il conclu que son éclaireur est soit « tué » soit sa destination a été modifiée donc « perdu ». Il met lôhôte responsable dans la liste noire et décide de ne pas y aller. Il doit refaire un autre éclaireur puis lôenvoi vers une autre destination.

3.2 Si lôagent éclaireur revient, lôagent mobile le compare avec la copie quôil a : sôil trouve le code ou les données statiques modifiés, il met lôhôte responsable dans la liste noire, décide de ne pas y aller, refait une autre copie de l'éclaireur et lôenvoi vers une autre destination.

lu code et des données statiques, lôagent mobile vérifie de l'éclaireur. Cette signature permet dôéviter la non répudiation. En effet, le fait de trouver des résultats non signés signifie que lôhôte pourra nier la réception de l'éclaireur. Lôagent mobile met donc lôhôte responsable dans sa liste noire, réinitialise lôAE, et lôenvoi vers une autre destination.

3.4 Sôil nôy a aucune modification au niveau du code et des données statiques, et tous les résultats partiels sont signés, alors lôagent mobile compare les actions du rapport de l'éclaireur, avec la bibliothèque des événements :

a) Sôil trouve des actions interdites, il met lôhôte dans la liste noire (BLH) puis change de destination.

b) Sôil trouve des actions douteuses (tel que copie/coller), il met lôhôte dans la liste de sites douteux (UHL). Dans ce cas, lôagent mobile doit sôadapter en choisissant le composant le plus approprié.

c) Si toutes les actions du rapport sont permises, alors lôagent mobile enregistre lôhôte H1 dans sa liste des sites de confiance (THL), lôajoute et lôencrypte dans son itinéraire puis effectue la migration.

4. À la fin de sa mission, lôagent mobile retourne vers son hôte dôorigine avec toutes les données collectées, ainsi que les listes des sites malicieux, douteux, et de confiance. Ces listes sont utilisées pour les futures migrations.

5. Avant de continuer son chemin vers un hôte H2, lôagent mobile vérifie que ce site nôest pas dans sa liste noire, il réinitialise son agent éclaireur et refait les mêmes étapes.

La figure 3.3 illustre le diagramme de protocole pour représenter les différentes interactions entre les agents du système, en utilisant le AUML (Agent Unified Modeling Language) [7, 8].

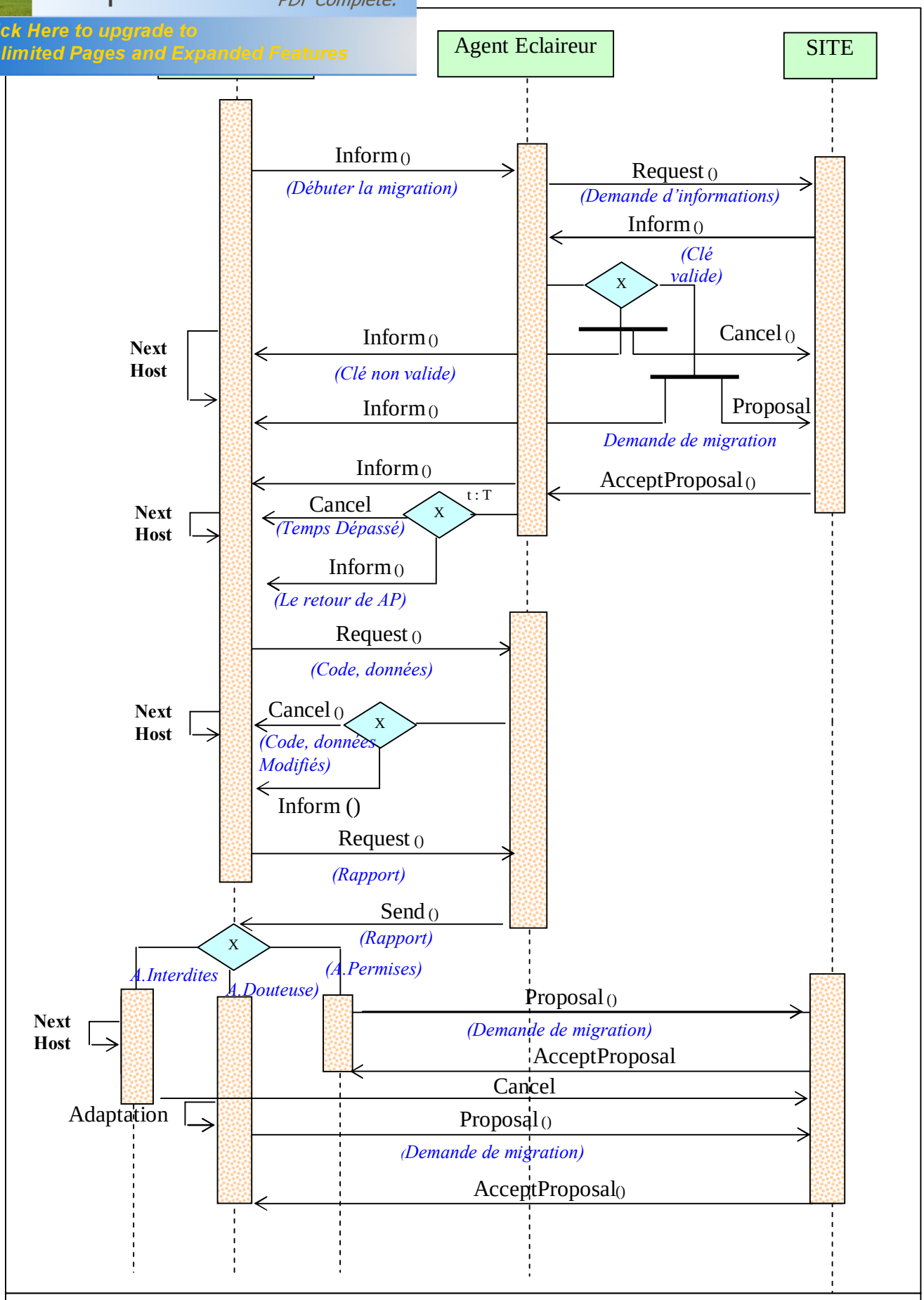


Fig. 3.3 Diagramme de Protocole en AUML

...nances

T. Sander et al. [92], Karjoth et al. [62], Yao et al. [106] et Guan et al. [38], définissent les points suivants comme propriétés de sécurité. Nous allons les utiliser pour évaluer notre proposition.

III.3.1. La confidentialité

Dans la technique proposée, lôanalyse du rapport de lôagent éclaireur permet de savoir si son code a été inspecté ou non. En plus, le calcule de la clé environnementale permet de garantir la confidentialité, car seuls les hôtes autorisés peuvent y accéder dôabord à lôagent éclaireur, puis à lôagent mobile.

III.3.2 L'intégrité

Protéger lôintégrité dôun agent mobile exige la détection de la modification de son code et de son état dû à une mauvaise exécution par un hôte malicieux. Cela est vérifié au moment de lôanalyse de lôagent éclaireur par lôagent mobile qui compare son code et ses données avec une copie sauvegardée.

III.3.3 La non répudiation

La signature des résultats par lôhôte permet dôassurer la non répudiation. Lôenregistrement et lôencryptage de lôitinéraire garantiront aussi cette propriété, puisque aucun site ne pourra pas nier être visité.

III.3.4 La disponibilité

La disponibilité permet dôéviter des attaques de types déni de service, où un site malicieux peut ignorer les demandes de service ou introduire des délais inacceptables pour des tâches critiques. Le fait de calculer le temps dôexécution de l'éclaireur et le comparer avec le temps estimé, permet de détecter si lôagent a été tué, capturé ou détourné.

III.3.5 L'authentification

Lôauthentification du site visité est vérifiée par lôagent éclaireur dés le moment où la clé environnementale est calculée et vérifiée. La signature des résultats partiels par le site visité permet aussi de lôauthentifier.

er tout utilisateur non autorisé d'avoir la possibilité d'analyser les flux des données. Si le site visité a l'intention d'effectuer cette action, l'agent mobile le saura en analysant le rapport de l'agent éclaireur. Le chiffrement des données permet aussi de vérifier cette propriété.

III.4 Conclusion

Le but de ce chapitre était de présenter une proposition pour protéger l'agent mobile contre différentes attaques des hôtes malicieux visités.

L'idée est basée sur la prévention et exige la présence d'un deuxième agent prototype (l'éclaireur) en plus de l'agent mobile.
L'agent éclaireur est une copie restreinte de l'agent mobile avec du code et des données sans importance. Un agent mobile, avant de migrer vers un hôte, crée un agent éclaireur puis l'envoi en premier vers l'hôte de destination.
A son retour, l'éclaireur est analysé afin de pouvoir détecter d'éventuelles attaques. Si l'agent éclaireur est attaqué, l'agent mobile choisit une politique d'adaptation qui consiste à remplacer des composants appropriés aux situations d'attaques. La structure de l'agent mobile est donc adaptée.

Contrairement aux approches existantes, notre proposition permet de minimiser les dégâts sur l'agent mobile, en exposant d'abord l'agent éclaireur aux différentes menaces des sites malveillants.
Nous avons discuté la capacité de ce protocole à vérifier les différentes propriétés de sécurité, tel que la confidentialité, l'intégrité, la non-répudiation et la disponibilité.

Pour illustrer le fonctionnement de l'architecture proposé, nous effectuerons dans le chapitre suivant l'implémentation des agents mobile et Eclaireur.
Nous avons choisi JADE, qui est une plate-forme multi-agents développée en Java, dotée d'une interface graphique et peut être répartie sur plusieurs serveurs.

4 IMPLEMENTATION SOUS JADE

L'architecture présentée dans le chapitre précédent permet de protéger l'agent mobile contre plusieurs types d'attaques.

Afin de la mettre en évidence, nous allons implémenter les différents agents proposés. Nous avons opté pour la plate-forme JADE (Java Agent Development Framework) [13], vu qu'elle propose les outils nécessaires pour l'intégration et la communication des différents éléments du protocole.

JADE est une plate-forme pour le développement des systèmes multi-agents, codée en Java et le développeur doit implémenter ses agents Java pour pouvoir l'exploiter.

Le but de ce chapitre est de donner un aperçu sur quelques principes d'implémentation de l'architecture proposée dans le chapitre précédent sous la plate-forme JADE.

En plus, nous allons illustrer une expérimentation comparative entre le protocole proposé et deux autres approches de protection des agents mobiles.

IV.1.1. Un aperçu

JADE [11, 139] est une plate-forme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA [3]. JADE comprend deux composantes de base : une plate-forme agents compatibles FIPA et un paquet logiciel pour le développement des agents en Java.

JADE inclut :

- Un environnement d'exécution distribué, dans lequel des agents peuvent s'exécuter.
- Une bibliothèque de classes permettant le développement d'agents.
- Un ensemble d'outils graphique permettant l'administration et la surveillance des activités des agents.

IV.1.2 La norme FIPA

La FIPA (Foundation for Intelligent Physical Agents) [3] est une organisation à but non lucratif fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes. Par la combinaison d'actes de langages, de logique des prédicats et d'ontologies publiques, la FIPA cherche à offrir des moyens standardisés permettant d'interpréter les communications entre agents de manière à respecter leur sens initial, ce qui est bien plus ambitieux que XML, qui ne standardise que la structure syntaxique des documents. Afin d'atteindre ce but, FIPA émet des standards couvrant :

- Les applications (applications nomades, agent de voyage personnel, applications de diffusion audiovisuelles, gestion de réseaux, assistant personnelé) ;
- Les architectures abstraites, définissant d'une manière générale les architectures d'agents ;
- Les langages d'interaction (ACL), les langages de contenu (comme SL, CCL, KIF ou RDF) et les protocoles d'interaction ;
- La gestion des agents (nommage, cycle de vie, description, mobilité, configuration);

s : représentation (textuelle, binaire ou XML) des
ar IIOP, WAP ou HTTP) de ces messages.

Ces standards évoluent, et sont régulièrement mis à jour, ainsi que de nouveaux standards qui sont nouvellement proposés. Les standards qu'édicte la FIPA ne constituent pas vraiment une plate-forme de construction multi-agents. Elle normalise une plate-forme d'exécution standardisée dans un but d'interopérabilité. Ces normes s'appliquent donc pour la plupart en phase de déploiement. Elles n'abordent pas les phases d'analyse ni de conception. Elles peuvent cependant guider certains choix d'implémentation.

IV.1.3 Composants de la plate-forme JADE

La plate-forme JADE inclut tous les composants obligatoires qui contrôlent un SMA. Ces composants sont lôAMS, lôACC et le DF [13, 14]

- AMS (Agent Management System): Cêst l'agent qui exerce le contrôle de supervision sur l'accès et l'usage de la plate-forme, il est responsable de lôauthentification des agents résidents et du contrôle d'enregistrements.
- ACC (Agent Communication Channel): Cêst l'agent qui fournit la route pour les interactions de base entre les agents dans et hors la plate-forme. Cêst la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages. Il doit aussi être compatible avec le protocole IIOP pour l'interopérabilité entre les différentes plates-formes multi-agents.
- DF (Directory Facilitator): Le Facilitateur d'Annuaire est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents.

JADE fournit une interface graphique utilisateur GUI (Graphic User Interface) pour la gestion à distance des agents RMA (Remote Management Agent), surveillant le statut des agents, permettant par exemple dôarrêter et de remettre en marche des agents.
Le GUI (fig. 4.1) permet également de créer et de lancer un agent sur un serveur à distance, à condition quôun conteneur d'agent fonctionne déjà. Le GUI permet de commander d'autres plates-formes d'agent à distance.

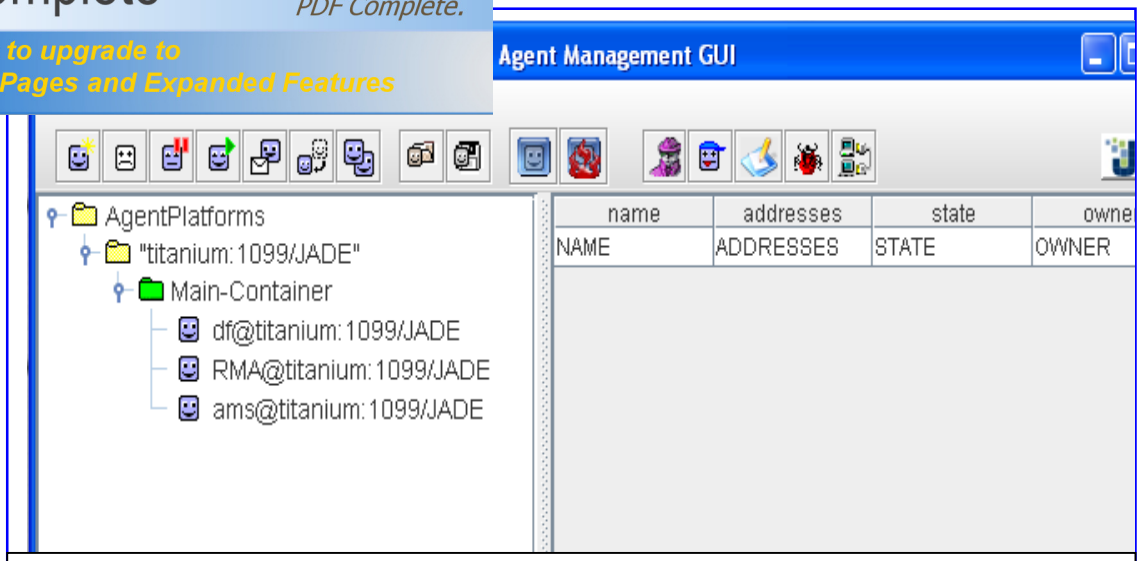


Fig. 4.1 Interface de la plate-forme JADE: l'agent GUI

En plus, il existe dans JADE d'autres outils graphiques qui soutiennent la phase de correction, habituellement très complexes dans les systèmes répartis :

- L'agent Dummy est un outil simple et très utile pour visualiser des échanges de messages entre agents. L'agent Dummy (fig. 4.2) facilite la validation d'un agent avant l'intégration dans le SMA et facilite le débogage au cas où un agent échouerait. L'interface graphique fournit un support pour éditer, composer et envoyer des messages ACL aux agents. Elle permet également de recevoir et dôinspecter les messages des agents, et par la suite de sauvegarder ou charger ces messages sur le disque dur.

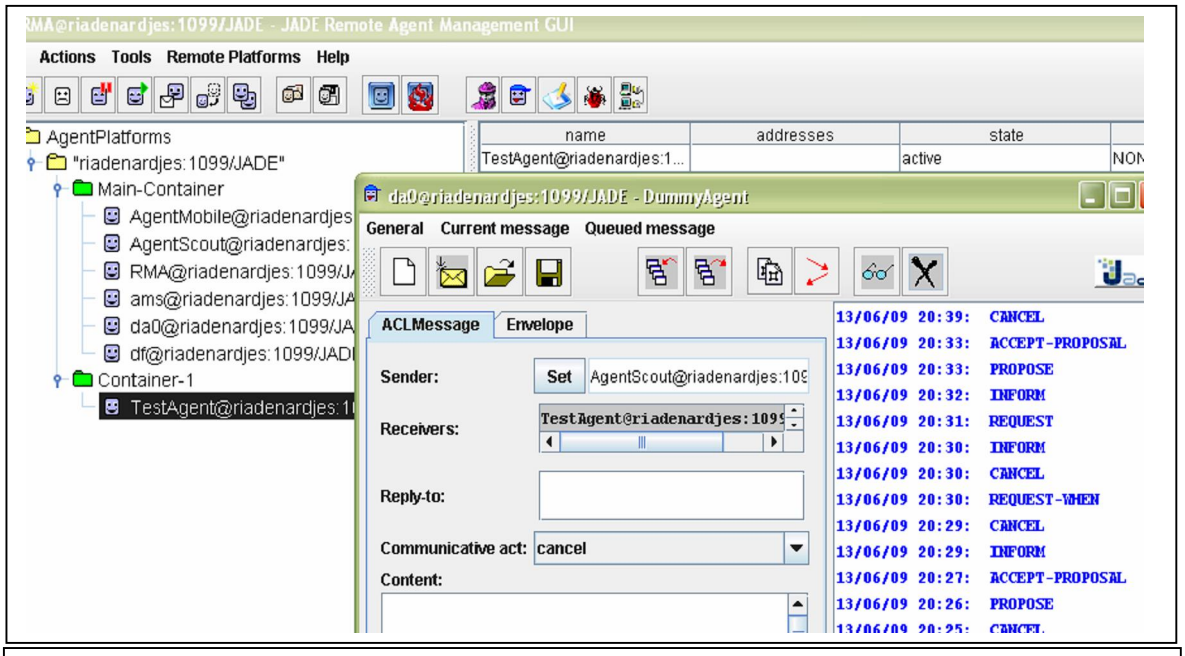


Fig. 4.2 Agent Dummy

les messages échangés dans une plate-forme d'agent « renifler » un agent, ou un groupe dôagents, chaque message dirigé vers ou venant de cet agent, ou du groupe, est dépisté et montré dans la fenêtre de Sniffer (fig. 4.3). L'utilisateur peut alors regarder, sauvegarder, charger chaque message pour une analyse postérieure.

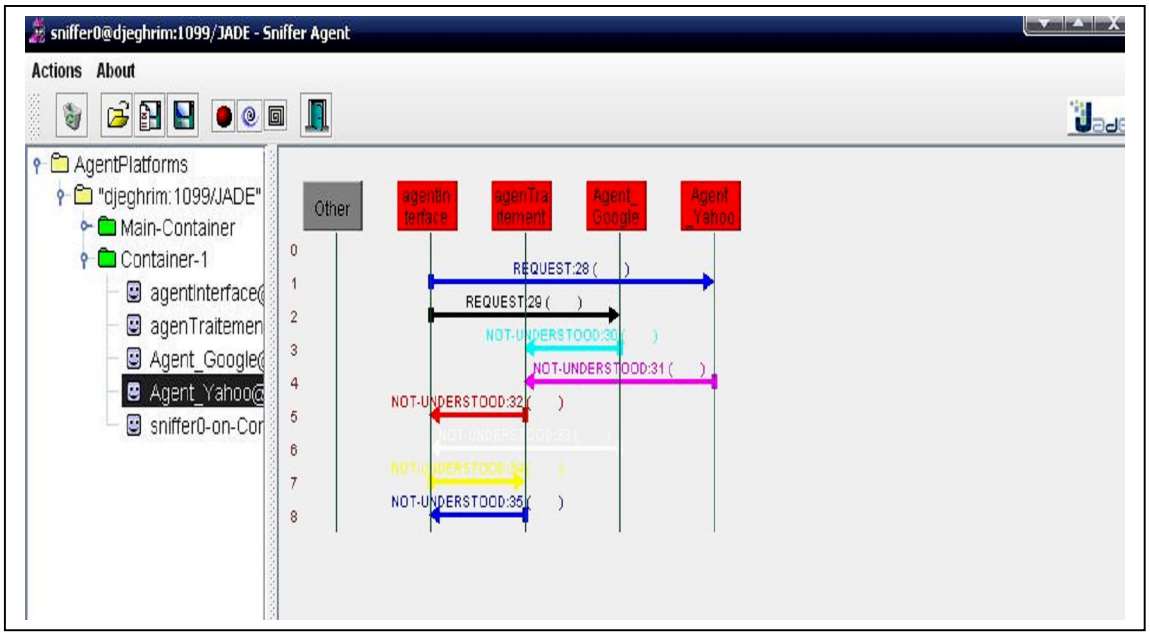


Fig.4.3 Agent Sniffer

- L'agent Introspector (fig. 4.4) permet de surveiller et de commander le cycle de vie d'un agent courant ainsi que ses messages échangés (provenant de la file d'attente des messages envoyés et reçus).

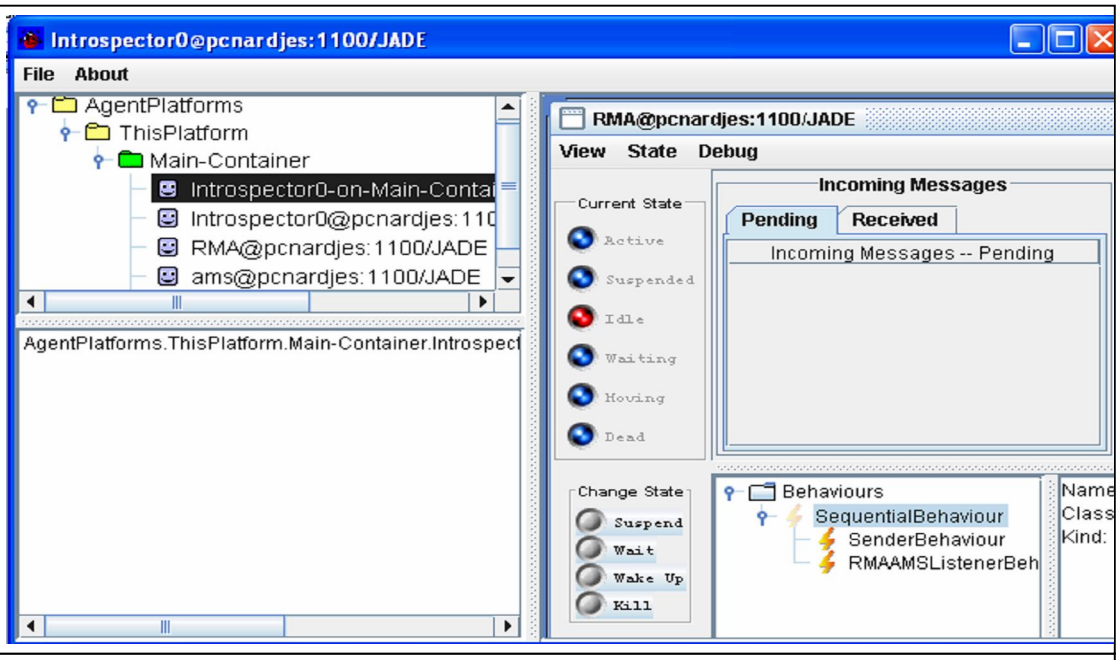


Fig.4.4 Agent Introspector

...nent (Behaviour) [13] pour modéliser les tâches qu'un agent peut exécuter, et les agents instancient leurs comportements selon leurs besoins et leurs capacités. Un agent doit être capable de gérer plusieurs tâches de manière concurrente en réponse à différents évènements extérieurs.

Afin de rendre efficace cette gestion, chaque agent de JADE est composé d'un seul thread et chaque comportement qui le compose est en fait un objet de type *Behaviour*.

Des agents multi-thread peuvent être créés, mais il n'existe pour l'heure actuelle aucun support fournis par la plate-forme (excepté la synchronisation de la file des messages ACL). JADE fournit, sous forme de classes, un ensemble de comportements ainsi que des sous-comportements prêt à l'emploi. Elle peut les exécuter selon un schéma prédéfini.

Par exemple, la classe *SequentialBehaviour* est supportée et exécute des sous-comportements de manière séquentielle. Toutes les classes prédéfinies dans JADE héritent de la classe abstraite *Behaviour*.

IV.1.5 Langage de communication de JADE

La communication entre les différents agents [55] est assurée par les messages de type ACL (Agent Communication Language). Le langage FIPA ACL [3] est le langage standard des messages et impose le codage, la sémantique et la pragmatique des messages.

FIPA spécifie que les messages transportés entre les plates-formes devraient être codés sous forme textuelle. On suppose que l'agent est en mesure de transmettre cette forme textuelle. La norme FIPA préconise des formes communes pour les conversations entre agents par la spécification de protocoles d'interaction, qui incluent des protocoles simples de type requête-réponse.

IV.2 Implémentation de l'architecture proposée sous JADE

Les différents agents du protocole proposé, doivent être intégrés dans la plate-forme JADE. Les divers points nécessaires pour cela sont présentés dans les sections suivantes.

IV.2.1 Simulation des attaques

Nous avons proposé un agent de test (AT), en plus des agents mobiles et éclaireur, afin de simuler les différentes attaques. L'agent de test génère à chaque fois des actions sur l'agent éclaireur (fig. 4.5). Nous avons crée des attaques graves, tel que modification du code et

fication de la destination. Quelques attaques douteuses copie-coller et analyse de l'agent.

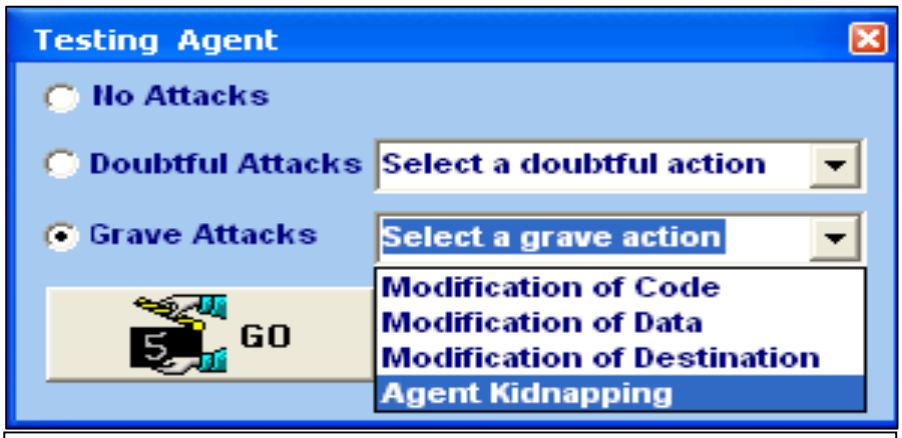


Fig. 4.5 L'interface de l'agent de Test

IV.2.2 Intégration des agents

Le système est composé d'un agent mobile, un agent éclaireur et un agent de test pour simuler les attaques. L'agent mobile est créé en premier et c'est à lui de créer l'agent éclaireur à chaque migration.

L'agent de Test est conçu pour simuler différentes actions, à savoir, des attaques graves, des attaques douteuses et des actions neutres, ou permises.

IV.2.2.1 Inscription dans le DF

Tous les agents du système doivent s'inscrire dans le DF (service des pages jaunes) pour avoir une bonne perception sur les agents de la plate-forme et les services rendus par ces derniers.

Les agents Mobile, Eclaire et l'agent Test sont inscrits dans le DF, en fournissant leurs noms et les services rendus par chacun d'eux.

IV.2.2.2 Les conteneurs

Chaque instance du JADE est appelée conteneur (container), et peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plate-forme.

Chaque plate-forme doit contenir un conteneur spécial appelé main-container et tous les autres conteneurs s'enregistrent auprès de celui-là dés leur lancement.

Nous avons besoin de deux conteneurs (Fig. 4.6) :

nts de base de la plate-forme Jade (DF, AMS, RMA),

Container-1 : contient au départ lôagent Test, responsable de simuler les attaques sur lôagent Eclaireur.

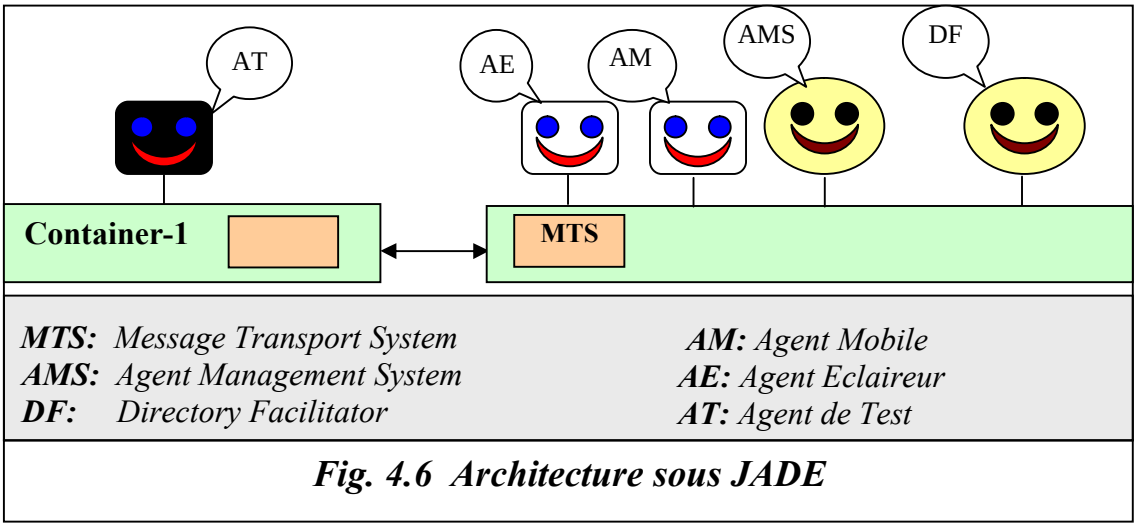


Fig. 4.6 Architecture sous JADE

IV.2.2.3 La communication

La classe *ACLMessage*, représente les messages qui peuvent être échangés par les agents. La communication de messages se fait en mode asynchrone. Lorsqu'un agent souhaite envoyer un message, il doit créer un nouvel objet *ACLMessage*, compléter ces champs avec des valeurs appropriées et enfin appeler la méthode *send*().

Lorsqu'un agent souhaite recevoir un message, il doit employer la méthode *receive*() ou la méthode *blockingReceive*().Un message ACL dispose obligatoirement des champs suivants (Tableau 4.1):

	type de l'acte de communication
	expéditeur du message
Receiver :	destinataire du message
reply-to :	participant de la communication
content :	contenu du message
language :	description du contenu
encoding :	description du contenu
ontology :	description du contenu
protocol :	contrôle de la communication
conversation-id :	contrôle de la communication
reply-with :	contrôle de la communication
in-reply-to :	contrôle de la communication
reply-by :	contrôle de la communication

Tableau 4.1 Champs d'un message ACL

Nous avons utilisé lâagent Dummy pour visualiser les différents messages échangés entre lâagent mobile et lâagent éclaireur (Fig. 4.7).

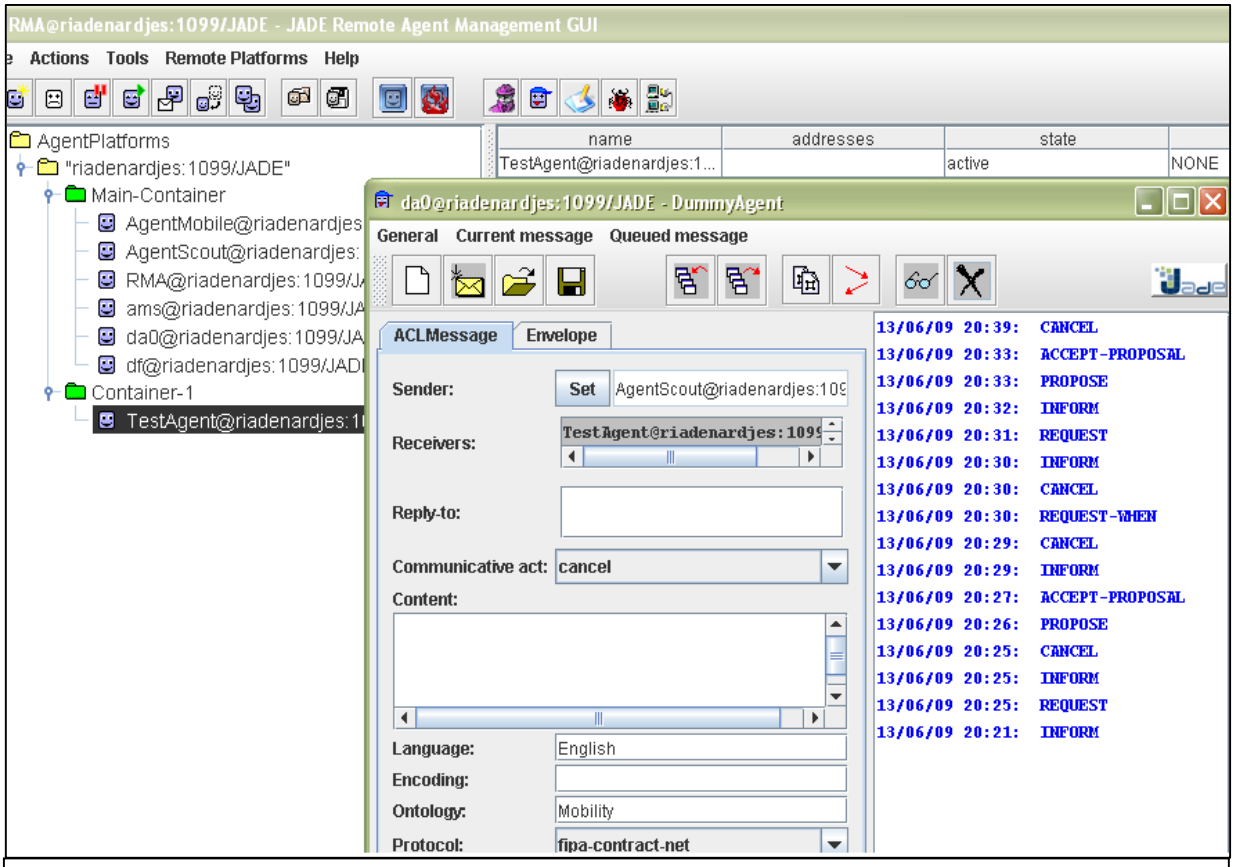


Fig. 4.7 Echanges de messages avec l'agent Dummy

t distribué, où les agents peuvent résider et se déplacer entre des conteneurs différents. Ceux-ci sont une abstraction pour étendre la plate-forme sur des hôtes multiples. Le service de mobilité de JADE permet seulement la migration entre des conteneurs, ce que l'on connaît comme la mobilité d'intra plates-formes.

IV.2.2.5 Classes et méthodes de JADE utilisées

a. La classe Agent

Cette classe est définie dans le package *jade.core* et représente une super classe commune pour tous les agents définis par lôutilisateur. Du point de vue programmeur, la conséquence est quôun agent JADE est simplement une classe JAVA qui étend la classe de base " Agent ".

Les méthodes de la classe *Agent* que nous avons utilisé sont :

- Setup () : sert à lôinitialisation de lôagent.
- tackDown () : sert à la dés-initiatisation de lôagent. Elle est appelée juste avant la suppression de lôagent de lôAMS.
- doDelete () : fait passer lôagent à lôétat " *Deleted* ", ce qui aura pour effet de supprimer lôagent de lôAMS (et appeler la méthode *tackDown* () juste avant).
- Send (ACLMessage) : lorsque un agent souhaite envoyer un message, il doit créer un nouvel objet *ACLMessage*, compléter ses champs avec des valeurs appropriées, et enfin appeler la méthode *Send* ().
- Receive (MessageTemplate) : lorsque un agent souhaite recevoir un message, il doit employer la méthode *Receive* ().
- addBehaviour(Behaviour) : permet dôajouter un comportement (*Behaviour*) à la file de comportement dôun agent.

b. La classe Behaviour

Cette classe est définie dans le package *jade.core*. JADE utilise lôabstraction comportement (*Behaviour*) pour modéliser les tâches quôun agent peut exécuter.

Tout objet de type *Behaviour* dispose dôune méthode *action ()* et une méthode *done()*.

- Action () : constitue le traitement à effectuer par un objet de type *Behaviour*.
- Done () : vérifie si le traitement dôun objet de type *Behaviour* est terminé.

classe abstraite qui hérite la classe *Behaviour,* définie dans le package *jade.core* et modélise un comportement qui s'exécute continuellement dans le temps.

d. La classe oneShotBehaviour

La classe *oneShotBehaviour* est une classe abstraite qui hérite de la classe *Behaviour,* définie dans le package *jade.core, et* modélise un comportement qui s'exécute seulement une fois dans le temps. Nous avons utilisé cette classe pour définir les comportements simples de l'agent éclaireur.

Chaque agent implémenté de la plate-forme JADE doit suivre les étapes d'exécution illustrées dans la figure suivante (Fig. 4.8):

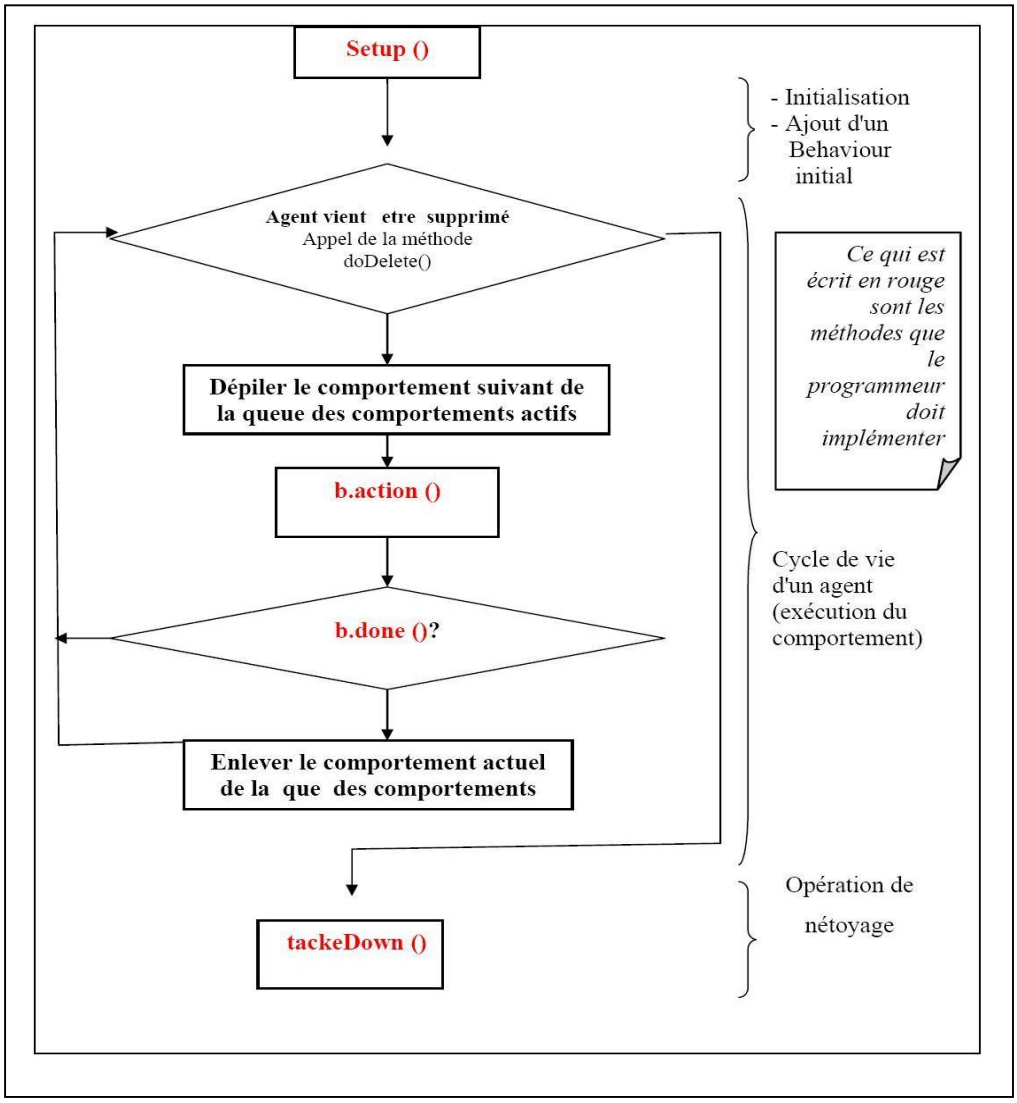


Fig. 4.8 Etapes d'exécution sous Jade

ackage *jade.lang.acl*, et représente les messages qui peuvent être échangés par les agents. Parmi les champs disposés par un message ACL remplis par un agent du système, les champs représentés dans le tableau 4.1.

Le paradigme de communication adopté entre les agents est le " passage de message asynchrone ". Chaque agent a une sorte de boite aux lettres (une queue de messages) où est posé tout message envoyé par un autre agent.

f. La classe DFservice

Elle est définie dans le package *jade.domain*. Nous avons utilisé trois méthodes implémentées par cette classe :

- Register : permet d'enregistrer un agent dans les pages jaunes du DF.
- Deregister : supprimer un agent des pages jaunes du DF.
- Search : permet à un agent d'obtenir une liste de tous les agents inscrits dans le DF qui peuvent réaliser la tâche qu'il cherche à exécuter.

La figure 4.9 Résume les classes de JADE utilisées.

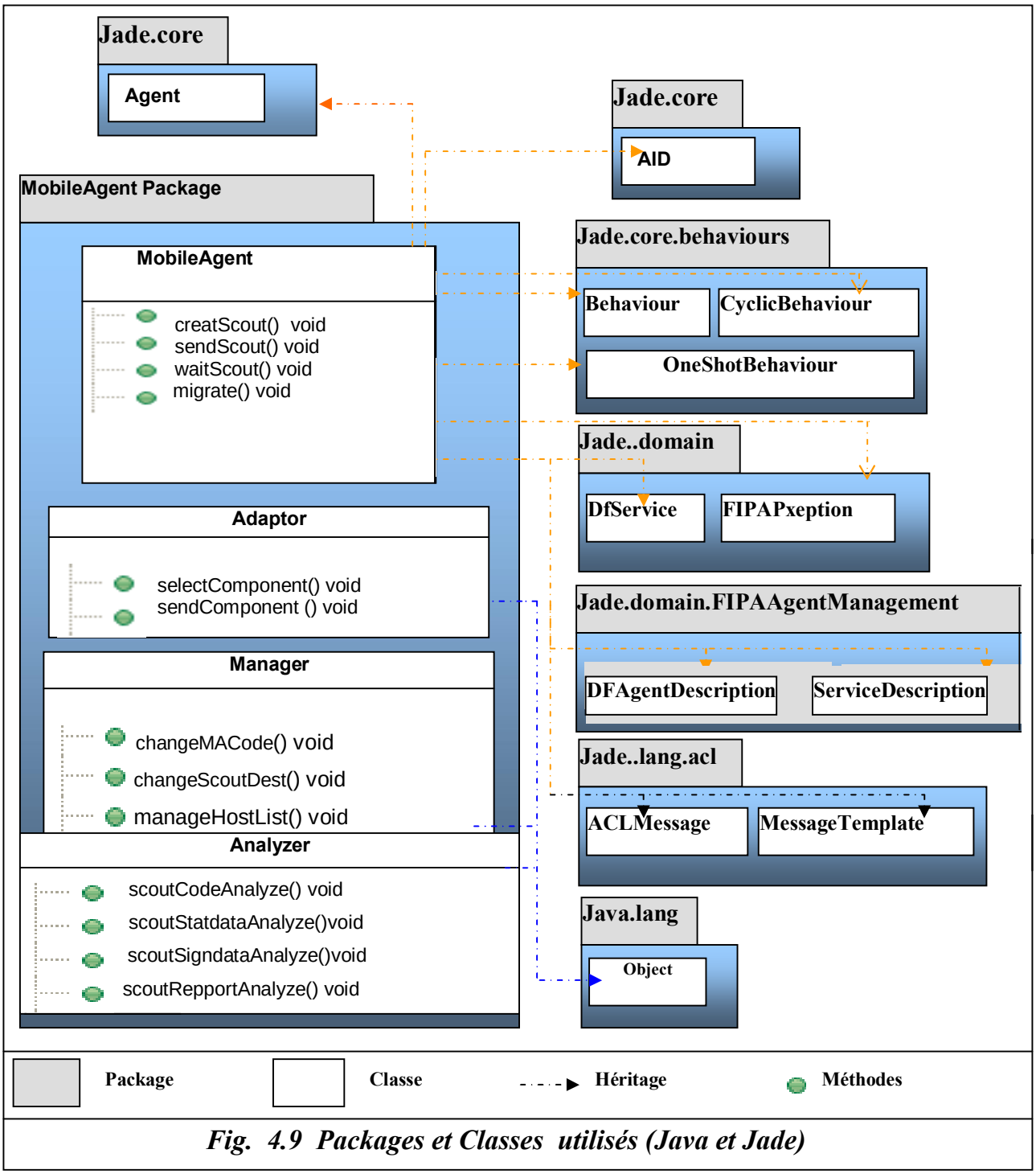


Fig. 4.9 Packages et Classes utilisés (Java et Jade)

A utilisées

ils dôimplémentation des modules les plus importants définis avec les classes Java (Fig.4.9), à savoir, lôanalyseur et lôadaptateur.

La bibliothèque de composants est une base de données qui contient des composants (ou des morceaux de code) remplaçables.

La bibliothèque dôactions est représentée par une base de données avec trois tables : une table pour les actions permises, une deuxième pour les actions douteuses et une troisième pour les actions interdites.

- ***Analyseur***

Le but de lôanalyseur est dôanalyser lôagent Eclaireur (code, données et rapport) (fig. 4.10.a). Nous présentons quelques méthodes de lôanalyseur :

- *ScoutCodeAnalyze* : Pour analyser le code de lôéclaireur après son retour, en le comparant avec une copie sauvegardée.
- *ScoutStatDataAnalyze :* Son but côest dôanalyser les données statiques qui ne doivent jamais être modifiées.
- *ScoutSignDataAnalyze :* Pour vérifier la signature des résultats par lôhôte visitée, sinon il est ajouté à la liste des sites malveillants.
- *ScoutRepportAnalyze :* Cette méthode est responsable dôanalyser le rapport de lôéclaireur où on fait lôanalyse de toutes les actions produites sur le site visité, pour détecter celles permises, non permises et douteuses.

  Si toutes les actions sont permises, lôanalyseur ajoute le site dans la liste de confiance. Si il trouve au moins une action interdite, il met le site dans la liste noire. Et enfin, sôil trouve des actions douteuses, il fait appel à lôadaptateur qui va choisir une adaptation du code, appropriée à lôaction.

- ***Adaptateur***

Contient principalement deux méthodes (Fig.4.10.b) :

- *selectComponent* : Le but de cette méthode est de sélectionner les composants nécessaires lors de lôadaptation. Lôadaptateur est composé de règles de types :

  If Action $_{(i)}$ ***then Select Composant*** $_{(i)}$
- *sendComponent :* Son rôle est d'envoyer les composants sélectionnés au manager, qui va effectuer des changements au niveau de lôagent mobile.

```
MA.creat (SA) ;          /  L'Agent Mobile crée l'Eclaireur
MA.send (SA, H1);     /   Puis l'envoi à l'hôte H1
MA.wait (SA) ;          /   L'Agent Mobile calcule le temps d'exécution de l'Eclaireur

If SA.RunTime > SA.EstimatedTime then
   Go to Lab1;
Else {
    MA.ComparCode (SA, CopyCode) ;
    MA.ComparData (SA, CopyData);
   If SAChangedCode = true or SAChngedData = true then
       go to Lab1;
   Else {
     If not SA.SignedData then go to Lab1;
      Else {
          MA.SARepportAnalize;       / Analyse du rapport de l'Eclaireur
          Case RepportAnalize = 0   / S'il contient des actions interdites (attaques graves)
                   Goto Lab1;
                RepportAnalize = 1    /Si toutes les actions sont permises
                   Goto Lab2;
                RepportAnalize = 2     /S'il y a des actions douteuses
                  Call MA.Adaptor;     /Appèl de l'adaptateur
       }
     }
}
Lab1: Coordinator.ChangeDestination; / Attaques graves
       BLH: = add (H1);                    / Le Coordinateur ajoute l'hôte dans la liste noire
Lab2: MA.MigrateTo(H1);
       TLH: = TLH + H1;
}
```

Fig.4.10.a: Pseudo programme de l'agent mobile proposé

```
Public void Adaptor () {
  Select case{
    Case Action1{
      Charge(Compenent1 from LibComponent);/ Choix du composant
       Send(Component1) to manager;
    }
    Case Action2{
      Charge(Compenent2 from LibComponent);
       Send(Component1) to manager;
    }
     ……
   Case Actioni {
      Charge(Compenent from LibComponent);
      Send(Component1) to manager;
    }
  }
}
```

Fig.4.10.b: Pseudo programme du module adaptateur

ons

Afin de valider notre proposition, nous avons développé une simulation dûne simple expérience comparative entre le protocole proposé et deux autres approches sur un réseau de trois postes.

Pour simuler le réseau, nous avons créé trois conteneurs : un conteneur principal (Main-container), considéré comme lôhôte dôorigine et deux conteneurs secondaires (container-1, container-2) pour les sites à visiter.

Ensuite, nous avons créé trois agents mobiles dont le code et les données sont identiques, mais chacun est doté d'un protocole de sécurité différent:

- Le premier agent (*AgentMobile1)* est protégé par une approche de détection, où une attaque nôest détectée quôaprès le retour au site dôorigine.
- Le deuxième agent (*AgentMobile2)* est protégé par l'approche de type tiers partie de confiance TTP (Trust Tiers Part) [26]. Dans ce cas on fait appel à un serveur de confiance, qui vérifie à chaque migration l'agent mobile et détecte s'il a été attaqué.

  Pour simuler le serveur de confiance, nous avons introduit un quatrième conteneur (container-3) (Fig.4.11).
- Le troisième agent (*AgentMobile3)* est protégé par le protocole proposé. Il est hébergé dans le main-container où il créé son agent éclaireur.

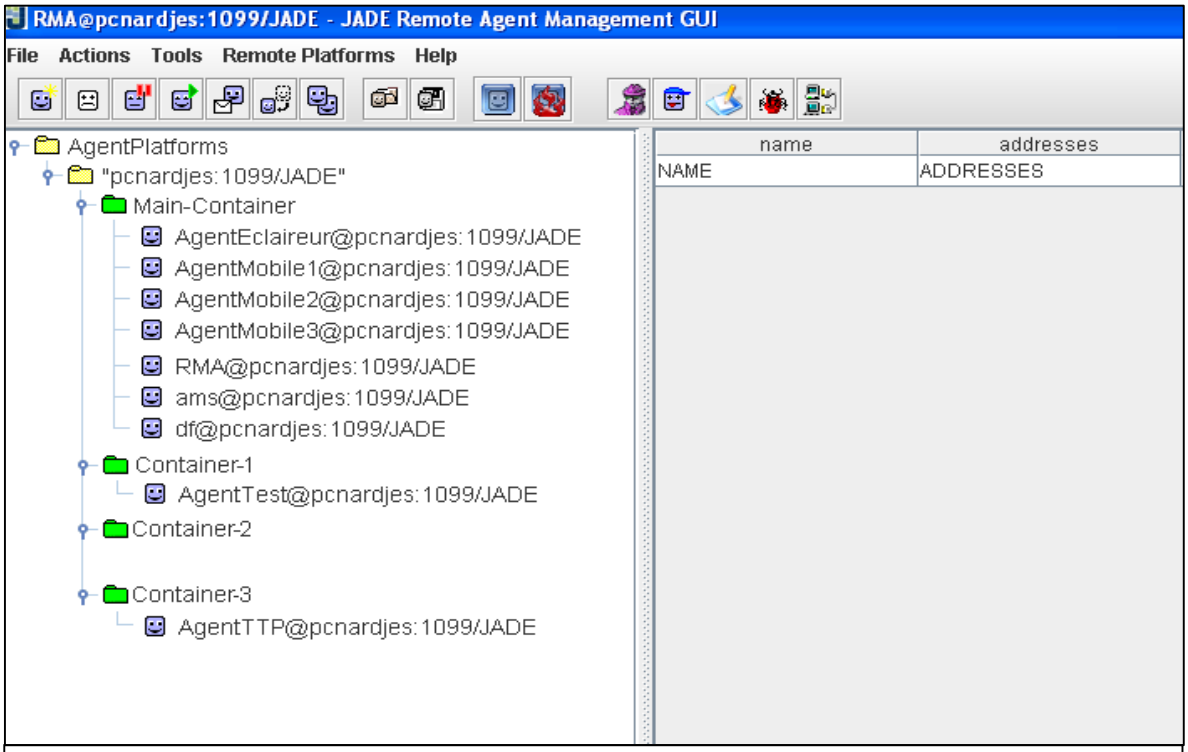


Fig.4.11 Simulation de l'expérimentation

nt de test est créé pour simuler les différentes actions :
.

Lôitinéraire prévu des trois agents (AgentMobile1, AgentMobile2, AgentMobile3) est le suivant : main-container, container-1, container-2, main-container.

Nous avons calculé, pour chaque agent, le temps quôil faut pour détecter les différentes attaques, sachant que le temps total pour aller et revenir vers le main-container est de 50 secondes.

Pour commencer, lôagent de test génère des comportements neutres (A1 : aucune attaque) :

- *AgentMobile1* doit revenir vers lôhôte dôorigine (main-container) pour être vérifié. Il lui faut 60 secondes (50 seconde pour le va et vient et 10 seconde pour la vérification).
- *AgentMobile2* est examiné dans le serveur TTP (container-3). Il va prendre 30 secondes (le temps de migration vers le container-3 (TTP) + le temps de vérification).
- *AgentMobile3* détecte que son éclaireur nôa pas été attaqué, en le comparant avec la copie sauvegardée. Pour cela il lui faut 20 secondes.

Les agents *AgentMobile1* et *AgentMobile2* mettent donc plus de temps que l'agent *AgentMobile3* pour savoir qu'il n'y a aucune attaque.

La deuxième étape de l'expérience concerne les comportements douteux : pour cela nous avons choisit les actions copie coller (A2) et analyse du code (A3):

- *AgentMobile1* nôaura aucune idée quôil a été copié ou analysé. Au meilleur des cas, il se rend compte après son retour à l'hôte d'origine (70 secondes pour A2 et 80 secondes pour A3).

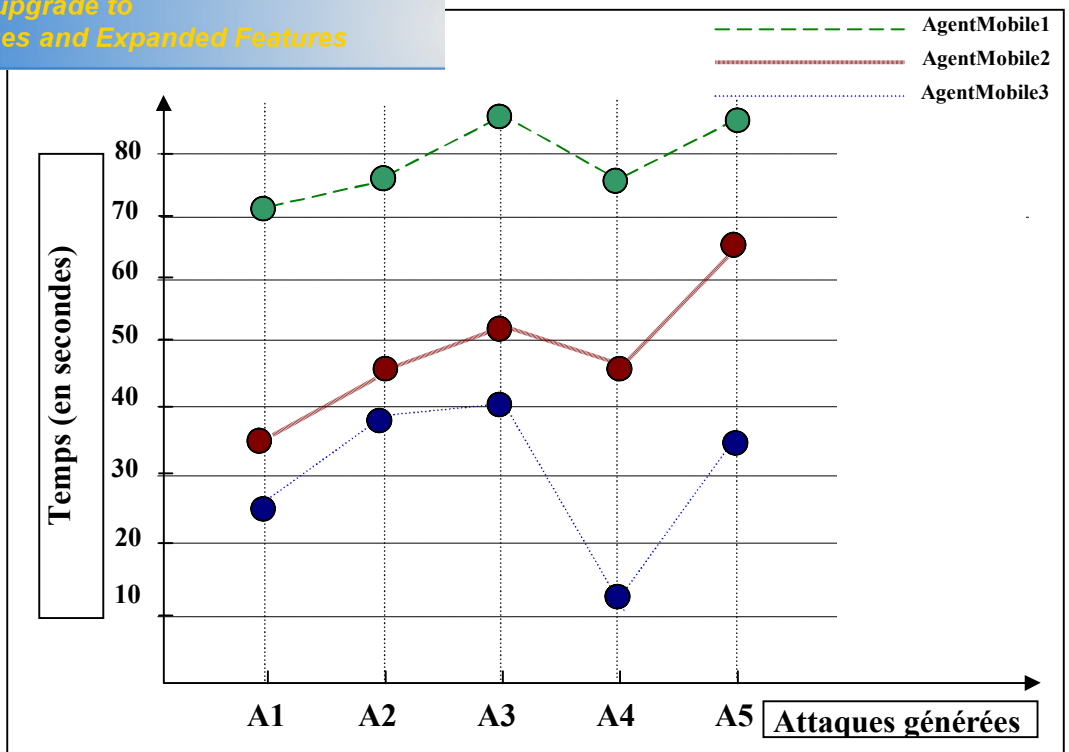
- *AgentMobile2* peut détecter ces actions, mais il lui faut un temps considérable, puisqu'il a besoin du TTP (migration vers container-3) qui va s'assurer de son intégrité (40 secondes pour A2 et 47 secondes pour A3).

s actions sur son éclaireur, puisquôil a la liste de toutes de test (30 secondes pour détecter A2 et 33 secondes pour détecter A3). Il met alors cet hôte dans la liste des sites douteux, sôadapte en choisissant le composant « obscurcissement du code », puis migre.

Finalement, lôagent de test génère des comportements interdits : modification du code (A4) et kidnapping de lôagent (A5) :

- *AgentMobile1* ne peut détecter ces attaques avant de rentrer à lôhôte dôorigine, en plus il peut perdre la trace de lôhôte responsable de lôattaque (70 secondes pour détecter lôaction A4 (retour + vérification) et 80 secondes pour A5 (lôagent ne revient pas)).
- *AgentMobile2*, et grâce au serveur de confiance (TTP), peut détecter ces attaques, mais après un certain temps (40 secondes pour détecter lôaction A4 et 60 secondes pour A5).
- *AgentMobile3* peut détecter lôaction A4 en comparant lôéclaireur avec la copie sauvegardée (8 secondes). Il peut détecter aussi le kidnapping de lôéclaireur après 30 secondes, et déduit ainsi que lôhôte visité est malicieux et le met dans sa liste noire.

Notons que plus le nombre de sites à visiter est grand plus le temps de détection des attaques par les agents *AgentMobile1* et *AgentMobile2* est plus important.
Nous avons résumé les résultats obtenus dans le graphe de la figure 4.12.



A1 : Aucune attaque **A2** : Copie Coller **A3** : Analyse de lâagent
A4 : Modification de lâagent **A5** : Kidnapping de lâagent

Fig. 4.12. Graphe de comparaison entre AgentMobile1, AgentMobile2, et AgentMobile3

IV.4 Conclusion

Afin de valider notre proposition, nous avons implémenté lâarchitecture proposée sous la plate-forme JADE (**J**ava Agent **DE**vlopment Framework). Nous avons opté pour JADE vu ses nombreux avantages pour la programmation agents.

Nous avons crée et intégré les différents agents du système, à savoir lâagent mobile et lâagent éclaireur. Nous avons proposé un agent de test pour simuler les différentes actions : interdites, douteuses et neutres.

Pour terminer, nous avons fait une comparaison entre le protocole proposé et deux autres approches : la première approche est une approche de détection, où les attaques ne sont détectées quâaprès le retour de lâagent vers lâhôte dâorigine.

un serveur de confiance (TTP, Trust Tiers Part), où migration par un serveur.

Un graphe de comparaison est élaboré afin de mieux constater la différence du temps de détection des attaques entre les trois approches.

Dans cette thèse, nous avons présenté une étude sur la protection des agents mobiles contre les différentes attaques menées par les sites malicieux. Nous avons commencé par bien souligner le champ de cette étude, à savoir les agents mobiles et leurs architectures, ensuite la sécurité des agents mobiles. Dans ce contexte, nous avons d'abord entamé la sécurité des plates-formes, qui est un axe bien maîtrisé. Puis, nous avons déterminé les différentes attaques possibles sur l'agent mobile ainsi que les approches existantes pour palier à ce problème.

Nous avons choisi de classer les techniques de protection des agents mobiles en deux grandes classes :
Classe dôapproches basées sur la prévention, dont le principe consiste à doter lôagent mobile de mécanismes de prévention pour éviter les attaques.
Classe dôapproches basées sur la détection, où les attaques sont détectées au lieu dôêtres évitées. Bien sure, chaque classe a ses avantages et ses inconvénients.

En se basant sur l'étude des différentes techniques et approches, nous avons proposé un protocole de protection de l'agent mobile, basé sur l'adaptabilité et un agent éclaireur.
Le principe général consiste à protéger l'agent mobile, en envoyant d'abord un agent éclaireur pour investiguer l'environnement. A son retour, l'éclaireur est analysé pour détecter d'éventuelles attaques et en conclure la malice du site visité.

D'une façon générale, l'architecture proposée permet de protéger l'agent mobile contre plusieurs types d'attaques, de juger si les sites à visiter sont dangereux, éviter donc de migrer et changer de destination.

système à travers son implémentation, en plus d'une expérimentation comparative pour montrer sa capacité par rapport à d'autres approches.

Cette thèse constitue une base de travail à partir de laquelle de nouvelles activités de recherche peuvent être lancées afin d'améliorer le travail présenté.
Les perspectives que nous proposons peuvent donc s'orienter vers les directions suivantes:

- L'approche proposée reste limitée à des applications non réelles. Une première perspective consiste donc à étendre l'approche proposée à des cas concrets tel que le commerce électronique ou la recherche d'information, dôabord sur un réseau locale spécialisé (pour une entreprise par exemple), puis sur Internet.

- Notre proposition était basée sur un seul agent éclaireur envoyé vers un seul site. Nous voudrions étendre cette proposition en utilisant plusieurs copies de lôéclaireur et les envoyer simultanément vers plusieurs sites. Cela permettra à lôagent mobile dôavoir une meilleure perception et de faire une évaluation de plusieurs sites à la fois.

- Lôapproche proposée reste limitée dans le cas où lôéclaireur nôest pas attaqué par un site donné, mais lôagent mobile lôest.
  Dans ce cas le protocole proposé peut être renforcé par un agent surveillant, dont le rôle est de veiller sur lôagent mobile, en le vérifiant après chaque migration.

ACL

Agent Communication Langage: Langage de communication qui combine habilement KQML et KIF.KQML est pris pour la partie administrative et KIF pour la représentation des données et/ou expressions à évaluer.

Framework

Ensemble d'outils (de conception, d'exécution, débogage) qui permettent de faire quelques chose, par exemple de mettre en î uvre d'autres programmes. Ces outils peuvent être des programmes, des méthodes de travail, des librairies, etc.

Java/JDK

Java, langage orienté objet créé en 1995 par Sun. Il présente le grand avantage d'être portable. JDK (Java Development Kit) est le kit de développement Java fournis par Sun pour le développement de programmes Java.

Thread

Programme qui s'exécute à l'intérieur d'un autre programme: ceci permet à un programme de faire plusieurs choses en même temps. Un thread est parfois appelé tâche.

C'est une organisation à but non lucratif produisant des normes pour l'inter opérabilité des agents hétérogènes de logiciel.

Framework

En informatique, un framework est un ensemble de bibliothèques, d'outils et de conventions permettant le développement d'applications. Il fournit suffisamment de briques logicielles et impose suffisamment de rigueur pour pouvoir produire une application aboutie et dont la maintenance est aisée.

Intergiciel

Un intergiciel, en anglais middleware, est un logiciel servant d intermédiaire de communication entre plusieurs applications, généralement complexes ou distribuées sur un réseau informatique.

[1] M.Aiken, E.Benzacar et C.A Cocosco, "ABC: An Intelligent Robocup Agent", Artificial Intelligence (304-526), Semestre d'hiver, 1998.

[2] J. Algesheimer, C. Cachin, J. Camenisch et G. Karjoth, ñCryptographic Security for Mobile Codeò. IEEE Symposium on Security and Privacy, 2001.

[3] J.Ametller, J.Cucurull, R.Mart, G.Navarro et S.Robles ,ñEnabling mobile agents vol. 4149, Springer, Edinburgh, UK, 2006.

[4] T.Antila, ñInternetworking with SOAPò, In Seminar on Internetworking 2004 (3cr).Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology, 27 Mai 2004.

[5] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A Sahai, S. Vadhan et K. Yang, ñOn the (Im)possibility of Obfuscating Programsò, Advances in Cryptology, Proceedings of Crypto'2001, Lecture Notes in Computer Science, Vol. 2139, pages 1ð 18, 2001.

[6] A.M. Barika, "Vers un IDS Intelligent à base dôAgents Mobiles", Thèse de DEA, laboratoires: LERIA-EPITECH-PARIS, France 2003.

[7] B. Bauer,òExtending UML for the Specification of Interaction Protocolsò, submitted for the 6th Call for Proposal of FIPA, 1999.

[8] B.Bauer, J. P.Müller, J.Odell, ñAn Extension of UML by protocols for Multiagent Interactionò, Proceeding Fourth International Conference on Multi Agent Systems, ICMAS 2000, Boston, IEEE Computer society, 2000.

[9] A. Beimel, M. Burmester, ñComputing Functions of a Shared Secretò, SIAM J Discrete Math., Vol. ~13, No.3, 324ð 345, 2000.

[10] N. M. Belaramani, ñA component-based software system with functionality adaptation for mobile computingò, Master thesis, 2002.

[11] F.Bellifemine, A.Poggi, G.Rimassa, "JADE, A FIPA-compliant agent framework", CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM'99, London, pp.97-108, April 1999.

[12] F.Bellifemine, C.Giovani, T.Tiziana, G.Rimassa, "Jade Programmer's Guide, version 2.6 " (http://sharon.cselt.it/projects/jade/), 2000.

[13] F. Bellifemine, G. Caire, T. Trucco, et G. Rimassa,ò Jade Programmerôs Guide, version *3.2* ", Livre, édition Juillet 2004.

[14] Y.Benali, ñCréez votre premier agent avec JADE et ECLIPSE », Devlopper.com, 2009.

[15] Y.S. Bennet, ñA Sanctuary for Mobile Agentsò, Secure Internet Programming, LNCS, Vol. 1603, pages 261-273, 1999.

nsuring the Integrity of Agent- Based computations by
nd International Workshop on Mobile Agents, LNCS, Vol. 1477, pages 183-194, 1998.

[17] E. Bierman et E. Cloete, ñClassification of Malicious Host Threats in Mobile Agent Computingò, Proceedings of SACICSIT2002, pages 141ð 148, 2002.

[18] D.A Birrell et B.J Nelson, òImplementing remote procedure callsò, ACM Transactions on Computer Systems, 2(1) :39ï 59, February 1984.

[19] M. Blum et S. Kanan, "Designing programs to check their work", Technical report TR-88-009, International Computer Science Institute, December 1988.

[20] D. Boneh, R.J. Lipton, ñAlgorithms for black-box fields and their application to cryptographyò, Advances in cryptology, CRYPTOð96, 1996.

[21] J.P. Briot et Y. Demazeau, ñPrincipes et architecture des systèmes multi-agents », Collection IC2, Hermes-lavoisier, 2001.

[22] A.Carzaniga, G.P Picco, et G.Vigna, òDesigning distributed Applications with Mobile Code Paradigmsò, In R. Taylor, editor, proceedings of the 19th International Conference on Software Engineering (ICSEð97), pages 22ï 32, ACM Press, 1997.

[23] B.Chaib-Draa, "Distributed Artificial Intelligence: An Overview", In: Encyclopedia of Computer Science and Technology, Vol 29 (Kent, A. and Williams, J. G. eds.), Marcel Dekker, Inc., 1994, pp. 215-243.

[24] D. Chess, C. Harrison et A. Kershenbaum, ñMobile Agents: Are They a Good Idea ?ò, Technical Report RC 19887, IBM Research Division, T.J. Watson Research Center, 1994.

[25] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris et G. Tsudik, ñItinerant Agents for Mobile Computingò, Technical Report, IBM T.J. Watson Research Center, NY, 1995.

[26] A. Corradi, R.Montanari, et C.Stefanelli, ñMobile agents integrity in e-commerce applications,ò in Proc. of the 19th IEEE International Conference on Distributed Computing Systems Workshop (ICDCSð99), 1999, Austin, Texas: IEEE Computer Society Press, pp. 59-64

[27] C. CUBAT, ñAgents Mobiles Coopérants pour les Environnements Dynamiquesò, Thèse de doctorat de lðInstitut National Polytechnique de Toulouse (ENSEEIHT), Décembre 2005.

[28] A.El Rhazi, S.Pierre, H.Boucheneb, « Secure protocol in mobile agent environnement », IEEE CCECE 2003, May 4-7, vol.2, Montereal,

[29] European Comunities,òInformation technology security evaluation criteriaò,

http://www.is-frankfurt.de/publikationenNeu/RecentDevelopmentinInformation.pdf. Juin 1991

. Swarup, ñSecurity for Mobile Agents: Authentication the European Symposium on Research in Computer Security (ESORICS), pages 1180 130, 1996.

[31] W. Farmer, J. Guttmann et V. Swarup, "Security for Mobile Agents: Issues and requirements", in: Proceedings of the National Information Systems Security Conference (NISSC 96), 1996.

[32] J.Ferber, « Les Systèmes Multi-Agents. Informatique Intelligence Articielleò, InterEdition, 1995.

[33] E. Filiol, ñStrong Cryptography Armoured Computer Viruses Forbidding Code Analysis: the Bradley virusò, Proceedings of the 14th EICAR Conference, 2005.

[34] A.Fuggetta, GP.Picco et G.Vigna, "Understanding Code Mobility", IEEE Transaction on Software Engineering, 24(5): 242-361, Mai 1998.

[35] JM.Geib, C.Gransart, et P.Merle, "CORBA : des concepts à la pratique", Editions Dunod, Paris, France, Octobre 1999.

[36] L. Gong, M. Mueller, H. Prafullchandra et R. Schemers, ñGoing Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2ò, In Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, December 1997.

[37] L. Gong, ñJava Security Architecture (JDK1.2)ò, Technical Report, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A, 1998.

[38] H.Guan, X.Meng et H.Zhang, ñA forward integrity and itinerary secrecy protocol for mobile agents,ò Wuhan University Journal of Natural Sciences, China, vol.11, No.6, pp. 1727-1730, 2006.

[39] Z.Guessoum, ñModèles et architectures d'agents et de systèmes multi-agents adaptatifsò, Thèse d'habilitation de l'Université de paris 6, Laboratoire d'Informatique de Paris 6, 2003.

[40] Z.Guessoum, M.Ziane et N.Faci, ñMonitoring and Organizational Level Adaptation of Multi-Agent Systemsò, AAMASô04, ACM, New York, 2004.

[41] A.Guillemet, G.Haïk, T.Meurisse, J.P.Briot, M.Lhuillier, « Mise en oeuvre dôune approche componentielle pour la conception dôagents », in M.P. Gleizes, P. Marcenac (eds), Septièmes Journées Francophones sur lôIntelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMAô99), Hermès Science Publications, Paris, France, p. 53-66, November, 1999.

[42] S. Hacini, "Using Adaptability to Protect Mobile Agents Codeò, IEEE International Conference on Information Technology ITCC 2005, Las Vegas, pages 49- 53, USA, 2005.

ufaida, ñUsing a Trust-Based Key to Protect Mobile eering, Computing and Technology, vol. ~16, ISSN 1305-5313 © 2006 World Informatika Society, CCIS'2006, Venice, Italy, pages 326-332, 2006.

[44] S. Hacini, C. Cheribi et Z. Boufaïda, "Dynamic Adaptability using Reflexivity for Mobile Agent Protection", in Transactions On Engineering, Computing And Technology Enformatika V17 2006 ISSN 1305-5313, Cairo, Egypte, pages 222-227, December 2006.

[45] S. Hacini, Z. Boufaïda et C. Cheribi,òMobile Agent Protection in e-business Applications: A Dynamic Adaptability Based Approachò, The Second Symposium On Information Security (ISô07). Springer Lncs. Vilamoura, Algarve, Portugal, November 2007.

[46] S. Hacini, Z. Guessoum et Z. Boufaïda, ñTAMAP: A New Trust-based Approach for Mobile Agent Protectionò, Journal in Computer Virology, Springer Paris, ISSN 1772-9890 (print) 1772-9904 (online), vol.3, n°4/Nov.2007, pages 267- 283, DOI 10.1007/s11416-007-0056-y.

[47] S. Hacini, « Sécurité des Systèmes dôInformation : Mise en oeuvre de la confiance et de lôadaptabilité pour la protection de lôagent mobile", Thèse de doctorat, Université de Constantine, 2008

[48] M. Hauswirth, C. Kerer et R. Kurmanowytsch, ñA secure execution framework for Java,ò In Proceedings of the 7th ACM conference on computer and communications security (CCS 2000), Athens, Greece, pages 43-52, November 2000.

[49] F. Hohl, ñAn approach to solve the problem of malicious hostsò, Universität Stuttgart, Fakultät Informatik, Fakultätsbericht Nr. 1997/03, 1997.

[50] F.Hohl, ñA Model of Attacks of Malicious Hosts against Mobile Agentsò, In Fourth Workshop on Mobile Object Systems (MOS'98): Secure Internet Mobile Computations http://cuiwww.unige.ch/coopws/ws98 /papers/ hohl.ps, 1998.

[50] F.Hohl, ñA Protocol to Detect Malicious Hosts Attacks by Using Reference Statesò, Technical Report No.09/1999, Faculty of Informatics, University of Stuttgart, Germany, August 1999.

[51] F.Hohl, ñA framework to Protect Mobile Agents by Using References Statesò, In Proceedings of ICDCS 2000, 2000

[52] L.Ismail et D.Hagimont, « Spécialisation de serveurs par des agents mobiles », Colloque International sur les Nouvelles Technologies de lôInformation (NOTEREô98), Montréal, Canada, novembre 1998, http://sirac.inrialpes.fr/~hagimont/papers/98-notere-PUB.ps.gz.

[53] L. Ismail et D. Hagimont, ñA performance evaluation of mobile agent paradigmò, In Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLAô99), pages 306ï 313, November 1999.

j ISO/IEC 9594-8, "Information Technology Open ry: Public Key and Attribute Certificate Frameworks, March 2000. (www.infosecurity.org.ch/content/pki_pmi/x509v4.pdf)

[55] M.W. Jang et G. Agha,"On efficient communication and service agent discovery in multi-agent systemsò, In 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2004), pages 27ï 33, 2004.

[56] W. Jansen et T. Karygiannis, ñMobile Agent Securityò, NIST Special Publication 800-19, National Institute of Standard and Technology, 2000.

[57] W. Jansen, ñCountermeasures for mobile Agent securityò, Computer Communications, Special issue on Advance Security Techniques for Network Protection, Elsevier Science, 2000.

[58] W. Jansen, "Determining Privileges of Mobile Agents", in Proceedings of the Computer Security Applications Conference, December 2001.

[59] W. Jansen, "A Privilege Management Scheme for Mobile Agent Systems", First International Workshop on Security of Mobile Multiagent Systems, Autonomous Agents Conference, May 2001.

[60] W. Jansen, "Countermeasures for Mobile Agent Security", National Institute of Standards and Technology, Gaithersburg, MD 20899, USA, October 2001

[61] N. R. Jennings et M. J. Wooldridge, ñApplications of Intelligent Agentsò, In Nicholas R. Jennings and Michael J.Wooldridge, editors, Agent Technology: Foundations, Applications, and Markets. Springer-Verlag Berlin Heidelberg New York, pages 3ï 28, 1998.

[62] G. Karjoth, N. Asokan et C. Gülcü, ñProtecting the computation results of free-roaming agentsò, In Kurt Rothermael and Fitz Hohl, editors, proc. Of the second International Workshop, Mobile Agents 98, Springer-Verlag Lecture Notes in Computer Science No.1477, pages 195-207, 1998.

[63] D. B. Lange et M. Oshima, ñSeven good reasons for mobile agentsò, Commun, ACM 42, 3, pages 88-89, March 1999.

[63] P. Lee, ñProof-carrying code,ò Retrieved December 28, 2003, from Web site: http://www-2.cs.cmu.edu/~petel/papers/pcc/pcc.html

[64] S.Leriche et J.P.Arcangeli, ñUne architecture pour les agents mobiles adaptablesò, Dana Journées Composants, JCõ2004, Lille 2004.

[65] S.Lerriche, "Architectures à composants et agents pour la conception dõapplications réparties adaptables", Thèse de doctorat, Université Toulouse III, France, 2006.

[66] S. Loureiro, R. Molva et Y. Roudier, "Mobile Code Security," Institut Eurocom, France, 2001.

ction", Thèse de doctorat, Institut Eurocom, France,

[68] J. Malenfant, F. Peschanski, L. Seinturier et J-P. Briot, "ALADYN : Architectures logicielles pour lôauto-adaptabilité dynamique", Université Pierre et Marie Curie UFR dôinformatique, 2004.

[69] A. Maña et E. Pimentel, ñAn efficient software protection schemeò, University of Malaga, Spain, 2002.

[70] S.McGrath, D.Chacón et K. Whitebread, ñIntelligent Mobile Agents in Military Command and Control,ò MILCOM 2001, Vol.1, 2001.

[71] G. McGraw et E. Felten, ñSecuring JAVA [Electronic version]ò, John Wiley and Sons, 1996. http://www.securingjava.com

[72] MJ.Mendes et FM. de Assis Silva, "Mobile agents in autonomous decentralized systems". In ISADS, pages 258ï 260, 1999.

[73] B.Meyer, "Object-Oriented Software Construction", The Object-Oriented Series. Prentice-Hall, Englewood Cliffs (NJ), USA, 2nd edition, 1997.

[74] D.S.Milojicic, F. Douglis, et R. Wheeler, "Mobility: processes, computers and agents", Addison-Wesley, 1999.

[75] D.S. Milojicic, F. Douglis, Y. Paindaveine, R.Weeler, et S. Zhou, "Process migration", ACM Computing Surveys, 32(3):241ï 299, September 2000.

[76] J. J. Ordille, ñWhen Agents Roam, who Can You Trust?ò, Proceedings of the First Conference on Emerging Technologies and Applications in Communications, Portland, Oregon, May 1996.

[77] A.Ouardani, S.Pierre et H.Boucheneb, ñA Security protocol for mobile agents based upon the cooperation on sedentary agentsò, Journal of Network and Computer Applications, pp 1228-12, 2006.

[78] S. Perret, ñAgents mobiles pour lôaccès nomade à lôinformation répartie dans les réseaux de grande envergureò, Thèse de Doctorat, Université Joseph Fourier - Grenoble I, 1997.

[79] D. Phan, ñCognitive Economics, chapter From Agent-Based Computational Economics towards Cognitive Economicsò, pages 371-398. Springer Verlag, 2004.

[80] G.P.Picco, "Understanding, Evaluating, Formalizing, and Exploiting Code Mobility", PhD hesis, Politecnico di Torino, Italy, Février 1998.

[81] S.Pierre, "Mobile Agents for Telecommunication Applications", Third International Workshop, MATA 2001, Montreal, Canada, August 14-16, 2001, Proceedings (Paperback), Series: Lecture Notes in Computer Science.

[82] S.Pierre, "Réseaux et systèmes informatiques mobiles: Fondements, architectures et applications", Livre, Editeur sur Presse inter Polytechnique, 2003.

ts for Management Systems using Mobile Agentsò, um on Computers and Communications: ISCC 2000, Antibes, France, pages 160-165, 2000.

[84] L. Rejeb, ñSimulation multi-agents de modèles économiques: Vers des systèmes multi-agents adaptatifsò, Thèse de doctorat, Université de Reims Champagne-Ardennes, France, 2005.

[85], J. Riordan, B. Schneier, ñEnvironment key Generation towards Clueless Agentsò, Lecture Notes in Computer Science, Vol. 1419, pages 15-24, 1998.

[86] V. Roth, ñSecure Recording of Itineraries Through Cooperating Agentsò, Proceedings of the ECOOP Workshop on Distributed Object Security and 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, INRIA, France pages 147-154, 1998.

[87] V. Roth, ñMutual Protection of Cooperating Agentsò, Secure Internet Programming: Security Issues for Mobile and Distributed Objects, J. Vitek and C. Jensen (Eds.), Springer Verlag, 1999.

[88] V. Roth, ñObstacles to the adoption of mobile agentsò, In 5th IEEE International Conference on Mobile Data Management (MDM 2004), IEEE Computer Society, pages 19-22, January 2004.

[89] K.Rothermel et M.Schwehm,"Mobile Agents", A. Kent and J.G Williams (Eas): Encyclopedia for Computer Science and Technology, New York: M. Dekker Inc, 1998

[90] S.Rouvrais, " Utilisation d'Agents Mobiles Pour la Construction de Services Distribués", Thèse de Doctorat de l'Université de Rennes, 2002.

[91] D. Rubin et D. E. Geer, "Mobile code security," IEEE Internet Computing, 1998, Li Gong, òSecure java class loading,ò IEEE Internet Computing, pages 56-61, 1998.

[92] T.Sander et C.Tschudin, ñProtecting Mobile Agent against Malicious Hostsò, G.Vigna (Ed.), Mobile Agents and Security, Lecture Notes in Computer Science, Vol. 1419, ©Springer-Verlag Berlin Heidelberg, Berlin, 1998.

[93] F. B. Schneider, ñTowards fault-tolerant and secure agentryò, In M. Mavronicolas and P. Tsigas, editors, Proceedings of the Eleventh International Workshop on Distributed Algorithms, number 1320 in LNCS. Springer-Verlag, Berlin, pages 1-14, 1997.

[94] V. Swarup, ñTrust Appraisal and Secure Routing of Mobile Agentsò, DARPA Workshop on Foundations for Secure Mobile Code, Position Paper. Monterey, CA, USA, March 1997.

[95] H.Tan et K.Moreau, ñTrust Relationships in a Mobile Agent Systemò, In G. P. Picco, editor, Fifth IEEE International Conference on Mobile Agents, Lecture Notes in Computer Science, vol. 2240, Springer-Verlag, Atlanta, Georgia, 2001.

[96] H. K. Tan et L. Moreau, ñExtending execution tracing for mobile code securityò, In Fischer, K. and Hutter, D., Eds. Proc. of 2nd Intl Workshop on Security of Mobile MultiAgent Systems (SEMAS'2002), Bologna, Italy, pages 51-59, July 2002.

ges for mobile code", ACM Computing Surveys,

[98] C. Tschudin, "Apoptosis - the Programmed Death of Distributed Services", Secure Internet Programming 1999: 253-260.
http://www.informatik.uni-trier.de/~ley/db/conf/ecoopw/secure99.html#Tschudin99

[99] G. Vigna, ñProtecting mobile agents through tracingò, In Proceedings of the Third ECOOP Workshop on Operating System support for Mobile Object Systems, Finland, pages 137-153, June 1997.

[100] G. Vigna, ñMobile Code Securityò, Lecture Notes in Computer Science, Vol.1419, Springer-Verlag Berlin Heidelberg, Berlin, 1998.

[101] G. Vigna, ñMobile Agents: Ten Reasons for failureò, In 5th IEEE International Conference on Mobile Data Management (MDM 2004), pages 298-299. IEEE Computer Society, pages 19-22, January 2004.

[102] T. Wang, S. Guan et T. Khoon Chan, ñIntegrity Protection for Code-On-Demand Mobile Agents in E-Commerceò, The Journal of Systems and Software 60, pages 211-221, 2000.

[103] J. E. White, ñTelescript Technology: The foundation for the electronic marketplaceò, White paper, General Magic, Inc., 2465 Latham Street, Mountain View, CA 94040, 1994

[104] U. G.Wilhelm, S. Staamann et L. Buttyan, ñOn the Problem of Trust in Mobile Agent Systemsò, IEEE Symposium on Network and Distributed system Security, San Diego, California, 1998.

[105] U.G.Wilhelm, "Cryptographically Protected Objects", Technical Report, Ecole Polytechnique Federale de Lausanne, Switzerland, 1997.

[106] M.Yao, M.Peng, et E.Dawson, ñUsing óFair Forfeitô to Prevent Truncation Attacks on Mobile Agents,ò in Proc. of the 10th Australasian Conference on Information Security and Privacy, ACISP 2005, Brisbane, Australia, pp.158-169,2005

[107] M.Yao, K.Peng, M.Henricksen, E.Foo, et E.Dawson, ñUsing Recoverable Key Commitment to Defend Against Truncation Attacks in Mobile Agents,ò in Proc. of the 5th International Conference on E-Commerce and Web Technologies (EC-Web 2004), volume 3182 of Lecture Notes in Computer Science, pp.164-173, Springer-Verlag, 2004.

[108] B.Yee, ñA sanctuary for mobile agentsò, In Jan Vitek and Christian Jensen, editors, Secure Internet Programming: Security Issues for Mobile and Distributed Objects, number 1603 in LNCS. Springer-Verlag, Berlin, pages 261-274, 1999.

[109] J. Zachary, ñProtecting mobile code in the wildò, IEEE Internet Computing, 7(2), pages 78ï 82, 2003.

[110] W. Zhi et G. Zhogwen, ñA Dynamic Security Adaptation Mechanism For Mobile Agentsò, Proceedings of the International Computer Congress, China, page 334-339, 2004.

[111] www.twinforces.com/frictionle

[112] http://whois.domaintools.com/auctionbot ss.com

[113] http://www.awt.be/web/can/index.aspx?page=can,fr,int,200,030

[114] http://www.conversagent.com/

[115] http://fr.wikipedia.org/wiki/Webdiffusion

[116] http://cormas.cirad.fr/

[117] http://jade.tilab.com/

[118] www.fipa.org/repository/aclspecs.html

[119] http://www.madkit.or

[120] (Le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier) www.lirmm.fr/

[121] http://www2.lifl.fr/SMAC/projects/magique/

[122] http://www.jagent.org

[123] http://www.hds.utc.fr/~barthes/SMAS/

[124] http://www.agent-software.com

[125] http://www.alexa.com

[126] http://www.dev-heba.wikispaces.com/file/view/odyssey.doc

[127] http://global.mitsubishielectric.com/

[128] http://www.inf.fu-berlin.de/lehre/WS99/VS/Misc/Voyager/API/

[129] http://www.cs.dartmouth.edu/~agent/general/agenttcl.html

[130] www.tacoma.cs.uit.no/

[131] http://portal.acm.org/citation.cfm?id=895112

[132] http://www-cdr.stanford.edu/ProcessLink/papers/JATL.html

[133] http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm

tech/aglets

[135] http://www.disi.unige.it/person/MascardiV/Software/jessInJADE-Tutorial.pdf

[136]http://iml.univ-mrs.fr/ati/files/short-rolland-smfhttp://iml.univ-mrs.fr/ati/files/short-rolland-smf

[137] http://fr.wikipedia.org/wiki/Cryptographie_sym

[138] http://fr.wikipedia.org/wiki/Middleware

[139] http://sharon.cselt.it/projects/jade/

[140] ftp://ftp.cs.bham.ac.uk/pub/authors/J.L.Wyatt/rl97.ps.gz