Aperçu général sur la technologie des agents mobiles

Zakaria Maamar

Pattern Analysis and Machine Intelligence Lab
Department of Systems Design Engineering
University ofWaterloo, Waterloo, Ontario, Canada N2L 3G1
e-mail:maamar@watfast.uwaterloo.ca

Introduction

Les agents mobiles sont une catégorie particulière des agents logiciels dont la caractéristique prédominante est leur capacité de se transporter entre les nœuds d'un réseau ou de plusieurs réseaux [1]. Les agents mobiles (ou transportables) sont une extension directe du paradigme client/serveur. Dans ce paradigme, les entités communicantes ont des rôles fixes et bien définis. Un serveur offre des services alors que le client procède à leur utilisation. La communication qui prend place entre le client et le serveur est basée sur un échange de messages, obligeant les programmeurs à gérer divers aspects liés à la communication comme les adresses réseaux et les points de synchronisation.

Le système des appels de procédures à distance (RPC : Remote Procédure Call) développé par la compagnie Sun MicroSystems tente de réduire les détails de la communication, en permettant au client d'exécuter un service distant de la même manière qu'un appel de fonctions locales. L'emplacement du serveur, l'initiation du service et le transport des résultats sont pris en charge au profit du client de manière transparente.

Cependant malgré l'introduction du système RPC, un problème fondamental existe dans les architectures client/serveur. Si le serveur ne fournit pas le service que le client exige, alors ce dernier doit réaliser plusieurs appels à distance afin d'obtenir le service requis. Ceci peut résulter en une augmentation des temps d'attente et une transmission d'informations intermédiaires non utiles et non efficaces.

Une autre tentative de résolution des problèmes de fonctionnement des architectures client/serveur est l'introduction de la sous-programmation [1]. Elle permet aux clients d'exécuter leurs programmes dans le nœud où le service réside. De cette manière, les

requêtes peuvent être initiées localement et les programmes peuvent traiter les résultats intermédiaires avant de transmettre les résultats finaux au client une fois le tout terminé. Bien que les programmes puissent migrer de leurs sites originaux, ils ne peuvent pas se déplacer une deuxième fois vers d'autres systèmes ou ressources.
De plus, ces programmes sont généralement écrits pour des clients spécifiques, rendant ainsi leur réutilisation difficile.

La technologie des agents mobiles tente d'adresser les diverses insuffisances liées aux paradigmes client/serveur et de la sous-programmation. Les caractéristiques typiques des agents mobiles sont leurs capacités de migrer quand ils veulent, leur autonomie dans leurs actions, leur liens avec les autres agents et leur indépendance par rapport à leur emplacement original [1].

.

La mobilité est une caractéristique désirable dans les agents pour plusieurs raisons [1] :

- Efficacité : si un agent peut se diriger vers des emplacements désirés, alors le trafic réseau pourra être réduit. L'agent peut traiter l'information et décider si elle est importante à transférer. L'aspect de transfert est crucial quand il s'agit de situations où la connexion réseau est de faible débit.
- Persistance : une fois qu'un agent mobile est mis en place, il n'est pas dépendant du système qui l'a initié et ne doit pas être affecté si celui-ci venait à tomber en panne. La capacité de se déplacer entre les noeuds des réseaux donne à l'agent la capacité de "survivre" et d'atteindre le plus de ressources possibles. Ceci est particulièrement utile pour les usagers ayant des ordinateurs mobiles; ces usagers peuvent se connecter, initier l'agent, se déconnecter et contrôler de façon intermittente son progrès.
- Communication "peer-to-peer" : un inconvénient majeur du paradigme client/ serveur est la non capacité des serveurs à communiquer entre eux. Les agents mobiles sont considérés comme des entités paritaires ("peer") et peuvent ainsi adopter la position la plus appropriée à leurs besoins actuels, en étant des clients ou des serveurs.
- Tolérance aux fautes : dans une situation d'échec du serveur ou du réseau pendant une requête, il est difficile pour le client de retrouver la situation initiale afin de se synchroniser de nouveau avec le serveur. Les agents mobiles n'ont pas besoin de maintenir des connexions permanentes, étant donné que leurs états d'exécution sont centralisés à l'intérieur d'eux.

Les applications qui utilisent la mobilité comme un mécanisme pour s'adapter aux changements de ressources doivent satisfaire trois exigences. Premièrement, ces applications doivent être conscientes de leurs environnements d'exécution. En particulier, elles ont besoin de contrôler le niveau et la qualité des ressources que leurs environnements disposent. Deuxièmement, elles ont besoin d'être capables de réagir aux

changements de la disponibilité des ressources. Troisièmement, elles doivent être capables de contrôler la manière avec laquelle les ressources sont utilisées en leurs noms.

1.1 Domaines d'applications

Il existe plusieurs applications pour lesquelles les agents mobiles pourraient être utilisés. La majorité de ces applications s'intéresse à la recherche d'informations au nom de l'usager et possiblement à l'exécution de transactions spécifiques quand les informations appropriées sont rencontrées. Voici ci-dessous une liste d'applications susceptibles d'utiliser le paradigme des agents mobiles comme mécanisme de fonctionnement [2,3].

- Collecte de données de plusieurs places : une différence majeure entre le code mobile, comme les applets, et les agents mobiles est l'itinéraire. Alors que le code mobile voyage d'ordinaire d'un point A à un point B, les agents mobiles ont un itinéraire plus complexe et peuvent voyager de manière séquentielle entre plusieurs sites. Une application naturelle par conséquent est la collecte d'informations localisées dans plusieurs ordinateurs d'un réseau.
- Recherche et filtrage : étant donné le volume important (et en croissance) d'informations disponibles sur l'Internet, la collecte d'informations exige la recherche dans un grand espace de données pour déterminer des portions d'informations pertinentes. Le filtrage des informations non-pertinentes est souvent un processus consommateur de temps. À la demande de l'usager, un agent mobile pourrait visiter plusieurs sites, chercher à travers les informations disponibles dans chaque site et construire un index des liens pour les portions d'information qui répondent aux critères de sélection.
- Négociation : au lieu de chercher dans des bases de données ou dans des fichiers, les agents peuvent obtenir de l'information en interagissant avec d'autres agents. Si par exemple une personne désire organiser une rencontre avec diverses personnes, elle pourrait envoyer son agent mobile pour interagir avec les agents représentatifs de chacune de ces personnes. Les agents pourraient négocier et décider du temps et du lieu de la rencontre selon les contraintes de chaque personne. Ces contraintes sont exprimées dans un langage de spécification compréhensible aux agents.
- Ainsi que d'autres domaines d'application, comme la surveillance des réseaux [4J, le traitement parallèle et le commerce électronique.

1.2 Structure d'un agent mobile

Un agent mobile se compose d'un programme code et d'un ensemble d'états persistants. Le code, en principe, est écrit dans n'importe quel langage de programmation. Cependant, pour des questions pratiques certains types de langages sont potentiellement plus utiles. Une des considérations est que le code de l'agent devrait s'exécuter de la même manière sur chaque site accessible à l'agent. Ainsi, les langages traditionnels comme le

1. Telescript de la compagnie General Magic est conçu pour le développement de grands systèmes distribués. La sécurité a été l'un des facteurs qui ont orienté la conception du langage, ainsi que la capacité de la migration des unités de traitement pendant leur exécution.
2. Obliq de DEC est un langage interprété, basé sur les objets et non-typé. Les objets Obliq sont locaux aux sites mais il est possible de déplacer des traitements d'un site vers un autre.
3. Tel (Tool Gommand Language) étendu permet à un script Tel en exécution de se déplacer d'un site à un autre.
4. MO de l'université de Genève est un langage interprété et basé sur les piles qui implantent les concepts de messagers. Ceux-ci sont des séquences d'instruction qui sont transmises entre des sites et inconditionnellement exécutées une fois reçues.
5. Tycoon de l'université de Hamburg est un langage fonctionnel, d'ordre supérieur, polymorphique et persistant. Le langage est aussi étendu pour supporter les caractéristiques des systèmes mobiles.
6. Facile de l'ECRC de l'université de Munich est un langage fonctionnel qui étend le langage ML standard avec des primitives de distribution, de concurrence et de communication.

2.2 Question de la sécurité

La sécurité dans les systèmes d'agents mobiles touche deux parties [6]. La sécurité du système contre les atteintes d'un agent dangereux et malveillant, et la sécurité d'un agent contre un système qui tente de l'espionner, de lui voler des informations confidentielles ou de changer son code d'exécution. Dans ce cas, l'agent pourrait fonctionner au nom du système alors que les frais de traitement seraient chargés au propriétaire original de l'agent.

Dans les systèmes d'agents mobiles, les questions reliées à la sécurité sont très diverses. Elles incluent : (1) la communication entre les plates-formes doit être sécure, (2) les agents entrants doivent être authentifiés et autorisés à exécuter des opérations en fonction de leurs identités, (3) toutes les interactions entre l'infrastructure de support des agents et les ordinateurs hôtes doivent être contrôlées et vérifiées, (4) l'infrastructure doit être protégée des agents malveillants et (5) les agents doivent être protégés entre eux. Le Tableau suivant reprend les divers risques ainsi que les techniques de défense généralement utilisées.

Composantes de la sécurité dans les systèmes d'agents mobiles

Startél		
Réseau	accès à des informations privées encapsulées dans un agent touchant à son état	techniques de cryptographie, authentification, canaux sécures
Système hôte	accès non-autorisé à des ressources locales	& accès au hôte accordé par l'infrastructure selon une base discriminatoire
Plate-forme	Interférence dans l'état de l'infrastructure en lisant, écrivant ou exécutant du code	communication agent-infrastructure sécure et comptabilisation de l'usage des ressources
Agent	Interférence dans l'état de l'agent en lisant, écrivant ou exécutant du code	communication inter-agents sécure et comptabilisation de l'usage des ressources

La sécurité du réseau de communication et de l'authentification est un problème bien étudié et souvent résolu par des techniques de cryptographie. La protection du système hôte et des droits d'accès a une grande similarité avec la protection des systèmes d'exploitation. La difficulté est de définir les ressources à protéger afin de mettre en place des mécanismes de contrôle d'accès flexibles. La protection des infrastructures de support des agents peut être réalisée en imposant certaines restrictions aux programmes, comme l'interdiction de manipuler arbitrairement les zones mémoires. La communication entre les agents est aussi importante, étant donné que les agents ont le pouvoir de communiquer librement, tout en leur interdisant d'interférer dans les opérations d'exécution des autres agents.

Un agent peut être soumis à quatre catégories d'attaques [6] : violation de la partie privée, violation de l'intégrité, "masquerading" (Usurpation de l'autorité ou de l'identité d'un autre agent pour une série d'actions) et dénégation d'un service (Consommation excessive d'une ressource finie comme du temps de traitement, de la mémoire ou du sous-système de communication)

III. Agent Tel, un exemple de systèmes d'agents mobiles

Le système d'agents mobiles Tel est un projet du département d'informatique de l'université de Dartmounth [7]. L'architecture de l'agent TCL est basée sur un modèle de serveurs supporté par Telescript et Tel étendu. Le projet Agent Tel vise à atteindre les objectifs suivants :

- Réduire l'opération de migration à une simple instruction comme le go de Telescript et permettre à cette instruction d'apparaître à tout moment et à n'importe quel niveau du programme code.
- Fournir une communication transparente entre les agents.
- Supporter de multiples langages et mécanismes de transport tout en permettant des extensions.
- Fournir une sécurité efficace dans le monde incertain de l'Internet.

Tous les services qui sont disponibles à l'intérieur du système sont fournis par des agents, mobiles ou stationnaires. Ces agents peuvent être écrits dans un langage qui supporte l'interprétation comme Tel ou Java. Les agents dont le langage est compilé sont aussi supportés mais avec certaines restrictions.

Dans chaque site d'agent Tel, un serveur réside et assure la gestion des agents locaux et des nouveaux agents en entrée. Le serveur fournit également les mécanismes de sécurité et un nom d'espace hiérarchique dans lequel les agents peuvent être référencés et adressés. Actuellement, le serveur est implanté comme deux processus coopératifs; un processus surveille le réseau tandis que l'autre gère une table d'agents en exécution. Dans l'architecture Tel, le serveur Tel réalise les fonctions suivantes :

- Statut : le serveur garde trace des agents qui sont exécutés sur sa machine et répond aux requêtes à propos de leurs statuts.
- Migration : le serveur accepte chaque agent en entrée, authentifie l'identité de son propriétaire et transmet l'agent authentifié à l'interpréteur approprié. Le serveur sélectionne le meilleur mécanisme de transport pour chaque agent.
- Communication : le serveur offre un espace hiérarchique de noms pour les agents, ce qui permet aux agents d'envoyer des messages à chacun d'eux à l'intérieur de cet espace.
- Sauvegarde persistante : le serveur fournit des accès à une unité de stockage persistante, pour que les agents puissent récupérer leurs états internes. Le serveur peut aussi récupérer les agents de cette unité dans le cas d'une panne de la machine.

Les agents se déplacent entre les sites selon une manière orientée-état en invoquant la commande de mobilité agentjump. Cette commande regroupe l'état de l'agent et le transfère vers le site de destination où le serveur redémarre l'agent à l'instruction qui suit

immédiatement la commande de mobilité. La méthode avec laquelle un agent est communiqué est déterminée par le système de transport du site serveur tel que le TCP/IP, le email, etc.

L'exécution de l'agent est supportée par un interpréteur qui est approprié à son langage source. Dans l'architecture Tel, l'interpréteur de Tel a été étendu pour intégrer trois modules supplémentaires : le module de sécurité qui prévient qu'un agent n'exécute des actions malveillantes, le module de capture des états qui regroupe et récupère l'état interne de l'agent et le module serveur API qui permet les interactions avec les serveurs pour fournir les fonctionnalités de migration et de communication.

La version alpha de l'agent Tel supporte seulement la communication synchrone et orientée-réseau à travers les commandes agent meet pour initier une communication avec un autre agent et agent accept qui finalise et synchronise les deux agents. Egalement, le mouvement entre les sites est prédéterminé avec un ensemble de sites à visiter. L'objectif est de permettre aux agents de tenir compte de leurs environnements et de planifier une stratégie de migration.

IV. Aglet d'IBM, un exemple d'environnements de développement d'agents mobiles

Le principe des applets de Java a révolutionné le monde du WWW. Les applets sont des programmes codes qui peuvent être chargés, initiés et exécutés dans des navigateurs Web. Récemment, ce concept a été égalé par l'introduction des servlets. Une servlet déplace du code d'un programme dans le sens opposé d'une applet; ainsi, elle permet au programme client d'exécuter des opérations additionnelles au niveau du serveur. Le code de la servlet est par la suite instantié et exécuté dans le serveur.

L'Aglet représente la prochaine étape dans l'évolution des opérations d'exécution dans l'Internet, introduisant l'idée que le code d'un programme peut être transporté avec ses états [3,8]. Les Aglets sont des objets Java qui peuvent se déplacer d'un site vers un autre. Une Aglet qui s'exécute sur un site peut subitement arrêter son exécution, s'expédier vers un autre site distant et reprendre son exécution de nouveau. Quand l'Aglet se déplace, elle emmène avec elle son programme code et ses données. De manière conceptuelle, une Aglet est un agent mobile, étant donné, qu'elle supporte les principes de l'exécution autonome et le routage dynamique au niveau de son itinéraire.

L'infrastructure Java-Aglet API (J-AAPI) est un standard pour l'industrie qui est proposé pour des agents d'Internet basés sur Java. J-AAPI a été développé par une équipe de

chercheurs d'IBM Tokyo. Cette infrastructure vient en réponse à un appel d'offre pour une plate-forme uniforme pour les agents mobiles dans des environnements hétérogènes comme l'Internet. Egalement, cette infrastructure contient des méthodes pour l'initialisation d'une Aglet, le support de messages et l'envoi, le retrait, l'activation/ désactivation, le clonage et la suppression d'une Aglet.

Les concepts associés à l'infrastructure J-AAPI sont :

- Une Aglet est un objet Java mobile qui visite des sites supportant des Aglets dans un réseau d'ordinateurs. Elle est autonome puisqu'elle s'exécute sur son propre thread d'exécution après arrivée au site et réactive à cause de sa capacité à répondre aux messages entrants.
- Un contexte est la place de travail de l'Aglet. C'est un objet stationnaire qui fournit un moyen pour le maintien et la gestion des Aglets en exécution dans un environnement uniforme où le système du site hôte est sécurisé contre des Aglets malveillantes. Un nœud dans un réseau d'ordinateurs peut héberger plusieurs contextes.
- Un proxy, qui est le représentant de l'Aglet, sert comme son bouclier (shield) de protection contre les accès directs aux méthodes publiques. Le proxy fournit aussi une transparence pour l'emplacement de l'Aglet, en cachant l'emplacement réel de l'Aglet.
- Un message est un objet échangé entre les Aglets de manière synchrone comme asynchrone. Le passage de messages peut être utilisé par les Aglets pour collaborer et échanger des informations.
- Un gestionnaire de messages permet le contrôle concurrentiel des messages entrants.
- Un itinéraire est le plan de déplacement de l'Aglet.
- Un identificateur est associé à chaque Aglet. Cette identification est globalement unique et inchangeable pour toute la période de vie de l'Aglet.

Le comportement des agents est supporté par les opérations suivantes : création, clonage, transfert ("dispatching"), récupération ("retraction"), désactivation, activation, suppression ("disposai of') et l'envoi de messages.

- La création d'une Aglet prend place dans un contexte. La nouvelle Aglet se voit assigner un identificateur, insérer dans un contexte et initialiser. L'Aglet débute l'exécution une fois son initialisation réussie.
- Le clonage d'une Aglet produit une copie identique de l'Aglet originale dans le même contexte. Les seules différences sont l'identificateur assigné et le fait que l'exécution recommence pour la nouvelle Aglet. Notons que les threads d'exécution ne sont pas clones.
- Le transfert d'une Aglet d'un contexte à un autre procède à la suppression de son contexte courant et à l'insertion dans le contexte de destination, où elle continue son

Références Bibliographiques

[1] **J. Dale.** A mobile agent architecture to support distributed resource information management. Rapport technique, Department of Electronics and Computer Science....

[2] **M. Huhns et M. Singh.** Agents on the web, mobile agents. *IEEE Internet Internet Computing,* pages 80-82, Mai-Juin 1997.

[3] **B. Venners.** Solve real problems wifh aglets, a type of mbile agent. Rapport technique, Netscape World Magazine, Mai 1997.

[4] **T. Magedanz, K. Rothermel et S. Krause.** Intelligent agents : An emerging technology for next génération télécommunications? In *INFOCOM'96,* San Francisco, USA, Mars 24-28 1996.

[5] **G.Cugola, C.Ghezzi, G.Picco, et G.~Vigna.** Analyzing mobile code languages In *Mobile Object Systems Towards the Programmable Internet,* pages 94-109. Jan Vitek and Christian Tschudin (Eds.), 1996.

[6] **J. Vitek.** Secure object spaces. In *Proceedings of the 2"" ECOOP Workshop on Mobile Object Systems,* pages 41-48, Linz, Austria, Juillet 8-9 1996.

[7] **R.Gray.** Agent tel: A flexible and secure mobile-agent system. In *Proceedings of Fourth Annual Usenix Tcl/Tk Workshop,* 1996.

[8] **D. Lange and M. Oshima.** Programming mobile agents in java - with the java aglet api. Rapport technique, IBM Research, Tokyo, 1997.

[9] ***K. Bowes, A. Brayford et*** *E.* ***Domshy.*** *Intelligent agent topic report. Rapport* technique, Calcagry Unïversïty, 1997.