n°d'ordre

USTL

LIFL Laboratoire d'Informatique Fondamentale de Lille

CNRS

Thèse

présentée à

L'Université des Sciences et Technologies de Lille

pour l'obtention du titre de

Docteur en Informatique

par

David Carlier

Représentation permanente, coordonnée par une carte à microprocesseur, d'un utilisateur mobile

Soutenue le Vendredi 9 Janvier 1998 devant le jury :

Président :	Jean-Marc Geib, Professeur USTL
Rapporteurs :	Marc Bui, Professeur UTC
	Bertil Folliot, MdC habilité Paris 6
Examinateurs :	Vincent Cordonnier, Professeur USTL
	Jean-Marie Place, MdC USTL
	Didier Donsez, MdC LIMAV
	Pierre Paradinas, Gemplus

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE
U.F.R. d'I.E.E.A. Bât M3. 59655 Villeneuve d'Ascq CEDEX
Tél. 03.20.43.47.24 Fax. 03.20.43.65.66

UNIVERSITE DES SCIENCES
ET TECHNOLOGIES DE LILLE

DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M. H. LEFEBVRE, M. PARREAU

PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF, LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

PROFESSEUR EMERITE

M. A. LEBRUN

ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PARREAU, J. LOMBARD, M. MIGEON, J. CORTOIS, A.DUBRULLE

PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

M. P. LOUIS

PROFESSEURS - CLASSE EXCEPTIONNELLE

M. CHAMLEY Hervé	Géotechnique
M. CONSTANT Eugène	Electronique
M. ESCAIG Bertrand	Physique du solide
M. FOURET René	Physique du solide
M. GABILLARD Robert	Electronique
M. LABLACHE COMBIER Alain	Chimie
M. LOMBARD Jacques	Sociologie
M. MACKE Bruno	Physique moléculaire et rayonnements atmosphériques

M. MIGEON Michel	EUDIL
M. MONTREUIL Jean	Biochimie
M. PARREAU Michel	Analyse
M. TRIDOT Gabriel	Chimie appliquée

PROFESSEURS - 1ère CLASSE

M. BACCHUS Pierre	Astronomie
M. BIAYS Pierre	Géographie
M. BILLARD Jean	Physique du Solide
M. BOILLY Bénoni	Biologie
M. BONNELLE Jean Pierre	Chimie-Physique
M. BOSCQ Denis	Probabilités
M. BOUGHON Pierre	Algèbre
M. BOURIQUET Robert	Biologie Végétale
M. BRASSELET Jean Paul	Géométrie et topologie
M. BREZINSKI Claude	Analyse numérique
M. BRIDOUX Michel	Chimie Physique
M. BRUYELLE Pierre	Géographie
M. CARREZ Christian	Informatique
M. CELET Paul	Géologie générale
M. COEURE Gérard	Analyse
M. CORDONNIER Vincent	Informatique
M. CROSNIER Yves	Electronique
Mme DACHARRY Monique	Géographie
M. DAUCHET Max	Informatique
M. DEBOURSE Jean Pierre	Gestion des entreprises
M. DEBRABANT Pierre	Géologie appliquée
M. DECLERCQ Roger	Sciences de gestion
M. DEGAUQUE Pierre	Electronique
M. DESCHEPPER Joseph	Sciences de gestion
Mme DESSAUX Odile	Spectroscopie de la réactivité chimique
M. DHAINAUT André	Biologie animale
Mme DHAINAUT Nicole	Biologie animale
M. DJAFARI Rouhani	Physique
M. DORMARD Serge	Sciences Economiques
M. DOUKHAN Jean Claude	Physique du solide
M. DUBRULLE Alain	Spectroscopie hertzienne
M. DUPOUY Jean Paul	Biologie
M. DYMENT Arthur	Mécanique
M. FOCT Jacques Jacques	Métallurgie
M. FOUQUART Yves	Optique atmosphérique
M. FOURNET Bernard	Biochimie structurale
M. FRONTIER Serge	Ecologie numérique
M. GLORIEUX Pierre	Physique moléculaire et rayonnements atmosphériques
M. GOSSELIN Gabriel	Sociologie
M. GOUDMAND Pierre	Chimie-Physique
M. GRANELLE Jean Jacques	Sciences Economiques
M. GRUSON Laurent	Algèbre
M. GUILBAULT Pierre	Physiologie animale
M. GUILLAUME Jean	Microbiologie
M. HECTOR Joseph	Géométrie
M. HENRY Jean Pierre	Génie mécanique
M. HERMAN Maurice	Physique spatiale
M. LACOSTE Louis	Biologie Végétale
M. LANGRAND Claude	Probabilités et statistiques

M. LATTEUX Michel	Informatique
M. LAVEINE Jean Pierre	Paléontologie
Mme LECLERCQ Ginette	Catalyse
M. LEHMANN Daniel	Géométrie
Mme LENOBLE Jacqueline	Physique atomique et moléculaire
M. LEROY Jean Marie	Spectrochimie
M. LHENAFF René	Géographie
M. LHOMME Jean	Chimie organique biologique
M. LOUAGE Françis	Electronique
M. LOUCHEUX Claude	Chimie-Physique
M. LUCQUIN Michel	Chimie physique
M. MAILLET Pierre	Sciences Economiques
M. MAROUF Nadir	Sociologie
M. MICHEAU Pierre	Mécanique des fluides
M. PAQUET Jacques	Géologie générale
M. PASZKOWSKI Stéfan	Mathématiques
M. PETIT Francis	Chimie organique
M. PORCHET Maurice	Biologie animale
M. POUZET Pierre	Modélisation - calcul scientifique
M. POVY Lucien	Automatique
M. PROUVOST Jean	Minéralogie
M. RACZY Ladislas	Electronique
M. RAMAN Jean Pierre	Sciences de gestion
M. SALMER Georges	Electronique
M. SCHAMPS Joël	Spectroscopie moléculaire
Mme SCHWARZBACH Yvette	Géométrie
M. SEGUIER Guy	Electrotechnique
M. SIMON Michel	Sociologie
M. SLIWA Henri	Chimie organique
M. SOMME Jean	Géographie
Melle SPIK Geneviève	Biochimie
M. STANKIEWICZ François	Sciences Economiques
M. THIEBAULT François	Sciences de la Terre
M. THOMAS Jean Claude	Géométrie - Topologie
M. THUMERELLE Pierre	Démographie - Géographie humaine
M. TILLIEU Jacques	Physique théorique
M. TOULOTTE Jean Marc	Automatique
M. TREANTON Jean René	Sociologie du travail
M. TURRELL Georges	Spectrochimie infrarouge et raman
M. VANEECLOO Nicolas	Sciences Economiques
M. VAST Pierre	Chimie inorganique
M. VERBERT André	Biochimie
M. VERNET Philippe	Génétique
M. VIDAL Pierre	Automatique
M. WALLART Françis	Spectrochimie infrarouge et raman
M. WEINSTEIN Olivier	Analyse économique de la recherche et développement
M. ZEYTOUNIAN Radyadour	Mécanique

PROFESSEURS - 2ème CLASSE

M. ABRAHAM Francis	
M. ALLAMANDO Etienne	Composants électroniques
M. ANDRIES Jean Claude	Biologie des organismes
M. ANTOINE Philippe	Analyse
M. BALL Steven	Génétique
M. BART André	Biologie animale
M. BASSERY Louis	Génie des procédés et réactions chimiques
Mme BATTIAU Yvonne	Géographie
M. BAUSIERE Robert	Systèmes électroniques
M. BEGUIN Paul	Mécanique
M. BELLET Jean	Physique atomique et moléculaire
M. BERNAGE Pascal	Physique atomique, moléculaire et du rayonnement
M. BERTHOUD Arnaud	Sciences Economiques
M. BERTRAND Hugues	Sciences Economiques
M. BERZIN Robert	Analyse
M. BISKUPSKI Gérard	Physique de l'état condensé et cristallographie
M. BKOUCHE Rudolphe	Algèbre
M. BODARD Marcel	Biologie végétale
M. BOHIN Jean Pierre	Biochimie métabolique et cellulaire
M. BOIS Pierre	Mécanique
M. BOISSIER Daniel	Génie civil
M. BOIVIN Jean Claude	Spectrochimie
M. BOUCHER Daniel	Physique
M. BOUQUELET Stéphane	Biologie appliquée aux enzymes
M. BOUQUIN Henri	Gestion
M. BROCARD Jacques	Chimie
Mme BROUSMICHE Claudine	Paléontologie
M. BUISINE Daniel	Mécanique
M. CAPURON Alfred	Biologie animale
M. CARRE François	Géographie humaine
M. CATTEAU Jean Pierre	Chimie organique
M. CAYATTE Jean Louis	Sciences Economiques
M. CHAPOTON Alain	Electronique
M. CHARET Pierre	Biochimie structurale
M. CHIVE Maurice	Composants électroniques optiques
M. COMYN Gérard	Informatique théorique
Mme CONSTANT Monique	Composants électroniques et optiques
M. COQUERY Jean Marie	Psychophysiologie
M. CORIAT Benjamin	Sciences Economiques
Mme CORSIN Paule	Paléontologie
M. CORTOIS Jean	Physique nucléaire et corpusculaire
M. COUTURIER Daniel	Chimie organique
M. CRAMPON Norbert	Tectolique géodynamique
M. CURGY Jean Jacques	Biologie
M. DANGOISSE Didier	Physique théorique
M. DE PARIS Jean Claude	Analyse
M. DECOSTER Didier	Composants électroniques et optiques
M. DEJAEGER Roger	Electrochimie et Cinétique
M. DELAHAYE Jean Paul	Informatique
M. DELORME Pierre	Physiologie animale
M. DELORME Robert	Sciences Economiques
M. DEMUNTER Paul	Sociologie
Mme DEMUYNCK Claire	Physique atomique, moléculaire et du rayonnement
M. DENEL Jacques	Informatique
M. DEPREZ Gilbert	Physique du solide - christallographie

M. DERIEUX Jean Claude	Microbiologie
M. DERYCKE Alain	Informatique
M. DESCAMPS Marc	Physique de l'état condensé et cristallographie
M. DEVRAINNE Pierre	Chimie minérale
M. DEWAILLY Jean Michel	Géographie humaine
M. DHAMELINCOURT Paul	Chimie physique
M. DI PERSIO Jean	Physique de l'état condensé et cristallographie
M. DUBAR Claude	Sociologie démographique
M. DUBOIS Henri	Spectroscopie hertzienne
M. DUBOIS Jean Jacques	Géographie
M. DUBUS Jean Paul	Spectrométrie des solides
M. DUPONT Christophe	Vie de la firme
M. DUTHOIT Bruno	Génie civil
Mme DUVAL Anne	Algèbre
Mme EVRARD Micheline	Génie des procédés et réactions chimiques
M. FAKIR Sabah	Algèbre
M. FARVACQUE Jean Louis	Physique de l'état condensé et cristallographie
M. FAUQUEMBERGUE Renaud	Composants électroniques
M. FELIX Yves	Mathématiques
M. FERRIERE Jacky	Tectonique - Géodynamique
M. FISCHER Jean Claude	Chimie organique, minérale et analytique
M. FONTAINE Hubert	Dynamique des cristaux
M. FORSE Michel	Sociologie
M. GADREY Jean	Sciences économiques
M. GAMBLIN André	Géographie urbaine, industrielle et démographie
M. GOBLOT Rémi	Algèbre
M. GOURIEROUX Christian	Probabilités et statistiques
M. GREGORY Pierre	I.A.E.
M. GREMY Jean Paul	Sociologie
M. GREVET Patrice	Sciences Economiques
M. GRIMBLOT Jean	Chimie organique
M. GUELTON Michel	Chimie physique
M. GUICHAOUA André	Sociologie
M. HAIMAN Georges	Modélisation,calcul scientifique, statistiques
M. HOUDART René	Physique atomique
M. HUEBSCHMANN Johannes	Mathématiques
M. HUTTNER Marc	Algèbre
M. ISAERT Noël	Physique de l'état condensé et cristallographie
M. JACOB Gérard	Informatique
M. JACOB Pierre	Probabilités et statistiques
M. JEAN Raymond	Biologie des populations végétales
M. JOFFRE Patrick	Vie de la firme
M. JOURNEL Gérard	Spectroscopie hertzienne
M. KOENIG Gérard	Sciences de gestion
M. KOSTRUBIEC Benjamin	Géographie
M. KREMBEL Jean	Biochimie
Mme KRIFA Hadjila	Sciences Economiques
M. LANGEVIN Michel	Algèbre
M. LASSALLE Bernard	Embryologie et biologie de la différenciation
M. LE MEHAUTE Alain	Modélisation,calcul scientifique,statistiques
M. LEBFEVRE Yannic	Physique atomique,moléculaire et du rayonnement
M. LECLERCQ Lucien	Chimie physique
M. LEFEBVRE Jacques	Physique
M. LEFEBVRE Marc	Composants électroniques et optiques
M. LEFEVRE Christian	Pétrologie
Melle LEGRAND Denise	Algèbre
M. LEGRAND Michel	Astronomie - Météorologie
M. LEGRAND Pierre	Chimie
Mme LEGRAND Solange	Algèbre
Mme LEHMANN Josiane	Analyse
M. LEMAIRE Jean	Spectroscopie hertzienne

M. LE MAROIS Henri	Vie de la firme
M. LEMOINE Yves	Biologie et physiologie végétales
M. LESCURE François	Algèbre
M. LESENNE Jacques	Systèmes électroniques
M. LOCQUENEUX Robert	Physique théorique
Mme LOPES Maria	Mathématiques
M. LOSFELD Joseph	Informatique
M. LOUAGE Francis	Electronique
M. MAHIEU François	Sciences économiques
M. MAHIEU Jean Marie	Optique - Physique atomique
M. MAIZIERES Christian	Automatique
M. MANSY Jean Louis	Géologie
M. MAURISSON Patrick	Sciences Economiques
M. MERIAUX Michel	EUDIL
M. MERLIN Jean Claude	Chimie
M. MESMACQUE Gérard	Génie mécanique
M. MESSELYN Jean	Physique atomique et moléculaire
M. MOCHE Raymond	Modélisation,calcul scientifique,statistiques
M. MONTEL Marc	Physique du solide
M. MORCELLET Michel	Chimie organique
M. MORE Marcel	Physique de l'état condensé et cristallographie
M. MORTREUX André	Chimie organique
Mme MOUNIER Yvonne	Physiologie des structures contractiles
M. NIAY Pierre	Physique atomique,moléculaire et du rayonnement
M. NICOLE Jacques	Spectrochimie
M. NOTELET Francis	Systèmes électroniques
M. PALAVIT Gérard	Génie chimique
M. PARSY Fernand	Mécanique
M. PECQUE Marcel	Chimie organique
M. PERROT Pierre	Chimie appliquée
M. PERTUZON Emile	Physiologie animale
M. PETIT Daniel	Biologie des populations et écosystèmes
M. PLIHON Dominique	Sciences Economiques
M. PONSOLLE Louis	Chimie physique
M. POSTAIRE Jack	Informatique industrielle
M. RAMBOUR Serge	Biologie
M. RENARD Jean Pierre	Géographie humaine
M. RENARD Philippe	Sciences de gestion
M. RICHARD Alain	Biologie animale
M. RIETSCH François	Physique des polymères
M. ROBINET Jean Claude	EUDIL
M. ROGALSKI Marc	Analyse
M. ROLLAND Paul	Composants électroniques et optiques
M. ROLLET Philippe	Sciences Economiques
Mme ROUSSEL Isabelle	Géographie physique
M. ROUSSIGNOL Michel	Modélisation,calcul scientifique,statistiques
M. ROY Jean Claude	Psychophysiologie
M. SALERNO Françis	Sciences de gestion
M. SANCHOLLE Michel	Biologie et physiologie végétales
Mme SANDIG Anna Margarette	
M. SAWERYSYN Jean Pierre	Chimie physique
M. STAROSWIECKI Marcel	Informatique
M. STEEN Jean Pierre	Informatique
Mme STELLMACHER Irène	Astronomie - Météorologie
M. STERBOUL François	Informatique
M. TAILLIEZ Roger	Génie alimentaire
M. TANRE Daniel	Géométrie - Topologie
M. THERY Pierre	Systèmes électroniques
Mme TJOTTA Jacqueline	Mathématiques
M. TOURSEL Bernard	Informatique
M. TREANTON Jean René	Sociologie du travail

M. TURREL Georges	Spectrochimie infrarouge et raman
M. VANDIJK Hendrik	
Mme VAN ISEGHEM Jeanine	Modélisation,calcul scientifique,statistiques
M. VANDORPE Bernard	Chimie minérale
M. VASSEUR Christian	Automatique
M. VASSEUR Jacques	Biologie
Mme VIANO Marie Claude	
M. WACRENIER Jean Marie	Electronique
M. WARTEL Michel	Chimie inorganique
M. WATERLOT Michel	géologie générale
M. WEICHERT Dieter	Génie mécanique
M. WERNER Georges	Informatique théorique
M. WIGNACOURT Jean Pierre	
M. WOZNIAK Michel	Spectrochimie
Mme ZINN JUSTIN Nicole	Algèbre

Remerciements

Les travaux m'ayant conduit à écrire ce document ont été réalisés au sein de l'équipe *Recherche et Développement Dossier Portable* (*RD2P*) du *Laboratoire d'Informatique Fondamental de Lille* (*LIFL*). C'est pourquoi, je tiens avant tout à présenter mes remerciements au Professeur Vincent Cordonnier qui dirige cette équipe et à Pierre Paradinas actuellement directeur de l'équipe de recherche de *Gemplus*. J'ai apprécié leur encadrement et leurs remarques qui ont permis l'évolution de mon travail.

Je remercie aussi vivement la société *Gemplus* et le *Conseil Régional du Nord - Pas-de-Calais* qui ont bien voulu m'apporter un appui financier par l'octroi d'une bourse de recherche.

Le professeur Takashi Nanya reçoit toute ma reconnaissance pour l'accueil qu'il m'a réservé pendant quatre mois dans son équipe de recherche de l'*Institut Technologique de Tokyo*. Ce séjour dans son équipe m'a notamment permis d'écrire quelques publications scientifiques mais aussi de découvrir un merveilleux pays.

J'exprime, à présent, ma profonde à gratitude aux membres du jury :

- A son président, le professeur Jean-Marc Geib de l'*Université de Lille 1*
- Aux rapporteurs, qui bien voulu examiner mon travail, le professeur Marc Bui de l'Université Technologique de Compiègne et le maître de conférence habilité Bertil Folliot de l'*Université de Paris 6*.
- Aux examinateurs, le professeur Vincent Cordonnier et Jean-Marie Place de l'*Université de Lille 1*, Didier Donsez de l'*Université de Valenciennes* et Pierre Paradinas de la société *Gemplus*.

J'exprime toute ma reconnaissance à Didier Donsez, Sylvain Lecomte et Patrick Trane pour les travaux que nous avons effectués ensemble mais surtout pour l'amitié qu'ils me donnent en partage. De la même manière, j'ai particulièrement apprécié l'aide apporté par le travail de Lamia Aouni et de Christophe Ferres.

Je souhaite aussi remercier, en plus des personnes citées précédemment, Gilles Grimaud et Bernard Noclercq pour la lecture pointilleuse de certains chapitres de ce document.

Il me tient à coeur de présenter ma gratitude aux anciens et actuels membres de l'équipe *RD2P*, du *Département d'Information Médicale* (*DIM*) de Lille et à plusieurs personnes du *LIFL* qui ont participés à égayer ma vie sur mon lieu de travail pendant ces trois années d'étude.

Enfin, pour être complet, il me faut remercier ceux qui m'apportent au quotidien leur amitié ou leur affection : mes amis de Lille, Tokyo, Paris, Marseille ou ailleurs, ma famille et une petite brune du pays du soleil levant nommée Chiharu à qui j'aimerais transmettre un chaleureux *Dômo-arigatô*.

David Carlier

Table des matières

Liste des tableaux

Table des figures

Introduction

Naissance de l'informatique mobile

Depuis près de vingt ans, l'utilisation d'ordinateurs individuels, c'est-à-dire d'outils et d'environnements informatiques personnels permettant le traitement de données, est devenue courante. Après avoir fait son apparition en entreprise, dans les administrations, l'ordinateur est adopté dans beaucoup de foyers dans les pays développés. A tel point qu'il est devenu quasiment indispensable pour beaucoup de personnes qui l'utilisent régulièrement même en déplacement. C'est pourquoi, des ordinateurs portables, des organiseurs électroniques, assistants personnels sont apparus et servent, à tout moment et tout endroit, de traitement de texte, d'agenda, de calculatrice, d'outil de programmation,...

Parallèlement, avec la généralisation de cartes réseau permettant des connexions à un réseau et le développement d'Internet, l'utilisateur a changé son approche vis-à-vis de l'informatique. L'ordinateur est devenu un moyen de communication avec le monde entier, d'échange de messages, de fichiers, d'envoi et de réception de facsimilés,... L'utilisation d'un ordinateur portable permet de disposer d'un outil d'un niveau de performance sensiblement inférieur aux ordinateurs individuels fixes. Cependant, lorsqu'il s'agit de communiquer avec l'extérieur, l'ordinateur doit utiliser des liaisons sans-fil de type *GSM*, par exemple, dont les performances sont loin d'égaler celles des liaisons cablées, sans oublier la faible autonomie en énergie qui oblige les mobiles à être le plus souvent déconnectés du réseau [FZ94]. De plus, comme l'ordinateur portable, ou maintenant *mobile*, sert aussi de téléphone, la demande pousse les constructeurs à proposer des appareils qui s'approchent en taille et en poids des téléphones portables. L'ordinateur mobile idéal, doit disposer des avantages des ordinateurs fixes en puissance de calcul, en mémoire utilisable, en autonomie d'énergie, en débit de communication dans un volume et un poids toujours plus réduits.

L'ordinateur mobile évolue et augmente petit à petit ses performances pour un encombrement et un poids qui diminuent. Cependant, dans le même temps, les ca-

ractéristiques des ordinateurs fixes continuent de progresser au moins dans les mêmes proportions. Si bien que l'écart entre ces différents produits ne se résorbe pas. De même, les performances des réseaux pour mobiles sont en net progrès, mais la marge pour égaler les réseaux cablés reste énorme, l'autonomie limitée impose toujours de fréquentes déconnections et inaccessibilités de l'extérieur, enfin ces réseaux cablés voient aussi leur performance augmenter.

Orientations du travail proposé

Nous nous proposons d'élaborer de nouvelles stratégies afin de réduire considérablement les problèmes liés à l'utilisation d'ordinateurs mobiles. Nous prenons aussi en considération la mobilité de l'utilisateur en plus de celle de l'ordinateur. L'utilisateur doit pouvoir se connecter sur différents terminaux au grès de ses déplacements.

Les technologies *agent mobile* et *carte à microprocesseur* apparaissent comme des moyens adaptés pour atteindre ces objectifs.

Les principales directions sont les suivantes :

- **La délégation de tâches**
  La délégation de tâches d'un ordinateur mobile vers une station fixe apporte de nouvelles perspectives. D'une part, les stations fixes disposent de ressources de meilleures qualités et en plus grande quantité. D'autre part, elles sont continuellement connectées au réseau, donc accessibles. Le modèle client-serveur a été conçu pour ce type de station disposant d'une liaison permanente vers le réseau. Par contre, le mobile ne dispose que d'une liaison intermittente. Un modèle basé sur l'échange de messages de manière asynchrone est donc beaucoup plus adapté.

  Une architecture reposant sur un système à agents mobiles, appelés agents de représentation, est introduite afin de fournir à chaque utilisateur mobile un support personnel de délégation de tâches. Ces agents de représentation sont accueillis sur des stations fixes connectées au réseau. Les propriétés des agents mobiles permettent à l'agent de représentation de se déplacer afin d'être continuellement «proche» de l'utilisateur pour ne pas pénaliser leurs échanges de messages.

  La délégation de tâches donne la possibilité d'envoyer un programme vers l'agent de représentation qui met à sa disposition les ressources de la station fixe d'accueil. La tâche, ayant un accès permanent au réseau, communique dans un modèle client-serveur classique. Par contre, entre l'utilisateur et son agent de représentation, les communications reposent sur un modèle d'échange de messages en mode asynchrone.

- **Le retour des résultats**
  Les tâches envoyées vers l'agent de représentation ont pour but de fournir des résultats à l'utilisateur. Cependant, les caractéristiques matérielles imposées par la mobilité sont particulièrement contraignantes. Les résultats d'une tâche doivent être adaptés à la fois aux préférences de l'utilisateur et aux caractéristiques du matériel utilisé tel que l'interface homme-machine du terminal et

la liaison entre l'utilisateur et l'agent de représentation. Ce traitement sur les résultats vise à réduire la consommation inutile de ressources.

Une tâche fournit des résultats au système de l'agent de représentation qui se charge de les adapter et de les retransmettre lors d'une connexion de l'utilisateur.

- **L'utilisation de différents terminaux**
  La mobilité de l'utilisateur ne signifie pas seulement l'utilisation d'un ordinateur mobile, mais plus généralement l'utilisation d'outils informatiques à partir de différents endroits offrant un accès à un réseau.

  En effet, l'ordinateur mobile est bien souvent un outil utilisé uniquement lors de déplacements, non seulement en raison de faibles performances mais aussi de son interface homme-machine limitée (clavier et écran réduits, pointeur graphique peu pratique). Une station fixe lui est préférée pour les travaux dans l'entreprise ou à domicile.

  Dans ce contexte, interviennent des problèmes liés à la sécurité et à la personnalisation du terminal. La carte à microprocesseur, élément personnel et portable, est un support adapté à ce genre de problème. Elle assure mieux que tout autre composant actuel les fonctions essentielles de sécurité (signature, authentification, chiffrement). En outre, elle est admise tant par sa taille et son usage que par les multiples applications qui l'exploitent comme outil individuel, fiable et très facilement transportable. Par des données et des capacités de traitements dont elle dispose, la carte à microprocesseur permet donc la personnalisation d'une station dans laquelle elle est insérée et une identification/authentification du porteur.

Organisation du document

Le document s'organise en six parties :

- La **première partie** donne un état de l'art permettant d'avoir suffisamment d'éléments dans les trois domaines qui sont à l'intersection des problèmes que nous proposons de traiter : l'informatique mobile, les agents mobiles et les cartes à microprocesseur.

- La **seconde partie** présente les problèmes et les besoins d'une architecture proposant l'utilisation d'agents mobiles. Ces agents mobiles jouent le rôle de support disponible pour l'envoi de tâches par l'utilisateur et comme représentant de celui-ci vis-à-vis de l'extérieur.

- La **troisième partie** propose la modélisation d'une solution basée sur les agents mobiles dédiés à un utilisateur mobile. L'utilisateur est ainsi capable d'envoyer des tâches, de recevoir des résultats adaptés à son matériel et à ses préférences. De plus, il accède à son agent mobile quelque soit la station qu'il utilise.

- La **quatrième partie** décrit le travail d'implantation informatique réalisé pour expérimenter et valider le modèle.
- La **cinquième partie** résume brièvement le document, l'intérêt du modèle et conclut sur les perspectives de ce projet.
- Enfin, la **sixième partie** regroupe les annexes, une liste des acronymes et des références bibliographiques utilisées tout au long de ce document.

Chapitre 1

Etat de l'art

1.1 Introduction

Le système que nous proposons vise à doter un utilisateur de la possibilité de pouvoir accéder à un nombre important de services à tout moment, quelque soit sa localisation et avec un niveau de sécurité important. Ce système s'appuie sur trois domaines différents :

- L'*informatique mobile* permet, grâce aux terminaux mobiles et aux réseaux sans-fil, une large utilisation dans l'espace.
- Les *agents mobiles* constituent un moyen de transférer des traitements vers des stations continuellement actives et disposant de meilleures fonctionnalités.
- Les cartes à microprocesseur apportent un niveau de sécurité élevé et permettent une personnalisation des terminaux utilisés.

Dans ce chapitre, nous détaillons l'état de l'art de ces trois domaines utilisés dans le document.

1.2 L'informatique mobile

L'informatique mobile se développe actuellement à un rythme particulièrement soutenu et est déjà arrivée, sous une forme particulière, dans beaucoup de foyers

grâce à la téléphonie mobile. L'informatique mobile ne signifie pas seulement l'utilisation d'un outil informatique lors d'un déplacement, mais aussi la communication lors de ces déplacements. Elle tend aujourd'hui à se trouver à l'intersection de trois domaines :

- l'informatique
- les biens portables, informatiques ou non, tels qu'un dossier papier, un agenda, une carte d'identité, une clé, un badge d'accès,...
- les réseaux

Définition 1.1 *Nous définissons l'informatique mobile comme l'utilisation informatique mobile pouvant communiquer que nous appellons ordinateurs ou terminaux mobiles.*

Nous détaillerons, d'abord, les contraintes engendrées par la mobilité de l'outil informatique et par les communications durant cette mobilité. Puis, nous décrirons les principaux réseaux actuellement les plus utilisés notamment en téléphonie sans fil. Enfin, nous présenterons quelques voies proposées par la communauté scientifique pour prendre en compte les contraintes de la mobilité qui sont toujours présentes dans les réseaux actuels.

1.2.1 Les contraintes de l'informatique mobile

Les contraintes de l'informatique mobile varient en fonction du degré de mobilité de l'utilisateur. Deux niveaux de mobilité peuvent être identifiés :

1. la mobilité *relative* : l'utilisateur connecte son matériel informatique sur un réseau à partir d'une liaison filaire et l'alimente directement branché sur un réseau électrique. L'utilisateur est cédentaire lors d'une phase d'utilisation.
2. la mobilité *totale* : le matériel informatique peut être utilisé et communiquer avec l'extérieur en déplacement sans aucune liaison physique.

L'informatique mobile telle la plus contraignante est celle qui assure l'utilisateur d'une mobilité totale. C'est donc celle que nous allons développer en compte dans ce chapitre.

Les problèmes rencontrés dans le domaine de l'informatique mobile [FZ94] [IB94] sont essentiellement dûs à des contraintes matérielles :

1. la portabilité de l'équipement impose des contraintes du poids, de taille. Ce qui implique une puissance de calcul et de stockage inférieure
2. la mobilité impose l'utilisation d'un réseau sans fil n'offrant qu'un faible débit
3. l'utilisation de batteries réduit l'autonomie en énergie qui impose une intermittence des connexions

L'équipement mobile se décompose en deux catégories :

1. les *ordinateurs portables*, qu'il faudrait plutôt appeler *transportables*, sont une version sensiblement réduite en poids et en taille des ordinateurs de bureau. Ils offrent des performances légèrement inférieures et disposent de moins de ressources. Leur défaut le plus contraignant est surtout leur encombrement et leur poids.

2. les *téléphones portables* et PDA[1] qui proposent une interface homme-machine réduite, des performances beaucoup plus faibles mais peuvent accompagner l'utilisateur sans gêne importante.

La deuxième catégorie d'équipement mobile dépasse aujourd'hui en nombre d'unités la première. Cependant, les problèmes entre ces deux catégories de mobiles sont similaires (voir le tableau 1.1) même si la contrainte de puissance de calcul sur les ordinateurs portables doit être relativisée.

PDA et téléphone mobile	**Station fixe**
faible puissance de calcul	puissance de calcul importante
souvent déconnecté	connexion constante
faible autonomie en énergie	pas de problème d'autonomie
faible débit	haut débit
peu de ressources disponibles	beaucoup de ressources disponibles
adresse physique dynamique	adresse physique statique

TAB. 1.1 - *Comparaison entre un ordinateur mobile et une station fixe*

A titre de comparaison, le Nokia 9000 est un ordinateur mobile avec liaison radio disposant d'un processeur Intel 386. Il permet des communications téléphoniques, en mode terminal, l'envoi de Facsimiles, de courriers électroniques, l'utilisation d'un navigateur *web*,... Le poids de près de 400 grammes et des dimensions de 173x64x38 mm ne permettent à ce produit de ne disposer que d'un écran en dégradé de gris d'une résolution de 640x200 points, une mémoire totale de 8 ko (système d'exploitation, données personnelles et mémoire d'exécution) [Nokia97].

Le poids et la taille, tous deux limités, rend l'ordinateur mobile beaucoup moins performant et moins autonome en énergie qu'un ordinateur fixe. Ce dernier point est très important, car l'autonomie réduite des batteries encourage une attention particulière afin d'économiser l'énergie par divers moyens comme :

1. la mise en veille du mobile c'est-à-dire la diminution de la fréquence d'horloge, suspension de l'utilisation des disques, de l'écran,...

2. un temps de déconnexion du mobile très important

3. le privilège à la réception de messages qui consomment beaucoup moins d'énergie que l'émission

1. Personal Digital Assistant (Assistant électronique personnel)

4. la délégation de tâches qui permet de reporter une exécution du terminal mobile vers l'extérieur. Ceci permet d'économiser de l'énergie en éteignant le mobile alors que l'exécution se poursuit à l'extérieur [MDC93]

De plus, un équipement totalement mobile ne possède aucun lien physique vers l'extérieur lorsqu'il est utilisé pendant un déplacement. Il communique avec le monde extérieur par une liaison hertzienne. Les communications d'un mobile ont, par conséquent, un même débit et une fiabilité beaucoup plus faible que dans le cas d'une station fixe utilisant par exemple des liaisons optiques. Etant donné que l'utilisateur d'un mobile peut se déplacer, sa localisation physique peut évoluer constamment. Les messages destinés à un mobile seront donc souvent envoyés vers un point différent selon le moment de l'émission. L'adresse physique du mobile varie en fonction des déplacements du porteur d'une zone géographique vers une autre.

1.2.2 Les réseaux sans fil

Les principaux réseaux sans fil s'appuient des des voies de communication hertziennes. Un réseau mobile s'appuie sur des équipements mobiles munis de récepteurs et/ou émetteurs et sur une zone géographique dans laquelle ces équipements peuvent être utilisés. Dans les réseaux mobiles, deux zones peuvent être distinguées :

1. la zone primaire qui est accessible directement

2. la zone secondaire, facultative, qui étend la zone primaire par une passerelle vers un réseau fixe

On peut distinguer deux types de réseaux : les réseaux de mobiles et les réseaux locaux d'entreprise. Les premiers fournissent un moyen de communication destiné à un large public et sur une grande échelle, les autres couvrent uniquement une entreprise et ne sont destinés qu'à son personnel.

1.2.2.1 Les réseaux de mobiles

Les communications pour mobiles sont actuellement en plein essor grâce à une diffusion croissante du nombre de téléphones portables utilisés. Ce moyen de communication est aujourd'hui en place dans la plupart des pays et est devenu économiquement accessible au grand public.

Nous présentons les caractéristiques de fonctionnement des trois différentes générations de réseaux mobiles : l'ancienne, l'actuelle et la future. Les techniques d'accès au réseau et le protocole de communication ne seront pas détaillés.

La première génération La première génération de réseaux de mobiles [P95b] concerne le téléphone mobile avec des communications analogiques et est apparue à la fin des années 70 principalement dans les pays scandinaves. Ces réseaux sont constitués d'un assemblage de plusieurs cellules qui recouvrent un espace géographique dans lequel des communications peuvent être établies. Une cellule délimite une zone géographique dans laquelle se trouve un émetteur/récepteur. Les cellules

se recouvrent partiellement de façon à offrir le plus souvent possible une couverture continue (voir figure 1.1). Leur diamètre est de quelques kilomètres. Cependant il dépend, comme pour les autres générations, du nombre d'utilisateurs dans la cellule. Si la densité d'utilisateurs dans une cellule est grande, la cellule sera petite et inversement.

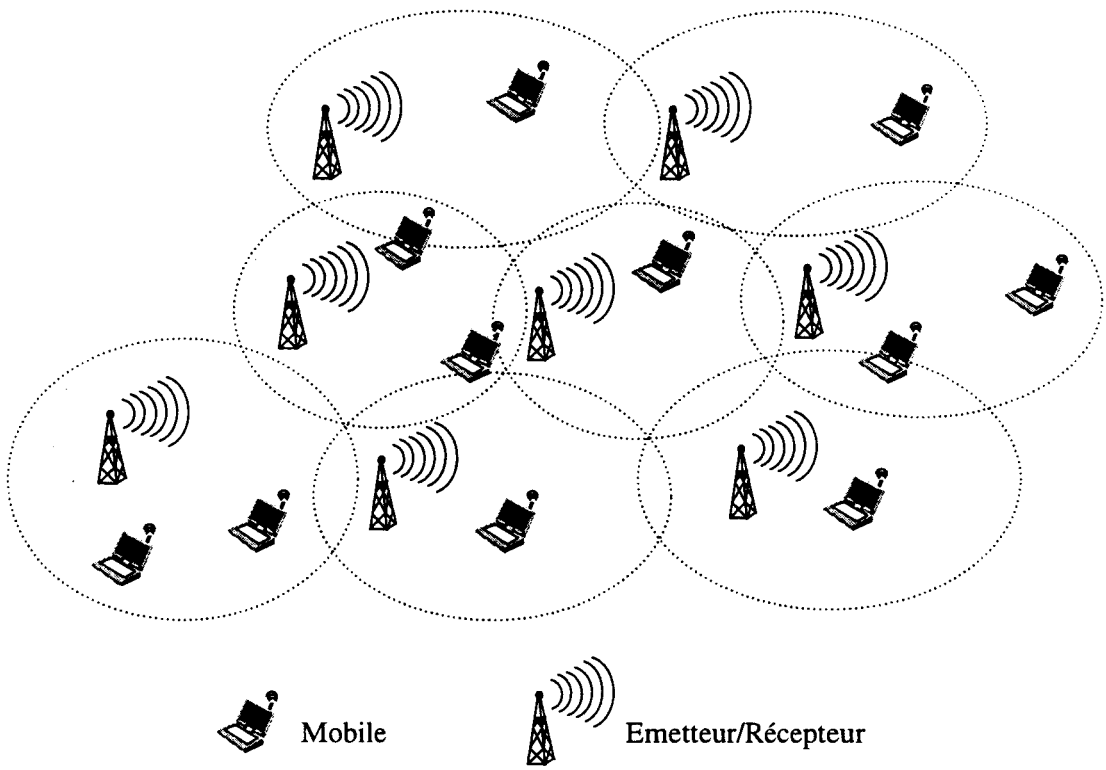


FIG. 1.1 - *Réseau cellulaire*

Note 1.1 *Cette première génération, par manque de normalisation, a donné naissance à des systèmes incompatibles d'un pays à l'autre.*

La seconde génération La seconde génération englobe tous les réseaux de téléphonie mobile numérique. La première norme de cette génération à avoir été mise en exploitation, et la plus largement diffusée dans le monde, est le GSM[2] [MP92]. Les premiers réseaux GSM sont apparus au début des années 90. Cette norme, au départ destinée à harmoniser les réseaux en Europe, s'est répandue sur tous les continents.

L'équipement mobile, dans le GSM, nécessite pour fonctionner l'introduction d'une carte à microprocesseur (voir partie 1.4) appelée carte SIM[3]. Cette carte a comme rôle principal d'identifier l'abonné vis-à-vis du réseau.

2. GSM : Global System for Mobile communications
3. SIM : Subscriber Identity Module

Un réseau GSM est composé de plusieurs cellules gérées chacunes par une station de base appelée BTS[4] qui est en charge des émissions et réceptions de messages à l'intérieur de la cellule. Plusieurs BTS sont reliées à un contrôleur de stations de base nommées BSC[5] (voir figure 1.2).

Un réseau GSM contient des commutateurs MSC[6] qui communiquent avec les différents systèmes radio auxquels sont reliés les BSC. Ces MSC ont accès à :

- un registre de localisation d'abonnés HLR[7] qui est en fait une base de données permettant la localisation dans le réseau d'un mobile abonné
- un registre de localisation de visiteurs[8] VLR[9] qui permet de localiser un mobile abonné au réseau en visite dans la zone couverte par cette MSC

Une BTS gère une cellule délimitée par la portée d'un émetteur/récepteur vers les mobiles. Elle sert d'interface entre la liaison hertzienne vers les mobiles de la cellule et la partie cablée, celle qui relie toutes les stations intervenant dans le réseau. Les BTS sont rassemblées en groupe géographique contrôlé par une BSC formant un sous-système réseau. Les MSC ont un rôle d'interconnexion entre les BTS ou avec les autres réseaux de télécommunication, c'est-à-dire entre la zone primaire et la zone secondaire. Le HLR gère les abonnés et permet leur localisation alors que le VLR enregistre la localisation des mobiles visiteurs connectés à une cellule contrôlée par un MSC. Le VLR, contrairement au HLR est une base de donnée dynamique qui varie en fonction des mobiles visiteurs connectés au réseau. La figure 1.2 représente les différents participants dans un réseau GSM et la hiérarchie existant entre-eux.

Les limites des réseaux GSM sont principalement dûes aux faibles débits (9600 b/s par canaux) et aux communications téléphoniques. Cependant un réseau GSM permet l'envoi d'un message SMS[10] sous forme de texte vers un mobile en veille. Ces SMS peuvent être utilisés pour notifier des évènemements.

Note 1.2 *Même si le GSM est la norme la plus utilisée et donc la plus représentative des réseaux de seconde génération, des concurrents existent. Ils diffèrent du GSM principalement par des bandes de fréquences et des modes d'accès au réseau différents. On peut citer PHS[11] au Japon et PACS[12] aux Etats-Unis.*

La troisième génération Cette génération est en cours de normalisation [P95b]. Elle se distingue principalement par rapport à la génération précédente par la prise en compte d'applications multimédias.

4. BTS : Base Transceiver Station
5. BSC : Base Station Controller
6. MSC : Mobile service Switching Center
7. HLR : Home Location Register
8. Visiteur : abonné d'un autre réseau GSM que celui en cours d'utilisation. Par exemple, un abonné à un réseau GSM français utilise un réseau GSM allemand en Allemagne. Il est donc considéré comme visiteur du réseau allemand.
9. VLR : Visitor Location Register
10. SMS : Short Message Services
11. PHS : Personal Handyphone System
12. PACS : Personal Access Communications Services

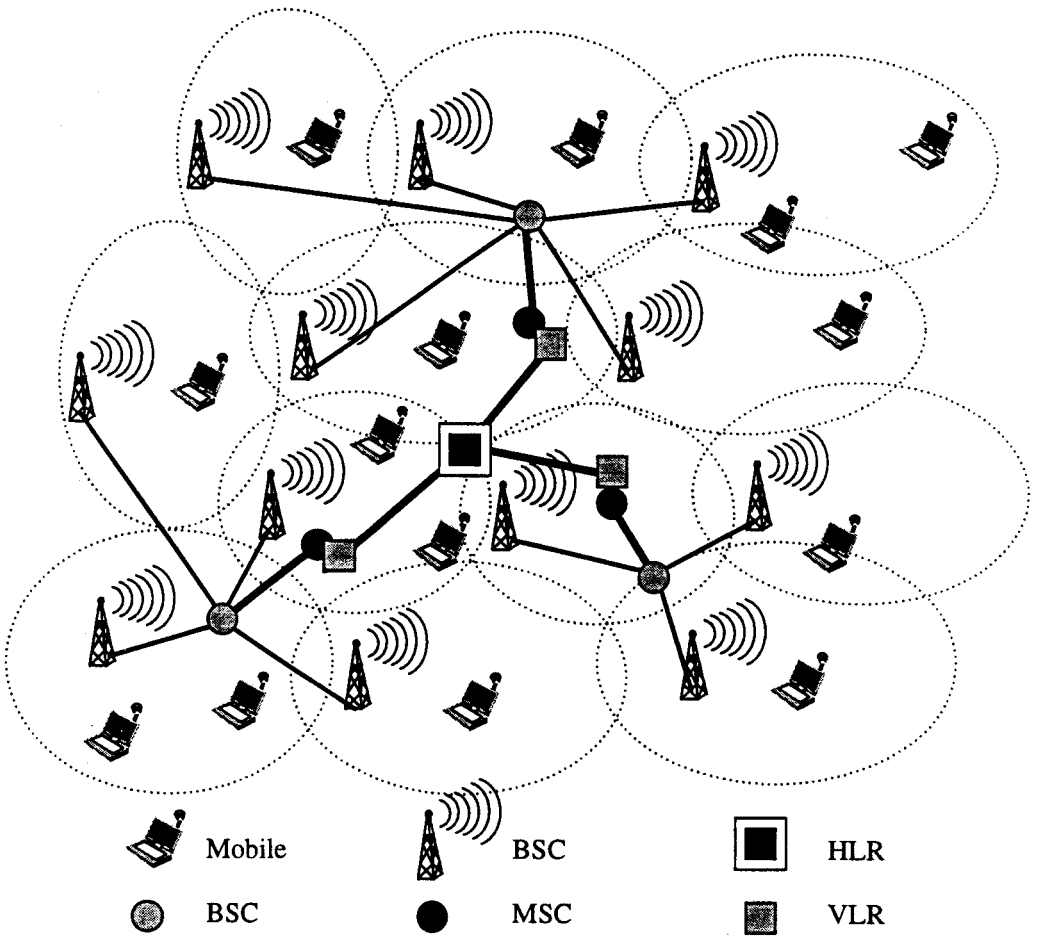


FIG. 1.2 - *Le Réseau GSM*

Afin de garantir un débit beaucoup plus élevé, certaines cellules, les *picocellules* réservées pour les villes à densité très élevée auront une bande de fréquences beaucoup plus élevée et un diamètre de quelques dizaines de mètres (par exemple un couloir de métro). De plus, des algorithmes de compression seront utilisés pour transporter des données, le son ou l'image.

Cette norme vise à permettre la création d'un réseau mondial. C'est pourquoi, l'utilisateur aura un identifiant identique indépendamment de sa localisation dans le monde. Cependant, si l'utilisateur possède deux abonnements, par exemple un abonnement professionnel, l'autre personnel, il pourra recevoir indifféremment un appel professionnel et personnel sur son équipement mobile.

1.2.2.2 Les réseaux locaux sans fil

Nous allons décrire rapidement les réseaux locaux sans fil [P95b] en s'attachant surtout à énoncer les différences par rapport aux réseaux prédents.

Alors que les précédents réseaux offraient une couverture sur un large territoire, les réseaux locaux, comme le nom l'indique, couvrent un secteur réduit tel qu'un

site constitué de plusieurs bâtiments ou même un seul bâtiment. De plus, les trois générations de réseaux de mobiles étaient avant tout dédiés à la téléphonie sans fil contrairement aux réseaux locaux.

Ces réseaux offrent des débits bien plus importants que les réseaux pour mobiles en atteignant les dizaines de Mbit/s [P95b]. Ils sont constitués :

- soit d'un émetteur/récepteur couvrant tout un site par voie hertzienne
- soit d'un ensemble de microcellules interconnectées entre-elles par un réseau cablé utilisant soit les ondes hertziennes, soit les infrarouges pour véhiculer l'information.

Les équipements mobiles peuvent dialoguer entre-eux soit par un intermédiaire fixe soit directement contrairement aux réseaux de mobiles.

Deux normes de réseaux locaux existent. Elles ont été développées séparément et sont incompatibles en particulier au niveau des fréquences.

1. au niveau européen, la norme HiperLAN[13] ETSI TC-RES [Etsi95]
2. au niveau américain, la norme IEEE 802 [Ieee94]

1.2.3 Les orientations des recherches en informatique mobile

Comme cela a été développé dans la partie 1.2.1, les caractéristiques de l'informatique sont très contraignantes pour un utilisateur. Les réseaux actuels offrent des moyens de communications avec l'extérieur de bas niveau et sont soit dédiés à la téléphonie pour les réseaux cellulaires avec un faible débit, soit ne couvrent qu'un espace géographique limité pour les réseaux locaux.

Dans le domaine de la recherche, beaucoup de solutions proposées actuellement reposent sur l'utilisation d'intermédiaires dans la communication entre un mobile et l'extérieur qui permettent une utilisation beaucoup plus large des communications mobiles en particulier au niveau des applications. Nous allons donc présenter les limites des communications directes puis quelques modèles proposant une approche basée sur l'utilisation d'intermédiaires.

1.2.3.1 Limite du contact direct

La limite du contact direct d'un utilisateur mobile est la conséquence des changements de localisation de l'utilisateur, de ses fréquentes déconnexions et de l'utilisation de deux supports de communication différents : l'un sans-fil, l'autre câblé.

13. HiperLAN : High Performance LAN

Le changement continuel de localisation Comme déjà évoqué dans la partie 1.2.1, une personne qui utilise un terminal mobile a la particularité de changer continuellement d'adresse physique lors de ses déplacements. Par conséquent, toute entité désirant contacter, puis établir une communication est théoriquement obligée d'avoir *conscience* de la mobilité de leur interlocuteur et donc, pour maintenir le contact, de rechercher la localisation physique de cet interlocuteur ou, plus vraisemblablement de faire appel à un serveur d'adresses.

L'interface entre deux réseaux différents Les protocoles de communication, le débit, les temps de réponse, la fiabilité [CI94] peuvent être complètement différents entre un réseau cablé et un réseau sans-fil. Par exemple, en ce qui concerne le protocole TCP/IP[14] [S96], les temps de garde sont adaptés à des réseaux ayant un débit relativement important par rapport aux réseaux peu fiables et à faible débit comme ceux des réseaux cellulaires. Des variantes de ce protocole plus adaptées existent comme le protocole *VIP*[15] [TUS94] développé par *Sony CSL*[16] ou le protocole développé à l'Université *Columbia* [IDM91]. Deux protocoles doivent être utilisés en transmission TCP/IP rendant la communication d'une station fixe vers une station mobile quasiment impossible sans intermédiaire.

Les fréquentes déconnexions Un mobile est souvent déconnecté du réseau principalement pour des raisons d'économie d'énergie. Il existe cependant un mode de veille[17] qui permet juste de recevoir des messages, mais n'empêche pas l'épuisement des batteries ou l'absence de couverture du réseau. Par conséquent, ces déconnexions, fréquentes, interdisent l'envoi de message directement vers le mobile à tout moment. Le mobile est donc très souvent inacessible, puisque déconnecté, principalement pour des raisons d'économie d'énergie (voir la partie 1.2.1). Un intermédiaire permet de laisser un message destiné à un mobile comme une boîte de courrier électronique ou tout simplement un répondeur téléphonique.

1.2.3.2 L'intermédiaire fixe

Plusieurs travaux de recherche proposent des systèmes mettant en oeuvre des intermédiaires dans une communication à partir d'un mobile. Nous allons présenter des systèmes qui tirent avantage de l'utilisation d'un intermédiaire: *I-TCP*, le *proxy service* et *Oracle Mobile Agent*.

Le protocole I-TCP Le protocole I-TCP[18] [BB95] est un protocole développé par l'université Rudgers pour la communication dans un réseau cellulaire. Il permet notamment d'utiliser la station de base (appelée BTS dans un réseau GSM, voir partie 1.2.2.1) comme intermédiaire dans une communication TCP/IP entre un ter-

14. TCP/IP : Transmission Control Protocol / Internet Protocol
15. VIP : Virtual Internet Protocol
16. Sony CSL : Sony Computer Science Laboratory
17. basse consommation
18. I-TCP : Indirect - Transfer Control Protocol

minal mobile et une station fixe. Ce protocole concerne principalement les couches *Transport*, *Session* et *Présentation* du modèle OSI[19] [T90], norme de l'ISO[20].

Le mobile ne dialogue pas directement avec une station fixe mais plutôt avec la station de base grâce à un protocole adapté aux mobiles, celui développé par l'université *Columbia* (voir partie 1.2.3.1). La communication entre un mobile et une station fixe est décomposée en deux communications : d'une part, entre le mobile et la station de base actuelle et, d'autre part, entre la station de base et la station fixe. La station de base stocke le message provenant du mobile et le retransmet ensuite par le protocole TCP/IP vers la station destinataire de ce message. Inversement, la station de base sert d'intermédiaire pour un message provenant d'une station fixe et destiné à un terminal mobile. Lorsque le mobile change de cellule, l'intermédiaire, c'est-à-dire la station de base, change. Un protocole entre stations de base permet le transfert de données et d'états des communications en cours de la station de base précédente vers la nouvelle et assure donc une continuité des communications lors d'un changement de cellule.

Ce protocole permet des communications sous forme de sockets [PR92] entre un mobile et une autre station. Plus récemment, l'université *Rudgers* propose désormais, à un niveau supérieur, un protocole baptisé M-RPC[21] [BB96] l'utilisation de RPC[22] [S96].

L'utilisation d'un *proxy* pour une application Le travail sur le *proxy service* est mené à l'université *JAIST*[23] [HKN96]. Alors que le travail précédent proposait l'utilisation d'intermédiaire au niveau du protocole de communication, ici le niveau est beaucoup plus élevé et, se place au niveau de la couche *Application.*

Le concept de ce travail consiste à diviser une application de l'utilisateur en deux parties : une, présente sur le mobile, l'autre, envoyée vers une station fixe. La partie qui réside sur la station intermédiaire fixe est appelée *proxy service.* Ce *proxy service* permet notamment de filtrer les données destinées à l'utilisateur, d'envoyer les données utiles en fonction de l'équipement dont il dispose, ce qui diminue la taille des données circulant sur la liaison entre le réseau cablé et le terminal. De plus, les calculs demandant beaucoup de puissance sont délégués vers le *service proxy.* La partie de l'application sur le mobile et celle correspondante sur la station fixe sont reconfigurées à chaque fois que la configuration matérielle de l'utilisateur change.

Un exemple d'application est donné dans [NH97] utilisant les précédents apports de ce modèle. Il s'agit de permettre la réception d'animation provenant d'une caméra à partir d'un mobile. Pour cela, l'application du mobile utilise un *proxy service* composé d'un ensemble d'objets constitués d'un contrôleur vidéo, un gestionnaire de communication pour chaque type de réseau suceptible d'être utilisé et, selon le débit de la liaison, d'un outil de réduction de la taille d'image grâce notamment à un *proxy service* convertissant des images couleur en images noir et blanc.

Lors d'un changement de zone géographique tel qu'une cellule de réseau cellulaire,

19. OSI : Open System Interconnection
20. ISO : International Standardization Organisation
21. M-RPC : Mobile - Remote Procedure Call
22. RPC : Remote Procedure Call
23. JAIST : Japan Advanced Institute of Science and Technology

les objets accompagnés de l'ancien *proxy service* le plus souvent de leur état du *proxy service* sont transférés vers une autre station fixe afin de créer un nouveau *proxy service* [HN97].

Oracle Mobile Agent *Oracle* propose, par son produit Oracle Mobile Agent [Oma95], un environnement permettant de créer des agents notamment destinés aux utilisateurs qui désirent accéder à des informations de façon sûre et efficace. Ce produit est principalement destiné aux utilisateurs de terminaux mobiles. Il permet de grouper un ensemble de requêtes en une seule. Cette requête est envoyée vers un serveur qui exécute, puis renvoie les résultats. Le tout s'effectue de manière asynchrone, l'utilisateur n'est pas obligé d'être connecté pendant le traitement. Le client peut aussi initier une transaction en envoyant un message vers son agent, qui effectuera la transaction par un ensemble d'échanges de messages avec l'extérieur et retournera les résultats. D'une manière générale, l'agent envoie un message et reçoit une réponse (voir Figure 1.3). Seuls deux messages circulent entre le client et l'agent économisant la liaison entre ces deux entités. Le gain est particulièrement intéressant lorsque le client utilise une liaison sans fil. Enfin un agent peut régulièrement envoyer des données vers l'utilisateur sans nouvelle requête de la part de ce dernier. Il s'agit en fait dans ce cas d'une requête permanente. Ces données peuvent être une nouvelle information souhaitée par le client. L'agent peut, par exemple, surveiller l'évolution de la valeur d'une action pour un courtier en bourse et le prévenir en cas de chute de la valeur.

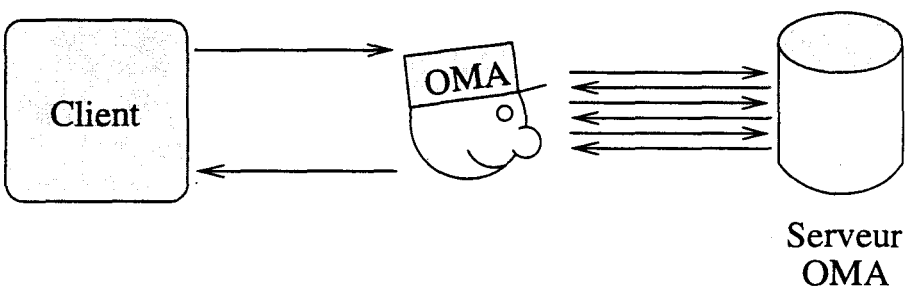


FIG. 1.3 - *Oracle Mobile Agent*

La sécurité est une préoccupation importante [Oma96]. Des mécanismes de signature, d'authentification et de chiffrement basés principalement sur des algorithmes standards tels que le RSA [Rsa91] et le DES [Des93] sont proposés pour le dialogue entre le client et l'agent. Ainsi la sécurité d'une liaison entre un client et un de ses agents peut être considérées comme satisfaisante : les données sont chiffrées et les messages sont signés.

Remarque 1.1 *Bien que OMA signifie* Oracle Mobile Agent, *il faut préciser que ces entités restent fixes, c'est-à-dire sur la même station. Il ne faut pas les confondre avec les* agents mobiles *qui seront introduits dans le chapitre suivant.*

1.3 Les agents mobiles

L'utilisation d'intermédiaires dans une communication est une approche qui permet de pallier les problèmes liés aux fréquentes déconnexions, aux changements de localisation et à l'interface entre un réseau sans fil et un réseau cablé. L'approche des *agents mobiles*, comme cela est montré dans ce document, est une autre manière d'associer un intermédiaire à un utilisateur. Cet intermédiaire lui est dédié et est capable d'agir pour lui, en son nom, même lorsque l'utilisateur est déconnecté. Ce chapitre introduit donc la notion d'agent mobile qui sera utilisée par la suite.

1.3.1 Les agents

Avant de définir ce qu'est un agent mobile, il convient de définir plus généralement ce qu'est un agent en informatique bien que chaque domaine, notamment l'IA[24], par sa branche IAD[25] ait une définition légèrement différente.

Définition 1.2 *Un* agent *est une entité physique ou virtuelle capable d'agir sur elle-même et sur son environnement, de percevoir son environnement, de communiquer et de négocier avec d'autres agents et qui poursuit un but [F94] [D95].*

Le concept d'agent en IAD permet de distribuer de la connaissance vers des entités autonomes appelés agents. Chaque agent a une expertise propre (compétence, connaissance, raisonnement) et collabore avec d'autres agents afin d'entreprendre différentes tâches. Ces agents vont donc communiquer afin de partager leur connaissance et leur expertise. Pour cela, plusieurs solutions existent passant par une mémoire centralisée appelée «tableau noir» ou par envoi de messages notamment grâce à des langages comme KQML[26] [CFF92] qui permettent l'échange de connaissances entre agents.

Le sous-ensemble du monde des agents qui va nous intéresser est celui des agents mobiles utilisés dans les systèmes distribués.

Définition 1.3 *Un* agent mobile *est un programme qui n'est pas lié à l'endroit où son exécution a commencé. Il a la possibilité de se déplacer d'un système dans le réseau vers un autre. La possibilité de voyager permet à un agent mobile d'aller vers le système avec lequel il veut interagir et tirer avantage d'être sur le même site ou réseau que son interlocuteur [Omg97].*

Un agent mobile est donc un concept permettant d'organiser l'utilisation de ressources distribuées. Il peut être vu comme un objet contenant des données et du code, mais aussi doté d'un but et d'une autonomie, il a une tâche à accomplir.

Un système basé sur les agents requiert l'utilisation de deux éléments différents :

- Un *initiateur d'agent* est le créateur de l'instance de l'agent

24. Intelligence Artificielle
25. Intelligence Artificielle Distribuée
26. KQML : Knowledge Query and Manipulation Language

- Un *accepteur d'agents*, ou *site accepteur d'agents*, est un site fournissant des ressources, un environnement d'exécution (CPU, mémoire, accès à des services,...) à un ou plusieurs agents. Un site accepteur peut être vu comme un «serveur d'exécution»

Un agent mobile est donc créé par un initiateur, puis envoyé sur un site accepteur. Cet agent peut, en général (comme dans les systèmes de créations d'agents mobiles présentés plus loin) *migrer* de site en site accepteur puis retournera le résultat de la tâche à son initiateur (voir Figure 1.4) [W96]. Cependant, ce n'est pas le cas dans tous les systèmes.

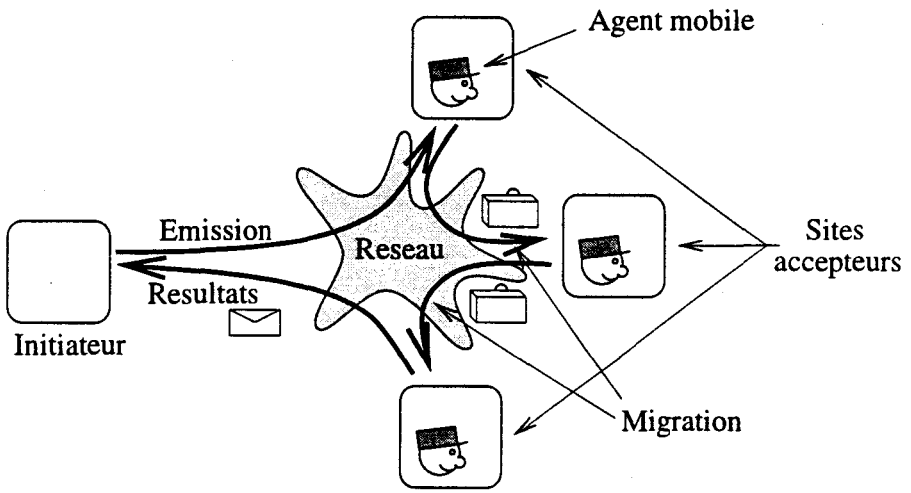


FIG. 1.4 - *Migration d'un agent mobile*

Définition 1.4 *La* migration *est une action qui permet à un agent de se déplacer d'un site accepteur vers un autre afin de poursuivre sa tâche sur le nouveau site [W96]. Lorsque cette opération est possible pour un agent, cet agent devient un agent mobile.*

Note 1.3 *En général, cela signifie que l'agent garde pendant son déplacement tout ou partie des valeurs des données d'exécution (compteur ordinal, contenu de registres, piles et/ou variables d'exécution) du site origine (juste avant la migration) au site destination (juste après la migration).*

Exemple 1.1 *L'illustration qui est souvent utilisée pour expliquer le concept d'agent mobile est celui de l'agent qui voyage de site en site afin d'obtenir le meilleur prix pour un billet de transport. Il peut ainsi négocier avec plusieurs agences de voyage le prix en fonction de préférences de son initiateur et ensuite retourner un compte-rendu des recherches et éventuellement réserver le billet au nom de l'initiateur.*

1.3.2 Besoins fonctionnels des systèmes à agents mobiles

Une plate-forme à agents mobiles doit offrir un environnement adèquat pour assurer les fonctions telles que la négociation et la migration d'un agent vers un site. Les besoins en portabilité du code, moyens de communication offerts, sécurité, diffusion du langage de programmation de ces agents apparaissent comme des bases importantes pour un système à agent mobile.

1.3.2.1 La portabilité

Le code des agents mobiles doit obligatoirement être portable dans des environnements distribués, qui sont aujourd'hui hétérogènes au niveau du matériel, du système d'exploitation ou de la configuration. En effet, un agent, qui arrive sur un site accepteur, doit pouvoir être exécuté sur celui-ci. Pour cela, la meilleure solution est d'utiliser du code qui est interprété par le site accepteur. Dans ce cas, le code de l'agent peut être exécuté par des sites accepteurs qui peuvent avoir une architecture matérielle et un système d'exploitation complètement différent.

1.3.2.2 Les communications

L'agent mobile se déplace sur le réseau avant tout dans un but de simplification et d'optimisation des communications. Un agent est envoyé sur un réseau pour utiliser des ressources extérieures et, pour cela, il doit pouvoir être doté d'outils de communications performants. Par conséquent, le langage dans lequel sera écrit cet agent doit permettre facilement d'effectuer ces communications. L'idéal est de pouvoir invoquer des méthodes distantes afin d'échanger des objets. La plupart des langages ne proposent pas de bibliothèques fournissant un ensemble d'outils de communication, bien souvent seule l'utilisation de sockets est possible. Dans ce cas, des bibliothèques standards devraient être développées à la fois pour les créateurs, initiateurs et accepteurs d'agents. Cependant la situation est en train de changer grâce au développement de systèmes de communication entre objets indépendemment des langages de programmation tels que CORBA[27] [Corba97] de l'OMG[28].

1.3.2.3 La sécurité

Le code de l'agent mobile circule de site en site. Cela implique des problèmes de sécurité. D'une part, pour le site accepteur, du code provenant de l'extérieur est importé et exécuté. D'autre part, au niveau de l'initiateur, du code est envoyé et exécuté sur une station extérieure en son nom.

Au niveau du serveur Un des problèmes les plus importants concernant la sécurité est la protection des sites accepteurs. En effet, exécuter du code venant de l'extérieur sur un site accepteur pose problème, il s'agit d'éviter qu'un agent soit aussi néfaste qu'un virus. En effet, un agent est un programme créé par une personne et qui est envoyé vers un autre site. Si ce programme a un objectif «malveillant» ou

27. CORBA : Common Object Request Broker Architecture
28. OMG : Object Management Group

est involontairement mal conçu, il risque de détruire ou accéder à des ressources non authorisées du site qui l'exécute. L'interprétation du code de l'agent est une solution à ce problème. Ainsi l'accès aux ressources du site accepteur est protégé par l'interpréteur. L'exécution de l'agent est confinée dans un ensemble de primitives autorisées. Par exemple, un agent n'aura pas le droit de lire sur un disque du site accepteur, ni de détruire un processus ne lui appartenant pas,... Toute action réalisée par l'agent est d'abord vérifiée par l'interpréteur. L'interprétation permet un contrôle à la volée de l'exécution [V97].

Au niveau de l'agent L'agent doit être sûr du site accepteur vers lequel il migre. En effet, cet agent est exécuté sur un site qui a accès à toutes les données contenues à l'intérieur de cet agent. Par conséquent, il est nécessaire d'envoyer cet agent vers un site qui garantisse la confidentialité des données de l'agent et qui réalise correctement la tâche pour laquelle il a été créé. Pour cela, un agent ne doit être envoyé que vers un site de «confiance». D'une part, ce site doit être répertorié comme tel et, d'autre part, les connexions à distance, telles que *ftp*, *telnet*, *rlogin* diminuent le niveau de sécurité et doivent donc etre impossibles.

1.3.2.4 Diffusion du langage

Comme défini dans la partie 1.3.1, un agent mobile peut se déplacer de site en site. Plus les sites accepteurs sont nombreux sur un réseau pour un type d'agent mobile, plus leur utilisation sera justifiée et leur apport important. De même, plus le nombre de programmeurs connaissant le langage dans lequel est écrit cet agent est répandu, plus ceux-ci programmeront volontiers de tels agents. Par conséquent, le langage dans le lequel est écrit le programme de l'agent doit être lui-même répandu. Un produit proposant une architecture d'agents mobiles basée sur un nouveau langage entrainera une réticense de la part des programmeurs et des utilisateurs.

1.3.2.5 Le paiement de l'utilisation de ressources

Le code d'un agent exécuté sur un site accepteur va consommer des ressources : de la puissance de calcul, de la mémoire et, éventuellement, des services présents sur le site. Il est, par conséquent, nécessaire pour les fournisseurs de services de pouvoir facturer ou imposer des quotas sur l'utilisation de ces ressources. Ceci peut aussi éviter l'usage abusif, dans un but malveillant ou non, d'agents sur les sites accepteurs afin de réduire toute saturation du site accepteur[W96]. Des mécanismes de paiement de ressource existent aujourd'hui et notamment sur Internet comme les projets Millicent [GMA95] ou NetBill [ST95] qui permettent de payer de façon sûre la consommation de ressources même lorsqu'il s'agit de petite somme. Le paiement est réalisé via un tiers qui se présente comme une autorité de confiance.

1.3.2.6 L'interopérabilité

L'interopérabilité permet de fournir des outils standardisés de communication entre objets. Ce thème est actuellement à l'étude par des membres de l'OMG

[Omg97] pour fournir des spécifications sur le MAF[29]. Le MAF est une collection de définitions et de spécifications qui vise à apporter une interface générique standardisée d'interopérabilité pour les systèmes à agents mobiles. Deux interfaces sont proposées :

1. l'interface `MAFAgentSystem` définit les opérations sur un site accepteur, appelé système à agents, pour la gestion des agents. Les opérations sur un système à agents sont principalement :
   - `receive` pour recevoir un agent sur le système
   - `create` pour en créer un
   - `suspend` pour suspendre son exécution
   - `terminate` pour terminer son exécution
2. l'interface `MAFFinder` définie les opérations pour enregistrer, retirer du registre et de localisation d'agents et de systèmes à agents. Le `MAFFFinder` est un système de service de noms associé à une région administrative. Cette interface est d'autant plus nécessaire que la spécificité du code mobile n'a pas encore été prise en compte dans les travaux de l'OMG. Le `MAFFinder` permet de créer des objets avec une interface standardisée nécessaires pour la localisation d'un agent mobile et d'un système à agents.

Ces interfaces permettent l'interopérabilité entre un système à agents et un client qui peut être un agent. Les communications sont effectuées grâce à un bus CORBA.

Le MAF ne définit que des spécifications sur l'interface, mais ne prend pas en compte l'implantation logicielle des agents mobiles et des services associés. Ces implantations sont de la responsabilité des développeurs d'architectures pour agents mobiles.

1.3.3 Les plate-formes pour agents mobiles

Dans cette section, nous décrivons d'abord Telescript qui a été le premier produit à proposer une plate-forme pour agents mobiles. Ensuite, nous introduirons le langage Java pour présenter les nouvelles plate-formes basées sur ce language. D'autres langages permettent de transporter du code à travers un réseau sans fournir d'outils pour créer des agents mobiles [T97], ils ne seront abordés dans cette section.

1.3.3.1 Les agents *Telescript*

Telescript [W94] de *General Magic* est un des premiers environnements proposés basés sur les agents mobiles. Ce produit était, jusqu'à il y a peu de temps, considéré comme une des références du marché disposant de propriétés telles que portabilité, communication à travers le réseau, sécurité,... Telescript est composé à la fois d'un langage et d'un environnement pour utiliser des agents mobiles. Des agents mobiles vont pouvoir être créés par un client et être envoyés sur le réseau et voyager de site

29. MAF : Mobile Agent Facility

accepteur en site accepteur afin de réaliser une tâche. Des agents Telescript sont capables de rencontrer d'autres agents par la procédure `meet`, de négocier avec eux et ensuite de migrer vers un autre site accepteur par la procédure `go`. Après une migration, l'environnment d'exécution est intégralement restitué à l'agent.

Lorsqu'un agent Telescript est envoyé sur le réseau, l'initiateur n'a plus de contrôle sur lui. L'agent s'exécute, migre de site en site sans interaction possible de l'utilisateur. Il est autonome jusqu'à ce qu'il retourne vers l'utilisateur avec le résultat.

Une particularité de Telescript est de pouvoir attribuer des *Teleclick* à un agent. Le Teleclick est une unité monétaire. L'utilisateur peut charger son agent d'un certain nombre de Teleclicks de façon à ce qu'il puisse payer les ressources qu'il utilise ou les commandes, les achats effectués pour son initiateur.

Il existe deux langages Telescript :

1. Telescript de *haut niveau* est un langage orienté-objet lisible pour le programmeur

2. Telescript de *bas niveau* est un langage interprété par un site accepteur

Le langage de haut niveau est ainsi traduit en langage de bas niveau avant d'être envoyé vers un moteur Telescript contenu par un site accepteur pour être interprété. General Magic a choisi cette solution pour des raisons de performance et de sécurité. L'agent est interprété pour pouvoir contrôler son exécution, mais une phase de précompilation permet d'avoir un agent écrit dans un langage plus proche de la machine.

Note 1.4 *Le seul défaut qui a géné la diffusion de ce produit est la conséquence de la stratégie propriétaire de* General Magic *qui fournissait notamment peu de spécifications. L'apparition de Java, langage très répandu, et des systèmes d'agents mobiles basés sur ce langage a entrainé la fin du développement et de la promotion des agents Telescript par* General Magic.

Remarque 1.2 *Le langage* Obliq *[C95a] a aussi servi de base pour la création d'agents mobiles [BC95] migrant de site en site en préservant le contexte d'exécution. Cependant aucun outil de sécurité ou paiement n'y a encore été implanté contrairement à Telescript.*

1.3.3.2 Le langage Java pour les systèmes à agents mobiles

Cette partie s'impose étant donnée l'importance et l'omniprésence de Java [GG95] dans le monde informatique et, en particulier, dans celui des agents mobiles. Java, par ses qualités de portabilité, de diffusion, d'interprétation est en effet un langage tout à fait désigné pour écrire des agents. D'ailleurs, General Magic vient d'abandonner son produit Telescript pour proposer un produit basé sur Java comme nous le verrons un peu plus loin. Cependant, la constatation suivante s'impose : actuellement l'évolution, comme pour tous les produits basés sur le langage Java, est très rapide. Deux conséquences sont à prendre en compte. D'une part, les caractéristiques des

produits existants évoluent toujours, et souvent à un rythme très soutenu. il faut fiabiliser le produit, ajouter de nouvelles fonctionnalités et tenir compte de l'évolution du langage Java, notamment de l'ajout de nouvelles API[30]. D'ailleurs, la plupart des systèmes d'agents mobiles Java actuels ne sont disponibles qu'en version de test et vont donc encore subir des modifications. D'autre part, de nouveaux produits apparaissent encore aujourd'hui. Par conséquent, après une présentation du langage Java, une liste de plate-formes pour agents mobiles Java sera présentée, cependant elle ne se veut pas du tout exhaustive mais suffisamment représentative.

Le langage Java Le langage Java dispose de plusieurs caractéristiques intéressantes par rapport à d'autres langages comme le résume le tableau 1.2. Le langage Java y est comparé à plusieurs langages ou type de langages assez connus pour des raisons de diffusion dans la communauté des informaticiens ou pour des propriétés exceptionnelles dans un domaine. La liste des langages est très réduite et ne se veut pas du tout exhaustive.

La description du langage Java et de ses qualités seront développées plus en détail dans les paragraphes suivants.

Propriétés	Java	SmallTalk	C++	Perl	Shells
Simplicité	☺	☺	☹	☺	😐
Orienté-objet	☺	☺	😐	☺	☹
Interprété	☺	☺	☹	☺	☺
Portabilité	☺	😐	😐	☺	😐
Dynamicité	☺	☺	☹	☺	😐
Exceptions	☺	☺	😐	☺	☹
Diffusion[a]	☺	☹	☺	😐	☹
Performances	😐	😐	☺	😐	☹

Signification des symboles : ☺ bon 😐 moyen ☹ mauvais

TAB. 1.2 - *Comparaison entre plusieurs langages*

[a] Combinaison du degré de diffusion chez les utilisateurs de langages et du degré d'acceptation du marché

Description de son fonctionnement Le langage Java est un langage interprété après une phase de pré-compilation. Le *programme Java*, écrit par un programmeur, est ainsi envoyé vers un *compilateur Java* qui va le traduire en *code symbolique*. Ce code n'est pas du code directement exécutable par le processeur et doit être envoyé vers une *Machine Virtuelle Java*[31] [LY97].

30. API : Application Programming Interface
31. machine virtuelle Java : traduction française de *Java Virtual Machine*

Le langage Java est un langage qui peut combiner à la fois système d'exploitation et environnement de programmation. La machine virtuelle Java repose sur un système d'exploitation et sert d'interpréteur ou alors elle joue le rôle de système d'exploitation et d'interpréteur en même temps (voir figure 1.5). Elle représente alors la sur-couche directement située au dessus de la couche matérielle. Au dessus de cette machine virtuelle se trouve un ensemble d'API sous la forme de classes Java utilisées par les programmes Java des utilisateurs.

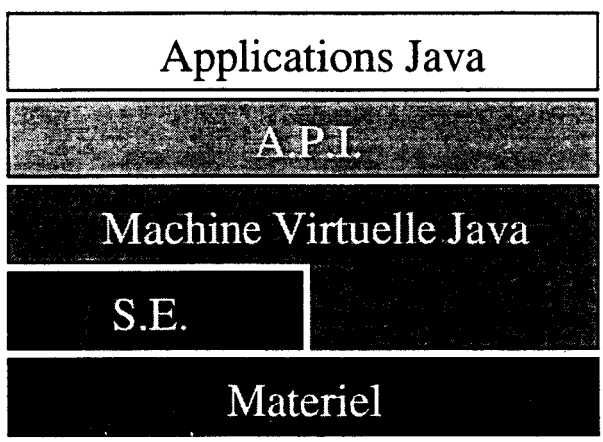


FIG. 1.5 - *Machine Virtuelle Java*

Intérêts du langage Java Les principaux apports du langage Java se situent au niveau des points suivants :

- Portabilité
  L'exécution d'un programme Java repose sur une machine virtuelle. Cette machine virtuelle agit comme une couche entre le système d'exploitation et le programme Java. En théorie, tout programme Java s'exécutant sur la machine virtuelle Java d'une station est portable vers la machine virtuelle Java d'une autre station. En fait, un programme est portable quand il est écrit à partir des API définies dans les spécifications des interfaces du corps Java [Java97a], c'est-à-dire en n'utilisant pas de partie en code binaire propre.

- Facilités de communication
  Le langage Java offre des facilités de communication assez remarquables. Les communications sont très aisées grâce aux API donnant un accès simple aux sockets. Plus récemment, les RMI[32] [Java96] ont permis, grâce à de nouvelles API standardisées, aux objets Java distribués de communiquer entre-eux grâce à un bus de communication. Un objet Java peut invoquer une méthode d'un autre objet Java distant au travers un réseau. Dernièrement, il est devenu possible de faire communiquer des objets Java par un bus CORBA, ce qui

32. RMI : Remote Method Invocation

permet de faire interagir des objets Java avec d'autres objets et d'utiliser les services du monde CORBA [E97].

Une autre facilité liée aux communications est la possibilité de faire «voyager» des classes à travers le réseau. D'une part, il est possible de télécharger [Java97a] une classe à partir de n'importe quel autre site sur internet grâce notamment à la classe abstraite `ClassLoader`. D'autre part, grâce au mécanisme de sérialisation[33] [Java96], un objet peut être envoyé sous forme de chaîne binaire d'un site vers un autre au travers du réseau. De plus, la sérialisation permet la persistance sur disque d'objets Java.

- Sécurité
  L'exécution d'un programme Java est soumis à des règles sécurité assez strictes [FM96]. Il existe en particulier des API Java, dont la classe `SecurityManager`, qui permettent d'utiliser un gestionnaire de sécurité[34]. Celui-ci définit, en particulier, les ressources utilisables par un programme Java. Par exemple, dans le cas d'une *Applet* [GG95], le gestionnaire de sécurité crée une *Sandbox* qui a pour but de limiter l'accès aux ressources telles que les lectures / écritures sur les disques, les accès aux réseaux (seules les communications, pour des raisons de sécurité, vers sur le site d'où provient l'applet sont possibles). La machine virtuelle Java, qui interprète le code Java précompilé, contient un vérificateur de code binaire qui empêche le chargement de séquences de code ne respectant pas les règles imposées par le gestionnaire de sécurité.

  Il existe quelques limitations au niveau de la sécurité de l'exécution d'un programme Java. Il est impossible, pour un programme Java d'obtenir un degré suffisant d'authentification de l'origine d'un code Java téléchargé. Cependant, un ensemble d'API permettant l'utilisation de signatures digitales, qui authentifient l'auteur d'un document, ont été développées par [E96], celles de *Javasoft* sont maintenant disponibles sur leur serveur[35].

  De plus, le principal problème du gestionnaire de sécurité est de ne pas être suffisamment flexible. Un programme Java peut avoir pleinement accès à toutes les ressources ou, comme pour les applets, un accès très limité (et parfois trop limité) aux ressources. Néanmoins, le gestionnaire de sécurité, défini par un ensemble de classes, est remplaçable de façon à accepter par exemple le chargement de code Java sous certaines conditions dépendant du niveau de confiance accordé au serveur. L'utilisation de signature peut être prise en compte pour authentifier l'origine de code Java. Ces classes ont d'ailleurs été développées par [E96].

- Large utilisation
  Utiliser un langage largement répandu et connu par la communauté informatique est un avantage indiscutable. C'est aujourd'hui le cas pour Java. Actuellement, le nombre de machines virtuelles Java installées croît à un rythme

33. sérialisation : traduction française personnelle du mot anglais *serialization*
34. gestionnaire de sécurité : traduction française de *security manager*
35. sous réserve de modifications ultérieures : http://www.javasoft.com/

soutenu, d'autant plus que cette machine est peu onéreuse, voire parfois gratuite. De plus, Java est un langage très polyvalent qui peut être utilisé pour la conception de grosses applications, pour créer des programmes chargés à partir d'un navigateur *World Wide Web* (les Applets) ou comme système d'exploitation [M97] pour des stations de travail comme la *JavaStation* [JavaStation97] de Sun Microsystems ou des très petits systèmes comme la *JavaCard* [JavaCard97] (voir partie 1.4.3.3). De plus, la syntaxe du langage Java est très proche de celle du C++, rendant le langage rapidement familier à un plus large éventail de programmeurs.

Définitions des agents mobiles Java Dans un premier temps, il convient de différencier les termes *code mobile* et *agent mobile.*

Définition 1.5 *Un* code mobile *est une séquence binaire représentant tout ou partie d'un programme exécutable. Il peut, comme un document, être transporté à travers un réseau d'un site vers un autre sur lequel il pourra être exécuté grâce à des propriétés de portabilité [R96].*

Remarque 1.3 *Une classe Java peut être considérée comme du code potentionnellement mobile. En effet, une classe Java peut être téléchargée d'un autre site ou envoyée à travers le réseau par sérialisation (voir la partie 1.3.3.2). Typiquement, des applets sont du code mobile Java envoyé d'un serveur HTTP*[36] *vers un navigateur* World Wide Web, *de même que les servlets [JavaServer97], sont des objets Java envoyé d'un client vers un serveur HTTP. Dans un cas comme dans l'autre, des migrations de site en site ne sont pas prévues par les environnements respectifs.*

Définition 1.6 *Un* agent mobile *est comme son nom l'indique un agent qui accomplit un tâche sur site accepteur et peut ensuite se déplacer de site en site à travers un réseau afin de poursuivre sa tâche.*

Remarque 1.4 *Un* agent mobile *répond à la définition de* code mobile *et peut donc être considéré ainsi.*

1.3.3.3 Les produits existants basés sur Java

Les premiers produits Plusieurs projets ont contribué à proposer des agents mobiles basés sur le langage Java dans sa version 1.0 ne prenant pas en compte des outils tels que les *RMI* et la *sérialisation* (voir la partie 1.3.3.2) apparus dans la version 1.1. Parmi ces projets, on trouve JNA[37] [MSF97] de l'*Université de Paris 6*, *Java-To-Go* [LM96] de *Berkeley* ou *Mole* [SBH96] de l'*Université de Stuttgart.* Cependant, ces produits ne sont que des prototypes, n'ont pas de vocations commerciales et ne semblent pas être destinés à être largement diffusés. Par conséquent, peu de services supplémentaires viendront augmenter les possibilités de ces plateformes. De plus, ils proposent peu d'outils facilitant leur utilisation, mettent peu ou pas de documentation à disposition.

36. HTTP : HyperText Transfer Protocol
37. JNA : JavaNetAgent

C'est pourquoi, nous ne décrirons que trois produits qui semblent les plus aboutis pour le moment : Les *Aglets*, *Odyssey* et *Voyager*.

La quasi-totalité des systèmes d'agents Java partagent les caractéristiques suivantes [KZ97] :

- Ces systèmes fournissent tous un *serveur d'agents*, c'est-à-dire un objet accueillant des agents et leur mettant à disposition un espace pour leur exécution. Il s'agit en fait d'un site accepteur d'agents.
- Les agents peuvent *migrer* d'un serveur d'agents vers un autre en gardant leur état avec eux.
- Le code des agents est *chargé de plusieurs manières différentes* vers les serveurs d'agents. Une spécialisation de la classe abstraite `ClassLoader` de chargement de classes Java définie dans les spécifications du langage Java est utilisée pour cela. Elle utilise alors différents protocoles de communication.
- Tous ces langages sont pleinement *compatibles avec la version 1.1 de Java.* Par conséquent, il suffit d'une machine virtuelle Java et des API adéquates pour pouvoir être en mesure d'exécuter des agents.

Les *Aglets* L'*Aglet*[38] *Workbench* [LC96] est un environnement développé par le *TRL*[39] d'*IBM*[40] qui permet la création et l'utilisation d'agents mobiles Java. Cet environnement est certainement destiné à devenir la référence dans le monde des agents mobiles basés sur le langage Java étant donné qu'il est, pour le moment, le plus abouti.

Plusieurs API appelées J-AAPI[41] ont étaient développées [LO97]. Ces API assurent la portabilité des aglets. Une aglet[42] hérite de la classe Java `aglet` qui est une classe générale abstraite. Des méthodes permettant l'initialisation de l'aglet, la manipulation de ses messages, son activation, sa désactivation, son clonage et sa destruction sont prédéfinies. Les aglets sont basées sur les *RMI* et la *sérialisation* pour la communication à travers le réseau.

Un protocole de transport d'agents ATP[43] est proposé dans l'environnement. Une aglet migre d'un site vers un autre en étant d'abord sérialisée, puis est transportée sous forme binaire d'un site accepteur, appelé *Aglet Host* vers un autre. Sur le site destination, l'aglet est *désérialisée* et réactivée.

Plusieurs points caractérisent les aglets. D'abord, une interface nommée *Tahiti* permet une manipulation assez simple d'un ou plusieurs agents. Au moyen de cette interface, il est par exemple possible de charger une aglet et de l'envoyer sur le réseau accomplir un tâche. Lorsque l'aglet reviendra retourner son résultat chez son initiateur, celui-ci pourra la désactiver et la sauvegarder sur disque en vue d'un usage ultérieur grâce au mécanisme de sérialisation.

38. Aglet : référence aux *AGents et appLETs*
39. TRL : Tokyo Research Laboratory
40. IBM : Internation Business Machines
41. J-AAPI : Java Aglet Application Programming Interface
42. Affectation personnelle du genre féminin au terme d'origine anglaise *Aglet*
43. ATP : Agent Transport Protocol

La sécurité des serveurs est assurée par un gestionnaire de sécurité qui protège les serveurs contre d'éventuelles aglets mal conçues ou malveillantes. Ce gestionnaire est suffisamment flexible pour proposer différents niveaux de sécurité basés sur la signature du code chargé sur le serveur. Par contre, pour l'instant, aucun outil ne permet de garantir la sécurité des aglets, ni le paiement électronique.

Note 1.5 IBM *propose son produit et notamment les API J-AAPI et ATP comme standard pour la création et la manipulation d'agents auprès de l'OMG.*

Odyssey *Odyssey* est un produit de *General Magic* qui remplace Telescript. Le but de *General Magic* serait de fournir un produit se rapprochant le plus possible de son ainé mais en utilisant le langage Java, largement plus universel que ne l'était Telescript, comme langage d'écriture des agents.

Comme pour le produit Telescript, la collaboration entre agents est facilitée en proposant la notion de rencontre *meeting*. Celle-ci permet de programmer la rencontre d'un agent avec un ou plusieurs agents sur une machine d'un réseau. Une autre facilité particulièrement intéressante permet d'échanger des références entre agents d'un même système Odyssey par un mécanisme de publication de références. Un agent peut ainsi publier un objet, ce qui permet ainsi de fournir une référence globale de cet objet. Enfin, Odyssey supporte l'accès aux objets CORBA et l'interrogation de bases de données relationnelles grâce au produit JDBC[44] [HC97] développé par *JavaSoft* et faisant partie du *JDK 1.1*.

Le transport des agents est, comme pour les aglets, basé sur la sérialisation d'objets Java.

L'inconvénient lié à l'utilisation de ce produit est, à ce jour, le manque de documentation y compris sur le serveur *web* de *General Magic*[45].

Voyager Le système *Voyager* [Voyager97] est aussi un produit récent développé par la société *ObjectSpace*. Voyager a été conçu avec plusieurs caractéristiques innovantes.

La notion d'*Objet virtuel* proposée par *ObjectSpace* est particulièrement intéressante. A partir d'une classe Java et du *Virtual Code Compiler*, il est possible de créer un objet virtuel de cette classe qui permettra de créer une instance de cette classe n'importe où sur le réseau. Cet objet virtuel pourra ensuite être utilisé comme *proxy*. L'utilisation d'objets virtuels permet notamment de créer un agent, non pas localement, mais sur un autre site du réseau et de communiquer facilement avec lui. De plus, l'initiateur de l'agent peut dialoguer avec son agent mobile par l'intermédiaire d'un objet virtuel de cet agent.

ObjectSpace prévoit rapidement l'intégration avec CORBA et l'utilisation d'un modèle de sécurité complet et robuste.

44. JDBC : Java DataBase Connectivity

45. Actuellement l'adresse de ce serveur est, sous réserve de modifications ultérieures, http://www.genmagic.com/Odyssey/

1.4 Les cartes à microprocesseur

Depuis son invention dans les années 70 par le Français Roland Moréno, les *cartes à microprocesseur*, communément appelées *cartes à puce*, ont envahi à cadence soutenue les marchés internationaux. Après la France il y a dix ans, le développement s'est, en effet, confirmé en Europe, en Asie et, maintenant, en Amérique du Nord et du Sud. Son champ d'utilisation depuis son apparition comprend principalement les domaines de la téléphonie, de la banque, de la médecine ou des collectivités [GLR88] [CardTech96].

La carte à microprocesseur est un élément indispensable pour garantir un accès contrôlé à des ressources ou services [WD97]. C'est pourquoi, la carte SIM a été choisie comme moyen d'accès à un réseau de mobiles GSM. De plus, en introduisant sa carte dans un terminal mobile anonyme, l'abonné le personnalise, le terminal devient le sien. La carte peut, en effet, contenir des services propres à l'utilisateur.

Ces propriétés seront utilisées dans l'étude pour protéger les communications entre un agent mobile et pour que l'utilisateur puisse accéder, non seulement au réseau de mobiles, mais aussi à son environnement quelque soit le terminal utilisé en le transportant d'un terminal à l'autre au moyen d'une carte à microprocesseur.

1.4.1 Présentation de la carte à microprocesseur

Les cartes à microprocesseur, malgré des performances limitées telles en taille mémoire ou en puissance de calcul, offrent néanmoins un intérêt très important en terme de très faible encombrement et de sécurité.

1.4.1.1 Caractéristiques techniques

La carte à microprocesseur est un objet informatique portable dotée d'un microprocesseur fournissant une capacité de traitement, d'une mémoire pour stocker des informations et de communication avec le terminal dans lequel elle est introduite.

Composants La carte à microprocesseur est constituée d'une plaquette de plastique de la taille d'une carte de crédit classique dans laquelle est incorporé un microprocesseur recouvert d'une pastille de contacts utilisée pour les entrées/sorties et l'alimentation de la carte.

Comme pour la plupart des microprocesseurs des ordinateurs classiques, au microprocesseur de la carte sont attachés, dans la carte, les composants suivant :

- une mémoire indélébile constituée de ROM[46] ou PROM[47] qui permet le stockage du système d'exploitation de la carte et des informations permanentes d'identification ou de sécurité
- une mémoire volatile constituée de RAM[48] qui sert de mémoire de travail lors de traitements effectués par le microprocesseur

46. ROM : Read Only Memory
47. PROM : Programmable Read Only Memory
48. RAM : Random Access Memory

- une mémoire de stockage telle que de l'EEPROM[49] qui permet à la fois l'enregistrement de données et d'applications

- un bloc de sécurité qui garantit la confidentialité des données lors d'attaques physiques de la carte

- des entrées/sorties qui permettent à la carte de communiquer avec l'extérieur

Note 1.6 *Depuis quelques années est apparue la carte à microprocesseur sans contact [TR94] qui reprend les caractéristiques des cartes avec contacts. Cependant la carte est, comme son nom l'indique, dépourvue de contacts. Par conséquent, l'alimentation se fait par un signal électromagnétique envoyé à la carte et les informations sont échangées grâce à une antenne située à l'intérieur de la carte.*

Le système d'exploitation Le système d'exploitation permet la gestion de ces ressources.

Les premiers systèmes d'exploitation pour cartes à microprocesseur étaient écrits en assembleur et enregistrés en langage natif dans la mémoire ROM. Avec l'apparition du C-Card développé à RD2P [GP91], un langage de plus haut niveau, le C, est utilisé pour écrire le système d'exploitation qui est ensuite pré-compilé avant d'être chargé dans la carte. Il est interprété lors de l'utilisation par une machine virtuelle. La JavaCard [G97a] [JavaCard97] reprend ce concept, mais l'étend en permettant, pendant la phase d'utilisation, le chargement de nouveaux éléments de codes appelés API.

Normalisation La plupart des cartes à microprocesseur avec contacts respectent des normes publiées par l'ISO parmi lesquelles :

- la norme ISO 7816-1 qui normalise la taille de la carte et la position du micromodule, c'est-à-dire le microprocesseur [Iso87],

- la norme ISO 7816-2 qui normalise la position des contacts de la carte entre le microprocesseur et le monde extérieur [Iso88],

- la norme ISO 7816-3 qui normalise les signaux électriques et les protocoles de transmission de données entre une carte et un lecteur [Iso89],

- la norme ISO 7816-4 qui normalise les commandes de base, la structure des fichiers et la structure des messages [Iso95],

- la norme ISO 7816-5 qui normalise le codage et les mécanismes de sélection d'une application [Iso94],

- la norme ISO 7816-6 qui normalise le codage et les moyens de localisation des données utilisations par différentes applications de la carte [Iso96].

49. EEROM : Electrically Erasable Programmable Read Only Memory

Performances Actuellement, les cartes à microprocesseur ont des performances relativement modestes (voir tableau 1.3). Cependant, l'évolution des techniques et des architectures, l'utilisation de microprocesseurs tels que des processeurs RISC[50] [CG95], l'utilisation de cartes légèrement plus importantes en taille[51] augmenteront sensiblement les performances de ces différents composants.

Composants	Performances
Microprocesseur	8 à 32 bits
RAM	128-512 octets
ROM	16 kilo-octets
EEPROM	16-32 kilo-octets

TAB. 1.3 - *Caractéristiques des composants de cartes à microprocesseur*

1.4.1.2 Fonctionnement de la carte

Une carte à microprocesseur n'est pas utilisable d'elle-même. Elle doit être auparavant introduite dans un *lecteur de carte*, appelé aussi *terminal*. Un échange d'informations s'effectue alors entre le microprocesseur de la carte et le lecteur. Cet échange s'opère en trois phases qui sont illustrées dans la figure 1.6.

1. La *réponse à la remise à zéro* permet à la carte, après la mise sous tension de la carte par le lecteur, de communiquer au lecteur des valeurs permettent la communication avec elle telle que le protocole et le débit de communication.

2. La *négociation* permet de compléter la phase précédente. Le lecteur envoie une donnée correspondant aux choix émis par la carte durant la remise à zéro, la carte renvoie les mêmes données si elle est d'accord.

3. La phase de communication consiste en un échange de commandes du lecteur vers la carte et de réponses de la carte vers le lecteur.

L'accès aux données enregistrées dans la carte ne peut s'effectuer que par une commande envoyée vers le microprocesseur de la carte. Le microprocesseur est l'interface obligatoire pour accéder en lecture ou en écriture à une information stockée à l'intérieur de la carte. En effet, une commande externe est interprétée par le microprocesseur. Un système d'exploitation présent dans le microprocesseur gère à la fois les entrées/sorties et l'accès à la mémoire de la carte.

1.4.1.3 Intérêts

Par ses qualités de faible encombrement, la carte à microprocesseur permet à un individu de transporter facilement sur lui à la fois un composant capable d'effectuer des traitements mais aussi des données informatiques.

50. RISC : Reduced Instruction Set Computer

51. orientation vers la taille d'une carte PCMCIA (Personal Computer Memory Card International Association)?

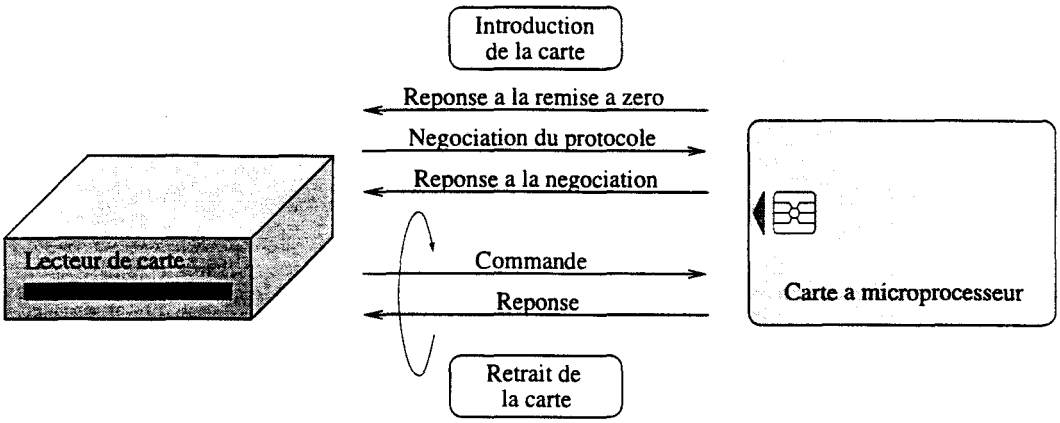


FIG. 1.6 - *Communication entre un lecteur et une carte*

La carte à microprocesseur est particulièrement utilisée lorsqu'il s'agit d'apporter de la sécurité à des données relatives à un individu, notamment le porteur de la carte. Cette sécurité comprend la confidentialité des données vis-à-vis des personnes non autorisées, l'identification et l'authentification du porteur.

D'abord, l'accès à une donnée d'une carte à microprocesseur est protégé physiquement par le bloc de sécurité contre le changement de tension, de fréquence d'horloge, les rayonnements,...

Ensuite, le microprocesseur protège les accès, via les entrées/sorties de la carte, aux données. Alors que, pour une disquette qui permet aussi à un individu le transport d'informations, l'accès se fait directement sur le support, dans le cas d'une carte à microprocesseur, cet accès s'effectue sous le contrôle du microprocesseur. Le système d'exploitation du microprocesseur est capable de garantir la confidentialité des données contenues à l'intérieur de la carte vis-à-vis de personnes non autorisées.

Le fait de posséder une capacité de traitement donne aussi à la carte des possibilités d'identification ou d'authentification sûres à un individu en utilisant des algorithmes cryptographiques présents dans la carte [GUQ91] que seul le microprocesseur peut réaliser. L'algorithme RSA est souvent utilisé pour l'authentification [Rsa78] [Rsa91].La carte est dans ce cas un support pour y stocker des clés privées de chiffrement.

La carte à microprocesseur est enfin un excellent outil pour chiffrer ou déchiffrer des données grâce à ses capacités de traitement et à ses possibilité de stockage de clés. L'algorithme de chiffrement/déchiffrement DES[52] [Des93] est souvent programmé à l'intérieur de cartes réalisant ce genre d'opérations.

1.4.2 Applications

Trois grands domaines d'utilisation des cartes à microprocesseur peuvent être identifiés. L'identification/authentification, le paiement électronique et le dossier portable représentent les principales applications de la carte aujourd'hui.

52. DES : Data Encryption Standard

1.4.2.1 La carte d'accès

Une carte d'accès autorise un accès à une ressource après identification du porteur de la carte. Pour cela, le porteur doit prouver son identité en se faisant reconnaître de la carte par une des façons suivantes :

1. par ce qu'il sait (mode passe, NIP[53])
2. par ce qu'il est (caractéristiques biométriques de l'individu)
3. par ce qu'il fait (comportement de l'individu)

La carte est un support adéquat car elle est capable de stocker un secret qui permet une authentification de l'individu et une capacité de traitement pour vérifier l'identité de la personne [A95].

Trois catégories de cartes d'accès peuvent être énumérées :

1. La *carte d'identification* est une carte permettant d'identifier une personne au préalable avant une action. On peut citer la carte SIM qui permet d'identifier un abonné dans un réseau GSM en vue d'une utilisation de ce réseau.
2. La *carte d'accès logique* permet à une personne autorisée d'accèder à un service après une identification et une authentification.
3. La *carte d'accès physique* permet à une personne autorisée d'accèder à un bâtiment, une pièce, un coffre,...

1.4.2.2 Le paiement

Une carte de paiement est une carte permettant un achat au même titre que des billets de banque, des pièces de monnaie ou un chèque bancaire. Deux types de cartes de paiement sont répertoriées :

1. Les *cartes de pré-paiement* permettent un chargement à la fabrication ou à l'utilisation de la carte en unités de consommation. Ces unités sont ensuite consommées au cours de l'utilisation. La télécarte, le PME[54] sont des applications de cartes de pré-paiement très répandues aujourd'hui.
2. Les *cartes de paiement* sont, contrairement aux précédentes, des cartes de paiement différé. Le montant d'un achat sera débité plus tard sur le compte du porteur. Dans ce cas, la carte à microprocesseur permet une identification du porteur et une description de la transaction.

53. NIP : Numéro d'Identification Personnel
54. PME : Porte-Monnaie Electronique

1.4.2.3 Le dossier portable

Le dossier portable permet le stockage d'informations relatives au porteur de la carte, mais aussi éventuellement à un objet (notamment dans le cas de badge contenant des informations sur un produit). Ce type de carte à microprocesseur utilise la mémoire de la carte pour regrouper des informations dans un support portable tout en préservant la confidentialité des informations vis-à-vis de personnes non-autorisées. Les cartes dossier portable les plus utilisées actuellement sont les cartes santé [P88] [DAP94] [G97b], mais aussi les cartes étudiant ou ville.

1.4.3 Les différents dossiers portables

Les cartes à microprocesseur les plus «sophistiquées» restent les dossiers portables. Les différentes cartes dossier portable se caractérisent entre-elles par leur système d'exploitation qui permet la gestion des données et des applications contenues à l'intérieur de la carte.

1.4.3.1 Les cartes de données organisées en fichiers

Les cartes organisées sous la forme d'une arborescence de fichiers permettent le stockage de données dans un support sécurisé. Une application extérieure peut mémoriser des informations à l'intérieur de la carte dans des fichiers EF[55]. Ces fichiers sont rassemblés dans des répertoires DF[56]. Une racine MF[57] est le point de départ permettant l'accès à cette structure hiérarchique de répertoire en répertoire jusqu'à un fichier. Un mot de passe peut être associer aux répertoires. La norme ISO 7816-4 normalise cette structuration de données hiérarchique et les commandes APDU[58] qui sont envoyés vers la carte pour utiliser cette structure.

Un ensemble de commandes applicatives peuvent être ajoutées, à la fabrication, aux commandes d'accès aux données et chargées en ROM. Par conséquent, ces cartes ne sont pas évolutives, elles ne peuvent pas accueillir de nouvelles commandes

Note 1.7 *La carte SIM reprend cette structure hiérarchique de fichiers et répertoires dans laquelle les fichiers contiennent des informations concernant l'abonné et l'utilisation du réseau GSM [Sim96].*

1.4.3.2 Les cartes base de données

La carte base de données est apparue à RD2P sous le nom de CQL[59] [G92]. Le système d'exploitation est conçu de manière à pouvoir être interrogé par des requêtes SQL[60] simples (`Select`, `Insert`, `Update` ou `Delete`). La structure de données est composée d'un ensemble de tables, vues ou dictionnaires.

55. EF : Element File
56. DF : Dedicated File
57. MF : Master File
58. APDU : Application Protocol Data Unit
59. CQL : Card Query Language
60. SQL : Structured Query Language

La carte CQL est un produit pouvant être intégré beaucoup plus facilement dans les SGBD[61] classiques [VH96]. Elle peut être à la fois considérée comme une base de données à part entière ou comme un élément d'une base de données beaucoup plus importante regroupant un ensemble de cartes.

Une version dite «active» est actuellement à l'étude. Elle permet, à partir de requête effectuée sur la carte CQL, un calcul qui peut modifie d'autres tables [PN97].

1.4.3.3 Les cartes génériques

Les cartes génériques sont des cartes dans lesquelles il est possible de créer de nouvelles applications au cours de l'utilisation de la carte. La carte peut ne contenir aucune application lors de son émission, celles-ci seront chargées au fur et à mesure de son utilisation [P95a]. Ce chargement de nouvelles applications pendant la phase d'utilisation de la carte implique un chargement de code, ce qui va à l'encontre des règles de sécurité qui sont l'un des atouts de la carte. C'est pourquoi, il convient de sécuriser l'exécution du code chargé en le confinant dans un espace de données et d'exécution et en vérifiant la validité de chaque instruction de code exécutée par une machine virtuelle [V97].

La *JavaCard* (déjà citée précédemment) est un exemple de carte générique. Son système d'exploitation est constitué d'une machine virtuelle Java sensiblement «allégée» sur laquelle des API sont disponibles. Des applications [LD97] sont, en fait, des classes Java respectant, les restrictions du standard JavaCard, qui sont chargées à l'intérieur de la carte.

1.5 Conclusion

Les contraintes liées à la mobilité sont nombreuses et pénalisantes pour l'utilisateur. Les réseaux mobiles actuels ne répondent pas aux exigences des utilisateurs visant à pouvoir utiliser un ordinateur mobile avec les mêmes caractéristiques que les ordinateurs fixes. Les solutions proposées dans la communauté scientifique montrent l'intérêt d'utiliser des intermédiaires dans une communication entre un mobile et le monde extérieur. Cependant ces solutions présentent des limites. I-TCP est un protocole de bas niveau. Les *service proxies* ne permettent un gain que pour une seule application, mais pas pour l'utilisateur dans sa globalité. OMA est dédié base de données et ne prend pas en compte la mobilité de l'utilisateur, si bien que l'utilisateur, par ses déplacements, peut être pénalisé par la distance entre son agent et lui.

Les agents mobiles permettent à un utilisateur d'envoyer sur le réseau du code accompagné de données. Ce code est exécuté sur une station afin de remplir une tâche pour l'utilisateur. L'application de tels concepts en informatique mobile offre la possibilité à un utilisateur mobile d'envoyer un agent mobile sur le réseau cablé qui est exécuté et agit au nom de cet utilisateur. Toutes les communications de l'agent sont initiées à partir du réseau cablé. Par conséquent, elles ne souffrent pas d'un manque de débit et ne subissent pas de déconnexion comme dans le cas de processus

61. SGBD : Système de Gestion de Bases de Données

exécutés sur des ordinateurs mobiles communiquant par une liaison sans fil [C95b]. De plus, par sa capacité à se déplacer de site en site, l'agent mobile, comme cela est développé dans les chapitres suivants, peut toujours être disponible pour l'utilisateur initiateur. En se déplaçant de façon à être suffisamment proche de l'utilisateur, cet agent pourra recevoir des ordres et retourner des résultats en profitant des ressources disponibles sur le site accepteur.

La carte à microprocesseur est un outil très utile à la fois en informatique mobile mais aussi dans le domaine des agents mobiles. Elle permet une identification/authentification de son porteur vis-à-vis d'un réseau de mobiles ou vis-à-vis des agents du porteur. De plus, par ses capacités de stockage de données protégées, elle permet de personnaliser des terminaux en ajoutant, par sa présence, des données qui peuvent être confidentielles à un terminal anonyme.

Chapitre 2

Problématique et besoins

Ce chapitre a pour but de présenter les besoins de représentation d'un utilisateur mobile et les problèmes soulevés. Cette représentation est destinée à fournir un accès permanent à plus de services et de meilleure qualité à un utilisateur mobile. La délégation de tâches vers des stations disposant de meilleures ressources et l'adaptation des résultats retournés vers l'utilisateur constituent actuellement un manque certain dans les systèmes pour mobiles.

2.1 Représentation d'un utilisateur mobile

Comme cela a été développé dans la partie 1.2.3, l'utilisation d'intermédiaires apporte beaucoup au niveau des communications entre un mobile et un réseau de services. Dans cette section, nous allons d'abord nous intéresser aux applications pour équipements mobiles. Nous montrerons ainsi les différents bénéfices que procurent un intermédiaire logiciel que nous appellerons, dans un premier temps, *représentation.* Cette représentation est un intermédiaire «actif» dans les communications entre l'utilisateur et ses applications, d'une part, et le monde extérieur, de l'autre. Enfin, afin que cette représentation soit toujours disponible et accessible dans les meilleures conditions, sa mobilité est introduite.

2.1.1 Les applications pour équipement mobile

2.1.1.1 Définition

Définition 2.1 *Une application peut être définie comme une implémentation informatique d'un ou plusieurs services ou à un accès à des services offerts à un ou plusieurs usagers.*

Dans le cas d'un équipement mobile, l'application s'appuie sur un support matériel. Elle se compose de code, d'une interface utilisateur et de données. Le tout permet de fournir un ou plusieurs services.

2.1.1.2 L'évolution des applications mobiles

Les applications pour équipement mobile ont évolué et évoluent toujours en fonction des nouvelles propriétés des supports matériels. Les premiers produits ne proposaient que des architectures mono-applicatives et déconnectées. Maintenant, les équipements mobiles pouvant accueillir plusieurs applications à la fois et disposant de plus de ressources, en particulier en communication, se généralisent (voir figure 2.1).

Les premières applications pour équipements mobiles étaient fournies sur des équipements non connectés à un réseau. Ces applications étaient très limitées et l'application était fournie en même temps que le support matériel. On peut citer la calculatrice, l'agenda ou les jeux électroniques.

Depuis la fin de la décénie précédente, les ordinateurs portables fournissent un environnement comparable aux postes de travail fixes et donc tout un ensemble d'applications présentes sur ces postes.

Les supports portables offrent maintenant des possibilités de communication avec l'extérieur grâce au développement et à la diffusion des réseaux sans fil. Par conséquent, les applications ont évoluées et tendent à vouloir fournir les mêmes services que les applications présentes sur les stations fixes connectées sur un réseau cablé. Cependant, les performances ne sont pas comparables en particulier en matière de débit de communication.

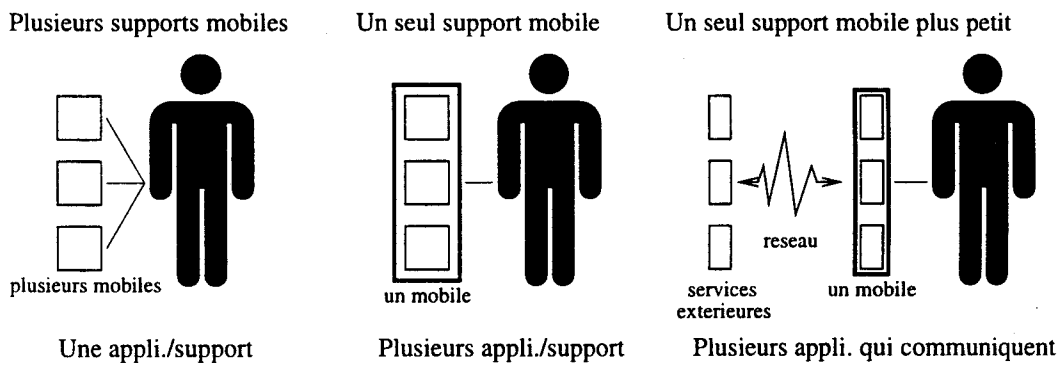


FIG. 2.1 - *Evolution des applications et des équipements mobiles*

2.1.1.3 Les applications pour utilisateurs mobiles

Jusqu'à maintenant, nous présentions l'utilisateur mobile comme une personne qui utilise un terminal mobile. Cependant, il est possible de considérer cette catégorie d'utilisateurs comme une partie de l'ensemble des utilisateurs mobiles. En effet, une personne qui utilise différents terminaux à des endroits différents peut elle aussi être considérée, par extension, comme un utilisateur mobile. Elle se connecte à partir de différents points. On peut aussi remarquer qu'un utilisateur de terminal mobile, n'en est pas moins, dans la plupart des cas, aussi un utilisateur de station fixe.

Par exemple, une personne, qui possède un téléphone portable, garde bien souvent un abonnement pour un poste fixe pour des raisons de qualité de transmission mais surtout pour le coût en communication. L'usage des deux systèmes est bien déterminé, le téléphone fixe pour le domicile ou sur le lieu de travail, le téléphone cellulaire seulement lors des déplacements.

En ce qui concerne le matériel informatique portable, l'utilisation de différents terminaux dépendra du lieu et du moment de la journée. L'utilisateur typique sera la personne qui travaille sur une station de travail au bureau, un micro-ordinateur chez lui et un ordinateur portable ou PDA avec liaison radio lors de ses déplacements [BCK93].

Définition 2.2 *Nous définissons, à partir de maintenant, un utilisateur comme* mobile *lorsqu'il accède au même réseau à partir de localisations géographiques différentes.*

Les terminaux utilisés pour avoir une connexion peuvent avoir des caractéristiques très différentes les uns des autres [KKT93]. Comme cela a déjà été abordé précédemment, le terminal peut-être :

1. Une station de travail qui bénéficie d'une capacité mémoire, de calcul très performante, d'une résolution graphique très importante et surtout d'une liaison vers le réseau avec un débit particulièrement haut.

2. Un micro-ordinateur avec des propriétés assez semblables aux stations de travail mais une connexion vers le réseau par l'intermédiaire d'une carte modem et du réseau téléphonique qui offre un débit sensiblement plus faible.

3. Un ordinateur portable qui, dans le cas d'un PDA, est doté d'une mémoire, d'une puissance de calcul et d'une résolution d'écran, avec des performances beaucoup plus faibles. De plus, une connexion sur un réseau téléphonique sans fil offre un débit particulièrement réduit.

D'où l'intérêt de pouvoir tenir compte du matériel de l'utilisateur et, en fonction, de ses caractéristiques, adapter aux mieux les données par du filtrage, de la compression, de l'envoi automatique ou à la demande de données,...

2.1.1.4 Les applications communiquant avec l'extérieur

Fournir des applications capables de communiquer avec l'extérieur aux équipements mobiles de la même façon que les stations fixes parait bien souvent utopique. La liaison vers l'extérieur a des caractéristiques beaucoup moins avantageuses qu'une liaison cablée et peut être, par conséquent, considérée comme une liaison «dégradée» avec des propriétés telles que :

- faible débit de communication
- très fréquentes déconnexions
- couverture et qualité de la liaison non uniformes

De plus, il faut ajouter les inconvénients dûs à la portabilité tels que la faible puissance de calcul et le nombre réduit de ressources dont dispose un terminal mobile (voir partie 1.2.1).

Les caractéristiques matérielles du terminal d'un utilisateur peuvent varier considérablement [GC94]. L'interface avec l'utilisateur, l'autonomie en énergie, la liaison extérieure (liaison sans fil, liaison téléphonique, LAN,...) sont autant de paramètres qui peuvent changer d'une connexion à l'autre de l'utilisateur. L'utilisateur peut, en effet, relier son ordinateur mobile à un réseau au moyen d'un cable, à un grand écran fixe et à une source d'énergie permanente lorsqu'il en a la possibilité. Le système spportant des applications doit leur permettre de s'adapter en conséquence, non seulement vis-à-vis de l'utilisateur par une interface appropriée mais aussi vis-à-vis de l'extérieur de façon à utiliser la connexion en fonction de ses propriétés.

Enfin, utiliser différents terminaux implique obligatoirement des problèmes liés à la sécurité tels que l'identification/authentification de l'utilisateur ou la confidentialité des données envoyées à partir du terminal ou reçues du terminal. De plus, comme le terminal utilisé peut être anonyme comme un terminal UNIX ou une JavaStation [JavaStation97], l'utilisateur doit pouvoir personnaliser son environnment. La carte à microprocesseur, support de très faible encombrement, disposant de capacités de traitement et de mémoire, est tout à fait adapté à ce genre d'utilisation [L94].

2.1.2 Représentation permanente de l'utilisateur

Pour pouvoir atténuer les inconvénients liés à une liaison sans fil, et pour répondre à l'adaptation des applications aux changements éventuels de cette liaison, l'utilisation de composants logiciels sur le réseau est une approche avantageuse. Cela permet, en effet, de prendre en compte les caractéristiques de cette liaison. Ces composants logiciels peuvent être associés à l'utilisateur ou à ses applications.

2.1.2.1 Limites des solutions actuelles

Les solutions décrites dans la partie 1.2.3 apportent beaucoup au niveau des communications avec des ordinateurs mobiles. La première, I-TCP, offre un protocole de communication. La deuxième basée sur les *proxy services* apporte un moyen de déporter une partie du travail normalement effectué sur le mobile vers une station

fixe. De plus les résultats de ces traitements déportés sont adaptés en fonction de l'environnement et du matériel dont dispose l'utilisateur. Ces deux modèles sont complémentaires car ils concernent des couches de communication différentes. Ils montrent l'apport constitué par l'utilisation d'un intermédiaire dans une communication entre un terminal mobile et une autre entité. En effet, l'intermédiaire adapte les échanges d'informations aux mobiles, ce qui facilite l'intégration des systèmes mobiles dans les systèmes et réseaux actuels.

La principale limite de ces systèmes est de fournir un ensemble de services exclusivement pour un envoi de message pour I-TCP ou une application pour le modèle basé sur les *proxy services*. De plus, rien n'est prévu dans le cas d'une absence de liaison entre le terminal mobile et le monde extérieur, même si le *proxy service* peut être complété par des objets supplémentaires de façon à pouvoir gérer les déconnexions de l'utilisateur mobile.

2.1.2.2 Les intermédiaires «*idéaux*»

A partir d'une définition générale du terme «intermédiaire», nous allons énumérer les services qu'ils pourraient apporter à un utilisateur mobile.

Définition 2.3 *Nous définissons le terme d'*intermédiaire *entre deux entités A et B comme une entité logicielle agissant comme l'interlocuteur de A et de B dans une communication entre A et B.*

Un intermédiaire fixe Un intermédiaire fixe reprend les caractéristiques des modèles présentés précédemment, non plus au niveau d'un envoi de message, mais d'une communication ou d'une transaction (c'est-à-dire un ensemble indivisible d'échanges de messages entre deux entités). Il permet une intégration plus facile des ordinateurs mobiles. Leur déplacement est masqué vis-à-vis des stations fixes par un intermédiaire fixe pendant toute une communication. Cet intermédiaire se contente de retransmettre les messages entre le terminal mobile et les stations fixes.

Un intermédiaire permanent Un intermédiaire permanent contribue à éviter qu'un utilisateur devienne complètement inaccessible pour le monde extérieur. Contrairement aux modèles précédents, il permet à l'utilisateur mobile d'avoir toujours une entité qui le représente, même de façon dégradée, connectée au réseau. Ainsi cet intermédiaire peut recevoir des données envoyées vers l'utilisateur. Il peut obtenir par exemple les réponses aux requêtes envoyées lors de la précédente connexion de l'utilisateur, tout cela sans nécessairement l'existence d'une connexion de l'utilisateur au réseau. Cela autorise l'extinction de l'équipement mobile et donc une économie d'énergie.

Un intermédiaire assistant Cette propriété, comme pour les proxy services (voir partie 1.2.3.2), permet à l'utilisateur de recevoir des données en fonction des caractéristiques matérielles de son terminal (taille et couleur de l'écran, débit de la liaison entre le terminal et le réseau cablé, capacité de stockage, etc...) et de ses préférences (confidentialité, priorité, etc...). L'intermédiaire effectue donc un prétraitement à

partir des données reçues de l'extérieur et destinées à l'utilisateur mobile afin de les adapter en fonction de caractéristiques matérielles et des préférences.

Cet intermédiaire joue donc le rôle d'interface logicielle entre l'utilisateur et les réseaux utilisés. C'est pourquoi, il doit pouvoir être joignable quelque soit le moyen de communication utilisé (liaison téléphonique, liaison cellulaire, LAN, etc...).

Un intermédiaire de délégation Cet intermédiaire permet de déléguer des tâches du terminal mobile vers une station fixe. Il accepte des tâches de l'utilisateur ou d'une de ses applications et mettrait ainsi à disposition un environnement d'exécution offrant beaucoup plus de ressources telles que de la puissance de calcul, de la mémoire, des moyens de communication qu'une station mobile. De plus, une tâche envoyée vers cet intermédiaire peut continuer son travail même si l'utilisateur se déconnecte du réseau.

2.1.2.3 Représentation d'un utilisateur

Définitions A partir de la définition générale du mot «représentation», nous allons proposer une définition dans le contexte de notre travail.

Définition 2.4 *Dans le vocabulaire courant, une* représentation *de quelqu'un est quelque chose qui remplace, qui agit à sa place [R97].*

Définition 2.5 *A partir de la définition précédente, nous pouvons définir une* représentation logicielle d'un utilisateur *comme une entité logicielle capable d'accéder à des services au nom de cet utilisateur et ayant une disponibilité permanente afin de pouvoir remplacer cet utilisateur.*

Dans le cas d'un utilisateur mobile, l'utilisation d'une représentation associée à cet utilisateur possédant les propriétés citées dans la partie 2.1.2.2 permet d'augmenter considérablement les capacités des équipements mobiles. Les rôles de la représentation sont :

1. Une représentation fixe : la représentation se présente comme une interface fixe vis-à-vis de l'extérieur pour communiquer avec l'utilisateur
2. Une représentation permanente : la représentation est continuellement accessible même lorsque l'utilisateur n'est pas connecté
3. Un assistant : adapte les données destinées à l'utilisateur
4. Un support de délégation de tâches

La représentation vis-à-vis des applications de l'utilisateur Le rôle de la représentation vis-à-vis des applications est principalement d'être l'interlocuteur privilégié de celles-ci sur le réseau. En fait, adapter les communications au terminal de l'utilisateur et à sa liaison extérieure implique à la fois :

1. Une adaptation des données au débit par le filtrage et la compression

2. La possibilité de communications asynchrones

3. Eventuellement un chiffrement pour sécuriser les liaisons peu fiables

Un des objectifs est, de plus, de permettre sans modification des services et serveurs sur le réseau leur utilisation afin de garantir une intégration de la mobilité sans remettre en question l'existant. C'est pourquoi, une application pour mobile doit se trouver sur le réseau afin, d'une part, d'être une interface entre l'équipement mobile et le monde extérieur et, d'autre part, d'adapter les communications en fonction de critères programmés par l'application vis-à-vis de la liaison terminal-extérieur.

2.1.3 Vers une solution basée sur les agents mobiles

L'utilisation de la représentation d'un utilisateur mobile comporte néanmoins quelques limites, notamment dûes à la variation de la distance entre l'utilisateur et sa représentation sur le réseau. Cette distance est un paramètre qui a d'importantes conséquences sur la latence et le temps de transfert des messages échangés. L'optimisation constante de cette distance implique une mobilité de la représentation. Le concept d'agent mobile (voir partie 1.3) conjugue à la fois les concepts de représentation d'un utilisateur et la mobilé de cette dernière.

2.1.3.1 Limites de la représentation fixe

Une représentation fixe telle qu'elle a été définie précédemment a l'avantage de servir d'intermédiaire pendant une communication. Cependant, le fait d'être toujours fixe peut aussi se révéler être un handicap. En effet, tout message envoyé du terminal de l'utilisateur vers le monde extérieur, et inversement, circule obligatoirement via la représentation fixe de l'utilisateur. Dans le cas d'un modèle client-serveur (plus détaillé dans la partie 2.2.1.1), une communication entre un client et un serveur se décompose un envoi de message du client vers le serveur et un retour du serveur vers le message. Avec une représentation, quatre envois de messages sont nécessaire : utilisateur-représentation, représentation-serveur, serveur-représentation, représentation-utilisateur. Le coût global en temps de communication, indépendamment de la taille des données, risque dans certains cas d'être beaucoup plus pénalisant comme cela est démontré ci-après.

Définition 2.6 *Soit C_{min} représentant le coût minimum d'une requête effectuée par l'utilisateur sur un serveur dans la configuration courante (localisation de l'utilisateur et du serveur). Soient C_{loc} et C_{dist} représentant le coût d'un envoi de message respectivement local et distant. Soit la fonction $C(A, B)$, représentant le coût de communication entre l'entité A et B.*

Note 2.1 $C_{min} > 2 * C_{loc}$

Définition 2.7 *Une zone géographique représente un domaine dans lequel toute communication entre deux entités présentes dans une même zone est considérée comme locale. Toute communication entre deux entités présentes dans des zones géographiques différentes est considérée comme distante.*

Prenons le cas d'un utilisateur habitant dans une zone géographique arbitraire ZG1 et ayant donc sa représentation dans cette zone ZG1. Lorsqu'il envoie une requête vers un serveur présent dans sa zone, le coût en temps s'élève à :

$$C1_{min} = 2 * C_{loc}$$

$$C1 = C(term, repr) + C(repr, srv) + C(srv, repr) + C(repr, term)$$
$$C1 = 4 * C(ZG1, ZG1)$$
$$C1 = 4 * C_{loc}$$
$$C1 = 2 * C1_{min}$$

Remarque 2.1 *Dans le cas d'une communication directe client-Serveur, le coût s'élèverait à* $2 * C_{loc}$*, c'est-à-dire la moitié.*

Cet utilisateur voyage et se déplace dans une zone géographique différente ZG2 de façon à ce que le coût d'une communication locale soit négligeable par rapport au coût d'une communication distante. L'utilisateur désire envoyer une requête vers un serveur local, c'est-à-dire dans la même zone ZG2 que l'utilisateur. Comme la représentation, toujours dans la zone ZG1, reste un intermédiaire (voir Figure 2.2), le coût s'élève donc à :

$$C2_{min} = 2 * C_{loc}$$

$$C2 = C(term, repr) + C(repr, srv) + C(serv, repr) + C(repr, term)$$
$$C2 = 2 * C(ZG2, ZG1) + 2 * C(ZG1, ZG2) = 4 * C_{dist}$$
$$C2 \gg C2_{min}$$

Le coût d'une requête en utilisant une représentation dans la même zone géographique est du même ordre que le coût d'un appel direct. Par contre, il devient beaucoup plus important lorsque la représentation se trouve dans une zone différente de l'utilisateur et du serveur.

Le tableau 2.1 présente les coûts d'utilisation de la représentation fixe en fonction de la position locale ou distante du client, de la représentation et du serveur.

2.1.3.2 Besoin d'une représentation mobile

On s'aperçoit aisément que, lorsque la représentation est dans une zone différente soit de l'utilisateur, soit du serveur, le surplus en coût de communication est relativement faible, juste deux communications locales. D'où l'intérêt d'utiliser une représentation mobile de l'utilisateur en s'inspirant notamment des travaux réalisés sur les agents mobiles (voir partie 1.3) de façon à réduire ou le coût de la transmission entre l'utilisateur et la représentation ou celui entre la représentation et le serveur. Cependant rapprocher la représentation du serveur est inconcevable étant donné que l'utilisateur peut émettre des requêtes vers plusieurs serveurs à la fois dans des zones géographiques différentes. Par conséquent, une représentation «suivant» l'utilisateur lors de ses déplacements réduit considérablement la distance terminal de

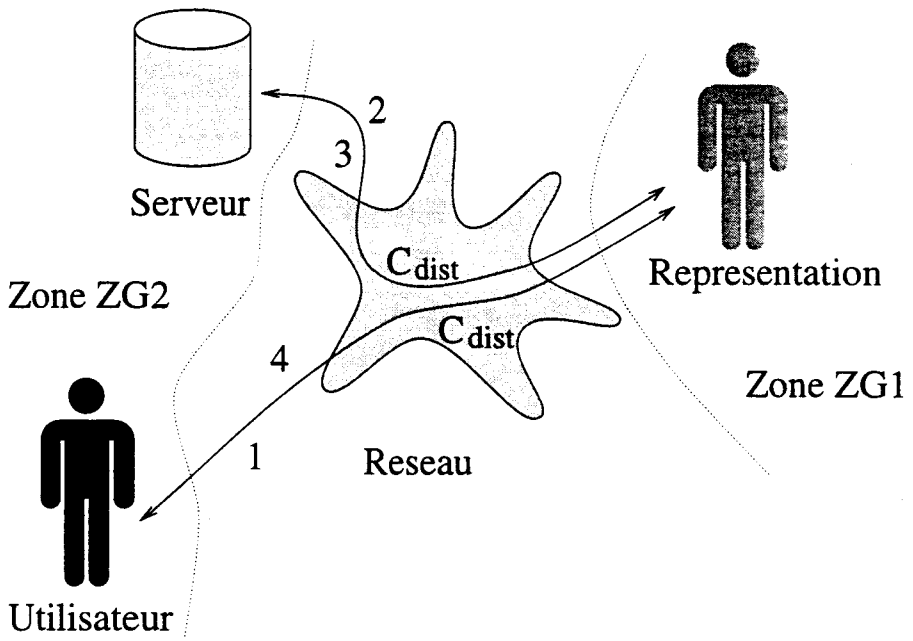


FIG. 2.2 - *Limite de la représentation fixe*

l'utilisateur-représentation et donc les coûts de communication, la latence des messages mais aussi le traffic de messages circulant à travers le réseau quelque soient le nombre et la localisation des services utilisés.

Nous appelerons une représentation mobile basée sur les technologies d'agent mobile un *agent de représentation.*

2.2 Délégation de tâches

Comme évoqué dans la partie 2.1.2.2, la délégation de tâches contribue à déporter une partie de l'exécution des tâches du terminal de l'utilisateur vers une station sur le réseau, ce qui permet de bénéficier d'un environnement d'exécution bien plus performant, et disposant de plus de ressources de traitement, de stockage et de communication pour y réaliser une tâche.

2.2.1 Présentation de la délégation de tâches

2.2.1.1 Le modèle client-serveur

Une communication client-serveur met en place deux intervenants [CDK94] :

1. Un *client*
   L'entité qui demande un traitement à un *serveur* dans le but d'obtenir un résultat.

2. Un *serveur*
   L'entité qui reçoit une demande de traitement afin de produire le résultat.

Zone des Eléments	Coût avec représentation	Coût sans représentation	Diff.	Rapp.
U, R, S : ident.	$4 * C_{loc}$	$2 * C_{loc}$	$2 * C_{loc}$	2
U, R : ident., S : diff.	$2 * C_{loc} + 2 * C_{dist}$	$2 * C_{dist}$	$2 * C_{loc}$	2
U, S : ident., R : diff.	$4 * C_{dist}$	$2 * C_{loc}$	$4 * C_{dist}$	grand
U : diff., R,S : ident.	$2 * C_{loc} + 2 * C_{dist}$	$2 * C_{dist}$	$2 * C_{loc}$	équiv.
U, R, S : diff.	$4 * C_{dist}$	$2 * C_{dist}$	$2 * C_{dist}$	grand

U : Utilisateur, R : Représentation, S : Serveur
ident. : identique, diff. : différent, équiv. : équivalent

TAB. 2.1 - *Coût en fonction des zones de l'utilisateur, représentation, serveur*

Elle se décompose en trois étapes :

1. la transmission d'une requête du client vers le serveur
2. l'exécution de la requête par le serveur
3. la transmission d'une réponse vers le client

2.2.1.2 La délégation de tâches

Définition 2.8 *La délégation de tâches est un modèle basé sur le celui du client-serveur qui consiste à déporter l'exécution d'un programme d'un client vers un serveur afin de produire un service.*

Nous pouvons distinguer quatre éléments intervenant dans la délégation de tâches :

1. Le *client*
   L'entité qui demande l'éxécution d'une tâche à un *serveur* dans le but d'obtenir un résultat. C'est l'initiateur de la délégation.
2. La/les *tâche(s)*
   Le(s) travail(aux) qui est(/sont) délégué(s) par le client vers le serveur.
3. Le *serveur*
   L'entité qui reçoit la demande d'exécution d'une tâche afin de produire le résultat. C'est le récepteur, l'accepteur de la tâche à accomplir.
4. Le *résultat*
   Le résultat peut être soit l'accomplissement d'un calcul décrit par la tâche et retourné au client sous forme d'un ou de plusieurs résultats espacés ou non dans le temps, soit un changement d'état d'une entité extérieure au client. Dans le premier cas, le service répond à une tâche *requête* en retournant au moins un résultat, ce qui est comparable à la définition d'une *fonction* en programmation. Dans le second cas, le client n'attend pas de réponse, mais la tâche modifie des entités faisant partie de son environnement, ce qui correspond au concept de *procédure*. Il faut alors retourner un acquittement ou un compte-rendu.

2.2.1.3 Intérêt de la délégation de tâches

Les raisons pour lesquelles une délégation de tâches est demandée peuvent être classées en plusieurs catégories :

1. Accès à des ressources supplémentaires ou gain de performance
   La délégation d'une tâche vers un site disposant de plus de ressources, justifie souvent cette opération. Ces ressources peuvent provenir de plusieurs domaines : communications (cas d'un routeur vers lequel une tâche de redirection est demandée), périphériques (serveur d'impression), puissance de calcul (serveur multi-processeurs ou disposant d'un processeur dédié à certains calculs).

2. Demande de compétences
   Ce point est, dans une certaine mesure, lié au précédent. Cependant, les ressources supplémentaires acquises par la délégation de tâches et fournies par le serveur relèvent d'une habilitation fournie par une autorité. Dans ce cas, le serveur de la délégation fournit des ressources sous le contrôle de cette autorité (serveur de clés, de chiffrement/déchiffrement de données, d'autorisation, de capacités,...).

3. Parallélisme de tâches
   Ce gain peut être obtenu en déléguant la tâche vers un serveur pouvant l'accomplir plus rapidement et/ou en déléguant la tâche afin de réaliser une autre tâche en parallèle. Cela permet, en informatique mobile, pour l'utilisateur d'avoir une tâche s'exécutant sur un serveur alors que son terminal exécute d'autres processus ou est éteint.

2.2.1.4 Différentes formes de délégation

Il existe deux sortes de délégations. Elle peut être initiée soit par l'envoi d'une requête dans le modèle client-serveur classique, soit par l'envoi d'un morceau de programme (code ou algorithme) à exécuter.

L'envoi d'une requête L'envoi d'une requête reprend la définition classique du modèle client-serveur. Cela consiste à demander l'exécution d'une opération prédéfinie à une machine distante. Le client envoie des données (*paramètres*) au serveur afin d'obtenir le service (voir Figure 2.3).

Le RPC[1] [BN84] est un exemple qui répond tout à fait à la définition précédente. Un appel de procédure est effectué vers un autre processus serveur (local ou distant), des paramètres peuvent éventuellement personnaliser, adapter l'exécution au client et, après la fin de l'exécution de la tâche, le client reçoit du serveur un ou des résultats.

L'avantage d'une telle délégation est de permettre l'utilisation de code et de ressource se trouvant sur un site distant. Le client ne connait pas le code du serveur mais juste les spécifications. Le code est sur le serveur, ce qui permet un transfert

1. RPC : *Remote Procedure Call* - Appel de procédure distante

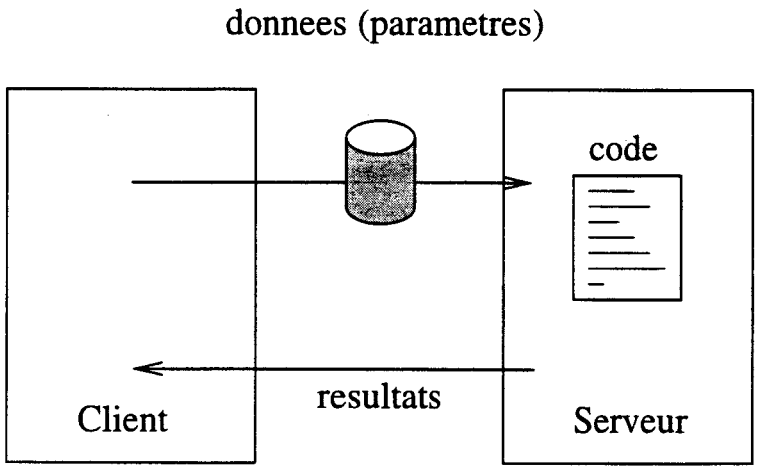


FIG. 2.3 - *Délégation par requêtes*

de données plus réduit, une mise en commun pour plusieurs clients et une mise à jour de tâches générale pour tous les clients potentiels.

La délégation par requêtes entraine néanmoins un problème important. Le code se trouve chez le serveur et donc le client ne peut pas le modifier, l'adapter à ses besoins. La seule façon de modifier son exécution passe par la modification des paramètres de la requête.

L'envoi de programme La délégation de tâches peut être réalisée par l'envoi du code (compilé ou sous forme de script[2]) et des données à partir du client vers le serveur. Ce code décrit, par un ensemble de commandes, la tâche à réaliser par le serveur afin de produire un ou des résultats qui seront transmis à la fin de la tâche au client ou un changement d'état sur un site distant. Cette délégation suppose que le serveur soit capable de comprendre et d'utiliser le programme qui lui est envoyé (voir Figure 2.4).

La délégation répondant aux caractéristiques précédentes n'est pas nouvelle en informatique. On peut citer comme exemple : l'envoi d'un fichier postscript vers une imprimante, d'une *applet* vers un navigateur internet (*Netscape*, *Microsoft Explorer* ou *HotJava*), l'envoi d'agent sur un réseau.

L'inconvénient de cette méthode réside dans un volume de données transférées beaucoup plus important : les données (qui peuvent correspondre aux paramétres de la délégation par requêtes) mais en plus le code de la tâche.

L'avantage est bien sur de fournir un moyen pour que le client puisse déléguer une tâche «*sur mesure*» vers le serveur de la délégation. Le client demande l'exécution d'une tâche qu'il a défini au moyen d'un langage reconnu par le serveur et, non pas, l'exécution de code proposée par le serveur comme dans le cas précédent.

2. Script : suite de caractères décrivant une tâche. En général, l'exécution d'un script se déroule par interprétation

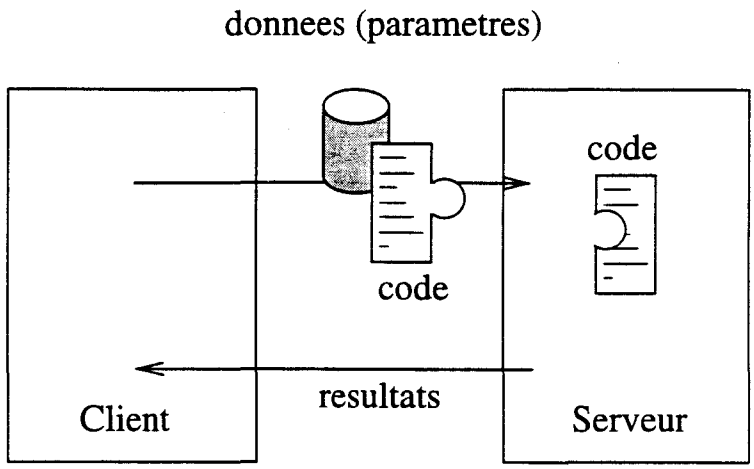


FIG. 2.4 - *Délégation par envoi de programme*

2.2.2 La délégation de tâches pour l'informatique mobile

La délégation de tâche, contrairement à l'envoi d'une requête, permet d'envoyer un programme personnalisé vers une autre station qui dispose de plus de ressources. De plus, comme l'exécution de tâches est asynchrone, un traitement pour le mobile peut être effectué alors que le mobile est éteint, ce qui permet la création de nouvelles applications personnalisées à l'utilisateur étant active sur le réseau sans que l'utilisateur soit connecté.

2.2.2.1 Intérêts

La délégation de tâches permet l'accès à des ressources extérieures. Par conséquent, un terminal mobile bénéficie, de cette façon, de ressources et de services en plus grande quantité et de meilleure qualité sur une station fixe [MDC93].

L'autonomie en énergie étant limitée, déléguer une tâche vers une station extérieure permet aussi à l'unité mobile de basculer en mode éteint ou en veille afin d'économiser l'énergie des batteries dont elle dispose. Dans ce cas, une tâche peut être effectuée vers une station fixe et être active alors que le mobile, lui, ne l'est pas.

Enfin, déléguer vers une station fixe une tâche permet d'avoir des outils de communication plus performants. En effet, la station fixe peut avoir accès à divers réseaux par des passerelles, bénéficier de meilleurs débits, recevoir des volumes d'informations très important.

2.2.2.2 Délégation vers l'agent de représentation

Le principal intérêt de la délégation de tâches du terminal de l'utilisateur vers une autre entité est de disposer de plus de ressources. L'agent de représentation est de toute évidence l'*accepteur de tâches* tout désigné. En effet, il est, d'une part, associé à l'utilisateur et peut donc être considéré comme une extension (une représentation)

de l'utilisateur sur le réseau. D'autre part, l'agent est au service de l'utilisateur, son rôle est de lui faciliter le travail.

L'envoi de tâches vers l'agent de représentation associé apporte plusieurs avantages à l'utilisateur et en particulier s'il est mobile.

Programmation de tâches La programmation de tâches porte tout son intérêt dans le cas où l'utilisateur est mobile. Dans cette situation, l'utilisateur est souvent déconnecté et n'est donc ni joignable ni capable de joindre une personne. En envoyant du code vers l'agent, il est possible de programmer à l'avance le travail à effectuer, donc de créer une description de la tâche qui sera envoyée vers l'agent, exécutée au moment opportun puis l'utilisateur pourra récupérer les résultats à la connexion suivant la fin de l'exécution de la tâche. D'ailleurs, afin d'éviter l'utilisation d'une liaison sans fil, les résultats peuvent n'être envoyés que lorsque l'utilisateur disposera d'une liaison cablée. De plus pour des raisons économiques, il est parfois préférable que les résultats soient envoyés pendant les créneaux horaires les moins chers ou qu'une seule partie des résultats soit transmis, c'est-à-dire les données les plus pertinentes.

Abonnement Une autre utilisation de la délégation de tâches est l'application de type *abonnement*. En effet, l'abonnement se particularise par le fait que les communications ont principalement lieu dans un sens: du fournisseur de documents vers le client qui, ici, est l'utilisateur mobile. Dans la plupart des cas, quand un utilisateur veut obtenir des données provenant régulièrement d'un serveur, il doit envoyer, à chaque fois, une requête et ces données lui sont retournées. Or une requête implique à l'utilisateur d'émettre sur une liaison (comme une liaison radio) qui peut être coûteuse en énergie. C'est pourquoi, il est possible de créer un programme qui ira chercher les informations à partir de l'agent de représentation (donc du réseau cablé) et les transférera automatiquement (sans requête) vers l'utilisateur lorsqu'il sera connecté.

Exemple 2.1 *Une tâche peut envoyer régulièrement les mises à jour des valeurs des différentes monnaies utilisées par un commerçant.*

Cette utilisation de la délégation de tâches permet l'envoi de messages sans requête. La technologie *Push* [L97] propose une utilisation assez semblable. Elle permet, en effet, d'envoyer généralement à plusieurs utilisateurs des données, en particulier des mises-à-jour. Les utilisateurs ne sont pas obligées de constamment émettre des requêtes pour vérifier si de nouvelles données sont disponibles, celles-ci sont automatiquement envoyées. La délégation de tâches offre une implantation de cette technologie destinée à un utilisateur. De plus, au lieu que la programmation de l'envoi automatique soit faite du côté serveur, elle est faite du coté de l'utilisateur client, donc mieux adaptée à ses besoins.

Adaptation du résultat retourné L'envoi de tâches sur le réseau permet aussi de déléguer un pré-traitement en vue d'obtenir des données répondant à des exigences du client. Cette forme de délégation consiste à envoyer une requête vers

l'accepteur de tâches accompagnée de code permettant de faire ce traitement sur les données avant de les retourner. Dans un premier temps, la tâche obtient un résultat à retourner en contactant divers serveurs qui fournissent des données «brutes» qui ne sont pas toujours adaptées aux systèmes pour mobiles. Dans un second temps, la tâche adapte ces données grâce à des mécanismes de filtrage, de compression, de chiffrement ou bien d'autres de façon à réduire la taille des données à transférer sur la liaison terminal - extérieur ou garantir la confidentialité de celles-ci. Ceci peut éviter, par exemple, d'envoyer des images si l'écran est celui d'un téléphone cellulaire qui n'affiche que des caractères.

Cette tâche permet, comme dans le cas d'une requête sur un serveur utilisant le protocole http, de décrire les données que le terminal peut recevoir. Dans une requête http, le client peut utiliser les commandes `ACCEPT`, `ACCEPT-ENCODING` et `ACCEPT-LANGUAGE` qui permettent respectivement de décrire quels types de données (types d'images, de sons, de documents), quels types de codage (compression) et quelles langues l'utilisateur est capable de recevoir ou préfère [Http97]. Dans le cas d'un envoi de programme, le programme doit avoir connaissance de ces mêmes paramètres.

2.2.2.3 Le site accepteur

Rôle du site accepteur Le rôle d'un site accepteur est de fournir des ressources et des services nécessaires à l'accueil d'agents de représentation. En plus de la puissance de calcul, de la mémoire et des moyens de communication, des services tels que la compression de données, le chiffrement de données, l'utilisation de protocole de communication (ftp[3], telnet, http,... [C95c]),... Par conséquent, des serveurs seront disponibles à l'intérieur même du site accepteur et pourront recevoir des requêtes provenant des agents de représentation locaux au site (voir Figure 2.5).

Langage de tâches Des tâches sont envoyées du terminal de l'utilisateur vers son agent de représentation présent sur un site accepteur. Le programme envoyé vers le site accepteur peut prendre deux formes : compilée (en code natif) ou interprétée (sous forme de script).

Tâche sous forme de code natif La forme compilée a l'avantage d'être la plus efficace au niveau de la rapidité d'exécution. La tâche exécutée est écrite de façon à être directement utilisable par le processeur et adaptée à celui-ci.

Cependant, le fait d'être aussi adaptée au processeur interdit l'utilisation de cette tâche sur d'autres processeurs dans des conditions optimales, un émulateur ou convertisseur est nécessaire. Dans ce cas, cela signifie que tous les sites accepteurs devront être normalisés et avoir le même processeur et le même système d'exploitation, ce qui réduit énormément la souplesse de création d'un nouveau site accepteur. De plus, le code est directement exécuté par le processeur. Le site n'a donc aucun moyen de se prémunir contre toute attaque d'une tâche envoyée par une personne malveillante (cheval de Troie), de tout mauvais fonctionnement du à une erreur de programmation de la tâche. Le site ne peut pas non plus veiller aux autorisations

3. FTP : File Transfer Protocol

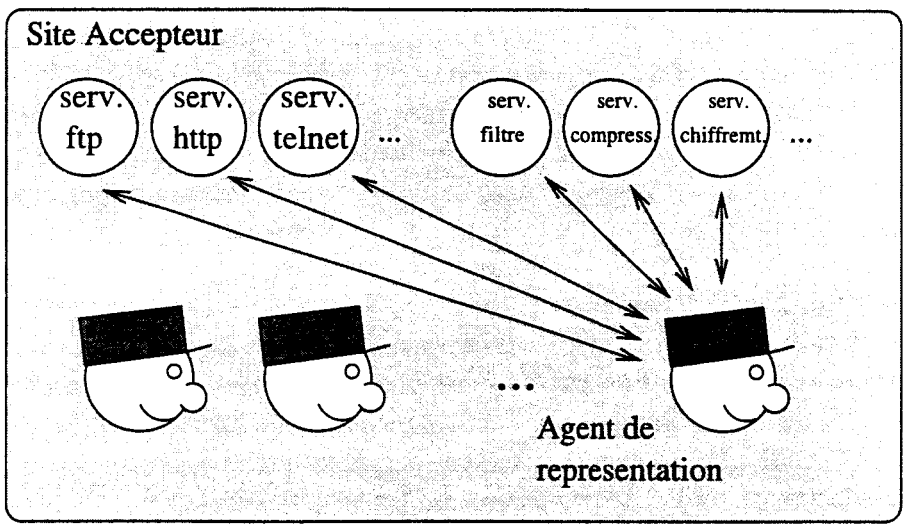


FIG. 2.5 - *Description d'un site accepteur*

d'accès à l'espace mémoire et, donc, garantir la confidentialité des données entre les agents. Une tâche peut, par exemple, lire ou modifier les données d'un autre agent sauf dans le cas d'un système d'exploitation limité qui offre peu d'accès aux ressources mais, en revanche, le système est moins souple.

Par conséquent, utiliser des tâches directement en code natif oblige, pour des raisons de performance, à utiliser le processeur pour lequel ce code a été créé. La portabilité de ce code vers d'autres architectectures nécessite la présence d'un émulateur qui réduit sensiblement l'intérêt de l'utilisation de code natif. De plus, pour garantir un niveau de sécurité important, le système d'exploitation doit être en charge de la vérification des droits sur les commandes exécutées.

Tâche sous forme interprétée Envoyer un script vers l'agent de représentation consiste à envoyer une description de programme, d'algorithme sous format texte. Ce script sera interprété par un *interpréteur* associé à l'agent de représentation [CT96c].

Les avantages sont doubles. D'une part, l'interpréteur se chargeant de l'exécution du script peut être écrit pour plusieurs architectures matérielles et plusieurs systèmes d'exploitation, ce qui confère au script des qualités de portabilité contrairement au code natif. L'exécution du script peut, d'autre part, facilement être contrôlée par l'interpréteur. Un accès à la mémoire doit, par exemple, être autorisé en lecture ou en écriture pour être effectué. Chaque instruction n'est exécutée que si elle est autorisée.

L'inconvénient de l'utilisation de script pour décrire une tâche est d'être plus lent que l'utilisation de code natif. L'interpréteur doit non seulement vérifier la validité de chaque instruction, mais aussi la traduire en code adapté au matériel utilisé. Cependant, une solution comme l'utilisation de langage pré-compilé, est un bon compromis. La tâche est écrite en langage de haut niveau, traduite en langage de

plus bas niveau (plus «proche» du matériel) et envoyée sous cette nouvelle forme vers une machine virtuelle qui l'exécute beaucoup plus rapidement. La machine virtuelle est conçue pour la machine réelle utilisée. Cette solution a été choisie, en autre, pour Java (voir partie 1.3.3.2) et Telescript (voir partie 1.3.3.1).

2.2.2.4 Conception de tâches

Conception «manuelle» des scripts La conception de scripts vers l'agent est à la charge de l'utilisateur propriétaire de l'agent. La première méthode pour l'utilisateur de créer un script est d'écrire manuellement le script. Cette méthode est tout à fait concevable mais a le grand inconvénient de ne s'adresser qu'à des informaticiens et, par conséquent, exclut la plupart des utilisateurs de matériel informatique non spécialistes qui sont maintenant largement majoritaires.

Les interfaces avec générateur de scripts L'utilisateur peut disposer d'outils lui apportant une aide dans l'écriture de ces scripts particulièrement pour des personnes qui n'ont pas de formation en informatique mais sont juste des utilisateurs. Des outils facilitant l'écriture de scripts ou de programmes existent aujourd'hui comme les éditeurs de pages HTML[4] [Html97], les traitements de textes générant du postscript [Postscript97], les ateliers de génie logiciel permettant la création de souches et de squelettes entre autres pour RPC (*rpcgen*) ou pour des objets écrits en Java *rmic* [Java97b] qui permettent de décrire facilement l'interface des services entre deux processus.

L'utilisation d'assistants sous forme d'interfaces graphiques ou en ligne (questionnaire en direct), par contre, est beaucoup plus abordable. Cette méthode permet à un utilisateur de créer son script en répondant à une série de questions et, en fonction des réponses, un script sera généré. Par exemple, pour un script de type abonnement à un journal, un menu contenant une liste de rubriques, des thèmes peuvent être proposés à l'utilisateur qui sélectionnera ses préférences. De plus, il pourra paramétrer la fréquence de recherche des informations désirées (quotidiennement à une heure précise, le lundi seulement,...). Avec ces informations, un script pourra être généré à partir d'un modèle «vierge». En fait, il s'agit ici pour l'utilisateur de remplir un formulaire. Le générateur ainsi que le script «vierge» peuvent être importés sur le terminal de l'utilisateur à partir du réseau ou tout simplement d'une disquette.

Conception par une application Une application peut prendre en charge totalement ou partiellement la création et l'envoi d'une tâche. Cette tâche est ainsi générée afin d'obtenir des avantages liés à la délégation de tâches en informatique mobile tels que ceux énumérés dans la partie 2.2.2. L'application recevra ensuite des résultats qu'elle pourra utiliser. Cette délégation est la plus accessible pour tout le monde. Elle a l'avantage d'être transparente pour l'utilisateur qui n'a pas à être conscient des problèmes liés à l'informatique mobile.

Exemple 2.2 *Une application* `PrévisionsMétéo` *se trouvant sur un ordinateur mobile peut envoyer une tâche qui va chaque jour à heure précise demander les*

4. HTML : HyperText Markup Language

cartes et le bulletin des prévisions météorologiques auprès d'un serveur et va retourner à l'application ces données avec des cartes de format réduit et une extraction locale du bulletin (seulement la ville de résidence par exemple).

2.2.3 Besoins pour la délégation de tâches

Les besoins en terme de délégation de tâches sont doubles. D'une part, une architecture doit être définie pour accueillir ces tâches et, d'autre part, un langage permettant l'écriture de ces tâches doit être choisi.

2.2.3.1 Une architecture pour la délégation de tâches

L'architecture de l'agent de représentation doit fournir un système permettant d'accueillir des tâches provenant de l'utilisateur. C'est-à-dire qu'il doit prévoir plusieurs composants :

1. un *chargeur de tâches* qui est appelé par l'utilisateur ou une de ses applications afin d'importer dans l'agent de représentation la tâche et la transmettre à un interpréteur

2. un *accès à des services* qui permet aux tâches d'accéder à des services de communication, de compression, chiffrement, filtrage,... qui peuvent être commun à toutes les tâches

3. une *file d'attente de sortie* qui permet de retourner un résultat même quand l'utilisateur n'est pas connecté. Ce composant gère ainsi le retour des résultats des tâches en fonction des connexions de l'utilisateur

4. des moyens de double identification/autentification entre l'utilisateur et l'agent de représentation et une confidentialité des données circulant entre-eux

2.2.3.2 Un langage d'écriture de tâches

Ce langage doit avoir en fait quasiment les mêmes qualités décrites dans la partie 1.3.2 concernant les langages d'écriture d'agents mobiles tels que :

1. la portabilité : indépendance vis-à-vis du matériel

2. la sécurité : notamment pour protéger l'agent de représentation et le site accepteur en cas de mauvaise exécution volontaire ou non d'une tâche

3. les facilités de communications : par l'accès à différents composants logiciels prédéfinis.

Les deux premiers points sont deux arguments en faveur de l'utilisation d'un langage interprété pour décrire des tâches.

2.2.3.3 Conclusion

La délégation de tâches permet aux applications d'un utilisateur de déléguer une partie de leur exécution vers des machines dotées de performances supérieures et d'une connexion au réseau continue. Les tâches envoyées peuvent ainsi disposer de meilleures ressources et être actives même lorsque l'utilisateur s'est déconnecté. La délégation de tâches permet à l'utilisateur ou à ses applications de programmer une exécution, de recevoir des données régulièrement, d'obtenir un résultat adapté au terminal ou aux désirs de l'utilisateur.

L'agent de représentation est un support adapté à la réception de tâches. Il est, en effet, dédié à l'utilisateur. Le système de l'agent de représentation est en charge non seulement de fournir aux tâches un environnement d'exécution mais aussi de l'acheminement de ces données vers l'utilisateur. Une tâche ne doit pas, par exemple, se préoccuper de l'état courant du terminal utilisé, c'est-à-dire connecté ou non, ses caractéristiques matérielles. Ainsi les tâches envoient leurs résultats au système de l'agent de représentation qui les retourne à l'utilisateur en fonction des caractérisques matérielles du terminal et des préférences de l'utilisateur lorsque celui-ci est connecté.

2.3 Gestion des données destinées à l'utilisateur

Dans la partie précédente, nous avons développé l'envoi de tâches de l'utilisateur vers l'agent de représentation. Nous allons maintenant nous intéresser à la réception des résultats de tâches et décrire les besoins en composants et en services à la fois au niveau du terminal de l'utilisateur qu'au niveau de son agent de représentation. Ces composants doivent permettre la communication entre, d'une part, le terminal de l'utilisateur et, d'autre part, l'agent de représentation et les tâches qui lui ont été envoyées (voir section précédente). De plus, ils doivent permettre la gestion des données provenant de l'extérieur et destinées à l'utilisateur et, éventuellement un traitement sur ces données de façon à ce qu'elles soient adaptées au matériel dont dispose l'utilisateur et à ses préférences.

2.3.1 Composants de l'agent de représentation

L'agent de représentation est un intermédiaire dans les communications entre le terminal de l'utilisateur et le monde extérieur. Il appelle et reçoit donc des messages et des données provenant de l'extérieur et destinées à l'utilisateur. Ces messages seront ensuite retransmis ou mis en attente à l'intérieur de l'agent de représentation dans le cas où l'utilisateur ne serait pas accessible.

Pour être plus précis, l'utilisateur, via une de ses applications, envoie une tâche vers son agent de représentation. Cette tâche émet des requêtes, échange des messages avec des serveurs sur le réseau. La tâche reçoit ainsi des données qu'elle utilise lors de son exécution afin de produire des résultats destinés à l'utilisateur. Ces résultats, qui sont de nouvelles données, sont envoyés vers l'utilisateur par l'intermédiaire du système de l'agent de représentation. Ce système gère les communications avec le terminal de l'utilisateur. Lorsque l'utilisateur se reconnecte, les données sont envoyées vers le terminal de l'utilisateur où elles sont stockées et rangées en vue d'une

utilisation par l'application destinataire. La figure 2.6 représente le flux de données provenant du monde extérieur vers l'utilisateur et ses applications.

Exemple 2.3 *Pour une application de courrier électronique, des messages arrivent vers la tâche* `Courrier` *recevant le courrier. Cette tâche peut effectuer quelques traitements, par exemple, convertir un message* audio *en texte (voir les outils de dictée vocale avec vocabulaire limité [Dragon97] jusqu'au système à vocabulaire important [CML97]) regrouper les mails en fonction de l'émetteur et donner une présentation sous forme de pages HTML aux messages. Ensuite ces messages sont envoyés à ce que nous appelerons pour le moment* système *de l'agent de représentation. Le système, en fonction des recommandations de la tâche* `Courrier` *dépendant du système, va effectuer des traitements sur les données comme une compression, un chiffrement, un filtrage. Toujours en fonction des recommandations de la tâche* `Courrier`*, certains messages seront envoyés automatiquement à l'utilisateur, d'autres seulement après une requête venant de l'utilisateur. Les messages enfin arrivent sur le terminal de l'utilisateur et sont mis à disposition de la tâche* `Courrier`*.*

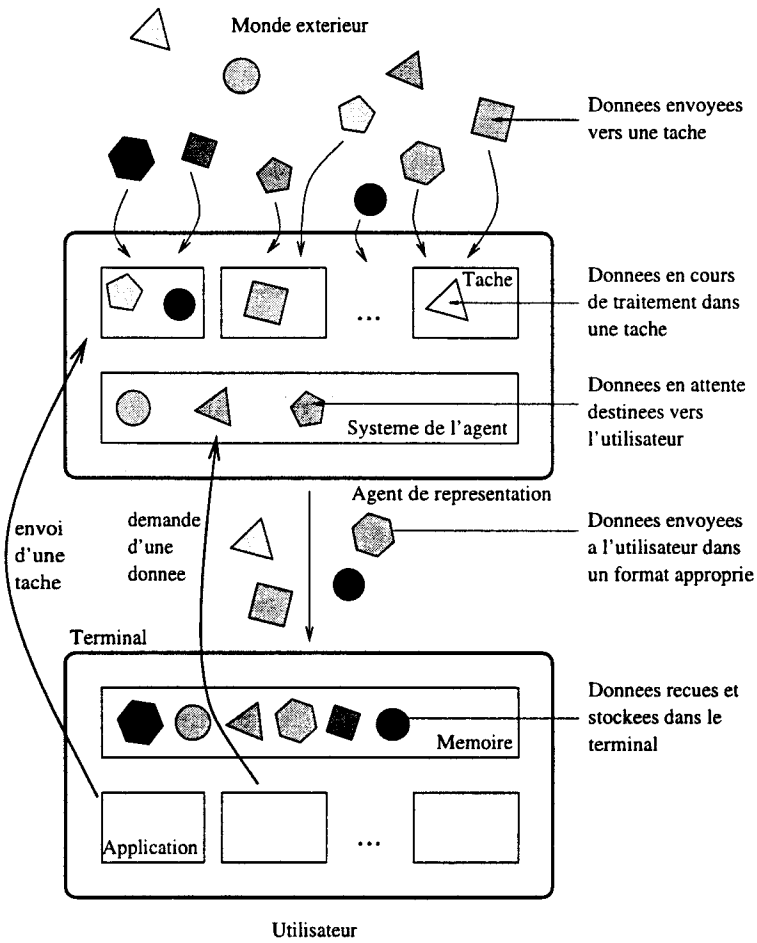


FIG. 2.6 - *Circulation des données*

2.3.2 Envoi de données vers l'utilisateur

Les données provenant d'une tâche et destinées à l'utilisateur ne sont pas obligatoirement envoyées telles quelles mais peuvent subir un ou plusieurs traitements auparavant : compression, filtrage, chiffrage,... Cependant ces traitements n'ont pas à être effectués sur toutes les données comme cela sera développé.

2.3.2.1 Modes d'envoi de données

Nous allons énumérer des traitements indispensables sur les données envoyées par l'agent de représentation vers l'utilisateur. Cependant d'autres traitements intéressants peuvent être envisagés.

Envoi *automatique* ou *à la demande* Etant donné l'asymétrie des coûts de communication à partir d'un mobile [FZ94], recevoir automatiquement une donnée, c'est-à-dire sans requête, nécessite sensiblement moins d'énergie que la recevoir sur demande [IV94]. Les résultats des tâches destinés à l'utilisateur pourraient donc être envoyés :

1. *Automatiquement* si l'utilisateur est connecté et dispose d'un débit suffisamment important
2. *A la demande* si l'utilisateur demande explicitement par une requête l'obtention d'un résultat, ce qui permet d'envoyer ces résultats à un moment précis choisi par l'utilisateur

Soit une tâche retourne un résultat destiné à l'utilisateur en mode automatique, soit le résultat arrive peu de temps après sur le terminal de l'utilisateur, soit il est mis en attente jusqu'à une nouvelle connexion de l'utilisateur. Par contre, un résultat de tâche envoyé à la demande est stocké à l'intérieur de l'agent de représentation, puis, après une requête de l'utilisateur, sera retourné. Néanmoins, lorsqu'un nouveau résultat est créé par une tâche, un message doit être envoyé à l'utilisateur de façon à ce qu'il sache qu'un nouveau résultat sera disponible sur l'agent de représentation et comment l'obtenir.

Par conséquent un résultat envoyé automatiquement demande moins d'énergie qu'à la demande. L'énergie dépensée pour émettre la requête est le surplus de consommation nécessaire dans le cas d'un envoi à la demande. Cependant, pour certaines données demandant des ressources particulières, par exemple, image couleur ou son, il est inutile de les envoyer automatiquement alors que l'utilisateur ne pourra pas les utiliser. De même, une donnée importante en taille n'étant pas nécessairement urgente, n'a pas à être envoyée automatiquement si le débit de la liaison utilisée est très faible ou coûteuse. L'utilisateur peut se connecter plus tard sur une liaison haut débit pour demander cette donnée ou dans un créneau horaire permettant un transfert meilleur marché [BCK93].

Exemple 2.4 *Suite à l'exemple 2.3, la tâche* `Courrier` *peut juger en fonction de l'origine que des messages sont prioritaires, d'autres moins urgents. Dans ce cas, lorsqu'elle envoie au système de l'agent de représentation un courrier destiné à l'utilisateur, la tâche précise sur cet envoie doit être automatique ou à la demande.*

Remarque 2.2 *Ce type d'envoi* automatique *ou* à la demande *peut être considéré comme un filtrage des données.*

Permanence des données dans l'agent de représentation Lorsqu'une donnée n'a pas encore été envoyée à l'utilisateur, elle est stockée à l'intérieur de l'agent de représentation avant l'émission. Après avoir reçu une donnée de l'agent de représentation, l'utilisateur peut préférer soit qu'elle reste dans l'agent de représentation, soit qu'elle en soit retirée. Les raisons peuvent être les suivantes :

- l'utilisateur ne peut ou ne veut pas sauver cette donnée sur son disque par manque de place
- l'utilisateur travaille à partir de plusieurs terminaux

L'agent de représentation joue le rôle de mémoire commune pour tous les terminaux sur lesquels l'utilisateur se connecte.

Un résultat provenant d'une tâche peut être :

- soit *permanent* dans l'agent de représentation, c'est-à-dire qu'après avoir été envoyé vers l'utilisateur le résultat reste stocké à l'intérieur de l'agent de représentation. Dans ce cas, l'utilisateur enverra un message de destruction pour effacer ce résultat de l'agent de représentation.
- soit *non permanent*, le résultat est effacé automatiquement après émission vers l'utilisateur. Ce type de résultat est plus économique en envoi de message et est tout à fait approprié pour des données informatives qui ne sont consultées qu'une seule fois.

Exemple 2.5 *Dans le cas du courier électronique, certains messages, voire tous les messages, jugés important par la tâche peuvent être envoyés vers le système de l'agent de représentation et rester disponibles pour l'utilisateur jusqu'à ce qu'il décide explicitement de les détruire. D'autres messages, provenant de publicitaires par exemple, peuvent être détruits automatiquement après l'envoi.*

Compression/Chiffrement La compression et le chiffrement sont des opérations qui permettent de coder des données grâce à un alogrithme. Ces données pourront être ensuite décodées en utilisant de nouveau un algorithme.

Une donnée destinée à l'utilisateur peut être envoyée sous forme compressée ou non afin de réduire la taille des données transférées. L'agorithme de compression doit être choisi en fonction des algorithmes de décompression disponibles sur l'application de l'utilisateur. De plus, le choix de l'agorithme peut dépendre d'un compromis entre taille des données et les ressources requises sur l'équipement mobile pour la décompression. C'est pourquoi, une tâche doit pouvoir délivrer un résultat à l'utilisateur ou à une de ses applications sous une forme compressée.

Remarque 2.3 *Un filtrage sur une donnée peut être considéré comme une compression particulière. Les informations jugées superflues sont effacées. C'est une compression avec perte.*

Le chiffrement est à considérer de la même manière que la compression. Une donnée envoyée à l'utilisateur peut être dans certains cas confidentielle. Comme une liaison sans fil peut être «écoutée», c'est-à-dire qu'une personne peut lire les données circulant sur cette liaison, un chiffrement reste la seule alternative pour garantir la confidentialité. Par conséquent, une donnée doit pouvoir être chiffrée selon un algorithme parmi ceux disponibles dans l'agent de représentation.

2.3.2.2 Modes d'envoi conditionnels de données

Pouvoir gérer les différents types de données tels que *automatique* ou *à la demande*, *permanent* ou *non permanent*, *compressé* ou *non compressé*, *chiffré* ou *en clair* à partir de conditions constituerait un apport très important pour un utilisateur mobile utilisant différents terminaux.

Exemple 2.6 *l'utilisateur dispose d'un téléphone mobile comme terminal. Le débit est faible, la mémoire réduite, la liaison peu sûre. A partir de ces paramètres, certaines données doivent être plutôt envoyées* à la demande compressées *et* chiffrées *avec un algorithme qui demande peu de ressources lors du déchiffrement. De plus, ces données pourraient rester* permanentes *dans l'agent de représentation. L'économie de la liaison sans fil sera importante et, comme les données reçues par l'utilisateur resteront dans l'agent de représentation, ces données n'auront pas à être sauvegarder dans le terminal.*

Exemple 2.7 *L'utilisateur est maintenant connecté sur une station de travail à haut débit et disposant de ressources importantes. La* compression *n'est plus intéressante, les données peuvent être envoyées* automatiquement. *Si la station peut servir d'archive pour certaines données, celles-ci peuvent donc être* non permanentes *dans l'agent de représentation. Enfin, la liaison peut être considérée comme plus sûre et dans ce cas le chiffrement n'est pas indispensable.*

Les exemples 2.6 et 2.7 montrent ici qu'une tâche ne peut pas déterminer la façon d'envoyer un résultat à l'utilisateur. Cela dépend de paramètres autres que les résultats eux-mêmes. Suivant la configuration, l'envoi d'un résultat doit pouvoir varier.

La démarche du *S-Shell* [T95] est très similaire. Il permet de décrire sous forme d'un court script la politique de sécurité à appliquer sur une donnée. Dans notre cas, une description permet de définir comment va être traité un objet-résultat ou ses composants.

2.3.3 Répartition des rôles

L'agent de représentation accueille des tâches qui sont exécutées et retourne un ou des résultats. Les tâches vont faire appel à des services proposés par l'agent de représentation. C'est pourquoi, il est nécessaire de trouver une répartition du travail entre, d'un coté, les tâches et, de l'autre, l'agent de représentation.

2.3.3.1 Rôle des tâches

Une tâche représente une application de l'utilisateur. Elle effectue donc le traitement qui a été décrit avant son émission vers l'agent de représentation et retourne des résultats à l'utilisateur.

Une tâche n'a pas à effectuer des traitements en fonction des propriétés du matériel de l'utilisateur, de ses déconnexions [SNK94]. Elle ne fera que communiquer les résultats au système de l'agent de représentation avec des recommandations selon le profil de la situation, c'est-à-dire un mode d'envoi associé correspondant à ceux décrits dans la partie 2.3.2.

2.3.3.2 Rôles de l'agent de représentation

L'agent de représensation a principalement le rôle de fournir un environnement, des services communs à toutes les tâches.

Accueil des tâches L'agent de représentation doit fournir un environnement d'exécution à de nouvelles tâches. Une application de l'utilisateur envoie une tâche vers l'agent de représentation qui va lui proposer un ensemble de services permettant aux tâches de s'exécuter, de communiquer avec l'extérieur, et de renvoyer des résultats.

Gestion des données destinées à l'utilisateur L'agent de représentation est, d'une part, responsable de la fourniture de données destinées à être envoyées le plus tôt possible à l'utilisateur, c'est-à-dire lorsqu'il est connecté. Pour cela, l'agent de représentation doit disposer d'une file d'attente [Iso93] de sortie destinée à l'utilisateur et ses applications présentes sur son terminal.

L'agent de représentation est, d'autre part, en charge du stockage des données destinées à l'utilisateur lorsqu'il en fait la demande. Dans ce cas, il envoie dans la file d'attente de sortie les données demandées.

Traitement sur les résultats L'agent de représentation, à partir d'un résultat et d'un mode d'envoi provenant tous les deux d'une tâche, est en charge d'effectuer le traitement nécessaire sur ce résultat en fonction du mode d'envoi (compression, chiffrement, automatique / à la demande, permanence).

2.3.3.3 Rôles du terminal dans la gestion des données

Un des rôles du terminal est de mettre à disposition les données reçues de l'agent de représentation aux différentes applications de l'utilisateur. Pour cela, le terminal doit pouvoir :

1. fournir aux applications la liste des données déjà reçues de l'agent de représentation *(la liste des courriers disponibles sur le terminal dans le cas de l'exemple 2.3)*

2. fournir aux applications les données déjà reçues de l'agent de représentation *(l'utilisateur peut obtenir sur requête un courrier présent sur le terminal)*

3. fournir aux applications la liste des données leur étant destinées présentes dans l'agent de représentation mais pas dans le terminal *(la liste des courriers disponibles sur l'agent de représentation)*

4. fournir aux applications une donnée présente dans l'agent de représentation mais non présente dans le terminal *(l'utilisateur peut obtenir sur requête un courrier présent sur l'agent de représentation)*

De plus, le terminal doit, lors de la réception d'une donnée, remettre celle-ci dans un format utilisable. Par exemple, il décompressera une donnée compressée. Cette opération doit être effectuée avant de la transmettre à l'application destinataire.

2.4 Conclusion

L'utilisation d'un agent de représentation, comme il a été défini, nécessite une prise en compte des problèmes liés à la fois aux communications entre l'utilisateur et le monde extérieur, aux tâches envoyées vers l'agent de représentation et à la gestion des résultats destinés à l'utilisateur. Chaque composant doit avoir un rôle précis. Les tâches sont acceptées par l'agent de représentation, produisent des résultats, le système de l'agent de représentation se charge de les envoyer sous une forme appropriée, le terminal les met à disposition des applications de l'utilisateur.

Le travail propose la mise en oeuvre de ces agents de représentation en définissant un système permettant leur utilisation sur un réseau, la délégation de tâches, le retour de résultats adaptés au matériel et aux préférences de l'utilisateur. Le système doit cependant garantir un niveau de sécurité important et la possibilité de personnaliser des terminaux.

Chapitre 3

Système basé sur les agents de représentation

Introduction

Ce chapitre présente le système proposé sur deux niveaux. Le premier décrit le réseau permettant l'utilisation d'agents de représentation en se concentrant sur la localisation de ces agents et leur migration. Le second niveau présente la délégation de tâches vers l'agent de représentation et les communications effectuées entre une tâche d'un agent de représentation et des serveurs.

3.1 Réseau pour agents de représentation

Dans cette section, nous présentons une architecture globale permettant l'utilisation d'agents de représentation. Le rôle de l'agent de représentation et du site accepteur entraine des problèmes de localisation. L'agent de représentation doit être toujours accessible en qualité de représentant de l'utilisateur. Pour cela, il est nécessaire de connaître sa position. Le principe de migration de l'agent de représentation implique qu'il doit trouver un site accepteur répondant à ses exigences.

3.1.1 Description des différents composants

Cette partie décrit principalement le rôle de l'agent de représentation et celui du site accepteur.

3.1.1.1 L'agent de représentation

L'agent de représentation est une représentation active mobile et permanente d'un utilisateur mobile. Cet agent de représentation est chargé de recevoir des tâches provenant d'applications de l'utilisateur et de retransmettre ou archiver les résultats destinés à l'utilisateur. Ces mécanismes seront respectivement détaillés dans les parties 3.2 et 4.1.

L'utilisateur reçoit un identifiant qui permet de le référencer. Cet identifiant est ainsi associé à l'agent de représentation de l'utilisateur.

Dans la majorité des cas, un utilisateur mobile peut déterminer une zone géographique dans laquelle il est le plus souvent : sa zone de résidence ou de travail. Une zone géographique est une zone administrative dans laquelle se trouve un ensemble de sites accepteurs.

L'utilisateur désirant obtenir un service basé sur les agents de représentation demande la création d'une RIF, Représentation Initiale Fixe, sur son site personnel. Ce site personnel peut être une station constamment connectée appartenant, par exemple, soit à l'entreprise de l'utilisateur, soit à un prestataire de services proposant l'accueil d'agents de représentation.

La RIF de l'utilisateur joue le rôle d'un agent de représentation lorsque l'utilisateur est dans sa zone géographique. Lorsqu'il quitte cette zone, la RIF va générer un agent de représentation qui va «suivre» l'utilisateur.

L'utilisateur dispose donc d'une RIF sur son site personnel et d'un agent de représentation [CD97]. Lorsque l'utilisateur se trouve dans la zone de son site personnel, la RIF et l'agent de représentation sont confondus. En déplacement, l'utilisateur dispose d'un agent de représentation local à sa position courante.

Lorsque l'utilisateur retourne dans la zone de son site personnel, l'agent de représentation et la RIF mettent en commun leurs données et éventuellement les nouvelles tâches que l'utilisateur aurait envoyées vers l'agent de représentation.

3.1.1.2 Le site accepteur

Le site accepteur accueille des agents de représentation en leur fournissant des ressources et une connexion permanente au réseau. On peut distinguer, pour un usager, deux types de sites accepteurs :

1. Le site personnel accueillant la RIF de l'utilisateur. Ce site peut appartenir à l'utilisateur, à son entreprise, à une collectivité ou un prestataire de services qui fournit ce service sur abonnement.

2. Les sites accepteurs accueillant les agents de représentation émis par la RIF. Ces sites appartiennent à des prestataires de services ou à l'entreprise de l'utilisateur (dans le cas d'un réseau d'entreprise privé).

Lorsqu'un agent de représentation désire se déplacer d'une zone A vers une zone B, celui-ci peut désirer un site qui fournit certains services particuliers ou qui appartient à une entreprise ou à un prestataire de services désigné. Pour cela, il a besoin de consulter un serveur qui lui donne une liste de sites accepteurs répondant à ses exigences. Un ensemble de serveurs basé sur une architecture X.500 [Ccitt88] répond à ces exigences. Ces serveurs de localisation permettent, en effet, de fournir une liste d'éléments répondant à des conditions booléennes. Les entrées d'un service X.500 sur les sites accepteurs auraient des attributs constitués de :

- `Zone` : la zone du site accepteur
- `Organisation` : l'organisation à laquelle appartient le site accepteur
- `Services` : les services proposés
- `Coût` : unité de coût d'utilisation
- ...

Un agent de représentation demande à un de ces serveurs la liste des sites accepteurs de la zone B répondant aux exigences :

- $Zone = B$
- $Organisation = Education$
- $Services = LiaisonTelephonique$
- $Services = Fax$
- ...

A partir de cette liste, l'agent de représentation choisit un site accepteur. Si celui-ci est indisponible, il en choisit un autre.

Un service X.500 peut proposer, dans une liste, des sites inaccessibles, même si un attribut `Accessibilité` est ajouté aux entrées. En effet, à cause de la présence de caches, une mise-à-jour globale de ces caches sur l'attribut `Accessibilité` d'un site peut prendre plusieurs minutes [K92].

3.1.2 La migration

La migration de l'agent de représentation consiste à rapprocher cet agent de l'utilisateur auquel il est associé. Cette migration peut être effectuée soit automatiquement par l'agent de représentation soit à la demande de l'utilisateur. Ce choix de configuration entre les deux stratégies est de la responsabilité de l'utilisateur.

3.1.2.1 Problématique

La migration d'un agent de représentation pose de nombreux problèmes. La quantité de données à faire migrer avec l'agent de représentation peut être très importante, la localisation d'un agent mobile ne doit pas être trop lourde. Enfin la continuité des communications avec un agent de représentation doit être assurée même après une migration, alors que l'agent de représentation n'a plus la même localisation.

Migration partielle de données Nous allons d'abord définir ce que nous appelons *migration partielle* de l'agent de représentation.

Définition 3.1 *La* migration partielle de données *de l'agent de représentation est le fait de faire migrer seulement une partie des données de cet agent de représentation.*

La migration partielle de l'agent de représentation est intéressante pour deux raisons :

1. La migration vers un site accepteur est un processus qui demande le transfert à la fois des tâches présentes dans l'agent, mais aussi celui des résultats destinés à l'utilisateur, qui peuvent être des données de grande taille : images, sons, programmes,...

2. Les sites accepteurs peuvent faire payer l'hébergement d'agents de représentation en fonction de la mémoire utilisée.

L'utilisateur peut demander à ce que la migration ne concerne qu'une partie des résultats de l'agent de représentation. L'autre partie reste sur le site accepteur personnel dans la RIF. La migration partielle permet d'éviter de transférer des données avec l'agent de représentation qui ont peu de chances d'être utilisées lors du déplacement. La tâche de l'agent de représentation est responsable de la désignation des données devant ou non migrer en fonction de critères tels que la taille ou le besoin de l'utilisateur de la donnée. Une donnée peut concerner un usage local au site personnel et donc être inutile dans une autre région, par exemple, des données concernant le trafic routier local au site personnel. Une implantation sera proposée dans la partie 4.1.3.3.

Cependant, si l'utilisateur désire obtenir une donnée restée dans la RIF après une migration, cette demande sera retransmise par l'agent de représentation à la RIF.

Localisation d'un agent de représentation Un agent de représentation créé sur un site accepteur doit être publié pour pouvoir être contacté. Publier un agent de représentation consiste à diffuser le couple composé de l'identifiant de l'utilisateur et de la localisation de son agent de représentation. Les destinataires peuvent être des bases de données, des serveurs d'adresses ou des interlocuteurs potentiels choisis par l'utilisateur ou une de ses applications.

Le problème de la migration réside dans le fait que ces adresses publiées nécessitent une mise-à-jour constante en fonction des déplacements de l'utilisateur. La

RIF peut servir de serveur d'adresse de l'agent de représentation. Pour cela, l'agent de représentation doit, après chaque migration, envoyer sa nouvelle adresse à la RIF. La partie 3.1.3 développe plus en détail la localisation d'un agent de représentation.

Remarque 3.1 *La RIF retourne sa propre adresse dans le cas où elle est confondue avec l'agent de représentation.*

Communications après la migration Après une migration, l'agent de représentation arrive sur un nouveau site accepteur et donc change d'adresse physique. Par conséquent, des interlocuteurs de l'agent de représentation peuvent avoir gardé une adresse qui n'est plus valide.

Il est donc nécessaire qu'un *gestionnaire de messages* soit présent sur chaque site accepteur. Il est chargé de la redistribution des messages vers les agents de représentation destinataire. Lorsque celui-ci n'est pas sur le site, soit le site connait la nouvelle adresse de l'agent, soit il la demande à la RIF (voir la partie 3.1.2.1). après, il retransmet le message vers le gestionnaire de messages du site accepteur ou se trouve l'agent de représentation (voir figure 3.1).

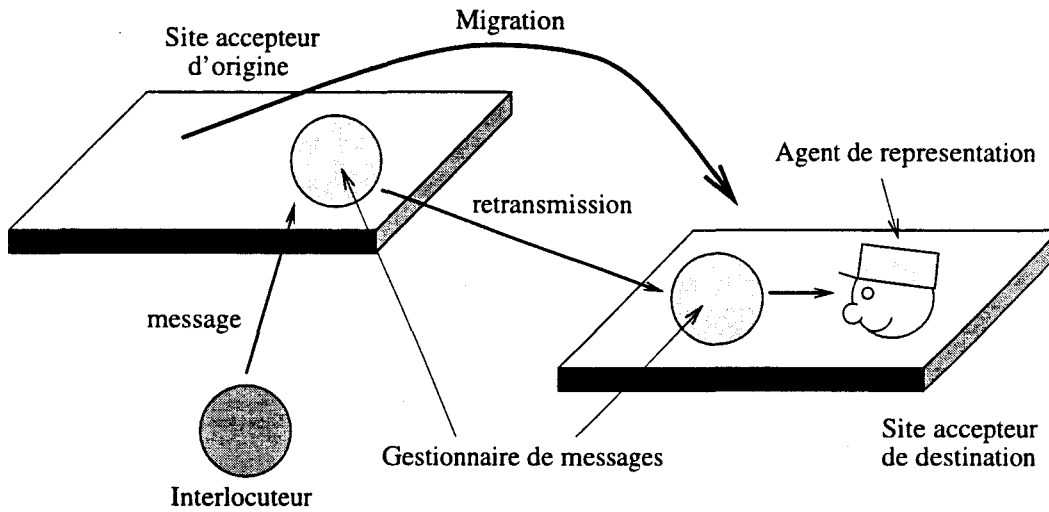


FIG. 3.1 - *Retransmission des messages destinés à un agent de représentation*

3.1.2.2 Différentes situations pour la migration

Cette partie résume les différentes opérations liées à la migration dans trois situations. La première demande la création d'un agent de représention à partir du RIF, la seconde la migration de cet agent d'un site accepteur vers un autre et enfin la troisième le retour vers le site du RIF.

Dans le cas où l'utilisateur quitte sa zone géographique, la RIF crée un agent de représentation contenant les données devant migrer et l'envoie sur un site accepteur dans la nouvelle zone de l'utilisateur. L'adresse d'un site accepteur disponible est obtenue en interrogeant une base X.500 (voir section précédente). La RIF se

charge de fournir sur requête l'adresse courante de l'agent de représentation et de retransmettre tout message lui étant destiné.

Lorsque l'utilisateur passe d'une zone vers une autre, toujours différente de celle de la RIF, l'agent de représentation migre vers un site destinataire de la nouvelle zone de l'utilisateur. L'ancien site accepteur se charge de retransmettre les messages destinés à l'agent de représentation.

Quand l'utilisateur retourne dans la zone où se trouve sa RIF, son agent de représentation va migrer vers le site dans la RIF et toutes les nouvelles données vont être ajoutées à la RIF. La RIF prend la place de l'agent de représentation qui n'a plus lieu d'exister.

3.1.3 La localisation de l'agent de représentation

Contacter un utilisateur revient à contacter son agent de représentation. Cependant celui-ci est mobile. Il faut donc au préalable envoyer une requête à un serveur d'adresses ou à la RIF de l'utilisateur. Cependant contacter une RIF peut entrainer un surcoût en communication car il peut être dans une zone géographique différente de l'utilisateur (problème similaire à la partie 2.1.3.1) alors qu'un serveur d'adresses pourrait être dans la même zone. C'est pourquoi, l'interlocuteur a intérêt à contacter un serveur d'adresses proche, quand cela est possible, plutôt que la RIF de l'utilisateur.

Des modèles utilisant des serveurs d'adresses, appelés serveurs de noms, sont présentés. Ils permettent une utilisation dans un réseau à grande échelle et sont chargés à partir d'un identifiant de retourner l'adresse physique.

3.1.3.1 Les services de nommage actuels

Le DNS Le modèle basé sur les *DNS*[1] [M87] est actuellement le plus représenté, grâce au développement d'Internet. Il est utilisé pour convertir un nom de machine ou de domaine de machines (par exemple `jenlain.lifl.fr`) en adresse IP (`134.206.10.5`) dans un réseau global comme Internet.

Le DNS permet à une machine d'avoir une référence hiérarchique globale lisible pour un humain. Une machine nommée `Jenlain` est placée dans un domaine `lifl` qui est lui-même sous-domaine de `fr`. Le nom complet, c'est-à-dire `jenlain.lifl.fr`, référence une machine unique.

Un nom de sous-domaine ou de machine est attribué par le domaine hiérarchiquement supérieur. Ainsi le sous-domaine `lifl.fr` a attribué le nom `jenlain` à une machine qui le compose (voir figure 3.2).

La conversion d'un nom de domaine ou de machine en adresse IP est effectuée par des *serveurs de noms.* Il en existe plusieurs permettant une conversion sur un sous domaine du réseau. Par exemple, un serveur de noms associé au domaine `fr` est capable de résoudre une adresse se trouvant à l'intérieur de ce domaine (par exemple `jenlain.lifl.fr`), ou connait le serveur de noms du sous domaine (`lifl.fr`) capable de faire cette conversion. Le serveur du domaine `fr` peut cependant résoudre plus de conversions, même en dehors du domaine, s'il est optimisé par des caches.

1. DNS : Domain Name System

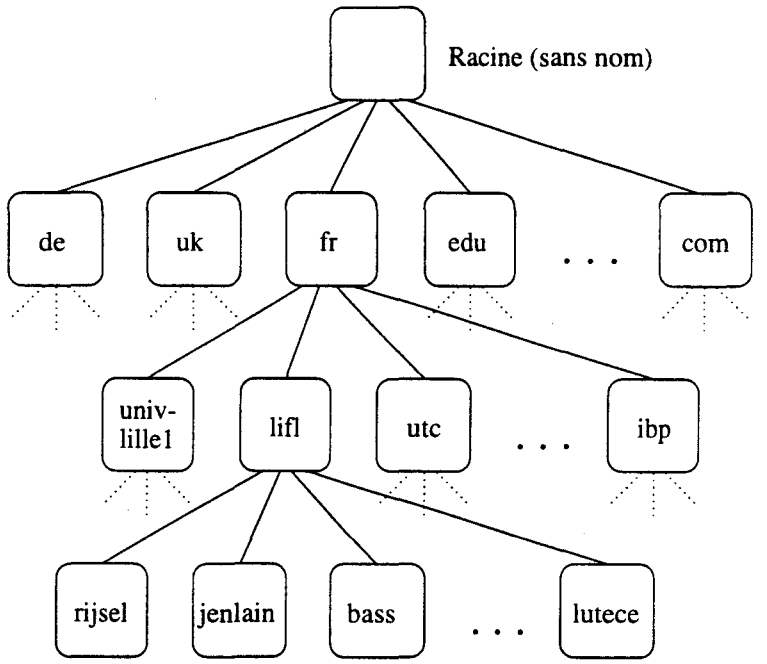


FIG. 3.2 - *Hiérarchie de nommage de domaines sur Internet*

Une demande de résolution du nom `N` est envoyée à un serveur de noms local `S` qui effectue la conversion s'il le peut et retourne le résultat. Dans le cas contraire, deux possibilités existent :

1. Si `N` est dans un sous-domaine, `S` envoie une requête à un serveur de noms de ce sous-domaine de `S`.
2. Si `N` est dans un sur-domaine, `S` envoie une requête à un serveur de noms de ce sur-domaine de `S`.

Exemple 3.1 *La machine* `gamay.rd2p.lifl.fr`*, qui désire connaître l'adresse de la machine* `jeanlain.lifl.fr`*, contacte le serveur de noms du domaine* `rd2p.lifl.fr`*. Ce serveur, n'étant pas capable de résoudre cette conversion, envoie une requête au serveur du sur-domaine* `lifl.fr` *qui lui renverra l'adresse de* `jeanlain.lifl.fr`*.*

Les DNS sont efficaces dans des réseaux comme Internet qui relie des machines, des domaines fixes. Dans le cas d'agents mobiles, la situation est différente et le DNS n'est pas adapté à la résolution de nom de domaines mobiles. La résolution se fait à partir de bases de données rarement modifiées et les mécanismes de caches deviennent difficilement gérables. C'est pourquoi, le modèle du DNS permettant l'obtention d'une adresse n'est pas adapté, sous cette forme, aux agents mobiles.

Le GNS Le *GNS*[2] [L86] est aussi un modèle basé sur des serveurs de noms. Il reprend une structure hiérarchique comme pour les DNS, mais permet de déplacer

2. GNS : Global Name Service

un sous-domaine SD (ou une machine) d'un domaine D1 à un autre D2 en plaçant une redirection dans le domaine D1 vers SD.D2. La résolution n'est pas obligatoirement effectuée en interrogeant un serveur de nom du domaine D1 ou D2, mais peut l'être aussi à partir d'autres serveurs de noms qui utilisent des caches.

Exemple 3.2 *Supposons que le LIFL désire changer de domaine et être non plus un sous-domaine de* `fr` *mais de* **edu**. *Dans ce cas un sous-domaine* `lifl` *est créé dans le domaine* **edu**. *Lorsqu'une demande de résolution du nom* `lifl.fr` *arrive sur un serveur de noms du domaine* `lifl.fr` *celui retourne l'adresse du domaine* `lifl.edu`.

Les caches sont utilisés de manière statique, c'est-à-dire que les changements d'adresse sont considérés comme rares. Or, un agent mobile risque d'avoir une adresse physique qui évolue régulièrement.

3.1.3.2 Solution proposée

Les modèles DNS et GNS sont adaptés pour la résolution de noms dans le cas où, au nom, est toujours associé la même adresse. Peu d'évolutions sont accordées et, dans ce cas, le processus est relativement long. C'est pourquoi, localiser un agent de représentation dans un réseau basé sur IP nécessite quelques changements au niveau des serveurs de noms.

Limites du modèle DNS classique Un utilisateur dispose à la fois d'une RIF et de son agent de représentation. La RIF est fixe et permanente et ne nécessite que l'utilisation de tables statiques de conversion. Par conséquent, la résolution de son nom peut être similaire à une résolution effectuée dans un DNS. Cependant la communication avec l'utilisateur ne passe pas par la RIF mais par l'agent de représentation. Contacter l'utilisateur requiert l'adresse de son agent de représentation qui n'est pas constante et qui ne peut pas être obtenue dans un système DNS classique. En effet, le DNS utilise des caches qui ne pourraient pas contenir d'adresses qui évoluent sans une gestion lourde des mises-à-jour et un flux de messages supplémentaires important. De plus, si ces adresses ne sont pas contenues dans des caches, la recherche d'un agent de représentation serait globale au réseau entier entrainant un temps de recherche et un nombre de messages considérablement important.

La représentation initiale fixe comme serveur d'adresses Beaucoup de solutions proposent pour obtenir l'adresse physique d'un mobile de lui associer deux adresses [IDM91] [BP93] [J93] [WYO93] [TUS94] :

1. une *adresse* courante qui évolue en fonction des déplacements du mobile
2. une *adresse constante* qui permet d'obtenir l'adresse courante du mobile

Dans le cas de l'obtention de l'adresse d'un agent de représentation, le modèle utilisé peut s'inspirer de ces solutions.

Remarque 3.2 *Dans le cas où l'utilisateur se trouve dans sa zone géographique d'origine, la résolution de nom de l'agent de représentation s'effectue classiquement. Comme la RIF et l'agent de représentation sont confondus, l'utilisateur ne*

dispose que d'une adresse pour le contacter, aucune conversion d'adresse constante en adresse courante n'est nécessaire.

L'adresse de la RIF est accessible à partir du système de nommage classique comme le DNS. Par conséquent, elle peut servir de serveur d'adresses pour contacter l'agent de représentation de l'utilisateur qui a une adresse qui évolue. L'adresse de la RIF représente ici l'*adresse virtuelle* de l'utilisateur

Supposons que la RIF de l'utilisateur *Pierre Martin* se trouve dans le domaine `mobile.lifl.fr`. La RIF peut donc être référencée `martin.mobile.lifl.fr` qui représente un nom de l'adresse virtuelle de *Martin*. Pierre Martin est en voyage en Allemagne et son agent de représentation se trouve dans le domaine `visitor.mobile.de`. Son agent de représentation reçoit un identifiant temporaire `IdT(Martin)`, unique dans le domaine `visitor.mobile.de`, de façon à ce que son agent de représentation soit référencé `IdT(Martin).visitor.mobile.de`.

Note 3.1 *Le protocole IP actuel permet un adressage dont le nombre d'adresses disponibles est de plus plus en plus faible. Ceci rend difficile l'affectation d'adresses IP temporaires à des agents de représentation. Le nouveau protocole, IPv6, permet un adressage beaucoup plus important [H96] et peut donc mettre à disposition de domaines pour mobiles une collection d'addresses IP importantes.*

Remarque 3.3 *Les domaines `mobile.lifl.fr` et `visitor.mobile.de` peuvent être des stations accueillant respectivement les RIF des utilisateurs de l'organisation* LIFL *et les agents mobiles des visiteurs mobiles en Allemagne.*

La résolution des domaines `fr`, `lifl` et `mobile` peut se dérouler comme dans les DNS classiques. Par contre, la résolution de `martin` sera effectuée en envoyant une requête à la RIF de *Martin* qui retournera l'adresse courante de l'agent de représentation, c'est-à-dire l'adresse de `IdT(Martin).visitor.mobile.de`. Par conséquent, le domaine `mobile.lifl.fr` se comporte comme dans le cas d'un serveur de noms dans le GNS après un déplacement du domaine `martin` sous domaine de `mobile.lifl.fr` vers le domaine `visitor.mobile.de` sous le nom `IdT(Martin)`.

La figure 3.3 illustre ce mécanisme composé de quatre étapes :

1. Un client envoie une demande de résolution de l'adresse virtuelle `martin.mobile.lifl.f`
2. Le serveur `mobile` reçoit la requête et interroge la RIF de *Martin*.
3. La RIF retourne l'adresse physique de l'agent de représentation de Martin.
4. Le serveur `mobile` retourne cette adresse au client.

Améliorations Trois circonstances différentes peuvent être identifiées dans la résolution d'une adresse d'un agent de représentation :

1. Un interlocuteur de la zone géographique d'origine de l'utilisateur obtient une conversion rapide de l'adresse virtuelle et obtient donc rapidement l'adresse de l'agent de représentation. Le serveur de noms, la RIF, est en effet local à l'interlocuteur.

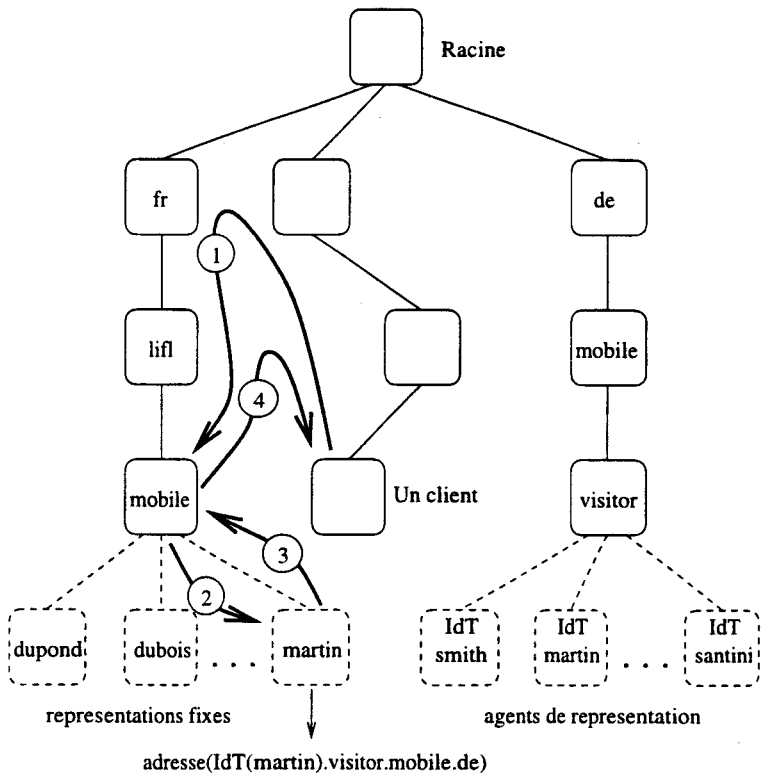


FIG. 3.3 - *Résolution de noms pour agents de représentation*

2. Dans le cas d'un interlocuteur qui se trouve dans une zone différente à la fois de la RIF et de l'agent de représentation. La conversion de l'adresse virtuelle de l'utilisateur nécessitera de contacter la RIF. Dans ce cas, des communications distantes sont nécessaires, mais pas plus que dans le cas d'une résolution classique d'adresse qui ne serait pas présente dans des caches.

3. Si l'interlocuteur se trouve dans la zone géographique de l'agent de représentation, la résolution d'adresse pour contacter l'agent de représentation nécessite des communications distantes alors que l'agent de représentation est dans la même zone géographique que l'utilisateur.

La résolution effectuée dans le troisième cas, lorsque l'interlocuteur et l'agent de représentation sont dans la même zone géographique, peut être améliorée par une résolution locale à la zone de l'agent de représentation. Ce mécanisme a déjà été évoqué dans le cas de l'obtention de l'adresse d'un mobile par un serveur fixe dans [BIV92] et est aussi utilisé dans le réseau GSM (voir partie 1.2.2.1).

Par conséquent, pour reprendre l'exemple de *Pierre Martin*, cela signifie que le serveur de noms du domaine `de` dispose, en plus des tables de conversion statiques, une table de conversion dynamique qui permet une conversion de noms des visiteurs du domaine `de`. Cette table permet donc la résolution du nom `martin.mobile.lifl.fr` en l'adresse de `IdT(Martin).visitor.mobile.de`.

Conséquences dans le système de nommage Deux procédures doivent être mises en place pour obtenir une bonne conversion à partir de l'adresse d'une RIF en adresse de l'agent de réprésentation.

Lorsqu'un agent de représentation arrive sur un site accepteur, il doit impérativement communiquer son adresse à la RIF. Le site qui accepte l'agent de représentation doit communiquer le couple composé de la référence de la RIF et l'adresse de l'agent de représentation au serveur de nom du domaine du site.

Lorsque l'agent de représentation quitte un site accepteur, ce site demande au sur-domaine de retirer le couple d'adresses permettant la conversion dans la zone locale à l'agent de représentation. Cependant, les messages en transit sont retransmis vers le nouveau site accepteur par l'ancien.

3.2 Un modèle de la délégation de tâches

Comme cela a déjà été introduit dans la partie 2.2, la délégation de tâches vers l'agent de représentation permet à une application de reporter une partie de son travail. Cet agent, présent sur un site accepteur, bénéficie d'un environnement plus performant et toujours déconnecté pour mener à bien une tâche. Dans cette section, nous allons introduire un nouveau concept de délégation de tâches : la délégation de tâche incomplète. Ce type de tâche se charge de trouver sur le réseau en cas de besoin les composants nécessaires pour se compléter.

3.2.1 L'envoi de tâches

L'utilisateur envoie des tâches à partir d'un terminal vers son agent de représentation. L'agent de représentation a en charge l'acceptation de code de tâches, leur exécution et le retour de résultat vers l'utilisateur.

3.2.1.1 Composants dans l'agent de représentation

L'agent de représentation est composé de :

1. un chargeur de code de tâche qui vérifie la provenance de la tâche (voir partie 3.2.3.1) et de la transmettre à l'interpréteur
2. un interpréteur qui permet l'exécution d'une tâche en lui offrant un environnement d'exécution (CPU, mémoire), un accès à des ressources et des services
3. une structure de données pour retourner les résultats (détaillée dans la partie 4.1)

3.2.1.2 Limites

La délégation de tâches, dont le code est complètement défini, offre un moyen de décharger une exécution de code du terminal de l'utilisateur vers l'agent de représentation. Ce dernier bénéficie de caractéristiques plus avantageuses que le terminal telles que puissance de calcul, connexion continue, ressources en plus grand nombre

et de meilleure qualité. Cependant, une tâche risque d'être envoyée et utilisée pour une longue période sans possibilité de modification (c'est le cas notamment des tâches abonnement). De plus, aucune mise en commun de code n'est possible entre différentes tâches appartenant à différents utilisateur, par exemple, tous les utilisateurs désirant recevoir le bulletin météorologique quotidien vont créer chacun une tâche dont le code est similaire et l'envoyer vers leur agent de représentation respectif.

Limite de la délégation de tâche par envoi de programmes «complets» La délégation par envoi de programmes décrivant entièrement une tâche ne permet pas, contrairement à la délégation par requêtes, la mise en commun de code définissant une tâche (voir partie 2.2.1). De plus, la délégation par envoi de programme peut entrainer un surplus d'utilisation de la liaison sans fil contrairement à la délégation par requêtes. Ce surplus est dû à la taille du code de la tâche envoyée. L'envoi d'une tâche non complètement définie permet de bénéficier des avantages des deux formes de délégations à la fois : la délégation par requête et celle par envoi de programme. Le complément de la tâche se trouve, dans ce cas, sur le réseau comme cela sera décrit ultérieurement.

Inconvénients des requêtes émises à partir de tâches La solution recherchée est de pouvoir conjuguer les avantages, à la fois, de la délégation par envoi de programmes et de celle par envoi de requêtes pour mettre à disposition un ensemble de services déjà présents sur le réseau et accessible par plusieurs tâches clientes. Une des possibilités serait de permettre l'appel de requêtes à partir d'une tâche émise (voir partie 2.2). Cependant le problème des requêtes, dans certains cas, est d'envoyer les données sous forme de paramètres au serveur. Ces données peuvent être importantes en taille ou confidentielles. La tâche peut préférer chiffrer une donnée de très grande taille avant de l'envoyer sous forme confidentielle à une autre entité. C'est pourquoi, il est préférable que cette donnée reste dans l'agent de représentation.

3.2.2 Importation de code dans une tâche

L'envoi vers l'agent de représentation d'une tâche qui n'est pas complètement définie mais qui est capable d'importer le complément à partir d'autres sites est une solution. Cela permet la création de code plus petit circulant sur le réseau et la mise en commun de code par plusieurs tâches. De plus, plutôt que d'exporter des données pour leur faire subir des traitements à l'extérieur, le code est importé sur le site de la tâche. C'est pourquoi, nous allons définir le concept d'importation de code distant pour une tâche.

3.2.2.1 Définitions

Définition 3.2 *L'*importation de code distant *permet à une tâche T définie par le code C1 en cours d'exécution de demander le chargement de code C2 disponible auprès d'un serveur, d'exécuter C2 en fonction d'éventuels paramètres, de récupérer une valeur et ensuite de continuer l'exécution du code C1 (voir Figure 3.4).*

L'importation de code à partir de sites distants ressemble beaucoup au chargement à partir de sites distants de classes Java, appelées *Applets* [Java96] dans un navigateur *Hotjava* [HotJava97] ou à l'importation de *Servlets* dans un serveur *JavaServer* [JavaServer97]. En effet, les applets viennent compléter par du code l'interface du navigateur, les servlets apportent un nouveau service au serveur HTTP.

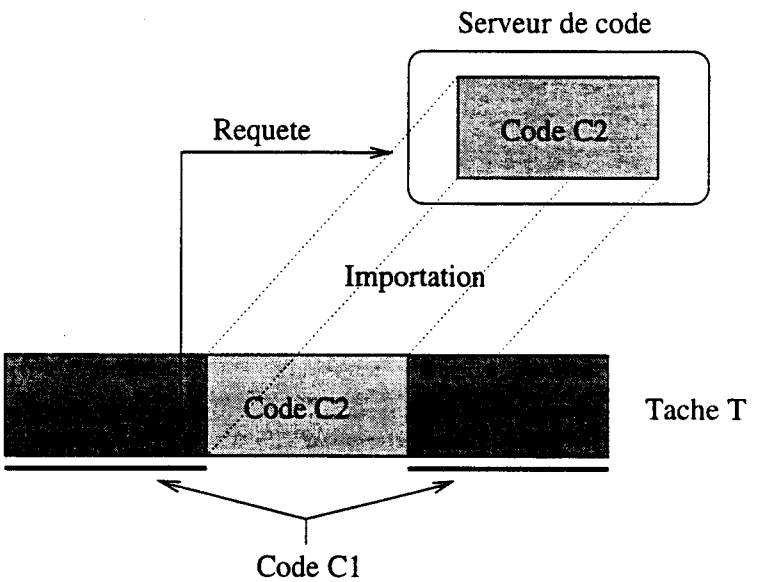


FIG. 3.4 - *Description d'une importation de code distant*

Définition 3.3 *Un* Code Personnel de Description de Tâche *ou* CPDT *est un script créé à partir du terminal de l'utilisateur et envoyé ensuite vers l'agent de représentation associé afin d'accomplir la tâche décrite par le script [CT97a].*

Définition 3.4 *Un* Code Importé Distant *ou* CID *est un script pouvant être importé d'un autre site et intégré dans un autre script (CPDT ou CID).*

Note 3.2 *Une tâche destinée à l'agent de représentation est complètement définie par l'union d'un CPDT et de l'ensemble des CID nécessaires pour être accomplie.*

3.2.2.2 Intérêts de l'importation de code dans une tâche

Trois types de délégation de tâches peuvent être recensé :

1. L'envoi d'une tâche autonome, ne faisant appel à aucun code extérieur.
2. L'envoi d'une tâche faisant appel à du code extérieur par des requêtes.
3. L'envoi d'une tâche important du code provenant de serveurs.

L'importation de code dans une tâche déléguée apporte, notamment par rapport aux autres types de tâche, un moyen d'utiliser plus modérément la liaison utilisateur-réseau et une meilleure encapsulation des données présentes dans la tâche.

Nous présentons ci-après les apports de l'importation de code distant [CT97b].

Economie de l'utilisation la liaison utilisateur-réseau La création d'une tâche par une application de l'utilisateur peut impliquer un chargement manuel ou automatique de fonctions ou procédures déjà définies et disponibles sur le réseau. C'est souvent le cas, lorsqu'en programmation, on charge de nouvelles bibliothèques provenant d'autres sites.

Le code de ces fonctions/procédures est chargé sur le terminal de l'utilisateur par la liaison extérieure, inséré dans le code en cours d'élaboration puis ré-exporté vers le réseau dans le code de la tâche qui vient d'être créée. Dans ce cas, ces fonctions/procédures transitent deux fois par la liaison utilisateur-réseau. De plus, elles peuvent être chargées sans être utilisées, si par exemple elles décrivent des actions à réaliser en cas de panne d'un serveur par exemple. L'importation de CID dans une tâche, pendant l'exécution et non plus à la création de la tâche, permet de charger un morceau de code en cas de besoin seulement. Ce chargement s'effectue seulement vers l'agent de représentation et n'utilise pas, par conséquent, la liaison réseau-utilisateur.

Description d'accès à des ressources extérieures L'obtention de certaines informations peut s'avérer fastidieuse pour le programmeur. Par exemple, l'utilisateur peut être amené à d'abord contacter un serveur d'adresses puis un serveur d'autorisations et enfin le serveur lui-même et cela avec des protocoles de communication qui doivent être définis. Dans ce cas, l'utilisateur envoie vers son agent de représentation une tâche qui n'a qu'à importer puis exécuter le code prédéfini importé à partir de serveurs de CID.

Facilités de mise-à-jour Prenons le cas d'un serveur qui propose des CID décrivant les opérations à effectuer pour communiquer avec lui (par exemple le protocole et la sécurité). L'importation de code seulement pendant l'exécution offre la possibilité au serveur d'effectuer des mises-à-jour, des modifications des CID. Ce serveur peut ainsi changer à tout moment son protocole de communication ou de sa politique de sécurité sans que les tâches communiquant avec ce serveur ne deviennent inutilisables car périmées. Ces taches importeront toujours un CID de la même façon.

3.2.2.3 Types d'importations de code distant

Deux types d'importation de code peuvent être identifiés. Celle de CID généraux, pouvant être utilisés par différentes tâches et celle de CID personnalisés dont l'usage est restreint à un nombre limité d'usagers (voire un seul).

Importation de CID généraux Le fait de mettre à disposition des «morceaux» de codes généraux (c'est-à-dire utilisables par plusieurs clients) permet un partage entre différentes tâches d'un même code. Ce code peut être vu comme une fonction / procédure définie en dehors du CPDT provenant de l'utilisateur. Le code des tâches présentes dans l'agent de représentation est, par conséquent, plus petits et ne contient qu'un ensemble d'actions de haut niveau, beaucoup plus lisibles pour le programmeur. Ce code consiste en une demande d'exécution de commandes plus complexes et importées seulement en cas de besoin d'exécution. Ceci apporte un

gain de place utilisée au niveau des agents de représentation et donc au niveau des sites accepteurs.

Importations de CID personnalisés Il est possible d'obtenir des codes distants personnalisés à partir d'une demande d'importation effectuée par une tâche. La requête sur un serveur pour obtenir un CID personnalisé doit contenir des paramétres de génération provenant de la tâche cliente.

Exemple 3.3 *Ces codes distants sont utiles lorsqu'un utilisateur mobile demande une description des communications à effectuer avec des serveurs locaux. Un CID est importé dans la tâche pour obtenir, par exemple, la carte de la ville dans laquelle il se trouve. Le paramètre de génération est, dans ce cas, la localisation de l'utilisateur. Il reçoit ainsi le code à exécuter pour obtenir ces renseignements et l'adresse des serveurs locaux à contacter.*

L'utilisation des CID personnalisés est très intéressante pour l'accès à des services locaux pour l'utilisateur ne sait pas comment les contacter. La génération de ces CID est ici personnalisée à partir d'une donnée relative à l'utilisateur: sa localisation géographique. Le serveur fournit un code à partir d'une base de données.

L'utilisation de CID personnalisés est aussi très utile afin de donner au client des droits sur une ressource. Si ces droits varient en fonction de l'individu, une requête vers un serveur de droits ou d'autorisation, peut conduire à l'obtention d'un CID qui décrit les moyens d'accès à ces ressources et lui donne des capacités. Par conséquent, le CID ou même seulement la capacité permettant l'accès doit être illisible pour toute personne. Dans ce cas, soit les données sensibles sont chiffrées et l'utilisateur est le seul à détenir la clé de déchiffrement (elle peut avoir été acquise lors de l'abonnement au service par l'application émétrice de la tâche), soit le serveur chiffre les données à partir d'une clé de chiffrement publique relative à l'agent qui est le seul à détenir la clé de déchiffrement (voir partie 4.2.1).

3.2.2.4 Architecture de caches pour l'importation de CID

L'importation de code distant peut être très fréquente et risque de diminuer les performances relatives à l'excécution du code d'une tâche. L'utilisation d'une mémoire cache mise à disposition de l'agent de représentation permet de réduire le temps d'attente pour obtenir un CID. Par contre, il est nécessaire de prendre en compte les problèmes de mise-à-jour des CID présents dans la mémoire cache.

Pourquoi et comment utiliser un cache L'utilisation de code distant peut amener à importer plusieurs fois consécutivement le même. Le cas le plus crucial est la demande d'importation d'un même CID effectuée à l'intérieur d'une boucle. Il est tout à fait inconcevable de contacter un serveur puis d'importer le même code à chaque traitement du corps de la boucle. L'utilisation de mémoire cache est une bonne solution comme cela sera décrit dans les prochains paragraphes.

Il est nécessaire de définir les propriétés d'identité entre deux CID afin de déterminer si un CID est déjà dans le cache où s'il faut l'importer. Deux CID importées sont identiques si :

- Le serveur contacté est le même
- l'identificateur de CID sur ce site est le même
- si les paramètres liés à la génération du CID sont identiques

Par conséquent, lors d'une demande d'importation de CID, il est indispensable de distinguer deux types de paramètres :

- Les paramètres liés à la génération du CID par le serveur
- Les paramètres liés à l'exécution lors de l'exécution de ce CID dans l'agent de représentation

Le test de présence d'un CID dans le cache peut s'effectuer grâce à une fonction de hachage qui renvoie l'adresse du code distant dans le cache à partir de paramètres tels que l'adresse du serveur, l'identificateur de ce code et, si nécessaire mais cela augmentera le temps de cacul, de tout ou une partie des paramètres liés à la génération du code distant.

Responsabilité de la gestion du cache Nous allons nous intéresser à identifier les différents acteurs de la gestion du caches du CID et définir leur rôle. Ces acteurs sont, soit l'agent de représentation pour un cache individuel, soit le site accepteur pour un ensemble d'agents de représentation, donc d'utilisateurs.

L'agent de représentation responsable de l'importation de code Les requêtes pour obtenir un CID peuvent être prises en charge par l'agent de représentation. Dans ce cas, chaque agent de représentation est responsable de la gestion de l'utilisation de CID et notamment, c'est à lui qu'il revient de gérer une mémoire cache.

Le site accepteur responsable de l'importation de code La solution précédente semble utopique dans le cas d'une mise-à-jour de CID. En effet, dans ce cas, ce serveur est obligé d'envoyer un message à tous ses anciens clients, des agents de représentation, qui auraient gardé le CID dans leur cache. Par conséquent, les serveurs doivent maintenir des listes importantes. Par contre, si le site accepteur est responsable de la gestion du cache, dans ce cas, la liste des caches à mettre à jour est moins importante. Les messages de mise-jour sont envoyés au sites accepteurs, pas à l'ensemble des agents de représentation.

De plus, la présence d'un cache au niveau du site accepteur permet de mettre en commun les caches de tous les agents de représentation sur le même site. Pour une disponibilité globale de codes identique, beaucoup moins de mémoire cache est nécessaire. Tout dépend en fait du recouvrement des codes importés par chaque agent de représentation d'un même site accepteur. Il est cependant prévisible que beaucoup de CID seront appelés plusieurs fois par différents agents de représentation notamment en ce qui concerne les CID appelant des serveurs locaux.

Différents niveaux de cache Il est possible de faire co-exister une mémoire cache sur deux niveaux différents (voir Figure 3.5) : celui du site accepteur et celui de l'agent de représentation. Ce qui conduit à utiliser une hiérarchie de caches comme dans [D94]. D'une part, les temps d'accès peuvent être sensiblement meilleurs sur le cache de l'agent que sur celui du site accepteur. D'autre part, certains CID, appelés seulement par un seul agent de représentation parmi ceux présents sur le site accepteur, peuvent ne pas subsister longtemps dans un cache géré par le site. En effet, à l'échelle du site accepteur, ces CID seront peu appelés alors que, à l'échelle de l'agent de représentation, les demandes d'accès seront proportionnellement plus nombreuses. D'où l'utilisation de deux niveaux de cache :

1. un niveau général pour tous les agents de représentation du site accepteur
2. un niveau réservé à un seul agent de représentation.

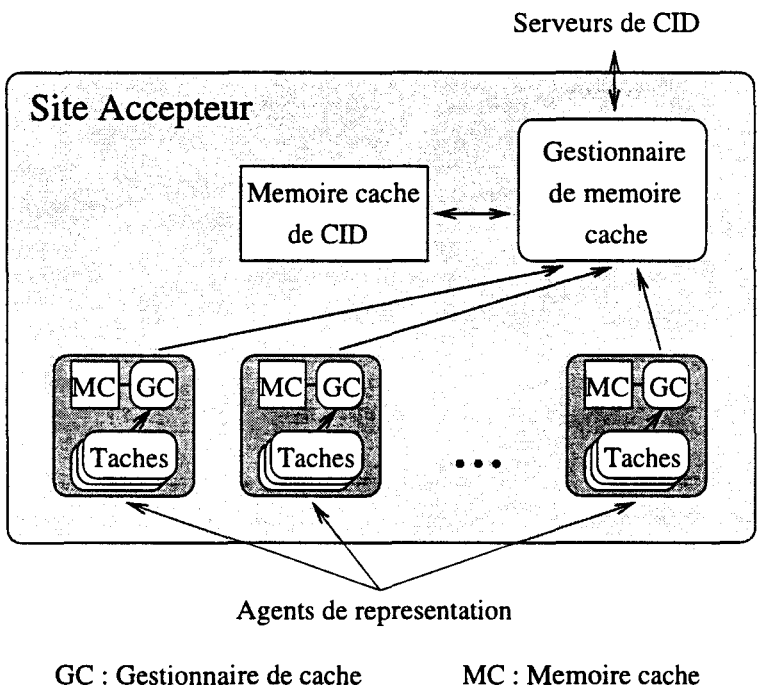


FIG. 3.5 - *Description du cache dans un site accepteur*

Amélioration de la gestion des caches de CID Une amélioration de la gestion des caches peut être proposée. En même temps que de retourner un code, un serveur de CID peut l'accompagner d'une donnée qui garantit la cohérence de ce CID pour au moins une durée déterminée. Cette technique est déjà utilisée et appelée TTL[3] dans les serveurs de noms du DNS [C95b]. Dans ce cas, le serveur de codes distants est déchargé de toute obligation de prévenir les sites accepteurs

3. TTL : Time To Live

d'une éventuelle mise-à-jour. Après l'expiration de la durée de cohérence garantie, la version du code importé dans le cache deviendra inutilisable et le chargement d'une mise-à-jour sera effectué lors d'une requête suivante. L'utilisation d'une durée de cohérence garantie est donc intéressante pour les CID qui varient peu, voire pas du tout.

3.2.3 Sécurité

Déléguer une tâche de l'utilisateur vers son agent de représentation, ou importer du code de sites extérieurs imposent plusieurs règles qui dépendent notamment du niveau de sécurité désiré. La confidentialité, l'authentification peuvent être indispensables. L'utilisation d'exceptions permet de garantir une exécution dans un mode dégradé lorsque certains critères ne sont pas respectés.

3.2.3.1 Tâches provenant de l'utilisateur

L'envoi d'une tâche de l'utilisateur vers son agent de représentation peut imposer une politique de sécurité entre les deux. Ces règles sont générales aux messages échangés entre l'utilisateur et son agent de représentation et seront développées dans la partie 4.2.1.

3.2.3.2 Tâches distantes

L'importation de code distant consiste à obtenir d'un serveur extérieur du code et de l'exécuter localement. De plus, l'obtention d'un CID peut exiger le respect d'une politique de sécurité afin de garantir confidentialité, identification, authentification, etc...

Classification des CID La politique de sécurité liée à l'utilisation d'un code distant dépend de la nature de celui-ci. C'est pourquoi, il est intéressant de classer les CID, en fonction du but de leur utilisation, et, à partir de cette classification, de définir différentes manières de les obtenir.

CID généraux/personnalisés Comme décrit en 3.2.2.3, deux types différents de codes distants peuvent être obtenus : les CID généraux (dont le code est le même pour tous les clients), les CID personnalisés (dont le code est adapté au client). La personnalisation peut être effectuée grâce aux paramètres de génération. Ces paramètres déterminent certaines phases (voire toutes) de la génération du code et sont donc tout à fait désignés pour ce genre d'utilisation. Un appel de code distant sans paramètre de génération peut être considéré comme général alors qu'avec des paramètres de génération, le CID est personnalisé.

CID libres/certifiés Un CID certifié est un code dont le client est sûr de l'authenticité contrairement à un CID libre. La vérification de l'authenticité peut être effectuée par l'interpréteur de la façon suivante. Soit le couple de clés de chiffrement-déchiffrement (`Kc`,`Kd`) associé à l'obtention d'un code distant ou d'une famille de

codes distants (par exemple fournis par le même prestataire de services). Kc est une clé privée exclusivement détenue par le serveur de code désiré ou éventuellement par une classe de serveurs habilités à le délivrer. Kd est la clé publique de déchiffrement associée à Kc telle que Kc(Kd(m))=m. L'interpréteur envoie la demande de code vers le serveur qui retourne le code désiré chiffré par la clé Kc. Ce code peut aussi se trouver dans le cache sous la même forme (c'est-à-dire chiffré par Kc). L'interpréteur déchiffre grâce à la clé publique de déchiffrement Kd. L'interpréteur est ainsi sûr que le CID provient d'un détenteur de la clé Kc, donc d'un serveur habilité à délivrer ce CID. Ces mécanismes sont classiques et mieux détaillés dans [S94] et dans la partie 4.2.1.

CID publics/protégés Un CID public est un CID accessible par tout client alors qu'un CID protégé est un CID qui doit être envoyé seulement à des personnes autorisées. On peut, de la même façon que précédemment, utiliser un couple de clés (Kc,Kd), mais cette fois-ci, la clé Kd est privée et n'appartient qu'à l'agent de représentation, alors que la clé Kc est publique [S94]. Le serveur enverra ainsi le code distant sous forme chiffrée par Kc, puis l'interpréteur utilisera la clé Kd disponible sur l'agent de représentation pour déchiffrer le code. Dans ce cas, un code protégé, c'est-à-dire dédié à l'utilisation d'une personne ciblée, n'a pas à se trouver dans le cache du site accepteur mais plutôt dans celui plus personnel de l'agent de représentation.

CID maillons/terminaux Un CID peut demander l'importation d'autres CID. Cependant dans certains cas, notamment lorsqu'une tâche (à partir d'un CPDT ou d'un CID) demande l'importation d'un code certifié, une autre importation imbriquée peut ne pas être désirée. En effet, lorsqu'un code importé est certifié, cela permet d'être sûr de son origine qui est un serveur de confiance pour l'opération désirée. Cependant, si ce code appelle d'autres codes distants sur des serveurs, la relation de transivité dans le domaine de la confiance n'est pas garantie. C'est pourquoi, il convient de proposer, deux types de CID : (1) les CID maillons autorisés à demander l'importation d'autres codes distants et (2) les CID terminaux qui ne le peuvent pas. L'interpréteur du site accepteur est en charge de vérifier l'autorisation ou non d'importation en fonction du type de code en cours de traitement.

Serveurs de clés Certaines classes nécessitent l'obtention de clés publiques notamment pour la certification de CID. Cependant l'interpréteur demandant une clé à un serveur pour authentifier l'origine du CID doit pouvoir être sûr de l'information qu'il recevra. C'est-à-dire qu'il doit être sûr du serveur de clé et que l'information retournée n'ait pas été modifiée.

Sécurité assurée à l'exécution Lorsqu'un CID est importé, celui-ci est complétement intégré à l'exécution de la tâche appelante. Pendant le temps de son exécution, le code a accès à l'espace de donnée de la tâche appelante. Par conséquent, il a autant de droit de lecture et de modification sur les données que le script appelant. C'est pourquoi, l'interpréteur doit garantir au client du code (c'est-à-dire à la tâche appelante) que seules les données passées en paramètres d'exécution soient lisibles

ou / et modifiables selon les droits en lecture / écriture accordés, comme dans des langages tels que ADA [Ada87].

3.2.3.3 Les exceptions

L'éxécution de tâches, les opérations d'importation ou d'exécution de code importé peuvent ne pas se dérouler comme prévu. Par conséquent, l'utilisation d'exceptions permet de ne pas interrompre «brutalement» une tâche. Le programmeur peut en effet décrire des actions à réaliser dans le cas où un problème surviendrait et prévoir un mode dégradé.

Utilisation d'exceptions

Définition 3.5 *Sous certaines circonstances, l'exécution d'un composant peut ne pas être effectuée comme prévu et entrainer une erreur. Un tel évênement est appelé une exception [M92].*

L'importation de code distant peut entraîner un échec de l'éxécution de ce code et donc de la tâche appelante elle-même. L'exécution normale d'un code distant peut ne pas être assurée pour diverses raisons et notamment :

1. le CID n'est pas accessible (panne réseau, panne du serveur de CID,...)
2. l'obtention du CID est refusée (pas d'autorisation)
3. l'exécution du CID échoue pour d'autres raisons

Dans le cas d'un échec lors de l'exécution de code distant, l'exécution est abandonnée et une exception est levée. Celle-ci peut être rattrappée à condition qu'elle soit déclenchée dans une zone où des actions ont été définies dans le cas où cette exception est déclenchée. Dans le cas contraire, elle est propagée vers la tâche appelante (CPDT ou un autre CID) jusqu'à ce qu'un traitement spécifique ait été définie pour traiter cette exception. Si ce traitement n'existe pas, la tâche ne peux plus fonctionner, elle est détruite. Un message peut être envoyé vers l'utilisateur pour lui signaler les raisons de cette destruction, par exemple, sous la forme d'une trace de propagation des exceptions.

Exceptions identifiées Plusieurs types d'exception peuvent être prédéfinis tels que :

- *ERREUR_TACHE*: une erreur d'exécution de tâche est survenue.
- *CID_INTROUVABLE*: Un serveur n'est pas accessible à cause d'une panne de celui-ci, du site accepteur ou du réseau,...
- *CID_ECHEC_CERTIFICATION*: L'authenticité d'un CID est remise en cause.
- *CID_CREDIT_INSUFFISANT*: Le coût pour l'obtention d'un CID dépasse le crédit disponible.

- *CID_TERMINAL* : Un CID comme CID terminal tente d'appelé un autre CID.

Il est possible aussi à l'intérieur d'une tâche distante ou personnelle de définir d'autres exceptions utilisables de la même façon.

3.3 Conclusion

Ce chapitre présente un modèle et une architecture permettant à des agents de représentation de voyager de site en site et d'accepter des tâches provenant de l'utilisateur.

Malgré ses déplacements, l'utilisateur doit toujours être joignable. C'est pourquoi, il était nécessaire de proposer un mécanisme de localisation performant. La migration impose aussi à l'agent de représentation de trouver un site accepteur qui réponde à des critères comme les services proposés. Là encore, un système de recherche de sites accepteurs adéquats a été proposé.

La délégation de tâches permet à un utilisateur d'envoyer un programme qui sera exécuté sur une machine plus performante. Cette tâche peut être définie partiellement. Elle fédère des composants logiciels appelés sur le réseau lorsqu'elle en éprouve le besoin. Cela permet notamment d'envoyer des tâches dont le code est réduit et d'importer du code toujours valide.

Les applications courantes sont souvent basées sur le modèle client-serveur. Ce modèle suppose une liaison fiable et constante. La délégation de tâches permet une communication asynchrone de la tâche et du résultat. La conception d'une application nécessite à la fois l'écriture de l'application sur le terminal mais aussi éventuellement d'une ou plusieurs tâches qui sont envoyées vers l'agent de représentation par l'application. Ces tâches retournent des résultats aux applications du terminal comme cela va être décrit dans le chapitre suivant.

Chapitre 4

Communications entre l'utilisateur et l'agent de représentation

Introduction

Ce chapitre présente le système de communication entre l'utilisateur et son agent de représentation. Ce système doit permettre le retour de résultats de tâches de l'agent vers l'utilisateur en les adaptant au matériel de l'utilisateur et à ses préférences. De plus, la mise en place d'une politique permettant de rendre plus sûre la liaison utilisateur - agent de représentation est nécessaire pour garantir confidentialité et authentification entre les deux entités.

Nous présentons un modèle et une architecture mettant en oeuvre différents objets prenant en compte ces objectifs. La carte à microprocesseur trouve un rôle adapté à ses caractéristiques et à son utilisation (voir partie 1.4).

4.1 Structure de données et communications

Dans ce chapitre, nous décrivons une architecture de composants logicielles pour réaliser un système utilisant des agents de représentation. Nous nous attacherons à détailler les éléments intervenant dans la communication des données provenant d'une tâche sur l'agent de représentation vers le terminal de l'utilisateur. Dans la section 4.1.1, nous présenterons les principaux composants ainsi que leurs rôles. Dans la section 4.1.2, la gestion des données destinées à l'utilisateur sera détaillée. Enfin

en 4.1.3, la structure des résultats sera présentée.

4.1.1 Composants pour la communication terminal-agent

La communication des données des tâches vers l'utilisateur passe par des composants à la fois présents dans l'agent de représentation et dans le terminal. Ces composants ont chacun un rôle spécifique permettant de gérer «au mieux» les inconvénients de la mobilité de l'utilisateur.

4.1.1.1 Les gestionnaires de communication

L'agent de représentation est un intermédiaire obligé dans toute communication entre le terminal et le monde extérieur. Par conséquent, il convient d'établir un canal de communication entre le terminal et l'agent de représentation. Ce canal de communication correspond aux couches présentation de la norme OSI[1] [T90]. Pour cela, deux *gestionnaires de communication* (*GC*) sont nécessaires (voir Figure 4.1) :

- Au niveau du terminal, le GC du terminal (GC_{term}), pour fournir les données venant du monde extérieure (via l'agent de représentation) aux applications présentes dans le terminal
- Au niveau de l'agent de représentation, le GC de l'agent (GC_{agt}), pour fournir les données demandées par une requête sous forme de tâche de l'utilisateur au GC_{term}

4.1.1.2 Les espaces de stockage des données

Les espaces de stockages permettent :

- de stocker des données destinées à l'utilisateur dans l'agent de représentation lors d'une deconnexion
- de rendre accessible des données aux applications de l'utilisateur soit sur le terminal, soit dans l'agent de représentation

Les données provenant de l'agent de représentation et destinées au terminal de l'utilisateur sont la conséquence de résultats de tâches initiées par l'utilisateur ou par une de ses applications. Ces données, qui lui sont destinées, sont stockées à deux endroits différents :

1. Dans le terminal, pour les données gérées par le GC_{term}qui correspondent aux données déjà reçues par l'utilisateur
2. Dans l'agent de représentation, pour les données gérées par le GC_{agt}correspondant aux données présentes dans l'agent et destinées à être envoyées vers l'utilisateur

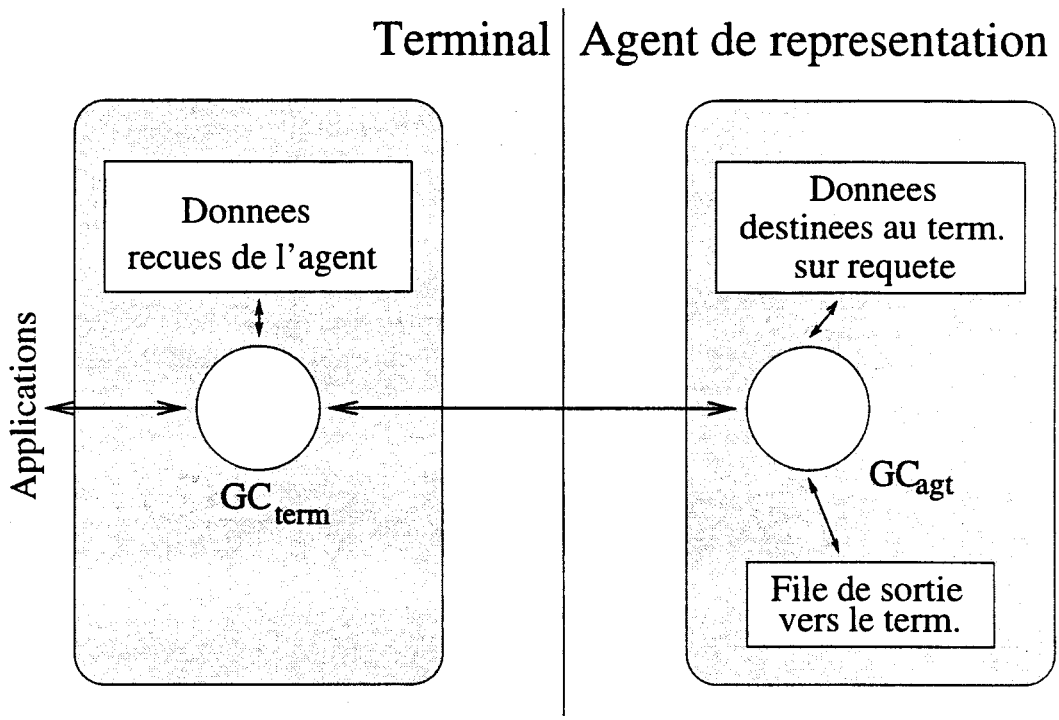


FIG. 4.1 - *Communication entre le terminal et l'agent de représentation*

Lorsqu'une application de l'utilisateur demande au GC_{term} une donnée déjà présente localement, celui-ci lui retourne la donnée directement sans utilisation de la liaison Terminal-Agent (voir Figure 4.2).

Par contre, dans cas où le GC_{term} ne trouve pas la donnée localement dans sa zone mémoire associée, le GC_{term} demande cette donnée au GC_{agt} qui lui fournit à partir de son espace de données. Le GC_{term} retourne donc cette donnée à l'application cliente et la stocke dans son espace de données de façon à ce qu'une autre demande de cette même donnée soit effectuée sans utilisation de la liaison Terminal-Agent (voir Figure 4.3).

Remarque 4.1 *Une demande de donnée absente du Terminal ne peut être traitée que si une liaison terminal-Agent est possible, c'est-à-dire que le terminal est connecté au réseau.*

4.1.1.3 Données destinées au terminal

Une donnée destinée à un utilisateur est envoyée par le GC_{agt} vers le GC_{term}. Si aucune liaison entre le terminal et l'agent de représentation n'existe, une file d'attente est utilisée pour y stocker les données à envoyer vers le terminal jusqu'à ce que la liaison redevienne opérationnelle.

1. OSI : Open System Interconnection

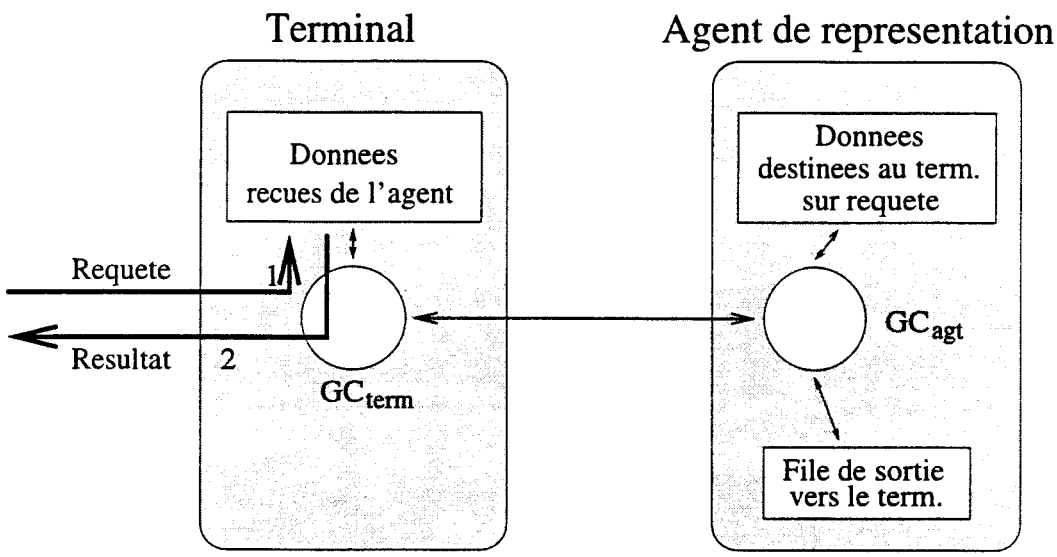


FIG. 4.2 - *Demande de données présentes dans le GC_{term}*

Cependant une donnée destinée à l'utilisateur peut être envoyée à l'objet gérant la file d'attente. Cet objet décide, soit de la transmettre immédiatement au GC_{term} si une liaison existe entre le terminal et l'agent de représentation, soit de mettre cette donnée en attente dans le cas contraire.

Les actions de la file d'attente, c'est-à-dire l'interface de l'objet *file d'attente*, peut être énumérées comme suit :

- **AjouterDonnée** : Ajoute une donnée dans la file
- **RetirerDonnée** : Envoie une donnée vers le GC_{term} et la retire de la file
- **DétruireDonnée** : Détruit une donnée sans l'envoyer au GC_{term}
- **ViderFile** : Envoie une à une les données vers le GC_{term} en les retirant de file

Ces actions sont utilisées dans des cas précis en fonction de la présence ou non d'une liaison entre le terminal et l'agent de représentation, c'est-à-dire de la connection du terminal au réseau. Le tableau 4.1 décrit les actions réalisées en fonction de l'état du terminal.

Remarque 4.2 *Lorsque le terminal est connecté, la file d'attente est soit vide, soit en train de se vider. Elle est surtout utilisée comme espace de stockage temporaire des données destinées à être envoyées au terminal (GC_{term}) dans le cas où le terminal n'est pas connecté et qu'une donnée lui est destinée.*

4.1.2 Gestion des données

4.1.2.1 Problématique d'identification des résultats

Une structure pour le stockage des données permet une mémorisation ordonné des résultats. Une donnée présente dans un espace de données doit être facilement

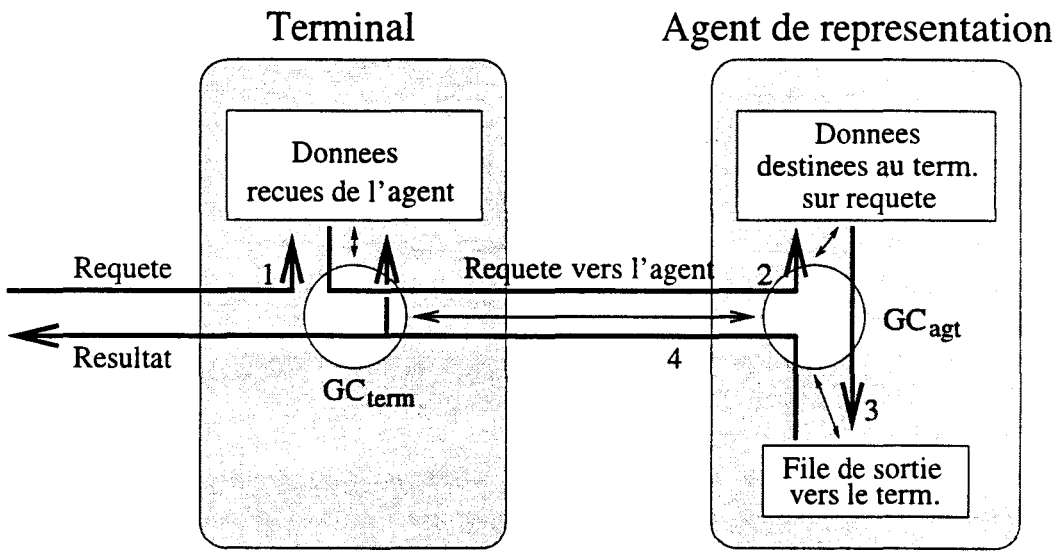


FIG. 4.3 - *Demande de données absentes du GC_{term}*

Etat du terminal	**Actions à réaliser**
connecté	réception d'une donnée ⇒ AjoûterDonnée + RetirerDonnée
non connecté	réception d'une donnée ⇒ AjoûterDonnée
connection + Δt	ViderFile

TAB. 4.1 - *Actions réalisées par la file d'attente*

localisée par un identificateur unique. De plus, ces données correspondent à des résultats délivrées par des tâches exécutées sur l'agent de représentation. Comme cela a été déjà introduit, trois espaces dans lesquels sont mémorisés les résultats sont présents :

1. Un dans le terminal correspondant à l'espace de données utilisée par le GC_{term}
2. Deux dans l'agent de représentation : l'espace de données utilisée par le GC_{agt}et la file d'attente

Il est donc nécessaire de définir une identification des données qui permette d'y accèder rapidement et une classification des résultats par tâche qui en sont à l'origine. De plus, une donnée peut être stockée dans un premier temps dans l'espace de données géré par le GC_{agt}, puis dans la file d'attente et enfin dans l'espace de données du GC_{term}. Par conséquent, pour des raisons de facilités, il est beaucoup plus intéressant qu'une donnée garde le même identificateur quelque soit l'endroit où elle est stockée.

4.1.2.2 Structure des espaces de données

Utilisation de listes de données associées à une tâche Comme énoncé dans la section 2.2.2.2, une tâche peut être envoyée à l'agent de représentation par l'utilisateur, soit pour retourner un résultat, soit pour retourner plusieurs résultats dans le cadre notamment d'une tâche de type *abonnement.* Par conséquent, l'ensemble des résultats retournés par une tâche constitue par exemple une liste de données qui peut être une liste d'articles de journaux, une liste de messages électroniques, etc... Une liste est composée d'un seul élément dans le cas d'une tâche retournant un seul résultat ou de plusieurs éléments si la tâche renvoie plusieurs résultats. Dans certains cas, cette liste peut même être vide si une tâche n'a pas encore envoyé de résultat ou si les résultats d'une tâche ont été détruits.

Ces listes de données associées chacune à une tâche doivent être stockées dans les espaces de données utilisés par le GC_{term}et le GC_{agt}. Par conséquent, la structure de données des deux espaces peut avoir une représentation hiérarchique : un premier niveau permet de choisir la tâche productrice de la donnée recherchée puis un second niveau permet de choisir une donnée parmi celles qui ont déjà été retournées par la tâche et enfin, le dernier niveau correspond aux données proprement dites, c'est-à-dire aux résultats élémentaires retournés par les tâches.

Hiérarchie d'accès aux données L'expression de la hiérarchie des données peut prendre la forme de celles des gestionnaires de fichiers tels qu'ils existent dans les systèmes de fichiers *Unix.* Cette représentation sous forme d'arborescence de fichiers sera celle utilisée.

Par conséquent, plusieurs éléments doivent être définis :

- La *racine* permet d'obtenir un ensemble de *répertoires* correspondant à la liste des données fournies par une tâche.
- Les *répertoires de tâches* contiennent des *fichiers* contenant chacuns une donnée envoyée par la tâche à laquelle est associé le répertoire de tâche.
- Les *données* peuvent être considérées (dans un premier temps) comme des fichiers contenant une donnée envoyée par une tâche.

Comme représenté dans la figure 4.4, une donnée `D` produite par une tâche `T` peut être identifiée par `/Id(T)/Id(D)`, `/Id(T)` étant le chemin où est enregistrée la donnée D. L'identificateur d'une tâche est, par exemple, un nombre attribué à partir d'un compteur présent dans le terminal, incrémenté à chaque émission de tâche et retourné à l'émetteur de la tâche. L'identificateur de la donnée peut aussi être un nombre `N` correspondant au fait que cette donnée soit la N^{ieme} retournée par la tâche.

Structure de la file d'attente La file d'attente est un espace de stockage de données situé dans l'agent de représentation et contenant des données destinées au terminal de l'utilisateur. Ces données, arrivées sur le terminal par l'intermédiaire du GC_{term}, sont rangées dans l'espace de données hiérarchique géré par le GC_{term}. Le GC_{term}joue, dans ce cas un rôle de répartiteur de données. Par conséquent, un

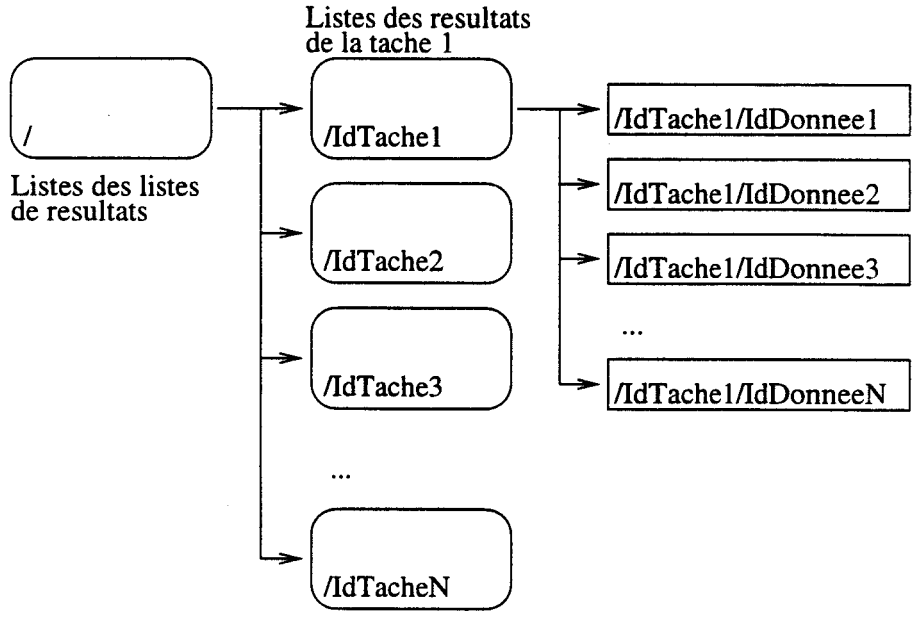


FIG. 4.4 - *Structure de données des espaces de données associés aux GC*

élément de la file d'attente doit contenir, non seulement la donnée elle-même, mais en plus une identification de l'endroit dans la structure de données utilisée par le GC_{term} où ranger cette donnée, c'est-à-dire la chemin et l'identificateur de la donnée (voir figure 4.5).

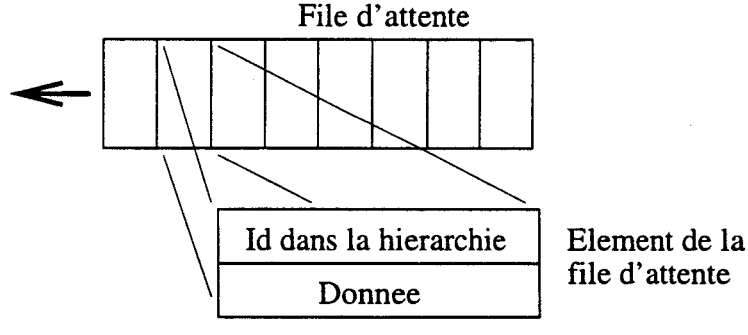


FIG. 4.5 - *Structure des données de la file d'attente*

4.1.2.3 Structure des données destinées à l'utilisateur

Objets-Résultats Les éléments destinés à l'utilisateur, considérés comme les feuilles de la structure hiérarchique, représentent les résultats provenant de tâches. Ils ont volontairement été qualifiées de «données». Pour affiner leur description et pour utiliser la notion d'objet, celles-ci seront qualifiées, à partir de maintenant, d'*objets-*

résultats de tâche ou plus brièvement de *résultats.*

Définition 4.1 *Un* objet *est une encapsulation d'un ensemble à la fois d'opérations ou de méthodes qui peuvent être invoquées de l'extérieur et d'un état qui évolue en fonction des méthodes [BGH91].*

Définition 4.2 *Une* classe *est une définition d'une implantation (méthodes, c'est-à-dire les opérations, et structure de données) partagée par un groupe d'objets [BGH91].*

Tous les objets-résultats provenant de tâches sont des instances d'une même classe abstraite de résultats et contiennent le résultat d'une tâche et des méthodes pour y accéder.

Hiérarchie de résultats Dans les paragraphes précédents, une donnée retournée était juste un objet-résultat et donc, dans les répertoires `/Id(T_i)` des espaces de données gérés par le GC_{term}et le GC_{agt}, se trouvaient une liste d'objets-résultats comme indiqués dans la partie précédente. Cependant, pour profiter pleinement de la structure de données hiérarchique, il est possible de créer des sous-répertoires sur plusieurs niveaux aux répertoires `/Id(T_i)` permettant tous d'accéder finalement après un parcours de l'arborescence aux feuilles, c'est-à-dire à des objets-résultats.

La hiérarchie de ces répertoires est définie par la tâche envoyée par l'application.

En prenant l'exemple d'une tâche `MonJournalFavori`, qui est un abonnement au journal que l'utilisateur préfère, tous les objets-résultats sont des objets instances d'une classe `Article`. Dans le répertoire `/Id(TacheJournal)` de l'espace de données du GC_{term}et du GC_{agt}réservé à la sauvegarde des résultats de la tâche peuvent se trouver en fait une liste de répertoires correspondant aux envois de données de la tâche, un sous-répertoire par numéro par exemple. A l'intérieur de ces sous-répertoires, une liste de rubriques est présentes (Politique, Economie, Faits divers, Sport,...) comme décrit dans la figure 4.6. A la fin du parcours d'une branche, on accède à une liste d'objets `Article`.

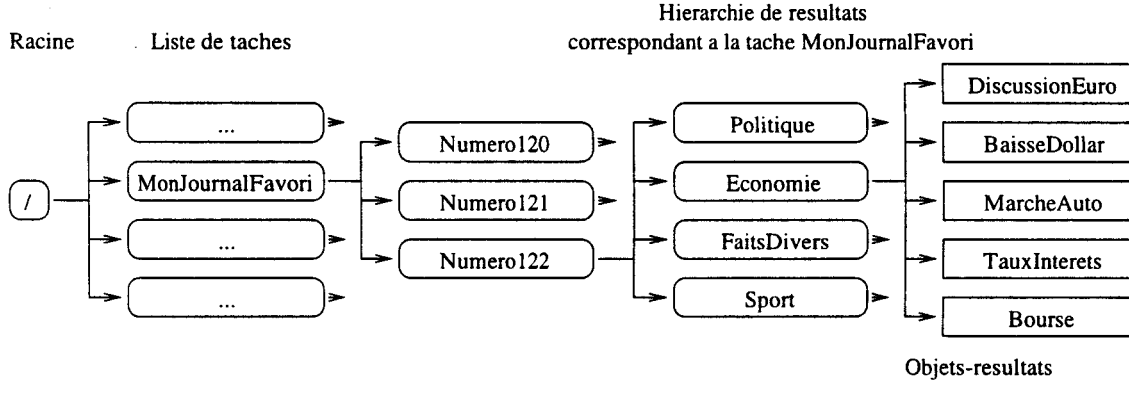


FIG. 4.6 - *Exemple de hiérarchie de résultats*

Type d'envoi de données D'après la partie 2.3.2.1, certains résultats destinés à un ordinateur mobile sont envoyés :

- automatiquement vers le mobile pour éviter une utilisation de la liaison sans-fil
- seulement par requête dans le cas où la taille d'un résultat est beaucoup trop importante et pour lequel on préfère l'envoyer seulement après une requête de la part de l'utilisateur

Le répartiteur est un objet encapsulé dans le GC_{agt}et / ou communiquant avec le GC_{agt}dont la fonction est de déterminer si un résultat provenant d'une tâche est à envoyer vers la file d'attente ou / et vers l'espace de donnée comme décrit par la figure 4.7. Cependant la décision d'envoyer à l'un ou l'autre des composants un résultat n'est pas du ressort du répartiteur, mais de la responsabilité des tâches. C'est pourquoi, les tâches associent une description du *type d'envoi* aux résultats. Les valeurs de cette description peuvent être *Automatique* ou *ALaDemande*.

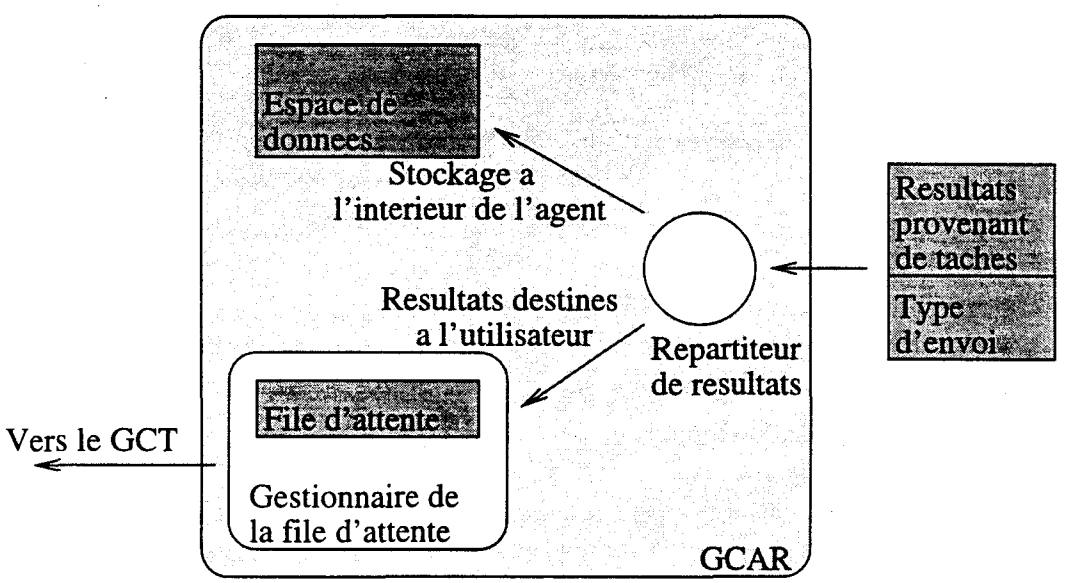


FIG. 4.7 - *Répartiteur de données*

stratégies du répartiteur Deux stratégies peuvent être mises en place :

1. Economie de l'espace mémoire utilisé par l'agent de représentation

   Afin de réduire l'epace mémoire utilisé par le GC_{agt}, on peut décider de ne pas conserver les résultats déjà envoyés vers l'utilisateur. Les implications de cette stratégie sont doubles. D'une part, le répartiteur envoie un résultat, ou vers l'espace de stockage de données utilisé par le GC_{agt}, ou vers la file d'attente et non pas vers les deux à la fois. D'autre part, en ce qui concerne les résultats

délivrés seulement après une requête de l'utilisateur, ceux-ci sont retirés de l'espace de stockage après une demande initiée par l'utilisateur. Cette stratégie permet de réduire le plus possible la taille des données présentes dans l'agent de représentation en faisant en sorte que l'intersection entre l'ensemble des données stockées dans l'agent et celui des données stockées dans le terminal soit vide. L'ensemble des données stockées dans l'agent de représentation représente ainsi le complémentaire des données déjà reçues par l'utilisateur par rapport à l'ensemble des résultats qui lui ont été destinés.

Cette stratégie a aussi le mérite d'imposer à l'utilisateur de ne transférer au plus qu'une seule fois les résultats par la liaison sans fil. De plus, aucun ordre d'effacement de résultats dans l'espace mémoire du GC_{agt}n'est à envoyer du terminal vers l'agent de représentation dans le cas où ces résultats ont déjà été consultés par l'utilisateur. L'effacement est automatique après une lecture. L'utilisation de la liaison terminal - agent de représentation est ainsi sensiblement réduite.

Pour mettre en place cette stratégie, il faut configurer le répartiteur, pour qu 'il n'envoie pas un résultat vers l'espace de données du GC_{agt}lorsque le résultat est destiné à la file d'attente. De plus, lorsqu'une donnée est demandée au GC_{agt}, celui-ci la retire de son espace de données.

2. Economie de l'espace mémoire à l'intérieur du terminal

Si le terminal de l'utilisateur dispose d'une mémoire de stockage de résultats réduite, il est intéressant pour lui d'utiliser en partie son agent de représentation comme espace de stockage. Cependant l'appel à une donnée sur l'agent de représentation lui coûte plus «cher» en temps et en consommation dû à l'utilisation de la liaison terminal - agent de représentation.

Par conséquent, tout résultat envoyé automatiquement à l'utilisateur est à la fois placé par le répartiteur dans la file d'attente et dans l'espace de stockage géré par le GC_{agt}. De plus, tout résultat demandé par l'utilisateur à son agent de représentation est, non seulement retourné, mais reste dans l'agent de représentation. Une destruction de résultat est donc la conséquence d'une initiative de l'utilisateur.

Remarque 4.3 *Cette stratégie visant à rendre accessible une partie des résultats (choisie par l'utilisateur) à partir de l'agent de représentation permet à ce même utilisateur de consulter ces données même s'il se connecte à partir de terminaux différents. Au lieu qu'un terminal défini contienne tous les résultats, c'est l'agent de représentation, facteur commun de tous les terminaux, qui, étant accessible par l'utilisateur à partir de n'importe quel terminal, lui fournit les résultats demandés. La réponse à une requête sera coûteuse si le terminal utilise une liaison sans fil, beaucoup moins pénalisante dans le cas d'une liason à haut débit.*

C'est à l'utilisateur de choisir sa stratégie de sauvegarde des résultats en fonction de ses besoins et de ses préférences. Il doit pour cela configurer son agent de

représentation. La première stratégie lui permet d'avoir un agent de représentation plus petit et d'économiser la liaison terminal-agent. La seconde lui permet d'accéder à chaque donnée plusieurs fois ne lui imposant pas l'utilisation d'un même terminal continuellement. L'application stricte de ces stratégies, c'est-à-dire ou complètement l'une ou complètement l'autre, sera par la suite rendue plus souple en proposant de les appliquer à un niveau beaucoup plus fin, non plus au niveau de l'ensemble des résultats, mais aux résultats provenant d'une application identifiée voire même à des «parties» de résultats.

4.1.3 Description de résultats

4.1.3.1 Problématique

Dans le modèle qui vient d'être décrit, il est possible de définir si un objet-résultat de tâche est à envoyer automatiquement vers le teminal de l'utilisateur ou seulement après une requête de celui-ci. De la même façon, une tâche pourrait très bien pour un objet-résultat communiquer au GC_{agt}un degré de sécurité associé à cet objet (par exemple: *est-ce-que le résultat doit être communiqué à l'utilisateur chiffré ou en clair?*)... Cependant, le principal problème réside dans le fait que cette description du type de chargement ou de la politique de sécurité soit définie pour un objet-résultat complet alors que la taille d'un de ces objets peut être très importante et regrouper plusieurs éléments de nature différente. C'est pourquoi, un modèle de *décomposition* des objets-résultats permet non plus d'appliquer une description sur tout un objet-résultat mais plutôt, à un niveau plus fin, sur des *composants* de cet objet. Ceci permettrait à l'agent de représentation, à partir d'un résultat, de ne communiquer automatiquement qu'une vue partielle sur un résultat et, à partir de cette vue, l'utilisateur pourrait ou non demander au GC_{agt}à obtenir le complément ou seulement une partie du complément de ce résultat.

4.1.3.2 Utilisation de la notion d'*objet composite*

Définition des objets composites

Définition 4.3 *Un* objet composite, *d'après [BCG87], est un objet avec une hiérarchie d'*objets composants *exclusifs qui peuvent être eux-mêmes des objets composites (voir Figure 4.8).*

Exemple 4.1 *Une voiture peut être considérée comme un objet composite constitué de roues, de portières, d'un moteur,... Chacun de ces composants sont des parties exclusives de la voiture. Un moteur peut, lui aussi, être considéré comme objet composite constitué de soupapes, de culasse,.. Cependant, la voiture reste la racine d'une hiérarchie d'objets composites.*

Définition 4.4 *Un* objet dépendant, *toujours d'après [BCG87], est un objet dont l'existence dépend d'un autre.*

Remarque 4.4 *Un objet composant est un objet dépendant de l'objet composite.*

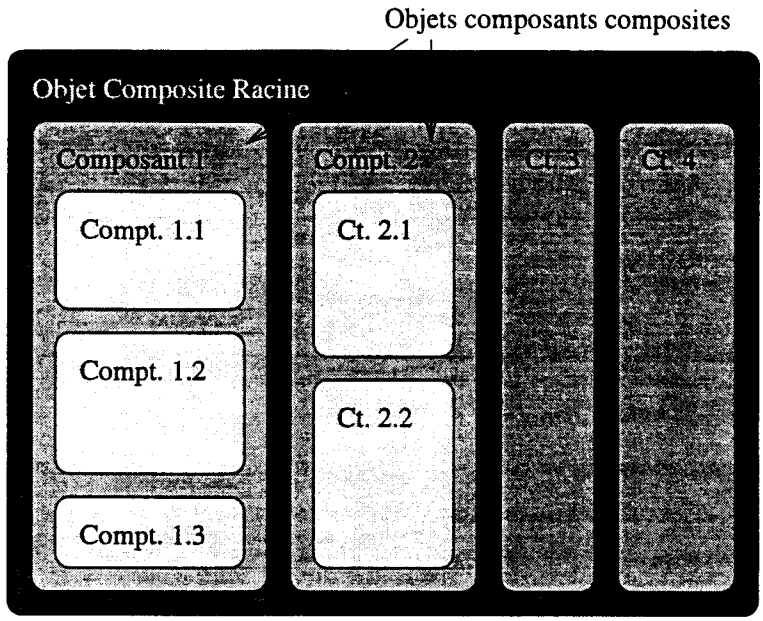


FIG. 4.8 - *Objet composite*

L'utilisation d'objets composites conduit à introduire deux types de références dans les systèmes orientés objets [ABD89]. Les références `estUnePartieDe` et les références générales. Les références `estUnePartieDe` ont la sémantique de liens composites entre objets composants et objets composites.

Le résultat sous forme d'objet composite La décomposition des objets-résultats conduit à utiliser tout naturellement les objets composites. Un objet-résultat est ainsi la racine d'une hiérarchie de composants.

Exemple 4.2 *Si l'objet-résultat est un objet* `Article`*. Dans ce cas, l'objet composite est l'*`Article` *lui-même. Les objets composants sont un objet* `Titre`*, un objet* `Résumé`*, des objets* `Section`*, un objet* `Conclusion`*. Une section est composée d'un objet* `Titre`*, d'objets* `Paragraphe`*, d'objets* `image`*, d'objets* `son`*, etc...*

Dans l'exemple précédent, la tâche reçoit une donnée `Article`, elle effectue un traitement sur cette donnée de façon à produire un objet composite résultat selon un format utilisable par le système de l'agent de représentation et le terminal. Lorsque l'objet composite `Article` est envoyé par une tâche au répartiteur, celui-ci va *décomposer* l'objet composite `Article`, envoyer automatiquement vers la file d'attente (donc destiné à l'utilisateur) une copie partielle de l'objet composite contenant des composants de petites tailles mais donnant un aperçu révélateur de l'article (par exemple : le titre de l'article et des sections, le résumé et la conclusion), une autre copie (au moins complémentaire dépendant des stratégies décrites dans la partie

4.1.2.3) de l'objet composite vers l'espace de données géré par le GC_{agt}.

Note 4.1 *Des exemples de décomposition de documents existent déjà. On peut citer, par exemple, SGML*[2] *[Sgml97], VRML*[3] *[Vrml96] ou HTML ou MIME*[4] *[BF93] [M93].*

4.1.3.3 objets de description associés à un objet-résultat

L'objet `DescriptionEnvoi` Afin de permettre au répartiteur de déterminer la nature des composants : *Automatique* à envoyer dès que possible vers le GC_{term}ou *ALaDemande* à stocker dans l'espace de données géré par le GC_{agt}, une tâche envoie un objet-résultat contenant un composant suplémentaire : un objet `DescriptionEnvoi` qui décrit la nature des autres composants de l'objet-résultat. Cet objet peut être tout simplement un tableau composé d'éléments `DescriptionComposant` contenant d'une part le nom **nom** d'un composant du résultat (ce champ peut prendre la forme d'une chaîne de caractères) et, d'autre part, le type d'envoi `TypeEnvoi` (de même pouvant être sous forme de chaîne de caractères) à effectuer sur ce composant correspondant à *Automatique* ou *ALaDemande*.

Note 4.2 *Comme un objet composite est la racine d'une hiérarchie de composants, il convient de définir une notation pour identifier un composant à partir d'une racine connue et de son chemin dans l'arborescence de composants et de sous-composants. On notera* `OC.Compt.SsCompt`*, le sous-composant* `SsCompt` *du composant* `Compt` *de l'objet composite* `OC`*. Plus généralement,* `C1.C2.C3....Cn` *détermine un chemin de composants en sous-composants partant de la racine l'objet composite* `C1` *jusqu'au composant* `Cn`*.*

La figure 4.9 montre à partir de l'exemple 4.2 le fonctionnement du répartiteur à partir d'un objet-résultat et d'un objet `DescriptionEnvoi`. A partir d'un objet-résultat, le répartiteur consulte le composant `DescriptionEnvoi` et crée deux objets-résultats *partiels* :

1. un objet-resultat destiné à être stocké dans l'agent de représentation. Il contient tous les composants devant être demandés pour être envoyés à l'utilisateur plus éventuellement le composant `DescriptionEnvoi`.

2. un objet-resultat destiné à l'utilisateur et donc à envoyer au GC_{term}. Il contient tous les composants devant être envoyés automatiquement à l'utilisateur avec leur valeur et tous les composants pouvant être obtenus seulement à la demande sans leur valeur. Le composant `DescriptionEnvoi` n'est pas obligatoirement envoyé. D'ailleurs, pour des raisons de coût de communication, il est préférable qu'il ne le soit pas.

L'utilisateur, qui a reçu et stocké l'objet-résultat répertorié, à la fois dans l'espace de données géré par le GC_{term}et celui géré par le GC_{agt}, `/Chemin/ObjetRésultat`,

2. SGML : Standard Generalized Markup Language
3. VRML : Virtual Reality Modeling Language
4. MIME : Multipurpose Internet Mail Extensions

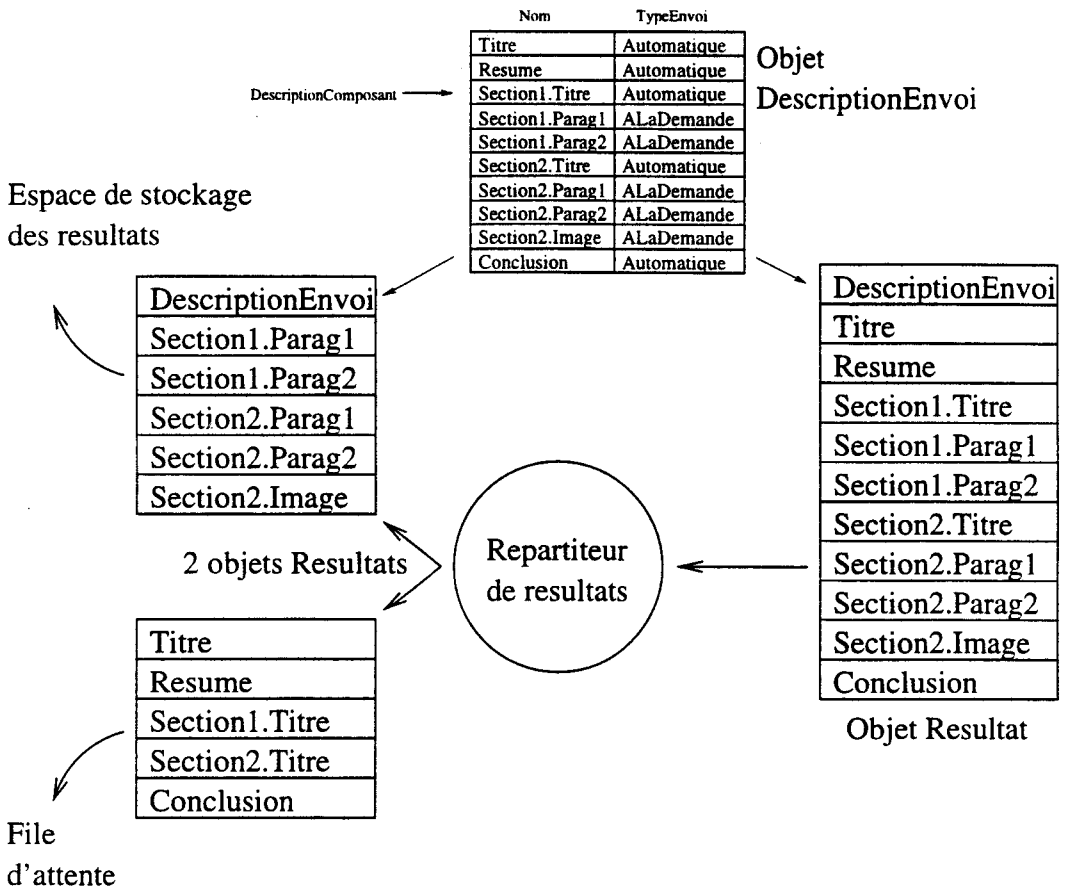


FIG. 4.9 - *Répartition des composants par le répartiteur*

connaît la liste des champs dont il a les données (ceux présents dans l'objet-résultat avec leur valeur), plus ceux dont il n'a pas encore les valeurs (les composants «vides»). Supposons que l'utilisateur désire obtenir la valeur du composant `Composant`, l'utilisateur peut le demander en invoquant, par l'intermédiaire du GC_{term}, la méthode `ObtenirRésultat(/Chemin/ObjetRésultat.Composant)` du GC_{agt}. Celle-ci retirera le composant `Composant` de l'objet `/Chemin/ObjetRésultat` présent dans l'espace de données utilisé par le GC_{agt}, puis retournera, de manière asynchrone par la file d'attente du GC_{agt}, un objet-résultat partiel contenant seulement le composant `Composant`. Le GC_{term} ajoutera alors la valeur de ce composant à l'objet `/Chemin/ObjetRésultat`.

Note 4.3 *Ce scénario, ainsi que la figure 4.9, présente des séquences d'actions réalisées dans le cas d'un stratégie du répartiteur favorisant l'économie de mémoire au niveau de l'agent (voir partie 4.1.2.3). Dans le cas d'une stratégie visant à réduire l'espace mémoire utilisé par le terminal, le répartiteur enverrai un nouvel objet-résultat avec tous ses composants dans l'espace de données géré par le GC_{agt} et ne détruirait pas de composant après une requête de composant venant du GC_{term}.*

Remarque 4.5 *IMAP*[5] *[C94] propose déjà, au niveau de serveurs de courriers électroniques, l'envoi d'une partie du courrier électronique pour réduire la quantité de données transférée. Les attachements du courrier sont codés par un codage MIME (déjà cités) correspondant à des composants (texte, image, son, video,...). Le mail peut être envoyé avec seulement des entêtes sur ces composants MINE. Le client, qui désire recevoir un composant, doit explicitement envoyer une requête vers le serveur de courriers électroniques.*

Les descriptions conditionnelles Jusqu'à présent, un composant de résultat devait être à envoyer ou non vers l'utilisateur par l'intermédiaire du gestionnaire de la file d'attente. L'idéal serait de pouvoir moduler, conditionner cet envoi de façon à tenir compte des préférences et de l'environnement de l'utilisateur. Le champ `TypeEnvoi` d'une `DescriptionElement` pourrait non seulement toujours prendre les valeurs `Automatique` ou `ALaDemande`, mais aussi des valeurs conditionnelles.

En ce qui concerne l'envoi automatique ou à la demande, pour un composant important en taille, il est possible de préciser que ce composant doit être envoyé automatiquement dans certaines conditions (si le débit est important et si le creneau horaire permet une tarification *heures creuses* dans le cas d'une liaison téléphonique) ou sinon à la demande.

Les opérations conditionnelles peuvent reprendre la syntaxe classique utilisée dans beaucoup de langages comme celle décrit dans la grammaire située en annexe A.1

Si on reprend l'exemple 4.2, on peut définir un objet `DescriptionElement` contenant des champs `TypeEnvoi` conditionnel comme décrit dans le tableau 4.2. Dans ce tableau, `Débit`, `Seuil1` et `Seuil2` sont des variables. La gestion des variables sera décrite dans la partie 4.1.3.3.

Autres objets de description D'autres objets de description des champs d'un objet-résultat peuvent être calqués sur le model de l'objet `DescriptionEnvoi`. La liste des objets de description suivante présente différents objets de description particulièrement intéressants mais ne se veut pas du tout exhaustive. D'autres objets de description décrivent une utilisation particulière de chaque composant d'un objet-résultat.

1. L'objet `DescriptionSecurite`

   Cet objet peut se présenter, comme pour l'objet `DescriptionEnvoi`, sous la forme d'un tableau d'éléments composés d'un `Nom` et d'un `TypeSécurité`. Il est un composant particulier d'un objet-résultat décrivant les conditions de sécurité pour chaque composant d'un objet-résultat et, en particulier, l'algorythme de chiffrement (DES, RSA,...) et la clé de chiffrement à partir desquels les composants seront chiffrés.

   Le répartiteur se charge du chiffrement des composants.

5. IMAP : Internet Message Access Protocol

Nom	TypeEnvoi
Titre	Automatique
Resume	Automatique
Section1.Titre	Automatique
Section1.Parag1	SI Débit >= SeuilDébit ALORS Automatique SINON ALaDemande FIN_SI
Section1.Parag2	SI Débit >= SeuilDébit ALORS Automatique SINON ALaDemande FIN_SI
Section2.Titre	Automatique
Section2.Parag1	SI Débit >= SeuilDébit ALORS Automatique SINON ALaDemande FIN_SI
Section2.Parag2	SI Débit >= SeuilDébit ALORS Automatique SINON ALaDemande FIN_SI
Section2.Image	SI Débit >= SeuilDébit ET Ecran == 'Couleur' ALORS Automatique SINON ALaDemande FIN_SI
Conclusion	Automatique

TAB. 4.2 - *Exemple de description conditionnelle*

La syntaxe d'un élément TypeSécurité peut être définie par la grammaire présentée en annexe A.2.

Remarque 4.6 *Un objet* `DescriptionCompression` *peut s'inpirer du modèle* `DescriptionSecurite`.

Le chiffrement/déchiffrement sera plus détaillé dans le chapitre 4.2.

2. L'objet `DescriptionPermanence`

   La structure de cet objet est un tableau d'éléments composés d'un `Nom` et d'un `TypePermanence`. L'objet `DescriptionPermanence` est un composant particulier d'un objet-résultat qui permet de définir si un composant est permanent dans l'agent ou non, c'est-à-dire de moduler au niveau du composant d'un objet-résultat les stratégies du répartiteur décrites dans la partie 4.1.2.3.

   La syntaxe d'un élément `TypePermanence` peut être définie sous la forme présentée en annexe A.3

3. L'objet `DescriptionMigration`

   Cette objet associé à un objet-résultat permet de définir si ce résultat doit se déplacer avec l'agent de représentation ou est destiné à rester sur la RIF (voir partie 3.1). Ainsi des résultats concernant un service local au lieu de résidence de l'utilisateur peuvent rester sur la RIF conformément à la `DescriptionMigration` envoyée par la tâche. La syntaxe d'un tel élément est semblable aux précédentes descriptions.

La gestion des variables utilisées pour les conditions Pour pouvoir obtenir des descriptions conditionnelles, il faut être capable d'écrire des conditions et, par conséquent, d'utiliser des données qui paramètrent, décrivent l'environnement ou les préférences de l'utilisateur. Ces variables sont mises à jour par le terminal lorsque l'utilisateur est connecté.

Ces variables doivent être présentes dans l'agent de représentation et utilisables par les modules du GC_{agt}tels que l'espace de stockage des données, la file d'attente et le répartiteur. Pour cela, un gestionnaire de variables à partir d'un nom de variable est utilisé dans le GC_{agt}ou communique avec lui. Celui-ci est utilisable en lecture pour les modules du GC_{agt}, mais utilisable en écriture à la fois par ces modules, mais aussi par le GC_{term}.

La structure de l'objet `GestionnaireVariables` est principalement composée d'une collection d'éléments contenant le nom de la variable et un objet correspondant à la valeur.

Le gestionnaire de variables peut être invoqué par deux méthodes :

1. `AjoûterVariable(ChaîneDeCaractères Nom, Object Valeur)`

   Cette méthode ajoute un champ composé de `Nom` et `Valeur` dans la collection, mais retourne éventuellement l'exception `DejaPresente` si la variable existe déjà.

2. `ModifierVariable(ChaîneDeCaractères Nom, Object Valeur)`

   Cette méthode écrase la valeur de la variable `nom` par la valeur `Valeur`, mais retourne éventuellement l'exception `VariableAbsente` si la variable n'existe pas dans la collection.

3. `LireVariable(ChaîneDeCaractères Nom)`

   Cette méthode retourne la valeur de la variable `nom`, mais retourne éventuellement l'exception `VariableAbsente` si la variable n'existe pas dans la collection.

4. `SupprimerVariable(ChaîneDeCaractères Nom)`

   Cette méthode supprime le champ correspondant à la variable `nom` de la collection, mais retourne éventuellement l'exception `VariableAbsente` si la variable n'existe pas dans la collection.

Les deux paragraphes suivants présentent les deux types de variables qui peuvent être identifiées.

Les *variables d'environnement* La valeur des variables d'environnement est fixée par le système. Elle concerne en particulier l'environnement sur lequel travail l'utilisateur. Ces variables seront souvent mises-à-jour lors de nouvelles connections de l'utilisateur avec son agent de représentation. A ce moment-là, les variables d'environnement du terminal de l'utilisateur seront envoyées au gestionnaire de variables par le GC_{agt}.

Les principales variables d'environnement sont celles décrites dans le tableau 4.3.

Nom	Type	Signification
`Date`	nombre	Date courante
`heure`	nombre	Heure courante
`Débit`	nombre	Débit de la liaison entre le terminal et l'agent
`TypeLiaison`	Chaîne de caractères	Type de la liaison : `'Cellulaire'`, `Téléphonique`, `Cablée`
`CoûtLiason`	nombre	Coût horaire de l'utilisation de la liasion
`RésolutionEcran`	(nombre,nombre)	Nombre de pixels en largeur et en hauteur de l'écran utilisé
`CouleurEcran`	Chaîne de caractères	Type d'écran utilisé : `'NoirBlanc'`, `'Couleur'`

TAB. 4.3 - *Les variables d'environnement*

Les *variables personnelles* Ces variables peuvent être crées par l'utilisateur ou une application de l'utilisateur. La valeur de ces variables est affectée par l'utilisateur via le GC_{term}en fonction notamment de ses préférences.

Le tableau 4.4 présente quelques exemples de variables possibles. Cependant une application de l'utilisateur peut envoyer une tâche du terminal vers l'agent de représentation qui retournera des résultats «manipulés» par le GC_{agt}en fonction des variables créées par cette application dans l'agent de représentation.

Nom	Type	Signification
`SeuilDébit`	nombre	Borne inférieure de débit de communication entre le terminal et l'agent
`PlafondDébit`	nombre	Borne supérieure de débit de communication entre le terminal et l'agent
`SeuilCoût`	nombre	Seuil de coût de communication
`PlafondCoût`	nombre	Plafond de coût de communication
`RésolutionEcranMinimale`	(nombre,nombre)	Résolution minimale de l'écran du terminal de l'utilisateur
`CléRSA`	nombre	Clé RSA

TAB. 4.4 - *Les variables personnelles*

4.2 Apports de la carte à microprocesseur

La carte à microprocesseur assure des fonctions de stockage et de traitement comme cela a été présenté dans la partie 1.4.2. Son utilisation dans un système basé sur les agents de représentation intervient dans plusieurs domaines grâce à ses capacités de mémoire et de traitement (voir partie 1.4) [CLT97]. D'abord, la carte à microprocesseur permet l'identification et l'authentification de l'utilisateur vis-à-vis de l'agent de représentation. Elle apporte, de plus, des paramètres de personnalisation d'applications et d'accès à l'agent de représentation à un terminal anonyme. Elle stocke des données relatives à l'utilisateur ou à ses applications. Elle apporte enfin des outils de sécurité pour chiffrer/déchiffrer des données.

L'utilisation d'une représentation personnelle extérieure nécessite, en effet, une prise en compte de la sécurité :

- au niveau de la confidentialité des données
- au niveau de l'identification/authentification pour assurer un accès protégé l'agent de représentation.

Le fait d'utiliser des terminaux anonymes impose une phase de personnalisation du terminal en fonction de l'utilisateur. La carte à microprocesseur est un support très utilisé pour ce genre de problème (voir partie 1.4). De plus, la carte à microprocesseur est un support personnel qui permet à un individu un accès à son agent de représentation et aux résultats contenus à l'intérieur à partir de terminaux anonymes.

Dans ce chapitre, nous nous proposons de montrer l'intérêt de l'utilisation de la carte à microprocesseur dans un système à base d'agents de représentation et son intégration dans ce système [CLT96].

4.2.1 Sécurité apportée par la carte

La carte à microprocesseur est capable de calcul et de renfermer des secrets tels que des clés. C'est donc un support adéquat pour l'identification/authentification et le chiffrement/déchiffrement de données.

4.2.1.1 Double identification/authentification

L'utilisateur dispose d'un agent de représentation qui lui est dédié et qui lui appartient. Cet agent de représentation se trouve sur un site différent. C'est pourquoi, l'utilisateur doit recevoir une preuve que l'interlocuteur est bien son agent de représentation et non quelqu'un qui désire se faire passer pour lui. Inversement, l'agent de représentation, qui peut être amené à envoyer des données confidentielles à l'utilisateur, doit recevoir une preuve de son identité. C'est pourquoi, un mécanisme de double identification/authentification doit être mis en place entre l'utilisateur et son agent de représentation [CT95].

La carte à microprocesseur permet une identification/authentification du porteur. Pour cela, l'utilisateur doit donc disposer d'une clé privée de chiffrement k_u [6]

6. k_u : Clé servant à l'authentification de l'utilisateur

présente dans sa carte à microprocesseur et d'une clé de déchiffrement k_u^{-1} publique vérifiant la propriété suivante :

$$\forall m, k_u(k_u^{-1}(m)) = k_u^{-1}(k_u(m)) = m$$

De même, l'agent de représentation dispose d'une clé de chiffrement k_{ar} [7] et d'une clé de déchiffrement k_{ar}^{-1} publique.

Remarque 4.7 *L'algorithme utilisé pour le calcul des clés k et k^{-1} peut être le RSA [Rsa78]. Il permet, en effet, la génération de deux clés différentes dont l'une permet le chiffrement, l'autre le déchiffrement.*

L'authentification de l'agent de représentation (voir figure 4.10) vis-à-vis de l'utilisateur peut s'effectuer en trois étapes [S94] :

1. La carte à microprocesseur de l'utilisateur envoie un défi *random*, nombre aléatoire, à l'agent de représentation.
2. L'agent de représentation retourne $k_{ar}(random)$.
3. La carte calcule $k_{ar}^{-1}(k_{ar}(random))$ et retrouve donc la valeur *random*. Comme k_{ar} est une clé privée possédée seulement par l'agent de représentation, ce dernier doit recevoir la preuve de son identité.

L'authentification de l'utilisateur vis-à-vis de agent de représentation reprend les mêmes étapes que dans le cas précédent.

Note 4.4 *La carte SIM permet déjà un identification/authentification d'un utilisateur vis-à-vis du réseau GSM par des algorithmes basés sur le RSA.*

Les clés publiques k_u^{-1} et k_{ar}^{-1} peuvent être disponibles sur des serveurs ou respectivement enregistrées dans l'agent de représentation et dans la carte à microprocesseur.

4.2.1.2 Confidentialité des Communications avec l'agent de représentation

Les données circulent entre l'utilisateur et son agent de représentation. Comme une grande partie de ces données risque d'être personnelle et sensible pour l'utilisateur, il est donc nécessaire d'offrir des mécanismes permettant la confidentialité de ces données.

L'utilisateur doit donc disposer, dans sa carte à microprocesseur, d'une clé privée de déchiffrement C_u^{-1} [8]. L'agent de représentation posséde, de la même façon, une clé privée de déchiffrement C_{ar}^{-1} [9]. Aux clés de déchiffrement C_u^{-1} et C_{ar}^{-1} sont respectivement associées les clés de chiffrement C_u et C_{ar}.

7. k_{ar} : Clé servant à l'authentification de l'agent de représentation
8. C_u : Clé de chiffrement de donnés de l'utilisateur
9. C_{ar} : Clé de chiffrement de donnés de l'agent de représentation

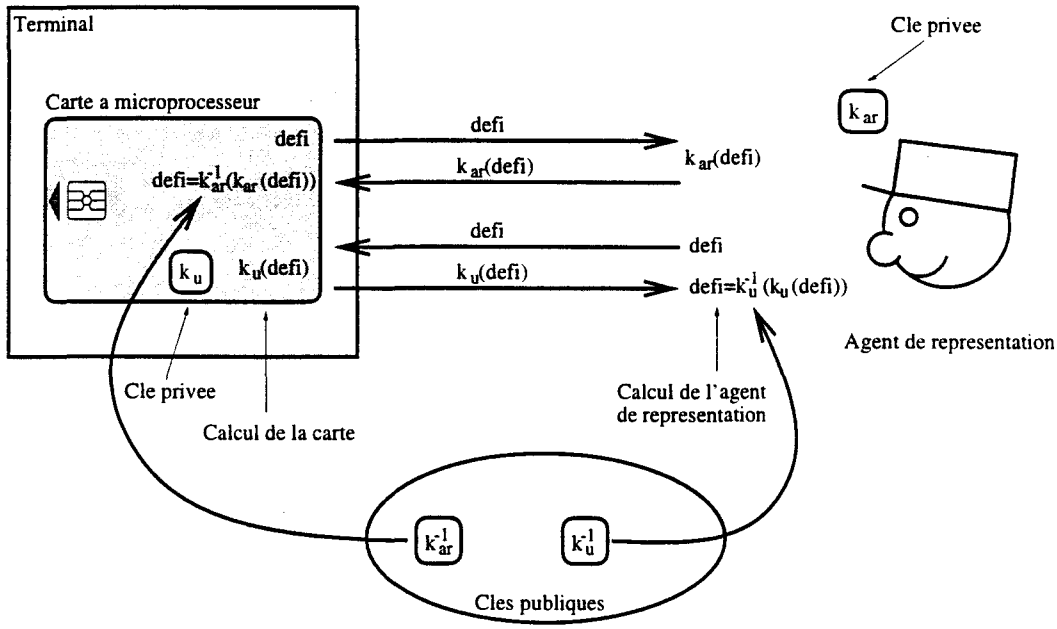


FIG. 4.10 - *Double authentification entre la carte et l'agent de représentation*

L'utilisateur envoie donc des messages confidentielles après chiffrement de la sa carte avec la clé C_{ar}. Seul l'agent de représentation possède la clé de déchiffrement associée C_{ar}^{-1}, il est donc le seul à pouvoir lire ce message.

Inversement, l'agent de représentation chiffre les messages confidentiels destinés à l'utilisateur avec la clé de ciffrement C_u. La carte à microprocesseur, détenant seule la clé C_u^{-1}, est la seule capable de déchiffrer ce message pour l'utilisateur.

Remarque 4.8 *Les clés de chiffrement C_u et C_{ar} peuvent être publiques mais ce n'est pas nécessaire.*

Remarque 4.9 *Les couples de clés (C_u, C_u^{-1}) et (C_{ar}, C_{ar}^{-1}) peuvent être les mêmes à partir du moment ou la clé de déchiffrement n'est seulement connue de la carte à microprocesseur et de l'agent de représentation. Cependant, ce n'est pas obligatoire. La carte à microprocesseur étant un support beaucoup plus «sûr» que l'agent de représentation accueilli par un site accepteur, partager un secret entre deux entités entraine une diminution du niveau de sécurité global (carte+agent) au niveau du plus faible, c'est-à-dire l'agent de représentation.*

4.2.2 Stockage des données dans la carte

La carte à microprocesseur a un rôle très important dans le contexte d'utilisation de différents terminaux anonymes. Grâce à la mémoire contenue à l'intérieur, elle peut en effet stocker des données qui sont transportées de terminaux en terminaux.

4.2.2.1 Personnalisation de l'environnement de l'utilisateur

La personnalisation d'un terminal permet à l'utilisateur de transformer un terminal anonyme en terminal lui étant dédié. La carte à microprocesseur et l'agent de représentation sont les éléments informatiques qui fournissent à ce terminal cette personnalisation.

La carte à microprocesseur apporte les moyens de communiquer avec l'agent de représentation. En effet, celle-ci peut contenir des références, des adresses vers d'autres objets en particulier, dans un système à base d'agents de représentation, l'adresse de l'agent associé (voir chapitre 3.1.3). Ainsi l'établissement d'une communication entre l'agent de représentation et l'utilisateur est plus rapide, il est inutile de demander la résolution d'une adresse pour retrouver l'agent de représentation (voir partie 3.1). L'agent de représentation migre pour se rapprocher de l'utilisateur. Pour cela, ce dernier doit avoir une connexion pour fournir à l'agent de représentation son adresse courante. Lorsque l'agent de reprentation migre, il peut donc, grâce à la connexion de l'utilisateur, fournir sa nouvelle adresse à la carte.

4.2.2.2 Support de sauvegarde de configuration

La carte à microprocesseur a des capacités de stockage de données et en particulier de données individuelles. La carte sert déjà de support pour des données relatives à la sécurité comme les clés servant à l'authentification et la confidentialité des données de l'utilisateur. Elle peut aussi contenir des données qui sont fournies lors de l'insertion de la carte au système et aux applications du terminal [L94].

Les données apportées au terminal doivent rendre moins contraignant pour un usager le fait de se connecter souvent sur des machines différentes. Pour éviter que l'utilisateur perde du temps à reconfigurer son système et donc pour qu'il ne soit pas perturbé par l'interface du système, la carte apporte des paramètres d'environnement graphique utilisés par le terminal. Même si la résolution n'est pas toujours la même, le terminal donne à l'utilisateur une configuration la plus proche possible de ses préférences.

Au niveau applicatif, la carte peut contenir des données de configuration pour des applications. Par exemple, elle peut disposer d'un ensemble de références URL[10] [B94] pour un navigateur *web* ou l'identité de l'utilisateur pour un gestionnaire de courrier électronique.

4.2.3 Intégration dans le système

Cette partie décrit les éléments du modèle présenté dans les chapitres précédents qui peuvent être transportés dans la carte à microprocesseur de l'utilisateur.

4.2.3.1 Elément de la structure de données du terminal

La carte à microprocesseur peut être un support jouant le rôle de l'espace de stockage de données du terminal (voir partie 4.1.1). Ainsi cet espace est accessible

10. URL : Uniform Resource Locator

de n'importe quel terminal puisqu'il est apporté par la carte au terminal et est protégé. Un accès aux données n'est possible qu'après authentification de l'utilisateur. Cependant, les cartes à microprocesseur ne disposent pas encore d'une mémoire assez grande pour pouvoir stocker une grande quantité de données. Lorsque les cartes disposeront d'une capacité mémoire suffisamment importante, tout l'espace de stockage de données pourra être à l'intérieur, mais actuellement il faut faire un choix sur les données placées à l'intérieur de la carte à microprocesseur.

L'approche la plus réaliste actuellement avec une carte à microprocesseur est d'y stocker uniquement la structure hiérarchique des données. Les données restent ou sur le terminal s'il appartient à l'utilisateur, ou doivent être demandées à l'agent de représentation. Chaque élément de la structure hiérarchique contient, en plus d'un identifiant dans la structure hiérarchique, des informations telles que :

1. donnée déjà reçue par l'utilisateur et disponible sur l'agent de représentation
2. donnée déjà reçue par l'utilisateur mais plus disponible sur l'agent de représentation
3. donnée jamais reçue par l'utilisateur et disponible sur l'agent de représentation

Remarque 4.10 *Le deuxième cas est le plus dommageable pour l'utilisateur. C'est pourquoi, s'il a l'habitude de changer régulièrement de terminal, il peut choisir une stratégie de permanence des données sur l'agent de représentation (voir partie 4.1.2.3), c'est-à-dire que les résultats envoyés vers l'utilisateur par l'agent de représentation reste dans l'agent jusqu'à un ordre d'effacement explicite de l'utilisateur.*

Le terminal peut servir de mémoire tampon, qui lors d'une utilisation par un usager, permet d'y stocker provisoirement des résultats.

Quant aux données présentes dans l'agent de représentation, la carte à microprocesseur détient les identifiants permettant d'accéder sur l'agentde représentation.

Lorsqu'un nouveau résultat arrive dans le terminal, la carte est mise à jour. Lorsqu'un résultat est retiré de l'espace de stockage du terminal, la carte doit aussi prendre en compte le changement.

4.2.3.2 Structures de cartes appropriées

Toute carte est adaptée pour y enregistrer la structure de données. En effet, toutes les cartes permettent la sauvegarde de suite binaire. Un objet comme cette structure de données peut être convertie en suite binaire comme cela se fait grâce, par exemple, au mécanisme de sérialisation en Java (voir partie 1.3.3.2). La structure est ensuite stockée ainsi à l'intérieur de la carte.

Une carte de données organisées sous forme de fichiers et répertoires (voir partie 1.4.3.1), les cartes à microprocesseur les plus courantes, convient tout à fait grâce à sa structure hiérarchique comparable à celle de la structure de données à enregistrer à l'intérieur (voir partie 4.1.2.2).

Si cette structure est écrite en Java, elle peut être plus facilement chargée à l'intérieur d'une JavaCard. De même que les modifications seront plus faciles à effectuer

puisqu'il est possible d'utiliser des méthodes appropriées telles que `AjouterDonnée`, `DétruireDonnée`, `CréerRépertoire`, `DétruireRépertoire`.

4.3 Conclusion

Le modèle présenté permet aux tâches présentes sur l'agent de représentation d'envoyer des résultats vers l'utilisateur. Cependant, ces résultats vont subir des traitements avant d'être envoyé. Ce qui permet aux applications d'obtenir des résultats adaptés aux caractéristiques matérielles du terminal utilisé ou de préférences. Tous ces traitements sont effectués par l'agent de représentation, et non pas par les tâches. Ces dernières se contentent de donner des recommandations que l'agent de représentation applique.

La carte à microprocesseur est un élément important dans un système à base d'agents de représentation. Même si sa capacité mémoire apparait encore limitée, elle permet aujourd'hui de fournir au terminal un historique des données déjà envoyées vers l'utilisateur et de celles contenues dans l'agent de représentation. Elle permet facilement le stockage de données personnalisant l'environnement et le système aux préférences de l'utilisateur. Enfin, elle apporte beaucoup au niveau de la sécurité en fournissant confidentialité, identification et authentification dans les communications entre l'utilisateur et son agent de représentation. La JavaCard est un support adéquat pour la création d'un prototype car elle permet d'intégrer plus facilement la carte dans un environnement constitué d'objets.

Chapitre 5

Expérimentation du modèle

Introduction

Ce chapitre a pour but de présenter le travail pratique réalisé. Une maquette a été construite de manière à expérimenter et valider le modèle proposé. Nous présentons dans ce chapitre les objectifs de cette maquette, les moyens utilisés et l'architecture implantée.

5.1 Objectifs et moyens de l'implantation

Cette section décrit les objectifs de l'implantation informatique réalisée à partir du modèle qui a été proposé dans les chapitres précédents. Le but de cette implantation est l'expérimentation et la validation du modèle. Nous expliquons ensuite nos choix concernant l'environnement logiciel (langage, interface et agents mobiles). Enfin, nous présentons l'environnement matériel utilisé et les conséquences sur le développement d'une simulation d'interface de terminal mobile.

5.1.1 Buts de l'implantion

5.1.1.1 Objectifs

La réalisation d'une maquette porte tout son intérêt dans deux situations :

1. l'expérimentation

2. la validation

L'expérimentation permet de mettre à l'épreuve une théorie et de trouver un modèle adéquat pour l'appliquer.

La validation est la phase suivante qui permet de prouver le bon fonctionnement d'un modèle.

L'implantation informatique qui a été réalisée s'inscrit dans les deux ojectifs. La programmation a permis de mettre en place un modèle. Il existait déjà avant le début de la phase de programmation, mais il a pu être légèrement modifié, amélioré, complété jusqu'à ce qu'un modèle définitif soit validé.

L'objetif principal n'est pas de créer des agents mobiles, d'utiliser des cartes à microprocesseur, de charger du code dans un autre code. Toutes ces implémentations existent déjà. L'objectif est plutôt d'expérimenter et valider une structure de données, un protocole de communication, des communications entre plusieurs composants d'après le modèle présenté dans les chapitres précédents.

5.1.1.2 Hors Objectifs

L'implantation réalisée n'est pas un prototype, mais plutôt une maquette. En effet, le modèle qui a été présenté est destiné à un réseau d'entreprise ou global. Par conséquent, à partir des moyens du laboratoire (voir partie 5.1.3.1), il n'est possible que de réaliser une maquette utilisée dans une simulation de réseau global. De plus, il ne s'agit que d'expérimentation et de validation d'un modèle. Cependant, les spécifications, les interfaces des différents objets créés peuvent être réutilisées.

5.1.2 Utilisation du langage Java

Le choix d'un langage pour créer une maquette est un élément très important. Quel langage faut-il utiliser pour programmer une maquette suffisante pour l'expérimentation en réduisant le plus possible le temps de développement. Java est apparu comme le meilleur produit. Il offre, d'une part, un très grand nombre d'API disponibles en standard dans le langage. D'autre part, de nombreuses plate-formes basées sur ce langage, en particulier, celles à agents mobiles sont présentes sur le marché.

5.1.2.1 La délégation de tâches

Avantages de Java Le langage Java, comme précédemment énoncé dans la partie 1.3.3.2, est doté de nombreux avantages, beaucoup de critères plébicitent ce langage pour la délégation de tâches vers l'agent de représentation et donc la programmation à la fois des CPDT et CID (voir partie 3.2.2.1).

Java est devenu un langage très répandu dans le domaine de l'informatique et le nombre de nouveaux utilisateurs est en constante augmentation. De plus, il est relativement simple à utiliser pour un programmeur. Par conséquent, écrire une tâche en Java est beaucoup plus facilement accepté par les programmeurs qu'avec la plupart des autres langages et s'intègre plus facilement dans les systèmes actuels.

D'où la nécessité de pouvoir écrire une tâche qui puisse être exécuté sur toute station. De plus, l'agent peut migrer vers d'autres sites disposant de caractéristiques

matérielles et d'un système d'exploitation différent du site d'origine. Java est un langage portable et interprété et donc permet l'envoi d'un même code correspondant à une tâche ou à un agent vers n'importe quelle station hébergeant l'agent de représentation.

Java permet de faire circuler très facilement des objets à travers un réseau. Par conséquent, L'envoi de tâches personnelles écrites en Java vers un agent de représentation, lui-même écrit en Java, ne pose pas de difficultés majeures. De même, l'importation de CID est aussi aisée.

L'envoi de tâches du terminal de l'utilisateur vers le réseau se fait avant tout dans le but de faciliter les communications de l'utilisateur avec l'extérieur. Par conséquent, une tâche doit pouvoir facilement utiliser des outils de communication performants. Java, dans sa version de base, a l'avantage de disposer d'API permettant de facilement faire communiquer des clients-serveurs et d'utiliser les protocoles de communication d'Internet. Avec les RMI, l'intégration dans CORBA ou les possibilités de chargement dynamique de classes, les outils de communications sont nombreux en Java.

Limites de Java Java est un langage polyvalent, ce qui ne veut pas dire qu'il soit le plus adapté à un système qui répond aux spécifications des agents de représentation. C'est pourquoi, certaines limites peuvent apparaître.

L'évolution très rapide de Java rend obsolète des API spécifiques ou les rend inutilisables à cause d'API remplissant le même cahier des charges mais, elles, standardisées. On peut citer les mécanismes de sérialisations qui ont été créés notamment pour les premières plate-formes pour agents mobiles Java avant qu'ils ne soient intégrés dans le JDK 1.1.

5.1.2.2 Les agents mobiles

Le choix du langage d'écriture de l'agent de représentation était un choix relativement facile. La plupart des nouveaux produits permettant la création d'agents mobiles sont aujourd'hui basés sur le langage Java. De plus, ils sont librement et gratuitement disponibles sur Internet.

Nous avons choisis les Aglets d'IBM comme plate-forme d'agents mobiles pour plusieurs raisons. D'abord, lorsque le choix à dû être fait, les Aglets étaient le seul produit disponible sur le marché. De plus, IBM prévoyait et prévoie toujours de proposer ses solutions vis-à-vis de l'OMG. Enfin, la documentation est suffisamment importante, disponible et gratuite. IBM possède aussi une liste de diffusion dans laquelle il est possible de recevoir des réponses aux éventuelles questions sur le produit.

5.1.2.3 L'interface utilisateur

L'interface utilisateur ne constitue pas un objectif dans le projet et ne fait pas partie du modèle présenté dans ce document. Nous avons choisi d'utiliser simplement un navigateur *web* pour simuler par des pages HTML et des applets l'interface homme-machine du terminal. Une description de cette interface est présentée en 5.1.3.2.

5.1.3 Les moyens matériels utilisés

5.1.3.1 Les moyens disponibles

L'idéal pour expérimenter et valider une maquette s'appuyant sur les agent de représentation est d'avoir un réseau de grande échelle composé d'un ensemble de terminaux. Ces derniers peuvent avoir des caractéristiques matérielles différentes et communiquer par divers types de liaison. Ils disposent aussi de lecteur de cartes à microprocesseur.

Comme il ne s'agissait que d'implanter une maquette, nous avons pu mettre à profit le matériel disponible à savoir un réseau éthernet sur lequel sont reliées cinq stations Sun disposant du système d'exploitation Solaris 2.4 ou 2.5. Elles servent à simuler chacune une zone géographique.

Si un utisateur est connecté sur la machine `M`, cela signifie qu'il se trouve dans la zone `M`. Par conséquent, l'agent de représentation doit se trouver ou migrer dans la zone `M`.

5.1.3.2 Simulation d'interface de terminal

Ne disposant pas de réseau pour mobiles à échelle suffisamment grande, nous avons choisi de créer un terminal virtuel avec une liaison intermittente. Ce terminal se présente sous la forme d'une fenêtre d'un navigateur *web*. Les données affichées sont des pages HTML et des Applets Java. L'écran du terminal est simulé par une fenêtre du navigateur. Des applets permettent de simuler des déplacements dans différentes zones géographiques et d'afficher des résultats automatiquement dans la fenêtre du navigateur.

Le terminal est composé de trois parties comme illustré dans la figure 5.1 :

1. Le GC_{term}, bien que présent dans le terminal, ne fait pas partie de l'interface du terminal, mais est un élément de la maquette du modéle. Il sera détaillé dans la partie 5.2.

2. L'interface homme-machine permet de consulter des données déjà reçues de l'agent de représentation ou de les lui demander. Cette interface est simulée par la fenêtre d'un navigateur *web* (voir copie d'écran 5.2) se composant de :

   - une applet proposant un bouton permettant de connecter/déconnecter le terminal, un menu permettant de simuler des déplacements en sélectionnant différentes zones géographiques correspondant aux machines disponibles sur le réseau.
   - une autre applet permettant d'afficher des messages provenant du GC_{term}. L'état de connexion ou la zone géographique dans laquelle l'utilisateur s'est déplacé est ainsi indiquée dans la fenêtre d'exécution de cette applet.
   - une zone de la fenêtre simulant l'écran du terminal. Elle permet d'afficher, à partir de requêtes HTTP, des pages HTML. Elles correspondent à une représentation des composants d'un résultat contenu dans le GC_{term}. Cette zone de la fenêtre est aussi utilisée pour demander un composant contenu dans l'agent de représentation. Les pages obtenues permettent

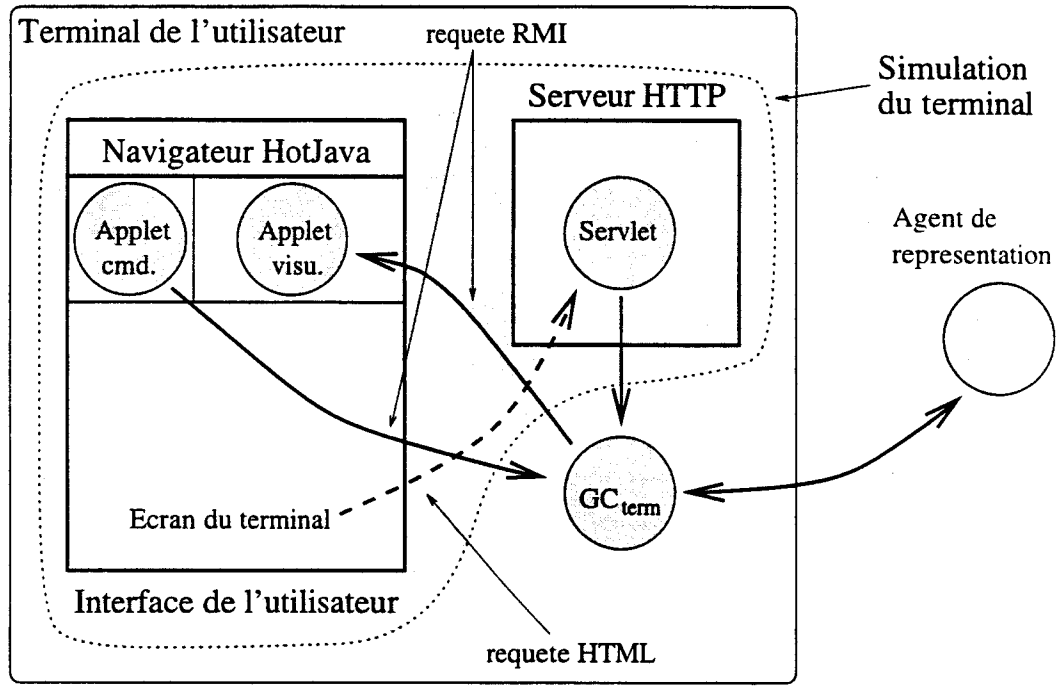


FIG. 5.1 - *simulation d'un terminal utilisateur*

de se déplacer dans la hiérarchie de résultats contenue à la fois dans le GC_{term}et dans le GC_{agt}(voir partie 4.1.2.2).

3. Un serveur de données HTML fournissant des résultats provenant du GC_{term}sur requête HTTP

Remarque 5.1 *Le navigateur* web *choisi pour consulter des pages HTML et charger des applets est HotJava. Il est en effet le seul à accepter des applets écrites et compilées par le JDK 1.1, ce qui leur permet d'utiliser les communications basées sur les RMI.*

Remarque 5.2 *Le serveur HTTP est nécessaire pour convertir des requêtes HTML en invocation RMI. Une servlet associée au serveur sert d'interface entre le navigateur et le* GC_{term}.

5.2 Implantation

5.2.1 Description générale du système implanté

L'implantation a permis la réalisation d'une maquette s'appuyant sur le modèle présenté tout au long de ce document. Le but est de simuler la mobilité d'un utilisateur et de lui fournir un agent de représentation lui permettant la délégation de

tâches. Le résultat de ces tâches est envoyé vers l'utilisateur en fonction de son état de connexion sur le réseau.

Les tâches envoyées ont été choisies en fonction de l'intérêt qu'elle représentent dans la délégation.

- Une première tâche permet de tester un modèle client-agent-serveur. Il s'agit, en fait, d'une tâche vers laquelle l'utilisateur demande à partir d'une URL la donnée correspondante. La requête est traitée de manière asynchrone par l'agent de représentation.

- Une seconde tâche permet d'expérimenter le modèle de tâches abonnement. La tâche est utilisée pour envoyer régulièrement vers l'utilisateur les nouveaux messages électroniques qui lui sont destinés.

Afin de réaliser une implantation qui prenne en compte l'envoi par composant en automatique ou a la demande, l'application de courrier électronique transmet les messages partiellement. Les composants d'un message tels l'expéditeur, la date, le titre sont automatiquement envoyés vers l'utilisateur, alors que le contenu du message est transmis seulement sur demande.

Ces deux applications retournent des données au format HTML. Par conséquent, l'interface de lecture de ces données peut être un navigateur *web* comme décrit dans 5.1.3.2.

Plusieurs éléments participent à l'architecture générale :

- le terminal contenant les applications
- l'agent de représentation
- les tâches présentes dans l'agent de représentation
- les sites accepteurs

5.2.2 Les API

L'implantation réalisées s'appuie sur des API Java déjà définies en standard ou développées pour la création d'objets particuliers tels que les agents mobiles. Au dessus de ces API, nous avons développées nos propres API Java qui implantent le modèle proposé.

5.2.2.1 Le terminal de l'utilisateur

Le terminal de l'utilisateur repose sur une machine virtuelle Java sur laquelle sont utilisées les API du JDK 1.1. Ces API permettent l'interopérabilité entre les objets Java sur le terminal ou avec ceux de l'extérieur.

Le GC_{term}, développé en Java, est une API que nous avons ajoutée. Il permet pour les applications :

1. le stockage de résultats des tâches provenant de l'agent de représentation

2. l'obtention de résultats contenus dans le terminal par invocation RMI

3. la demande de résultats à l'agent de représentation lorsqu'ils n'ont pas encore été envoyés au terminal. Les requêtes pour obtenir un résultat se font par invocation RMI.

Les applications invoquent la méthode `ObtenirResultat` du GC_{term} avec en paramètre un identifiant du résultat désiré. Le GC_{term} retourne ce résultat vers l'application. Le mécanisme d'obtention d'un résultat sera plus détaillé dans les prochains paragraphes.

5.2.2.2 L'agent de représentation

Dans l'implantation qui a été réalisée, l'agent de représentation accueille à la fois des tâches (voir partie 3.2) et gère les résultats à envoyer au terminal en fonction de recommandations des tâches sur ces résultats (voir partie 4.1).

La plate-forme choisie est celle de l'Aglet Workbench présentée en 1.3.3.3. Cette plate-forme utilise les API du JDK 1.1. L'agent de représentation est un classe qui a été définie en héritant de la classe `Aglet`. Les sites accepteurs d'agents de représentation sont des *aglets hosts* présents sur toutes les stations constituant le réseau.

5.2.3 Le stockage des résultats

Le stockage des résultats reprend une structure hiérarchique comme celle décrite en 4.1.2.2. Un espace de résultats est associé au GC_{term}, un autre au GC_{agt}. Une file d'attente contenue dans l'agent de représentation permet au GC_{agt} de placer un résultat destiné au GC_{term} accompagné d'une référence dans la structure hiérarchique. Ce résultat est envoyé lorsque l'utilisateur est en mode connecté.

5.2.3.1 La classe `Resultat`

Une classe abstraite `Resultat` a été créée. Tout résultat envoyé est une instante d'une sous-classe de `Resultat`. Un objet `Resultat` contient notamment :

- une méthode qui retourne la liste des composants du résultat
- une liste définissant pour chaque composant s'il doit être envoyé automatiquement ou à la demande
- une liste constituée d'un nom de composant et d'un objet qui correspond au composant

Lorsqu'une tâche retourne un résultat, il est envoyé à un répartiteur qui crée deux objets `Resultat` :

1. l'un est envoyé vers l'espace de stockage des résultats associé au GC_{agt}. Il contient tous les composants à envoyer à la demande au GC_{term} via la file d'attente.

2. l'autre est envoyé vers la file d'attente. Il contient tous les composants à envoyer automatiquement au GC_{term}. Dans la liste contenant les couples constitués d'un nom de composant et de l'objet composant, une référence nulle remplace le composant qui n'est pas envoyé automatiquement. Ainsi, en interrogeant le GC_{term}, les applications connaissent les composants manquants. Cette référence est remplacée par l'objet composant, lorsque l'utilisateur demande explicitement ce composant.

5.2.3.2 Obtention d'un résultat

Une application de l'utilisateur obtient un résultat ou un composant par requête sur le GC_{term}qui lui fournit la valeur soit parce-qu'elle est contenue dans le terminal, soit après une requête vers l'agent de représentation. Ce mécanisme est développé dans la figure 5.3 en détaillant les différents envois de messages.

- **Message 1** : la tâche communique un résultat au GC_{agt}.
- **Message 2** : le GC_{agt}communique une partie du résultat au GC_{term}via la file d'attente de l'agent de représentation, l'autre reste dans l'espace de stockage des résultats du GC_{agt}.
- **Message 3** : le GC_{term}rend disponible le résultat à l'application correspondante. L'application reçoit aussi l'identifiant des composants restés sur l'agent de représentation.
- **Message 4** : une application demande un résultat ou un composant de résultat au GC_{term}. Si celui-ci est dans l'espace de stockage de résultats associé au GC_{term}, le GC_{term}lui fournit immédiatement en envoyant le message 5. S'il est sur l'agent de représentation :
  - **Message 4a** : le GC_{term}demande la valeur au GC_{agt}.
  - **Message 4b** : le GC_{agt}la retourne au GC_{term}.
- **Message 5** : Le GC_{term}retourne la donnée demandée par la requête 4 à l'application.

Tous ces envois de messages sont réalisés par invocation de méthodes distantes grâce aux API RMI du JDK 1.1.

5.2.4 La délégation de tâches

L'implantation permettant d'effectuer une délégation de tâches du terminal vers l'agent de représentation est effectuée en prenant pour base le langage Java. Cependant, toute implantation dans un autre langage reste possible.

5.2.4.1 Les chargeurs de tâches

Pour que l'agent de représentation puisse recevoir une tâche provenant du terminal de l'utilisateur, il faut qu'il puisse la télécharger par une liaison terminal - Agent de représentation. De la même façon, même si cela est optionnel, l'objet du terminal fournissant cette tâche à l'agent de représentation doit charger cette tâche à partir d'une zone mémoire, des disques par exemple, contenue dans le terminal (voir Figure 5.4).

Le *Chargeur de Tâches du Terminal* ou *CTT* est un objet du terminal dont le rôle de charger des classes de tâches à partir d'une mémoire contenant le code et de les envoyer vers le CTAR.

Le *Chargeur de Tâches de l'Agent de Représentation* ou *CTAR* est un objet de l'agent de représentation téléchargeant des classes décrivant des tâches provenant du CTT.

Ecrites en Java, les deux classes, dont le CTT et le CTAR sont respectivement les instances, implémentent la classe abstraite définie dans le standard Java `ClassLoader`. Le CTT peut, par exemple, obtenir les classes décrivant les tâches par accès au disque local du terminal. Le CTAR obtient ces mêmes classes du CTT par requêtes RMI.

5.2.4.2 La classe abstraite `Tache`

La classe abstraite `Tache` La classe abstraite `Tache` contient la structure et l'interface communes aux classes `TacheCPDT` et `TacheCID` définissant respectivement les CPDT et CID. Par conséquent, ces deux classes sont des sous-classes de la classe `tache` (voir Figure 5.5).

Cette classe définit principalement une interface de méthodes abstraites permettant un accès standardisé aux méthodes d'un CPDT ou d'un CID. Ces méthodes sont les suivantes :

- `Demarrer` : Lance l'éxécution de la tâche.
- `Detruire` : Détruit la tâche.

La classe `TacheCPDT` définissant à un CPDT La classe `TacheCPDT` hérite de la classe `Tache`. Elle se présente sous la forme d'une classe abstraite dont tout CPDT hérite et implémente les méthodes abstraites héritées `Demarrer` et `Detruire`.

Cette classe contient une instance d'un chargeur de CID que nous appelons `ChargeurCID`.

Cette classe contient aussi deux méthodes supplémentaires :

- `DemanderMigration` : Demande si une migration est possible. Si oui, l'exécution de la tâche est gelé.
- `Redemarrer` : Continue l'exécution de la tâche après une migration.

La classe `TacheCID` définissant un CID La classe `TacheCID` est une classe abstraite héritée par tout CID proposé par un serveur.

L'importation de CID n'a cependant pas encore été expérimentée. Il faut souligner que l'implantation d'une importation de CID ne présente pas de problèmes techniques majeurs.

5.2.4.3 Les tâches réalisées

Comme déjà mentionnée, deux tâches ont été implantées: une permettant de recevoir des courriers électroniques, une autre permettant de demander des pages HTML à partir d'URL.

La tâche de courrier électronique Cette tâche consulte régulièrement un serveur de courriers électroniques. Lorsqu'un nouveau courrier destiné à l'utilisateur est arrivé, il est chargé dans la tâche. Ensuite, il est décomposé en plusieurs champs qui apparaîtront sous la forme de composants dans l'objet résultat qui sera envoyé au répartiteur. Les composants `Emetteur`, `Sujet`, `Date` sont des composants à transmettre automatiquement à l'utilisateur pour lui donner un aperçu du courrier, le composant `Contenu` est lui, par contre, envoyé que sur demande.

La tâche de demande de pages HTML Cette tâche permet à l'utilisateur d'envoyer une requête demandant à la tâche de lui transmettre une page correspondant à une URL. Cette tâche crée, après réception de la page, un objet résultat constitué d'un seul composant correspondant à la page. Ce composant est envoyé automatiquement à l'utilisateur.

5.3 Conclusion

L'implantation a pour objectifs l'expérimentation et la validation du modèle proposé et ne se veut pas être un prototype. L'environnement mobile est simulé sur un réseau limité de machines. L'interface du terminal mobile se présente ainsi sous la forme d'une interface *web*.

Le langage choisi est Java pour sa portabilité, sa large diffusion, son grand nombre d'outils disponibles. Ainsi, avec Java, il est possible de réaliser sans difficulté majeure à la fois une interface utilisateur, des agents mobiles et des tâches à envoyer vers un agent de représentation. L'implantation réalisée s'appuie, notamment en ce qui concerne la migration, sur une plate-forme d'agents mobiles, l'Aglet Workbench d'IBM.

La maquette réalisée a permis principalement l'expérimentation d'une structure de données à la fois présente dans le terminal et dans l'agent de représentation et celle des communications entre leur gestionnaire de communication respectif. L'envoi de résultats sous forme de composants sur lesquels un traitement concernant l'envoi (automatique ou à la demande) a pu aussi être testé.

Finalement, cette expérimentation prouve la faisabilité du modèle présenté tout au long de ce document.

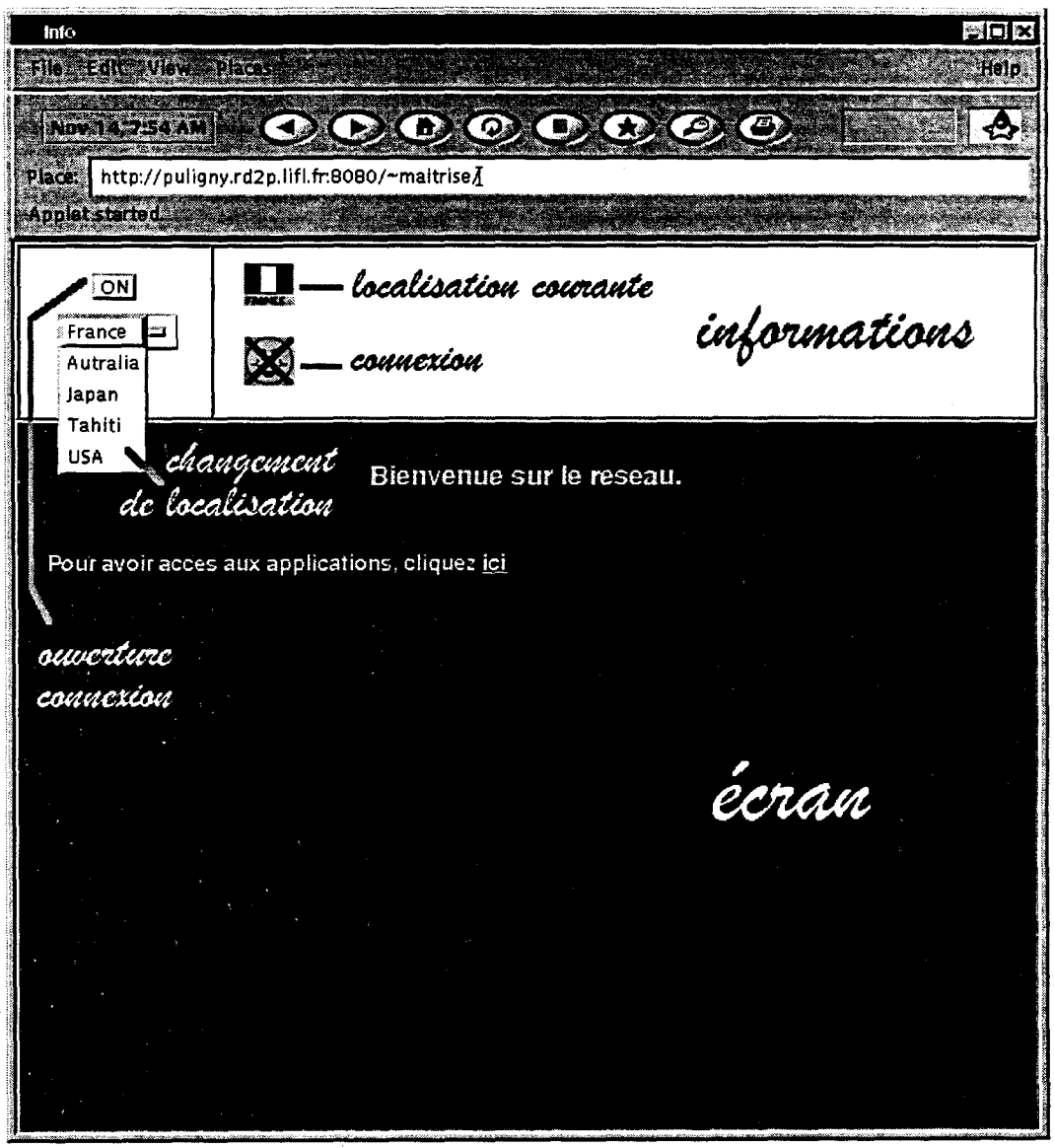


FIG. 5.2 - *simulation d'un terminal utilisateur*

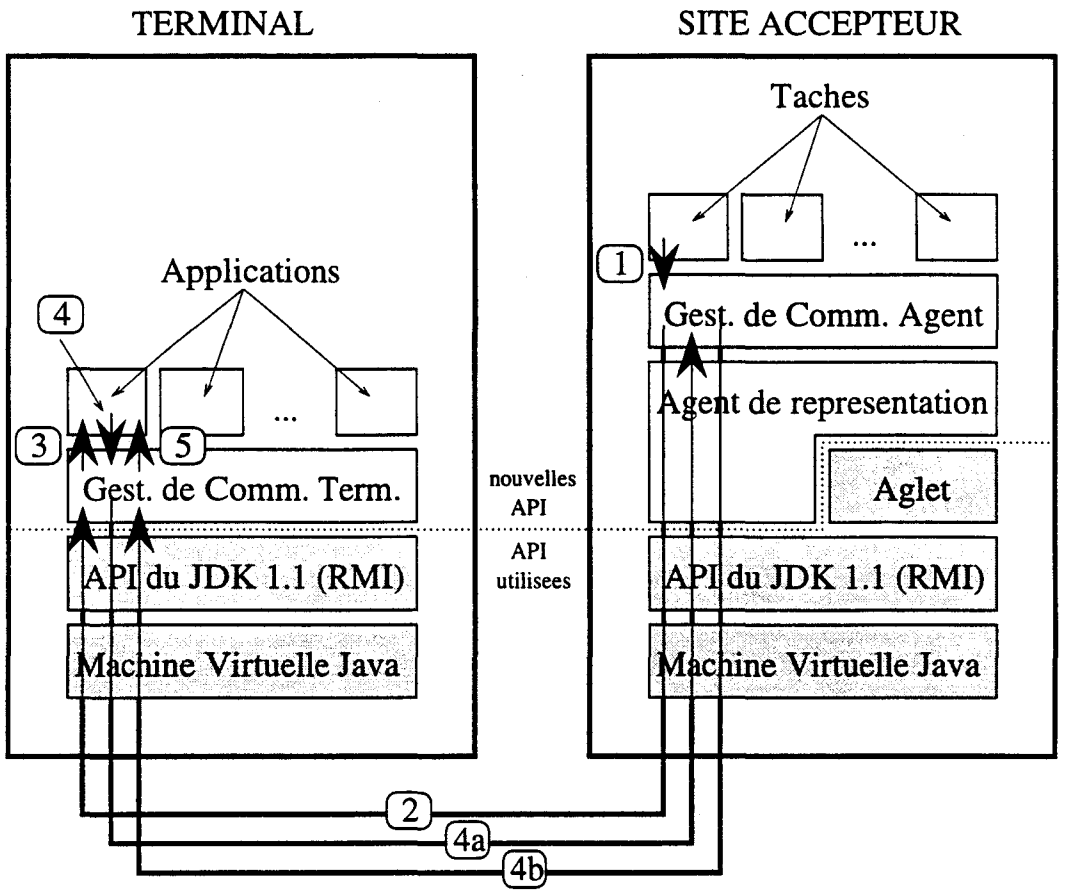


FIG. 5.3 - *Les API utilisées pour la communication de résultats*

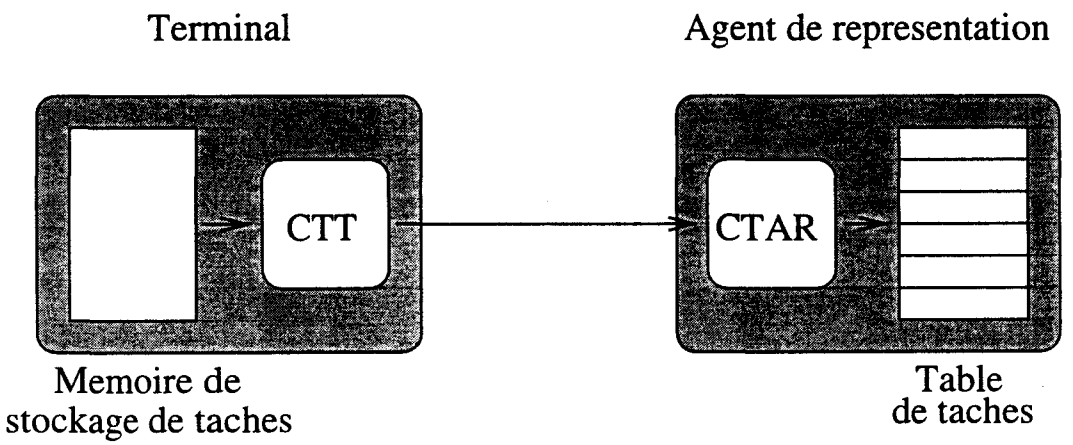


FIG. 5.4 - *Chargement d'une tache*

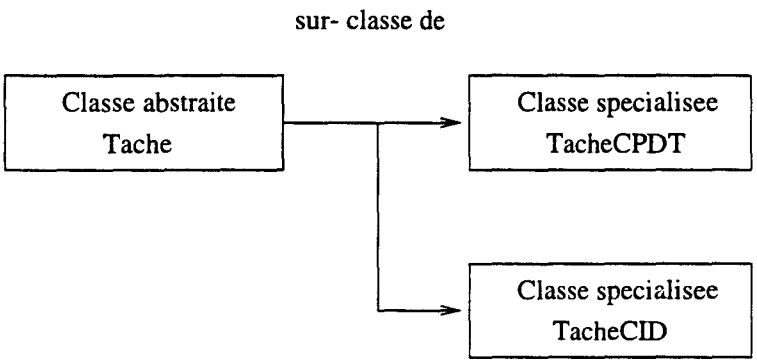


FIG. 5.5 - *Hiérarchie des classes de tâches*

Chapitre 6

Conclusion et perspectives

6.1 Apports

Le système basé sur les agents de représentation proposé dans ce document constitue un apport important dans le domaine de l'informatique mobile. En effet, les conséquences des contraintes matérielles telles que la taille, le poids, l'absence de liens physiques avec l'extérieur pour la transmission ou l'alimentation en énergie sont en grande partie résorbés.

La délégation des tâches telle qu'elle a été présentée dans les chapitres précédents est, sans contexte, un outil adapté à l'utilisateur mobile. Une application peut avoir besoin d'une puissance de calcul plus importante, d'une connexion continue pour traiter une opération ou communiquer même lorsque l'utilisateur n'est pas connecté. En déléguant une tâche vers l'agent de représentation de l'utilisateur, une partie de cette application est toujours active et dispose de ressources en plus grand nombre et de meilleur qualité.

Les tâches envoyées vers l'agent de représentation ont un traitement à effectuer et des résultats à retourner. Une tâche peut d'ailleurs renvoyer à l'application régulièrement des données sans requête venant du mobile, ce qui évite l'émission, grande consommatrice d'énergie. Les tâches, en même temps qu'elles retournent des résultats via le système de l'agent de représentation, leur associent des recommandations quant à la manière de transmettre ces résultats. Ces recommandations permettent d'adapter les données en fonction des caractéristiques matérielles du terminal utilisé, des préférences et des connexions au réseau de l'utilisateur. Une tâche, par conséquent, n'a pas à faire de traitement particulier à partir de ces paramètres. Elle

retourne les résultats au système de l'agent de représentation qui les adapte pour toutes les tâches présentes.

Grâce au couple constitué de l'agent de représentation et de la représentation initiale fixe, l'utilisateur est toujours accessible. Le modèle de localisation de l'agent de représentation présenté dans ce document est adapté aux réseaux actuels. De plus, même déconnecté, cet utilisateur est toujours joignable et peut recevoir des données de l'extérieur qui lui seront fournies lors de prochaines connexions.

Enfin, la carte à microprocesseur joue le rôle d'élément commun entre les différents terminaux utilisés au long des déplacements de l'utilisateur. Elle permet de personnaliser l'environnement d'un terminal anonyme et fournit les moyens nécessaires à l'utilisateur pour lui garantir une identification/authentification fiable et une confidentialité des données échangées entre le terminal et l'agent de représentation.

6.2 Intérêt du travail

L'informatique mobile, personnelle et dotée de moyens de communication est un domaine encore récent et en plein développement. Parti de peu il y a cinq ans, le nombre de téléphones cellulaires a très vite dépassé le million d'unités. Dans un même temps, l'utilisation d'assistants personnels et d'ordinateurs portables s'est développée. Ces deux domaines actuellement tendent à ne faire qu'un afin de proposer des terminaux mobiles pouvant communiquer. Actuellement, les utilisateurs mobiles ont un choix en équipement assez restreint :

- Un ordinateur portable avec un système provenant de stations fixes à partir duquel la liaison extérieure est utilisée de la même façon qu'une liaison sur un réseau cablé. La plupart des applications sont, par conséquent, difficilement ou partiellement utilisables. Le faible temps de connexion et le débit ne sont pas pris en compte et limitent l'utilisation de ces applications communiquantes. Le courrier électronique est, par exemple, utilisé mais bien souvent seulement pour le texte.

- Les téléphones mobiles sur lesquelles les applications sont adaptées à la liaison mais sont limitées par un réseau dédié à la téléphonie ou au transfert de courts messages. Rien n'est pour le moment proposé pour le transfert de données informatiques plus complexes.

En fait, l'association des communications sans-fil à l'informatique portable, n'est aujourd'hui qu'une application de l'un à l'autre sans souci des contraintes au niveau applicatif. Le système des mobiles est bien souvent inadapté aux communications à faible débit et intermittentes.

L'utilisation d'un système basé sur les agents de représentation permet la prise en compte et l'intégration dans les réseaux actuels d'applications mobiles communiquantes. En effet, une partie de ces applications peut être toujours accessible même lorsque l'utilisateur n'est pas connecté. De plus, le transfert de données est adapté au terminal de l'utilisateur et à la liaison utilisée.

Une proposition de projet européen s'oriente dans une direction similaire. Ce projet dirigé par *Ericsson Eurolab* est intitulé «An Open Communication Environment

Using Agent Technology». Il propose l'utilisation de la technologie agent mobile pour la téléphonie mobile. Il est plutôt orienté application. La carte à microprocesseur est utilisée afin de garantir la sécurité.

Ce projet, en cours de démarrage, prouve la bonne orientation des recherches que nous avons menées depuis trois ans. Il tend, par l'utilisation d'agents mobiles, à diviser en deux parties les applications, une sur le terminal, une sur le réseau.

6.3 Perspectives

Le modèle présenté dans ce document s'est attaché à décrire une architecture générale. Avant une exploitation d'un tel système, il est nécessaire de prendre en compte d'autres aspects qui n'ont pas encore été suffisamment développés.

La sûreté de fonctionnement des agents de représentation est notamment indispensable surtout si l'agent de représentation contient des données importantes. Plusieurs aspects sont à étudier :

1. La perte d'une représentation
   Des problèmes interviennent lorsque l'agent de représentation ou la RIF d'un utilisateur devient inaccessible ou est perdue. L'utilisation de réplicats peut permettre une duplication des données de ces deux éléments. Des points de contrôle valident les opérations entre les copies. Il est nécessaire de déterminer un mécanisme permettant à un réplicat de remplacer un élément en panne [CT96b].

2. les mécanismes de détection de sites accepteurs en panne
   Ils permettent d'éviter la migration vers des sites accepteurs inaccessibles ou en panne [CT96a]. Ces sites ne sont, par conséquent, plus proposés aux agents désirant migrer. La détection de sites accepteur permet aussi de déterminer les agents de représentation devenus inaccessibles [TC96].

3. la réplication
   Nous avons choisi un mécanisme de rapprochement de l'agent de représentation vers l'utilisateur basé sur le déplacement de code. Un mécanisme de duplication vers un site plus proche aurait pu être choisi. Ce mécanisme est plus tolérant aux pannes grâce à des copies sur le réseau. Cependant, les réplications nécessitent des mises-à-jour entre les différents réplicats et les problèmes pour reconstruire une représentation à jour en cas de perte de la représentation principale.

Ces différents problèmes impliquent des études à mener pour créer un produit utilisable *grandeur nature*. Ils ne changent pas l'utilisation et les caractéristiques du modèle présenté. Ils sont transversales au modèle et indépendants de ses fonctions.

Annexes

Annexe A

Grammaire de description de résultats

A.1 Envoi *Automatique* ou *A la demande*

- <TypeEnvoi> = <NatureEnvoi> | <OperationConditionEnvoi>

- <NatureEnvoi> = Automatique | ALaDemande

- <OperationConditionEnvoi> = SI <Condition>
  ALORS <NatureEnvoi>
  { SINON_SI <NatureEnvoi> }
  SINON <NatureEnvoi>
  FIN SI

- <Condition> = VRAI | FAUX | NOT <Condition> | (<Condition>) |
  <Condition> <OpérateurCondition> <Condition> |
  <ComparaisonNumérique> | <ComparaisonDeChaînes>

- <OpérateurCondition> = ET | OU

- <ComparaisonArythmétique> = <ExpressionNumérique>
  <ComparateurNumérique> <ExpressionNumérique>

- <ExpressionNumérique> = <ExpressionMultiplicative> |
  <ExpressionNumérique> <OpérateurAdditif> <ExpressionNumérique>

- <ExpressionMultiplicative> = <Nombre> | <NomVariable> |
  (<ExpressionNumérique>) | <ExpressionMultiplicative>
  <OpérateurMultiplicatif> <ExpressionMultiplicative>

- <ComparateurNumérique> = '==' | '!=' | '<' | '>' | '<=' | '>='

- <OpérateurAdditif> = '+' | '-'

- <OpérateurMultiplicatif> = '*' | '/'

<ComparaisonDeChaînes> = <Chaîne> <ComparateurChaîne> <Chaîne>

- <ComparateurChaîne> = '==' | '!='

Remarque A.1 *La partie SINON est obligatoire dans une opération conditionnelle. En effet, comme pour des fonctions dans les langages structurés, la fin d'une opération conditionnelle doit toujours renvoyer une valeur que la condition soit vérifée ou non.*

A.2 Sécurité de l'envoi

- <TypeSécurité> = <NatureSécurité> |
  <OperationConditionSécurité>

- <NatureSécurité> = RSA(<ExpressionNumérique>) |
  DES(<ExpressionNumérique>) | ...

- <OperationConditionSécurité> = SI <Condition>
  ALORS <NatureSécurité>
  { SINON_SI <NatureSécurité> }
  SINON <NatureSécurité>
  FIN SI

A.3 Permanence de l'envoi

- <TypePermanence> = <NaturePermanence> |
  <OperationConditionPermanence>

- <NaturePermanence> = Permanent |
  NomPermanent

- <OperationConditionPermanence> = SI <Condition>
  ALORS <NaturePermanence>
  { SINON_SI <NaturePermanence> }
  SINON <NaturePermanence>
  FIN SI

Annexe B

Acronymes utilisés

APDU : Application Protocol Data Unit *(Carte à microprocesseur)*
API : Application Programming Interface
ATP : Agent Transport Protocol *(Systèmes d'agents)*

BSC : Base Station Controller *(Réseaux mobiles)*
BTS : Base Transceiver Station *(Réseaux mobiles)*

CID : Code Importable Distant
CORBA : Common Object Request Broker Architecture
CPDT : Code Personnel de Description de Tâche
CPU : Central Processing Unit
CQL : Card Query Language *(Carte à microprocesseur)*
CSL : Computer Science Laboratory (Sony-CSL) *(Organisation)*

DES : Data Encryption Standard
DF : Dedicated File
DNS : Domain Name System

EEPROM : Electrically Erasable Programmable Read Only Memory
EF : Element File
ETSI : European Telecommunications Standards Institute

FTP : File Transfer Protocol

GC : Gestionnaire de Communication
GC_{agt} : Gestionnaire de l'Agent de Représentation
GC_{term} : Gestionnaire de Communication du Terminal
GNS : Global Name Service *(Réseaux)*
GSM : Global System for Mobile communications *(Réseaux mobiles)*

HiperLAN : High Performance LAN *(Réseaux)*
HLR : Home Location Register *(Réseaux mobiles)*
HTML : HyperText Markup Language
HTTP : HyperText Transfer Protocol

IA : Intelligence Artificielle
IAD : Intelligence Artificielle Distribuée
IBM : International Business Machines *(Organisation)*
IdT : Identifiant Temporaire
IEEE : Institute of Electrical and Electronics Engineers *(Organisation)*
IMAP : Internet Message Access Protocol
IP : Internet Protocol *(Réseaux)*
I-TCP : Indirect - Transmission Control Protocol *(Réseaux mobiles)*
ISO : International Standardization Organisation *(Organisation)*

J-AAPI : Java Aglet Application Programming Interface *(Systèmes d'agents)*
JAIST : Japan Advanced Institute of Science and Technology *(Organisation)*
JDBC : Java DataBase Connectivity *(Java)*
JDK : Java Development Kit *(Java)*
JNA : Java Network Agent *(Systèmes d'agents)*

KQML : Knowledge Query and Manipulation Language

LAN : Local Area Network *(Réseaux)*
LIFL : Laboratoire d'Informatique Fondamentale de Lille *(Organisation)*

MAF : Mobile Agent Facility *(Systèmes d'agents)*
MF : Master File *(Carte à microprocesseur)*
MIME : Multipurpose Internet Mail Extensions
MSC : Mobile service Switching Center *(Réseaux mobiles)*
M-RPC : Mobile - Remote Procedure Call *(Réseaux mobiles)*

NIP : Numéro d'Identification Personnel *(Carte à microprocesseur)*

OMG : Object Management Group *(Organisation)*
OSI : Open System Interconnection *(Organisation)*

PACS : Personal Access Communications Services *(Réseaux mobiles)*

PCMCIA : Personal Computer Memory Card International Association
PDA : Personal Digital Assistant *(Réseaux mobiles)*
PHS : Personal Handyphone System *(Réseaux mobiles)*
PME : Porte-Monnaie Electronique *(Carte à microprocesseur)*
PROM : Programmable Read Only Memory

RAM : Random Access Memory
RD2P : Recherche Développement Dossier Portable *(Organisation)*
RFC : Request For Comments
RIF : Représentation Initiale Fixe
RISC : Reduced Instruction Set Computer
RMI : Remote Method Invocation *(Java)*
ROM : Read Only Memory
RPC : Remote Procedure Call

SIM : Subscriber Identity Module *(Carte à microprocesseur) (Réseaux mobiles)*
SGBD : Système de Gestion de Bases de Données
SGML : Standard Generalized Markup Language
SMS : Short Message Services *(Réseaux mobiles)*
SQL : Structured Query Language

TCP/IP : Transmission Control Protocol / Internet Protocol
TRL : Tokyo Research Laboratory (IBM-TRL) *(Organisation)*
TTL : Time To Live

URL : Uniform Resource Locator

VIP : Virtual Internet Protocol *(Réseaux mobiles)*
VLR : Visitor Location Register *(Réseaux mobiles)*
VRML : Virtual Reality Modeling Language

Annexe C

Bibliographie

[A95] T. Alexandre, *Manipulation de données multimédia dans la carte à microprocesseur: application à l'identification biométrique et comportementale*, Thèse de Doctorat de l'Université des Sciences et Technologie de Lille, Février 1995

[ABD89] Atkinson, Bancilhon, Dewitt, Dittrich, Maier, Zdonik, *The Object-Oriented Database System Manifesto*, DOOD'89: International Conference on Deductive and Object-Oriented Databases, 1989

[Ada87] *Manuel de référence du langage ADA*, Alsys, Février 1987

[B94] T. Berners-Lee, *Uniform Resource Locators (URL)*, RFC 1838, Décembre 1994

[BIV92] B.R. Badrinath, T. Imielinsky, A. Virmani, *Locating Strategies for Personnal Communication Networks*, IEEE Globecom'92 Workshop on networking for personal communications applications, Décembre 1992

[BF93] N. Borenstein, N. Freed, *MIME Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, RFC 1521, Bellcore, Innosoft, Septembre 1993

[BB95] A. Bakre, B.R. Badrinath, *Handoff and Systems Support for Indirect TCP/IP*, 15th International Conference on Distributed Computing systems (ICDCS), Mai 1995

[BB96] A. Bakre, B.R. Badrinath, *Reworking the RPC Paradigm for Mobile Clients*, ACM/Baltzer Journal on Mobile Networks and Applications, Vol. 1, pp. 371-385, 1996

[BC95] K. Bharat, L. Cardelli, *Migratory Applications*, Actes du ACM Symposium on User Interface Software and Technology'95, pp.133-142, 1995

[BCG87] J. Banerjee, H.T. Chou, J.F. Garza, W. Kim, D. Woelk, N. Ballou, *Data Model Issues for Object-Oriented Applications* ACM TOIS, Vol.5 No.1, pp.3-26, Janvier 1987

[BCK93] A. Banerji, D.L. Cohn, C. Kulkarni, *Mobile Computing Personae*, IEEE - WWOS-IV'1993, Fourth Workshop on Workstation Operating Systems, 1993

[BGH91] G.S. Blair, J.J. Gallagher, D. Hutchinson, D. Shepard, *Object-Oriented Languages, Systems and Applications*, Halsted Press, 1991

[BN84] A. Birrell, B.J. Nelson, *Implementing Remote Procedure Calls*, ACM Transactions on Computer Systems, pp.39-59, Février 1984

[BP93] P. Bhagwat, C.E. Perkins, *A mobile Networking System Based on Internet Protocol (IP)*, Actes de USENIX Symposium on Mobile and Location-Independent Computing, pp.69-82, Août 1993

[C94] M. Crispin, *Internet Message Access Protocol - Version 4*, RFC 1730, University of Washington, Décembre 1994

[C95a] L. Cardelli, *Obliq : A langage with Distributed Scope*, Computing Systems, 8-1, pp.27-59, Janvier 1995

[C95b] D. Carlier, *Agents et mobilité*, Actes des journées de travail sur la notion de multi-agents, Ganymède-LIFL publication interne Num.163, Février 1995

[C95c] D.E. Cormer, *Internetworking with TCP/IP, Volume I : Principles, Protocols and Architecture*, Troisième éditions, Prentice Hall International Editions, 1995

[Ccitt88] *Recommandation X500 : The Directory - Overview of Concepts, Models and Service*, International Telecommunications Union, 1988

[CG95] C. Cormier, G. Grimonprez, *A RISC Approach for Smart Cards*, Euromicro'95, 1995

[CardTech96] *CardTech/SecureTech*, CardTech/SecureTech 1996 Conference proceedings, May 1996

[CD97] D. Carlier, D. Donsez, *Permanent Network Representation for Mobile User*, OPODIS'97 : International Conference On Principles Of Distributed Systems, Décembre 1997

[CDK94] G. Coulouris, J. Dollimore, T. Kindberg, *Distributed Systems : Concepts and Design*, seconde édition, Addison-Wesley Publishing Company, 1994

[CI94] R. Cceres, L. Iftode, *The Effects of Mobility on Reliable Transport Protocol*, In proceedings of the 14th Conference on Distributed Computing System, Juin 1994

[CFF92] H.Chalupsky, T. Finin, R. Fritzson, D. McKay, S. Shapiro, G. Wiederhold, *An Overview of KQML : A Knowledge Query and Manipulation Language*, KQML Advisory Group, Avril 1992

[CLT96] D. Carlier, S. Lecomte, P. Trane, *Smart card use to manage user's mobility*, Actes de Cardis 1996 : Smart Card Research and Advanced Applications, Septembre 1996

[CLT97] D. Carlier, S. Lecomte, P. Trane, *The use of a Representation as a Solution to Wireless Drawbacks : Application in the Context of Smard Card*, IEEE-ICPWC'97 : International Conference on Personnal Wireless Communications, Décembre 1997

[CML97] M.J. Caraty, C. Montaclé, F. Lefèvre, *Segmentation multi-canaux de vidéos en séquences*, Dynamic Lexicon for a Very Large Vocabulary Vocal Dictation, Eurospeec'97, 1997

[CT95] D. Carlier, P. Trane, *Security Requirements for Mobile Computing Systems*, FTS 95-70, Technical Report of IEICE, pp 57-65, Décembre 1995

[CT96a] D. Carlier, P. Trane, *On the Importance of Detecting Sites Failures in Mobile Computing Systems*, IEICE Spring Conference, Mars 1996

[CT96b] D. Carlier, P. Trane, *Fault Tolerant Issues Using Agents for Mobile Computing*, Second IEEE International On-Line Testing Workshop, pp 182-186, Juillet 1996

[CT96c] D. Carlier, P. Trane, *Designing Secure Agents with O.O. Technologies for User's Mobility*, IFIP'96 in Mobile Communications, pp.78-85, Septembre 1996

[CT97a] D. Carlier, P. Trane, *Task Delegation Model Assigned to Mobile computing*, IEEE-ICICS'97: First International Conference on Information, Communications and Signal Processing, Vol.1, pp.220-224, Septembre 1997

[CT97b] D. Carlier, P. Trane, *Task O.O. Approach for Secure Agents in Mobile Computing*, Revue Global Communications - Interactive'97, pp.288-290, Edistions Hanson Cooke Limited, Septembre 1997

[Corba97] *Common Object Request Broker Architecture : Specification and Architecture, revision 2.1*, Object Management Group, Septembre 1997

[D94] Didier Donsez, *WEA, un Gérant d'Objets Persistants pour des Environnements Distribués*, Thèse de doctorat de l'Université Pierre et Marie Curie (Paris 6), 1994

[D95] A. Derycke, *Le multi-agent : Une méthaphore pour la conception des interfaces homme-système coopératives et des collecticiels?*, Actes des journées de travail sur la notion de multi-agents, Ganymède-LIFL publication interne Num.163, Février 1995

[DAP94] P. Durant, P. Ardouin, M.J. Papillon, A. Gamache, G. Lavoie, J. Bérubé, J.P Fortin, *A Universal Memory Card Server*, Actes de CARDIS'94, First Smart Card Research and Advanced Application Conference, Octobre 1994

[Des93] *DES : Data Encryption Standard*, Federal Information Processing Standards Publication 46-2, National Institute for Standards and Technology, Décembre 1993

[Dragon97] *Dragon Naturally speaking*, Documentation technique, Dragon Systems, 1997

[E96] *Using the EC Trust Manager*, White paper, Electronics Communities, Septembre 1996

[E97] M. Ericson, *Java to IDL RFP*, OMG Technical Library orbos/97-03-08, Mars 1996

[Etsi95] *Radio Equipment and Systems (RES) : High Performance Radio Local Area Network*, Functional Specification, ETSI : European Telecommunications Standards Institute, draft prETS 300 652, Juillet 1995

[F94] J. Ferber, *La kénétique : des systèmes multi-agents à une science de l'interaction*, Revue internationale de Systémique, Vol.8 Num.1 pp.13-27, 1994

[FM96] J.S. Fritzinger, M. Mueller, *Java Security*, Technical report, Sun Microsystems Inc., Décembre 1996

[FZ94] G.H. Forman, J. Zahorjan, *The Challenges of Mobile Computing*, IEEE Computer, pp.38-47, Avril 1994

[G92] G. Grimonprez, *Etude et réalisation d'une carte à microprocesseur intégrée aux SGBDs*, Thèse d'habilitation à diriger des recherches, Université des Sciences et Technologies de Lille, Février 1992

[G97a] S.B. Guthery, *Java Card: Internet Computing on a Smart Card*, IEEE Internet Computing Vol.1 -No.1, Janvier 1997

[G97b] N. Ghosn, *La place de la carte à microprocesseur dans les systèmes d'information de santé*, Thèse pour le diplôme d'état de Docteur en médecine, Université de Santé et de Droit de Lille, Mai 1997

[GC94] S. Gadol, M. Clary, *Nomadic Tenets - A User's Perspective*, Sun Microsystems Laboratories Inc., Rapport technique SMLI-TR-94-24, Juin 1994

[GG95] J. Gosling, H. McGilton, *The Java Language Environment*, White paper, Sun Microsystems Inc., Octobre 1995

[GLR88] F. Guez, A. Lauret, C.Robert, *Les Cartes à microcircuit*, Editions Masson, 1988

[GMA95] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, P.G. Sobalvarro, *The millicent protocol for inexpensive electronic commerce*, World Wide Web Journal, dans les actes de Fourth International World Wide Web Conference, pp.603-618, Décembre 1995

[GP91] G. Grimonprez, P. Paradinas, *A new approach in code development : C-Card and Cossack*, dans les actes de CardTech/SecurTech'91, 1991

[GUQ91] L.C. Guillou, M. Ugon, J.J. Quisquater, *The Smart Card: A Standardized Security Device Dedicated to Public Cryptography*, Contemporary Cryptology, éditions G.J. Simmons, IEEE Computer Society, pp.561-614, 1991

[H96] C. Huitema, *IPv6: The New Internet Protocol*, Prentice Hall, 1996

[HC97] G. Hamilton, R. Cattel, *JDBC: A Java SQL API*, Sun Microsystems Inc., Janvier 1997

[HCK95] C.G. Harrison, D.M. Chess, A. Kershenbaum, *Mobile Agents : Are they a good idea?*, Technical report, IBM T.J.Watson, 1995

[HKN96] A. Hokimoto, K. Kurihara, T. Nakajima, *An Approach for Constructing Mobile Applications using Service Proxies*, IEEE 16th International Conference on Distributed Computing Systems (ICDCS'96), Mai 1996

[HN97] A. Hokimoto, T. Nakajima, *Robust Host Mobility Supports for Adaptative Mobile Applications*, First International Conference on Worldwide Computing and its Applications'97 (WWCA'97), Mars 1997

[HotJava97] *HotJava Browser*, Sun Microsystems Inc., 1997

[Html97] *The HTML 4.0 W3C*, World Wide Web Consortium, 1997

[Http97] *HTTP Protocol*, World Wide Web Consortium, 1997

[IB94] J. Imielinsky, B.R. Badrinath, *Mobile Wireless Computing: Challenges in Data Management*, Coomunication of the ACM, Vol.37, No.10, pp.19-28, Octobre 1994

[IDM91] J. Ioannidis, D. Duchamp, G.Q. Maguire, *IP-based Protocols for Mobile Internetworking*, Actes de ACM SIGCOMM'91, pp.235-245, Septembre 1991

[Ieee94] *Local and Metropolitan Area Networks*, IEEE Standard 802.11 Wireless LAN, IEEE: Institute of Electrical and Electronics Engineers Inc. , Decembre 1994

[Iso87] *Cartes d'identification - Cartes à circuit intégré à contacts - Partie 1: Caractéristique physiques*, Norme ISO 7816-1, ISO: International Standardization Organisation, 1987

[Iso88] *Cartes d'identification - Cartes à circuit intégré à contacts - Partie 2: Dimensions et emplacements des contacts*, Norme ISO 7816-2, ISO : International Standardization Organisation, 1988

[Iso89] *Cartes d'identification - Cartes à circuit intégré à contacts - Partie 3: Signaux électriques et protocoles de transmission*, Norme ISO 7816-3, ISO : International Standardization Organisation, 1989

[Iso93] *Information Transfert, Retrieval and Management for OSI*, Norme ISO/IEC JTC 1/SC 21, ISO : International Standardization Organisation, Mai 1993

[Iso94] *Cartes d'identification - Cartes à circuit intégré à contacts - Partie 5: Système de numérotation et procédure d'enregistrement d'identificateurs d'applications*, Norme ISO 7816-5, ISO : International Standardization Organisation, 1994

[Iso95] *Cartes d'identification - Cartes à circuit intégré à contacts - Partie 4: Commandes intersectorielles pour les échanges*, Norme ISO 7816-4, ISO : International Standardization Organisation, 1995

[Iso96] *Cartes d'identification - Cartes à circuit intégré à contacts - Partie 6: Eléments de données intersectoriels*, Norme ISO 7816-6, ISO : International Standardization Organisation, 1996

[IV94] T. Imielinsky, S. Viswanathan, *Adaptative Wireless Information Systems*, Actes de SIGDBS'94 Conference, Octobre 1994

[J93] D. Johnson, *Ubiquitous Mobile Host Internetworking*, Actes de 4th IEEE-Workshop on Workstation Operating Systems, Octobre 1993

[Java96] *Object Serialization Specification*, Sun Microsystems Inc., Décembre 1996

[Java97a] *Java Platform 1.1.3 Core API*, Sun Microsystems Inc., 1997

[Java97b] *Remote Method Invocation Specification, Revision 1.4, JDK 1.1*, Sun Microsystems Inc., Février 1997

[JavaCard97] *Java Card 2.0: Language Subset and Virtual Machine Specification*, Sun Microsystems Inc., Septembre 1997

[JavaServer97] *JavaServer: Developer Documentation*, Sun Microsystems Inc., 1997

[JavaStation97] *JavaStation - An Overview*, Sun Microsystems Inc., 1997

[K92] S. Kille, *Implementing X.400 and X.500: The PP and QUIPU systems*, Artech House, 1992

[KKT93] H. Koch, L. Krombholz, O. Theel, *A Brief Introduction into the World of 'Mobile Computing'*, Rapport Technique, Technische Hochschule Darmstadt, Allemagne, Mai 1993

[KZ97] J. Kiniry, D. Zimmerman, *A Hands-On Look at Java Mobile Agents*, IEEE Internet Computing, Vol.1 No.4, pp.21-30, Juillet-Août 1997

[L86] B.W. Lampson, *Designing a Global Name Service*, Actes du 5th ACM Symposium, Principles of Distributed Computing, pp.1-10, Août 1986

[L94] S. Lecomte, *Objets nomades et travail coopératif*, Mémoire de DEA de l'Université des Sciences et Technologies de Lille, Juillet 1994

[L97] T. Liao, *WebCanal : a Multicast Web Application*, Actes de la 6th International WWW Conference, Avril 1997

[LC96] D.B. Lange, D.T. Chang, *IBM Aglets Workbench, Programming Mobile Agents in Java*, White paper, Tokyo Research Lab., IBM Corporation, Septembre 1996

[LD97] S. Lecomte, D. Donsez, *Intégration d'un Gestionnaire de Transaction dans les cartes à microprocesseur*, NOTERE'97 : Nouvelles Techniques de Répartition, Novembre 1997

[LM96] W. Li, D.G. Messerschmitt, *Java-To-Go : Itinerative Computing Using Java*, Department of Electrical Engineering and Computer Science, University of California at Berkeley, Septembre 1996

[LO97] D.B. Lange, M.Oshima, *Java Agent API: Programming and Deploying Aglets with Java*, IBM - Tokyo Research Lab., éditions Addison-Wesley, 1997

[LY97] T. Lindholm, F. Yellin, *The Java Virtual Machine Specification*, Sun Microsystems Inc., Editions Addison-Wesley, 1997

[M87] P. Mockapetris, *Domain Names - Concepts and Facilities*, Rapport Technique RFC 1034, Internet Network Information Center, 1987

[M92] B. Meyer, *Eiffel: The Language*, Editions Prentice Hall Object-Oriented Series, 1992

[M93] K. Moore, *MIME Part Two: Message Header Extensions for Non-ASCII Text*, RFC 1522, University of Tennessee, Septembre 1993

[M97] P. Madany, *JavaOS: A Stand Alone Java Environment* White Paper, Sun Microsystems Inc., 1997

[MDC93] B. Marsh, F. Douglis, R. Cceres, *Systems Issues in Mobile Computing*, rapport technique, MITL : Matsushita Information Technology Laboratory, Février 1993

[MP92] M. Mouly, M.B. Pautet, *The GSM System for Mobile Communications*, publié par les auteurs, 1992

[MSF97] W. Merlat, C. Seyrat, J. Ferber, *Mobile Agents for Dynamic Organizations : The Conversational-Agent Paradigm*, Eighth European Workshop on Modelling Autonomous Agents in Multi-Agent World, Mai 1997

[NH97] T. Nakjima, A. Hokimoto, *Adaptative Continous Media Applications in Mobile Computing Environments*, IEEE International Conference on Multimedia computing and Systems (ICMCS'97), Juin 1997

[Nokia97] *Nokia 9000*, Documentation technique, Nokia, 1997

[Oma95] *Oracle mobile Agents*, White paper, Oracle, 1995

[Oma96] *Oracle mobile Agents, Security White Paper*, White paper, Oracle, 1996

[Omg97] *Mobile Agent Facility Specification*, Joint Submission, OMG TC Document, écrit en collaboration entre Crystaliz Inc., Ceneral Magic Inc., GMD Fokus, IBM, Juin 1997

[P88] P. Paradinas, *La BioCarte : Intégration d'une carte à microprocesseur dans un réseau professionnel santé*, Thèse de Doctorat de l'Université des Sciences et Technologie de Lille, 1988

[P95a] T. Peltier, *La Carte Blanche : Un nouveau système d'exploitation pour objets nomades*, Thèse de Doctorat de l'Université des Sciences et Technologies de Lille, Décembre 1995

[P95b] G. Pujolle, *Les Réseaux*, Deuxième édition revue et corrigée, Editions Eyrolles, pp.559-673, 1995

[PN97] J.M. Place, B. Noclercq, *An Advanced Card Operating System on the Cascade Platform*, EMMSEC'97 : European Multimedia, Microprocessor Systems and Electronic Commerce, Novembre 1997

[Postscript97] *Adobe Postscript 3*, Adobe Systems Inc., 1997

[PR92] D.L. Presotto, D.M. Ritchie, *Interprocessor Communication in the Eight Edition UNIX System*, Actes de la conférence USENIX 1992, USENIX Association, 1992

[R96] F. Rouaix, *A Web navigator with applets in Caml*, Fifth International World Wide Web Conference, Mai 1996

[R97] *Petit Robert : Dictionnaire de la langue française*, Dictionnaires Le Robert, 1997

[Rsa78] R.L. Rivest, A. Shamir, L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystem*, Communication of the Association for Computing Machinery, Vol.21, No.2, pp.120-126, Février 1978

[Rsa91] *PKCS#1 : RSA Encryption Standard*, RSA Laboratories, 1991

[S94] B. Schneier, *Applied Cryptography*, Editions John Wiley & Sons, 1994

[S96] W.R. Stevens, *TCP/IP illustré, Les protocoles*, Volume 1, Editions International Thomson Publishing France, 1996

[SBH96] M. Straßer, J. Baumann, F. Hohl, *Mole - A Java Based Mobile Agent System*, ECOOP'96 Workshop on Mobile Object Systems, Juillet 1996

[Sgml97] *Getting Started with SGML: A Guide to the Standard Generalized Markup Language and Its Role in Information Management*, White paper #2001-AT, SGML Open, 1997

[Sim96] *Digital cellular telecommunications system (Phase 2) ; Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (GSM 11.11)*, ETSI : European Telecommunications Standards Institute, draft prETS 300 608, Août 1996

[SNK94] M. Satyanarayanan, B. Noble, P. Kumar, M. Price, *Application-Aware Adaptation for Mobile Computing*, Actes de 6th ACM SIGOPS European Workshop, Septembre 1994

[ST95] M. Sirbu, J.D. Tygar, *NetBill : An Internet Commerce System Optimized for Network Delivered Services*, Dans les actes de l'IEEE CompCon, Mars 1995

[T90] A. Tanenbaum, *Les Réseaux : architectures, protocoles, applications*, éditions InterEditions, 1990

[T95] P. Trane, *Conception et réalisation d'un système de contrôle d'accès pour la carte à micro-processeur*, Thèse de Doctorat de l'Université des Sciences et Technologie de Lille, Septembre 1995

[T97] T. Thorn, *Programming Languages for Mobile Code*, ACM Computing Surveys, Vol.29, No.3, Septembre 1997

[TC96] P. Trane, D. Carlier, *Diagnosis Algorithm for Mobility-Oriented System*, IEEE-ASAP'96 : International Conference on Application-specific System, Août 1996

[TR94] Y. Throgné, R. Reitter, *Nouvelle technologie de la carte à mémoire : la carte sans contact*, Les Echos des recherches, CNET : Centre National d'Etudes des Télécommunications, No.138, pp.43-48, 1994

[TUS94] F. Teraoka, K. Uehara, H. Sunahara, J. Murai, *VIP : A Protocol Providing Host Mobility*, Communications of the ACM 37(8), Août 1994

[V97] J.J. Vandewalle, *Projet OSMOSE : modélisation et implémentation pour l'interopérabilité de services carte à microprocesseur par l'approche orienté-objet*, Thèse de Doctorat de l'Université des Sciences et Technologies de Lille, Mars 1997

[Voyager97] *Voyager : Core Package Technical Overview*, Rapport technique, ObjectSpace Inc., Juillet 1997

[VH96] M.P. Van Hoecke, *Contribution à la Modélisation des Systèmes d'Information Communicationnels intégrant des Cartes à micro-processeurs*, Thèse de Doctorat de l'Université des Sciences et Technologies de Lille, Janvier 1996

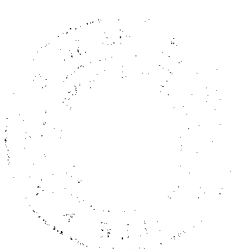
[Vrml96] *Virtual Reality Modeling Language Specification*, Version 2.0, ISO/IEC CD 14772, Août 1996

[W94] J. White, *Telescript Technology : The Foundation for the Electronic Marketplace*, White paper, General Magic, 1994

[W96] J. White, *Mobile Agent White Paper*, White paper, General Magic, 1996

[WD97] U.G. Wilhelm, X. Defago, *Objets Protégés Cryptographiquement*, Dans les actes de RenPar'9, May 1997

[WYO93] H. Wada, T. Yozawa, T. Ohnishi, T. Tanaka, *Mobile Vomputing Environment Based on Internet Packet Forwarding*, Actes de 1993 Winter USENIX Conference, pp.503-517, Janvier 1993



THESES

1998

09 Janvier — **David CARLIER**
Représentation permanente, coordonnée par une carte à microprocesseur, d'un utilisateur mobile.

16 Janvier — **Yves DENNEULIN**
Conception et ordonnancement des applications hautement irrégulières dans un contexte de parallélisme à grain fin.

23 Janvier — **Jaafar GABER**
Plongements et manipulations d'arbres dans les architectures distribuées.

30 Janvier — **Grégory SAUGIS**
Interfaces 3D pour le travail coopératif synchrone, une proposition.