N° d'ordre : 02 / 2000-M / IN

Université des Sciences et de la Technologie Houari Boumedienne
(U.S.T.H.B.) Alger

Institut d'Informatique

THESE

Présentée à l'U.S.T.H.B. en vue de l'obtention du diplôme de

MAGISTER

Spécialité : Informatique
Option : Systèmes répartis et réseaux

Par
Mr. Tahar GHERBI

STRUCTURATION D'APPLICATIONS REPARTIES DANS UN ENVIRONNEMENT MOBILE

Soutenue le 01 / 03 / 2000, devant la commission d'examen

Mme. H. Drias	Professeur (U.S.T.H.B.)	Président
Mr. N. Badache	Maître de conférences (U.S.T.H.B.)	Rapporteur
Mme. Z. Alimazighi	Maître de conférences (U.S.T.H.B.)	Examinateur
Mr. H. Khelalfa	Chargé de recherches (CE.R.I.S.T)	Examinateur
Mr. Y. Zafoune	Maître assistant (U.S.T.H.B.)	Invité

RESUME

Le développement d'applications réparties (ou distribuées) en environnement mobile soulève plusieurs problèmes. Premièrement, les ordinateurs mobiles ne sont pas connectés en permanence au reste du réseau et en sont souvent déconnectés pendant des durées assez longues. Deuxièmement, même lors de sa connexion au réseau, un ordinateur mobile communique avec le reste des sites du système à travers des liaisons sans fil, à faible bande passante et qui peuvent être rompues ou dégradées par des obstacles physiques. Troisièmement, un ordinateur mobile peut être forcé d'utiliser différents canaux de transmission, en fonction de sa position géographique; par conséquent, la qualité et le débit de sa liaison peuvent varier de manière significative d'une session à une autre. Finalement, à un ordinateur mobile peuvent correspondre différentes adresses réseau dépendant de sa localisation et de la nature du canal de transmission utilisé.

L'approche d'agents mobiles constitue un paradigme adéquat, puissant et efficace pour la structuration d'applications distribuées. Elle est mieux vue comme un outil général permettant de réaliser des applications distribuées quelconques. En plus, c'est un paradigme excellent, lorsqu'il s'agit d'environnement mobile.
En effet, un agent mobile est un programme qui peut se déplacer de manière autonome à travers un réseau, d'un site à un autre, et interagir avec d'autres agents et des ressources sur chacun des sites visités. Il peut, par exemple, migrer depuis un ordinateur et naviguer sur Internet pour collecter des informations pour son utilisateur. L'accès aux ressources nécessaires est, ainsi, plus efficace puisque l'agent mobile se déplace sur le site même de localisation de chacune des ressources, évitant l'envoi de multiples messages de requête et de réponse sur des liaisons à faible bande passante. En plus, un agent mobile n'est pas affecté par une soudaine rupture de connexion et peut continuer à accomplir sa tâche même si l'utilisateur correspondant éteint complètement son ordinateur ou se déconnecte du réseau. Lorsque l'utilisateur se reconnecte, l'agent mobile peut revenir sur son ordinateur transportant le résultat de son activité sur le réseau.

Vu l'adéquation de l'approche d'agents mobiles à la structuration des applications distribuées en environnement mobile, notre travail consiste à proposer une infrastructure parvenant à supporter les agents mobiles en environnement mobile. Cette infrastructure doit être ouverte, flexible, facilement acceptable, et doit permettre de réaliser des applications distribuées quelconques.

Mots clés:
Application distribuée, Modèle Client/Serveur, Agent, Communication sans fil, Ordinateur mobile, Environnement mobile, Agent mobile.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Les applications de traitement de l'information ont souvent exigé (et continuent à exiger) d'être de plus en plus distribuées sur de nombreux sites distants [PB95]. Les systèmes distribués qui les supportent sont souvent basés sur *le modèle Client/Serveur*, dans lequel on distingue deux entités: le fournisseur du service (ou serveur) et le consommateur du service (ou client) [Fiu94, Har95].
Le client et le serveur résident généralement sur des sites distincts; ainsi, les interactions entre les clients et les serveurs exigent souvent (pour s'accomplir) des communications à travers le réseau. [Gra95b]

Ces dernières années ont connu une abondance des informations accessibles aux applications; par conséquent, la capacité de transport des réseaux de communication est devenue considérablement insuffisante pour supporter efficacement les interactions entre les clients et les serveurs.
D'autre part, l'évolution rapide de la technologie dans le domaine de la communication cellulaire, des réseaux locaux sans fil et des services dispensés à travers les satellites de télécommunication ont donné naissance à un nouveau paradigme de calcul, appelé *l'environnement mobile*. Cet environnement n'astreint plus l'usager à une localisation fixe; mais, lui permet de se déplacer librement tout en restant connecté au réseau. L'utilisateur peut donc, tout en se déplaçant, accéder à l'information disponible n'importe quand, et à partir de n'importe quel endroit. Les *ordinateurs mobiles* supportés peuvent être de diverses configurations (avec ou sans disque, des capacités de mémorisation plus ou moins modeste, etc.); mais, doivent être équipés d'une *interface de communication sans fil* leur permettant d'accéder au réseau tout en étant mobiles.

L'abondance des informations disponibles aux applications, la capacité de transport limitée offerte par les réseaux de communication et, récemment, le calcul mobile ont rendu les modèles Client/Serveur existants inadéquats pour supporter le trafic important qui peut en résulter sur le réseau et mettre en œuvre des applications complexes. Il était alors nécessaire de développer d'autres approches pouvant servir efficacement les besoins des utilisateurs et utiliser intelligemment les ressources du réseau. C'est ainsi que *l'approche d'agents mobiles* est apparue pour étendre et remplacer le modèle Client/Serveur [Gra95b, Har95].

L'approche d'agents mobiles est une approche récente pour l'architecture et la réalisation des systèmes distribués. Elle modélise les applications à travers un ensemble d'agents mobiles. Ces derniers sont des programmes écrits dans un langage d'écriture d'agents (tel que: Telescript, Perl, Java, etc.), nommés, autonomes et pouvant communiquer et migrer quand ils le veulent, et là où ils le veulent.

L'approche d'agents mobiles est mieux vue comme un outil général permettant de réaliser des applications distribuées quelconques [GKNRC96]. Elle constitue, en fait, un bon support pour les applications distribuées et un excellent support lorsqu'il s'agit d'environnement mobile [GKNRC96, Har95].
Adopter l'approche d'agents mobiles pour structurer les applications distribuées en environnement mobile, constitue alors un choix très prometteur. Par conséquent, notre travail consiste à proposer une infrastructure parvenant à supporter les agents mobiles en

environnement mobile. Cette infrastructure doit être ouverte, flexible, facilement acceptable, et doit permettre de réaliser des applications distribuées quelconques.

Plusieurs systèmes d'agents mobiles existent; cependant, leurs infrastructures généralement diffèrent. Il n'y a pas d'infrastructure commune pour réaliser les agents mobiles; en plus, il est improbable qu'un standard commun pour réaliser les agents mobiles émergera dans le futur proche [LDD95].

Dans l'absence d'une infrastructure qui peut servir comme un standard pour développer les applications d'agents mobiles, nous nous sommes intéressés, parmi les systèmes d'agents mobiles existants, essentiellement à: Agent-Tcl et FFMAIN.

- Agent-Tcl est un système d'agents mobiles en cours de développement au collège de Dartmouth. Son infrastructure est assez complète, assez générale et clairement présentée à travers une documentation largement disponible. Par conséquent, l'infrastructure d'Agent-Tcl peut constituer un choix très intéressant pour le développement des applications d'agents mobiles pouvant fonctionner également en environnement mobile.
- FFMAIN (FrankFurt Mobile Agent INfrastructure) est un autre système d'agents mobiles en cours de développement à l'université de Frankfurt. C'est un système qui est également très intéressant et dont l'infrastructure est aussi assez générale. Cette dernière se rapproche globalement de l'infrastructure d'Agent-Tcl, et diffère par le fait qu'elle est complètement basée sur le protocole HTTP (HyperText Transfert Protocol, un standard d'Internet) et qu'elle supporte la communication des agents à travers un moyen de communication abstrait (appelé, espace d'information), accessible à l'aide de HTTP.

Dans l'espérance d'obtenir une infrastructure d'agents mobiles qui soit assez complète, assez générale, familière, plus convenable et largement acceptée, nous avons pensé à combiner essentiellement entre l'infrastructure d'Agent-Tcl et celle de FFMAIN. L'infrastructure résultante est celle que nous proposons pour supporter les agents mobiles dans un environnement mobile; nous l'avons appelée: SINDBAD.
SINDBAD est, en fait, l'infrastructure d'Agent-Tcl rendue complètement basée sur HTTP et supportant la communication des agents avec l'approche de l'espace d'information accessible à l'aide de HTTP.

Le chapitre (I) de ce document introduit l'environnement mobile, ses caractéristiques et les nouveaux problèmes qui en découlent. Le chapitre (II) examine l'adéquation de l'approche d'agents mobiles à la structuration des applications distribuées. Le chapitre (III) propose une infrastructure d'agents mobiles destinée à supporter les applications distribuées en environnement mobile. Ce chapitre est composé de deux parties: la partie (I) discute l'idée de choisir une infrastructure d'agents mobiles parmi les infrastructures d'agents mobiles existantes, et la partie (II) décrit l'infrastructure SINDBAD proposée pour supporter les applications d'agents mobiles en environnement mobile (mais, sans développer comment la rendre basée sur HTTP). Finalement, le chapitre (IV) décrit comment rendre SINDBAD complètement basé sur HTTP et énumère les avantages qui peuvent résulter.

CHAPITRE I

ENVIRONNEMENT MOBILE: DESCRIPTION, CARACTERISTIQUES ET PROBLEMES

1. INTRODUCTION

L'évolution rapide de la technologie dans les domaines de la communication cellulaire, des réseaux locaux sans fil et des services dispensés à travers les satellites de télécommunication va permettre à des usagers munis d'unités de calcul potables[1] d'accéder à l'information n'importe où, et n'importe quand. Ces unités peuvent être de diverses configurations: avec ou sans disque, des capacités de mémorisation et de calcul plus ou moins modestes et alimentées par des sources d'énergie autonomes (batteries). Elles sont équipées d'interfaces de communication sans fil pour l'accès aux réseaux d'information. [IB94, Bad95]
L'environnement de calcul résultant, appelé environnement *mobile* ou *nomade*, n'astreint plus l'usager à une localisation fixe, mais lui permet une libre mobilité tout en restant connecté[2] au réseau. Toutefois, lorsque les utilisateurs se déplacent avec leurs unités, apparaissent toute une série de problèmes, jusqu'alors totalement ignorés en environnements classiques fixes. En effet, un environnement mobile est caractérisé par de fréquentes déconnexions, des limitations significatives des sources d'énergie et du débit de transfert de l'information, des restrictions sur les ressources utilisées (capacité disque et mémoire, vitesses du processeur) et des changements fréquents de localisation. Des mécanismes tenant compte de ces caractéristiques sont alors indispensables pour avoir accès et employer les services du réseau. A cette fin, il est nécessaire de pouvoir nommer et localiser les unités mobiles, router les messages, gérer les diverses informations (localisation, fichiers, etc.), gérer les déconnexions dues au médium de communication sans fil, etc.
D'ailleurs, de nombreuses contributions ont été consacrées à la définition du rôle des sites mobiles et leur impact sur les solutions adoptées dans les systèmes distribués classiques (ou fixes). [Duc92, PB93, FZ94, IB94, Bad95]

D'après [FZ94], le calcul mobile est un nouveau paradigme qui va révolutionner la façon d'utiliser les ordinateurs. La mobilité et la portabilité permettront le développement de nouvelles classes d'applications: services d'informations avec accès à diverses bases de données en tout lieu et en tout temps (pages jaunes, distribution, spectacles, etc.) et des applications dites verticales[3]: compagnies de location, localisation d'employés dans une entreprise, etc.
La messagerie électronique connaîtra un développement spectaculaire: les usagers munis de communicateurs électroniques pourront envoyer et recevoir des messages de n'importe où; en plus, les nouvelles électroniques (electronic news) leur seront délivrées en fonction de leurs profils respectifs. [IB92, Bad95]

---

[1] Dans le futur proche, des dizaines de millions d'usagers posséderont des ordinateurs portables. [IB94]
[2] Il ne s'agit pas obligatoirement d'une liaison filaire pour se connecter au réseau, ou pour alimenter l'unité mobile.
[3] Les applications verticales sont écrites pour un domaine d'applications spécifiques, à l'opposé des applications orthogonales qui s'appliquent à des domaines indépendants. [Sai96]

La permanence de la connexion des usagers aux réseaux d'information, indépendamment de leur position géographique, contribuera également au développement des applications coopératives. [IB94, Bad95]

La section 2 de ce chapitre décrit un modèle de système distribué intégrant des sites mobiles, la section 3 présente les caractéristiques des réseaux sans fil et la section 4 discute les nouveaux problèmes rencontrés dans un environnement mobile.

2. MODELE DE SYSTEME DISTRIBUE AVEC SITES MOBILES

Dans un environnement mobile, on distingue deux ensembles d'entités: les sites (ou unités) fixes et les sites (ou unités) mobiles [BAI94a, BAI94b, Ach95]. Les sites fixes et les liens de communication qui les relient constituent un réseau statique classique (qui est souvent filaire: Wired Network).

Pour permettre à une unité mobile (qu'on note UM) de se déplacer tout en restant connectée au réseau, certains sites fixes, appelés Stations Support Mobile (qu'on note SSMs)[1], sont munis d'interface de communication sans fil. C'est à travers les SSMs que les unités mobiles vont pouvoir se connecter au réseau et communiquer avec les autres sites (fixes et mobiles). [Ach95, Bad95]

Une unité mobile ne peut communiquer directement avec une SSM (et vice versa) que si elle se trouve physiquement à l'intérieur d'une aire géographique, appelée cellule[2], prédéfinie autour de la SSM [BAI94b, Ach95]. A chaque SSM correspond donc une cellule à l'intérieur de laquelle les unités mobiles peuvent émettre et recevoir des messages [Bad95].

A un instant donné, une unité mobile ne peut appartenir (logiquement) qu'à une seule cellule. Cette dernière représentera la position courante de l'unité mobile dans l'environnement mobile. Il s'ensuit qu'une migration d'une unité mobile aura lieu lorsque l'unité mobile se déplace de sa cellule courante vers une cellule voisine. [Ach95]

Les unités mobiles qui se trouvent dans une même cellule[3] sont considérées locales à la SSM définissant cette cellule et partagent la bande passante que la SSM offre. [FZ94]

Chaque SSM forme avec les unités mobiles qui lui sont locales un réseau sans fil (Wireless Network). Par conséquent, le modèle de système distribué intégrant des sites mobiles et qui a tendance à se généraliser est composé d'un réseau statique augmenté de réseaux sans fil; où chaque réseau sans fil est attaché à une seule SSM. [BAI94a, BAI94b]

---

[1] Une SSM est appelée aussi Station de Base (SB). [Bad95]

[2] Une cellule détermine une zone où doivent se trouver les unités mobiles pour pouvoir profiter des capacités de la SSM leur permettant de se connecter au réseau et de communiquer avec les autres sites (fixes ou mobiles).

[3] On dit aussi: les unités mobiles connectées à une même SSM.

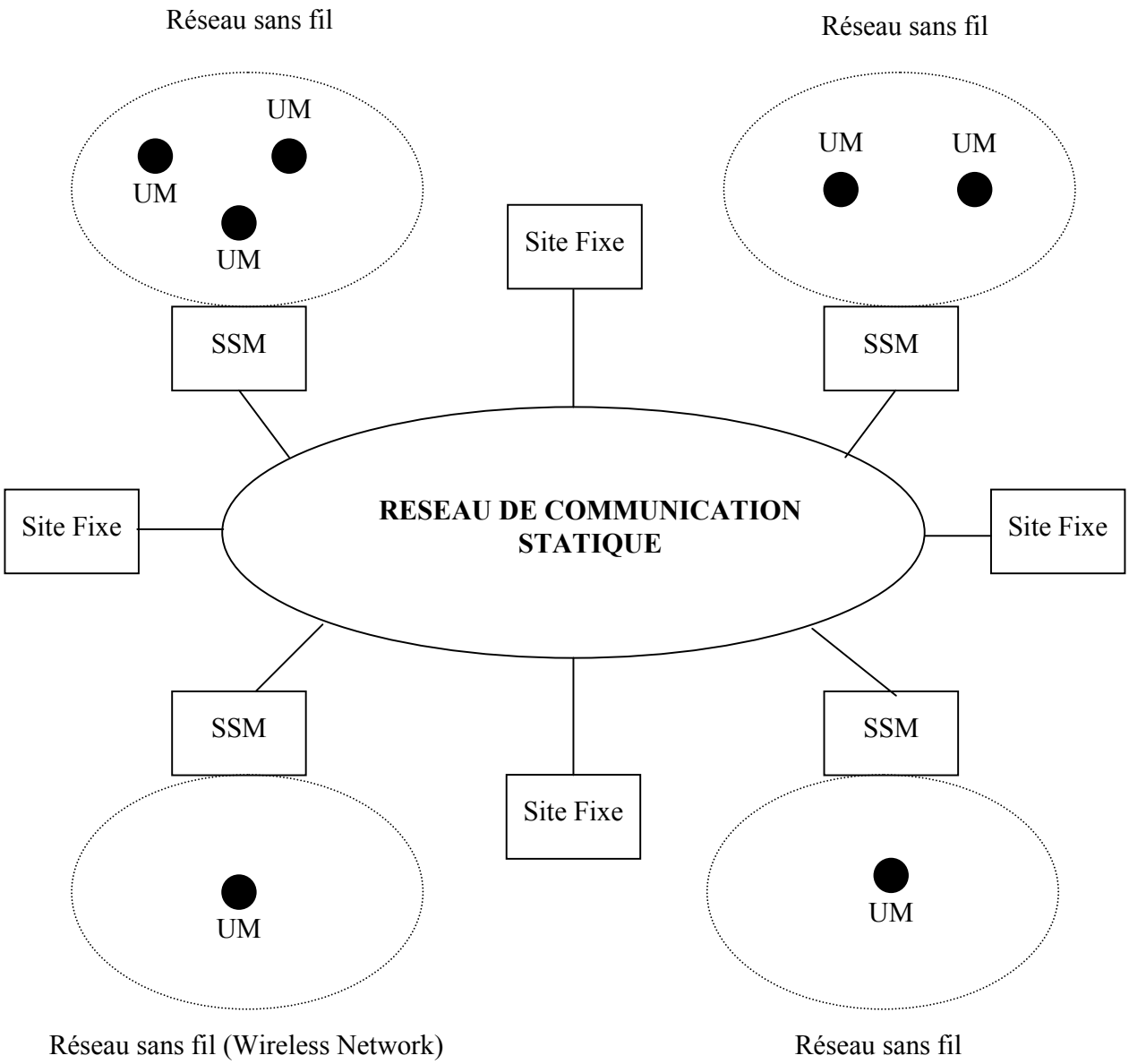


Figure I-1: Modèle de système distribué avec sites mobiles

Alors que les sites fixes sont interconnectés entre eux à travers un réseau de communication statique, généralement fiable et d'un débit élevé, les unités mobiles sont connectées aux SSM à travers des liaisons sans fil qui sont moins fiables et de faible bande passante réduisant sévèrement les informations échangées. [Bad95]

2.1. LA LIAISON SANS FIL

Si un réseau filaire assure, avec un délai de transmission arbitraire, un échange de messages fiable et séquentiel entre chaque couple de sites fixes, le réseau sans fil assure la

délivrance en FIFO[1] des messages échangés à l'intérieur d'une cellule entre une SSM et une unité mobile locale. [Ach95, BAI94a]

L'échange de messages entre une SSM et une unité mobile locale, s'effectue à travers deux canaux de communication: un canal (SSM-vers-UM) qui transmet les messages de la SSM vers l'unité mobile et un autre canal (UM-vers-SSM) qui transmet les messages de l'unité mobile vers la SSM.

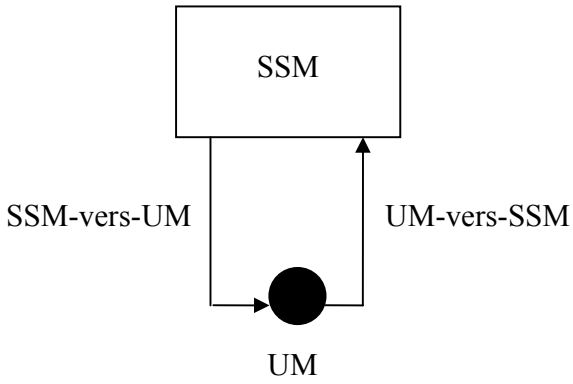


Figure I-2: Canaux de communication entre une SSM et une UM locale

Si une unité mobile ne quitte pas sa cellule courante, elle recevra alors (et dans leur ordre d'émission) tous les messages qui lui seront envoyés par la SSM.

2.2. COMMUNICATION ENTRE DEUX UNITES MOBILES

Soient U1 et U2 deux unités mobiles quelconques, et soient S1 et S2 les SSMs auxquelles U1 et U2 sont connectées respectivement. L'émission d'un message *m* de U1 vers U2 s'effectue alors comme suit [Bad95]: (i) U1 envoie *m* vers S1 à travers le canal U1-vers-SSM, (ii) S1 envoie ensuite, à travers le réseau fixe, le message *m* vers S2 et (iii) enfin, S2 délivre *m* à U2 via le canal SSM-vers-U2.

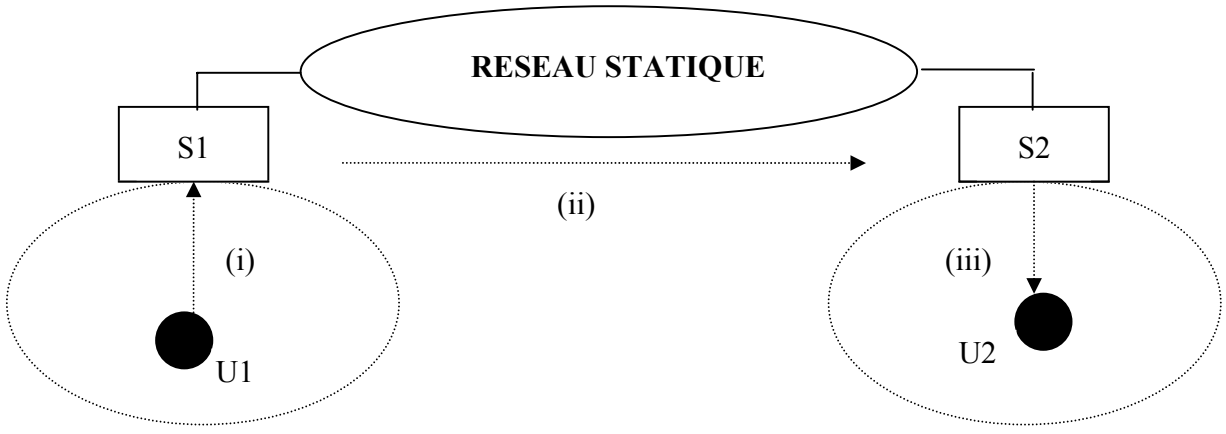


Figure I-3: Communication d'un message entre deux UMs

[1] FIFO: First In First Out

Puisqu'une unité mobile change de position à chaque migration, et puisque sa position courante n'est pas universellement connue dans le réseau, S1 a donc besoin de localiser d'abord U2 avant d'envoyer le message *m* à la SSM (S2) à laquelle U2 est connectée. Ceci constitue le problème essentiel que doivent régler les protocoles de routage de la couche réseau en présence de sites mobiles. [Bad95]

2.3. MIGRATION D'UNE UNITE MOBILE

La migration d'une unité mobile se traduit par son déplacement de sa cellule courante vers une autre cellule adjacente. Ce phénomène quand il se produit, doit être détecté par l'unité mobile. Une solution possible est le protocole *Beacon*. Dans ce protocole, chaque SSM indique sa présence aux unités mobiles qui se trouvent dans sa cellule en diffusant périodiquement son identité à l'aide d'un signal particulier appelé radiobalise (ou Beacon). [BAI94a, Ach95]
Une unité mobile détecte qu'elle s'est déplacée vers une cellule adjacente lorsqu'elle reçoit une radiobalise d'une nouvelle SSM et ne reçoit pas de radiobalise provenant de sa précédente SSM; ou bien, lorsqu'elle reçoit deux radiobalises: l'une provenant de sa précédente SSM et l'autre provenant d'une nouvelle SSM, tel que le signal provenant de la nouvelle SSM soit le plus puissant. [Ach95]

Rappelons que si une unité mobile ne quitte pas sa cellule courante, alors tous les messages qui lui sont envoyés par la SSM seront reçus, et dans leur ordre d'émission. Cependant, puisque l'unité mobile peut migrer à tout moment, la séquence de messages qu'elle recevra sera donc un préfixe de la séquence envoyée par la SSM, et une délivrance éventuelle des messages non reçus n'est pas garantie. En d'autres termes, si m1, m2, ..., mS représente la séquence de messages envoyés par la SSM vers une unité mobile locale, alors la séquence que recevra cette unité avant de migrer sera de la forme: m1, m2, ..., mR; tel que R<=S. [BAI94a, BAI94b]
Il est, par conséquent, nécessaire de s'assurer, lorsqu'une unité mobile se déplace d'une cellule à une autre, que les canaux de communication entre l'unité mobile et sa précédente SSM soient correctement purgés (C'est à dire, en Fifo). Ceci afin de conserver la délivrance et la séquentialité des messages échangés, indépendamment des déplacements de l'unité mobile entre les cellules [Ach95, Bad95, Sai96]. Une solution possible peut consister à numéroter les messages échangés ce qui permettra de connaître, après une migration, les messages qui ont été envoyés par la SSM et non reçus par l'unité mobile (et vice versa).

Quand une unité mobile détecte qu'elle a migré vers une nouvelle cellule, elle doit (si elle est aussi dans sa cellule précédente) envoyer, à travers le canal UM-vers-SSM de sa cellule précédente, le message final *quitter(r)*[1] , tel que *r* spécifie le numéro du dernier message reçu par l'unité mobile via le canal SSM-vers-UM. L'unité mobile n'enverra et ne recevra, par la suite, aucun autre message vers (ou de) sa précédente SSM.
Chaque SSM maintient une liste indiquant les unités mobiles qui lui sont locales. A la réception du message *quitter(r)*, la SSM supprime de sa liste l'unité mobile émettrice du message. L'unité mobile envoie ensuite, à l'intérieur de sa nouvelle cellule, le message *join(identité-de-l'UM, identité-de-la-précédente-SSM)* vers sa nouvelle SSM. Ceci déclenche, si nécessaire, une

[1] Notons que le message quitter(r) doit être *logiquement* envoyé avant que l'unité mobile ne se connecte (avec le message JOIN) à la nouvelle SSM; cependant, si les deux cellules (précédente et suivante) sont disjointes, l'unité mobile informera sa précédente SSM du numéro du dernier message reçu, par l'intermédiaire de sa nouvelle SSM.

procédure *HAND-OFF* qui permettra à la nouvelle SSM de poursuivre, entre autres, l'échange de messages envoyés et non reçus entre l'unité mobile et sa précédente SSM.

La procédure HAND-OFF

Certaines applications faisant intervenir des sites mobiles peuvent nécessiter une procédure Hand-off. En effet, une SSM peut maintenir des données relatives à la participation d'une unité mobile locale dans une application donnée. Pour cela, lorsque l'unité mobile se déplace vers une nouvelle cellule, les données qui lui sont associées doivent être transférées (HAND OVER) de l'ancienne SSM vers la nouvelle SSM afin que le fonctionnement de l'unité mobile ne soit pas affecté. Le contenu des données transférées est spécifique à l'application. [BAI94b]
La procédure Hand-off doit être transparente à l'utilisateur et ne doit provoquer aucune défaillance de connexion avec le réseau. Elle est déclenchée suite à une migration d'une unité mobile (quand ceci est nécessaire). [Sai96]

Fonctionnement de la procédure HAND-OFF

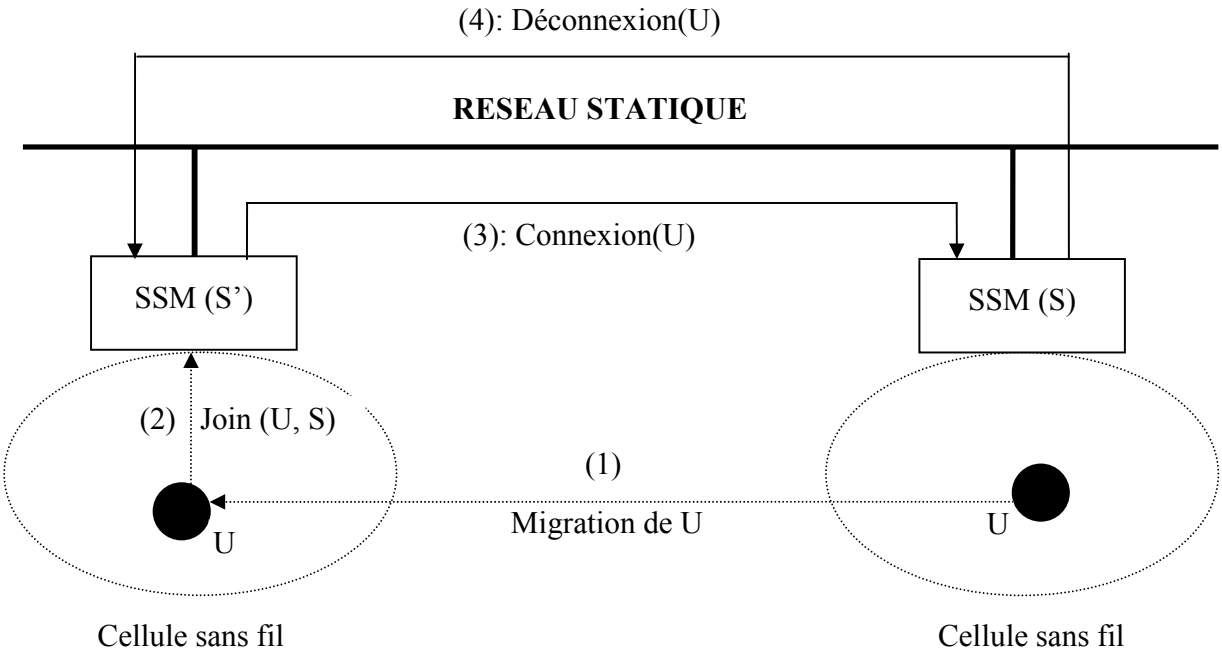


Figure I-4: Schéma de fonctionnement de la procédure Hand-off

Soit une unité mobile (U) migrant de la cellule (S) vers la cellule (S'). La procédure Hand-off se déroule comme suit: [Bad95]

- L'unité mobile U envoi le message join(U, S) à sa nouvelle SSM (S').
- S' accuse la réception du message à U, et envoie le message connexion(U) à la précédente SSM de U (qui est S) pour l'informer de la nouvelle localisation de U.

- S supprime U de sa liste d'unités mobiles locales[1], et renvoie à S' le message déconnexion(U) qui contient, entre autres, les informations d'état de U.
- A la réception du message déconnexion(U), S' ajoute U à sa liste d'unités mobiles locales et reprend, entre autres, l'échange de messages envoyés et non reçus entre l'unité mobile et sa précédente SSM (S).

2.4. PANNE D'UNE SSM

Pour faire partie du réseau, une unité mobile doit se connecter à une SSM. Il en résulte qu'une panne intervenant au niveau de la SSM rendra l'unité mobile inaccessible. Elle sera incapable de communiquer avec le reste du réseau et vice versa.
Ce problème peut être résolu en ayant des cellules qui se recouvrent. L'unité mobile pourra, ainsi, se connecter à une autre SSM dont la cellule recouvre sa position physique. En absence de recouvrement de cellules, l'unité mobile doit se déplacer physiquement vers une autre cellule si sa SSM courante tombe en panne. [Ach95, Sai96]

2.5. DECONNEXION D'UNE UNITE MOBILE

Pour conserver leurs batteries, les unités mobiles se déconnectent souvent du réseau. La déconnexion peut être considérée similaire à la migration; cependant, il y a une différence significative entre eux. En effet, bien que la migration soit une opération asynchrone (L'intervalle de temps entre le moment où une unité mobile quitte sa cellule courante et son apparition dans une autre cellule voisine n'est pas borné), l'unité mobile qui quitte sa cellule finit toujours par réapparaître dans le système (éventuellement dans une autre cellule); par contre, quand une unité mobile se déconnecte, rien ne garantit que plus tard elle va se reconnecter au système [BAI94b].
La déconnexion est également différente de la panne par le fait qu'elle est prévisible [BAI94b, Ach95, Bag96]. En effet, il est, tout à fait, possible à l'unité mobile de prévenir le système d'une éventuelle déconnexion et, par conséquent, d'exécuter, si nécessaire, un protocole de déconnexion spécifique à l'application [BAI94b, Bad95, Bag96]. Ce protocole permet à l'unité mobile de télécharger à son niveau les données dont elle a besoin afin d'assurer son fonctionnement normal indépendamment du reste des participants, et de répercuter ultérieurement (après sa reconnexion) les modifications qu'elle aura fait sur ces données. Il lui permet aussi de transmettre au reste du réseau, les données et les informations d'état que peut nécessiter la poursuite de l'exécution de l'application en son absence [Bad95].

3. CARACTERISTIQUES DES RESEAUX SANS FIL

Les réseaux sans fil présentent des caractéristiques particulières qui contrastent avec ce que nous avons l'habitude de rencontrer dans les architectures des réseaux fixes [FZ94]. D'une part, le médium de communication sans fil fournit de faibles débits avec des possibilités de déconnexions fréquentes. D'autre part, les unités mobiles doivent être facilement transportables, donc légères et de petite taille. De ce fait, les limitations sont nombreuses, tant du point de vue énergétique, qu'en matière de stockage des informations ou de puissance de calcul.
Dans ce qui suit, seront présentées, avec plus de détails, les caractéristiques des réseaux sans fil.

[1] Il s'agit ici de deux cellules disjointes; par conséquent, U n'a pas envoyé auparavant le message quitter(r). r (qui est le numéro du dernier message envoyé par S et reçu par U) est communiqué à S à travers le message connexion(U).

3.1. LES TYPES DE RESEAU DE COMMUNICATION SANS FIL

Les réseaux informationnels de demain dits PCN (Personal Communication Network) intégreront une large variété de services (voix, données, multimédia) offerts aux usagers indépendamment de leur position géographique. L'architecture générale de ces réseaux, bien qu'encore en débat, sera construite autour des infrastructures déjà existantes telles que: les réseaux téléphoniques cellulaires (à l'avenir microcellulaires) reliés au réseau téléphonique public, les réseaux locaux traditionnels tels que Ethernet, étendus à la communication sans fil et reliés à des réseaux plus étendus de type LAN, WAN, Internet, etc., et enfin, les architectures orientées vers des services spécialisés fournis par diffusion sur des portions d'ondes radio en modulation de fréquence ou par satellite à des usagers munis de terminaux spéciaux [PB93, IB94, Bad95].
Le tableau suivant présente des exemples de débits suivant la technologie employée: [Bad95]

Technologie	Débit
Infrarouge	11 Mbps
Radio	2 Mbps
Téléphone cellulaire	9 à 14 Kbps

Tableau I-1: Debit de transmission de technologies différentes

La même unité mobile peut, en principe, utiliser les trois types de technologies à différents moments. Par exemple, en se déplaçant de l'intérieur d'un bâtiment où elle interagit avec un réseau local pourvu d'une interface de communication sans fil (infrarouge, par exemple), à l'extérieur du bâtiment où elle interagit avec le réseau téléphonique cellulaire [IB94, Bad95].

3.2. LES UNITES MOBILES

Il est prévu que l'émergence d'un marché massif du calcul mobile, que la plupart des auteurs situent autour de la fin de cette décennie, verra le développement de diverses configurations d'unités mobiles plus ou moins évoluées [Bad95]. Ces configurations diverses permettent d'envisager deux cas extrêmes d'utilisation des unités mobiles dans les systèmes distribués: [Duc92, Nar94, Bad95]

- Soit qu'elles sont utilisées comme de simples terminaux dépourvus de capacité de calcul et dépendant d'un site fixe, pas forcement toujours le même. On parle dans ce cas, d'un modèle avec accès distant (Remote access model), où l'exécution de l'application s'effectue sur le réseau fixe.
- Soit qu'elles sont utilisées comme de véritables stations de travail avec suffisamment de puissance de calcul et d'espace mémoire pour effectuer des traitements de manière locale. On parle dans ce cas, d'un modèle avec cache (Caching model), où l'usager peut télécharger sur l'unité mobile des logiciels exécutables.

Les configurations d'unités mobiles existantes se décomposent alors en deux classes essentielles: [FZ94, Bad95]

- Les ordinateurs de poche (*Palmtops*), avec une fréquence d'horloge qui oscille entre 8 et 20 Mhz, une RAM de 1 Moctets à 4 Moctets et une ROM de 512 Koctets à 2 Moctets. Ils sont généralement dépourvus de disque; et
- Les ordinateurs portatifs (*Laptops*), avec une puissance comparable à celle d'un ordinateur personnel (PC) de bureau, avec une capacité mémoire de 2 à 8 Moctets et une fréquence d'horloge de 15 à 20 Mhz.

Des exemples d'unités mobiles sont donnés dans le tableau suivant: [IB94]

Nom	CPU	Mhz	RAM	Durée de la batterie (heure)	Dimensions LxLxH (cm)	Poids (Kg)	Ecran
HP-95LX	NEC V20H	5.37	512 KB/1M	50	16x8.6x2.5	0.308	40x16
Memorex	80C88	7.16	640 KB	50	23.7x11x2.9	0.581	CGA
Sharp PC-3	80C88	10	½ MB	50	22.3x11.2x2.54	0.558	CGA
IBM Tablet	80386 SX	20	4/6 MB	3	31.1x23.3x3	2.779	VGA
Infolio	MC68331	16	6 MB	12	28.4x23.5x3.1	1.543	VGA
NCR	80386 SL	25	8 MB	3	25.4x30.4x2.5	1.816	VGA

Tableau I-2: Caractéristiques physiques des UMs

Les unités mobiles accèdent à un réseau statique à travers des liaisons de communication sans fil; cependant, rien ne les empêche, lorsqu'elles restent stationnaires (tel que lors d'une réunion), de se connecter physiquement au réseau (liaison filaire) afin de bénéficier d'une connexion meilleure et moins chère. [FZ94]

3.3. LES MODES DE FONCTIONNEMENT D'UNE UNITE MOBILE

Du fait des diverses limitations dont souffrent les unités mobiles, et des caractéristiques intrinsèques du médium de communication sans fil, les unités mobiles disposent de modes de fonctionnement qui leur sont propres. En environnement fixe, nous disposons, en effet, de deux modes: connectée ou déconnectée; en environnement mobile, il en existe quatre: fortement connectée, partiellement connectée, veille et déconnectée, qui dépendent, entre autres, de la largeur de la bande passante attribuée aux unités mobiles. [Bag96]
Le passage d'un mode de fonctionnement à un autre peut être une action volontaire, pour des raisons d'économie d'énergie, ou involontaire, du fait de perturbations ou de surcharge.

La déconnexion d'une unité mobile du reste du réseau au cours de l'exécution d'une application à laquelle elle participe peut entraîner la suspension de l'application jusqu'à ce que l'unité mobile se reconnecte. Cependant, puisque la déconnexion est, rappelons-le, prévisible, l'unité mobile peut alors exécuter un protocole de déconnexion spécifique à l'application avant de se détacher physiquement du réseau. Ce protocole permettra à l'unité mobile de continuer à fonctionner indépendamment des autres sites, et à des sites éventuels d'être informés de cette déconnexion. [Bad95]

Contrairement à la déconnexion par laquelle l'unité mobile devient inaccessible jusqu'à sa nouvelle reconnexion (sous sa propre initiative), une unité mobile fonctionnant en mode veille reste accessible au reste du système et peut reprendre son mode normal dès la réception d'un message provenant du réseau. Une unité mobile se met en mode veille pour économiser de l'énergie lorsque sa participation à l'activité du système n'est pas nécessaire. Dans ce cas, aucun calcul n'est effectué, la vitesse du processeur est réduite et l'unité mobile se met en attente de réception d'un message. [BAI93, Bad95]
L'intérêt du mode veille découle du fait que certains algorithmes distribués, particulièrement ceux basés sur une topologie logique en anneau, impliquent l'ensemble des processus au calcul même s'ils ne sont pas directement concernés par son résultat. Ainsi, être en mode veille permet aux unités mobiles qui ne sont pas nécessaires au calcul de bénéficier du gain d'énergie offert par ce mode de fonctionnement. [BAI93, Bad95]

Une unité mobile peut aussi basculer en mode partiellement connectée en exécutant un protocole spécifique. Dans ce mode, toutes les communications avec le réseau doivent être limitées. [BAI93, Bad95]

La figure suivante résume les différents modes de fonctionnement d'une unité mobile [PB93]:

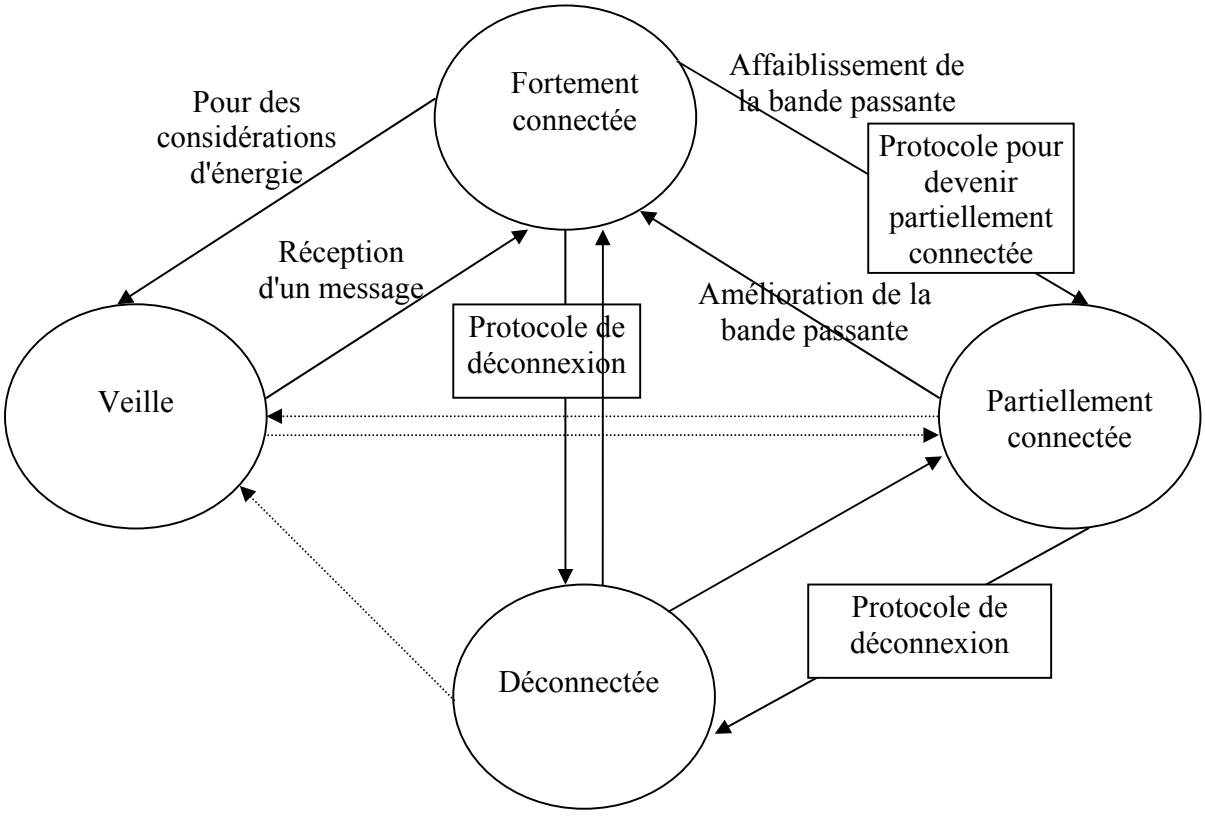


Figure I-5: Etats de fonctionnement d'une UM

3.4. LE MEDIUM DE COMMUNICATION SANS FIL

Le médium de communication sans fil supporte physiquement la diffusion; ainsi, une SSM peut émettre un message à toutes les unités mobiles localisées dans sa cellule en une seule

opération de transfert. Le coût du transfert est indépendant du nombre d'unités à l'intérieur de la cellule. [Bad95]
Notons par ailleurs que l'émission d'un message par une unité mobile consomme, en général, deux fois plus d'énergie que la réception d'un message de même taille [IB94]. Il est donc nécessaire que la communication dans les réseaux sans fil soit asymétrique: la SSM doit recourir au mieux à la diffusion pour communiquer avec les unités mobiles localisées dans sa cellule, par contre, les unités mobiles doivent minimiser les transferts de messages vers les SSM auxquelles elles sont connectées. [Bad95]

3.5. FIABILITE DE LA COMMUNICATION SANS FIL

La communication sans fil est moins fiable que la communication dans les réseaux filaires. La propagation du signal subit des perturbations (erreurs de transfert, microcoupures, timeout) dues à l'environnement, qui altèrent l'information transférée. Il s'ensuit alors un accroissement du délais de transit des messages à cause de l'augmentation du nombre de retransmissions.
En plus, la connexion peut être rompue ou altérée par la mobilité des sites. Un usager peut sortir de la zone de réception ou entrer dans une zone de haute interférence. Le nombre d'unités mobiles dans une même cellule peut, par exemple lors d'un rassemblement populaire, entraîner une surcharge du réseau. L'une aussi des limites de la communication sans fil découle de la relative faiblesse de la bande passante des technologies utilisées. Cette bande passante sera, rappelons-le, évidemment partagée entre tous les utilisateurs existant dans une même cellule.

Pour augmenter la capacité de service des réseaux sans fil, deux techniques sont utilisées: la technique de recouvrement des cellules sur différentes longueurs d'ondes et celle qui réduit la portée du signal pour avoir plus de cellules mais de rayon moindre. Cette dernière est généralement plus utilisée à cause de sa faible consommation d'énergie et d'une meilleure qualité du signal. [Bad95]

4. PROBLEMES RENCONTRES DANS UN ENVIRONNEMENT MOBILE

L'impact de la mobilité et de la portabilité sur la conception des systèmes peut avoir, en général, l'ampleur de l'impact qu'a eu la distribution dans le passé. La distribution a affecté l'exécution des transactions, le traitement des requêtes, le recouvrement après panne, les structures physiques des données (y compris le placement des données), l'intégrité, la sécurité et plusieurs autres aspects du système, tels que la conception du système d'exploitation. Par contre, la distribution n'a pas eu d'impact profond sur la conception des langages de requêtes, la conception logique des bases de données ou la modélisation des données. D'après [IB94], le passage d'un environnement statique à un environnement mobile a une influence similaire à celle qu'a provoqué le passage de l'environnement centralisé à l'environnement distribué: au lieu de parler de "Centralisé ou Distribué?", on parlera plutôt de "Statique ou Mobile?".

La mobilité et la portabilité des sites introduisent, en fait, de nouveaux problèmes non encore considérés dans la construction des systèmes distribués classiques. Ces problèmes affectent et le réseau statique et les réseaux sans fil qui lui sont rattachés. En effet, les déplacements multiples des unités mobiles impliquent une modification continuelle de la topologie physique du réseau; en plus, de nombreuses informations statiques en environnement fixe (les adresses des unités mobiles, les serveurs les plus proches, etc.) deviennent variables; ajoutons à ceci, les caractéristiques physiques des unités mobiles et des liaisons sans fil.

Différentes questions, par conséquent, se posent: Comment seront localisées les unités mobiles? Et comment leur mobilité affectera la distribution des données, le traitement des requêtes et l'exécution des transactions? Quel sera le rôle du médium de communication sans fil dans la distribution des informations? Et quelle sera l'influence de la durée de vie limitée des batteries sur l'accès aux données à partir des unités mobiles? etc.

Il en découle alors que dans un environnement mobile, il est nécessaire de nommer les unités mobiles, de les localiser, de leur permettre (grâce à un mécanisme de reconfiguration) de découvrir les ressources (imprimantes, serveurs de noms, etc.) des réseaux locaux qu'elles visitent, de router les messages de et vers les unités mobiles, de restructurer les algorithmes distribués, de revoir la gestion des données (requêtes, transactions, etc.), etc.

4.1. LES ALGORITHMES DISTRIBUES

La conception des algorithmes (et des protocoles) distribués était basée traditionnellement sur l'hypothèse que le réseau est constitué de sites fixes, c'est-à-dire, qui ne changent pas de localisation. Par conséquent, en absence de pannes de sites et/ou de liens de communication, la connectivité entre les sites du réseau ne change pas. Les algorithmes distribués exploitent cette caractéristique pour résoudre les problèmes de communication et de synchronisation. Ils se basent, en fait, sur un modèle constitué d'un ensemble de participants (ou processus) s'exécutant sur des sites fixes et reliés entre eux par des canaux logiques (point-à-point) pour former une structure de communication logique[1] (anneau, arbre, etc.). Chaque canal de communication logique peut comprendre plusieurs liens physiques; ces liens physiques et les sites aux extrémités des canaux logiques ne changent pas avec le temps. [BAI94b]
Ce modèle constitue, dans un environnement fixe, la base de la conception des algorithmes distribués; cependant, il ne capture pas les caractéristiques et les contraintes de l'environnement mobile [BAI94b]. Dans ce dernier, les unités mobiles se déplacent et se déconnectent fréquemment changeant (à chaque migration, déconnexion ou reconnexion) la connectivité physique du réseau. Ainsi, les structures logiques exploitées par plusieurs algorithmes distribués ne peuvent pas être construites statiquement dans le réseau sur un ensemble de connexions physiques [BAI94b]. Les algorithmes distribués ont donc besoin d'être restructurés pour pouvoir fonctionner dans un environnement mobile.

[1] Le but d'une structure de communication logique est d'établir un certain degré d'ordre et de précédence à la communication entre les participants dans un algorithme distribué [Ach95]. Les messages échangés dans cette structure suivront uniquement les chemins (ou canaux) logiques.

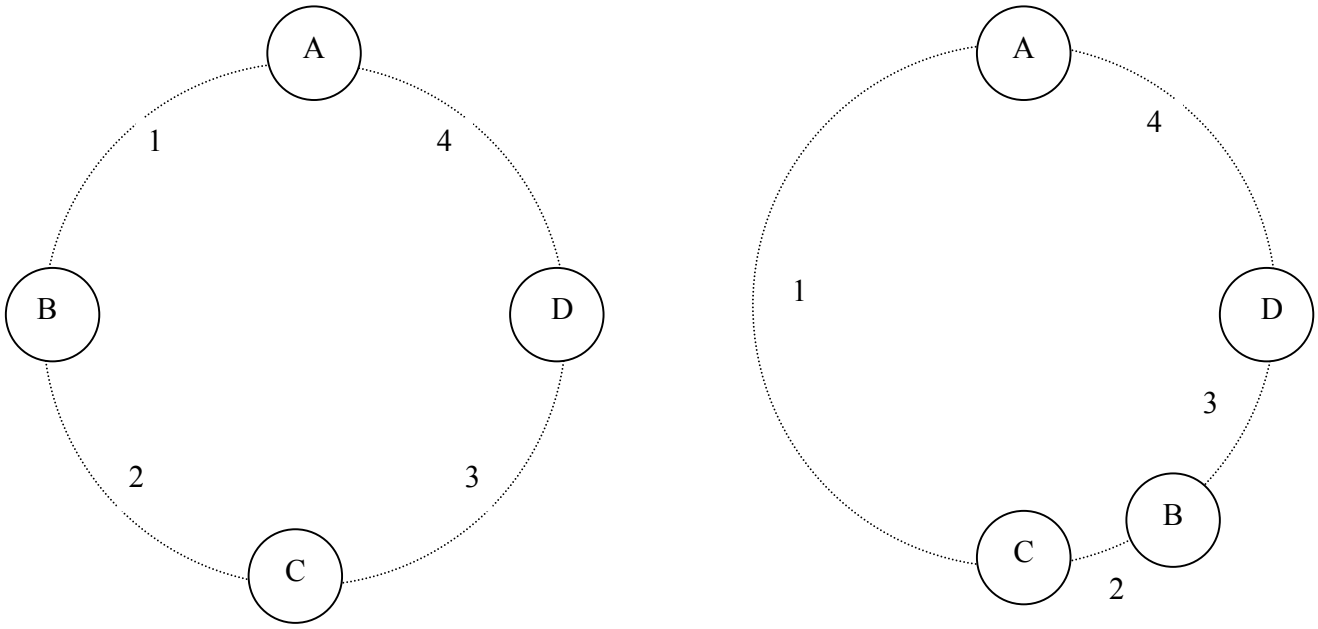


Figure I-6: Impact de la mobilité sur la connectivité physique du réseau

Dans [PB93, BAI94a, BAI94b, Ach95], les auteurs cherchent à réaliser un découplage entre la mobilité des sites et la construction des algorithmes distribués. L'idée consiste à structurer les algorithmes distribués de manière à ce que la majeure partie des communications et du calcul induit par l'algorithme soit prise en charge par le réseau statique. Ceci permettra d'éviter la localisation des unités mobiles participantes; ce qui minimisera le coût de la recherche induit par l'algorithme. En plus, le nombre des opérations exécutées au niveau des unités mobiles sera minimisé réduisant, par conséquent, la consommation d'énergie qui constitue une ressource critique pour les unités mobiles. [BAI94b]

4.2. RESEAU ET COMMUNICATION

Adressage et routage

La plupart des protocoles de routage sont basés sur l'hypothèse que les sites sont statiques. Dans Internet, par exemple, une adresse IP (Internet Protocol) d'un site donné dépend du réseau auquel ce site est connecté; et le routage des paquets dépend du numéro du réseau. De tels schémas d'adressage sont évidemment inadaptés dans un environnement mobile. Pour cela, des travaux sont menés, tels que l'extension du protocole IP, afin de prendre en charge la mobilité des sites et optimiser le coût de recherche nécessaire à la localisation d'un site mobile. [Bad95]

Compression

Il est intéressant d'utiliser dans un environnement mobile, la compression et la décompression des paquets de données constituant les messages émis de ou vers les unités mobiles. En effet, ceci permet de conserver la bande passante (qui est une ressource critique) et d'utiliser moins de mémoire; cependant, un calcul additionnel est nécessaire au niveau des

unités mobiles pour effectuer la compression et la décompression des paquets, ce qui consommera davantage d'énergie. [PB93]

Diffusion sélective

La diffusion sélective permet l'émission d'un message vers plusieurs destinataires [Bad95]. La délivrance d'un message multidestinataire, exactement une fois, à un site mobile est particulièrement difficile pour les raisons suivantes: [PB93, Bad95]

- La localisation d'une unité mobile peut ne pas être connue par l'émetteur et peut, en plus, changer au cours du transfert du message.
- Les délais de transfert des messages entre les sites fixes étant arbitraires, la réception par deux SSMs différentes d'un message diffusé peut éventuellement avoir lieu à des instants différents.

Il est donc possible qu'un message diffusé dans une cellule puisse ne pas être reçu par une unité mobile destinataire, car son entrée dans la cellule a eu lieu après la diffusion du message ou bien sa sortie de celle-ci a eu lieu avant la diffusion. D'autre part, une unité mobile peut également recevoir plusieurs copies d'un même message, issues de SSMs différentes. [Bad95]

Le schéma suivant montre l'asynchronisme induit par la mobilité des sites. En effet, SSM2 a diffusé un message *m* à l'attention des unités mobiles UM1, UM2, UM3 et UM4. Cependant et dû à la mobilité, UM4 a reçu *m* une seule fois, UM2 l'a reçu deux fois, alors que UM1 et UM3 ne l'ont pas reçu.

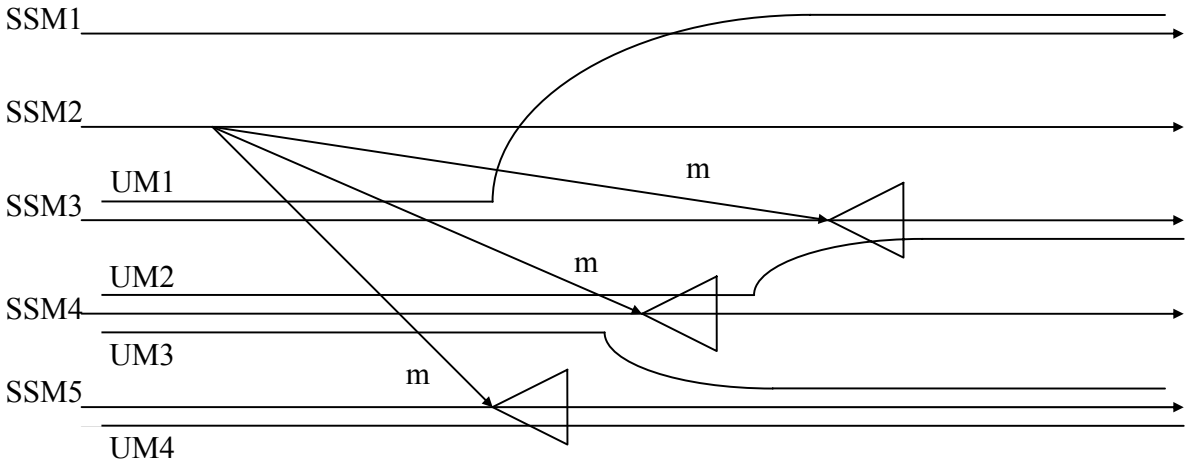


Figure I-7: Asynchronisme induit par la mobilité

4.3. LES SYSTEMES D'EXPLOITATION

Les systèmes d'exploitation doivent prendre en compte la contrainte d'énergie imposée par la durée de vie limitée des batteries utilisées dans les unités mobiles. Ils doivent considérer aussi les multiples et différentes technologies de communication sans fil, qui présentent de larges variations que ce soit dans la latence, dans la bande passante, dans les caractéristiques de la connexion ou dans le coût d'utilisation. Ils doivent, en plus, permettre aux unités mobiles d'être capable de découvrir et d'interagir avec les services et les équipements disponibles dans les réseaux locaux qu'elles visitent pendant leurs déplacements. [PB93]

Les systèmes d'exploitation doivent également garantir la sécurité du système, la protection des informations relatives aux utilisateurs mobiles, et permettre la manipulation des fichiers même lors de déconnexions des unités mobiles.

La sécurité

La sécurité est un aspect très important pour tous les systèmes et donc naturellement aussi dans un environnement mobile. En fait: [PB93]

- Quand une unité mobile se déplace vers une nouvelle cellule, elle a besoin pour continuer son fonctionnement d'utiliser les ressources de la nouvelle SSM; néanmoins, l'unité mobile doit être protégée des accès non autorisés que peut effectuer la SSM (et vice versa).
- D'autre part, la connaissance de certaines informations concernant un autre utilisateur mobile (telles que sa localisation) est, dans certains cas, utile (par exemple, pour contacter des collègues, pour faire orienter des appels téléphoniques à un utilisateur mobile, etc.); cependant, de telles informations doivent être suffisamment protégées contre l'usage malintentionné (par exemple, un cambrioleur qui cherche à connaître si les habitants d'une maison sont actuellement loin).
- En plus, rendre des ordinateurs transportables permet de les déplacer facilement; cependant, ceci augmente leur risque d'être endommagés physiquement, perdus, volés ou exploités sans autorisation. Pour cela, les unités mobiles ne constituent pas un support de stockage sûr: la sécurité des données peut être violée ou les données peuvent être complètement perdues. Pour minimiser les dégâts, il est préférable de réduire les données essentielles gardées sur les unités mobiles. En effet, une unité mobile qui sert seulement comme un terminal portable est moins exposée à la perte de données qu'un ordinateur puissant.

Les systèmes de fichiers

La motivation primaire derrière la conception de systèmes de fichiers pour des utilisateurs mobiles est de supporter les opérations de déconnexion [Ach95]. Une autre motivation consiste à conserver la bande passante des liaisons sans fil.
Avant de se déconnecter, l'unité mobile ramène à partir des serveurs statiques des copies de fichiers dont elle a besoin et les place dans son cache local pour pouvoir continuer à fonctionner (de manière indépendante) après sa déconnexion du réseau. En se reconnectant plus tard, l'unité mobile répercutera sur les serveurs, les modifications qu'elle aura faites sur les fichiers ramenés.
Un exemple de système de fichiers conçu pour un environnement mobile est le système CODA[1] [Bad95].

4.4. GESTION DE DONNEES

Les requêtes

L'environnement mobile a introduit un nouveau type de requêtes qui manipulent des informations dépendantes de la localisation actuelle de l'unité mobile [PB93]; Par exemple, Quel est le Taxi le plus proche? Ou encore, quelle est l'autoroute la plus proche qui soit la moins encombrée? etc.

[1] Le lecteur intéressé par le système Coda peut consulter, à titre d'exemple, les références suivantes: [KS92] et [LS95].

Les informations qui dépendent de la localisation peuvent changer durant l'évaluation de la requête. En plus, elles peuvent être imprécises puisqu'il est difficile de maintenir ces informations continuellement à jour.

Les transactions distribuées mobiles

Une transaction distribuée mobile est une transaction distribuée dont certaines parties s'exécutent sur des sites statiques et d'autres sur des unités mobiles. Les unités mobiles sont connectées au réseau par intermittence et évoluent dans un environnement non fiable. En plus, la mobilité des unités participantes permet à des transactions en cours d'exécution d'accéder à des systèmes d'information hétérogènes; et, par conséquent, auront à manipuler des données imprécises ou dont la localisation change fréquemment. Finalement, le faible débit des communications sans fil et la latence du réseau contribuent à faire allonger d'avantage la durée de vie des transactions. [Bad95]

Dans les bases de données traditionnelles, les utilisateurs interagissent avec le système à travers des transactions atomiques, cohérentes, isolées et durables (propriétés ACID). Une transaction est atomique si toutes ses opérations sont exécutées ou bien aucune d'elles n'est exécutée, cohérente si son exécution maintient la base de données dans un état cohérent, isolée lorsque son exécution n'interfère pas avec celle d'aucune autre transaction et enfin durable, dans la mesure où ses résultats deviennent permanents après sa validation.
Le problème qui se pose est de définir l'influence des caractéristiques d'un environnement mobile sur ces propriétés et en déduire un modèle de transactions mobiles.
Pitoura et Bhargava utilisent dans [PB94, PB95] le modèle imbriqué ouvert (Open Nested Model) pour modéliser les transactions mobiles. Une transaction mobile est alors structurée en un ensemble de sous-transactions, tel que chaque sous-transaction est une transaction plate[1] ou une transaction imbriquée ouverte.

Rappelons qu'une unité mobile peut être dans l'un des quatre modes de fonctionnement: déconnectée, partiellement connectée, veille ou fortement connectée. Il en résulte qu'un utilisateur mobile qui dispose d'une copie d'une portion d'une base de données partagée peut acquérir différents niveaux de cohérence. Lorsque l'unité mobile est fortement connectée au réseau fixe, l'utilisateur peut vouloir reproduire sur son cache les valeurs courantes des objets locaux. Par contre, si l'unité mobile est faiblement connectée, l'utilisateur peut adopter une approche plus flexible de cohérence faible qui lui permet de disposer d'une copie approximative des données localisées sur le réseau fixe. Ainsi, à chaque type de connexion peut correspondre un niveau de cohérence du cache sur l'unité mobile: plus la connexion au réseau fixe est faible, plus la cohérence requise est faible.

La redistribution transparente du calcul

La distance entre un client mobile et le serveur d'informations n'est pas un paramètre fixe du coût du service. Il est donc nécessaire de reloger une partie du calcul, en cours d'exécution sur le site fixe, vers un autre site plus proche du client, afin d'améliorer le temps de service, équilibrer la charge du réseau ou remédier à une défaillance du serveur. [Bad95]

[1] Une transaction plate est une transaction constituée d'une séquence d'opérations (de lecture et d'écriture) délimitée par un seul début et un seul point de rejet ou de validation. Ce modèle de transaction, appelé aussi modèle linéaire, est utilisé dans les Systèmes de Gestion de Base de Données (SGBDs) classiques. [Bad95]

5. CONCLUSION

Les progrès récents dans la technologie des ordinateurs et des communications sans fil ont rendu le calcul mobile possible. Un environnement de calcul mobile est caractérisé par de fréquentes déconnexions, des limitations significatives des sources d'énergie (batteries) et du débit de transfert de l'information (dû au médium de communication sans fil), des restrictions sur les ressources utilisées (capacité disque, espace mémoire, vitesse du processeur) et des changements fréquents de localisation. [Bad95]
La puissance des batteries étant limitée, une unité mobile sera, évidemment, souvent déconnectée du réseau (ou même éteinte). Les activités déclenchées par cette unité seront séparées (ou même interrompues) par des périodes arbitraires de déconnexion. En plus, une unité mobile peut migrer et peut traverser (éventuellement) plusieurs cellules. De ce fait, une unité mobile peut se trouver très souvent (à chaque migration ou reconnexion) dans un environnement complètement nouveau (quelque part autour du système). Ces comportements doivent, cependant, être cachés à l'utilisateur mobile; ce dernier doit continuer d'observer le même environnement de calcul indépendamment de sa localisation courante.

L'environnement de calcul mobile est un nouveau paradigme de calcul qui s'étend plus rapidement et plus largement que tout autre paradigme existant [IB94]. En effet, il est prévu que dans le futur proche, des dizaines de millions d'utilisateurs posséderont un portable muni d'une interface de communication sans fil et accéderont à un réseau d'information mondial [PB93].

Le reste de ce document est consacré entièrement à la structuration des applications distribuées en environnement mobile. Nous adoptons comme modèle de structuration, l'approche d'*agents mobiles*, et nous verrons que cette approche résout automatiquement et parfaitement plusieurs problèmes posés par l'environnement mobile.

CHAPITRE II

L'APPROCHE D'AGENTS MOBILES DANS LA STRUCTURATION DES APPLICATIONS REPARTIES EN ENVIRONNEMENT MOBILE

1. INTRODUCTION

Les applications de traitement de l'information ont souvent exigé (et continuent à exiger) d'être de plus en plus distribuées sur de nombreux sites distants [PB95]. Les systèmes distribués qui les supportent sont souvent basés sur le modèle Client/Serveur, dans lequel on distingue deux entités: le fournisseur du service (ou serveur) et le consommateur du service (ou client) [Fiu94, Har95].

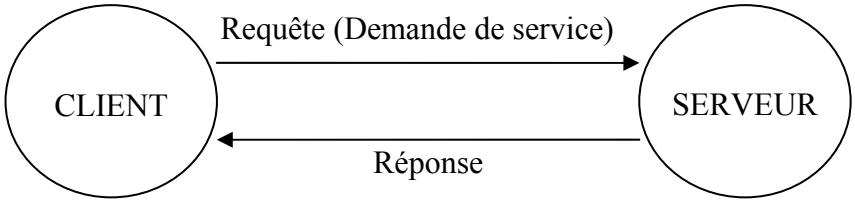


Figure II-1: Le modèle Client/Serveur.

Le client et le serveur résident généralement sur des sites distincts; par conséquent, les interactions entre les clients et les serveurs exigent souvent (pour s'accomplir) des communications à travers le réseau [Gra95b]. Ces communications ont été assurées par des méthodes très connues telles que: la communication par messages et l'appel de procédure à distance (ou RPC: Remote Procedure Call).

Le volume d'informations manipulées par les différentes applications n'a pas cessé de croître; et ce, à un rythme très accéléré dépassant largement le rythme des améliorations apportées aux réseaux [Har95]. La capacité de transport de ces derniers est devenue, peu à peu, considérablement faible devant le trafic important que peuvent engendrer les applications qui manipulent de très grands volumes d'informations distribuées. Pour cela, il était nécessaire de moderniser le modèle Client/Serveur afin de supporter plus efficacement les interactions entre les clients et les serveurs: en améliorant les méthodes de communication existantes, comme c'est le cas de SUPRA-RPC (SUbprogram PaRAmeters in Remote Procedure Call), ou en développant d'autres méthodes, telles que NCL (Network Command Language) et REV (Remote EValuation).

Ces dernières années, le volume d'informations disponibles aux applications a connu une très importante expansion avec l'information qui est devenue accessible à travers de grands réseaux, tels que: IBM Global Network, et particulièrement, Internet. D'autre part, les progrès récents de la technologie de communication sans fil ont permis à un ordinateur portable de se connecter au réseau à travers des liaisons sans fil, de communiquer avec le

reste du réseau et de se déplacer librement tout en restant connecté; ce qui a engendré un nouveau paradigme de calcul, appelé le calcul mobile.

L'abondance des informations disponibles, la capacité de transport limitée offerte par les réseaux de communication et récemment le calcul mobile ont rendu les modèles existants inadéquats pour supporter l'importance du trafic qui peut en résulter et la mise en œuvre des applications complexes [PB95]. Il était donc nécessaire de développer d'autres approches pouvant servir efficacement les besoins des utilisateurs et utiliser intelligemment les ressources du réseau. C'est ainsi que l'approche d'agents mobiles est apparue pour étendre et remplacer le modèle Client/Serveur [Gra95b, Har95].

Ce chapitre présente d'abord les modèles Client/Serveur, introduit ensuite l'approche d'agents mobiles en décrivant ses caractéristiques et ses avantages et discute enfin l'adéquation de l'approche d'agents mobiles à la structuration des applications distribuées, plus particulièrement, en environnement mobile.

2. LES MODELES CLIENT/SERVEUR

Deux types de modèle Client/Serveur existent [Gra95a]: Le modèle traditionnel et le modèle moderne.

2.1. LE MODELE CLIENT/SERVEUR TRADITIONNEL

Pour assurer les interactions Client/Serveur, les alternatives de communication dominantes sont [HCK95]: la messagerie, le datagramme simple, les sockets, l'appel de procédure à distance (RPC: Remote Procedure Call) et les conversations. Ces méthodes suivent soit des protocoles synchrones ou des protocoles asynchrones [HCK95].
La messagerie et le RPC, étant les méthodes de communications les plus populaires, seront, par conséquent, présentées.

2.1.1. LA MESSAGERIE

Un système de messagerie permet le passage de messages entre deux extrémités grâce à deux primitives: *Send* et *Receive*. La primitive *Send* permet d'envoyer un message vers un destinataire, et la primitive *Receive* permet au destinataire de recevoir un message arrivé. [Gra95a]

En utilisant la messagerie, le client (respectivement, le serveur) peut encapsuler sa requête (respectivement, sa réponse) dans un message qui sera transporté par le système de messagerie vers le serveur (respectivement, vers le client).

La messagerie est une méthode asynchrone: après que le client soumet son message au sous-système de messagerie et sauvegarde son état, il continue son exécution. A la réception du message, le serveur effectue le traitement correspondant puis envoie sa réponse au client. Lorsque le client reçoit la réponse, il restaure son état pour entreprendre l'action appropriée. [HCK95]

Un système de messagerie peut être rendu sûr, en lui ajoutant des facilités d'authentification et de cryptographie. La durée nécessaire à la réception d'un message augmentera; mais, cette augmentation sera cachée par l'asynchronisme de la communication. [HCK95]

La messagerie est une méthode puissante et flexible; cependant, le programmeur doit gérer des détails de bas niveau tels que: déterminer l'adresse réseau du serveur, associer les réponses reçues aux requêtes émises correspondantes et gérer les erreurs de communication [Gra95a, Gra95b]. Pour éviter au programmeur de tels détails, le RPC peut être utilisé.

2.1.2. APPEL DE PROCEDURE A DISTANCE

L'appel de procédure à distance (ou RPC: Remote Procedure Call) cache au programmeur tous les détails de bas niveau relatifs à la communication. Il permet au client d'invoquer les opérations (ou procédures) fournies par le serveur, en utilisant le mécanisme standard d'appel de procédure [Gra95a].

Le mécanisme traditionnel d'appel d'une procédure locale consiste à: [HCK95]

- Empiler les paramètres d'appel, les registres et l'adresse de retour; ensuite,
- Se brancher au point d'entrée de la procédure.

L'appel de procédure à distance, qui est une extension de ce mécanisme traditionnel, consiste à: [HCK95]

- ✓ Etablir un canal de communication entre le client et le serveur, et à
- ✓ Passer les paramètres d'appel à une routine d'interface qui les met dans une forme souhaitable à la communication, puis les envoie au serveur.
- ✓ Au niveau du serveur et à la réception des paquets RPC, ces derniers sont interprétés par une routine d'interface identique à celle utilisée par l'appelant. La routine reconstitue les paramètres d'appel, et les passe à la procédure. La procédure s'exécute et produit généralement un résultat qui est renvoyé au client.

L'implémentation la plus commune pour RPC est celle de [BN84]. Dans cette implémentation, les routines d'interface sont représentées par les procédures *STUBs* [Gra95a, Gra95b].

Le RPC, comme l'appel d'une procédure locale, est aussi synchrone: le client sauvegarde, lors de l'appel, son état complet et se suspend jusqu'à réception du résultat provenant du serveur. Cependant, le temps nécessaire au traitement des paramètres d'appel (mise sous forme souhaitable à la communication au niveau du client, transmission, extraction au niveau du serveur) fait que la durée d'un appel distant (qui est de l'ordre de quelques millisecondes, sans compter le temps d'exécution de la procédure elle-même) soit plus importante que la durée d'un appel local (qui est de l'ordre de quelques microsecondes). [HCK95]

L'appel de procédure à distance peut être rendu sûr, en ajoutant des procédés d'authentification et de cryptographie aux interactions Client/Serveur; mais, au détriment de la durée d'appel qui se verra encore augmentée. [HCK95]

Pour comparer entre le RPC et la messagerie de point de vue adéquation à la communication, rappelons qu'une communication basée sur la messagerie est asynchrone; ainsi, la durée nécessaire avant que le client ne reçoive la réponse du serveur est à la fois longue et moins prédictive, contrairement au RPC. De ce fait, la messagerie est moins appropriée que le RPC pour le type de communication un-à-un (one-to-one); mais, plus adéquate pour le type de communication un-à-plusieurs (one-to-many); ceci, parce que le client n'a pas à se suspendre en attendant les réponses. [HCK95]

2.1.3. L'INCONVENIENT DU MODELE CLIENT/SERVEUR TRADITIONNEL

Un problème critique avec le modèle Client/Serveur traditionnel est que le serveur fournit un ensemble fixe d'opérations auxquelles sera limité le client [Gra95a, Gra95b, Har95]. Ceci aurait pu ne pas être un problème si tout traitement que veut faire un utilisateur est explicitement offert comme un service disponible. Dans plusieurs applications pratiques, c'est en effet le cas. Cependant dans plusieurs autres applications, les besoins de l'utilisateur ne sont pas anticipés par la conception du serveur [Har95]. Ainsi, toutes les opérations qui ne sont pas disponibles dans l'ensemble fixe d'opérations fournies par le serveur doivent être exécutées chez le client [Gra95a]. En d'autres termes, si le serveur ne fournit pas une opération qui répond exactement au besoin du client, ce dernier doit faire une série d'appels distants au serveur. Le client ramène, à chaque appel, des données intermédiaires (un volume potentiellement important) à travers le réseau pour les traiter à son niveau [Gra95a]. Cette approche permet au client d'effectuer, en effet, tout genre de calcul sur les données; mais, au prix d'une perte de temps et surtout de bande passante [Gra95a, Gra95b, Har95]; particulièrement, lorsque les données intermédiaires transmises se révèlent inutiles pour le client.
Afin d'éviter cette inefficacité, les développeurs de serveurs fournissent souvent des opérations spécialisées pour chaque client [Gra95a, Gra95b]. Malheureusement, ceci devient vite inacceptable dès qu'il s'agit d'un grand nombre de clients, et n'est pas valable pour les clients non prévus au moment du développement des serveurs [Gra95b]. En plus, il y a violation du principe du génie logiciel moderne qui consiste à fournir des primitives simples et efficaces, au lieu de procédures complexes et spécialisées [Gra95a, Gra95b]. Pour remédier à cette inefficacité, il était alors nécessaire de moderniser le modèle Client/Serveur.

2.2. LE MODELE CLIENT/SERVEUR MODERNE

Pour remédier à l'inconvénient relatif au modèle Client/Serveur traditionnel, la solution était de conserver le concept du serveur qui fournit un ensemble de primitives simples et efficaces et de permettre au client de construire sa requête sous forme d'un sous-programme qui sera envoyé au serveur où il sera évalué. Le sous-programme s'exécute chez le serveur et renvoie seulement les résultats au client [Gra95b]. De cette façon, le transfert de données intermédiaires est éliminé, la bande passante est préservée et la lenteur des applications est globalement réduite.
Trois méthodes basées sur ce principe sont souvent rencontrées dans la littérature; il s'agit de NCL, REV et SUPRA-RPC. [Gra95a]

2.2.1. NCL

Falcone [Fal87] décrit un système dans lequel le client peut programmer le serveur et réciproquement, en utilisant NCL (Network Command Language) qui est une variante de LISP.
Chaque serveur fournit une bibliothèque de fonctions écrites en NCL. Un client qui exige un service donné, envoie une expression en NCL au serveur. L'expression peut utiliser les fonctions fournies par le serveur en plus des fonctions incorporées dans l'expression elle-même. Le résultat est aussi une expression en NCL et peut, par conséquent, engendrer une exécution complexe et arbitraire chez le client.
L'usage de NCL réduit la charge du réseau en évitant de multiples RPC; en effet; une expression en NCL permet d'invoquer, localement chez le serveur, les fonctions fournies par ce dernier.

2.2.2. REV

REV (Remote EValuation) est similaire à NCL dans le fait qu'une procédure peut être envoyée à un serveur distant qui l'évaluera; cependant, elle diffère par le fait qu'elle peut être utilisée avec tout langage.
REV utilise des *STUBs* (au niveau du client et du serveur) de la même manière que RPC. Ainsi, tout ce dont a besoin un nouveau langage pour supporter REV, sont des générateurs de *STUBs* et des facilités d'établissement de liens (pour que les procédures fournies par le serveur puissent être invoquées).
L'avantage de REV par rapport à NCL est qu'elle peut être incorporée avec tout langage de programmation, ce qui permet au programmeur d'utiliser le langage le plus approprié pour son application; mais en revanche, NCL est symétrique par le fait que les procédures peuvent être envoyées du client vers le serveur et du serveur vers le client.

2.2.3. SUPRA-RPC

La restriction qu'impose REV et NCL est que la procédure envoyée vers une machine distante soit autosuffisante: Toutes les fonctions et les variables référencées dans la procédure doivent être fournies par le serveur sur la machine distante ou envoyées avec la procédure. Ceci signifie qu'une procédure qui s'exécute sur une machine distante ne peut accéder à tout ce qui est externe au sous-programme transmis. Les développeurs de REV et NCL s'étaient préalablement intéressés à l'idée de déplacer le calcul vers une machine distante et ont imposé cette restriction pour simplifier l'implémentation.
SUPRA-RPC (SUbprogram PaRAmeters in Remote Procedure Call) vient pour remédier à ceci, en étendant le RPC traditionnel avec des *STUBs* additionnels. Ces derniers sont utilisés par le serveur pour faire appel au client afin de traiter le cas où une procédure contient une référence à une variable ou à une fonction non transmise avec le sous-programme et non fournie par le serveur.

Les méthodes NCL, REV et SUPRA-RPC présentent, cependant, de sérieuses limitations [Gra95a, Gra95b]:

- Elles maintiennent la division fixe de rôles entre les programmes (un programme est soit un client, soit un serveur).
- Les sous-programmes sont des entités anonymes; par conséquent, il n'y a pas de moyen convenable pour que deux sous-programmes puissent communiquer. Ceci rend difficile le partage de résultats partiels: au pire des cas, un sous-programme doit être divisé en plusieurs sous-programmes; au meilleur cas, les résultats partiels doivent être transmis deux fois (une première fois, lorsqu'ils sont transmis au site natal qui a lancé le sous-programme, et une deuxième fois, lorsque le site natal les envoie vers les sous-programmes concernés).
- En plus, les sous-programmes ne peuvent pas migrer après leur transfert initial.

L'approche d'agents mobiles est une nouvelle alternative qui permet d'éliminer automatiquement les limitations rencontrées avec les méthodes: NCL, REV et SUPRA-RPC. En effet, c'est une approche qui modélise les applications à travers un ensemble d'agents mobiles. Ces derniers sont des programmes nommés et autonomes, pouvant communiquer et migrer là où ils veulent, et quand ils le veulent. En plus, cette approche est basée sur un modèle pair/pair (Peer/Peer model), dans lequel les agents communiquent entre eux comme des pairs: en se comportant comme des clients ou des serveurs au dépend de leurs besoins actuels. [Gra95b]

3. L'APPROCHE D'AGENTS MOBILES

L'approche d'agents mobiles est un nouveau paradigme pour l'architecture et la réalisation des systèmes distribués [LD95]. Elle a été développée comme une extension et un remplacement au paradigme Client/Serveur [CGHLP95, Gra95b]. Ce dernier, rappelons-le, effectue une division fixe de rôles entre les programmes: un programme est soit un serveur fournissant un ensemble de services ou un client demandant ces services [Gra95b].

L'idée d'un programme qui peut se déplacer d'une machine à une autre sous son propre contrôle n'est pas nouvelle[1] [Gra95a]; cependant, l'usage de cette idée dans la structuration des applications distribuées n'a été popularisée que récemment par des chercheurs et des développeurs intéressés par les services intelligents du réseau, on peut citer: White et Miller de General Magic Incorporation et les développeurs de Tcl [HCK95].

Plusieurs auteurs affirment que l'approche d'agents mobiles constitue un paradigme de programmation adéquat, robuste et efficace pour les applications distribuées (applications transactionnelles, consultation d'informations, etc.), plus particulièrement, quand il s'agit d'environnement mobile. [HCK95]

3.1. LE CONCEPT D'AGENT MOBILE

Le concept d'agent a fait l'objet de recherches intenses dans divers domaines tels que l'intelligence artificielle et les systèmes distribués. Cependant, il n'y a pas encore de consensus sur la définition exacte d'un agent. [LDD95, LD96]

DEFINITION D'UN AGENT

D'après [GKNRC96], un agent est un programme nommé et suffisamment autonome pour agir de manière indépendante même quand l'utilisateur ou l'application qui l'a lancé n'est pas disponible pour le guider et pour interpréter les erreurs qui peuvent résulter de son activité.
Pour cela, il doit contenir un état persistant et doit être capable de communiquer avec son propriétaire, d'autres agents et son environnement en général. [LD96]

Un agent a pour but d'aider son propriétaire dans des tâches compliquées ou de réaliser des travaux de routines. Parmi les applications typiques d'agents, on trouve [LDD95]: la gestion de la messagerie électronique (un agent peut rendre des messages propriétaires, les envoyer, les effacer, les archiver, etc.), l'organisation de conférences (une personne peut lancer un agent qui négociera la date et les horaires, réservera une chambre, etc.) et le filtrage d'informations (suivant des critères et à partir de sources telles que: Usenet news, etc.).

DEFINITION D'UN AGENT MOBILE

Un agent mobile est un agent qui peut se déplacer à travers un réseau hétérogène sous son propre contrôle, migrant d'un site vers un autre et interagissant avec d'autres agents et ressources sur chacun d'eux, puis éventuellement revenir à son site natal une fois sa tâche accomplie. [GKNRC96]

[1] Le lecteur intéressé peut consulter la référence: [WVF89].

Le transfert d'un agent mobile d'un site à un autre s'effectue en utilisant un mécanisme basé sur la messagerie [HCK95]; par conséquent, l'approche d'agents mobiles est une approche asynchrone.

Si un système d'agents mobiles doit supporter la mobilité des agents, les agents en revanche, ne sont pas obligés d'être forcement mobiles. En effet, certains agents peuvent être assez longs pour se déplacer confortablement, d'autres peuvent ne pas éprouver le besoin de se déplacer. Ces agents stationnaires restent, cependant, capables de communiquer avec les autres agents (stationnaires et mobiles) pour accomplir leurs taches. [LD95]

Parmi les applications d'agents mobiles, on peut citer [LD96]: la consultation d'informations, les documents actifs, le commerce électronique, la gestion du réseau et le calcul mobile.

3.2. LES CARACTERISTIQUES DES AGENTS MOBILES

Pour mettre en œuvre un système d'agents mobiles, il est nécessaire de supporter les caractéristiques suivantes: [Bou99a]

a)- *La mobilité*:
La première caractéristique d'un agent mobile est sa mobilité. Elle concerne les mécanismes utilisés pour transporter l'agent entre les différents sites.

b)- *La communication*:
Un modèle de communication est nécessaire dans un système d'agents mobiles pour que les agents puissent communiquer. Ces derniers peuvent vouloir communiquer entre eux en étant sur le même site (communication locale), ou sur des sites différents (communication à distance).

c)- *La gestion des ressources*:
les agents mobiles se déplacent d'un site à un autre à travers le réseau et accèdent, naturellement, à différentes ressources. Ces dernières peuvent être de bas niveau (cycles CPU, accès disque, accès réseau, etc.) ou de haut niveau (accès bases de données, interface utilisateur, serveur SQL, etc.).
le contrôle des ressources concerne aussi bien les propriétaires des sites que les agents. En effet, les propriétaires des sites ont besoin de contrôler la quantité de ressources qu'ils allouent aux agents (le propriétaire d'une station de travail, par exemple, ne souhaite pas que son processeur soit monopolisé par un agent visitant le site; par conséquent, il doit avoir la possibilité de limiter l'accès à cette ressource). Les agents aussi ont besoin de contrôler leurs propres ressources, d'une part, pour optimiser leur propre performance, et d'autre part, pour répondre aux limites qui leur sont imposées par les détenteurs de ces ressources.
Un gestionnaire de ressources est donc nécessaire. Il doit pouvoir établir des contraintes d'accès, limiter les accès aux ressources en fonction des moyens (financiers) de l'agent, attribuer des coûts aux ressources ou les négocier et facturer à l'agent la consommation des ressources effectuée.
Bien que ce problème soit critique, la plupart des infrastructures existantes ne définissent aucun mécanisme de gestion.

d)- *La tolérance aux pannes de site*:
L'agent par sa nature d'exécution qui implique une interaction avec une grande quantité de sites est enclin à disparaître à cause d'une déconnexion soudaine ou imprévue du site qui l'exécute. De plus, les agents mobiles dépendent de sites statiques pour leur résidence,

(éventuellement) leur lancement, et leur exécution. Il est donc nécessaire de définir des mécanismes de tolérance aux fautes pour les sites qui peuvent disparaître à cause d'une défaillance.
Généralement des copies de sauvegarde des agents et/ou des serveurs d'agents sont effectuées afin de pouvoir les restaurer en cas de défaillance.

e)- *La sécurité*:
une attention particulière doit être accordée à la sécurité dans les environnements d'agents mobiles, d'une part, à cause de la nature même de leur modèle d'exécution (capacité de mobilité), et d'autre part, à cause de leur capacité à interagir avec différents systèmes et à accéder à leurs ressources.
Les recherches s'orientent vers deux approches principales: l'approche organisationnelle qui confie la gestion des agents mobiles à des institutions dignes de confiance, et l'approche de confiance/réputation qui stipule que les agents ne doivent se déplacer que vers des serveurs de confiance.
La quasi-totalité des environnements d'agents mobiles existants utilise une approche organisationnelle.

3.3. LES AVANTAGES DE L'APPROCHE D'AGENTS MOBILES

Les agents mobiles offrent plusieurs avantages améliorant la performance de plusieurs applications distribuées. Cette amélioration peut être ressentie dans l'utilisation du réseau, dans le temps de calcul, dans la commodité par rapport aux programmeurs ou simplement dans l'habilité de continuer l'interaction avec un utilisateur durant une déconnexion du réseau [GKNRC96].

L'idée des agents mobiles découle (comme nous l'avons vu) d'un examen critique du modèle Client/Serveur; par conséquent, les principaux avantages des agents mobiles sont toujours identifiés par rapport au modèle Client/Serveur.

Parmi les avantages offerts par l'approche d'agents mobiles:

- Les agents mobiles présentent un meilleur support pour les clients mobiles [HCK95]. En effet, les ordinateurs mobiles souffrent de nombreux problèmes [HCK95]:
  - Ils sont souvent déconnectés du réseau, et
  - Lorsqu'ils sont connectés, ils utilisent une faible bande passante; en plus,
  - La capacité de calcul et de stockage dont ils disposent est généralement assez modeste.

  En adoptant l'approche d'agents mobiles, ces problèmes se règlent avec flexibilité [HCK95]:
  - Un client mobile, même quand il est déconnecté, peut développer un agent (ou programme) représentant sa requête, lance l'agent durant une courte session de connexion, puis se déconnecte immédiatement; la réponse, s'il y en a une, lui sera communiquée lors d'une prochaine reconnexion.
  - Quand un agent mobile se déplace vers le site serveur, il peut effectuer et la consultation et le filtrage des données, ensuite revenir au client sur l'ordinateur mobile avec uniquement l'information utile. Ceci réduit le volume des informations circulant sur le réseau et répond bien au problème de la faiblesse de la bande passante.
  - En plus, en se déplaçant d'un ordinateur mobile vers un ordinateur fixe, l'agent mobile peut profiter des capacités importantes de calcul et de stockage offertes par ce dernier.

- Les agents mobiles facilitent les interactions en temps réel avec les serveurs [HCK95]. En fait, lorsqu'un serveur gère un équipement externe (un robot, par exemple), et si la durée de transmission à travers le réseau est plus grande comparée aux contraintes (en temps réel) imposées par cet équipement, alors, un programme déclenché par le client pour contrôler l'équipement doit se déplacer pour s'exécuter de préférence sur le serveur. Ce schéma d'exécution est naturellement supporté par les agents mobiles.

- Les transactions et les requêtes basées sur les agents mobiles sont plus robustes [HCK95]. En effet, dans les implémentations courantes, une communication via le RPC est relativement fragile. Les applications Client/Serveur utilisant le RPC n'ont été développées que pour les systèmes basés sur les réseaux de type LAN (Local Area Network), où les développeurs d'applications peuvent mettre des hypothèses fortes concernant l'intégrité des communications et la disponibilité des serveurs.
  L'expérience a montré que l'extension de ces applications pour couvrir les réseaux WAN (Wide Area Network), les rend moins fiables. Il semble que c'est ce problème qui a poussé GMI (General Magic Incorporation) à introduire les agents mobiles. [HCK95]
  Les agents mobiles offrent, en effet, (en étant basés sur la messagerie) un transport fiable entre les clients et les serveurs même si la couche de communication du réseau est non fiable. Ceci est dû au fait qu'un agent mobile est asynchrone; par conséquent, on n'a pas à fixer (contrairement au RPC) un délai de retransmission qui deviendra inacceptable dès qu'il s'agit de grands réseaux tels que: les réseaux WAN ou Internet. [HCK95]

- Les transactions basées sur les agents mobiles n'ont pas besoin de préserver leurs états d'exécution. En fait, pendant l'exécution d'une transaction distribuée, le client et le serveur doivent préserver à chaque interaction Client/Serveur l'état complet d'exécution. En plus, pour pouvoir faire le recouvrement en cas de panne, ils doivent disposer de méthodes leur permettant de rendre ces états préservés persistants.
  L'approche d'agents mobiles a l'avantage de libérer le client et le serveur d'une telle charge. En effet, un agent mobile encapsule son état et le transporte avec lui quand il se déplace, annulant ainsi le besoin de préserver l'état d'exécution de la transaction. [HCK95]

- Le taux d'exécution des transactions basées sur les agents mobiles est plus élevé que celui des transactions basées sur RPC [HCK95]. Cet avantage découle du fait que l'approche d'agents mobiles est une approche asynchrone contrairement au RPC qui est une approche synchrone; et toute approche asynchrone peut permettre un taux élevé de transactions entre les serveurs; ceci, car après l'émission de la requête (message, agent, etc.), on n'a pas à se suspendre jusqu'à l'obtention de la réponse. [HCK95]

- Rendre une transaction sûre est plus performant quand elle est basée sur l'approche d'agents mobiles que sur RPC [HCK95]. En effet, l'exécution d'une transaction distribuée nécessite généralement plusieurs appels RPC; cependant, un seul agent mobile peut l'accomplir. [HCK95]
  Quand un agent se déplace, il implique une seule transmission; par conséquent, rendre un agent mobile sûr revient à rendre sûre une seule transmission. Par contre, pour rendre sûre une transaction basée sur RPC, il faut rendre sûre toutes les transmissions résultantes des appels RPC; Ainsi, un RPC sûr est moins performant. [HCK95]

- Les agents mobiles permettent aux utilisateurs de personnaliser le comportement d'un serveur [HCK95]. En fait, un serveur offre généralement des fonctions de base à travers une application client à exploiter par les utilisateurs. Mais, un agent mobile peut ne pas se limiter aux fonctions offertes et, en se déplaçant vers le serveur, réalise des tâches

spécifiques par manipulation directe des données du serveur. Cet avantage est très intéressant pour les utilisateurs; mais, expose éventuellement les serveurs aux dangers, tels que les attaques des virus. [HCK95]

- Les agents mobiles peuvent servir pour construire des prototypes à des applications qu'on veut réaliser avec le RPC [GKNRC96, HCK95]. En fait, Pour réaliser une application avec le RPC, on doit d'abord l'implémenter, ensuite la standardiser puis l'installer largement. Ce processus est à la fois difficile et long. Par contre, une implémentation utilisant les agents mobiles peut s'effectuer rapidement. En effet, les agents sont mobiles, autonomes, flexibles et peuvent fonctionner malgré le fait d'être faiblement couplés (contrairement au RPC) avec les logiciels existants; ainsi, ils facilitent le développement, le test et l'utilisation des applications. [HCK95]
  Même si on pense qu'elle peut fonctionner moins efficacement qu'une implémentation basée sur le RPC, une implémentation utilisant les agents mobiles peut servir comme prototype aux applications. Le prototype permettra de prouver des concepts et d'évaluer des caractéristiques [HCK95]. Ensuite et à la lumière des résultats obtenus, on peut décider:
  - D'entamer une implémentation avec le RPC,
  - De garder l'implémentation avec les agents mobiles, ou
  - D'abandonner complètement l'application.

- Les agents mobiles n'exigent pas une pré-installation de logiciels spécifiques à l'application sur les sites vers lesquels ils peuvent migrer (à l'exception du système d'agents mobiles qui doit être présent). [GKNRC96]

- Les agents mobiles peuvent aussi se distribuer et se redistribuer dynamiquement sur le réseau pour équilibrer sa charge ou accélérer le traitement. [GKNRC96]

- L'usage d'agents mobiles élimine le besoin de détecter et de gérer les pannes de réseau excepté durant la migration d'un agent. [GKNRC96]

- En plus, cette approche mène le programmeur du modèle Client/Serveur rigide à un modèle plus flexible Pair/Pair (Peer/Peer Model). Dans ce modèle, les programmes (ou agents) communiquent entre eux comme des pairs en se comportant comme des clients ou des serveurs selon leurs besoins actuels [GKNRC96]. Ils peuvent ainsi bénéficier des services et offrir eux-mêmes des services, ce qui leur permet d'enrichir continuellement leurs connaissances et, par conséquent, les rend plus utiles et plus intéressants.
  Même si l'approche d'agents mobiles est vue comme une extension au modèle Client/Serveur traditionnel, elle restera, comme même, très intéressante: les clients peuvent programmer les serveurs et réciproquement; ce qui étend largement les fonctionnalités que les développeurs d'applications et de serveurs peuvent fournir à leurs consommateurs. [Gra95a]

4. L'APPROCHE D'AGENTS MOBILES EST-ELLE LA PLUS ADEQUATE?

Si on examine individuellement chaque avantage offert par l'approche d'agents mobiles, on peut souvent déduire la possibilité de l'obtenir sans recourir à cette approche [HCK95]. En effet:

- Les caractéristiques des ordinateurs mobiles posent de sérieux problèmes que les agents mobiles règlent avec souplesse. Cependant, ce n'est pas l'unique solution; une solution

alternative peut consister à créer sur le site fixe auquel est connecté l'ordinateur mobile, un représentant (agent fixe) qui accomplira la tâche du client mobile; mais cette fois-ci, à partir d'un ordinateur fixe et non mobile. [HCK95]

- Les agents mobiles facilitent les interactions en temps réel avec les serveurs; cependant, les applications (en temps réel) ne sont qu'un type d'application dans une panoplie d'applications distribuées. Ainsi, si l'approche d'agents mobiles est adéquate pour ce type d'application, le reste-t-elle pour les autres types? [HCK95]. En plus, les applications (en temps réel) peuvent également s'effectuer avec le modèle Client/Serveur moderne.

- Si les transactions et les requêtes deviennent plus robustes en adoptant l'approche d'agents mobiles, ceci est grâce à l'usage d'un mécanisme basé sur la messagerie pour transporter les agents. En plus, si ceci peut constituer un avantage motivant pour cette approche, il peut également être vu comme motivant à l'amélioration des autres alternatives (le RPC, par exemple) pour qu'elles demeurent puissantes même à travers de grands réseaux, tels que les réseaux WAN et même Internet. [HCK95]

- Rendre une transaction sûre est plus performant lorsqu'il s'agit d'agents mobiles que de RPC; cependant, le même résultat peut être obtenu avec le modèle Client/Serveur moderne. En plus, dans la pratique on ne cherche pas à rendre sûre chaque étape d'une transaction basée sur le RPC; les applications peuvent se contenter de rendre sûre uniquement la validation finale [HCK95].

- Les agents mobiles permettent aux utilisateurs de personnaliser le comportement d'un serveur; mais, cet avantage peut être obtenu aussi avec le modèle Client/Serveur moderne.

Comme on le voit, plusieurs avantages offerts par l'approche d'agents mobiles peuvent être également obtenus avec d'autres alternatives. Néanmoins, si chaque avantage peut probablement être obtenu avec une autre alternative, l'approche d'agents mobiles permet de les avoir tous en même temps. [HCK95]

L'approche d'agents mobiles est, par conséquent, une approche très prometteuse. Elle constitue un bon support pour les applications distribuées et un excellent support lorsqu'il s'agit d'environnement mobile [GKNRC96, HCK95]. En plus, elle est mieux vue comme un outil général permettant de réaliser des applications distribuées quelconques [GKNRC96].

L'approche d'agents mobiles ne peut, cependant, être utilisée que si le problème d'insécurité qu'elle expose est réglé. En effet, la souplesse avec laquelle les agents mobiles se déplacent et manipulent les données donne aux agents malintentionnés (virus, cheval de Troie, etc.) la possibilité d'exister et de causer, par conséquent, le dysfonctionnement du système.
En réglant ce problème (et plusieurs travaux s'effectuent continuellement dans ce domaine), l'approche d'agents mobiles peut être utilisée pour améliorer les performances des applications distribuées et supporter également l'environnement mobile.

5. CONCLUSION

Les systèmes distribués sont souvent basés sur le modèle Client/Serveur, dans lequel on distingue deux types d'entités: le fournisseur du service (ou serveur) et le consommateur du service (ou client) [Fiu94, Har95]. Le client et le serveur résident généralement sur des

sites distincts; par conséquent, les interactions Client/Serveur exigent souvent (pour s'accomplir) des communications à travers le réseau [Gra95b].

Ces dernières années ont connu une abondance des informations accessibles aux applications, ce qui a rendu la capacité de transport des réseaux considérablement insuffisante pour supporter efficacement les interactions Client/Serveur; en plus, un nouveau paradigme de calcul est apparu: le calcul mobile. Par conséquent, les modèles Client/Serveur existants sont devenus inadéquats devant le trafic important qui peut en résulter et pour mettre en œuvre des applications complexes [PB95]. Il était donc nécessaire de développer d'autres approches pouvant servir efficacement aux besoins des utilisateurs et utiliser intelligemment les ressources du réseau. C'est ainsi que l'approche d'agents mobiles est apparue pour étendre et remplacer le modèle Client/Serveur [Gra95b, Har95].

Aucune application n'exige cependant l'usage absolu des agents mobiles; la plupart des applications peuvent, en effet, être réalisées avec des programmes stationnaires et quelques paradigmes de communication adéquats, tels que, le RPC. Cependant, ceci sera au prix du chargement du réseau, et peut être, à l'incommodité par rapport aux utilisateurs. [LD95]

L'approche d'agents mobiles est une approche récente qui continue d'attirer l'attention grâce aux avantages qu'elle offre aux systèmes distribués. En réglant le problème d'insécurité dû à la mobilité des agents (et plusieurs travaux sont menés dans ce domaine), cette approche peut être utilisée comme une nouvelle et puissante méthode dans la structuration des applications distribuées, plus particulièrement, en environnement mobile. [HCK95]

CHAPITRE III

PROPOSITION D'UNE INFRASTRUCTURE D'AGENTS MOBILES POUR UN ENVIRONNEMENT MOBILE

A travers ce chapitre (structuré en deux parties) on cherche à proposer une infrastructure d'agents mobiles destinée à supporter les applications réparties en environnement mobile.

- Dans la partie (I), on discute l'idée de choisir une infrastructure d'agents mobiles parmi les infrastructures d'agents mobiles existantes.
- Dans la partie (II), on décrit l'infrastructure SINDBAD proposée pour supporter les applications d'agents mobiles en environnement mobile; mais, sans développer comment la rendre basée sur HTTP? (cette question fera, en effet, l'objet du chapitre IV).

PARTIE I

CHOIX D'UNE INFRASTRUCTURE D'AGENTS MOBILES POUR UN ENVIRONNEMENT MOBILE

1. INTRODUCTION

Les avantages que peuvent apporter les agents mobiles ont motivé le développement de systèmes permettant de les supporter[١]. Plusieurs systèmes d'agents mobiles ont dès lors existé[٢]; cependant, leurs infrastructures généralement diffèrent. Il n'y a pas d'infrastructure commune pour réaliser les agents mobiles; en plus, il est improbable qu'un standard commun pour réaliser les agents mobiles émergera dans le futur proche[٣] [DLD97].

En absence d'une infrastructure qui peut servir comme un standard pour réaliser les applications d'agents mobiles, nous développons dans cette partie l'idée de choisir une infrastructure parmi les infrastructures d'agents mobiles existantes. Pour cela, nous commençons par examiner quelques systèmes d'agents mobiles, nous discutons ensuite le choix d'un langage adéquat pour l'écriture des agents mobiles et nous terminons par proposer une infrastructure d'agents mobiles pour un environnement mobile.

2. QUELQUES SYSTEMES D'AGENTS MOBILES

Parmi les systèmes d'agents mobiles connus, on trouve [Gra95b]: Tacoma, Telescript, MΦ, IBM Itinerant Agent et Obliq.

- Tacoma (Tromsϕ and Cornell Moving Agents) est un système qui supporte des agents mobiles écrits avec le langage Tcl/Horus. Ce dernier est une version du langage d'écriture Tcl qui utilise Horus pour fournir la communication par groupe et la tolérance aux fautes. [Gra95b]
  L'infrastructure de Tacoma offre une seule abstraction qui est l'opération *Meet* permettant à un agent d'exécuter un autre. Notons que sur chaque site du système existe un serveur d'agents qui gère, entre autres, les requêtes de type *Meet*. Les autres services, excepté *Meet*, sont tous fournis pas des agents; par exemple, la migration est assurée par un agent de service, appelé *Ag_tcl*, existant sur chaque site du système. Ainsi, un agent qui veut migrer doit donc invoquer le service *Meet* pour exécuter l'agent *Ag_tcl* existant sur le site distant vers lequel il veut se déplacer.
  Tacoma présente des caractéristiques intéressantes: il dispose d'agents d'arrière garde (rear guard agents) permettant de relancer les agents perdus et supporte la monnaie

[١] Les systèmes d'agents mobiles sont développés dans le cadre de projets académiques ou industriels.
[٢] Les chercheurs dans le domaine des agents mobiles à l'université de Stuttgart ont enregistré plus de cinquante systèmes d'agents mobiles. Pour avoir une idée, l'annexe peut être consultée.
[٣] D'après Lingnau (Le développeur principal de la plate-forme FFMAIN), les exigences d'une infrastructure d'agents mobiles ne sont pas en réalité parfaitement déterminées actuellement; en plus, les infrastructures existantes diffèrent suffisamment d'une infrastructure à une autre rendant, par ce fait, difficile d'envisager une infrastructure unique (et standard) combinant toutes les caractéristiques et les avantages des différentes infrastructures.

électronique[1] servant à payer pour les services utilisés et à contrôler aussi la durée de vie des agents dans le système. [Gra95b]

Tacoma présente également des faiblesses: [Gra95b]

- La migration d'un agent n'est pas complètement transparente. En effet, l'interruption de l'exécution d'un agent mobile lors d'une migration n'est pas supportée. Après une migration, l'agent reprend son exécution depuis le début et non depuis l'instruction de migration. Ceci rend difficile l'écriture d'un agent qui doit préserver ses informations d'état tout en migrant à travers une séquence de sites. Néanmoins, ceci n'est pas impossible puisque les informations d'état peuvent être explicitement collectées puis passées avec le code de l'agent.
- En plus, Tacoma ne fournissait aucun mécanisme de sécurité et sa partie Horus est non disponible sur la plupart des plates-formes[2].

L'application la plus connue de Tacoma est l'évaluation des tempêtes. Il s'agit d'une simulation distribuée de temps qui manipule d'immenses volumes de données dont le transfert à travers le réseau est, par conséquent, pratiquement inacceptable. [GKNRC96]

- Telescript est le système d'agents mobiles le plus connu; il a été conçu par General Magic Incorporation (GMI) [GKNRC96]. Les agents s'écrivent avec un langage orienté objets propre à GMI, désigné pour la programmation des réseaux [Toh97]. Ce langage n'est pas un langage de programmation général, mais plutôt, un langage spécifique à la communication (comme, par exemple, le langage Postscript qui est spécifique à l'impression) [Toh97]

  La migration des agents dans Telescript est complètement transparente. Elle se réalise à l'aide d'une seule instruction Go qui peut apparaître n'importe où dans le code de l'agent. Après une migration, l'agent reprend son exécution à partir de l'instruction qui suit immédiatement l'instruction Go. La transparence du transfert des informations d'état de l'agent assurée par Telescript (lors de la migration) est plus adéquate comparée à l'agent *Ag_tcl* de Tacoma. [Gra95b]

  La communication des agents est basée dans Telescript sur le concept d'objets. Notons qu'un agent qui crée un objet devient alors son propriétaire. Pour qu'un agent puisse communiquer avec un autre agent se trouvant sur le même site, il doit obtenir les références des objets détenus par ce dernier. Cependant, si les agents se trouvent sur des sites distincts, leur communication nécessitera le transfert de leurs objets. [Gra95a, Gra95b]

  Telescript dispose au niveau de chaque site d'un serveur d'agents qui reçoit les agents arrivant au site, vérifie s'il peut les accepter[3], puis déclenche leur exécution. Le serveur d'agents s'assure également des contraintes de sécurité, gère le passage des objets et restitue, après une panne de site, les agents à partir de leurs informations d'état sauvegardées auparavant.

  Telescript offre aussi une sécurité assez complète: [Gra95a]

  - Chaque agent détient une signature cryptée servant à l'identifier auprès des autres sites.
  - En plus, chaque agent acquiert, au moment de son lancement dans le système, un certain pouvoir lui spécifiant les instructions qu'il peut exécuter ou les services qu'il peut utiliser. Ceci peut inclure la durée de vie maximale de l'agent

[1] La monnaie électronique (ou le payement digital) est un outil indispensable à la réalisation du commerce électronique qui est l'une des plus importantes applications envisagées pour les agents mobiles. [LDD95]

[2] Il y a à noter que des améliorations ont été apportées par la suite sur Tacoma consistant à l'augmenter avec quelques mécanismes de sécurité et à abandonner largement sa partie Horus. D'autre part, et en plus de Tcl, les agents peuvent, actuellement, être écrits aussi avec les langages C, Perl, Python et Scheme.

[3] Il doit vérifier, par exemple, qu'ils ne sont pas des agents malintentionnés (virus, cheval de Troie, etc.).

et/ou son budget[1] maximum. L'agent qui tente de violer le pouvoir qui lui a été accordé sera immédiatement terminé.

Telescript est un système d'agents mobiles assez complet qui a été utilisé dans des applications, telles que [GKNRC96]: la gestion des réseaux, la messagerie électronique active (active E-Mail) et le commerce électronique. Il est l'un des systèmes d'agents mobiles les plus robustes [Gra95a]; malheureusement, c'est un produit commercial qui reste peu accessible aux chercheurs [Gra95b]; autrement, il aurait pu constituer une bonne référence pour les développeurs de systèmes d'agents mobiles.

- MΦ, quant à lui, est un système qui permet à des fragments de code d'être envoyés pour s'exécuter sur un site distant. Chaque site fournit un environnement d'exécution qui inclut un interpréteur de langage, des primitives de communication et un dictionnaire spécifiant les données partagées.
  Les développeurs de MΦ se sont intéressés aux systèmes d'exploitation distribués. Pour cela, MΦ est un système de bas niveau: il n'offre pas directement la fonctionnalité des agents mobiles; mais, peut être utilisé, plutôt, comme une couche basse dans un système d'agents mobiles. [Gra95b]

- IBM Itinerant Agent[2] est un autre système qui combine les agents mobiles avec les techniques basées sur les connaissances (servant à découvrir les ressources). Les développeurs de ce système se sont intéressés, principalement, aux aspects concernant les connaissances. [Gra95b]

- Finalement, Obliq est un système orienté objet et est, comme Telescript, intrinsèquement lié à un langage spécifique (ou propre) qui est, sans que ceci ne soit nécessaire, complexe pour plusieurs applications [Gra95b]. D'autre part, et tel que Tacoma, Obliq ne supporte pas l'interruption de l'exécution d'un agent mobile lors d'une migration. Après une migration, l'agent reprend son exécution depuis le début et non depuis l'instruction de migration. [Gra95a]

Il y a plusieurs autres systèmes qui ne sont pas des systèmes d'agents mobiles; mais qui présentent, cependant, des aspects de la conduite d'agents mobiles: les routeurs intelligents fournissent une migration arbitraire et se déplacent d'une machine à une autre pour accomplir une tâche donnée; la combinaison Safe-Tcl/MIME permet aux scripts en Tcl d'être encapsulés dans des messages d'E-Mail; le navigateur HotJava permet aux scripts en Java d'être encapsulés dans des documents de type WWW (World Wide Web); une application Sodabot peut distribuer dynamiquement ses composantes; et, enfin, les programmes en Postscript sont souvent envoyés pour être affichés sur des machines distantes. [Gra95b]

3. CHOIX D'UN LANGAGE ADEQUAT A L'ECRITURE DES AGENTS MOBILES

D'une manière évidente, le code d'un agent mobile peut, en principe, être écrit avec n'importe quel langage de programmation qui convient au programmeur. Cependant, des problèmes pratiques rendent certains types de langage, potentiellement, plus utilisables que d'autres pour l'écriture des agents mobiles dans les réseaux hétérogènes. Une considération importante est que le code de l'agent doit être portable pour qu'il puisse s'exécuter de

[1] Le budget accordé à l'agent sert à payer pour les services utilisés.
[2] IBM Itinerant Agent est un projet qui a été arrêté après que son objectif était atteint. Le but de ce projet était, en effet, de voir comment concevoir avec les agents mobiles.

manière identique sur tout site vers lequel l'agent mobile peut migrer. Ceci rend les langages traditionnels, tels que C, inadéquats à l'écriture des agents mobiles. En effet, les agents écrits avec des langages traditionnels doivent être transférés en tant que codes sources (puisque ces langages sont toujours compilés vers des langages machines spécifiques à l'architecture matérielle); cependant, même entre, par exemple différents environnements du langage C implémentés sous UNIX, la complétude de ces environnements et leur conformité au standard varient à un degré qui rend impraticable la recompilation automatique et transparente du code de l'agent après une migration. Par conséquent, réaliser une telle portabilité et l'étendre ensuite pour couvrir d'autres environnements, tels que: MS-DOS, Macintosh et VMS constitue une tâche difficile. [LD95]

Il est donc plus adéquat d'utiliser des langages qui sont directement interprétables, tels que Tcl et Perl, ou des langages, tels que Java, qui sont compilables vers des langages intermédiaires qui soient interprétables et portables (pouvant, par conséquent, être transportés et exécutés sans recompilation).
Dans les deux cas, il suffit d'installer un interpréteur (pour le langage d'écriture d'agents s'il s'agit du premier cas, ou pour le langage intermédiaire s'il s'agit du deuxième cas) sur chaque site vers lequel l'agent mobile peut migrer. Ce dernier peut, ainsi, s'exécuter indépendamment des caractéristiques matérielles des sites. [LD95]

Bien qu'un programme écrit dans un langage compilable s'exécute plus rapidement que s'il est écrit dans un langage interprétable, la portabilité des langages interprétables a fait que, généralement, les systèmes d'agents mobiles les utilisent plutôt que d'utiliser des langages compilables. [Har95]
Une autre raison, également importante, motivant l'usage des langages interprétables concerne la sécurité du système d'agents mobiles. En effet, si un agent est écrit avec un langage compilable, son développeur peut, de manière intentionnelle ou non, inclure dans son corps des fragments de code arbitraires qui peuvent probablement violer la sécurité du système. Par contre, l'usage d'un langage interprétable permet de découvrir, avant l'exécution de l'agent, l'existence de telles menaces (ou dangers) et, par conséquent, d'y remédier. [Har95]

Maintenant qu'il est clair qu'un langage interprétable pour écrire les agents mobiles est nécessaire à tout système d'agents mobiles ouvert [Har95], une question évidente consiste à savoir: Quel est le meilleur langage interprétable?
D'après [LD95], il n'y a pas de consensus sur le langage de programmation qui soit le meilleur pour l'écriture des agents mobiles. En fait, la plupart des projets de système d'agents mobiles utilisent leurs propres langages qui souvent ne sont pas des standards; par exemple, General Magic Incorporation (GMI) utilise Telescript, IBM utilise Object REXX et Sun utilise Hotjava.
Le choix d'un langage d'écriture d'agents constitue alors une décision difficile. Un facteur important dans cette décision peut consister à choisir un langage communément utilisé, tel que Java ou Tcl. Néanmoins, bien que Java[1] et Tcl[2] sont devenus les langages évidents pour l'implémentation des systèmes d'agents mobiles, ils ne sont pas les langages de choix pour la plupart des chercheurs dans le domaine des agents intelligents. Dans ce domaine, des langages, tels que Lisp et Mit sont plus connus. Par conséquent, l'usage, dans le futur, des langages familiers à la communauté d'I.A. (Intelligence Artificielle) peut aider à intégrer le travail des chercheurs sur les agents intelligents avec celui des chercheurs sur les agents mobiles.

[1] L'annonce par Sun du langage de programmation Java constitue l'événement qui a popularisé, le plus, la notion du code mobile. [Toh97]
[2] Tcl est un langage de haut niveau qui a été développé en 1987 et qui a connu, depuis, une large popularité.

L'infrastructure d'agents mobiles que nous proposerons plus loin dans ce chapitre, est une infrastructure pour laquelle la question du choix d'un langage d'écriture d'agents ne se pose pas: elle permet de supporter différents langages d'écriture d'agent et accepte l'addition directe d'un nouveau langage.
En effet d'après [LD95], une infrastructure d'agents mobiles ouverte et viable ne doit pas obliger le programmeur d'agents à utiliser un ensemble fixe de langages; mais plutôt, elle doit lui permettre d'utiliser le langage qui soit le plus convenable à son application; ceci, tout en maintenant une base commune d'interopérabilité permettant aux agents de continuer à communiquer et à travailler ensemble même s'ils sont écrits avec des langages différents.

4. CHOIX D'UNE INFRASTRUCTURE D'AGENTS MOBILES POUR UN ENVIRONNEMENT MOBILE

Il existe principalement trois approches pour la conception et l'implémentation d'un système d'agents mobiles: [PK98, Bou99a]

١. La définition ou la création d'un langage spécialisé dont les caractéristiques répondent aux besoins d'un système d'agents mobiles (c'est le cas, par exemple, de Telescript).
٢. L'extension d'un système d'exploitation pour supporter en plus les besoins d'un système d'agents mobiles (c'est le cas, par exemple, de Tacoma qui étend le système d'exploitation UNIX[١]); ou
٣. La construction d'une application logicielle spécialisée qui s'exécute sur un système d'exploitation pour fournir les fonctionnalités d'un système d'agents mobiles.

Pour l'instant, la technologie des agents mobiles n'est pas assez mûre pour pouvoir certifier qu'elle est l'approche la plus adaptée [PK98]. Il est, cependant, important de souligner que la plupart des projets en cours ont opté pour la dernière approche. [Bou99a, Bou99b]

Les systèmes d'agents mobiles présentés dans la section 2 de ce chapitre, ainsi que plusieurs autres souffrent d'une ou de plusieurs des faiblesses suivantes: [Gra95b]

- La migration ne peut pas s'effectuer à des points arbitraires dans le code de l'agent ou exige une capture explicite des informations d'état de l'agent[٢].
- La communication entre les agents est inexistante ou s'effectue difficilement.
- Les mécanismes de sécurité sont inexistants.
- Les agents doivent être écrits avec un langage spécifique qui est souvent complexe.
- Des portions de l'implémentation tournent seulement sur des plates-formes spécifiques d'UNIX, comme dans Telescript.
- Le système est parfois non documenté, tel qu'est le système Telescript.

Afin de remédier à ces faiblesses, différents projets de recherches ont été lancés pour réaliser d'autres systèmes d'agents mobiles. Les projets auxquels nous nous sommes intéressés et desquels nous nous sommes principalement inspirés pour construire l'infrastructure que nous proposons pour supporter les agents mobiles dans un environnement mobile, sont: Agent-Tcl et FFMAIN.

---

[١] D'après Karmouch, il n'est pas clair comment Tacoma peut progresser, vu que c'est le seul projet qui supporte les agents mobiles par extension d'un système d'exploitation. [PK98]

[٢] Dans ce dernier cas, la migration est dite non transparente ou faible. Ce type de migration est rencontré dans Tacoma et aussi dans la plupart des infrastructures implémentées en Java, telles que: Aglet, Concordia, Voyager et Odyssey. [PK98]

4.1. AGENT-TCL

Agent-Tcl est un système d'agents mobiles en cours de développement au collège de Dartmouth. Les objectifs que les développeurs de ce système ont fixé consistent à: [Gra95b]

- Réduire la migration de l'agent à une seule instruction *Agent-Jump* (similaire à l'instruction *Go* de Telescript) et permettre à cette instruction d'apparaître à des points arbitraires dans le code de l'agent.
- Fournir une communication transparente entre les agents.
- Supporter plusieurs langages d'écriture d'agents, tels que Tcl, Java, Scheme et Python et plusieurs mécanismes de transport, tels que TCP/IP et E-Mail, tout en permettant l'addition directe d'un nouveau langage ou d'un nouveau mécanisme de transport.
- S'exécuter sur des plates-formes générales d'UNIX et être portable le plus facilement possible vers des plates-formes autres que celles d'UNIX.
- Assurer une tolérance aux fautes et une sécurité effective dans le monde incertain des grands réseaux, tels que Internet; et
- Etre largement documenté pour l'intérêt des autres chercheurs.

La figure suivante schématise l'architecture du système "Agent-Tcl"

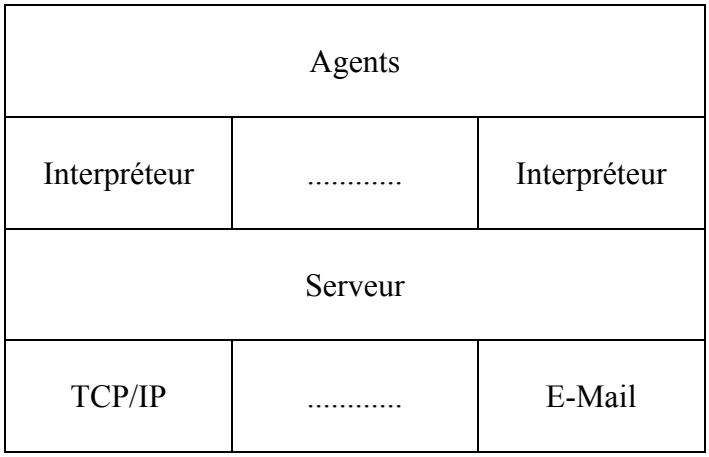


Figure III-I-1: Architecture d'Agent-Tcl

Chaque site du système Agent-Tcl est structuré en quatre niveaux: [Gra95b]

- Le plus bas niveau est constitué d'une IPA (ou API: Application Programming Interface) pour chaque mécanisme de transport supporté[1].
- Le deuxième niveau est un serveur d'agents dont le rôle est d'accomplir les tâches suivantes:
  - Maintenir la trace de l'ensemble des interpréteurs disponibles sur le site.
  - Accepter un agent arrivant, contrôler l'identité de son propriétaire, puis le faire passer à l'interpréteur approprié.
  - Maintenir la trace des agents s'exécutant sur le site et traiter les requêtes concernant leurs états courants.
  - Maintenir un espace de noms hiérarchique dans lequel chaque agent possède un nom unique.
  - Fournir l'accès à un support de stockage non volatile, où les agents peuvent enregistrer (ou sauvegarder), quand ils le désirent, leurs états courants.
  - Restaurer, après une panne de site, les agents à partir de leurs états sauvegardés.

[1] Actuellement, le seul mécanisme de transport supporté par Agent-Tcl est TCP/IP. [PK98]

- Permettre à un agent de migrer vers un site distant (ou de créer un agent fils et l'envoyer vers le site distant) tout en se chargeant de choisir le mécanisme de transport approprié à l'envoi de l'agent; et
- Supporter la communication entre les agents à travers l'échange de messages ou via une connexion directe[1]. Une connexion directe est un flot de messages nommé qui s'établit entre deux agents, lorsque le premier invoque la commande *Agent-meet* et le second accepte le meeting.
  Le serveur d'agent intervient dans le processus de communication:
  - ✓ En choisissant le mécanisme de transport approprié aux messages envoyés,
  - ✓ En mémorisant les messages arrivés, et
  - ✓ En créant une connexion directe quand un agent accepte la commande *Agent-meet* issue d'un autre agent. Cette connexion est ensuite contrôlée par le serveur d'agents ou bien par l'interpréteur du langage associé à l'agent.

Les autres services, tels que: la navigation dans le réseau, la communication par groupe, le payement digital, la tolérance aux fautes, etc., sont fournis, comme dans Tacoma, par des agents.

- Le troisième niveau dans l'architecture d'Agent-Tcl est constitué d'un interpréteur pour chaque langage d'écriture d'agents supporté[2].
  Chaque interpréteur est constitué de quatre composants (comme indiqué sur Figure III-I-2):
  - L'interpréteur lui-même.
  - Un module de sécurité qui empêche l'agent d'entreprendre des actions malintentionnées.
  - Un module d'état qui permet de capturer (respectivement de restaurer) l'état d'un agent en cours d'exécution (respectivement en vue de reprendre son exécution), et
  - Une IPA (ou API: Application Programming Interface) qui permet d'interagir avec le serveur d'agents afin de bénéficier des services qu'il assure, tels que: la migration et la communication.

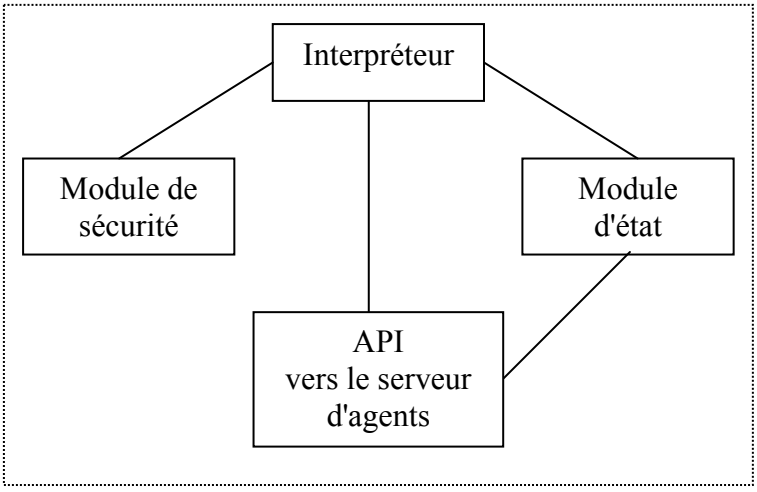


Figure III-I-2: Structure de l'interpréteur dans Agent-Tcl.

[1] Sur la base de ces deux moyens de communication, Agent-Tcl supporte, en plus, la communication entre agents à travers un moyen similaire au RPC. [Gra95b]

[2] Pour permettre la communication entre des agents écrits dans différents langages, Agent-Tcl a définit un protocole de communication particulier (ou propre); en plus, chaque langage doit être augmenté d'une bibliothèque de Stubs lui permettant d'utiliser le protocole.

- Finalement, le quatrième niveau dans l'architecture d'Agent-Tcl représente les agents.

Agent-Tcl est un système d'agents mobiles assez complet qui a permis le développement rapide et efficace d'applications distribuées, telles que, les applications de traitement de l'information. [Gra95b]

4.2. FFMAIN

FFMAIN (FrankFurt Mobile Agent INfrastructure) est un autre système d'agents mobiles en cours de développement à l'université de Frankfurt. Son infrastructure se rapproche de celle d'Agent-Tcl dans plusieurs points. En effet, sur chaque site du système existe un serveur d'agents qui contrôle les agents visitant le site et leur fournit les moyens de communication et de transport[١]. Il permet aussi au propriétaire d'un agent de consulter l'état d'exécution de son agent ou de lui soumettre des ordres, tels que: arrêter l'exécution, reprendre l'exécution ou mettre fin à l'exécution. les autres services, tels que: l'ordonnancement et l'adressage sont fournis, comme dans Agent-Tcl, par des agents. [LD95]
FFMAIN supporte également différents langages d'écriture d'agents, citons à titre d'exemple, Tcl et Perl. En plus, des services avancés, tels que le routage sémantique[٢], peuvent être rajoutés d'une manière modulaire. [LD95]

L'infrastructure de FFMAIN diffère de celle d'Agent-Tcl:

- Par le fait qu'elle est complètement basée sur HTTP (HyperText Transfert Protocol), qui est un standard d'Internet.
  Bien que Agent-Tcl dispose d'outils permettant d'interagir avec l'environnement WWW[٣] (Wide World Web), être basée sur HTTP permet, cependant, à une infrastructure d'agents d'être directement intégrable dans l'univers WWW. En plus, ceci permet d'adopter immédiatement les résultats des recherches intenses qui s'effectuent continuellement sur WWW, particulièrement, dans le domaine de la sécurité. Ces résultats, en plus d'être soigneusement élaborés, bénéficient du consensus de la communauté Web.
- FFMAIN diffère aussi par le fait qu'il supporte la communication des agents avec l'approche de l'espace d'information.
  Le serveur d'agents sur chaque site du système FFMAIN maintient un espace d'information local. L'espace d'information est une abstraction d'un moyen de communication qui permet aux différents agents s'exécutant sur son site de communiquer entre eux. Il s'agit d'un mécanisme de bas niveau, indépendant des langages, permettant d'effectuer différents styles d'interaction entre les agents (un-à-un, un-à-plusieurs et plusieurs-à-plusieurs) et sur lequel peuvent être organisées des communications plus structurées en utilisant des protocoles de haut niveau, tels que KIF (Knowledge Interchange Format) et KQML (Knowledge and Query Manipulation Language). [LD96]
  L'accès à l'espace d'information s'effectue dans FFMAIN à travers le protocole HTTP qui est supporté généralement par la plupart des langages d'écriture d'agents. Ainsi, un agent écrit avec n'importe quel langage d'écriture d'agents peut accéder à l'aide de HTTP à l'espace d'information; et ce, même si l'agent appartient à une infrastructure autre que FFMAIN. Il en résulte que FFMAIN permet, grâce à ses espaces d'information accessibles à l'aide de HTTP, de supporter la communication entre différents agents; ces

[١] Le transport des agents s'effectue dans FFMAIN avec le protocole HTTP. [LD95]
[٢] Le routage sémantique permet de déterminer la destination d'un agent mobile à partir de ce que l'agent veut accomplir, et non à partir d'une information d'adressage explicite. [LD95]
[٣] Tout système d'agents doit permettre, pour être plus utilisable, l'accès à WWW.

derniers peuvent être écrits avec un même langage, avec différents langages, et peuvent même appartenir à différentes infrastructures. [DLD97]

FFMAIN a été utilisé pour développer des applications, telles que: l'ordonnancement des rendez-vous et la consultation des informations. [LD95]

4.3. PROPOSITION D'UNE INFRASTRUCTURE D'AGENTS MOBILES POUR UN ENVIRONNEMENT MOBILE

D'après [Gra95a], un système d'agents mobiles doit supporter deux tâches de base: la communication des agents et la migration d'un agent d'un site vers un autre. En plus, le système doit être efficace, tolérant aux fautes, flexible et sécurisé.
En absence d'une infrastructure pouvant servir comme un standard commun pour supporter les agents mobiles, nous nous sommes intéressés, parmi les systèmes d'agents mobiles existants, principalement à: Agent-Tcl et FFMAIN.

- Agent-Tcl est un système d'agents mobiles dont l'infrastructure est assez complète, assez générale et clairement présentée à travers une documentation largement disponible. Par conséquent, l'infrastructure d'Agent-Tcl peut constituer un choix très intéressant pour le développement des applications d'agents mobiles pouvant fonctionner également dans un environnement mobile.
- FFMAIN est un autre système d'agents mobiles également très intéressant et dont l'infrastructure est aussi assez générale. Cette dernière se rapproche globalement de l'infrastructure d'Agent-Tcl, et diffère par le fait qu'elle est complètement basée sur HTTP et qu'elle supporte la communication des agents à travers un moyen de communication abstrait (appelé, espace d'information), accessible à l'aide de HTTP.
    - Le fait d'utiliser un moyen de communication abstrait (indépendant des langages) et accessible à l'aide de HTTP, permet à FFMAIN de supporter facilement la communication entre des agents qui peuvent être écrits avec différents langages et qui peuvent même appartenir à différentes infrastructures.
    - Et le fait d'être complètement basée sur HTTP, lui permet d'être largement accepté.

Dans l'espérance d'obtenir une infrastructure d'agents mobiles qui soit assez complète, assez générale, familière, plus convenable et largement acceptée, nous avons pensé à combiner essentiellement entre l'infrastructure d'Agent-Tcl et celle de FFMAIN. L'infrastructure résultante de ce mariage est celle que nous proposons pour supporter les agents mobiles dans un environnement mobile; nous l'avons appelé: SINDBAD.

5. CONCLUSION

Plusieurs systèmes d'agents mobiles existent; cependant, il n'y a pas de réponses évidentes spécifiant: Quels sont les langages les plus adéquats à l'écriture et à la communication des agents? Quels styles d'interaction entre les agents doit-on avoir? Ou comment arriveront-ils à découvrir leurs destinations? [LD95]
Depuis l'apparition des agents mobiles, plusieurs infrastructures ont été proposées pour les supporter. Ces infrastructures généralement diffèrent. Il n'y a pas d'infrastructure commune pour réaliser les agents mobiles; en plus, il est improbable qu'un standard commun pour réaliser les agents mobiles émergera dans le futur proche [DLD97].

En absence d'une infrastructure pouvant servir comme un standard commun pour développer des applications d'agents mobiles, nous avons pensé à combiner essentiellement entre deux infrastructures intéressantes: l'infrastructure d'Agent-Tcl et l'infrastructure de

FFMAIN; ceci, dans l'espérance d'obtenir une infrastructure qui soit assez complète, assez générale, familière, plus convenable et largement acceptée. Nous avons appelé cette infrastructure: SINDBAD. Il s'agit, en fait, de l'infrastructure d'Agent-Tcl rendue complètement basée sur HTTP et supportant la communication des agents à travers un moyen de communication abstrait (appelé espace d'information), accessible à l'aide de HTTP.
Il y a à noter, cependant, que l'usage de l'espace d'information n'empêche pas les agents (qui sont supposés être intelligents) d'utiliser, si nécessaire, d'autres moyens pour communiquer entre eux. En effet, si les agents veulent, par exemple, effectuer des transferts de données à haut débit, ils peuvent alors utiliser d'autres moyens de communication, tels que: les connexions directes et les mémoires partagées [LD95].

L'infrastructure SINDBAD, que nous proposons pour supporter les agents mobiles dans un environnement mobile, est largement détaillée et argumentée davantage à travers la partie (II) de ce chapitre.

PARTIE II

SINDBAD: UNE INFRASTRUCTURE D'AGENTS MOBILES POUR UN ENVIRONNEMENT MOBILE

1. INTRODUCTION

Rappelons que SINDBAD, l'infrastructure que nous proposons pour supporter les agents mobiles dans un environnement mobile, est le résultat d'un mariage effectué essentiellement entre deux infrastructures auxquelles nous nous sommes intéressés: Agent-Tcl et FFMAIN. Plus précisément, SINDBAD est l'infrastructure d'Agent-Tcl rendue complètement basée sur HTTP et supportant la communication des agents via un moyen de communication abstrait, appelé espace d'information.

Rendre une infrastructure d'agents mobiles basée sur HTTP n'est pas obligatoire, en effet, ça lui permet tout simplement d'être plus avantageuse. Pour cela, cette partie détaillera l'infrastructure SINDBAD sans développer comment est-elle basée sur HTTP? Cette question fera l'objet du chapitre suivant.
Dans ce qui suit alors, on commence par présenter la structure d'un agent mobile, on détaille ensuite l'infrastructure SINDBAD pour un environnement fixe, on l'étend après pour supporter aussi les ordinateurs (ou unités) mobiles, on examine ensuite la communication des agents et on termine par discuter la sécurité dans l'univers des agents mobiles.

2. STRUCTURE D'UN AGENT MOBILE

Un agent mobile est un programme qui s'exécute en tant que processus ayant une identité unique[GKNRC96]. La structure d'un agent mobile comprend[`]: [LDD95, LD95]

- Un code (écrit dans un langage d'écriture d'agents donné) qui défini la conduite (ou l'exécution) de l'agent.
- Un état (les variables internes de l'agent, etc.) qui permet à l'agent de reprendre, suite à une migration vers un autre site, son activité (ou exécution) à partir de l'instruction qui suit immédiatement l'instruction de migration, et
- Des attributs qui décrivent l'agent, ses exigences et l'histoire de ses déplacements à l'infrastructure d'agents.
  Ces attributs incluent certainement l'identificateur unique de l'agent, ainsi que son propriétaire (vers lequel seront envoyés les résultats intermédiaires, les messages d'erreurs et les commentaires sur les conduites anormales de l'agent). Ils peuvent spécifier aussi des restrictions sur la mobilité de l'agent (une date d'expiration, par exemple, ou des contraintes, telles que: ne pas se déplacer plus de trois fois à partir du site Foo.bar.com) ou des restrictions sur son horizon de recherche (telles que: ramener au moins 20 références et au plus 100 références).

[`] La structure d'agent mobile adoptée dans SINDBAD est celle proposée dans FFMAIN ; en effet, elle n'a pas été présentée dans les articles décrivant Agent-Tcl.

Les attributs peuvent exprimer également des exigences en ressources (par exemple, 5 Mo de RAM), des limitations sur les ressources (par exemple, 5,49$ seulement sont en possession de l'agent pour payer les services à utiliser), ainsi que des exigences sur l'environnement dont l'agent a besoin pour accomplir sa tâche (tels que, l'accès à une base de données (BDD) bibliographique selon les standards FOO et BAR). Les attributs doivent inclure, en plus, des informations d'authentification permettant à l'infrastructure d'authentifier l'agent et de s'assurer, par exemple, que seul le propriétaire de l'agent peut stopper l'agent, le reexécuter, consulter l'état de son exécution ou connaître ses clés et mots de passe (permettant d'accéder à des informations protégées).
Une partie des attributs peut être rendue accessible à l'agent; mais, ce dernier ne doit pas être capable de modifier ses attributs.

Remarquons qu'un agent stationnaire n'est, en fait, qu'un agent mobile qui ne bouge pas; par conséquent, les expressions "agent" et "agent mobile" vont être utilisées, dans cette partie et le chapitre suivant, pour désigner (selon le cas) les deux types d'agent.

٣. L'ARCHITECTURE DE SINDBAD POUR UN ENVIRONNEMENT FIXE

SINDBAD, comme toute autre infrastructure ouverte permettant de supporter les agents mobiles dans un environnement mobile, doit s'efforcer d'assurer les exigences suivantes: [GKNRC96]

- Réduire la migration d'un agent à une seule instruction, permettre à cette instruction d'apparaître à des points arbitraires dans le code de l'agent, et une fois appelée, capture de manière transparente l'image courante (ou état d'exécution courant) de l'agent et transmet cette image au site de destination. L'infrastructure doit cacher tous les détails de transmission (y compris le fait que le site de destination change d'adresse sur le réseau ou qu'il soit déconnecté) rendant, par ce fait, le processus de migration transparent au programmeur.
- Permettre aux agents de communiquer de manière transparente. Les primitives de communication doivent être souples, de bas niveau et doivent cacher tous les détails de transmission.
- Supporter plusieurs langages d'écriture d'agents, avec la possibilité d'addition directe d'un nouveau langage.
  Chaque langage d'écriture d'agents exige un environnement d'exécution spécifique, ainsi chaque site destiné à exécuter des agents doit contenir autant d'environnements d'exécution que de langages d'écriture d'agents qu'il supporte.
- Assurer une sécurité effective dans le monde incertain des grands réseaux, tels que Internet.

L'architecture suivante présente l'infrastructure SINDBAD dans un environnement fixe; en effet, elle ne prend pas encore en compte les ordinateurs mobiles.

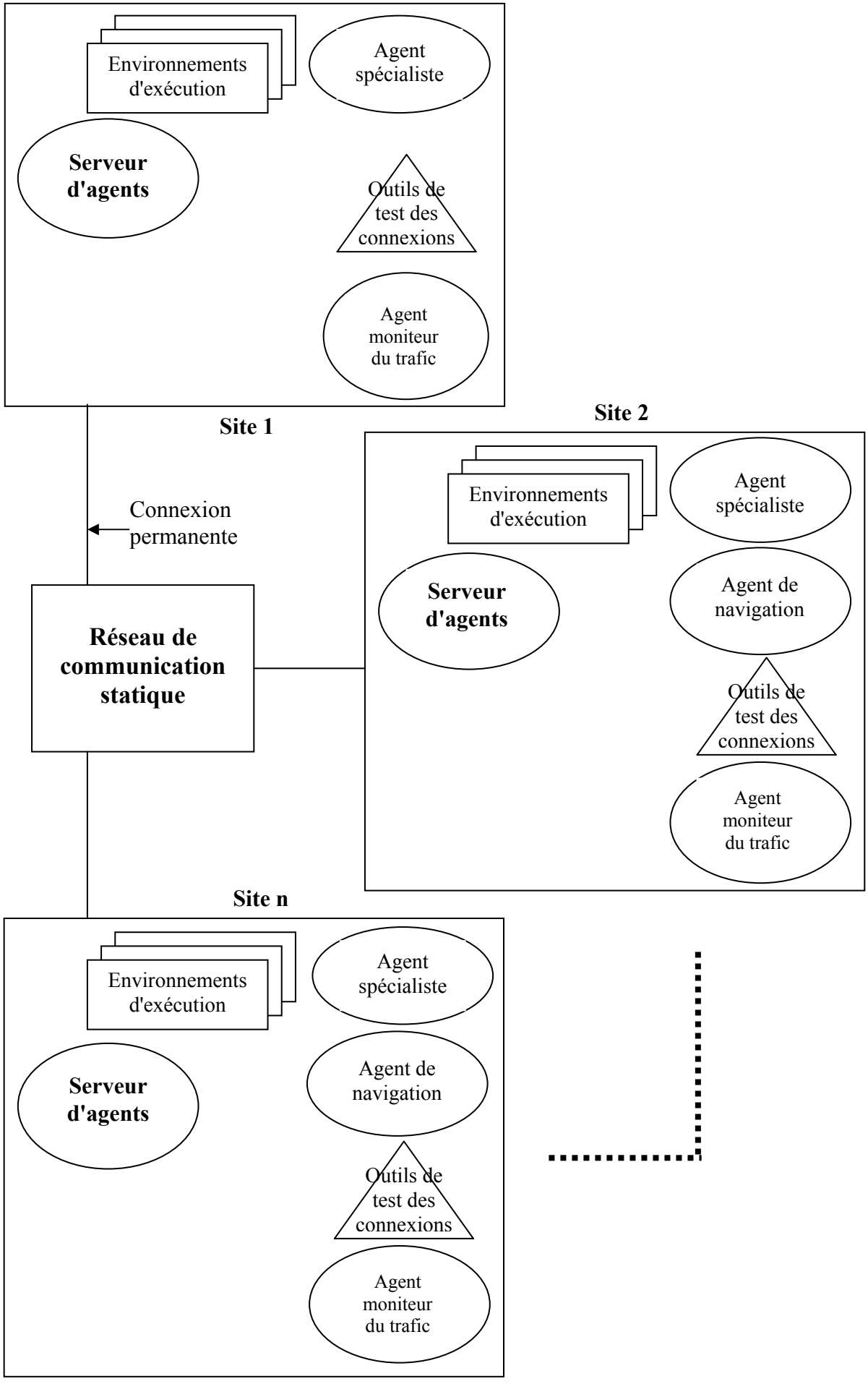


Figure III-II-1: L'architecture de SINDBAD pour un environnement fixe.

Dans cette architecture, figurent les différents composants qui doivent exister sur chaque site pouvant exécuter des agents mobiles. Ces composants contrôlent et assistent un agent depuis son lancement dans le système jusqu'à l'accomplissement de sa tâche, après avoir naviguer éventuellement sur le réseau.

3.1. SERVEUR D'AGENTS

le serveur d'agents est un **agent stationnaire** lancé lors du démarrage du site, son rôle consiste à:

- Garder la trace de l'ensemble des environnements d'exécution disponibles sur le site.
- Recevoir un agent lancé localement ou venant d'un autre site et vérifier s'il peut l'accepter.
- Attribuer pour chaque agent un environnement d'exécution approprié.
- Garder la trace des agents s'exécutant sur son site.
- Fournir aux agents des facilités de communication et de transport[\].

3.2. ENVIRONNEMENTS D'EXECUTION

A chaque langage d'écriture d'agents dans l'infrastructure d'agents est associé un environnement d'exécution. Cet environnement supporte l'exécution des agents écrits dans le langage d'écriture associé et assure le rôle d'interface entre ces agents et le serveur d'agents.

Un environnement d'exécution est composé de quatre composantes:

- L'interpréteur du langage d'écriture supporté.
  On parle d'interpréteur puisqu'il est supposé, pour des raisons de portabilité et de sécurité, que la plupart des langages d'écriture d'agents mobiles sont des langages interprétables.
- Un module de sécurité qui empêche les agents d'entreprendre des actions malintentionnées.
- Un module d'état qui permet de capturer (respectivement, de restaurer) l'image (ou l'état d'exécution) d'un agent en cours d'exécution (respectivement, en vue de reprendre son exécution), et
- Une IPA (ou API: Application Programming Interface) qui permet aux agents d'interagir avec le serveur d'agents. En effet, elle offre les primitives permettant d'accéder à partir du langage d'écriture d'agents supporté par l'environnement d'exécution aux services (ou primitives) de communication et de transport assurés par le serveur d'agents.
  Ainsi, tout agent écrit dans un langage d'écriture donné peut, à travers l'IPA de l'environnement d'exécution associé à son langage d'écriture, accéder aux services de communication et de transport offerts par le serveur d'agents.

[\] Le transport des agents sera examiné dans le chapitre suivant puisqu'il est assuré avec HTTP.

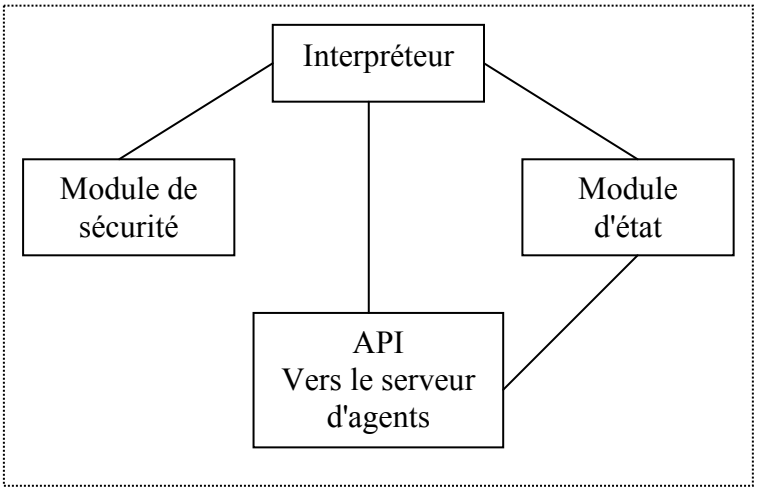


Figure III-II-2: Structure d'un environnement d'exécution

Il y a à noter, cependant, que certains langages n'assurent pas la migration des agents et permettent seulement l'envoie des agents fils; donc, les environnements d'exécution associés à ces langages n'auront pas besoin d'un module d'état. D'autre part, d'autres langages, tels que C et C++, peuvent supporter uniquement la communication des agents; les environnements d'exécution associés à ces langages n'auront, par conséquent, pas besoin de modules de sécurité et d'état; quant au module "interpréteur", il est remplacé, dans ce cas, par "compilateur". [Gra95b]

3.3. SCENARIO D'EXECUTION D'UN AGENT MOBILE

Un agent mobile est un programme informatique écrit dans un langage d'écriture d'agents. Une fois lancé, il doit s'enregistrer dans l'infrastructure d'agents en émettant une requête d'enregistrement vers le **serveur d'agents**, généralement, sur le site où il a été lancé. Le serveur d'agents, s'il trouve qu'il peut exécuter l'agent[1], lui déclenche l'environnement d'exécution approprié.
L'agent mobile s'exécute alors et peut décider d'accomplir sa tache complètement sur son site actuel, comme il peut décider sous son propre contrôle de continuer sur un autre site. Dans ce dernier cas, l'environnement d'exécution capture l'image courante de l'agent, l'encode (grâce au module de sécurité qu'il contient) et demande au serveur d'agents local de l'envoyer vers le site destinataire.

[1] Le serveur d'agents parvient à déterminer s'il peut accepter d'exécuter un agent donné, en examinant une partie de ses attributs pour connaître, par exemple, si le propriétaire de l'agent a effectivement le droit d'exécuter son agent sur ce site, et dans le cas affirmatif, si le site est capable de satisfaire aux exigences (ou ressources) réclamés par l'agent.

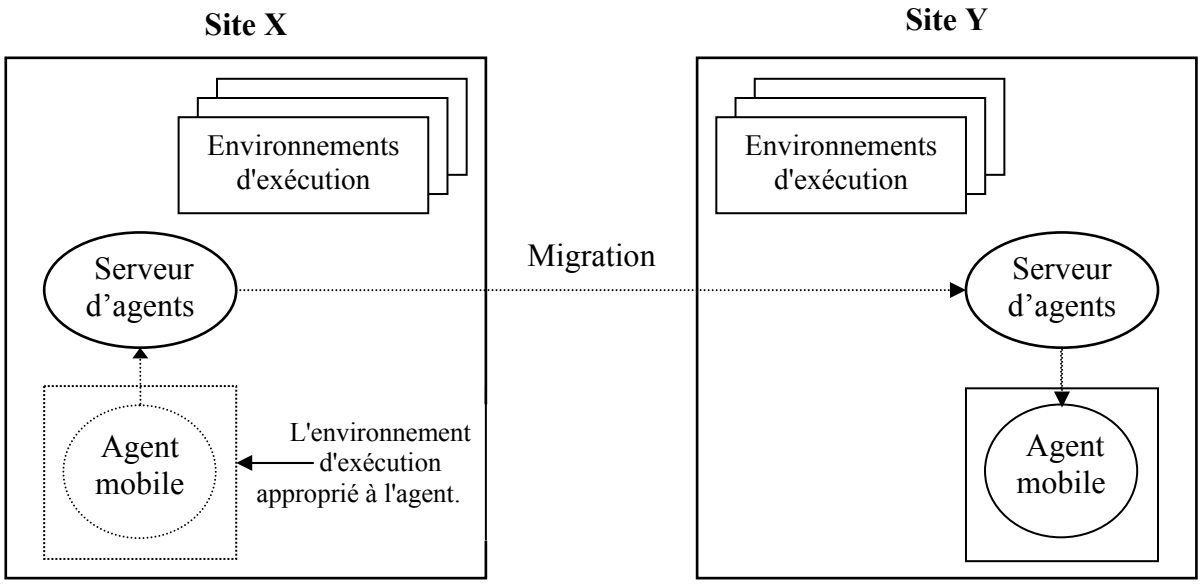


Figure III-II-3 Schéma sommaire d'exécution d'un agent mobile

Au niveau du site destinataire, l'image capturée de l'agent est reçue par le serveur d'agents qui déclenche, après avoir vérifier qu'il peut exécuter l'agent, l'environnement d'exécution approprié. L'environnement d'exécution décode l'image reçue, restaure l'agent et reprend son exécution à l'instruction qui suit immédiatement l'instruction de migration.
Quand l'agent mobile termine sa tâche, il supprime son enregistrement de l'infrastructure d'agents (en émettant une requête de suppression vers le serveur d'agents sur lequel il s'est enregistré), puis disparaît.

3.4. NECESSITE D'ASSISTER UN AGENT MOBILE

Le monde d'un agent mobile est dynamique et incertain: des machines peuvent tomber en panne, d'autres peuvent apparaître; les informations stockées sur les différents sites peuvent changer; en plus, la séquence exacte des destinations et des étapes nécessaires pour compléter une tâche est souvent non complètement connue au moment du lancement de l'agent.

Pour être autonome, un agent mobile a besoin donc de supports lui permettant de percevoir son monde extérieur afin d'adapter son comportement aux changements dynamiques dans son environnement. Ces supports peuvent être assurés par des outils testant l'état des connexions, et **trois types d'agents stationnaires**: Agent moniteur du trafic, agent de navigation et agent spécialiste.

3.4.1. OUTILS TESTANT L'ETAT DES CONNEXIONS

Considérons un agent mobile dont la tâche nécessite la visite de plusieurs ressources d'information existant sur différents sites, un agent intelligent doit être capable de s'adapter au fait que certains sites peuvent être actuellement non atteignables, et par conséquent, commencer par visiter les autres sites.

Les outils de test de l'état des connexions permettent à un agent mobile de connaître si son site actuel est physiquement connecté, et si des sites distants sont atteignables ou non. Ceci peut se faire à travers la commande "ping" standard.

3.4.2. AGENT MONITEUR DU TRAFIC

A coté des informations indiquant l'état des connexions, une estimation de la bande passante et des délais réels entre les sites du réseau rendra un agent mobile encore plus intelligent. En effet, ce dernier peut sur la base des délais, ordonner la liste des sites à visiter; de même, il peut en connaissant la bande passante actuellement disponible, déterminer le volume et le format des données à transporter.

Au lieu de mesurer la bande passante ou le délai en envoyant plusieurs messages à un site distant (ce qui peut prendre le temps nécessaire à l'envoi de l'agent mobile lui-même), l'agent moniteur du trafic peut enregistrer des informations sur le trafic antérieur (ou communications antérieures) délivrées par le serveur d'agents[١] local.

Quand un agent mobile a besoin d'informations sur la bande passante ou les délais vers un site quelconque, il contacte localement l'agent moniteur du trafic qui lui fournit une estimation calculée à partir des informations enregistrées.

Les routeurs et certains sites disposent d'informations précieuses concernant les connexions, la bande passante, et les délais sur le réseau qui leur permettent d'acheminer les paquets vers leurs destinations. Si ces informations peuvent être rendues disponibles à l'infrastructure d'agents mobiles, les agents moniteurs du trafic peuvent réaliser de meilleures prévisions.

3.4.3. AGENT DE NAVIGATION ET AGENT SPECIALISTE

Dans une infrastructure d'agents, des agents peuvent vouloir présenter leurs services pour l'intérêt d'autres agents. Pour les localiser, un agent mobile a besoin d'accéder à un index dynamique indiquant les agents de service et leurs localisations.
Cet index n'est, en fait, qu'un système de pages jaunes virtuelles, où sont décris les agents offrant des services et leur localisation. Il est organisé sous forme d'une base de données distribuée de localisation de services maintenue par un ensemble hiérarchique d'agents de navigation.

Un nouvel agent de service s'enregistre au niveau d'un ou de plusieurs agents de navigation afin de publier sa localisation. Il décrit le service qu'il offre à travers une liste de mots clés.

Sur chaque site où peuvent s'exécuter des agents mobiles existe aussi un agent spécialiste[٢]. Cet agent connaît la localisation d'un ou de plusieurs agents de navigation (qui connaissent à leur tour la localisation de quelques agents de service ainsi que d'autres agents de navigation).

Quand un agent mobile cherche un service donné, il contacte l'agent spécialiste sur son site actuel pour obtenir une liste d'agents de navigation. Il présente ensuite à l'agent de navigation choisi, une requête formée de mots clés décrivant le service recherché. L'agent de

[١] Le serveur d'agents obtient les informations concernant le trafic sur le réseau, en consultant les attributs des agents arrivant sur son site. En effet, chaque agent peut exprimer, à travers ses attributs, le site d'où il est venu et l'instant à laquelle il l'a quitté.

[٢] Les agents de navigation n'existent pas forcement sur tous les sites. Une raison peut être de faciliter leur gestion, surtout avec le contenu de la base de données maintenue qui peut devenir considérablement important. Pour cela, d'autres agents trop simples (les agents spécialistes) ont été introduit, au niveau de chaque site pouvant exécuter des agents mobiles, dans le seul but de localiser les agents de navigation.

navigation examine la requête et communique à l'agent mobile la localisation des services qui peuvent le satisfaire.

Exemple:
Dans cet exemple, il y a deux agents de navigation: un sur le site1 et l'autre sur le site2. L'agent spécialiste sur le site 3 connaît la localisation des deux agents de navigation.
L'agent de service (service1) est enregistré au niveau de l'agent de navigation2 (ceci en ayant suivi la démarche suivante: service1 a contacté d'abord l'agent spécialiste local (sur le site4) qui connaît la localisation de l'agent de navigation2, puis, a envoyé une requête d'enregistrement à l'agent de navigation2 qui l'a ajouté à sa base de données).

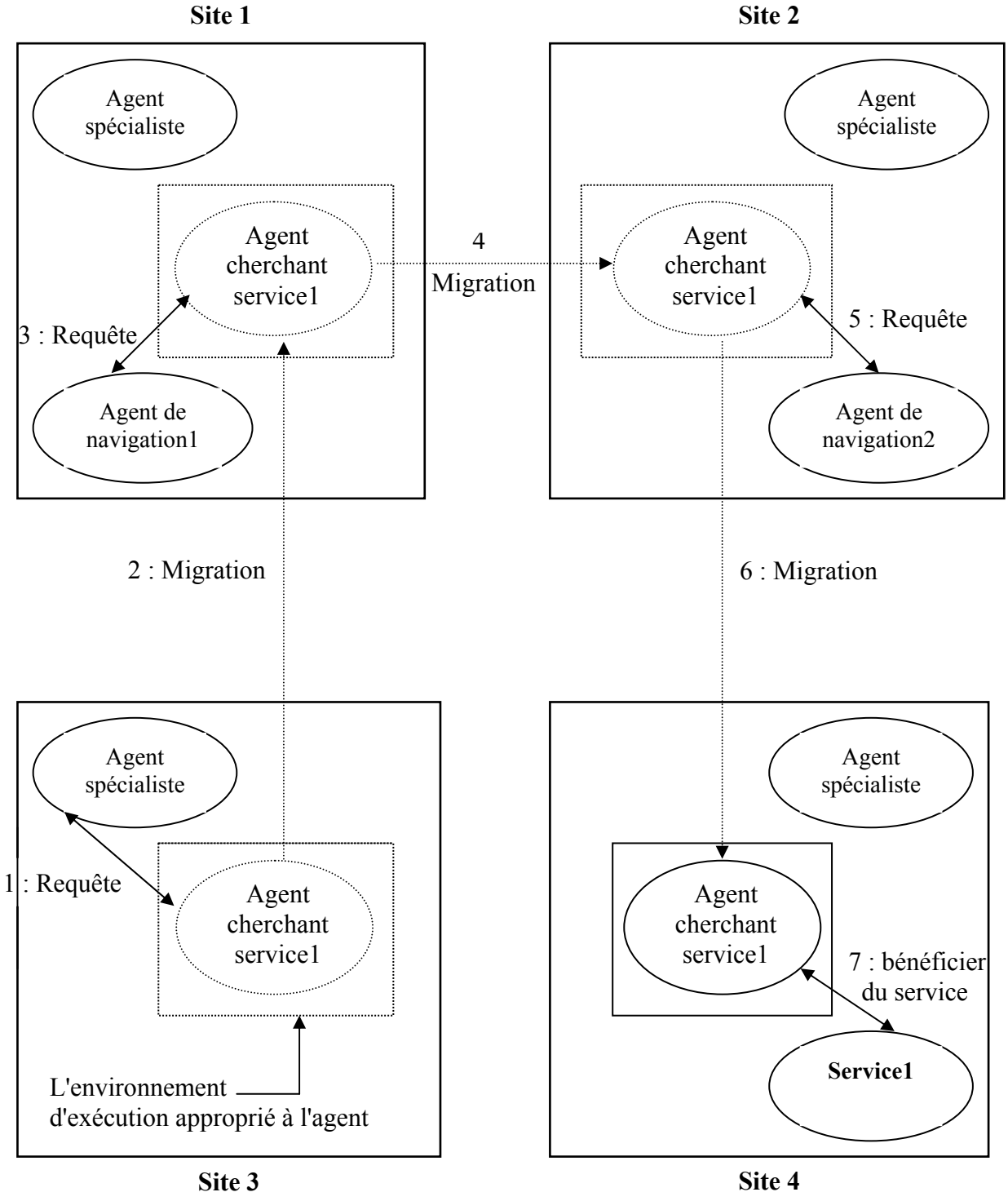


Figure III-II-4: Recherche d'un service.

Les étapes suivantes décrivent le scénario que suit l'agent mobile (cherchant service1) se trouvant sur le site3 pour atteindre service1:

- Il commence par contacter l'agent spécialiste local et obtient la localisation des agents de navigations 1 et 2.
- Se déplace ensuite vers le site1 et demande à l'agent de navigation1 la localisation du service1.
  L'agent de navigation1 ne connaît pas la localisation du service1, car ce dernier n'est enregistré qu'au niveau de l'agent de navigation2.
- L'agent mobile se déplace alors vers le site2 et obtient de l'agent de navigation2, la localisation du service1.
- Finalement, il se déplace vers le site4 contenant le service1, pour en bénéficier.

En conclusion, en consultant l'agent spécialiste local et en visitant un ou plusieurs agents de navigation, un agent mobile peut obtenir la liste nécessaire des services à visiter et leurs localisations. Puis, en contactant l'agent moniteur du trafic et en utilisant les outils permettant de tester l'état des connexions, l'agent mobile peut établir un plan de navigation **adaptatif** pour accomplir sa tâche.
La figure suivante illustre ceci.

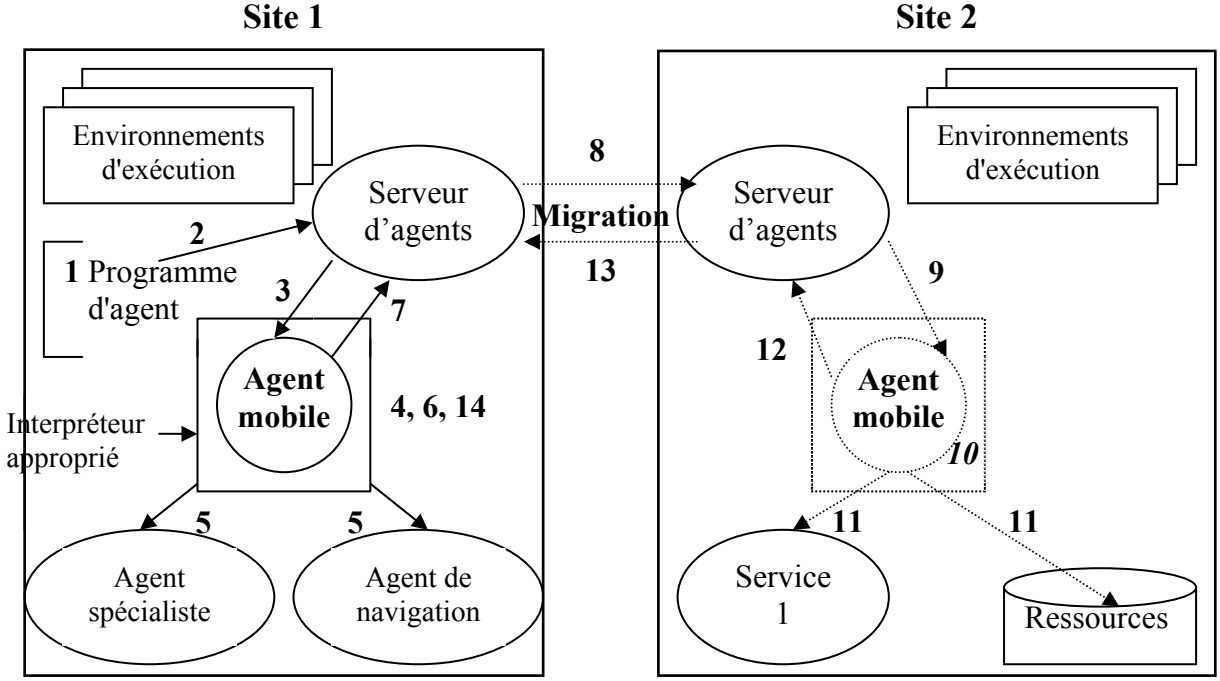


Figure III-II-5: Schéma d'exécution d'un agent mobile

Légende:

١. Un agent mobile est un programme écrit dans un langage d'écriture d'agents.
٢. Une fois lancé, demande au serveur d'agents de l'enregistrer.
٣. Le serveur d'agents vérifie s'il peut exécuter l'agent, puis lui déclenche l'environnement d'exécution approprié.
٤. L'agent commence son exécution sur le site1, et peut à un moment donné avoir besoin d'un service donné, par exemple: service1.
٥. Il consulte localement l'agent spécialiste puis l'agent de navigation et détermine que service1 se trouve sur le site2.

٦. L'agent décide alors de migrer vers le site2, en appelant l'instruction de migration[١].
٧. L'environnement d'exécution capture l'image courante de l'agent, l'encode et demande au serveur d'agents d'envoyer cette image vers le site2.
٩. Au niveau du site2, l'image de l'agent sera reçue par le serveur d'agents qui (après avoir vérifier s'il peut exécuter l'agent) déclenche l'environnement d'exécution approprié.
١٠. L'environnement d'exécution décode l'image reçue, restaure l'agent et reprend son exécution à partir de l'instruction qui suit l'instruction de migration.
١١. L'agent interagit sur le site2 avec l'agent fournissant service1, et peut être même avec d'autres agents et ressources; ensuite, décide de revenir au site1.
١٢. Le même processus de migration sera observé, et l'agent va se retrouver de nouveau sur le site1.
١٤. Au niveau du site1, l'agent termine son exécution en présentant éventuellement ses résultats, supprime son enregistrement du serveur d'agents et disparaît.

4. EXTENSION DE SINDBAD A UN ENVIRONNEMENT MOBILE

Parmi les systèmes d'agents mobiles existants, il y en a, d'une part, ceux qui supportent les agents mobiles dans un environnement fixe, citons à titre d'exemple [GKNRC96]: ARA; et de l'autre part, ceux qui s'étendent pour supporter les agents mobiles dans un environnement mobile, on peut citer [GKNRC96]: Telescript, Agent-Tcl et FFMAIN.

L'infrastructure SINDBAD, telle qu'elle a été présentée, s'applique à un environnement fixe; cependant, l'ajout d'une simple extension va lui permettre de prendre en compte aussi les ordinateurs mobiles[٢].

Rappelons qu'un ordinateur mobile présente les caractéristiques suivantes: [GKNRC96]
١. Il n'a pas une connexion permanente, et est souvent déconnecté du réseau pour une longue période.
٢. Quand il est connecté, la connexion a souvent une faible bande passante, une grande lenteur et peut être sujette à des pannes soudaines.
٣. La performance d'une connexion peut varier considérablement d'une session à une autre en fonction du canal de transmission utilisé.
٤. En plus, il peut avoir des adresses réseau différentes d'une connexion à une autre selon sa localisation ou la nature du canal de transmission utilisé.

L'usage de l'approche d'agents mobiles dans un environnement mobile règle automatiquement les points 2 et 3. Par contre, les points 1 et 4 nécessiteront un support supplémentaire; en effet:

- Un agent mobile continuera de s'exécuter même lorsque l'ordinateur est déconnecté. Par conséquent, si cet agent veut se déplacer (ou migrer) de (ou vers) un ordinateur déconnecté, l'approche traditionnelle (essayer→délais→dormir→réessayer→, etc.) va se révéler inadéquate ; surtout, si l'agent mobile ne réessaye pas l'opération de migration pendant les brèves périodes de reconnexion.
- En plus, un ordinateur qui change son adresse réseau doit communiquer sa nouvelle adresse à une entité spécifique dans l'infrastructure pour permettre l'acheminement des messages et agents qui lui sont destinés.

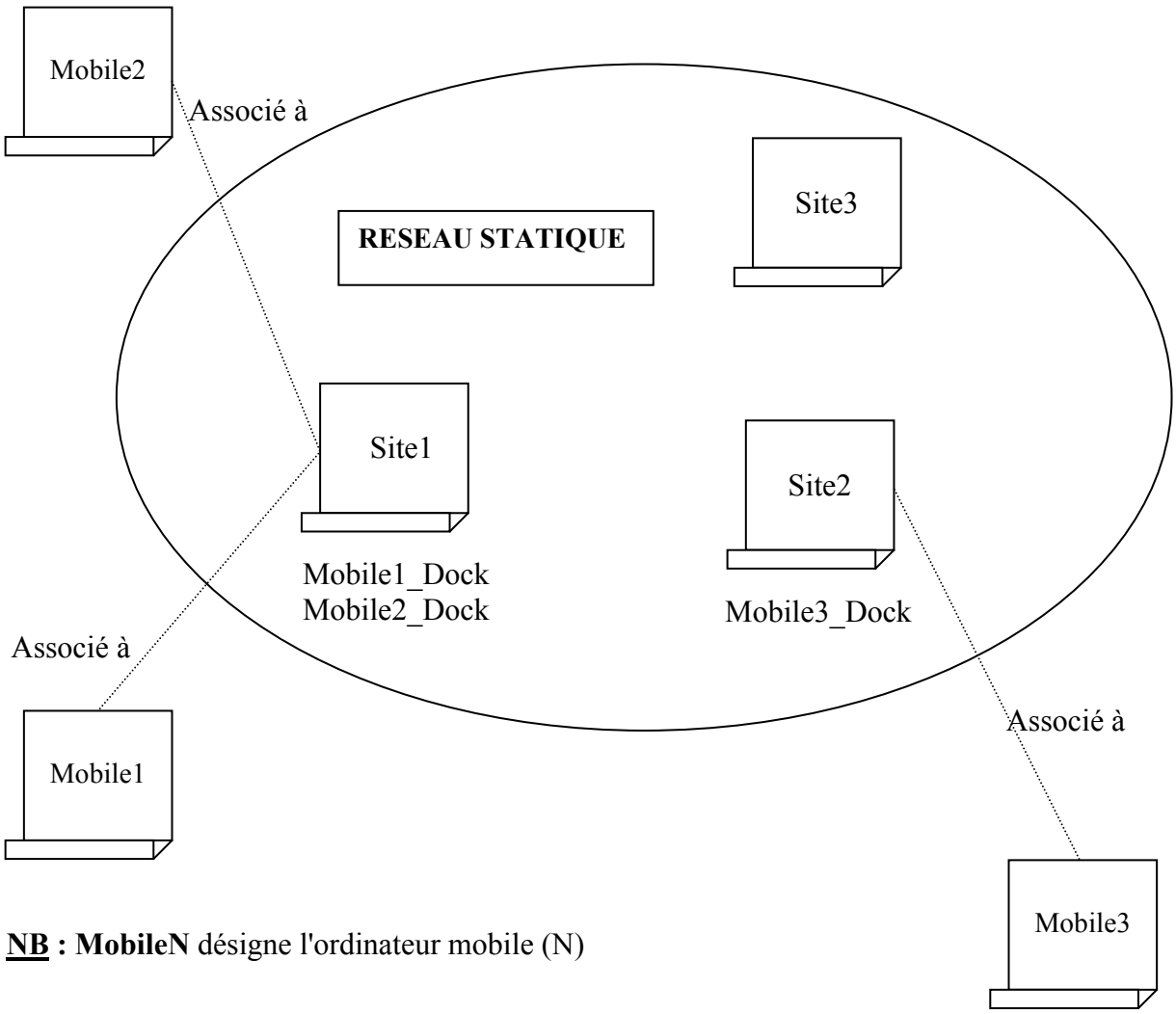
[١] Dans cet exemple, la tâche de l'agent mobile ne fait intervenir que deux sites (site1 et site2); c'est pour cela, que l'agent mobile ne se sert pas de l'agent moniteur du trafic et des outils permettant de tester l'état des connexions.
[٢] Par ordinateur mobile, on désigne aussi l'ordinateur connecté par modem. Ce dernier peut, en effet, avoir des périodes de déconnexion et peut également observer des changements sur son adresse réseau.

Une solution proposée pour régler aussi les points 1 et 4, ce qui permettra de supporter les ordinateurs mobiles et d'étendre, par conséquent, l'infrastructure SINDBAD à un environnement mobile, est l'approche dite des Docks. [GKNRC96]

L'APPROCHE DES DOCKS

Cette approche associe chaque ordinateur mobile nommé D à un site fixe connecté de façon permanente dans le réseau. Ce site jouera le rôle de DOCK pour D et sera appelé, par convention, D_DOCK.
Les techniques de désignation dans les réseaux permettent généralement d'associer plusieurs ordinateurs mobiles à un seul site connecté de façon permanente; par conséquent, seulement quelques sites du réseau vont servir comme DOCKs.

Exemple:
Dans cet exemple, le site1 joue le rôle de DOCK pour mobile1 et mobile2, le site2 joue le rôle de DOCK pour mobile3, alors que le site3 ne joue aucun rôle de DOCK.



NB : MobileN désigne l'ordinateur mobile (N)

Figure III-II-6: L'approche des DOCKs

Sur chaque ordinateur mobile et sur tout site jouant un rôle de DOCK, on rajoute un agent **stationnaire**: Agent Dock_master (ou agent gestionnaire du DOCK).

Pour comprendre le rôle de l'agent Dock_master, considérons un agent qui veut se déplacer d'un site fixe nommé C vers un ordinateur mobile déconnecté nommé D. Pour se faire, cet agent mobile exécute la commande de migration vers D. A la fin de cette commande, il doit se retrouver sur D; et ce, d'une manière transparente.

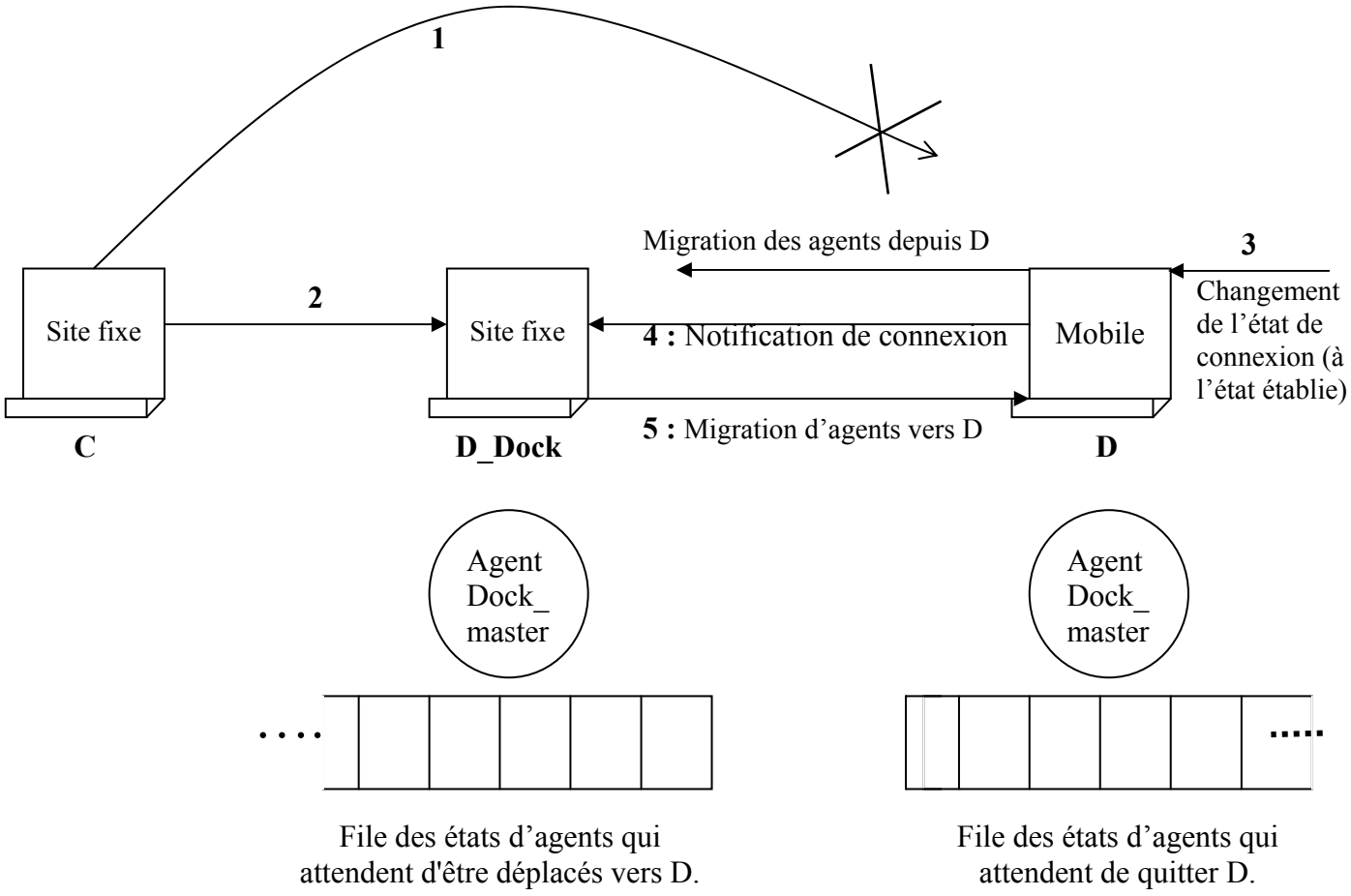


Figure III-II-7: Migration d'un agent entre un site fixe et un ordinateur mobile.

Le processus de migration de l'agent mobile de C vers D s'effectue de la manière suivante:

١. L'environnement d'exécution capture l'image courante de l'agent et la passe au serveur d'agents qui essaye de contacter D et échoue car D est déconnecté.
٢. Il essaye alors de contacter l'agent Dock_master sur le site D_Dock (jouant le rôle de Dock pour D). L'image de l'agent sera transférée sur D_Dock, mais l'agent mobile ne sera pas restauré, son image restera sauvegardée sur le disque de D_Dock sous le contrôle de l'agent Dock_master de D_Dock.
٣. Quand D se connecte,
٤. Son agent Dock_master contacte l'agent Dock_master de D_Dock. et lui communique la nouvelle adresse de D.
٥. Tous les agents en attente sur D_Dock peuvent alors être transférés vers D, où ils seront restaurés pour continuer leur exécution.

Si par la suite, un autre agent veut se déplacer vers D en spécifiant l'ancienne adresse de D, il échouera et son image courante sera transférée vers D_Dock pour ensuite atteindre D à sa nouvelle adresse.

Considérons maintenant un agent qui veut quitter un ordinateur mobile déconnecté nommé D. L'environnement d'exécution capture alors l'image courante de l'agent et la passe

au serveur d'agents. Ce dernier détecte que D est déconnecté et, par conséquent, sauvegarde l'image capturée de l'agent mobile sur son disque, puis prévient l'agent Dock_master local. L'agent Dock_master observe continuellement[`] l'état de connexion de D au réseau (en utilisant les outils de test de l'état des connexions); quand D est connecté, il transfert immédiatement tous les agents en attente vers leurs destinations où ils peuvent continuer leur exécution.

Considérons maintenant un cas plus compliqué : Un agent qui veut se déplacer d'un site S vers un autre site D, où S et D sont deux ordinateurs mobiles.

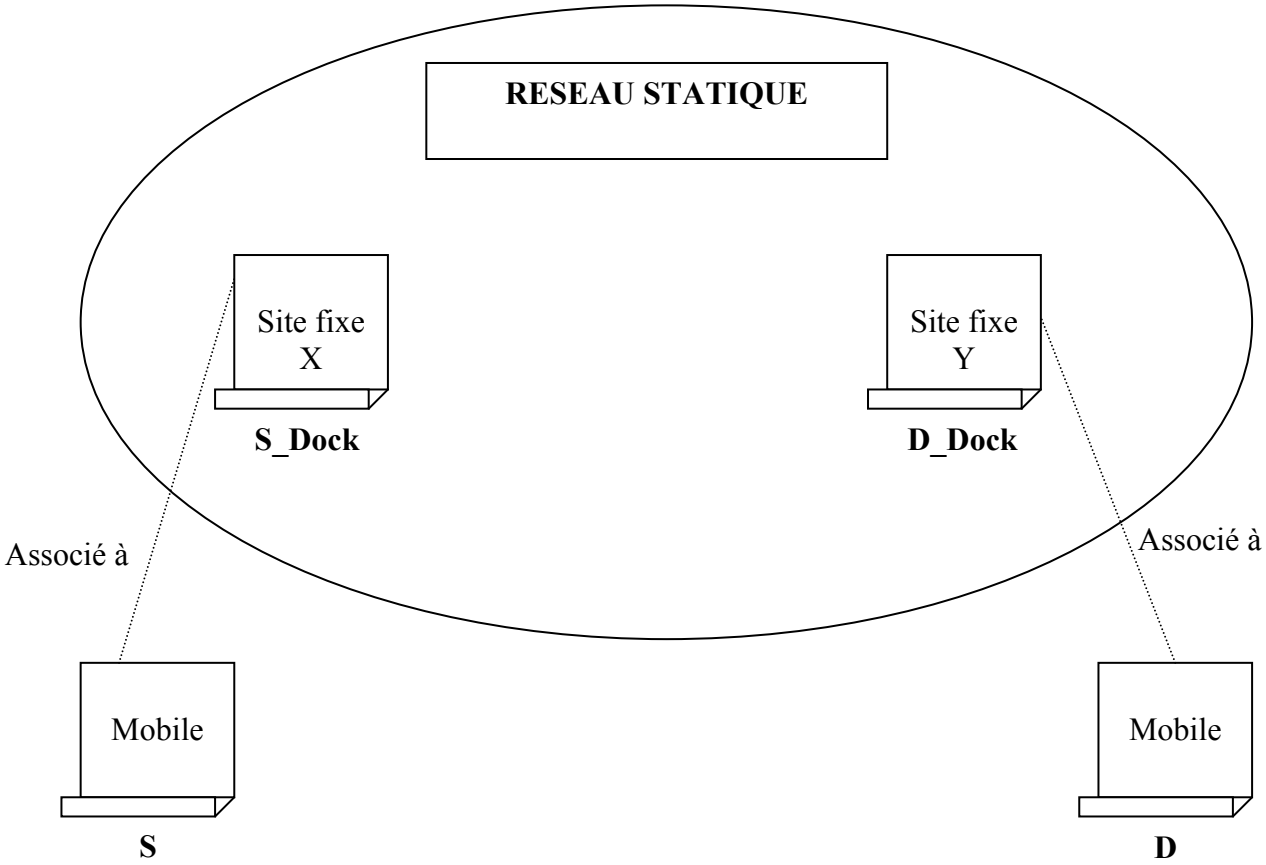


Figure III-II-8: Migration d'un agent entre deux ordinateurs mobiles

On peut facilement imaginer le cas où S et D ne se connectent jamais en même temps, rendant, par conséquent, une migration directe impossible. Dans ce cas, le scénario de migration sera le suivant:

- L'image courante de l'agent mobile est capturée et restera stockée sur le disque de S jusqu'à ce que l'agent Dock_master de S détecte une connexion au réseau.
- Une fois S connecté, son agent Dock_master essayera de contacter D; s'il échoue, il essayera de contacter D_Dock. Si D_Dock est non atteignable, peut être à cause d'un problème temporaire dans le réseau, il essayera alors de contacter S_Dock. Si S_Dock est aussi non atteignable, l'agent Dock_master de S effectuera une autre tentative ultérieurement.
  Par contre, si S_Dock est atteignable, l'agent Dock_master de S lui transférera l'image capturée de l'agent mobile.

[`] Une attente active peut être évitée si le système d'exploitation peut générer une interruption à chaque fois que la connexion locale au réseau change d'état (établie/ non établie).

- A son tour, l'agent Dock_master de S_Dock essayera périodiquement de retransmettre l'image de l'agent mobile vers D, sinon vers D_Dock.
- Cette image peut résider éventuellement sur D_Dock jusqu'à ce que D se connecte et communique à l'agent Dock_master de D_Dock sa nouvelle adresse. Une fois sur D, l'agent mobile est restauré à partir de son image pour continuer son exécution.

D'après ce qu'on vient de voir, l'approche des DOCKs présente les avantages suivants:
- Les agents qui attendent la connexion pour aller vers (ou pour quitter) un ordinateur mobile ont leurs images capturées et sauvegardées sur le disque, libérant ainsi la mémoire et le processeur, et résistant mieux aux pannes et à la mise hors tension.
- Le transfert des images sauvegardées s'effectuera **rapidement** dès qu'une connexion avec le réseau est établie.

Augmenter alors l'infrastructure SINDBAD présentée pour un environnement fixe d'un système de DOCKs, permettra de l'étendre à un environnement mobile. En plus, les agents voulant se déplacer de (ou vers) les ordinateurs mobiles peuvent le faire rapidement, en n'exigeant que de courtes durées de connexion.

5. COMMUNICATIONS D'UN AGENT MOBILE

Pour accomplir sa tâche, un agent mobile a besoin en général de communiquer avec d'autres agents (stationnaires ou mobiles), des utilisateurs et le site sur lequel il s'exécute.

5.1. COMMUNICATIONS AVEC UN AUTRE AGENT

Un agent mobile peut avoir besoin d'un service qu'un autre agent délivre; de même, il peut publier des informations pour l'intérêt d'autres agents. Un mécanisme de communication entre agents est alors nécessaire.

L'infrastructure SINDBAD que nous avons présentée supporte différents langages d'écriture d'agents. Ceci offre aux utilisateurs l'avantage d'une multitude de choix en ce qui concerne l'implémentation des agents; cependant, une question évidente sera de savoir: comment les différents agents arriveront-ils à communiquer entre eux?
Si l'infrastructure n'était dédiée qu'à un seul langage d'écriture d'agents, par exemple Tcl, les agents peuvent communiquer entre eux en utilisant la commande *Send* de Tcl [LD96]; cependant, puisqu'on peut avoir des agents écrits avec différents langages, il devient nécessaire d'avoir une abstraction d'un moyen de communication qui soit indépendante des langages.

Il sera très intéressant si cette abstraction: [LD96]
a) Permet le passage de requêtes entre les agents,
b) Permet à un agent de publier des informations pour l'intérêt d'autres agents, et
c) Supporte les différents styles d'interaction entre les agents, à savoir: un-à-un, un-à-plusieurs et plusieurs-à-plusieurs.

Pour permettre la communication entre des agents écrits dans différents langages et satisfaire aussi aux contraintes a), b) et c), nous avons adopté l'approche utilisée dans l'infrastructure FFMAIN: Il s'agit de l'approche de l'espace d'information. [LDD95]

ESPACE D'INFORMATION

L'approche de l'espace d'information est inspirée de LINDA[`] [LD96]. Il s'agit d'une abstraction d'un moyen de communication indépendante des langages.
Chaque serveur d'agents maintient un espace d'information local, qui est accessible aux différents agents s'exécutant sous son contrôle (c'est-à-dire, existant sur son site).

L'espace d'information est organisé en un ensemble d'enregistrements, où chaque enregistrement est un triplet de la forme (C,A,V), tel que:

- C: est une clé qui désigne un enregistrement de manière unique.
- A: est une liste (de critères, par exemple) précisant les agents qui peuvent accéder à l'enregistrement. Elle peut donc autoriser l'accès à l'enregistrement à des agents spécifiques, à un groupe d'agents ou à tous les agents. [LD95]
  Le serveur d'agents contrôle l'accès des agents aux enregistrements de l'espace d'information en se basant sur les critères spécifiés dans les champs (A) des enregistrements.
- V: est une valeur de contenu et de taille arbitraires.
  Le serveur d'agents doit, cependant, empêcher que la taille d'un enregistrement donné provoque un débordement dans l'espace d'information.

Les différents agents de l'infrastructure SINDBAD peuvent communiquer entre eux à travers l'espace d'information; et ce, à l'aide de trois opérations de base:

- WRITE(C,A,V) qui ajoute l'enregistrement (C,A,V) dans l'espace d'information.
- READ (C) qui lit à partir de l'espace d'information l'enregistrement désigné par la clé (C), et
- DREAD(C) (ou Destructive READ) qui lit l'enregistrement désigné par la clé (C), puis le supprime de l'espace d'information.

Pour préserver la consistance de l'espace d'information, les opérations WRITE, READ et DREAD doivent être atomiques. [LD96]

D'une manière évidente, un agent doit parcourir l'espace d'information pour rechercher les enregistrements auxquels il s'intéresse; cependant, une alternative intéressante peut consister à l'informer dès qu'un enregistrement qui l'intéresse est produit dans l'espace d'information.

[`] Le langage LINDA adopte une approche similaire à l'espace d'information: il utilise, en effet, un ensemble de tuples de tailles arbitraires pour coordonner le calcul parallèle. [LD96]

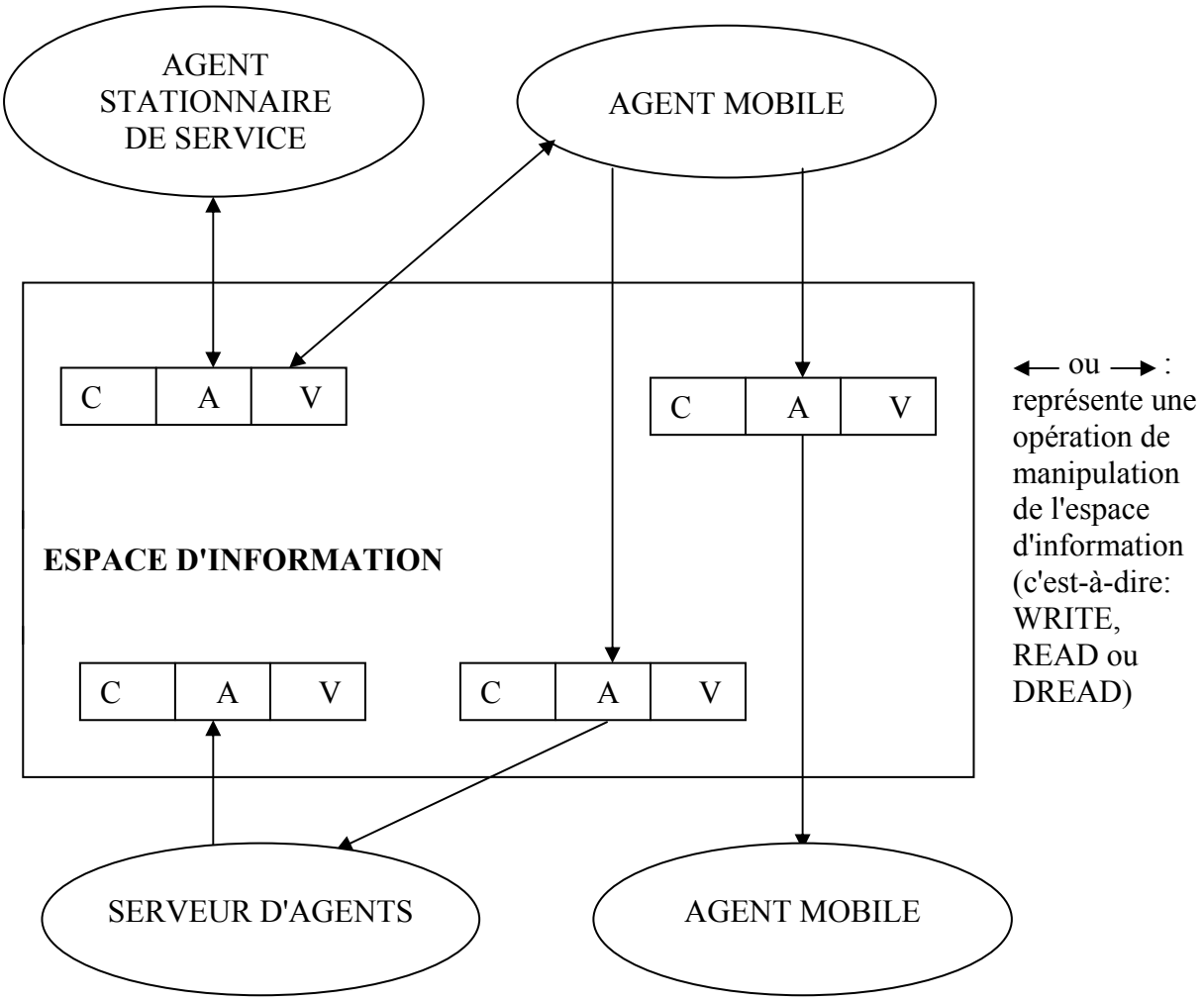


Figure III-II-9: Communications entre les agents à travers l'espace d'information

L'approche de l'espace d'information permet d'organiser différents styles d'interaction entre les agents: du simple avec un schéma "request-response" (style d'interaction un-à-un) aux sophistiqués avec des schémas tels que "diffusion" (style d'interaction un-à-plusieurs) et "blackboard systems" (style d'interaction plusieurs-à-plusieurs). [LD96]

L'espace d'information reste, cependant, un mécanisme de communication de bas niveau. En effet, il n'interprète pas les informations (ou enregistrements) qui transitent par lui. Il s'ensuit que les développeurs d'agents doivent connaître suffisamment d'informations sur les agents avec lesquels leurs agents vont interagir. Ils ont besoin, pour pouvoir communiquer effectivement, de connaître, en fait, les interfaces des autres agents (signatures des opérations, protocoles utilisés et format des données). [LDD95]
Pour faire éviter aux développeurs d'agents une telle difficulté, des protocoles de haut niveau (relevants du domaine de l'I.A.[1]) peuvent être utilisés pour structurer les communications des agents à travers l'espace d'information. Ceci inclut: [LD96]

- Un langage commun pour la représentation des informations (par exemple, KIF "Knowledge Interchange Format").
- Un langage commun pour la manipulation des informations (par exemple, KQML "Knowledge Query and Manipulation Language"), et

[1] I.A.: Intelligence Artificielle

- Des ontologies communes (qui assurent qu'entre les agents, les même mots signifient les mêmes choses).

D'après ce qu'on vient de voir, l'infrastructure SINDBAD dispose donc (pour supporter la communication entre ses agents) d'un espace d'information au niveau de chaque site pouvant exécuter des agents. Chaque espace d'information est maintenu par le serveur d'agents local, et est accessible uniquement aux agents s'exécutant sur le site.
On a, évidemment, pensé à généraliser ce moyen de communication pour offrir plutôt un espace d'information global, au lieu d'un ensemble d'espaces d'information locaux; cependant d'après [LD95], un tel espace d'information global ne pourra pas grandir arbitrairement sans perdre son efficacité; en plus, il n'est pas clair du tout qu'étendre ce modèle de communication dans ce sens sera plus bénéfique, étant donné que les agents peuvent, en fait, se déplacer facilement entre les sites.

Il y a à noter, finalement, que le fait d'adopter l'approche de l'espace d'information pour supporter la communication entre les agents, ne va pas, cependant, empêcher ces derniers (qui sont supposés être intelligents) d'utiliser d'autres moyens de communication. Par exemple, si un transfert de données à haut débit est désiré, les agents peuvent utiliser des connexions directes[1] ou même des mémoires partagées. [LD95]
Rajouter d'autres moyens de communication peut s'effectuer au niveau des environnements d'exécution [LD95], laissant, par conséquent, SINDBAD demeurer une infrastructure de bas niveau.

La figure suivante illustre l'architecture de l'infrastructure SINDBAD pour un environnement mobile. Dans cette figure, le site1 joue le rôle de dock pour mobile1et mobile2, le site2 joue le rôle de dock pour mobileM, alors que, le siteN ne joue aucun rôle de dock.

[1] Une connexion directe est, rappelons-le, un flot de messages nommé. (revoir la section 4-1 du chapitre III - Partie I)

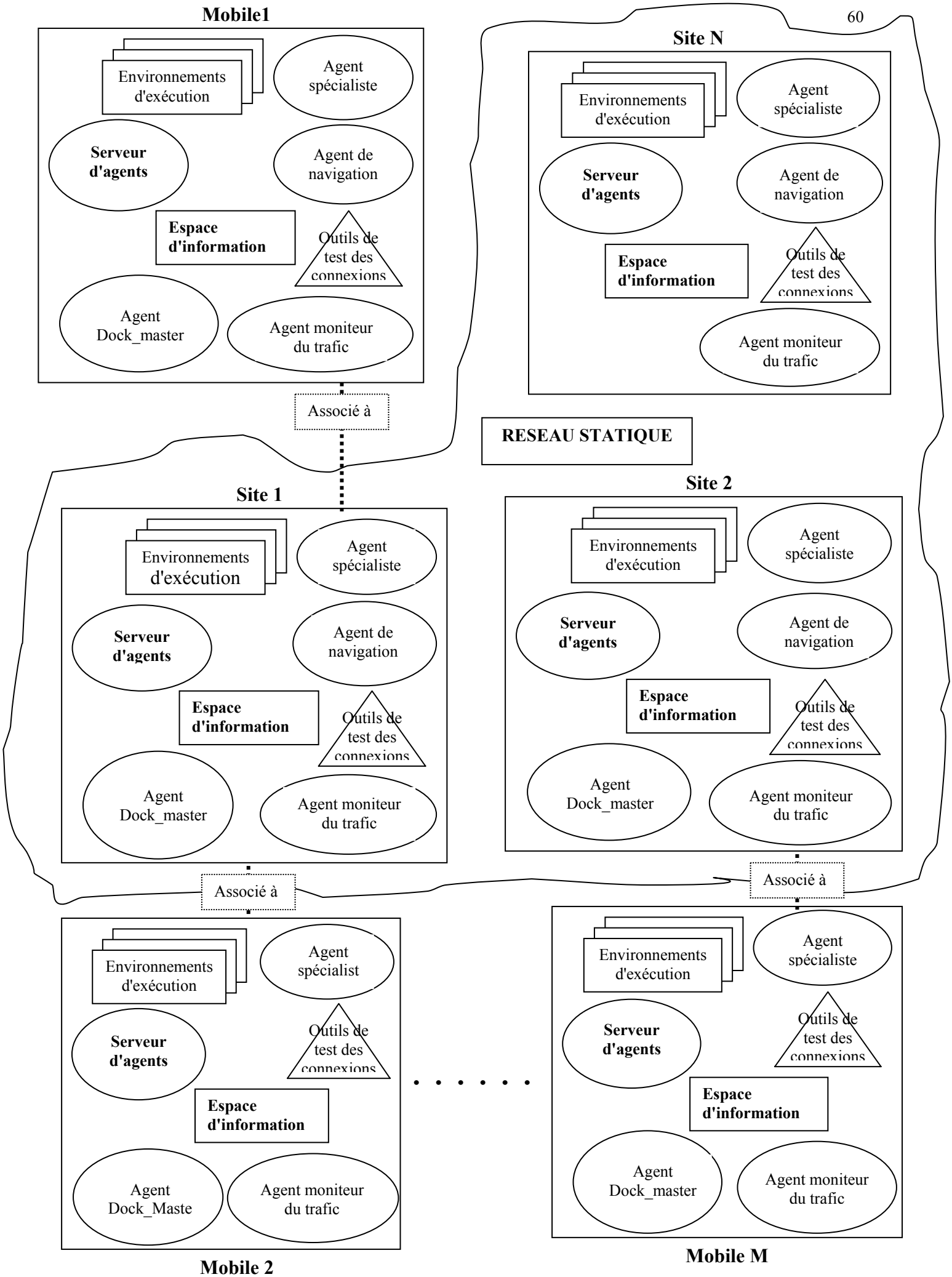


Figure III-II-10: Architecture de l'infrastructure SINDBAD pour un environnement mobile

5.2. COMMUNICATIONS AVEC UN UTILISATEUR

Dans une infrastructure d'agents, un utilisateur a besoin de communiquer avec des agents et un agent a besoin de communiquer avec des utilisateurs.

a) COMMUNICATIONS UTILISATEUR→AGENT

Un utilisateur a besoin de communiquer avec son agent: [LD98c]
- Pour lui affecter au lancement une tâche à réaliser (c'est-à-dire, définir sa mission),
- Pour le contrôler (consulter son état d'exécution, arrêter son exécution, reprendre son exécution, mettre fin à son exécution, etc.), ou
- Pour répondre aux requêtes que ce dernier peut formuler.

Il peut aussi communiquer avec un agent (autre que le sien), si ce dernier demande sa collaboration.

Les communications Utilisateur→Agent peuvent s'effectuer à travers un client (ou interface) que l'utilisateur utilise pour lancer son agent et interagir avec l'infrastructure d'agents. Le client ne nécessite pas une connexion permanente au reste de l'infrastructure et peut donc résider sur un ordinateur mobile.

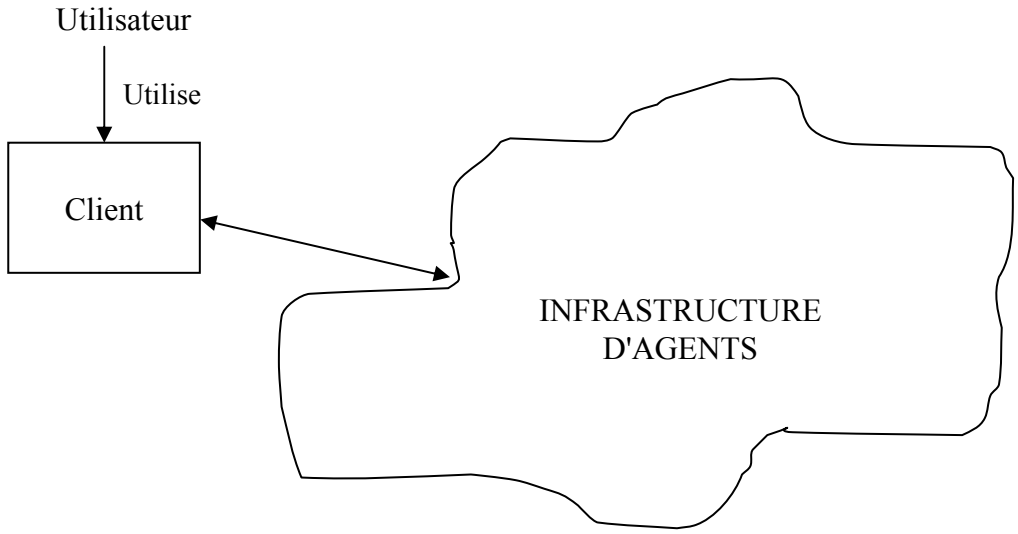


Figure III-II-11: Interactions de l'utilisateur avec l'infrastructure SINDBAD

Notons que l'ordinateur (ou l'unité) mobile dont se sert un utilisateur peut être trop modeste (un modèle avec accès distant[1], par exemple) et, par conséquent, ne pourra pas supporter les composants décrits sur la figure III-II-10. Dans ce cas, il se contentera de contenir uniquement le client permettant à l'utilisateur d'interagir avec l'infrastructure SINDBAD.

b) COMMUNICATIONS AGENT→UTILISATEUR

Un agent a besoin de communiquer avec son propriétaire: [LD98c]
- Pour lui reporter les **résultats** (intermédiaires ou finales) de son activité.
- Pour lui reporter les messages d'erreurs si des problèmes sont rencontrés. (les messages d'erreur peuvent être considérés comme des **résultats**), ou
- Pour demander son aide; par exemple, quand il négocie sur un service donné.

[1] Revoir la section 3.2 du chapitre I.

L'agent formule dans ce cas une **requête** et attend la réponse de son propriétaire.
En plus, dans des applications de type : questionnaire, pipeline, ordonnancement de rendez-vous, etc., un agent a besoin de communiquer avec un utilisateur autre que son propriétaire. Il formule pour cela une **requête** et attend la réponse de l'utilisateur.

Dans ce qui suit, nous examinons comment les résultats et les requêtes peuvent être communiqués dans une infrastructure d'agents.

Envoi des résultats

Un agent mobile peut envoyer ses résultats de différentes manières: [LD98c]

- Il peut revenir à son propriétaire et présenter localement ses résultats.
  Avantages:
  - Meilleur contrôle sur la présentation des résultats.

  Inconvénients:
  - Inefficace pour les grands agents produisant de petits résultats.
  - Difficile pour les résultats intermédiaires.
  - En plus, l'agent doit rester actif jusqu'à ce qu'il présente ses résultats.
- Il peut envoyer ses résultats à travers l'E-Mail.
  Avantages:
  - C'est une méthode de communication convenable.
  - Le cas d'un ordinateur éteint ou déconnecté est pris en charge automatiquement.
  - Elle fonctionne très bien pour les résultats intermédiaires.
  - Elle ne nécessite qu'un petit effort pour être supportée.

  Inconvénients:
  - L'E-Mail utilise une interface d'utilisateur différente de celle qu'utilise généralement une infrastructure d'agents.
- Il peut envoyer ses résultats à travers l'infrastructure d'agents (par exemple, les résultats seront déposés dans des endroits gérés par l'infrastructure d'agents).
  Inconvénients:
  - L'infrastructure d'agents doit être accessible pour pouvoir consulter les résultats.
- Il peut utiliser WWW[`] (World Wide Web) pour communiquer ses résultats.
  Avantages:
  - Les navigateurs WWW sont ubiquistes.
  - Cette méthode est bonne aussi pour les résultats intermédiaires ainsi que les requêtes, et
  - Facilite l'intégration avec les autres services basés sur WWW.

Envoi des requêtes

Pour communiquer sa requête à un utilisateur, l'agent la dépose dans l'espace d'information, puis attire l'attention de l'utilisateur avec un message d'E-Mail. L'utilisateur consulte la requête à l'aide du client qu'il utilise, construit sa réponse et la dépose dans l'espace d'information à l'attention de l'agent. [LD98c]

[`] Voir le chapitre suivant.

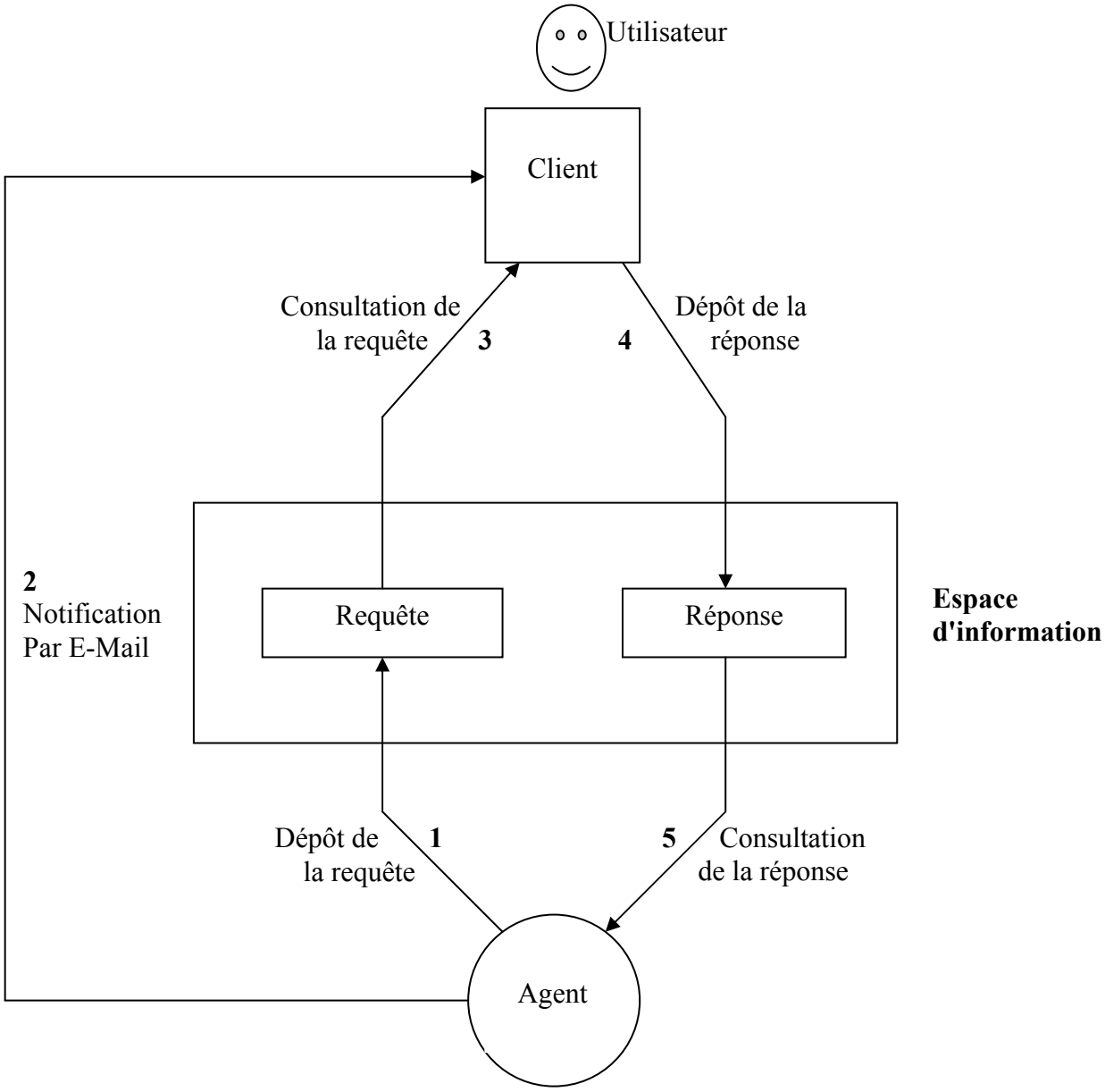


Figure III-II-12: Envoi des requêtes

5.3. COMMUNICATIONS AVEC LE SITE

Un agent a besoin de communiquer avec le site (ou l'hôte) sur lequel il s'exécute pour solliciter des informations ou pour bénéficier des services offerts par d'autres systèmes pouvant exister sur le site (par exemple, les services offerts par un serveur Web). Ce type de communication peut être réalisé de la manière suivante:

- Le serveur d'agents peut publier pour l'intérêt des agents visiteurs, à travers l'espace d'information, des informations relatives au site local. Par exemple, il peut publier la liste des agents s'exécutant actuellement sur le site; cette liste peut être utilisée, par exemple, par un agent mobile cherchant un autre agent.
- Les services offerts par les différents systèmes existant sur le site peuvent être rendus accessibles aux agents mobiles à travers des agents stationnaires. Ces derniers peuvent être développés localement et auront, par conséquent, un haut niveau de confiance contrairement aux agents mobiles qui sont des programmes arbitraires. La communication entre les agents mobiles et les agents stationnaires s'effectue, pour des raisons de sécurité, à travers l'espace d'information.

Exemple:
Soit un système de base de données bibliographiques existant sur un site pouvant exécuter des agents mobiles. Pour rendre ce système accessible aux agents mobiles, on développe un agent stationnaire qui a la permission de lire les fichiers de la BDD. Cet agent reçoit les requêtes exprimées par les agents mobiles à travers l'espace d'information, accède directement à la BDD locale et renvoie les résultats aux demandeurs par l'intermédiaire de l'espace d'information.

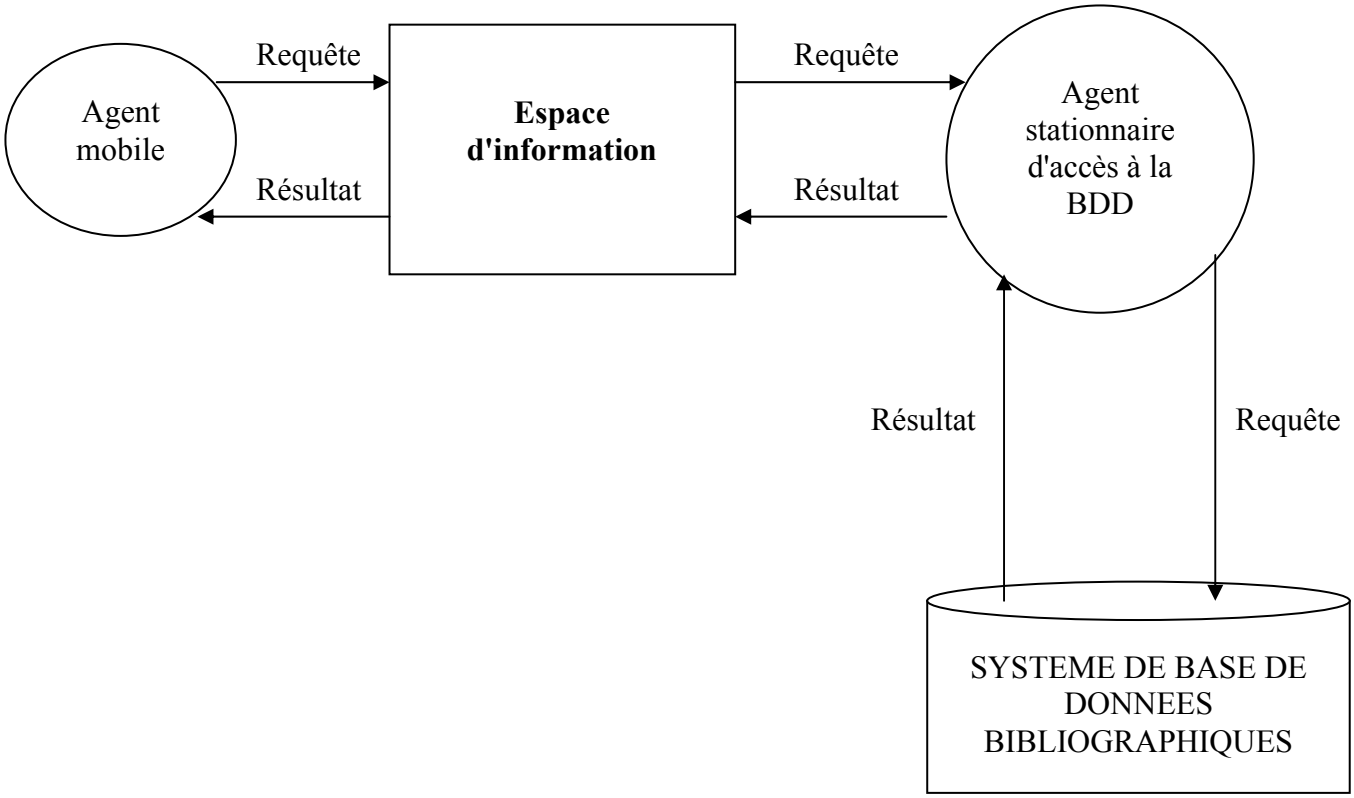


Figure III-II-13: Communications d'un agent mobile avec un autre système

Un agent mobile peut être autorisé à accéder directement aux informations et services disponibles sur le site au lieu d'être restreint à utiliser uniquement l'espace d'informations. Ceci est dangereux du point de vue "sécurité"; mais la finesse des contrôles offerte (dans le but d'assurer la sécurité des exécutions) par certains langages, tel que Tcl, peut rendre cette approche acceptable dans certains cas [LDD96]. Par exemple, si tous les agents mobiles sont développés exclusivement par les opérateurs du système, alors autoriser plus d'accès aux agents mobiles améliorera l'efficacité du système. Un contexte d'utilisation peut être la gestion des réseaux [MNK98].

Le tableau suivant résume les différents types de communication d'un agent mobile: [LD98c]

Communique avec	UTILISATEUR	AGENT	SITE[1] (HOTE)
UTILISATEUR	Au-delà du contexte de cette thèse	• Pour définir sa mission • Pour contrôler son exécution (consulter son état, reprendre son exécution, mettre fin à son exécution, etc.) • Pour répondre à ses requêtes	Au-delà du contexte de cette thèse
AGENT	• Pour lui envoyer les résultats (intermédiaires ou finales). • Pour lui envoyer des messages d'erreur. • Pour lui envoyer des requêtes.	• En publiant des informations • Pour lui demander des services (requêtes). • Pour répondre à ses requêtes.	• Pour consulter des informations. • Pour bénéficier des services.

Tableau III-II-1: Types de communication d'un agent mobile

٦. LA SECURITE DANS UNE INFRASTRUCTURE D'AGENTS MOBILES

le problème de la sécurité est, peut être, le plus important problème qui permet de déterminer si un système d'agents mobiles sera accepté ou non [LD95]. En fait, plusieurs systèmes d'agents mobiles limitent l'ensemble des applications réalisables par le fait qu'ils n'assurent pas une sécurité suffisante[Gra95a].

Les problèmes de sécurité pouvant être rencontrés dans un système d'agents mobiles concernent: [Bou99a, Bou99b]

a)- ***L'authentification:***
le serveur d'agents, avant d'accepter d'exécuter ou d'héberger un agent, doit pouvoir authentifier l'autorité de cet agent. Réciproquement, l'agent doit pouvoir authentifier le serveur d'agents qui l'exécute.
Dans l'environnement Aglet, par exemple, l'authentification d'un agent (ou aglet) se fait par l'authentification de son serveur d'agents d'origine (ou natal). En effet, un système d'authentification des serveurs d'agents appartenant à un même domaine est défini. Un

[1] Si les informations et services disponibles sur le site sont accessibles à travers des agents stationnaires, on peut supprimer cette colonne, car son cas sera inclut dans la colonne qui la précède.

domaine est un ensemble de serveurs d'agents supervisés par une autorité. Les serveurs d'agents peuvent s'authentifier les uns des autres en partageant une clé secrète. Une fois l'authentification entre deux serveurs d'agents est établie. Les credentials (ou attributs) de l'aglet (ou agent) sont envoyés avec elle. Le serveur d'agents fera confiance à cette aglet si, en consultant ses credentials, trouve qu'elle est envoyée à partir d'un serveur d'agents du même domaine que lui.

b)- *Le contrôle d'accès et l'intégrité des sites:*

L'infrastructure SINDBAD doit protéger l'ensemble des sites des accès non autorisés que les agents mobiles peuvent effectuer.

- On a déjà vu comment protéger les différents systèmes pouvant exister sur un site: en autorisant un agent mobile à manipuler uniquement l'espace d'information et en offrant les différents informations et services pouvant exister sur le site à travers des agents stationnaires (à qui on fait confiance). Cette façon de faire permet, en effet, à l'administrateur de l'infrastructure de retenir un maximum de contrôle sur l'usage des ressources.
- En plus, un agent mobile ne doit pas violer la sécurité de l'infrastructure en piratant, par exemple, les fichiers de mots de passe, en formatant le disque dur principal ou en altérant les fichiers spécifiant la politique de la gestion des ressources locales. Pour cela, les opérations dangereuses telles que: l'exécution arbitraire des commandes du système d'exploitation, l'ouverture de fichiers ou l'ouverture de connexions réseau doivent être extrêmement contrôlées.

  Les environnements d'exécution associés aux différents langages d'écriture d'agents peuvent accomplir ces contrôles en analysant les arguments des opérations dangereuses pour n'autoriser que les opérations saines; ou dans une autre alternative, peuvent interdire complètement ces opérations.

c)- *L'intégrité des agents mobiles*[\]*:*

l'intégrité d'un agent mobile doit être préservée des actions malintentionnées que peuvent effectuer sur lui les sites visités ou d'autres agents. En plus, un système d'agents doit garantir l'intégrité des messages échangés entre les agents.

L'infrastructure SINDBAD réalise ceci:

- En s'assurant que seul le propriétaire peut contrôler l'exécution de son agent (consulter les détails importants de l'agent, reprendre son exécution, mettre fin à son exécution, etc.).

  Une solution simple peut être "Magic cookies", où une chaîne longue et aléatoire (faisant partie des attributs de l'agent) doit être présentée par le propriétaire au serveur d'agents pour pouvoir accéder au contrôle de son agent.
- En contrôlant les accès aux enregistrements de l'espace d'information.

  Ce type de contrôle est effectué par le serveur d'agents en se basant sur les critères spécifiés dans les champs (A) des enregistrements de l'espace d'information, et
- En intégrant (de manière facile et directe) les solutions élaborées par la communauté Web, particulièrement, en ce qui concerne l'intégrité des agents et des messages transférés à travers le réseau.

d)- *La confidentialité:*

Il est très difficile de garder le caractère secret d'un agent qui doit visiter plusieurs serveurs d'agents. Pour de tels besoins de sécurité, il va falloir restreindre les capacités des agents

[\] Les systèmes d'agents mobiles basés sur Java profitent de la sécurité intrinsèque du langage et de la machine virtuelle. Les mécanismes mis en œuvre (tels que, le gestionnaire de sécurité et la vérification de bytecode) assurent l'intégrité du système hôte et l'intégrité des objets (ou agents) vis-à-vis des tentatives d'accès à des données privées.

mobiles, notamment, leur liberté de mouvement. Mais, cette dernière n'est-elle pas une de leurs caractéristiques les plus intéressantes?
Généralement, l'approche la plus employée pour conserver la confidentialité des agents durant leur transfert consiste à utiliser des techniques de cryptographie. En plus, les agents peuvent être également codés durant leur stockage sur les supports de masse (non volatiles).

7. CONCLUSION

SINDBAD, l'infrastructure présentée à travers ce chapitre, permet de supporter les agents mobiles dans un environnement mobile. C'est une infrastructure assez complète et assez générale, conçue dans le but de réaliser des applications distribuées quelconques. Elle offre une grande flexibilité aux agents, en leur permettant:

- D'être écrits avec différents langages d'écriture d'agents.
- De communiquer à travers un mécanisme de bas niveau (qui est l'espace d'information), supportant différents styles d'interaction entre les agents (un-à-un, un-à-plusieurs et plusieurs-à-plusieurs) et sur lequel peuvent être organisées des communications plus structurées en utilisant des protocoles de haut niveau, tels que KIF (Knowledge Intechange Format) et KQML (Knowledge and Query Manipulation Language). En plus, les agents peuvent, si nécessaire, utiliser d'autres moyens de communication qui peuvent alors être rajoutés.
- D'élaborer des plans de navigation adaptatifs, et
- D'interagir avec les différents systèmes pouvant exister sur les différents sites de l'infrastructure.

Le chapitre suivant examine comment rendre l'infrastructure SINDBAD complètement basée sur HTTP afin qu'elle puisse devenir familière, plus convenable et largement acceptée.

CHAPITRE IV

SINDBAD BASE SUR HTTP

1. INTRODUCTION

L'infrastructure SINDBAD que nous proposons supporte différents langages d'écriture d'agents; par conséquent, les agents mobiles et stationnaires peuvent être écrits avec différents langages. Ces agents ont besoin de communiquer et éventuellement de se déplacer. Pour cela, le serveur d'agents s'en charge de fournir les services de communication (à travers l'espace d'information) et de transfert (en utilisant un protocole de transfert donné).

Un agent qui est écrit dans un langage différent du langage d'écriture du serveur d'agents, accède aux services de communication et de transfert à travers une interface (ou API[1]: Application Programming Interface) offerte par son environnement d'exécution. Cette interface permet d'appeler, à partir du langage d'écriture de l'agent, les primitives de communication et de transfert fournies par le serveur d'agents.
Réaliser des interfaces pour les différents langages d'écriture d'agents supportés nécessitera, cependant, un effort considérable.

[1] Revoir la section 3.2 du chapitre III - Partie II.

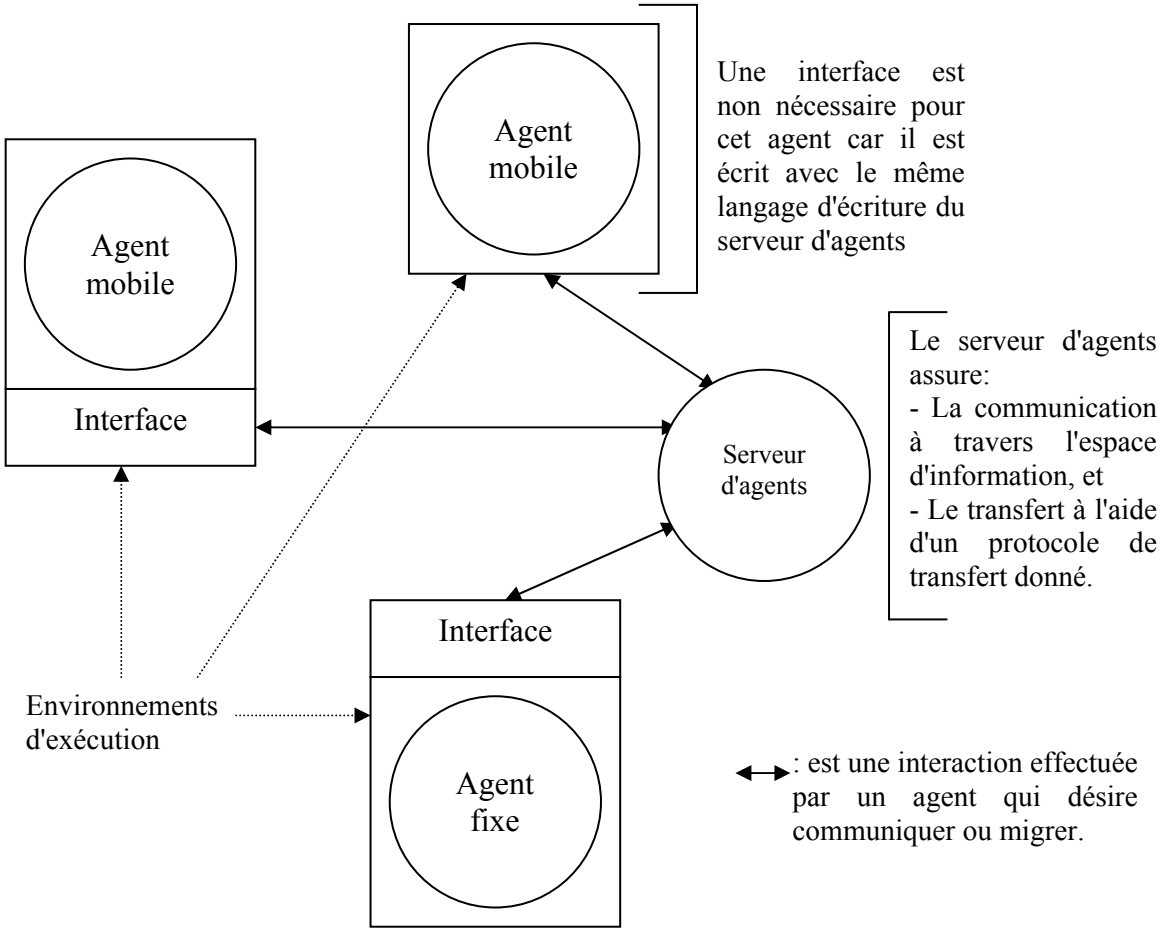


Figure IV-1: Nécessité des interfaces dans les interactions des différents agents avec le serveur d'agents.

Remarquons que pour être plus utiles, "les systèmes d'agents permettent à leurs agents de consulter des ressources sur WWW ou encore de fournir de telles ressources eux même. Il s'ensuit alors que les mécanismes nécessaires pour interagir à l'aide de HTTP sont déjà en place dans plusieurs systèmes d'agents mobiles". [DLD97]
HTTP (HyperText Transfert Protocol) est un standard d'Internet pour l'accès à l'information sur WWW (World Wide Web). [DLD97]

Puisque la plupart des langages d'écriture d'agents supportent HTTP, ce dernier peut donc servir comme plate-forme commune permettant d'effectuer les interactions entre les différents agents d'une part et le serveur d'agents de l'autre part. De cette façon, il n'est plus nécessaire d'intégrer des interfaces au niveau des environnements d'exécution[1].
HTTP peut, en plus, être utilisé pour transférer les agents mobiles. Ceci permettra de bénéficier des résultats de recherches intenses qui s'effectuent continuellement sur WWW, particulièrement, dans le domaine de la sécurité.
Finalement, la plupart des utilisateurs sont déjà familiers avec la technologie WWW; ils apprécieront alors l'idée de leur présenter les informations d'un système d'agents mobiles à travers les techniques basées sur HTTP et, par conséquent, accepteront plus facilement l'approche d'agents mobiles.

[1] Un environnement d'exécution sera constitué, par conséquent, d'un interpréteur, d'un module d'état et d'un module de sécurité. L'API n'est, en effet, plus nécessaire, puisque HTTP va être utilisé pour supporter les interactions des différents agents avec le serveur d'agents.

Les avantages découlant de l'usage de HTTP rendent donc plus intéressante, l'idée de concevoir une infrastructure d'agents mobiles complètement basée sur HTTP. Par conséquent, ce chapitre va examiner comment mettre en œuvre cette idée avec l'infrastructure SINDBAD. Il s'agit, en fait, de concevoir le serveur d'agents comme un serveur Web.
Dans ce qui suit, nous commençons alors par décrire comment concevoir le serveur d'agents comme un serveur Web, nous présentons ensuite, avec plus de détails, les avantages qui peuvent résulter et nous terminons par discuter l'interopérabilité des agents appartenant à des infrastructures hétérogènes.

2. CONCEVOIR LE SERVEUR D'AGENTS COMME UN SERVEUR WEB

Rappelons que sur chaque site où peuvent s'exécuter des agents mobiles existe un serveur d'agents. Ce dernier est un agent stationnaire écrit dans un langage d'écriture d'agents donné. Il assure les services de communication et de transfert aux différents agents et fournit aux utilisateurs (à travers les clients qu'ils utilisent) la possibilité de lancer leurs agents, de les contrôler et de répondre aux requêtes que les différents agents peuvent formuler.

Concevoir le serveur d'agents comme un serveur Web consiste alors:
- A réaliser le transfert des agents avec HTTP,
- A réaliser les différentes interactions entre le serveur d'agents et son monde extérieur (agents et utilisateurs) avec HTTP, et
- A présenter les informations aux utilisateurs à l'aide de techniques basées sur HTTP.

Pour interagir avec un serveur d'agents conçu comme un serveur Web, le client dont se sert l'utilisateur doit supporter la technologie WWW.

2.1. TRANSFERER LES AGENTS MOBILES AVEC HTTP

2.1.1. ENCAPSULATION D'UN AGENT MOBILE DANS UN FORMAT SIMILAIRE A MIME

Dans HTTP, les données (c'est-à-dire, le corps des requêtes ou des réponses) sont transférées dans un format basé sur MIME (Multipurpose Internet Mail Extensions). [LDD95]

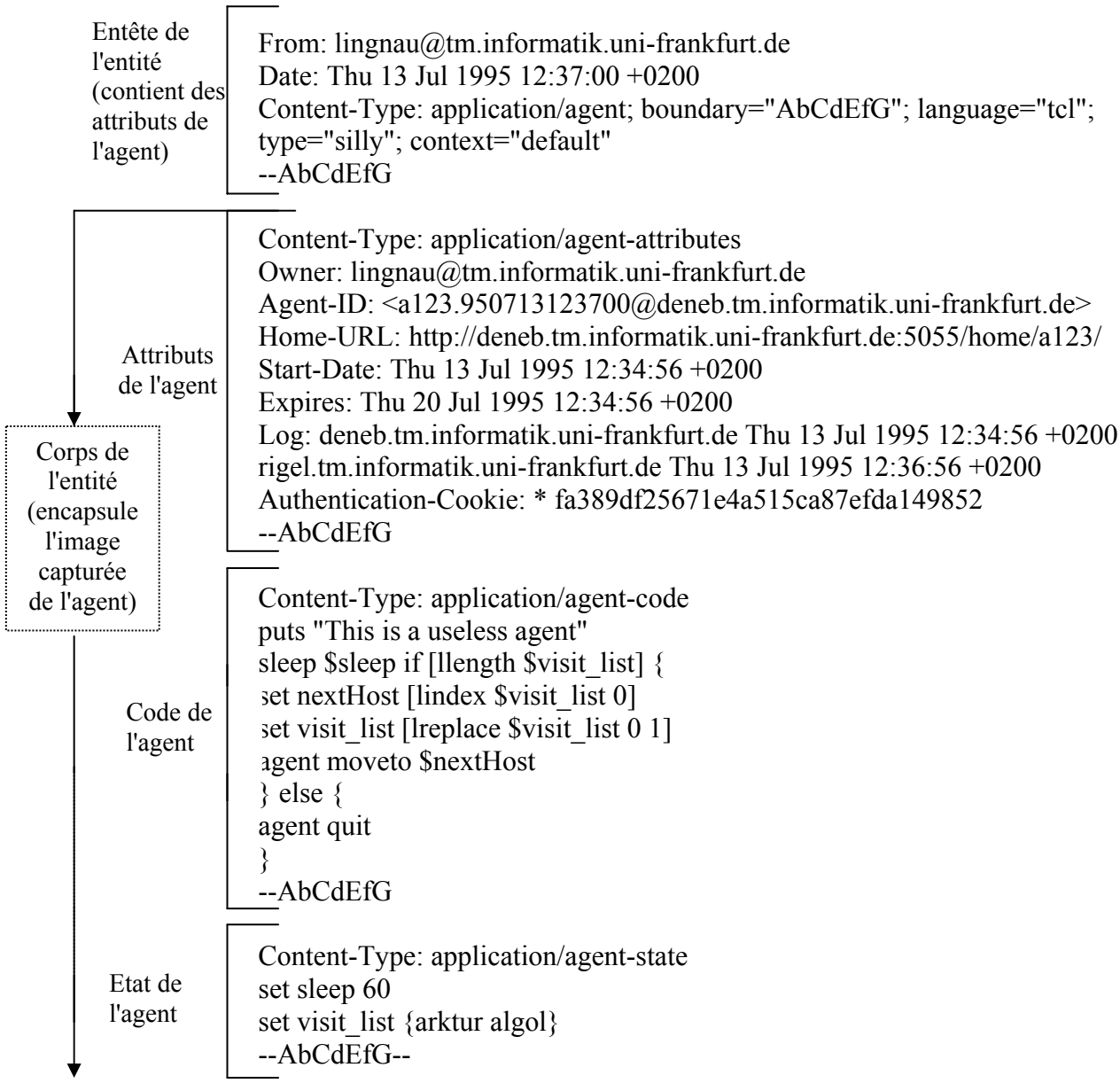
Pour qu'un agent migrant vers (ou lancé pour la première fois pour être déclenché par) un serveur d'agents puisse être reconnu par ce dernier (qui, rappelons le, est un serveur Web), FFMAIN a définit un type d'entité similaire à MIME appelé: Application/Agent.
Une entité de ce type sert à encapsuler un agent et est constituée de deux parties: une partie entête et une partie corps. [LDD95]
- La partie entête est destinée à contenir des attributs qui décrivent le langage d'écriture de l'agent, le genre (ou le type) de l'agent (par exemple, agent de recherche bibliographique), etc.. Elle sera utilisée par le serveur d'agents pour sélectionner l'environnement d'exécution approprié à l'agent ou pour rejeter l'agent si les attributs exprimés ne peuvent pas être satisfaits.
  Cette partie suit le format utilisé dans les entêtes des entités de type MIME.

- La partie corps, quant à elle, est destinée à contenir l'image courante de l'agent mobile à transférer. Rappelons qu'un agent mobile est constitué de trois composantes: un code, un état et des attributs.
  Le serveur d'agents doit consulter la composante "attributs" de l'agent pour pouvoir mettre à jour l'histoire de ses déplacements, connaître son propriétaire, etc.. La composante "attributs" doit donc être mise dans un format qui soit compréhensible pour le serveur d'agents; pour cela, elle suit le format de la partie précédente (c'est-à-dire, la partie entête).
  En revanche, le format des composantes "code" et "état" de l'agent n'est pas spécifié; le contenu de ses composantes est à définir d'une manière qui soit appropriée au langage d'écriture de l'agent.

Une fois encapsulé dans une entité de type Application/Agent, l'agent mobile peut alors être lancé vers un serveur d'agents ou transféré entre les serveurs d'agents en utilisant HTTP[1]. [LDD95]

[1] Il est également possible d'utiliser l'E-Mail pour transférer les agents encapsulés dans ce type d'entité. [LDD95]

Exemple:
Cet exemple présente un agent mobile encapsulé dans une entité de type Application/Agent[`]. [LDD95]



2.1.2. TRANSFERT D'UN AGENT MOBILE

Quand un serveur d'agents reçoit une entité de type Application/Agent, il doit distinguer deux cas: [LDD95]

- Cas d'un agent nouvellement lancé: L'entité reçue contient un agent nouvellement lancé qui demande à s'exécuter pour la première fois. Le serveur d'agents devient, dans ce cas, le serveur natal (HOME SERVER) de cet agent et s'en charge de maintenir la trace de son évolution dans l'infrastructure.
- Cas d'un agent migrant: L'entité reçue contient un agent qui a migré depuis un autre serveur d'agents, et qui demande à continuer son exécution localement. Le serveur d'agents doit, dans ce cas, communiquer la nouvelle localisation de l'agent à son serveur natal.

[`] Il y a à noter que la capture de l'image courante (ou l'état d'exécution courant) d'un agent mobile n'a pas été présentée dans les articles décrivant Agent-Tcl.

Pour distinguer entre les entités encapsulant les agents mobiles, le serveur d'agents maintient deux URLs[\]: Server/Create et Server/Move, tel que, Server est une abréviation de http://host:Port. [LDD95]

- Dans l'URL Server/Create, le serveur d'agents reçoit les entités encapsulant les agents nouvellement lancés; et
- Dans l'URL Server/Move, il reçoit les entités encapsulant les agents qui ont migré depuis un autre serveur d'agents.

Le lancement d'un nouvel agent s'effectue en déposant (avec la commande POST de HTTP) l'entité qui l'encapsule dans l'URL Server/Create maintenue par le serveur d'agents choisi. Tandis que, la migration d'un agent s'effectue en envoyant, également avec la commande POST de HTTP, l'entité qui l'encapsule à l'URL Server/Move maintenue par le serveur d'agents destinataire.

En recevant une entité de type Application/Agent, le serveur d'agents examine ses attributs pour voir s'il peut accepter l'agent encapsulé ou non. Si l'agent n'est pas accepté, le serveur d'agents peut décider de l'écarter, d'envoyer une description d'erreur à son propriétaire ou de lui proposer (s'il en est capable) d'autres sites qui peuvent le satisfaire. Par contre si l'agent est accepté, le serveur d'agents lui déclenche l'environnement d'exécution approprié et lui assigne une URL qui dépend du fait que l'agent est nouvellement lancé ou non. En effet: [LDD95]

- S'il s'agit d'un nouvel agent, le serveur d'agents lui assigne une URL permanente, appelée URL natale (ou Home URL), puis communique cette URL au client utilisé par le propriétaire de l'agent.
  L'URL natale est de la forme: Server/Home/Id, tel que Id identifie l'agent de manière unique chez son serveur d'agents natal.
- Par contre s'il s'agit d'un agent migrant, le serveur d'agents lui assigne une URL temporaire (VISITOR URL) de la forme: Server/Visit/Id, tel que Id identifie l'agent de manière unique chez le serveur d'agents. Le serveur d'agents communique ensuite la nouvelle localisation de l'agent à son serveur natal (Ceci, en envoyant l'URL: Server/Visit/Id au serveur natal de l'agent qui l'inclura dans le document pointé par l'URL natal de l'agent).

Les figures suivantes résument le processus de lancement d'un nouvel agent (Figure IV-2), et le processus de migration d'un agent (Figure IV-3) dans l'infrastructure SINDBAD.

[\] URL: Uniform Ressource Locator.

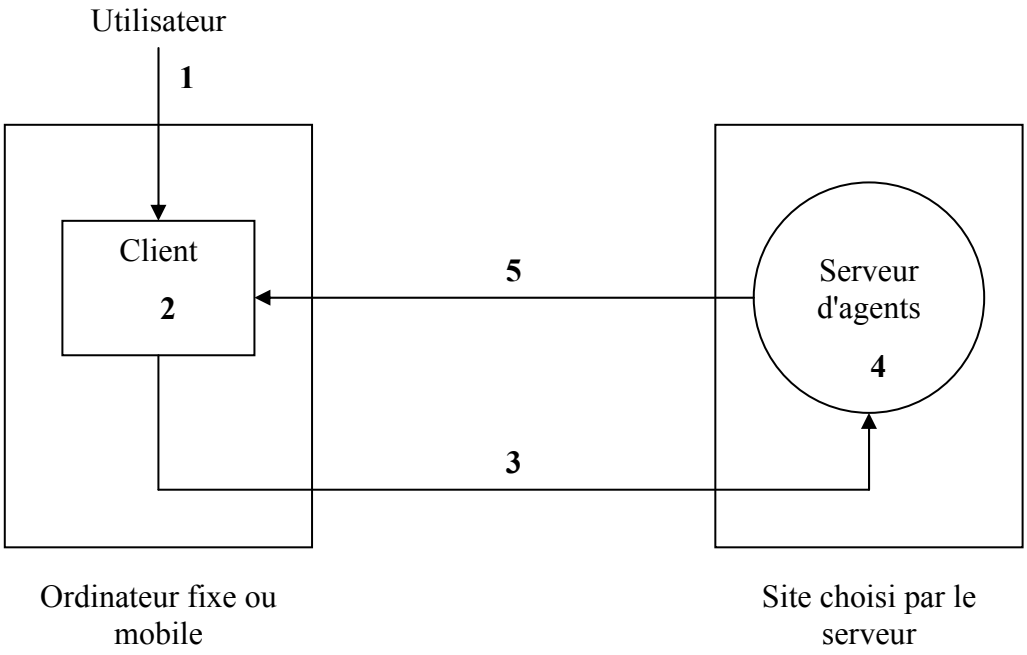


Figure IV-2: Scénario de lancement d'un agent mobile

Légende:

١. L'utilisateur fournit le fichier spécifiant le code de l'agent à lancer (ou dan une autre approche, définit la mission de son agent) au client qu'il utilise.
٢. Le client encapsule l'agent dans une entité de type (Application/Agent);
٣. Puis, dépose l'entité avec la commande POST de HTTP dans l'URL (Server/Create) maintenue par le serveur d'agents existant sur le site que le client choisira (ce site peut donc être le site local, comme il peut être un site distant).
٤. Le serveur d'agents sur le site choisi deviendra le serveur natal de l'agent et lui assignera une URL permanente, appelée URL natale, qui est de la forme: Server/Home/Id.
٥. Le serveur natal communique, ensuite, l'URL natale au client pour que l'utilisateur puisse contrôler, à tout moment, l'exécution de son agent.

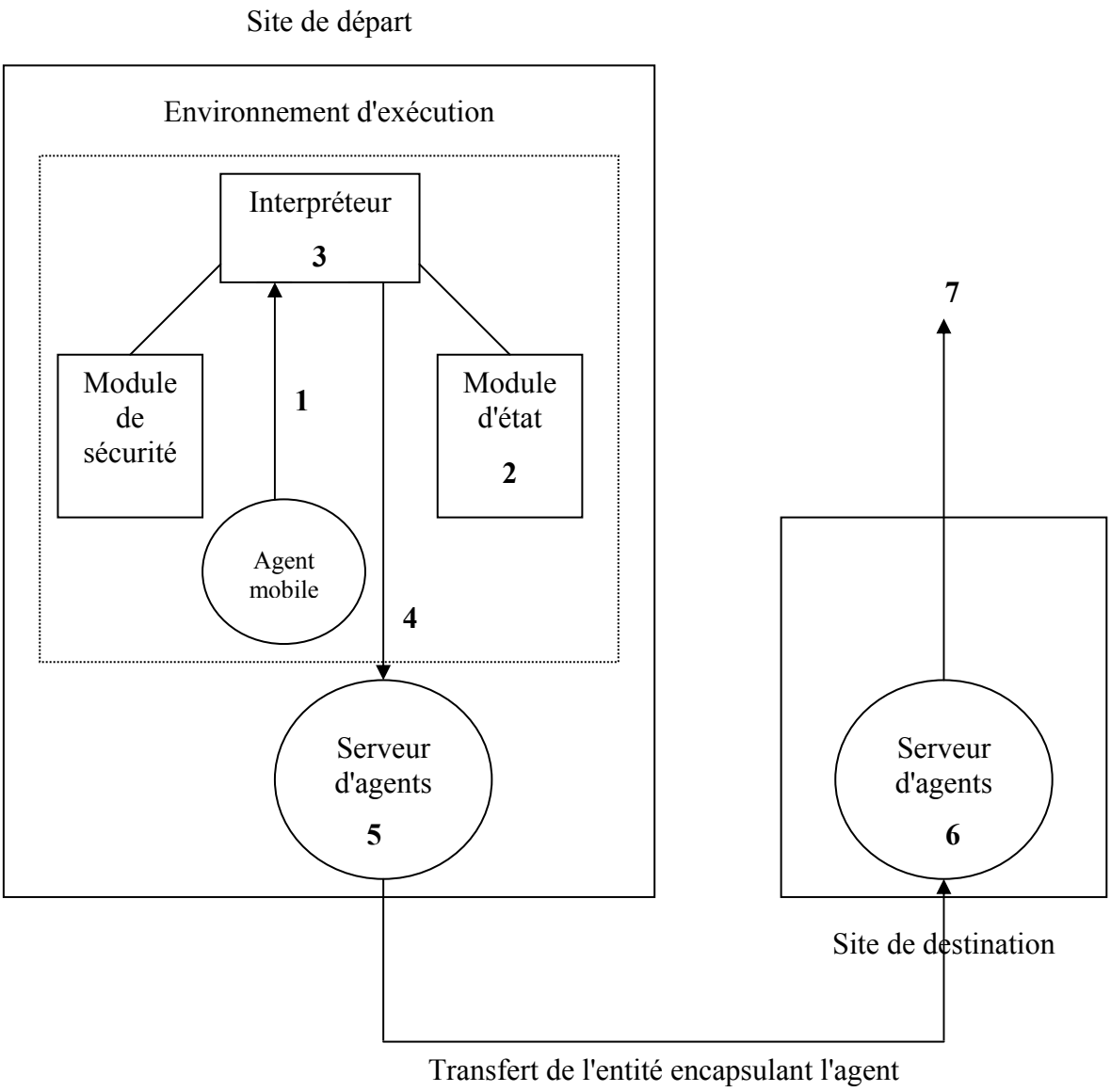


Figure IV-3: Scénario de migration d'un agent mobile.

Légende:

١. L'agent arrive, au cours de son exécution, à une instruction de migration.
٢. Le module d'état capture alors l'image courante de l'agent.
٣. L'interpréteur encapsule, ensuite, l'image capturée de l'agent dans une entité de type Application/Agent;
٤. Puis, demande au serveur d'agents local de déposer cette entité (avec la commande POST de HTTP) dans l'URL (Server/Move) maintenue par le serveur d'agents sur le site de destination.
٥. Si le serveur d'agents local n'est pas le serveur natal de l'agent, alors il lui a certainement attribué auparavant (c'est-à-dire, à son arrivé) une URL temporaire (VISITOR URL); cette URL doit être maintenant supprimée.
٦. Le serveur d'agents sur le site de destination assigne à l'agent arrivé une URL temporaire (VISITOR URL), de la forme: Server/Visit/Id, puis
٧. Communique la nouvelle localisation de l'agent (c'est-à-dire, son URL temporaire "VISITOR URL") à son serveur natal. Ce dernier inclura l'URL temporaire dans le document pointé par l'URL natale de l'agent. En fait, c'est ainsi que le serveur natal d'un agent parvient à maintenir la trace de l'évolution de l'agent dans l'infrastructure.

2.2. INTERGIR AVEC LE SERVEUR D'AGENTS A TRAVERS HTTP

Dans l'infrastructure SINDBAD, le serveur d'agents est au cœur de toutes les interactions effectuées. La figure suivante schématise ceci:

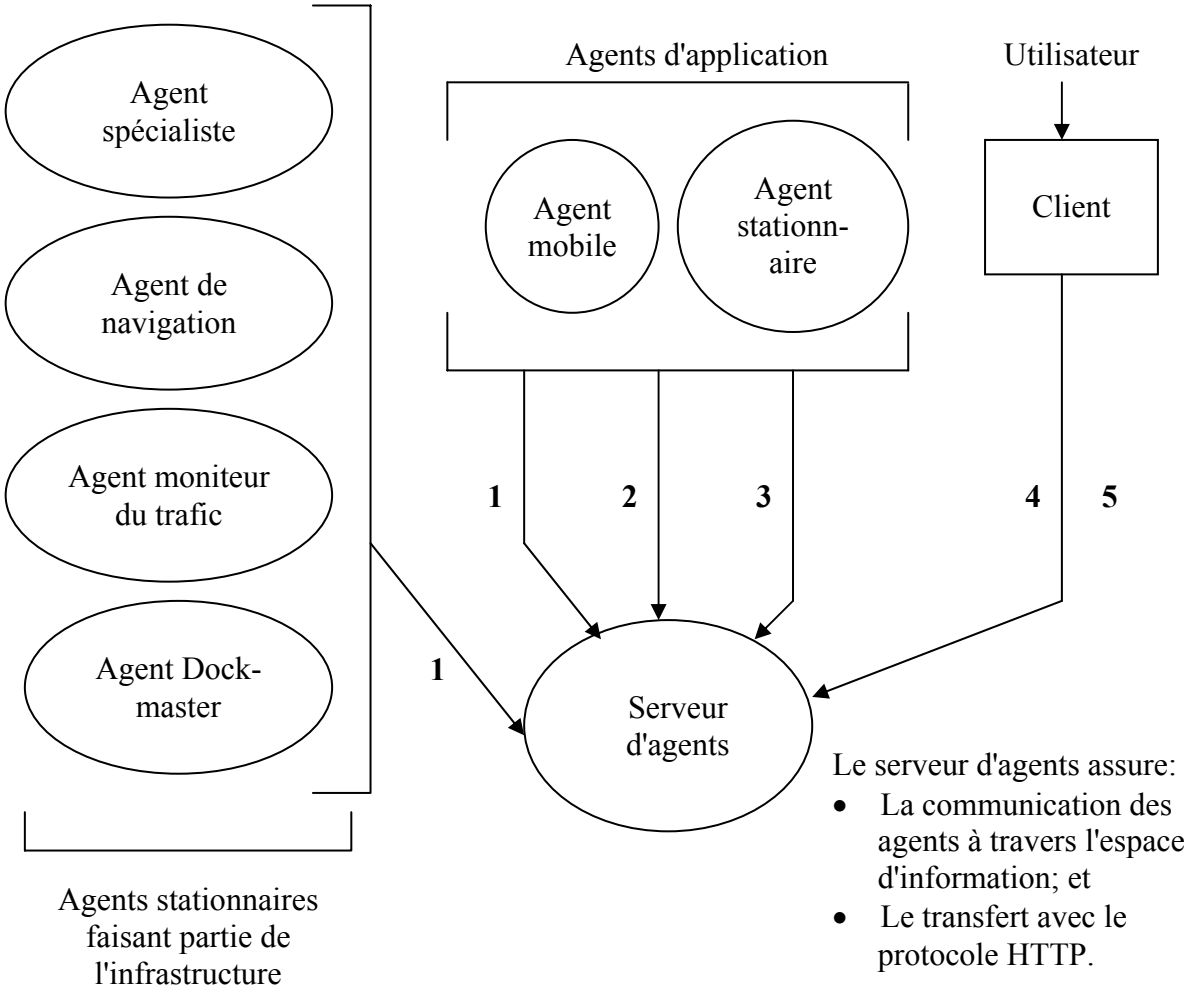


Figure IV-4: Les différentes interactions avec le serveur d'agents dans l'infrastructure SINDBAD.

Légende:

١. Interactions avec le serveur d'agents dans le but de communiquer avec un autre agent (mobile ou stationnaire) ou pour publier des informations. Il s'agit, alors, des opérations de manipulation de (ou d'accès à) l'espace d'information; c'est-à-dire: WRITE, READ et DREAD.
٢. Interactions avec le serveur d'agents dans le but de communiquer avec un utilisateur. Il s'agit aussi des opérations de manipulation de l'espace d'information.
٣. Interactions avec le serveur d'agents pour solliciter un transfert.
٤. Interactions avec le serveur d'agents pour lancer un agent ou pour contrôler son exécution.
٥. Interactions avec le serveur d'agents pour répondre aux requêtes formulées par des agents. Il s'agit, là aussi, des opérations de manipulation de l'espace d'information.

D'après cette légende, on déduit facilement que les interactions avec le serveur d'agents s'effectuent pour trois raisons:

- Afin qu'un utilisateur puisse lancer un agent et contrôler son exécution,

- Afin que l'agent ou l'utilisateur puissent manipuler (ou accéder à) l'espace d'information, ou
- Afin qu'un agent sollicite un transfert vers un autre site.

Dans ce qui suit, nous examinons comment contrôler l'exécution d'un agent et accéder à l'espace d'information avec HTTP. Les autres points (c'est-à-dire, le lancement et le transfert d'un agent) ont été, en effet, déjà présenté.

2.2.1. CONTROLER L'EXECUTION D'UN AGENT[1]

Après avoir lancé un agent dans l'infrastructure SINDBAD, son propriétaire peut le contrôler (Consulter son état d'exécution, arrêter son exécution, reprendre son exécution, etc.) à travers le client qu'il utilise.
En effet, pour connaître, par exemple, l'état d'exécution de son agent, le propriétaire consulte le contenu de son URL natale. Par contre, si l'agent a migré vers un autre serveur d'agents (c'est-à-dire, s'il a quitté son serveur d'agents natal), son propriétaire doit consulter le contenu de l'URL temporaire (VISITOR URL) assignée à l'agent par le serveur d'agents destinataire.
Le propriétaire de l'agent parvient à connaître l'URL (VISITOR URL) actuellement assignée à l'agent, en consultant le contenu de l'URL natale de l'agent. Cette URL, rappelons-le, doit certainement contenir l'URL (VISITOR URL). Ainsi, partout où l'agent peut aller, son propriétaire peut connaître son état d'exécution en consultant son URL natale ou en consultant (si l'agent a quitté son serveur natal) son URL (VISITOR URL).
Le résultat de la consultation est un document de type HTML fournissant non seulement des informations sur l'état d'exécution de l'agent, mais aussi des liens vers des URLs spéciales (permettant de contrôler l'exécution de l'agent), tels que[2]: [LDD95]

- VisitorURL/Attributs
- VisitorURL/State

qui permettent de consulter une partie des composantes "état" et "attributs" de l'agent.

- VisitorURL/Stop
- VisitorURL/Recall

qui permettent d'arrêter ou de reprendre l'exécution de l'agent.

2.2.2. ACCES A L'ESPACE D'INFORMATION

Un espace d'information supporte la communication entre les agents existant sur son site; et ce, à travers trois opérations de base: WRITE(C,A,V), READ(C) et DREAD(C).
Ces opérations peuvent être réalisées avec HTTP. En effet: [LDD95]

- L'opération (WRITE(C,A,V)) peut être réalisée en encapsulant l'enregistrement à ajouter dans l'espace d'information dans une entité (de type similaire à MIME) qui sera déposée avec la commande POST de HTTP à l'URL: Server/Info?clé, tel que:
  - Server est le serveur d'agents (qui maintient l'espace d'information).
  - Clé est destinée à contenir le champ (C) de l'enregistrement.
  - Le champ (A) de l'enregistrement constituera un champ (Access:) dans l'entête de l'entité. Et,
  - Le champ (V) de l'enregistrement constituera le corps de l'entité.
- L'opération (READ(C)) peut être réalisée avec la commande GET de HTTP.

[1] Le contrôle d'agents n'a pas été clairement discuté dans Agent-Tcl. [PK98]

[2] Les liens cités sont le résultat de la consultation de l'URL (VISITOR URL), en ayant supposé que l'agent n'est pas actuellement chez son serveur d'agents natal. Par contre, si on suppose que l'agent se trouve chez son serveur d'agents natal, son propriétaire se contentera alors de consulter uniquement son URL natale et le résultat de la consultation sera un document de type HTML contenant, plutôt, les liens suivants: HomeURL/Attributs, HomeURL/State, HomeURL/Stop et HomeURL/Recall.

- Et enfin, l'opération (DREAD(C)) peut être réalisée avec la commande DGET qu'on peut rajouter[\] à HTTP.

La figure VI-5 présente un type d'entité servant à encapsuler un enregistrement à déposer ou à retirer de l'espace d'information; alors que la figure VI-6 illustre comment accéder à l'espace d'information via HTTP.

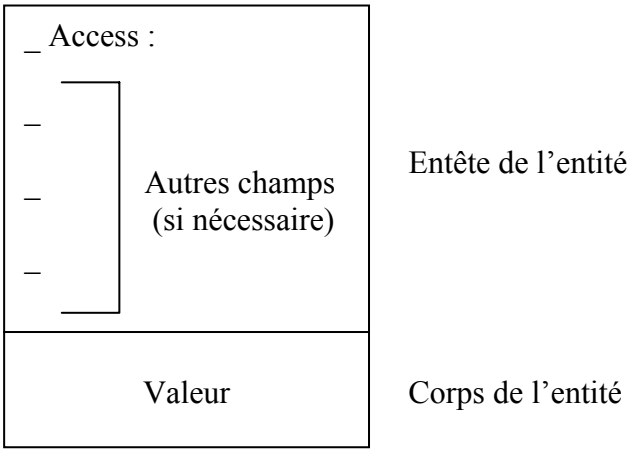


Figure IV-5: Type d'entité servant à encapsuler un enregistrement à déposer ou à retirer de l'espace d'information.

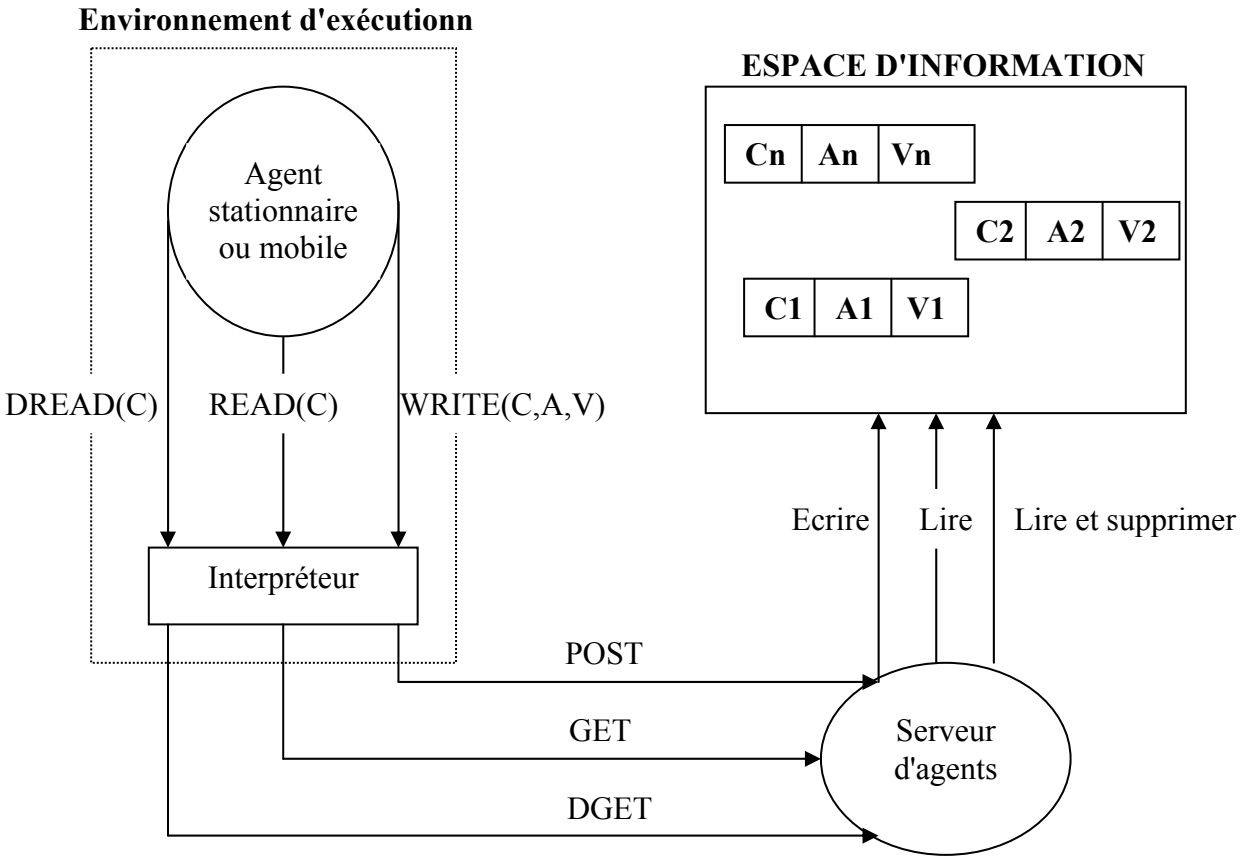


Figure IV-6: Accès à l'espace d'information à travers HTTP dans un site de l'infrastructure SINDBAD.

[\] Faire des extensions à HTTP est explicitement autorisé par la définition du protocole. [DLD97]

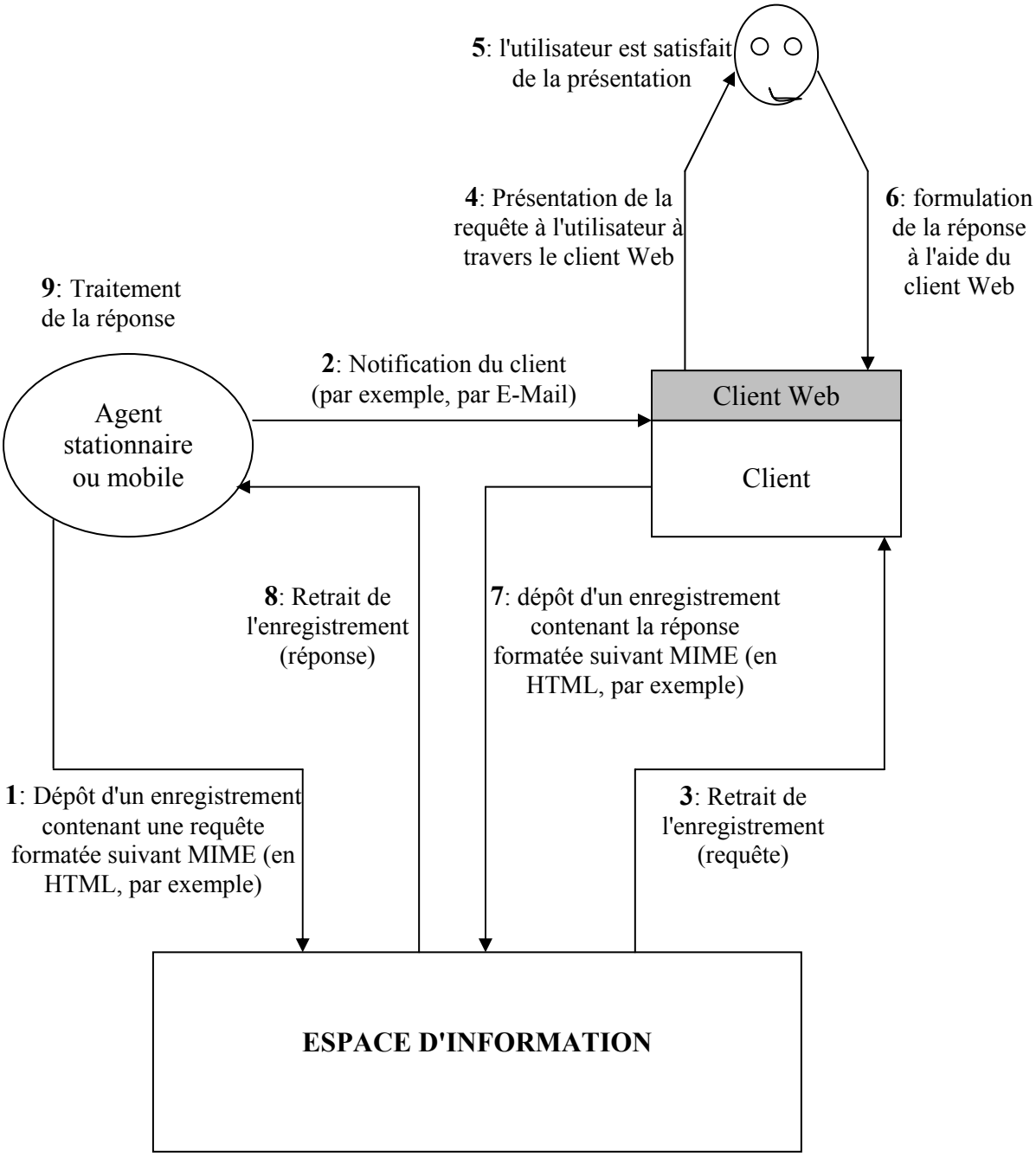
2.3. PRESENTER LES INFORMATIONS AUX UTILISATEURS AVEC DES TECHNIQUES BASEES SUR HTTP

La plupart des utilisateurs sont familiers avec la technologie WWW; ils apprécieront alors l'idée de leur présenter les informations d'un système d'agents mobiles avec des techniques basées sur HTTP et, par conséquent, accepteront plus facilement l'approche des agents mobiles.

L'infrastructure SINDBAD permet aux utilisateurs de percevoir les informations générées par l'exécution des agents (les informations de contrôle, les résultats, les erreurs, ainsi que, les requêtes formulées par les agents) à travers des techniques basées sur HTTP. En effet: [LDD95]

- Avec un client basé sur HTTP (en d'autres termes, un client Web), le propriétaire d'un agent peut définir la mission de son agent et contrôler son exécution à travers des documents de type HTML.
- D'autre part, un agent qui a une requête destinée à un utilisateur[١] a la possibilité de la mettre dans un format qui soit adéquat au client Web. En effet, rappelons que les champs (V) des enregistrements de l'espace d'information sont laissés non interprètés[٢]. Ainsi, avant de déposer sa requête dans l'espace d'information l'agent peut la mettre dans un format de type MIME. De cette façon, l'utilisateur peut consulter la requête d'une manière plus familière à travers une technique du WWW (HTML, Graphiques, Applets de Java, etc.).
- Finalement, les résultats[٣] de l'exécution d'un agent peuvent être reportés à son propriétaire de la même manière que les requêtes; c'est-à-dire, à travers l'espace d'information après les avoir formatés suivant le type MIME.
  Une autre alternative peut consister à envoyer les résultats par E-Mail.

[١] L'utilisateur n'est pas forcement le propriétaire de l'agent; il peut être une autre personne.
[٢] Revoir la section 5.1 du chapitre III - Partie II.
[٣] Les erreurs d'exécution sont considérées aussi comme des résultats.



NB: - L'utilisateur peut être une personne autre que le propriétaire de l'agent.
- Le propriétaire et l'agent peuvent être sur des sites distincts.
- Le retrait s'effectue, par exemple, avec l'opération DREAD.

Figure IV-7: Scénario d'une requête formulée par un agent pour un utilisateur.

3. AVANTAGES D'UN SERVEUR D'AGENTS CONCU COMME UN SERVEUR WEB

SINDBAD, comme toute autre infrastructure d'agents mobiles, a besoin de protocoles pour la communication et le transfert des agents. L'usage de HTTP qui est un standard largement utilisé constitue un choix très intéressant. En effet: [LDD95, DLD97]

- HTTP est un protocole très connu, bien compris et largement accepté. Par conséquent, populariser une infrastructure d'agents sera largement facile si elle est basée sur HTTP que sur un autre protocole personnel ou moins connu.

- HTTP contient toutes les primitives nécessaires pour supporter la mobilité des agents (citons, par exemple, la méthode POST qui peut être utilisée pour envoyer un agent vers un autre serveur d'agents); en plus, la spécification de HTTP permet d'ajouter des extensions si celles-ci se révèlent nécessaires.
- L'indépendance de la plate-forme Web facilite le support des agents mobiles dans un réseau hétérogène.
- Les navigateurs Web existant (par exemple, MOSAIC) peuvent être utilisés pour réaliser une grande partie du client qu'utilise le propriétaire de l'agent. Ceci réduit l'effort restant à faire et produit une interface familière aux utilisateurs. Ces derniers ne seront pas, de cette façon, obligés d'apprendre comment manipuler une autre interface pour pouvoir utiliser les systèmes d'agents mobiles.
- La communauté Web effectue des recherches intenses sur des sujets tels que la sécurité de la transmission à travers HTTP, ainsi que le payement digital nécessaire au commerce électronique[1].
  Il est alors plus judicieux d'utiliser les résultats des recherches effectués par la communauté Web que d'essayer de convaincre chaque personne que d'autres approches peuvent également bien fonctionner.
  Bénéficier de ces résultats à partir d'une infrastructure d'agents adoptant des protocoles différents est, cependant, difficile; mais en revanche, ces résultats seront directement utilisables avec une infrastructure basée sur HTTP.
- L'intégration entre les services basés sur agents et les services basés sur WWW peut se faire d'une façon meilleure, et les systèmes d'agents deviendront, par conséquent, des services que les fournisseurs de services Web peuvent offrir directement.
- Enfin, l'usage de HTTP permet à des infrastructures d'agents hétérogènes d'interopérer à travers les espaces d'information de l'infrastructure SINDBAD proposée.

4. INTEROPERABILITE ENTRE DES INFRASTRUCTURES D'AGENTS HETEROGENES

Plusieurs infrastructures d'agents existent. Les agents appartenant à une infrastructure donnée peuvent naturellement communiquer entre eux. Cependant, il serait très intéressant si un agent appartenant à une infrastructure donnée peut également communiquer avec un agent appartenant à une autre infrastructure.

Pour réaliser l'interopérabilité entre des infrastructures d'agents hétérogènes, une alternative consiste, par exemple, à examiner comment chaque infrastructure réalise les interactions entre ses propres agents, et proposer ensuite des interfaces de communication entre les différentes infrastructures.
Une autre alternative plus intéressante consiste à utiliser HTTP. Ce dernier constitue, en effet, un choix évident puisqu'il est probablement supporté d'une manière ou d'une autre par la plupart des infrastructures d'agents.

Notons qu'une interaction (ou communication) entre deux agents appartenant à une même infrastructure peut se faire de deux manières: [DLD97]

- Interaction directe: Ce type d'interaction est inspiré de l'approche orientée objet: Un agent exporte un ensemble de méthodes qu'un autre agent peut appeler directement.
  Ce style d'interaction est utilisé, par exemple, dans Telescript et Agent-Tcl[2].

[1] Le commerce électronique est l'une des plus importantes applications envisagées pour les agents mobiles. [LDD95]

[2] Agent-Tcl permet, en plus, le passage de messages entre les agents. Par conséquent, le style d'interaction indirecte est également supporté.

- Interaction indirecte: Dans ce type d'interaction, l'infrastructure d'agents s'interpose entre les agents qui désirent s'interagir. Un agent ne peut donc interagir avec un autre agent directement. Ce type d'interaction peut être réalisé en permettant, par exemple, aux agents de s'échanger des messages à travers l'infrastructure d'agents, ou bien en interposant entre eux un espace d'information à travers lequel ils peuvent s'échanger des informations.
  Ce style d'interaction est utilisé, par exemple, dans FFMAIN (et, par conséquent, dans SINDBAB aussi).

L'usage de HTTP dans l'infrastructure SINDBAD (comme d'ailleurs dans FFMAIN) permet, en fait, de supporter les deux styles d'interaction; et ce, entre des agents qui peuvent appartenir même à des infrastructures hétérogènes. En effet: [DLD97]

- Interaction directe: Quand un agent appartenant à une infrastructure donnée connaît suffisamment un autre agent pouvant appartenir éventuellement à une autre infrastructure, il peut alors interagir directement avec lui à l'aide de HTTP.
  HTTP représente, dans ce cas, un mécanisme simple de RPC, qui réalise une interaction directe entre deux agents à travers une URL. Ainsi, toute URL va correspondre à une invocation à distance d'une méthode appartenant à un agent donné, tel que:
  - Les arguments peuvent être passés dans l'URL, et
  - Les résultats peuvent être obtenus comme des pages Web.
- Interaction indirecte: L'infrastructure SINDBAD utilise des espaces d'information pour supporter la communication entre ses agents écrits dans différents langages. Si, en plus, on autorise l'accès à ces espaces d'information aux agents appartenant à d'autres infrastructures, tout agent peut alors à travers HTTP accéder à ces espaces et, par conséquent, interagir avec différents agents pouvant appartenir à différentes infrastructures.

5. CONCLUSION

Plusieurs systèmes d'agents mobiles (par exemple, Telescript) peuvent être considérés comme fermés (ils supportent un langage d'écriture d'agents donné) et propres à leurs concepteurs (ils n'utilisent pas des standards; mais plutôt, des protocoles propres à leurs concepteurs pour supporter la mobilité et les interactions des agents) [DLD97]. Par contre, l'infrastructure SINDBAD que nous proposons n'est liée à aucun langage d'écriture d'agents et réalise les interactions et le transfert des agents à travers un standard: le protocole HTTP.

Après avoir vu, dans le chapitre précédent, que SINDBAD est une infrastructure assez complète et assez générale, permettant de supporter les agents mobiles dans un environnement mobile; nous avons voulu, à travers ce chapitre, montrer que rendre cette infrastructure complètement basée sur HTTP, lui permet de devenir familière, plus convenable et, surtout, largement acceptée. En plus, ça lui permet d'offrir, à travers ses espaces d'information accessibles à l'aide de HTTP, un mécanisme de communication permettant l'interopérabilité entre des infrastructures d'agents hétérogènes.

CONCLUSION

L'approche d'agents mobiles constitue un paradigme de programmation adéquat, puisant et efficace pour les applications distribuées. Elle est mieux vue comme un outil général permettant de réaliser des applications distribuées quelconques. En plus, c'est un paradigme excellent quand des ordinateurs partiellement connectés (ordinateurs mobiles et/ou ordinateurs connectés par modems) sont utilisés. Aucune application n'exige, cependant, l'usage absolu des agents mobiles: la plupart des applications peuvent, en effet, être réalisées avec des programmes stationnaires et quelques paradigmes de communication convenables, tels que RPC (Remote Procedure Call). Néanmoins, ceci sera au prix du chargement du système et, peut être, à l'incommodité par rapport aux utilisateurs.

Vu l'adéquation de l'approche d'agents mobiles à la structuration des applications distribuées en environnement mobile, notre travail consistait à proposer une infrastructure d'agents mobiles pour un environnement mobile permettant de supporter des applications distribuées quelconques.

Plusieurs systèmes d'agents mobiles existent; cependant, leurs infrastructures généralement diffèrent. Il n'y a pas d'infrastructure commune pour réaliser les agents mobiles; en plus, il est improbable qu'un standard commun pour réaliser les agents mobiles émergera dans le futur proche.
En absence d'une infrastructure pouvant servir comme un standard pour développer les applications d'agents mobiles, nous avons pensé à combiner essentiellement entre deux infrastructures intéressantes (l'infrastructure d'Agent-Tcl et l'infrastructure de FFMAIN) dans l'espérance d'obtenir une infrastructure assez complète, assez générale, familière, plus convenable et largement acceptée. Nous avons appelé cette infrastructure: SINDBAD.
SINDBAD, l'infrastructure que nous proposons pour supporter les agents mobiles en environnement mobile, est, en fait, l'infrastructure d'Agent-Tcl rendue complètement basée sur HTTP et supportant la communication des agents avec l'approche de l'espace d'information accessible à l'aide de HTTP.

L'infrastructure SINDBAD offre à ses agents une grande flexibilité, en leur permettant:

- D'être écrits avec différents langages d'écriture d'agents.
- De communiquer à travers un mécanisme de bas niveau (qui est l'espace d'information), supportant différents styles d'interaction entre les agents (un-à-un, un-à-plusieurs et plusieurs-à-plusieurs) et sur lequel peuvent être organisées des communications plus structurées en utilisant des protocoles de haut niveau, tels que KIF (Knowledge Intechange Format) et KQML (Knowledge and Query Manipulation Language). En plus, les agents peuvent, si nécessaire, utiliser d'autres moyens de communication (par exemple: les connexions directes[\] et les mémoires partagées), qui peuvent alors être rajoutés.
- D'élaborer des plans de navigation adaptatifs, et
- D'interagir avec les différents systèmes pouvant exister sur les différents sites de l'infrastructure.

[\] Une connexion directe est un flot de messages nommé (revoir la section 4-1 du chapitre III - Partie I).

SINDBAD présente aussi l'avantage d'adopter *immédiatement* les résultats des recherches intenses qui s'effectuent continuellement sur WWW, particulièrement, dans le domaine de la sécurité; et d'être facilement intégrable avec les services basés sur WWW. En plus, elle permet, à travers ses espaces d'information accessible à l'aide de HTTP, de supporter l'interopérabilité entre des agents appartenant à des infrastructures hétérogènes.

Plusieurs systèmes d'agents mobiles (par exemple, Telescript) sont considérés comme fermés (ils supportent un langage d'écriture d'agents donné) et propres à leurs concepteurs (ils n'utilisent pas des standards; mais plutôt, des protocoles propres à leurs concepteurs pour supporter la mobilité et les interactions des agents). Par contre, SINDBAD n'est liée à aucun langage d'écriture d'agents et réalise les interactions et le transfert des agents à travers un standard: le protocole HTTP.

Bien que les domaines d'application des agents mobiles et leurs avantages ont été clairement identifiés et bien que différents systèmes d'agents mobiles ont été conçus et implémentés, peu d'applications utilisant les agents mobiles ont été développées. Par conséquent, le paradigme d'agents mobiles souffre encore d'un manque d'applications de démonstration. [Bou99b]
A l'heure actuelle, plusieurs projets (académiques et universitaires) concernant le concept d'agents mobiles sont en cours[١]: Université d'Ottawa, MIT media Lab., Université de Stuttgart, UMBC[٢], GMDFOKUS[٣], IBM, General Magic, Object Space[٤], Mitsubishi, Hitachi, British Telecom, etc.. En plus, plusieurs efforts de normalisation s'effectuent actuellement et se sont déjà traduits par le rajout de MAF (Mobile Agent Facility) à CORBA[٥]. [PK98, Bou99a]

Comme perspective à notre travail, nous proposons d'implémenter l'infrastructure SINDBAD en commençant par supporter de préférable un seul langage d'écriture d'agents (par exemple, Java ou Tcl[٦]). Le but est, en effet, d'acquérir (en premier temps) une expérience suffisante avec les agents mobiles permettant de valider l'infrastructure SINDBAD et d'établir un parallèle entre elle et les différentes infrastructures actuelles et futures. Il convient, ensuite, d'augmenter l'infrastructure SINDBAD avec d'autres langages d'écriture d'agents; puis, la rendre tolérante aux fautes et sûre pour qu'elle puisse être réellement exploitée.

---

[١] Pour avoir une idée, l'annexe peut être consultée.
[٢] UMBC: Université de Maryland Baltimore Country, assez active dans le domaine de l'agent. [Bou99a]
[٣] GMDFOKUS: Une boite allemande très active en matière d'agents mobiles. [Bou99a]
[٤] ObjectSpace: Une boite informatique qui a développer la plate-forme Voyager. [Bou99a]
[٥] CORBA (Common Object Request Broker Architecture) est une architecture qui permet, à l'aide d'un mécanisme basé sur RPC, l'invocation à distance d'objets dans un environnement distribué. L'interaction entre objets écrits dans différents langages est supportée, mais non la mobilité de leurs codes. [DLD97]
[٦] Rappelons que Java et Tcl sont devenus les langages évidents pour l'implémentation des systèmes d'agents mobiles.

BIBLIOGRAPHIE

[Ach95] Acharya A., "Structuring Distributed Algorithms and Services for Networks with Mobile Hosts", A dissertation submitted to the Graduate School - New Brunswick and to Rugters (the State University of New Jersy), in partial fulfillment of requirements for the degree of Doctor of Philosophy (Graduate Program in Computer Science), May 1995.

[AB94] Acharya A., Badrinath B. R., "Chekpointing distributed applications on mobile computers", 3rd IEEE International Conference on Parallel and Distributed Information Systems, October 1994.

[Bad95] Badache N., "La mobilité dans les systèmes répartis", Publication Interne N° 962, IRISA, Octobre 1995.

[Bag96] Baggio Aline, "Environnements mobiles: Caractéristiques et Problèmes", Dans le seminaire sur les systèmes et applications réparties, CNET-Issy les moulineaux, Fevrier 1996.

[Bar97] Barbara D., "Certification Reports: Supporting Transactions in Wireless Systems", In Proceeding of the IEEE, 17th International Conference on Distributed Systems, Baltimore, Maryland, U.S., May 1997.

[Bou99a] Boukhatem N., "Introduction au concept d'agents mobiles", support de cours, ENST- Ecole Nationale Supérieure des Télécommunications, Département d'informatique et Réseaux, Paris, 1998-1999.

[Bou99b] Boukhatem N., "Adistributed Query Service Based on Mobile Agent concept", First International Workshop on Mobile Agent Telecommunication Applications : MATA'99, Ottawa, Canada, Octobre 1999.

[BAI93] Badrinath B. R., Acharya A., Imielinski T., "Impact of Mobility on Distributed Computations", In ACM Operating System Review, 27(2), April 1993.

[BAI94a] Badrinath B. R., Acharya A., Imielinski T., "Designing distributed algorithms for mobile computing networks", Technical report, Department of Computer Science, Rugter University, 1994, U.S.

[BAI94b] Badrinath B. R., Acharya A., Imielinski T., "Structuring distributed algorithms for mobile hosts", Proceeding of the 14th International Conference on Distributed Computing Systems, Poznan, Poland, pages: 21-28, June 1994.

[BBIM93] Badrinath B. R., Bakre A., Imielinski T., Marantz R., "Handling Mobile Clinets: A case for Indirect Interaction", Technical Report, Department of Computer Science, Rugter University, 1993.

[BDPLT96] Borgia R., Degano P., Priami C., Leth L., Thomsen B., "Understanding mobile agents via a non-interleaving semantics for Facile", Technical Report, Departement d'informatique, Université de Pisa corso d'Italie et European Computer-Industry Research Center Arabellastr., Munich, Germany, 1996.

[BI92] Badinath B. R., Imielinski T., "Replication and Mobility", Proceeding of the 2nd IEEE Workshop on Management of replicated Data, pages: 9-12, November 1992.

[BN84] Birrell A. D., Nelson B. J., "Implementing remote procedure calls", ACM Transactions on Computer Systems 2(1), pages: 39-59, February 1984.

[BP95] Pitoura E., Bhargava B., "A framework for providing consistent and recovrable agent-based access to heterogeneous mobile databases", Sigmod Record 24(3), Pages: 44-49, September 1995.

[Chr93] Chrysanthis P. K., "Transaction Processing in Mobile Computing Environments", In Proceeding of the IEEE Workshop on Advances in Parallel and Distributed Systems, October 1993.

[CGHLP95] Chess D., Grosof B., Harrison C., Levine D., Parris C., "Itinerant Agent for Mobile Computing", IBM Research Division, T. J. Watson Research Center, March 1995.

[Duc 92] Duchamp D., "Issues in wireless mobile computing", In Proceeding of the Third Workshop on Workstation Operating Systems (Key Biscaryne, FL), IEEE, April 1992.

[DLD97] Dömel P., Lingnau A., Drobnik O., "Mobile agent interaction in heterogeneous environments", First International Workshop on Mobile Agents 97 (MA'97), Berlin, Germany, April 7-8, 1997.

[DM92] Douglis F., Marsh B., "The Workstation as a Waystation: Integrating Mobility into Computing Environments", Natsushita Information Technology Laboratory, Princelon, NJ, U.S., April 1992.

[DPB94] Davies N., Pink S., Blair G. S., "Services to support Distributed Applications in a Mobile Environment", Distributed Multimedia Research Group at Lancaster University (U.K.) and Swedish Institute of Computer Science (Sweden), 1994.

[Fal87] Falcone J. R., "A programmable interface language for heterogenous systems", ACM Transactions on Computer Systems 5(4), pages 330-351, November 1987.

[Fiu94] Fiuczynski M. E., "A programming methodology for disconnected operation", Technical Report, Department of Computer Science and Engineering, University of Washington, March 1994.

[FZ94] Forman G. H., Zahorjan J., "The challenges of mobile computing", IEEE Computer 27(4), pages: 38-47, April 1994.

[Ghe94] Gherbi T., "Tolérance aux pannes et SGBDRs répartis", Thèse d'Ingéniorat, I.N.I., Algérie, Décembre 1994.

[Gra95a] Gray R. S., " PH.D. Thesis Proposal: Transpotable Agents", Department of Computer Science, Dartmouth college, Hanover, May 1995.

[Gra95b] Gray R. S., "Agent Tcl: A transportable agent system", Department of Computer Science, Dartmouth college, Hanover, November 1995.

[Gra96] Gray R. S., "Agent Tcl: Alpha release 1.1", Department of Computer Science, Dartmouth College, Hanover, December 1995.

[GKNRC96] Gray R., Kortz D., Nog S., Rus D., Cybenko G., "Mobile agents for mobile computing", Technical Report, Department of Computer Science, Dartmouth College, May 1996.

[Har95] Harker K. E., "TIAS: A Transportable Intelligent Agent System", Department of Computer Science, Dartmouth college, Hanover, June 1995.

[Hil95] Hild S. G., "Disconnected operation for wireless nodes - Position Statement", Technical Report, IBM U.K.. Laboratories, Winchester, Hampshire, England, June 1995.

[HCK95] Harrison C. G., Chess D. M., Kershenbaum A., "Mobile agents: Are they a good idea?", Technical Report, IBM T. J. Watson Research Center, March 1995.

[IB92] Imielinski T., Badrinath B. R., "Querying in highly mobile distributed environments", Proceeding of the 18th VLDB, pages: 41-52, August 1992.

[IB93] Imielinski T., Badrinath B. R., "Data management for mobile computing", Sigmod Record, 22(1), pages: 34-39, March 1993.

[IB94] Imielinski T., Badrinath B. R., "Mobile wireless computing: solutions and challenges in data management", CACM vol. 37(10), pages: 18-28, October 1994.

[JBE94] Jing J., Burkhres O., Elmagarmid A., "Distributed lock Managment for Mobile Transactions", Technical Report, Depatment of Computer Science, Purdue University, 1994.

[Kia96] Kiared A. S., "Transactions dans les réseaux avec sites mobiles", Rapport de mini-projet (1ère année P.G.), USTHB, Algérie, 1996.

[Kna95] Knabe F. C., "Language Support for Mobile Agents", School of Computer Science, Carnegie Mellon University, Pittsburgh, December 1995.

[KKT93] Koch H., Krombholz L., Theel O., "A brief introduction into the world of mobile computing", Technical Report, Department of Computer Science, University of Darmstadt, May 1993.

[KPT94] Kaashoch M. F., Pinckney T., Tanber J. A., "Dynamic Documents: Mobile Wireless access to the WWW", In the Proceeding.of the IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994.

[KS92] Kistler J., Santayanaranyanan M., "Disconnected operations in the Coda file systems", ACM Transactions on Computer Systems, 10(1), February 1992.

[Lea99] Leach P., "HTTP Extension Framework", Microsoft Scott Lawrence, Agranat Systems, January 1999.

[LD95] Lingnau A., Drobnik O., "An Infrastructure for Mobile Agents: Requierements and Architecture", Proc. 13th DIS Workshop, Orlando, Florida, September 1995.

[LD96] Lingnau A., Drobnik O., "making mobile agents communicate: A flexible approach", In the 1st annual Conference on Emerging Technologies and Applications in Communication (etaCOM'96), Portland, Oregon, May 1996.

[LD98a] Lingnau A., Drobnik O., "Mobile Agents in a Mobile Communications System Database", 3rd IFIP TC6 Workshop on Personal Wireless Communications (Wireless Local Access), Tokyo, Japan, April 7-9, 1998.

[LD98b] Lingnau L., Drobnik O., "Agent-User Communications: Requests, Results, Interactions", Second International Workshop on Mobile Agents 98 (MA'98), Stuttgart, Germany, September 1998.

[LD98c] Lingnau L., Drobnik O., "Talking to Mobile Agents: Requests, Results, Interactions", in Rothermel, Hohl (eds.), _Mobile_Agents_, Lecture Notes in Computer Science 1477, Berlin, September 1998.

[LDD95] Lingnau A., Drobnik O., Dömel P., "An HTTP-based architectute for mobile agents", In the 4th International World Wide Web Conference, Boston, December 1995.

[LM98] Lardjane N., Medjek N., "Développement d'un environnement client Hypertext-Z39.50 intégré", Thèse d'Ingéniorat, Institut d'Informatique, U.S.T.H.B., Algérie, Septembre 1998.

[LS95] Lu Q., Satayanaranyanan M., "Improving Data Consistency in Mobile Computing using Isolation-Only Transactions", Technical Report, CMU-CS-95-126, School of Computer Science, Carnegie Mellon University, March 1995.

[MES95] Mummert L. B., Ebling M. R., Satyanarayanan M., "Exploiting Weack connectivity for Mobile File Access", School of Computer Science, Carnegie Mellon University, U.S., 1995.

[MKM98] Minar N., Kramer K. H., Maes P., "Cooperating Mobile Agents for Mapping Networks", MIT Media Lab., E15-305 20 Ames ST, Cambridge MA 02139, U.S.A., 1998.

[Nar94] Narasayya V. R., "Distributed Transactions in Mobile Computing Systems", Technical Report, Department of Computer Science, University of Washington, March 1994.

[PB93] Pitoura E., Bhargava B., "Dealing with mobility: Issues and Research Challenges", Technical Report CSD-TR-93-070, Department of Computer Science, Purdue University, November1993.

[PB94] Pitoura E., Bhargava B., "Revising Transaction Concepts for Mobile Computing", In Proceeding of the IEEE Workshop on Mobile Systems and Applications, Santa cruz, CA, December 1994.

[PB95a] Pitoura E., Bhargava B., "Maintaining consistency of data in mobile distibuted environments", Technical Report, Department of Computer Science, Purdue University, 1995.

[PB95b] Pitoura E., Bhargava B., "A framework for providing consistent and recovrable agent- based acces to heterogeneous mobile databases", In Sigmod Record 24(3), pages: 44-49, September 1995.

[PK98] Pham V. A., Karmouch A., "Mobile Software Agents: An Overview", IEEE Communication Magazine, July 1998.

[Rei93] Reichert F., "Integration of Mobile Communication into fixed Networks", NordUnet'93, Helsinki, February 1993.

[Sai96] Sail F., "Localisation et routage dans les réseaux avec sites mobiles", Rapport de mini-projet (1ère année P.G.), USTHB, Algérie, 1996.

[Sat93] Satyanarayanan M., "Mobile Computing", Technical Report, Carnegie Mellon University, September 1993.

[Toh97] Tohrn T., "Programming Language for Mobile Code", Publication Interne N° 1083, IRISA, Mars 1997.

[WC95] Walborn G., Chrysanthis P. K., "Suporting Semantics-based Transaction Processing in Mobile Database Applications", In Proceeding of the 14th IEEE Symposium on Reliable Distributed Systems, Germany, September 1995.

[WC97] Walborn G., Chrysanthis P. K., "Pro-Motion: Managment of Mobile Tansactions", In Procedding of the 4th ACM Annual Symposium on Applied Computing, Special Tack on Database Tehcnology, Pages: 101-108, San jose, CA, March 1997.

[WVF89] Wolfsow C. D., Voorhees E. M., Flatley M. M., "Intelligent routers", In Proceding of the Ninth International Conference on Distributed Computing Systems, Pages: 371-376, ICGHLP95, June 1989.

[YZ94] Yeo L. H., Zaslavsky A., "Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment", In Proceeding of The 14th International Conference on Distributed Computing Systems, June 1994.

ANNEXE

LISTE NON EXHAUSTIVE DE SYSTEMES D'AGENTS MOBILES

Cette annexe présente une liste (non exhaustive) de systèmes d'agents mobiles en cours, ainsi que l'état actuel des infrastructures Agent-Tcl (appelé aussi D'Agents) et FFMAIN. C'est, en fait, le contenu de la page Web maintenue à l'adresse: Http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/preview/preview.html