Découverte et fourniture de services adaptatifs dans les environnements mobiles

Ouahiba Fouial

ECOLE NATIONALE SUPERIEURE DES TELECOMMUNICATIONS

PARIS

MEMOIRE

présenté en vue d'obtenir

le grade de docteur de l'Ecole Nationale Supérieure des Télécommunications

Spécialité informatique et réseaux

par

Ouahiba FOUIAL

Découverte et fourniture de services adaptatifs dans les environnements mobiles

Soutenue le 30 Avril 2004

JURY

PRESIDENT : Mr Hervé GUYENNET de l'université de Franche Comté

RAPPORTEURS : Mr Guy BERNARD de l'INT
Mme Dominique GAÏTI de l'université de Troyes

EXAMINATEURS : Mme Nadia BOUKHATEM de l'ENST Paris (co-encadrante de thèse)
Mme Isabelle DEMEURE de l'ENST Paris (directrice de thèse)
Mr Frank SINGHOFF de l'université de Bretagne Occidentale

À la mémoire de mon père

À ma mère

À Karim

À mes sœurs et à mes frères

« L'extraordinaire nous attire un instant, la simplicité nous retient plus longtemps, parce que c'est en elle seule que réside l'essentiel.»

Garry Winogrand,

« L'imagination est plus importante que le savoir.»

Albert Einstein,

Remerciements

Contrairement à ce que pourrait laisser penser la présence de mon nom seul sur la couverture, ce mémoire est avant tout le reflet d'un travail collectif. Je précise ceci, non pas pour me dédouaner des imperfections inévitables d'une telle étude, mais pour que chacun de ceux qui ont contribué de près ou de loin à ce travail trouve ici une juste reconnaissance.

Je tiens à exprimer mes plus vifs remerciements à Mme Isabelle Demeure qui fut pour moi une directrice de thèse attentive et disponible malgré ses charges nombreuses. Sa compétence, sa clairvoyance, son charisme, son dynamisme m'ont beaucoup appris. J'ai bénéficié d'une grande liberté dans mon travail et elle m'a offert une ouverture vers d'autres domaines que l'informatique, ouverture que j'ai beaucoup appréciée. J'ai beaucoup appris à son contact.

Je suis extrêmement reconnaissante à Nadia Boukhatem, ma co-encadrante, pour l'aide qu'elle m'a fournie à mes débuts dans la recherche et pour ses avis toujours éclairés, ainsi que pour son attitude très confiante envers moi. Je tiens à la remercier, tant pour sa contribution au développement et à la présentation des idées décrites dans ce mémoire, que pour son soutien et pour la confiance qu'elle m'a témoignée tout au long de ces années de thèse.

Je remercie les deux rapporteurs de ce mémoire, Mr Guy Bernard de l'INT et Mme Dominique Gaïti de l'université de Troyes, ainsi que Mr Hervé Guyennet de l'université de Franche Comté qui a bien voulu présider la soutenance.

Je remercie Mr Frank Singhoff pour l'attention qu'il a accordée à mon travail, pour le temps qu'il a bien voulu consacrer à ce mémoire et enfin, pour ses questions et remarques constructives et intéressantes.

J'ai évidemment beaucoup de remerciements à adressesr à tous ceux qui ont un jour ou l'autre croisé mon chemin, que ce soit à l'ENST ou non.

Je remercie tous les membres du département Infres, en particulier les thésards et le personnel administratif, pour le climat sympatique dans lequel ils m'ont permi de travailler. J'adresse un clin d'oeil complice à mes camarades thésards et je leur souhaite tout le courage pour finir leur thèse.

J'adresse une pensée toute particulière aux différentes personnes qui ont partagé leur bureau avec moi pendant les années de thèse. Je pense particulièrement à Frank, Philippe, Basma, Sergio, Aliou, Jérôme, Hasnaa et enfin Barhoum.

À Rola, merci pour ton amitié précieuse. Je n'oublirai jamais les moments qu'on a passés ensemble. T'es vraiment une amie formidable et exceptionnelle !

Je remercie spécialement Salim, pour son aide quotidienne et sa disponibilité qui m'ont embellie la vie même lorsque le travail semblait insurmontable. Je le remercie pour ses

encouragements généreux et la confiance en moi qu'il m'a toujours transmise grâce à son caractère chaleureux. Ils m'ont été très précieux durant ma thèse.

Merci à mes amis pour l'affectueuse amitié dont ils ont toujours fait preuve. Je pense particulièrement à Jacques, Mohamad, Ahmad, Hasnaa, Rana, Hazar, Dany, Habib, Mehand, Yacine et Selma.

Un merci tout particulier à ma très chère amie Dalila pour sa disponibilité, pour son soutien, pour son aide et pour nos discussions passionnées !. Une pensée très affectueuse pour mon ami Ibrahim qui a fait de moi une personne heureuse. Son amitié, son soutien et son aide m'ont été indispensables, en particulier dans les derniers mois de ma thèse.

Les mots simples étant les plus forts, j'adresse toute mon affection à mes nombreux et merveilleux frères et sœurs. Malgré mon éloignement depuis de (trop !) nombreuses années, leur intelligence, leur soutien, leur confiance, leur amour me portent et me guident tous les jours. Est-ce un bon endroit pour dire ce genre de choses ? Je vous aime beaucoup.

Je souhaite terminer en remerciant trois personnes exceptionnelles. Je réserve une reconnaissance particulière à mon cher défunt père, avec la plus grande douleur de ne plus l'avoir parmi nous, pour l'amour et le soutien incomparables dont il m'a fait preuve depuis ma naissance.

Je dois à ma mère de m'avoir donné les moyens de réaliser mes rêves, même les plus fous et de n'avoir jamais remis en question mes choix de vie. L'amour et la confiance qu'elle m'a toujours témoignés m'ont permis, non seulement de traverser ces années de thèse très difficiles, mais également d'effacer mes doutes et d'acquérir une vision réaliste de la vie. Son soutien et son enthousiasme n'ont d'égal que l'amour qu'elle me porte. Merci infiniment mama.

Je ne serais pas parvenue à réaliser ce travail sans le total dévouement de mon frère Karim. Ses encouragements, sa disponibilité et sa présence ont été (et sont toujours) indispensables. Qu'il trouve ici (bien que ces mots soient dérisoires) le témoignage de tout mon estime et tout mon amour. Merci Kerroum. Tu es simplement le meilleur frère au monde !

Résumé

Avec l'avènement de la troisième génération, les réseaux de télécommunications mobiles entrent dans une nouvelle phase de leur évolution, d'un réseau omniprésent axé sur la téléphonie mobile, vers des systèmes de fourniture de services à grande échelle. Ces systèmes doivent permettre à des usagers (éventuellement mobiles) d'accéder à leurs services et leur environnement personnalisés quels que soient le réseau et le terminal qu'ils utilisent.

Le constat actuel est que la mise en place de la fourniture de services pour ces systèmes nécessite la conception d'une plate-forme distribuée complexe faisant intervenir des éléments hétérogènes (grand choix de réseaux, plusieurs types de terminaux de capacités différentes, multiples opérateurs réseau et fournisseurs de services, contenus de services hétérogènes, etc.).
Par ailleurs, l'évolution des terminaux mobiles (ordinateurs portables, assistants personnels, téléphones portables, etc.) et des réseaux mobiles (émergence des réseaux sans fil) permet le développement d'applications fondées sur la mobilité. L'environnement d'exécution de ces applications doit prendre en compte la variété des équipements et des ressources (ex : bande passante) ainsi que la mobilité de l'utilisateur. De plus, les usagers mobiles veulent pouvoir personnaliser les services et les utiliser de la même manière qu'en environnement fixe et ce, quel que soit le service à traiter et quelles que soient les capacités du terminal utilisé.

L'objectif de cette thèse est de donner des éléments de solution à ces problèmes. Elle traite de la découverte et la fourniture de services adaptables dans les environnements mobiles. Elle a pour but de proposer une architecture permettant aux usagers mobiles de découvrir et d'exécuter des services sensibles au contexte de leur exécution. Les services proposés sont adaptés aux préférences de l'usager, aux capacités de son terminal, à sa localisation, et enfin, aux ressources réseau disponibles.

Dans un premier temps, nous nous intéressons à l'utilisation des agents mobiles pour la fourniture de services dans les environnements mobiles. Ce modèle, bien adapté aux contraintes des environnements mobiles, s'avère intéressant, principalement, dans les applications qui entraînent la consultation de plusieurs services successifs sur le réseau (ex : visite de plusieurs prestataires de services dans le cadre d'une découverte de services).

Par la suite, nous proposons une plate-forme de fourniture de services sensibles au contexte appelée CASP (Context Aware Service Provision). Dans cette plate-forme, la sensibilité au contexte est prise en compte dans les deux phases de fourniture de services suivantes : la découverte de services et l'exécution du service sur le terminal mobile.

La découverte de services permet à ces derniers de se faire connaître et aux terminaux mobiles de les découvrir automatiquement. Les systèmes de découverte de services actuellement disponibles ne sont pas sensibles au contexte et ne prennent pas en considération

les paramètres qui changent selon le contenu dynamique de l'environnement mobile. Dans la plate-forme CASP, nous proposons un mécanisme de découverte de services qui permet à l'usager mobile de personnaliser la fourniture de services et de l'adapter à ses exigences. Ce mécanisme, mis en œuvre par un élément médiateur entre l'usager mobile et les fournisseurs de services, permet de ne proposer à l'usager que les services qui sont adaptés à ses préférences, aux capacités de son terminal et à sa localisation. La liste des services proposés à l'usager contient d'autres informations qui peuvent être utiles à l'usager (description du service, tarifs indicatifs, etc). L'implémentation de ce mécanisme de découverte de services est basée sur des concepts tels que VHE (Virtual Home Environment) et les profils, et utilise des outils standard tels que XML et CC/PP (Composite Capabilities/ Preference Profiles).

La sensibilité au contexte dans la plate-forme CASP est également utilisée pour adapter les services, une fois découverts par l'usager, à leur environnement d'exécution (besoins des usagers, caractéristiques techniques du terminal, localisation du terminal et ressources du réseau). Cette adaptation est réalisée au moment de l'exécution du service sur le terminal mobile. La solution proposée est basée sur l'utilisation de serveurs intermédiaires (Serveur Proxy) entre le terminal mobile et le fournisseur du service. Un service multimédia a été développé pour valider les concepts proposés dans cette thèse.

Enfin, nous terminons par une ouverture sur l'utilisation d'une approche de conception de services à base de composants qui offre une vue modulaire des services. Chaque service est constitué d'un ensemble de composants interconnectés, pouvant être supervisés et reconfigurés si nécessaire. L'adaptation des services est alors réalisée au moment de l'accès au service et au moment de son exécution sur le terminal.
L'adaptation au moment de l'accès au service consiste à sélectionner et assembler les composants du service en fonction du contexte. L'adaptation du service au moment de son exécution consiste à changer dynamiquement la composition du service suite aux variations qui surviennent dans l'environnement.

Mots clefs

Fourniture de services, environnements mobiles, agents mobiles, découverte de services, sensibilité au contexte, adaptabilité des services, composabilité des services

Abstract

With the advent of the third generation, mobile telecommunication networks enter a new phase of their evolution, from an omnipresent network centered around mobile telephony, towards large scale service provisioning systems. These systems have to allow (possibly mobile) users to get and use their services and personalized environment whatever the network and the terminal they use.

The settlement of service provisioning for these systems requires the design of a complex distributed platform using heterogeneous elements (large choice of networks, several types of terminals of different capabilities, multiple network operators and service providers, heterogeneous service contents, etc.). In addition, the evolution of mobile terminals (laptops, personal digital assistants, mobile phones, etc.), and mobile networks (emergence of the wireless networks) allows the development of applications based on mobility. Consequently, the execution environment of such applications is itself very fluctuating, in terms of bandwidth, of end user terminal capabilities and terminal location, etc. Moreover, mobile users may want to be able to personalize their services and to use them in the same way as in regular or wired environment and this, whatever the type and the volume of the service to be treated, independently of the terminal used and its capabilities.

The aim of this thesis is to address these issues and provide solutions to them. It handles aspects of discovery and provision of adaptable services in mobile environments, having for goal to propose a platform that makes it possible to mobile users to discover and run context aware services. Services are adapted to user preferences, to user location, to terminal capabilities, and finally, to available network resources.

At first, we focus on the use of mobile agents for service provisioning in mobile environments. This model, well adapted to the constraints of mobile environments, proves to be particularly interesting for applications which involve consultation of several successive service providers on the network in the context of service discovery.

In the sequel, we propose CASP (Context Aware Service Provision), a context aware platform for service provision. In this platform, context awareness is addressed in the two following service provision phases : service discovery and service execution on the mobile terminal.

Service discovery allows services to be published and mobile terminals to discover them automatically. Service discovery systems, currently available, are not context aware and do not consider parameters which change according to the dynamic contents of mobile environment. In our CASP platform, we propose a service discovery mechanism that allows the mobile user to personalize the service provision and to adapt it to his/her requirements. This mechanism, implemented by a mediator between the mobile user and service providers, provides the user only with services which are adapted to his/her preferences, to the

terminal capabilities and to the user location. The list of suggested services contains other information useful to the user (service description, indicative rate, etc.). The implementation of this service discovery mechanism is based on concepts such as VHE (Virtual Home Environment) and profiles, and uses standard tools such as XML and CC/PP (Composite Capabilities/ Preference Profiles).

Context awareness in CASP is also used for adaptation of services, once discovered by the user, to their execution environment (users needs, terminal design features, terminal location and network resources). This adaptation is ensured at service runtime on the mobile terminal. The proposed solution is based on the use of intermediate servers (proxies) between the mobile terminal and the service provider. A multimedia service was developed to validate the concepts proposed in this thesis.

To end, we conclude by an introduction on the use of a component-based service design approach which offers a modular aspect to services. Each service consists of a set of interconnected components, able to be supervised and reconfigured if necessary. Service adaptation is then ensured at service access time and service runtime.
Service adaptation at access time consists of the selection and the assembly of service components according to the context. Service adaptation at runtime consists in dynamically changing the service composition according to the context variations.

Keywords

Service provision, mobile environments, mobile agents, service discovery, context awareness, service adaptability, service composability.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction et contributions de cette thèse

Avec l'avènement de la troisième génération, les réseaux de télécommunications mobiles entrent dans une nouvelle phase de leur évolution, d'un réseau omniprésent axé sur la téléphonie mobile, vers des systèmes de fourniture de services à grande échelle. Ces systèmes doivent permettre à des usagers (éventuellement mobiles) d'accéder à leurs services et leur environnement personnalisés quels que soient le réseau et le terminal qu'ils utilisent.
Dans cette nouvelle vision de la fourniture de services, une grande variété de services sera proposée non seulement par des opérateurs réseau mais aussi par des fournisseurs de services. Pour ces deux acteurs qui cherchent à augmenter leurs revenus, chaque usager du réseau devient un client potentiel et chaque connexion de l'usager est une nouvelle occasion pour le fidéliser, lui proposer d'autres produits ou mettre à jour ses services.
L'élaboration d'une architecture ouverte est donc nécessaire pour la fourniture de services dans ces environnements mobiles, multi-acteurs, multi-réseaux et multi-terminaux. La conception de cette architecture doit se baser sur des technologies de gestion, de contrôle et d'exécution de services. De plus, des mécanismes et des approches du domaine des télécommunications et de la communauté Internet doivent être utilisés pour répondre aux besoins de l'introduction et de la gestion flexible de services multi-contenus (images, vidéo, texte, etc.) dans ces futurs réseaux.

Dans notre travail, nous désignons par environnement mobile ou nomade tout environnement faisant référence à la mobilité de l'utilisateur et/ou à la mobilité du terminal. Ces environnements permettent à leurs utilisateurs d'accéder à des services indépendamment de leurs positions physiques.
Ainsi, l'environnement mobile peut être composé d'un ensemble de terminaux portables de capacités différentes (PDA, téléphone portable, ordinateur portable, etc.) reliés soit à un réseau sans fil (où la liaison est non filaire), soit à un réseau fixe (où la connexion est filaire mais non statique). Dans les deux cas, la connexion est temporaire avec possibilité de déconnexions.

1.1 Problématique

Dans ce travail, nous nous intéressons à la conception et à la mise en œuvre d'un système ouvert de fourniture flexible de services pour environnements mobiles. Ce système doit, d'une part, répondre aux besoins de la fourniture de services dans les futurs réseaux mobiles et, d'autre part, se baser sur les technologies et les standards qui ont été définis dans ce cadre.

Dans ces nouveaux systèmes de fourniture de services, la diversité des équipements utilisés pour la mise en œuvre de ces services concerne aussi bien les terminaux d'accès (téléphones, ordinateurs, assistants électroniques, etc.) que les infrastructures de communication (réseaux sans fils, GSM, Internet, etc.). Ainsi, l'usager de ces services sera bientôt confronté à une situation marquée par une grande **hétérogénéité** sous divers aspects : grand choix de réseaux fixes et mobiles, plusieurs types de terminaux de capacités différentes, multiples opérateurs réseau et fournisseurs de services, contenus de services très variés (images, vidéo, texte), etc.
Le constat actuel, est que la mise en place d'un système de fourniture de services pour environnements mobiles nécessite la conception d'une plate-forme distribuée complexe faisant intervenir des éléments très hétérogènes. Il est donc essentiel pour le développement des services de pouvoir masquer cette hétérogénéité pour permettre à l'utilisateur un accès transparent à des services utilisables dans toutes les configurations possibles.

Par ailleurs, l'évolution technologique des terminaux mobiles (ordinateurs portables, assistants personnels, téléphones portables, etc.) et des réseaux (émergence des réseaux sans fil, développement d'Internet) permet le développement d'applications fondées sur la **mobilité**. L'environnement d'exécution de telles applications (y compris de l'application de fourniture de services) est donc très variable, en terme de ressources réseaux, de type d'équipements utilisables par l'usager mobile, de la localisation des usagers, etc.
De plus, les usagers mobiles veulent pouvoir **personnaliser** les services et les utiliser de la même manière qu'en environnement fixe et cela, quels que soient le type et le volume du service à traiter et quelles que soient les capacités du terminal utilisé.

Dans un tel environnement mobile où tout est en perpétuel changement, il est intéressant de tenir compte des variations de l'environnement dans la conception du système. En permettant l'accès au **contexte d'exécution**, on peut fournir aux usagers la possibilité de découvrir les services les mieux adaptés à ce contexte, et donc les plus proches aux besoins de l'usager.
Dans le cadre de la fourniture de services pour environnements mobiles, le contexte peut englober l'ensemble des informations liées aux préferences de l'utilisateur, à ses besoins, à sa localisation, aux capacités de son terminal, aux ressources réseau, etc.

La complexité globale d'un système de fourniture de services n'est pas uniquement inhérente à l'hétérogénéité et à la mobilité de l'environnement. Elle est également dûe au système de fourniture de services lui-même. Un tel système doit permettre le déploiement facile et la gestion flexible des services. Il est donc nécessaire d'utiliser dans le système un mécanisme permettant le déploiement dynamique de services. D'autre part, ces services,

étant dynamiques, il est indispensable de se servir d'un mécanisme de découverte de services sensible au contexte qui permet aux usagers de s'informer sur les services proposés qui répondent au mieux à leurs besoins.

Par ailleurs, la grande variabilité de la localisation des utilisateurs, des ressources réseau et de types de terminaux (y compris des mobiles à ressources limitées) nécessite de pouvoir configurer dynamiquement les services avec une grande souplesse. Ainsi, afin d'offrir le meilleur service possible à l'utilisateur dans des configurations très différentes, les services doivent **s'adapter** au contexte de leur exécution (aux conditions de transmission, aux ressources réseau, aux capacités du terminal, aux préférences de l'utilisateur, au contexte local ou d'usage du service, etc.) afin de garantir l'accès au service et le respect de sa meilleure qualité possible.

La problématique de la fourniture de services dans les futurs réseaux mobiles a été étudiée par différentes instances de normalisation telles que VHE [VHE], OSA [OSA01], WAP [For98], MExE [MEX], etc. Toutefois, les standards et les travaux de normalisation existants ne traitent que partiellement la fourniture de services selon le point de vue d'un des acteurs suivants : terminal mobile, opérateur réseau ou fournisseur de services. En effet, il n'existe pas de solution complète qui répond aux besoins de tous ces acteurs impliqués dans le processus de fourniture de services.

Contrairement aux travaux de normalisation et comme nous le verrons dans le chapitre suivant, la plupart des propositions issues du domaine de la recherche, disposent d'une vision globale de la problématique de fourniture de services et intègrent les principaux acteurs, à savoir : l'utilisateur mobile, le terminal mobile, les fournisseurs de services et éventuellement l'opérateur réseau. Cependant, comme cette problématique combine plusieurs domaines de recherche (mobilité, hétérogénéité, personnalisation, gestion de la QoS, adaptabilité, découverte de services, gestion de services, etc.), ces solutions ne traitent qu'une partie des besoins nécessaires à la fourniture de services. Comme exemples de ces solutions, nous pouvons citer les projets : VESPER [LYBP02a], CESURE [PPM+00], CLIMATE [CLI], CAMELEON [CAM], etc.

L'objectif de notre travail est donc de concevoir et de prototyper une infrastructure de fourniture de services qui permet la gestion flexible et le déploiement facile de services. D'un point de vue usager, l'infrastructure doit permettre la découverte, le téléchargement et l'exécution de services sensibles au contexte. Elle doit également assurer une adaptabilité optimale de ces services au contexte de leur exécution tout en assurant la meilleure qualité de service possible pour l'usager.

1.2 Besoins de la fourniture de services

Pour que l'usager mobile puisse accéder aux services désirés dans un tel environnement hétérogène, mobile et dynamique, et pour que le service soit utilisable par l'usager en toutes circonstances, il est nécessaire que la plate-forme de fourniture de services globale réponde aux besoins suivants :

- Intégrer des mécanismes de gestion et de déploiement flexibles de services. Ces mécanisme doivent permettre aux fournisseurs de services d'introduire et de déployer facilement leurs services quel que soit le type du réseau utilisé.
- Fournir un moyen de recherche et de découverte de services sensibles au contexte qui permette à l'usager de trouver facilement les services les mieux adaptés aux capacités de son terminal, à ses préférences, à sa localisation et aux ressources réseau disponibles.
- Proposer un mécanisme de téléchargement qui permette aux usagers mobiles de télécharger les services sur leur terminal. Le choix de ce mécanisme doit prendre en compte les déconnexions fréquentes en environnement mobile, la mobilité des terminaux et des usagers, et enfin les capacités limitées du terminal.
- Garantir à l'usager une qualité de service acceptable même lorsque les ressources disponibles (ressources réseau ou capacités du terminal) fluctuent et que l'usager se déplace (changement de localisation), ce qui requiert d'adapter les services en cours d'exécution sur le terminal, aux conditions de leur environnement d'exécution.

1.3 Contributions de cette thèse

Cette thèse traite de la fourniture de services adaptatifs et sensibles au contexte dans les environnements mobiles. Les contributions de cette thèse sont les suivantes.

Identification des besoins d'une plate-forme de fourniture de services pour les environnements mobiles.

Dans le but de bien concevoir notre plate-forme, nous avons commencé par identifier l'ensemble des besoins imposés par la fourniture de services dans les environnements mobiles. Ces besoins sont essentiellement liés, d'une part, à la mobilité et à l'hétérogénéité des environnements mobiles, et d'autre part, à la personnalisation des services pour le compte de l'usager. Trois besoins ont été étudiés en détail le long de ce mémoire. Il s'agit du téléchargement de services, de la découverte de services sensibles au contexte et de l'adaptabilité des services au contexte de leur exécution.

Analyse de l'existant.

Nous présentons dans cette thèse, un panorama des travaux de recherche, de standardisation et d'industrie sur la fourniture de services. Nous discutons des avantages et des limites de ces travaux et nous montrons en quoi ces solutions répondent ou non aux besoins que nous soulevons dans cette thèse. Le constat est que la plupart de ces travaux traitent partiellement l'ensemble des besoins nécessaires à la fourniture flexible des services dans les environnements mobiles.

Étude de l'applicabilité des agents mobiles pour la fourniture de services dans les environnements mobiles.

Pour étudier l'intérêt des agents mobiles pour la fourniture de services, nous avons comparé les agents mobiles au modèle Java RMI. Les résultats de cette étude montrent que la solution à base d'agents mobiles s'avère intéressante en particulier dans les applications qui entraînent la consultation de plusieurs services successifs sur le réseau (ex : visite de plusieurs prestataires de services dans le cadre d'une découverte de services). Malgré leur intérêt, les agents mobiles n'ont pas été utilisés dans le développement de notre solution pour des raisons liées à la non-interopérabilité et la lourdeur des plates-formes agent mobile actuellement disponibles. Une solution de type "web services" sur HTTP a été adoptée.

Présentation d'une plate-forme de fourniture de services sensibles au contexte.

La plate-forme de fourniture de services pour environnements mobiles que nous proposons répond aux besoins étudiés dans cette thèse. Elle est basée sur les concepts des futures générations de réseaux mobiles. L'architecture réseau prévue pour supporter cette plate-forme comprend trois acteurs principaux : le terminal mobile, l'opérateur réseau et les fournisseurs de services.
Cette plate-forme a été définie dans le cadre du projet européen IST MOBIVAS [MOB].

Proposition d'un mécanisme de découverte de services sensibles au contexte.

La découverte de services permet aux services de se faire connaître par leurs fournisseurs et aux terminaux mobiles de les découvrir automatiquement.
Il existe actuellement plusieurs solutions potentielles de découverte de services comme les protocoles SLP [Gut99], Jini [Mic99], UPnP [CL99] et Salutation [Con99]. Ces solutions ne sont pas sensibles au contexte et ne prennent pas en considération les paramètres qui changent selon le contenu dynamique de l'environnement mobile. De plus, elles ne permettent pas de retrouver les services les mieux adaptés aux besoins de l'usager.
Nous proposons dans cette thèse un mécanisme de découverte de services qui permet aux fournisseurs de services de déployer et d'introduire facilement leurs services et aux usagers mobiles de personnaliser la fourniture de services et de l'adapter à leurs exigences. Ce mécanisme, mis en œuvre par un composant médiateur entre l'usager mobile et les fournisseurs de services, permet de ne proposer à l'usager que les services qui sont adaptés à ses préférences, aux capacités du terminal et à sa localisation. La liste des services proposés à l'usager contient d'autres informations qui peuvent être utiles à l'usager (description du service, tarifs indicatifs, etc.).
L'implémentation de notre mécanisme de découverte de services est basée sur l'utilisation des profils et d'outils standards tels que XML [XML98] et CC/PP (Composite Capabilities/Preference Profiles)[NHO00].

Proposition d'une solution architecturale pour l'adaptabilité des services

La plate-forme que nous proposons permet l'adaptation des services au contexte de leur exécution (besoins des usagers, caractéristiques techniques du terminal, localisation du terminal et ressources du réseau). Cette adaptation est réalisée de manière statique au moment de l'accès au service, et de manière dynamique au moment de l'exécution du

service sur le terminal mobile. La solution proposée est basée sur l'utilisation des serveurs intermédiaires (Serveur Proxy). Un service multimédia a été développé pour valider les concepts proposés dans cette thèse.

Proposition d'une solution pour la composition de services.

Afin d'assurer l'adaptabilité dynamique des services, un même service doit supporter plusieurs configurations possibles. Chaque configuration correspond à un état particulier du contexte d'exécution. Pour ce faire, nous utilisons pour la conception des services, une approche à base de composants qui offre une vue modulaire des services. Chaque service est constitué d'un ensemble de composants interconnectés, pouvant être supervisés et reconfigurés si nécessaire. Un composant est une unité autonome d'encapsulation des fonctions qui offre une interface d'accès à son traitement.
Cette approche de conception de services modulaire permet de prendre en compte les différents contextes d'exécution possibles en permettant de composer dynamiquement à chaque instant, les composants applicatifs les mieux adaptés au contexte d'exécution.

Plan de ce mémoire de thèse

Le plan de ce mémoire est le suivant. Dans le chapitre 2, nous dressons un état de l'art sur la fourniture de services dans les environnements mobiles. Nous identifions les besoins d'une plate-forme de fourniture de services et nous nous intéressons en particulier aux besoins suivants : le téléchargement de services, la découverte de services sensibles au contexte, et l'adaptabilité des services au contexte d'exécution. Nous discutons des avantages et des limites des solutions de fourniture de services existantes et nous montrons en quoi ces solutions répondent ou non aux besoins que nous fixons dans cette thèse.

Dans le chapitre 3, nous étudions l'application et la faisabilité des agents mobiles pour la fourniture de services. Nous montrons, grâce à une étude comparative avec Java RMI, pourquoi les agents mobiles sont bien adaptés à la fourniture de services. Nous introduisons un scénario basé sur les agents mobiles pour démontrer l'intérêt et les avantages de ce modèle pour la fourniture de services. Pour finir, nous présentons la solution que nous avons retenue dans notre travail et nous expliquons pourquoi les agents mobiles n'ont pas été utilisés pour le développement de cette solution.

Dans le chapitre 4, nous introduisons le projet IST MOBIVAS autour duquel la présente thèse à été réalisée. Nous présentons notre système de fourniture de services sensible au contexte CASP, et nous donnons un bref aperçu de nos propositions pour répondre aux besoins suivants : la découverte de services et l'adaptabilité des services. Ces propositions sont détaillées dans les chapitres suivants.

Le chapitre 5 s'intéresse à la découverte de services sensible au contexte. Nous présentons les différentes technologies existantes et nous étudions pour chacune d'entre elles, les avantages et les inconvénients de son applicabilité dans le domaine de la fourniture de services pour environnements mobiles. Nous présentons dans le chapitre 6 notre système de découverte de services sensible au contexte et dans le chapitre 7 son implémentation.

Dans le chapitre 8, nous nous intéressons à l'adaptabilité dynamique des services à leur contexte d'exécution. Une large place est consacrée à un état de l'art sur l'adaptabilité dans les systèmes logiciels. Nous présentons les principes généraux de ce concept ainsi que les différents travaux menés autour de cette problématique. Nous passons en revue les différentes propositions prenant en compte l'adaptabilité pour la fourniture de services dans les environnements mobiles.

Le chapitre 9 décrit comment notre infrastructure supporte l'adaptabilité des services au contexte d'exécution, le prototype réalisé, l'application que nous avons implantée ainsi que son évaluation.

Le chapitre 10 est consacré aux perspectives de recherche de nos travaux qui sont l'adaptabilité dynamique basée sur la composabilité de services.

Pour finir, nous concluons cette thèse dans le chapitre 11 en résumant les principales contributions et en présentant nos futurs travaux de recherche.

Les publications associées à cette thèse sont présentées dans l'annexe C.

Chapitre 2

État de l'art et motivations

Pour pouvoir offrir des services multi-réseaux, multi-terminaux, adaptables et portables, il est nécessaire de déployer de nouvelles architectures qui feront le lien entre les fournisseurs de services et les utilisateurs mobiles connectés au réseau. Ces architectures doivent être utilisables dans des contextes très divers et doivent répondre aux différents besoins liés à la nature des environnements mobiles et aux spécificités de l'application de fourniture de services.
Il existe une très large gamme de travaux issus des domaines de normalisation, de recherche ou d'industrie qui répondent à la problématique de la fourniture de services pour environnements mobiles. Nous proposons dans ce chapitre de passer en revue ces solutions.
Ainsi dans la partie 2.1, nous fournissons certains concepts et définitions liés à la fourniture de services et aux environnements mobiles. Dans la partie 2.2, nous identifions les principaux besoins qu'une plate-forme de fourniture de services doit assurer. Les parties 2.3 et 2.4 sont dédiées à la présentation des différentes technologies existantes et aux travaux de recherche relatifs à notre problématique. Enfin, nous concluons dans la partie 2.5 en revenant sur les motivations de cette thèse.

2.1 Concepts généraux

Ce paragraphe définit le vocabulaire que nous utiliserons dans la suite de cette étude.

2.1.1 Quelques définitions

1. **Service**
   Un *service* est un ensemble de fonctions (liées aux applications, fonctions de télécommunications, contenus, produits, etc.) offertes aux utilisateurs par des fournisseurs, selon un accord de service implicite ou explicite [SPW+02].
   Dans la cadre de la fourniture de services, les *services à valeur ajoutée* sont des services offerts par un support réseau mais qui sont développés par des tiers (entités externes au support réseau). Ces services peuvent utiliser les fonctions que le support réseau offre telle que la localisation de l'utilisateur.

2. **Environnements mobiles**
   Dans notre travail, nous désignons par *environnement mobile* ou *nomade* tout environnement faisant référence à la mobilité de l'utilisateur et/ou la mobilité du terminal. C'est un système composé de sites mobiles qui permet à ses utilisateurs d'accéder à l'information indépendamment de leurs positions physiques.
   Un environnement mobile peut être vu comme une intégration de terminaux portables et d'un réseau sans fil, ou d'une combinaison de terminaux portables et d'un réseau fixe (où la connexion est filaire mais non statique). Dans les deux cas, la connexion est temporaire avec possibilité de déconnexions [Ngu00].

2.1.2 Les acteurs de la fourniture de services

Les acteurs prenant part à la fourniture de services dans un environnement mobile sont nombreux : ils vont du concepteur à l'utilisateur du service. Nous ne considérons ici que ceux qui entrent en jeu au moment de la fourniture du service. Ces mêmes acteurs seront utilisés dans notre architecture comme nous le verrons dans les chapitres suivants. Ils sont de trois types :

1. **Usager ou utilisateur mobile**
   L'utilisateur final du service, il a souscrit à l'utilisation de services. Il possède des informations d'accès individuelles lui permettant d'utiliser les services à partir de tout terminal mobile.
2. **Opérateur réseau**
   L'opérateur réseau gère l'abonnement de l'usager. Il fournit les informations d'accès à chaque abonné ainsi que l'infrastructure d'exécution et les applicatifs nécessaires à l'exécution du service. Il peut également proposer ses propres services aux utilisateurs mobiles.
3. **Fournisseur de services**
   L'opérateur réseau fournit également des services proposés par des fournisseurs de services indépendants. L'usager n'a pas nécessairement connaissance de l'existence de ces fournisseurs de services, car l'opérateur réseau fait l'intermédiaire et est en mesure de masquer son existence.

2.1.3 Exemples de services offerts

Cette partie donne un aperçu non exhaustif des services particulièrement pertinents dans le cadre des futurs réseaux. Les services sont amenés à évoluer fortement et de nouveaux services non connus à ce jour apparaîtront [arc02].

1. *Les services fournis par des tiers*

   Les réseaux de nouvelle génération, par leur ouverture, doivent favoriser l'évolution de l'utilisation des services considérés comme classiques dans le monde informatique. Ces services peuvent être gérés par des fournisseurs de services à travers le réseau.

2. *Le e-commerce*

   L'évolution des terminaux et le développement de solutions de paiement sécurisé devraient favoriser le commerce en ligne.

3. *La messagerie unifiée*

   Le service de messagerie unifiée est l'un des services les plus avancés. C'est le premier exemple de convergence et d'accès à l'information à partir de différents moyens d'accès. Le principe est de centraliser tous les types de messages : vocaux (téléphoniques), écrits (email, SMS) ou multimédia sur un serveur. Ce dernier ayant la charge de fournir un accès aux messages adapté au type du terminal de l'utilisateur.

4. *La diffusion de contenus multimédia*

   La diffusion de services multimédia commence à émerger et prendra une place de plus en plus importante avec l'évolution de la bande passante et les capacités de traitement des terminaux.

5. *Les services associés à la localisation*

   La possibilité de localiser les terminaux mobiles a été rapidement perçue comme une source de revenus supplémentaires pour les opérateurs réseau et les fournisseurs de services. En effet, la localisation permet de proposer aux utilisateurs finaux des services très ciblés à haute valeur ajoutée liés à la localisation du terminal.

2.1.4 Organismes de standardisation et standards existants

Avant de présenter les différentes plates-formes et technologies liées à la fourniture de services pour les futurs réseaux, nous présentons d'abord les organismes de normalisation et les standards existants qui nous intéressent le plus.

2.1.4.1 3GPP

Le 3GPP (3^{rd} Generation Partnership Project) [3GP] est le fruit d'une collaboration de plusieurs organismes de standardisation ainsi qu'un grand nombre de constructeurs ayant pour objectif de définir un ensemble de spécifications techniques pour les systèmes de $3^{ième}$ génération de réseaux mobiles (3G). En particulier, le 3GPP vise la définition d'une architecture permettant d'offrir le concept VHE (Virtual Home Environment). Il identifie le modèle OSA (Open Service Access) comme un des outils nécessaires à la mise en place de ce concept. Ces deux standards sont présentés dans les sections suivantes.

2.1.4.2 UMTS

L'UMTS (Universal Mobile Telecommunications System) [Les00] désigne une technologie cellulaire numérique de troisième génération en cours d'élaboration. Atteignant 2 millions de bits par seconde (2 Mb/s) dans certaines conditions, les vitesses de transmission seront nettement plus élevées que celles des réseaux GSM ou GPRS qui plafonnent les 9600 bits par seconde (9.6 Kb/s).
L'arrivée de la transmission de données à haut débit dans l'UMTS annonce une révolution

aussi importante que celle représentée par le développement d'Internet pour les ordinateurs de bureau. Ainsi, les usagers mobiles pourront non seulement accéder aux services vocaux, de fax et de données conventionnels, mais aussi à des services multimédia comme sur Internet. L'UMTS offrira initialement toute la gamme des services GSM, progressivement enrichie par de nouveaux services multimédia.

2.1.4.3 Virtual Home Environment (VHE)

Le VHE (*Virtual Home Environement*) [VHE] ou l'*Environnement de Service Personnalisé*, est un concept d'environnement de service personnalisé qui a été défini par le 3GPP dans le cadre de la normalisation de l'UMTS.
VHE définit un système qui permet à des utilisateurs nomades d'accéder et d'utiliser, d'une manière personnalisée, un ensemble de services quelques soient le réseau d'accès et le terminal utilisé (dans les limites et les capacités du réseau et du terminal) et avec les mêmes paramètres de personnalisation. Autrement dit, le VHE supporte l'idée d'une universalité du service qui s'adapte, d'une part aux paramètres de personnalisation de l'utilisateur, et d'autre part à son terminal et son réseau d'accès [Sab03].

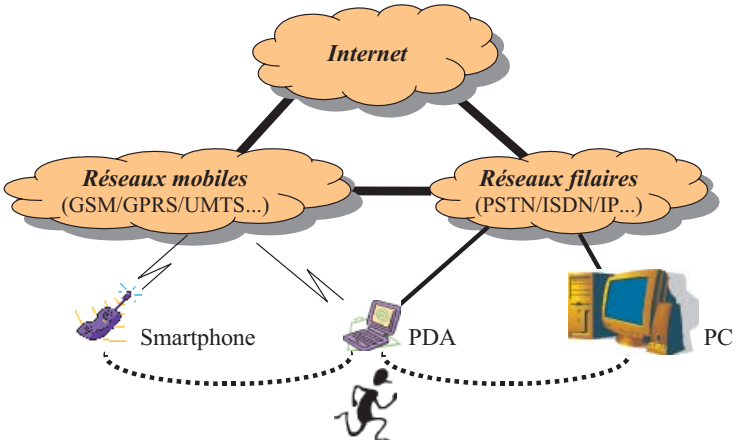


FIG. 2.1 – Le concept VHE

Dans son principe, le VHE n'a rien de spécifique aux réseaux mobiles. Donc il pourrait être aisément transposé au domaine des réseaux fixes.

Le VHE permet à un utilisateur de se voir offrir un environnement de services sur un site visité identique à celui auquel il a souscrit dans son abonnement avec son réseau nominal. Cet environnement peut s'appliquer à son environnement réel, que se soit chez lui ou au bureau. Sur son terminal, mobile ou non, il dispose donc du même environnement. Le VHE assure ainsi :

- *La personnalisation des services* pour des utilisateurs ayant des préférences et des besoins différents.

– *La portabilité de l'environnemnt personnalisé* à travers divers réseaux d'accès (fixe, mobile, IP) et sur des terminaux de natures et de capacités différentes.

Les informations individuelles de l'utilisateur qui déterminent comment les services personnalisés sont fournis et présentés, sont contenues dans un environnement de service personnalisé présenté sous forme de *profils* de l'utilisateur. Chaque profil consiste en un ensemble d'informations liées aux interfaces graphiques de l'utilisateur et à ses services.
Puisque les utilisateurs peuvent avoir plusieurs rôles pour différents besoins et différentes situations (être au travail, à la maison ou dans la voiture), un même utilisateur peut avoir plusieurs profils.
Dans VHE, les profils de l'utilisateur sont fournis et contrôlés par son réseau nominal qui peut avoir des contrats d'accord avec des fournisseurs de services à valeur ajoutée.
En plus des services offerts par le réseau nominal, l'utilisateur peut utiliser d'autres services fournis par des fournisseurs de services au sein des réseaux visités (autres que son réseau nominal).
La figure 2.2 illustre le concept de VHE [Zui02]. L'environnement nominal HE (Home Environment) gère tous les services qui sont accessibles dans le réseau nominal. L'environnement nominal a un accès aux profils de l'utilisateur pour adapter tous les services aux préférences de l'utilisateur. Le VHE exporte les services de l'environnement nominal et adapte les services aux capacités du réseau visité et du terminal utilisé.

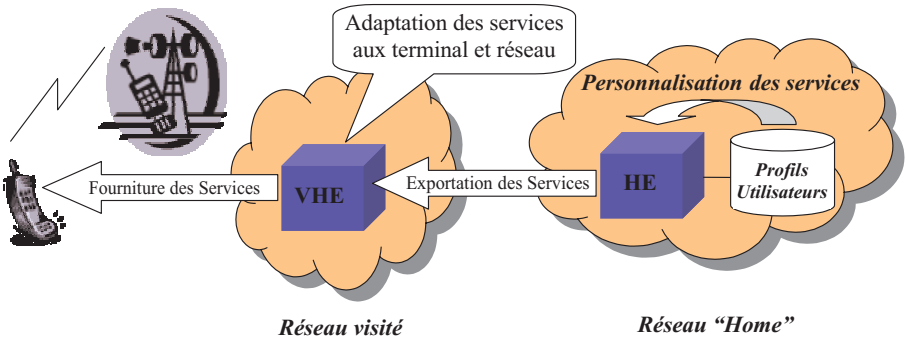


FIG. 2.2 – Le VHE et l'environnement nominal

2.2 Besoins de la fourniture de services

Les principales contraintes techniques que le concepteur d'une plate-forme de fourniture de services doit prendre en compte sont présentées dans cette section.

2.2.1 Mobilité

Le concept de mobilité dans les futurs réseau de communication renvoie à *la mobilité globale* qui regroupe *la mobilité personnelle*, *la mobilité du terminal* et *la mobilité de*

services.

1. **La mobilité personnelle** correspond à la capacité de l'utilisateur d'accéder à des services personnalisés à partir de n'importe quel terminal (fixe ou mobile) et la capacité du réseau de fournir ces services selon les préférences de l'utilisateur. La mobilité personnelle permet aux utilisateurs d'utiliser leurs services personnalisés indepen-demment du terminal utilisé.
2. **La mobilité du terminal** correspond à l'aptitude du terminal à accéder aux services quels que soient l'endroit où il se trouve et la vitesse de son déplacement.
3. **La mobilité de services**, aussi appelés **portabilité de services**, fait référence à la capacité du réseau à fournir les services souscrits à l'endroit où se trouvent le terminal et l'utilisateur. Les services exacts que l'utilisateur peut demander sur son terminal dépendent des propriétés du terminal (que l'on appelle les capacités du terminal) et du réseau qui sert ce terminal.

Un système de fourniture de services doit prendre en charge les utilisateurs itinérants en leur permettant d'accéder aux services fournis toujours de la même façon, même s'ils se déplacent. Grâce à cette fonctionnalité, l'utilisateur du système peut utiliser ses services dans n'importe quel réseau qu'il visite de la même manière et avec les mêmes caractéristiques que lorsqu'il se trouve dans son réseau d'abonnement. Il dispose donc d'un environnement personnalisé de services qui le suit partout où il se déplace.

2.2.2 Choix du paradigme de communication

Un des premiers aspects à considérer dans la conception d'une architecture distribuée est le choix du paradigme de communication à utiliser entre les différentes entités constituant l'environnement. Ce choix doit en premier lieu prendre en compte les besoins de l'environnement et de la plate-forme à développer.

Les environnements mobiles sont caractérisés par la mobilité des terminaux et par la variabilité de leurs supports de communication. Les terminaux mobiles peuvent se déplacer et se connecter au réseau de manière intermittente depuis des endroits différents. Ils peuvent, par exemple, utiliser une connexion sans fil pour communiquer pendant un déplacement et un point d'accès fixe au réseau pour se reconnecter après un déplacement.
Les supports de communication présentent des caractéristiques différentes de débit, de latence et de coûts. Il est important donc, pour la plate-forme, de cacher les particularités des réseaux et d'optimiser l'usage des connexions et leur coût.
Un autre problème crucial pour les terminaux mobiles est celui des performances de communication. Une connexion réseau peut devenir une ressource critique qui limite les performances perçues par l'utilisateur. Le choix du paradigme de communication doit proposer des moyens pour apporter un gain de performance significatif.

Les environnements mobiles constituent un des domaines les plus adaptés à l'utilisation des agents mobiles [FLP98, PK98]. En effet, les possibilités offertes par les agents mobiles

à savoir, le déplacement du traitement à l'endroit où se trouvent les données, et le fonctionnement asynchrone (le client peut se déconnecter pendant le traitement) sont compatibles avec les besoins des terminaux mobiles utilisant des liaisons lentes et coûteuses.

Dans le cadre de la fourniture de services, les agents mobiles peuvent être utilisés pour la fourniture instantanée de services, la recherche du service le plus adapté aux préférences de l'utilisateur, la personnalisation, la répartition de charge et la gestion des déconnexions.

2.2.3 Déploiement et gestion des services

Les fournisseurs de services ont intérêt à ce que leurs services soient utilisables depuis tous les terminaux d'accès et quel que soit le réseau utilisé. Ceci pose le problème de la portabilité des services dans des environnements différents.
L'utilisation d'une plate-forme de gestion de services qui joue le rôle de médiateur entre les différentes entités impliquées dans le processus de fourniture de service (terminaux, fournisseurs de services et opérateur réseau) est donc nécessaire. Cette plate-forme doit fournir des mécanismes pour assurer le déploiement et la gestion dynamiques des services dans n'importe quel environnement. Ces mécanismes regroupent l'ensemble des procédures nécessaires à l'introduction des services dans la plate-forme de fourniture de services ainsi que leur mise à jour (enregistrement de nouveaux services, mise à jour des informations associées aux services proposés, annonce de la disponibilité des services aux utilisateurs mobiles, suppression de services, etc.).

2.2.4 Personnalisation de services

La personnalisation des services offre aux utilisateurs le moyen de définir et modifier la manière dont les services leurs sont livrés afin de satisfaire leurs besoins.

La personnalisation est un des éléments clés d'un système de fourniture de services pour les futurs réseaux. Un tel système doit offrir à l'utilisateur un environnement de travail personnalisé construit sur la base de la notion de *profils*. En essence, un profil est une représentation structurée des besoins de l'utilisateur. Il comprend un ensemble de paramètres que l'utilisateur aura spécifiés sous forme de *préférences* soit dans son contrat de souscription soit au cours de son utilisation du système. L'utilisateur doit pouvoir :

- Créer un nombre quelconque de profils,
- Modifier ces profils,
- Sélectionner le profil qu'il veut appliquer à son environnement de travail et que l'on appelle *profil actif*. Un profil est dit actif si l'environnement courant de l'utilisateur respecte les préférences contenues dans ce profil.

2.2.5 Sensibité au contexte

Plusieurs définitions ont été proposées pour décrire ce qu'est le *contexte* dans le domaine de l'informatique. Schilit [Sch95], l'inventeur du terme *context-aware computing*, définit le contexte par rapport à la localisation, les identités des objets et personnes environnantes.

Brown [BBX97] définit le contexte comme étant les éléments de l'environnement de l'utilisateur que son ordinateur connaît. Il rajoute à la définition de Schilit la notion de temps, de saison, de température, etc.
Ce n'est qu'avec Dey [Dey00] que nous avons une définition précise du contexte :

> *"Le contexte est toute information qui peut être employée pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet qui est en rapport avec l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes".*

Dans [Win01], l'auteur fait une distinction entre le contexte et l'environnement. Il considère que le contexte est un terme opérationnel. Une entité ne peut faire partie du contexte que si elle peut être utilisée dans une interprétation, et non pour ses propriétés inhérentes. Le voltage d'une prise électronique par exemple, fait partie du contexte s'il y a des actions de l'utilisateur ou de l'application dont l'interprétation dépend du voltage, sinon cela fait juste partie de l'environnement.

La gestion du contexte est un élément essentiel de l'informatique mobile. En effet, il n'est plus possible de configurer les applications en une unique fois à l'initialisation du programme car la mobilité induit des changements permanents dans le contexte. Les informations sur le contexte sont obtenues grâce à des sondes. Chaque sonde est responsable de l'acquisition d'un type particulier d'informations.

Il existe deux modes de gestion du contexte dans les environnements mobiles :

- La première approche consiste à fonctionner comme dans les systèmes répartis fixes et à tenter de cacher toutes les variations du contexte aux applications. L'infrastructure masque alors les effets de la mobilité aux applications (indispensabilité de certains services, de certaines données, changement d'adresse, etc).
  Cette approche a l'avantage de simplifier la tâche au développeur d'application qui ne se soucie pas des problèmes de l'informatique mobile, et pourra programmer ses applications comme si elles étaient fixes. Cette approche est appelée : **informatique insensible au contexte**.
- La seconde approche consiste à notifier aux applications les changements de contexte. Cela permet aux applications de s'adapter aux nouveaux contextes. Elles pourront par exemple découvrir et utiliser les services locaux, accéder à des informations locales, éviter de faire appel à des services indisponibles, etc.
  Cette approche, même si elle paraît plus compliquée d'usage pour le programmeur, permet à l'application d'optimiser son comportement en fonction du contexte dans lequel elle se trouve. De plus, ce paradigme permet de définir une nouvelle classe d'applications. Ce modèle est appelé **informatique sensible au contexte** (*"Context-aware computing"*).

2.2.6 Découverte de services sensible au contexte

Compte tenu du nombre, de la variabilité et de la diversité des services qui seront disponibles aux utilisateurs mobiles dans les futurs réseaux de communication, il s'avère

indispensable de concevoir un mécanisme intelligent, efficace et flexible pour la découverte et la fourniture personnalisées des services aux utilisateurs mobiles. De plus, les utilisateurs veulent pouvoir accéder à ces services de la même manière quel que soit le type du terminal utilisé et quelles que soient ses capacités.
La plate-forme de fourniture de services doit donc offrir des mécanismes intelligents de découverte et de fourniture de services qui permettent à chaque utilisateur de localiser et d'exécuter de manière simple et efficace, des services qui répondent à ses besoins ainsi qu'aux caractéristiques du contexte de la fourniture de service (capacités du terminal, ressources réseau, localisation du terminal, etc.).
La conception d'un tel mécanisme nécessite l'utilisation d'outils sophistiqués pour la gestion des préférences de l'utilisateur, la représentation et la gestion des capacités du terminal ainsi que la gestion du contexte.

2.2.7 Adaptabilité

Plusieurs raisons appellent à l'adaptabilité des services dans les environnements mobiles. Ce type d'environnements présente une hétérogénéité importante, une grande variabilité et de nombreuses possibilités d'évolution, aussi bien au niveau des moyens d'exécution que des moyens de communication. En effet, les ressources offertes au niveau du terminal peuvent être extrêmement différentes selon que l'on utilise un assistant personnel, un ordinateur portable, un téléphone portable ou une station de travail. Il est nécessaire donc de réaliser la fourniture de services en fonction du type du terminal.
Les réseaux de communication présentent des infrastructures et des performances bien différentes et très hétérogènes (réseaux filaires, réseaux sans fil). De plus, ces éléments sont soumis à d'importantes variations au cours du temps. La disponibilité des ressources peut également varier considérablement dans le temps.
En plus de ces caratéristiques liées à la nature des environnements mobiles, l'utilisateur mobile veut pouvoir accéder, à partir de différents points de connexion, à des services personnalisés auxquels il a souscrit et qui peuvent éventuellement prendre en compte la localisation du terminal. Ainsi, la fourniture et l'exécution des services doivent nécessairement prendre en compte le contexte de l'usager et ses besoins pour garantir le respect de la qualité de service souhaitée.

L'adaptabilité dans le contexte de notre travail est la capacité d'harmoniser le service avec son environnement. Elle concerne simultanément la prise en compte du contexte d'exécution (les capacités du terminal, les ressources réseau, la localisation du terminal) et les fonctionnalités requises par l'utilisateur (besoins et préférences de l'utilisateur) afin de garantir le respect de la qualité du service.

L'adaptabilité suppose implicitement la capacité d'un système logiciel à être modulaire et composable. Un système monolithique ne peut être modifié qu'en le remplaçant intégralement. Dans un système construit sous forme de modules interconnectés, il est possible de modifier ou de remplacer certaines parties du système avec un minimum d'interférences avec le reste du système. Par opposition au terme monolothique, la *modularité* est une mesure de la séparation du système en ses éléments constituants. La *composabilité* est une

mesure des capacités d'assemblage entre les éléments constituants le système. La combinaison de ces deux propriétés autorise alors l'adaptabilité [Led01].
Nous nous intéressons dans notre travail à l'adaptabilité des services basée sur la composabilité.

2.3 Technologies existantes liées à la fourniture de services

Les technologies et les standards de service existants (ou leur combinaison) peuvent être utilisées comme un point de départ pour l'identification de la technologie de service sur laquelle la fourniture de services à valeur ajoutée sera basée [MOB00, Pon01, JJ02].

La problématique de la fourniture de services dans les futurs réseaux mobiles a été relevée par différentes instances de normalisation. Plusieurs technologies et travaux de standardisation ont traité le problème de la fourniture de services selon différents points de vue : selon le point de vue du client (ou du terminal), selon le point de vue du réseau de l'opérateur ou selon le point de vue du fournisseur de services.
Cette section présente un panorama de technologies et de standards qui permettent la réalisation des différents aspects de la fourniture de services. Dans la section 2.3.1, nous présentons les solutions et les architectures orientées réseau qui s'intéressent aux problèmes liés à la fourniture et au contrôle de services dans des réseaux hétérogènes. Les solutions orientées client ou terminal, qui s'intéressent à l'adaptabilité des services aux différents terminaux mobiles, sont présentées dans la section 2.3.2. Enfin, dans la section 2.3.3, nous décrivons les solutions orientées fournisseur de services qui fournissent des interfaces réseau permettant aux fournisseurs de services d'accéder aux fonctionnalités et aux capacités du réseau quelle que soit la nature de ce dernier.

2.3.1 Fourniture de services dans des réseaux hétérogènes

Afin de supporter tous les besoins de fourniture de services pour les futurs réseaux mobiles, les mécanismes de gestion de mobilité et de contrôle de services doivent pouvoir opérer dans divers réseaux.
CAMEL (Customized Applications for Mobile network Enhanced Logic) [CAM01] s'intéresse à la fourniture de services pour des utilisateurs en cas de nomadicité (roaming). Il permet de répondre à deux types de besoins. D'une part, il permet de réaliser des services GSM non normalisés. D'autre part, CAMEL est utilisé pour fournir aux abonnés mobiles, en cas de roaming hors de leur réseau nominal, des services indépendants des réseaux visités. Toutefois, CAMEL ne supporte que les services GSM spécifiques à l'opérateur et ne supporte pas la fourniture de services tiers.
Plusieurs intergiciels de télécommunication ont émergé pour combiner les services issus du domaine des télécommunications et d'Internet.
SPIRITS (Service in the Public switched telephone network/IN Requesting InTernet Service) [GBF+03] propose des mécanismes qui permettent à un réseau téléphonique de supporter des services Internet.
PINT (Public switched telephone network/Internet INTernetworking) [PC00] est l'inverse

de SPIRITS. Il permet aux serveurs Internet d'accéder à des services proposés par des réseaux téléphoniques.

2.3.2 Environnements d'exécution de services

Les terminaux jouent un rôle clé tant dans le développement des nouveaux services que des nouvelles architectures des futurs réseaux mobiles.
Des environnements standards d'exécution doivent être utilisés pour permettre aux services de s'exécuter indépendamment des plates-formes embarquées sur les terminaux. Dans ce qui suit, nous développons quelques exemples de technologies et de standards d'environnements d'exécution de services conçus pour les terminaux mobiles.

2.3.2.1 WAP

Le protocole WAP (Wireless Application Protocol) [For98] offre l'accès à Internet à partir de terminaux mobiles. Il permet de visualiser du contenu issu du web sur l'écran de quelques lignes d'un téléphone portable ou d'un assistant numérique.
A la fois mobile et personnel, le terminal compatible WAP ouvre un lien direct privilégié entre l'utilisateur et les services tels que le commerce mobile (ou m-commerce) ou les services bancaires en ligne. Plus techniquement, le WAP standardise l'échange d'informations entre le terminal mobile et une passerelle qui assure la liaison avec Internet. Cette passerelle assure la conversion de données entre le monde Internet et le monde du réseau mobile. Tout comme pour le web, le WAP a été conçu dans une approche client-serveur. Le terminal mobile incorpore un navigateur léger qui communique avec un serveur WAP. Les ressources des terminaux mobiles actuels étant limitées, le traitement des données est principalement assuré côté serveur. Les protocoles WAP se situent au dessus de la couche matérielle, le WAP est par conséquent indépendant du type de réseau de communication utilisé (GSM, GPRS et UMTS) ce qui lui procure une grande souplesse et une forte compatibilité au niveau de ses applications.
Un des problèmes majeurs de WAP réside dans l'incapacité de sa passerelle WAP à déterminer et extraire les parties d'une page web qui contiennent les informations essentielles à traduire, en particulier si ces pages web sont riches en terme de contenus graphiques. De plus, pour des rasions liées à la limitation des capacités des terminaux, les contenus graphiques ne peuvent pas être affichés sur les écrans des terminaux mobiles actuellement disponibles.

2.3.2.2 iMode

Le iMode (pour *information Mode*) [RCCC01] est un système élaboré par la société japonaise NTT-Docomo pour permettre l'accès à des services interactifs depuis un téléphone mobile. C'est une forme d'Internet mobile, tout comme le WAP. Le téléphone interprète des pages HTML simplifiées et adaptées à la taille de l'écran d'un téléphone portable.
La standard iMode utilise le langage C-HTML (Compact-HTML) et non WML du WAP. Le C-HTML est un sous-ensemble de HTML destiné à fonctionner sur des navigateurs

mobiles allégés.
Comme C-HTML est basé sur HTML, il ne nécessite pas d'étape intermédiaire de conversion, à l'inverse de WAP qui réclame l'emploi d'une passerelle. L'accès aux pages se fait en direct via les couches basses du réseau.
Le iMode est un service Internet mobile contrairement à WAP qui est un protocole décrivant les différentes couches d'une communication mobile.

2.3.2.3 MExE

MExE (Mobile station application EXecution Environment) [MEX] est une spécification technique émanant du 3GPP qui a pour but de décrire un environnement applicatif standardisé pour les terminaux mobiles de 3G. Il offre aux terminaux la possibilité de négocier leurs capacités avec les fournisseurs de services MExE, permettant ainsi aux applications d'être développées indépendamment des plates-formes des terminaux et d'être adaptées aux capacités du terminal.
MExE inclut un ensemble de technologies dont Java et le protocole WAP. Il spécifie un cadre gérant les aspects suivants : la gestion des profils des utilisateurs, l'adaptation du contenu aux capacités du terminal et la gestion du processus de négociation des capacités entre le réseau et le terminal.
MExE a défini trois classes de terminaux comme le montre la figure 2.3.

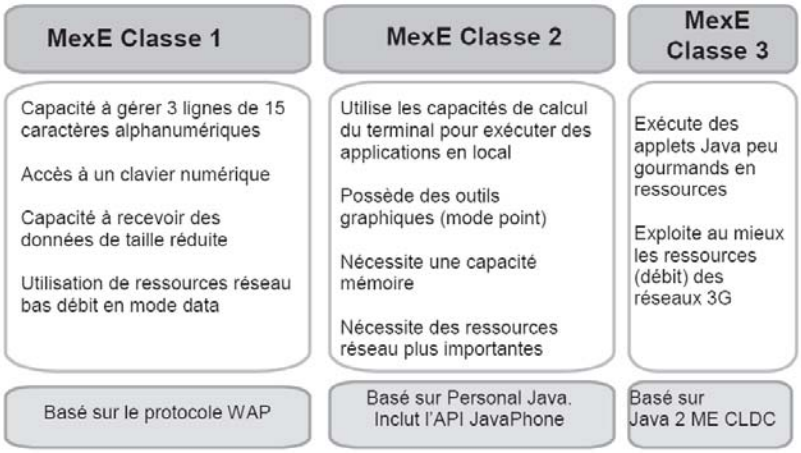


FIG. 2.3 – Les trois classes spécifiées par MExE

Il faut noter ici que :

- PersonalJava [PER] est un environnement applicatif standard Java optimisé pour les terminaux accédant à Internet et destiné au grand public.
- JavaPhone [JAV] est une extention de la plate-forme PersonalJava qui inclut les APIs gérant la téléphonie et la messagerie.

- Java 2 Micro Edition (ME) Connected Limited Device Configuration (CLDC) [CLD] est une évolution de la plate-forme Java destinée à la micro informatique embarquée. Elle s'adresse aux appareils de consommation courante et aux appareils de taille réduite. Elle inclut des outils pouvant gérer divers profils et sessions.

2.3.2.4 SIM Application Toolkit

Subscriber Identity Module Application Toolkit (SAT) [3GP01] est une technologie basée sur Java qui utilise la topologie client-serveur. Elle permet à des opérateurs réseaux d'envoyer des applications à travers des messages SMS (Short Message Service) [SMS00] pour mettre à jour les cartes SIM des utilisateurs avec des services nouveaux ou modifiés. SAT donne la possibilité d'entrer des données supplémentaires dans la carte SIM des terminaux mobiles à tout moment. Par exemple, un opérateur peut personnaliser la carte SIM de chaque utilisateur en envoyant le code source par SMS. SAT est aussi un bon outil pour permettre de lire des informations sur l'identité du client (cas du m-commerce ou des services bancaires).
SAT offre la possibilité de modifier certaines informations embarquées, de télécharger des applications de taille raisonnable, de les exécuter et de personnaliser le menu d'un téléphone.

2.3.3 Plates-formes unifiées et interfaces ouvertes

La standardisation dans le cadre de la fourniture de services pour les futurs réseaux mobiles propose l'utilisation d'architectures en couches. Selon cette approche, des interfaces standardisées sont utilisées entre la couche réseau constituée d'éléments réseau sous le contrôle de l'opérateur et la couche service constituée des serveurs tiers qui gèrent les services.
La séparation de la couche service des caractéristiques réseau permet aux fournisseurs de créer des services adaptés aux divers réseaux. De plus, l'utilisation d'interfaces standards permet aux opérateurs d'ouvrir leurs réseaux et de fournir leurs ressources aux applications de manière consistante.

La nécessité d'offrir des services multi-réseaux et multi-terminaux pousse à une transformation de l'architecture des plates-formes de services. Deux modèles principaux émergent actuellement :

- Une architecture basée sur l'interface de services normalisée du *modèle OSA/Parlay*. Ce modèle est plutôt adapté pour des services de type "télécoms".
- Un *modèle orienté "Web Services"*, basé sur des technologies et des protocoles issus du monde Internet. C'est une architecture de fourniture de services distribuée, plus adaptée à la fourniture de services avec une coopération forte du terminal.

2.3.3.1 Le modèle OSA/Parlay

L'organisme de standardisation 3GPP a proposé l'architecture **OSA** (*Open Service Access*) [OSA01] comme architecture de base pour le concept VHE. OSA est très proche

d'un autre framework appelé **Parlay** [Par00a, Par00b] introduit par le *Groupe Parlay* [GPa]. Actuellement, 3GPP travaille avec le Groupe Parlay pour aligner les deux standards. Le terme **OSA/Parlay** est alors utilisé pour désigner le fruit de ce travail.

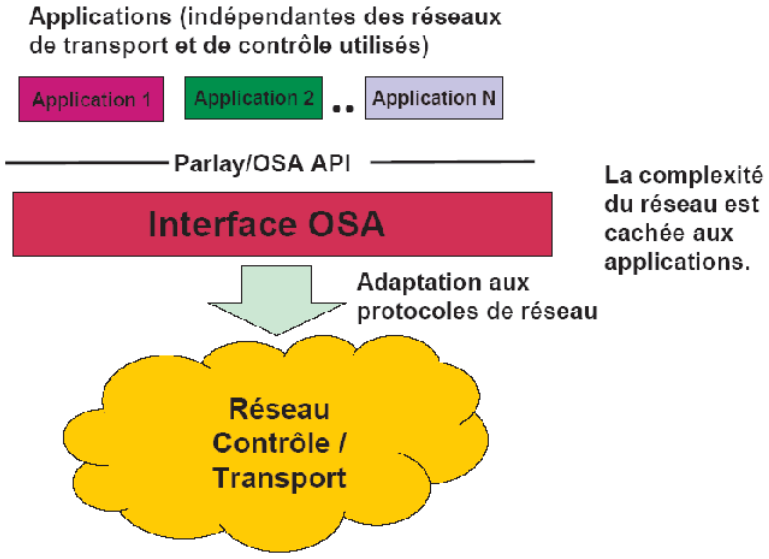


FIG. 2.4 – Rôle de l'interface OSA

OSA/Parlay propose une architecture permettant aux applications de services d'utiliser les ressources du réseau tout en leur masquant la complexité du réseau, de ses protocoles ainsi que de ses implémentations spécifiques. Concrètement, l'interface standardisée ouverte OSA/Parlay :

- est le lien entre les plates-formes de services et les couches basses du réseau et facilite l'accès au réseau pour les applications de services.
- offre des outils nécessaires au bon fonctionnement des services, tels que la gestion de la sécurité (authentification, autorisation) et la gestion des utilisateurs (profil, état).

2.3.3.2 Les Web Services

Alors que le modèle OSA/Parlay est orienté vers des architectures centrées sur les services télécoms, un autre modèle de fourniture de services pour les futurs réseaux apparaît. Celui-ci, basé sur une architecture et des protocoles orientés web, définit un modèle beaucoup plus distribué : ce sont les *Web Services* [Kre01, GSB+01].
Un *Web Service* est une interface ouverte qui décrit un ensemble d'opérations qui sont accessibles dans le réseau en utilisant un format standard de messages XML (Extensible Markup Language) [XML98]. La description du Web Service définit l'ensemble des interactions avec le service, incluant le format des messages et les protocoles de transport et de

localisation. Cette interface permet d'implémenter les services indépendamment des plates-formes logicielles et matérielles sur lesquelles ils sont installés et des langages dans lesquels ils ont été écrits.

Les Web Services sont basés sur des standards existants ou émergents tels que HTTP, XML, SOAP (Simple Object Access Protocol) [SOA], WSDL (Web Services Description Language) [WSD] et UDDI (Universal Description, Discovery and Integration) [UDD] :

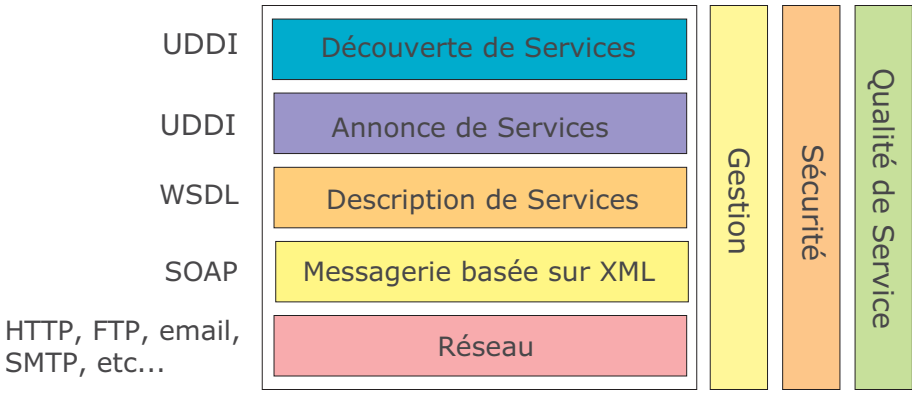


FIG. 2.5 – Suite de protocoles impliqués dans l'implémentation des Web Services

- **SOAP** définit la structure du message XML utilisé par les Web Services pour dialoguer entre-eux et automatiser ce dialogue. Il repose sur l'utilisation combinée de XML pour la structuration des requêtes et des réponses et de HTTP pour l'invocation de méthodes.
- **WSDL** est un format de description des composants (c'est-à-dire des services eux-mêmes) invocables par le biais de messages XML au format SOAP. Il permet de reconnaître les schémas XML utilisés et d'établir une connexion entre le client et le fournisseur.
- **UDDI** s'appuie lui-même sur des Web Services pour proposer un annuaire mondial d'entreprises. Il fournit ainsi, un outil pour communiquer tout type de coordonnées associées aux Web Services (adresse géographique, numéro de téléphone, fax, adresse de site, etc.).

2.3.3.3 Comparaison des modèles OSA/Parlay et les Web Services

Les deux modèles OSA/Parlay et Web Services ont un même objectif : permettre un accès personnalisé aux services. Ils utilisent une couche d'adaptation permettant de gérer l'interface entre les applications fournissant les services et les ressources réseau. Il s'agit de l'interface API OSA/Parlay pour le modèle OSA et une suite de protocoles (SOAP, WSDL et UDDI) pour les Web Services. Cependant, ces deux modèles de services diffèrent dans leur architectures :

- Le modèle OSA/Parlay a été soutenu par un grand nombre de constructeurs et d'opérateurs télécoms. En effet, cette architecture permet l'évolution des réseaux télécoms actuels.
- Le modèle préconisé par le W3C, les Web Services est pour sa part avancé et déjà mis en œuvre par les acteurs venant du monde Internet et informatique. Cette approche est basée sur des technologies et des protocoles issus du monde Internet avec une architecture distribuée.
- L'approche OSA/Parlay est incontournable pour les acteurs télécoms établis évoluant vers les futurs réseaux mobiles. Cependant, le succès de ce modèle se confirmera (ou s'infirmera) vraiment avec l'essor des réseaux et services UMTS.
- L'approche des Web Services est moins complexe à implémenter et donc plus accessible à des fournisseurs de services tiers que l'interface OSA/Parlay. La preuve en est l'existence à ce jour des premiers services issus de ce modèle [Bob01].

Les deux modèles présentés dans cette partie, offrent aux fournisseurs de services des moyens leur permettant de créer des services multi-réseaux. Ils s'intéressent en particulier, à la création et la gestion des services. Cependant, ils ne répondent pas à d'autres besoins nécessaires à la fourniture de services tels que la sensibilité au contexte, l'adaptabilité dynamique et la gestion de la mobilité.

2.4 Analyse de l'existant

Nous présentons dans cette section un panorama des travaux de recherche sur la fourniture de services pour les futurs réseaux mobiles. Nous discutons des avantages et des limites de ces travaux et nous montrons en quoi ces solutions répondent ou non aux besoins que nous avons soulevés précédemment.

2.4.1 Travaux autour de la personnalisation, la mobilité et le VHE

L'objectif du projet européen VESPER (Virtual Home Environment for Service Persoalization and Roaming Users) [LYBP02a] est la définition, le développement et la validation du concept VHE. Il propose des outils et des fonctionnalités qui assurent la personnalisation des services, l'adaptation aux capacités du terminal et aux ressources réseau, la portabilité des services et la mobilité des sessions pour les utilisateurs nomades (dans le cas du roaming). Plusieurs standards et technologies puissants (OSA/Parlay et MExE) ont été utilisés dans la réalisation du système. Les agents mobiles sont également utilisés dans l'adaptation et la fourniture de services multimédia.
VESPER est l'un des projets les plus proches de notre travail. A notre connaissance, c'est le projet de référence qui réalise le concept VHE et propose un prototype le validant.
VESPER propose une solution dans laquelle, des composants spécifiques VHE sont installés sur le terminal et sur le réseau. Bien que ces composants offrent des fonctionnalités évoluées absentes dans d'autres architectures, l'application de VESPER est réduite à un nombre limité de terminaux et de réseaux pouvant supporter les agents mobiles et OSA/Parlay qui

sont utilisés dans VESPER.
Par rapport à notre travail, VESPER n'assure pas la découverte de services sensible au contexte. Les services supportés se limitent aux services multimédia. De plus, l'adaptabilité des services n'est pas basée sur la composabilité. Elle consiste à transcoder les flux multimédia pour répondre aux changements du contexte.

Plusieurs travaux de recherche ont proposé l'utilisation de la technologie agents mobiles pour la réalisation du concept VHE et ont défini des architectures de fourniture de services basées sur ce concept. Ces travaux se sont intéressés en particulier à la personnalisation, la mobilité personnelle, la mobilité du terminal et la nomadicité. Les autres besoins de fourniture de services tels que la découverte de services et l'adaptabilité dynamique n'ont pas été traités.

SOMA (Secure Open Mobile Agent) [BCS02, BCS00] est une plate-forme agents mobiles (AMs) basée sur Java pour la fourniture de services Internet de type VoD (*Video on Demand*) aux utilisateurs mobiles.
Les AMs sont utilisés pour assurer les aspects suivants de la fourniture de services : la personnalisation des services, le VHE, la nomadicité et l'adaptabilité.
SOMA utilise des profils basés sur XML pour décrire les préférences de l'utilisateur et les capacités du terminal afin d'assurer la personnalisation des services et leur adaptabilité aux préférences de l'utilisateur et aux capacités du terminal.
L'adaptation se fait au moment de l'accès au service en transcodant la VoD en un format supportable par le terminal et qui répond au mieux aux besoins de l'utilisateur.

Dans le cadre du CLIMATE [CLI], le cluster du programme ACTS [ACT] dédié aux agents mobiles et intelligents dans les environnements de télécommunications, plusieurs projets ont étudié l'utilisation des agents dans plusieurs aspects tels que : le contrôle de services dans les réseaux fixes et mobiles, la gestion des télécommunications, le commerce électronique, les applications multimédia, etc.

Le projet MONTAGE (1998-2000) [mon, JVP+99] a étudié l'utilisation des agents mobiles dans les architectures TINA-C [TIN] pour la gestion de la mobilité personnelle, la personnalisation des services, la fourniture de services dans le cas de nomadicité (*roaming*) et la facturation des services.
MONTAGE utilise les agents mobiles pour la fourniture de services dans un environnement qui implique des utilisateurs mobiles et plusieurs détaillants de services concurrents et coopératifs. Un utilisateur mobile peut avoir un abonnement avec plusieurs détaillants, appelés détaillants principaux (Home Retailers). Il peut également utiliser des services offerts par d'autres détaillants dans son domaine de nomadicité et qui ont des accords de fédération avec les détaillants principaux. Chaque détaillant propose des services offerts par un nombre de fournisseurs de services et supportés par des opérateurs réseau.
MONTAGE utilise les agents mobiles dans plusieurs aspects de la mobilité personnelle notamment dans la négociation et la sélection du meilleure détaillant capable d'offrir les services les mieux adaptés aux profils de l'utilisateur. Les agents mobiles sont également utilisés pour la gestion de la nomadicité et la facturation des services.
L'implémentation de MONTAGE est basée sur Java et utilise la plate-forme d'agents mo-

biles Voyager [VOY].

Le projet AMASE [KRS99] a proposé une plate-forme agents mobiles basée sur Java qui permet la fourniture de services multimédia aux utilisateurs des réseaux sans fil.
AMASE a traité essentiellement les besoins de fourniture de services suivants : la nomadicité, la déconnexion des terminaux dans les réseaux sans fil, la personnalisation des services, l'adaptation des services suite à la mobilité personnelle et la mobilité du terminal, la gestion de la qualité de service (QoS) et enfin la mise à l'échelle.
AMASE utilise la plate-forme SWARM de Siemens [SWA] écrite entièrement en Java.

Le projet CAMELEON [CAM] a été défini pour la mise en œuvre du concept VHE. Il combine les technologies agents mobiles et agents intelligents pour permettre la fourniture de services à travers des réseaux très hétérogènes en particulier, dans le cas de la nomadicité.
CAMELEON a développé un gestionnaire de profils adaptatifs appelé APM (Adaptive Profile Manager) [FGB00] basé sur les agents. Il permet aux utilisateurs mobiles d'accéder à des services personalisés indépendamment du terminal utilisé (téléphone mobile, ordinateur mobile, PDA,...). La présentation du service est adaptée aux capacités du terminal et au profil de l'utilisateur.
L'APM implémente les principes de base de VHE. Dans ce cadre, il fournit les fonctionnalités suivantes : la fourniture flexible de services, la nomadicité, la personnalisation des services, la mobilité personnelle et enfin l'adaptabilité des services aux capacités du terminal.

2.4.2 Travaux autour de la sensibilité au contexte, l'adaptabilité dynamique et la composabilité des services

Le projet RNRT CESURE [PPM+00] s'intéresse à la modélisation et à l'exploitation d'applications de services à usagers potentiellement mobiles. L'objectif du projet est de fournir, d'une part des outils permettant la conception des applications de services, et d'autre part l'infrastructure pour la configuration, le déploiement et la supervision du système informatique distribué réalisant ces services. Ces opérations sont pilotées depuis le terminal de l'usager à l'aide d'une carte à puce permettant de stocker les préférences et les besoins de l'utilisateur.
Une des principales caractéristiques de cette approche est de permettre l'installation "en continu" de l'application. Il n'existe pas de phase spécifique exécutée avant toute utilisation. Celle-ci a lieu lors de la connexion d'un usager. La plate-forme logicielle permet l'installation, sur le terminal utilisé, d'une version adaptée des modules logiciels nécessaires au bon fonctionnement de l'application. L'adaptation concerne aussi bien, les préférences de l'utilisateur que les caractéristiques du terminal utilisé. Elle est réalisée au moment de l'accès au service, grâce à l'utilisation d'une approche de conception à base de composants [RPPM00b].
Par rapport aux besoins définis dans les sections précédentes, CESURE n'assure pas la découverte de services. Le contexte est modélisé juste par les préférences de l'utilisateur

et les capacités du terminal. La seule mobilité prise en compte est la mobilité personnelle. Enfin, l'adaptabilité des services est réalisée de manière statique une seule fois au moment de l'accès au service en composant dynamiquement ce dernier.

Dans le cadre de la fourniture de services pour environnements mobiles, les deux projets suivants se sont intéressés à l'adaptabilité sensible au contexte des services multimédia.

Le projet OnTheMove [MG97] a proposé et implémenté le middleware MASE (Mobile Application Support Environment) [KPM+98] pour la fourniture d'applications multimédia sensibles à la mobilité dans des environnements sans fil hétérogènes. La différence entre ces réseaux se traduit par la différence de la QoS que chacun peut offrir.
MASE fournit un ensemble d'interfaces qui permettent de masquer la complexité et l'hétérogénéité des réseaux sans fil, contrôler la QoS, réagir à ses variations et fournir des informations sur la localisation du terminal.
MASE a traité principalement deux besoins liés à la mobilité dans les environnements mobiles sans fil : la sensibilité au contexte et l'adaptabilité des services à leur contexte d'exécution. Les paramètres qui définissent le contexte d'exécution dans MASE sont décrits dans des profils. Ils concernent les préférences de l'utilisateur, les capacités du terminal et enfin les ressources réseau.
MASE adapte le service selon la localisation du terminal et le contexte d'exécution comme décrits dans les profils. L'adaptation du service multimédia consiste à compresser, convertir, transcoder ou réduire les objets multimédia constituant le service sur la partie fixe du réseau avant de transmettre le service sur le terminal mobile.

Les auteurs dans [HAK01] proposent une architecture basée sur les agents mobiles pour la fourniture de services multimédia personnalisés dans les environnements mobiles. Le système utilise plusieurs agents mobiles coopératifs et distribués sur différents sites. Des politiques et des règles d'autorisation et d'accès sont définies sur chaque site pour contrôler l'utilisation des ressources locales.
Des agents représentant l'utilisateur et d'autres représentant le site visité s'engagent dans des processus de négotiation en utilisant les politiques d'autorisation afin de permettre aux utilisateurs mobiles d'accéder à leurs services personnalisés sur des sites différents de leur site home.
En plus de la personnalisation, le système utilise les agents mobiles pour assurer l'adaptabilité des services aux capacités du terminal en cours d'utilisation.

2.4.3 Conclusion sur les travaux de recherche existants

Pour conclure, la fourniture de services pour environnements mobiles est une problématique qui combine plusieurs travaux de recherche. Les solutions existantes ne traitent pas tous les besoins identifiés précédemment (voir section 2.2) et proposent des solutions uniquement partielles.
L'ensemble des projets sus-cités ne constitue pas une liste exhaustive de ce qui se fait dans chaque domaine de recherche impliqué dans la fourniture de services. Nous avons présenté uniquement ceux réalisés dans le cadre de la fourniture de services. Nous en citerons davan-

tage tout au long des chapitres suivants, à chaque étude détaillée d'un besoin particulier de la fourniture de services.

2.5 Conclusions et motivations de cette thèse

Dans ce chapitre, nous avons identifié l'ensemble des besoins relatifs à la fourniture de services dans les environnements mobiles. Ces besoins sont liés d'une part, à la mobilité et à l'hétérogénéité de ces environnements et d'autre part, à la personnalisation des services pour le compte de l'usager. Nous avons présenté les principaux travaux qui ont été proposés dans le contexte de la fourniture de services. Ces travaux ont bénéficié d'un effort important de la part des communautés scientifique et industrielle.

La fourniture de services pour les environnements mobiles regroupe de multiples acteurs et partenaires dont les compétences sont diverses et nécessite donc le savoir-faire de plusieurs domaines. De ce fait, elle pose plusieurs défis techniques. La gestion de la mobilité, la localisation, la personnalisation, la découverte de services, l'adaptabilité et la sensibilité au contexte sont autant de besoins nécessaires pour la création et la fourniture de services interactifs et adaptables.
Ces besoins nécessitent de nouvelles architectures capables de déployer, gérer, découvrir et adapter dynamiquement et de manière transparente des services multi-réseaux et multi-terminaux. Ces architectures doivent contrairement aux architectures existantes, supporter des propriétés non traditionnelles telles que la sensibilité au contexte et l'adaptabilité et, doivent également offrir un accès à la fois aux services traditionnels et aux nouveaux services.

La problématique de la fourniture de services dans les futurs réseaux mobiles a été relevée par différentes instances de normalisation. Plusieurs travaux de standardisation ont traité cette problématique selon différents points de vue : terminal, réseau de l'opérateur ou fournisseur de services.
Les solutions orientées terminal (MExE, WAP, iMode, etc.) s'intéressent à l'adaptabilité des services aux différents terminaux mobiles. Les solutions et les architectures orientées réseau (telles que CAMEL, SPIRITS et PINT) s'intéressent aux problèmes liés à la fourniture et au contrôle des services dans des réseaux hétérogènes. Enfin, les solutions orientées fournisseur de services (telles que OSA et les Web Services) fournissent des interfaces réseau permettant aux fournisseurs de services d'accéder aux fonctionnalités et aux capacités du réseau quelle que soit sa nature.
Le constat relevé actuellement est l'absence d'une vision globale de cette problématique. En effet, les standards et les travaux de normalisation existants traitent partiellement la fourniture de services (selon le point de vue terminal mobile, opérateur réseau ou fournisseur de services). Ainsi, il n'existe pas de solution complète qui répond aux besoins de tous les acteurs impliqués dans le processus de fourniture de services.

Contrairement aux travaux de normalisation, la plupart des propositions issues du domaine de la recherche, disposent d'une vision globale de la problématique de fourniture de services et intègrent les principaux acteurs à savoir : l'utilisateur mobile, le terminal mobile,

les fournisseurs de services et éventuellement l'opérateur réseau. Cependant, comme cette problématique combine plusieurs domaines de recherche (mobilité, hétérogéinité, personnalisation, gestion de la QoS, adaptabilité, découverte de services, sensibilité au contexte, etc.), ces solutions ne traitent qu'une partie des besoins que nous avons identifiés dans ce chapitre. Certaines solutions comme CAMELEON, MONTAGE et SOMA, s'intéressent à l'applicabilité des agents mobiles pour la personnalisation des services et la nomadicité des utilisateurs dans les environnements mobiles. Ces travaux ne sont pas d'actualité puisque la technologie agents mobiles est actuellement en perte d'allure. D'autres solutions, relativement récentes, telles que VESPER et CESURE, s'intéressent plus à la sensibilité au contexte et à l'adaptabilité. Néanmoins, l'adaptabilité assurée est souvent statique et n'est pas réalisée dynamiquement selon le changement dynamique de l'environnement. De plus, le contexte est souvent représenté soit par les ressources réseau, soit par les capacités du terminal.

C'est pourquoi nous allons proposer, dans les chapitres suivants, une plate-forme de fourniture de services pour environnements mobiles qui permet de répondre à nos besoins. La plate-forme proposée traite essentiellement les problèmes liés à la fourniture de services sensible au contexte. Ces problèmes concernent la gestion et le déploiement des services, le concept VHE, la personnalisation, la sensibilité au contexte, la description du contexte, l'échange d'informations contextuelles, la découverte de services sensible au contexte, l'adaptabilité et la composabilité des services ainsi que les problèmes architecturaux.
Le prototype de notre proposition a été développé dans le cadre du projet Européen IST MOBIVAS [MOB, MOB00].

Chapitre 3

Les agents mobiles et la fourniture de services dans les environnements mobiles

Ce chapitre est consacré aux paradigmes de communication dans les environnements mobiles. Nous nous intéressons en particulier au code mobile et à l'utilisation des agents mobiles (AMs) pour la fourniture de services. Nous montrons, grâce à une étude comparative avec Java RMI, que ce paradigme est particulièrement bien adapté à la fourniture de services pour environnements mobiles.

Dans un premier temps, nous présentons brièvement quelques définitions relatives au code mobile. Nous discutons ensuite du meilleur paradigme de communication pour une architecture de fourniture de services pour environnement mobile. Ce choix est argumenté par les résultats d'une étude comparative entre les agents mobiles et Java RMI. Pour finir, nous présentons un scénario basé sur les agents mobiles pour démontrer l'intérêt et les avantages de ce modèle pour la fourniture de services.

3.1 Contexte

Un ensemble de paradigmes ont été conçus pour faciliter la programmation des applications réparties. Nous nous plaçons dans le cadre de la conception d'une architecture de fourniture de services à valeur ajoutée pour des utilisateurs de terminaux mobiles. La plate-forme proposée devra effectuer le téléchargement de services à la demande de l'utilisateur et le téléchargement sur le terminal des logiciels nécessaires pour l'exécution de ces services.
Le choix du paradigme de communication pour la plate-forme de fourniture et de téléchargement de services est crucial : il doit prendre en compte d'une part, les fonctionnalités que doit offrir la plate-forme et, d'autre part, les contraintes et les limites imposées par le réseau et les terminaux mobiles qui peuvent être de faibles capacités (mémoire, capacité du processeur, etc.).

Un des besoins spécifiques que la plate-forme doit assurer est la mobilité du code. Ce besoin consiste, dans le cadre de la fourniture de services, à transférer les services proposés par les fournisseurs de services au terminal mobile où ils pourront s'exécuter. La plate-forme choisie doit donc disposer d'un modèle d'exécution répartie qui assure la mobilité du code.

3.2 Définition du code mobile

Le domaine de recherche associé à la mobilité du code est si vaste que le terme *code mobile* ne possède pas de définition claire et définitive.
Informellement et de manière générale, on peut définir la notion de code mobile comme un paradigme permettant de concevoir des traitements amenés à s'exécuter sur différentes structures d'accueil. Plus communément, le code mobile représente *la capacité à changer le lien entre le code et l'endroit où celui-ci est exécuté* [CPV97].
Cette définition assez vaste, rend l'appréciation du code mobile difficile puisqu'elle ne précise pas s'il s'agit du code source, du code binaire ou encore du code avec son contexte d'exécution.

On trouve le concept de code mobile sous diverses formes, aussi bien dans les approches d'agents mobiles et dans les langages du code mobile (MCL) [Tho97] que dans les applets Java. Néanmoins, la notion du code mobile est souvent confondue avec celle de l'agent mobile.
Un agent mobile est un processus autonome, possédant un environnement d'exécution propre incluant du code et des données, pouvant se déplacer de machine en machine afin de réaliser la tâche qu'on lui a déléguée et retourner ses résultats au client qui l'a mandaté. Les agents mobiles sont une variété particulière du code mobile. Ils ont une autonomie et un pouvoir de décision que tous les codes mobiles n'ont pas nécessairement. Un agent mobile choisit lui-même, selon un parcours prédéfini, sa destination et le moment où il va s'y rendre. Une applet Java par exemple, est un code mobile qui est envoyé par le serveur pour être exécuté sur la machine hôte. Toutefois et contrairement aux agents mobiles, elle n'a pas la capacité de décision lui permettant de changer d'hôte.

3.3 Classification des paradigmes du code mobile

Cette section fait une synthèse des différents paradigmes du code mobile. Ces paradigmes peuvent être classés en fonction des entités (client, serveur, programme ou fonction) impliquées dans la mobilité du code, du type du code à déplacer, et de l'initiateur de la mobilité de ce code [Rou02].

1. **Code référençable**

   Dans cette approche, une entité émet une requête à un site cible afin de récupérer, après exécution d'un code disponible sur le site cible, un résultat.
   L'appel de procédure distante RPC [BN83] et l'invocation d'objets à distance se situent dans cette classe [Ber00].

2. **Code téléchargeable**

   Les modèles de cette classe sont des extensions du code référençable. Ils définissent la possibilité de migrer le code vers des ressources plutôt que de le référencer. Cette approche consiste à transférer le code d'une entité à une autre, soit afin d'économiser l'échange de messages, soit pour déplacer le service vers les ressources.
   Le code mobile téléchargé peut être un code source ou un code binaire. L'initiateur de la mobilité est un acteur externe au code.
   Le modèle d'évaluation à distance [BGP97, Bel01] décrit exactement le principe de cette approche.

3. **Migration d'activité**

   Les modèles de cette classe définissent le mouvement d'un processus en cours d'exécution (avec son contexte d'exécution) vers une autre machine. L'initiateur de la mobilité est le site où se trouve le code.
   Cette approche est conçue pour répondre aux problèmes de répartition de charge, de tolérance aux pannes ou de partage d'information.
   Nous retrouvons dans cette classe le principe du code à la demande [BGP97].

4. **Agents mobiles**

   Ce modèle définit la possibilité de déléguer une tâche à un agent mobile. Le code mobile est un processus avec son contexte d'exécution. La différence avec la migration d'activité réside dans l'initiation de la mobilité. Dans ce cas, c'est l'agent qui de manière autonome, décide de migrer.
   Le modèle agent mobile est proche de celui de l'évaluation à distance dans le sens où le site initiateur détient le code des opérations à effectuer et l'envoie pour exécution sur un site distant. La différence réside dans l'asynchronisme des interactions entre les deux sites : une fois migré sur un site distant, un agent mobile devient une entité autonome, qui peut, suivant l'exécution du code, se déplacer sur un autre site, accumuler des résultats, communiquer avec d'autres agents, signaler un événement, etc., sans qu'une connexion permanente soit maintenue avec le site initial [JK98, FLP98, GKN+96]. Une présentation plus détaillée de ce modèle sera donnée dans les sections suivantes.

Examinons maintenant l'impact de ces classes de code mobile lorsque l'on se place dans le cadre de la fourniture de services pour les environnements mobiles.

Une architecture de fourniture de services dans un environnement mobile repose sur une infrastructure qui doit permettre aux utilisateurs mobiles de télécharger des services pour pouvoir les exécuter sur leurs terminaux mobiles.
Cette architecture doit en premier lieu prendre en compte les besoins de l'application de fourniture de services. Pour cela, elle doit assurer l'optimisation des ressources et la gestion des connexions.
Contrairement au paradigme client-serveur (qu'il soit mis en œuvre par une technique de code référençable, code téléchargeable ou migration d'activité), le paradigme agents mobiles est basé sur le mode asynchrone. Ceci rend les agents très adaptés aux environnements

nomades et mobiles où l'infrastructure réseau est caractérisée par un coût de communication élevé et des équipements qui se connectent au réseau par intermittence. En utilisant des agents mobiles, le terminal se connecte au réseau seulement pour envoyer l'agent dans le réseau et pour le récupérer, évitant ainsi l'envoi d'une multitude de messages sur le lien à faible bande passante et à coût élevé.
La fourniture de services pour les environnements mobiles constitue ainsi un des domaines les plus adaptés à l'utilisation des agents mobiles.

L'intérêt de l'utilisation des agents mobiles pour la fourniture de services dans un environnement mobile est justifiée par une évaluation de performance que nous présentons dans la section suivante. Cette évaluation compare le modèle agent mobile au modèle client-serveur représenté par Java RMI.

3.4 Comparaison entre Java RMI et les agents mobiles

3.4.1 Java RMI versus Agents Mobiles

3.4.1.1 Java RMI

Java RMI (Remote Method Invocation) [RMI99] est la solution proposée par Sun Microsystems [SUN] au problème de la gestion des objets distribués. Grâce à RMI, un programme Java pourra appeler certaines méthodes d'objets existant sur un serveur distant (objets distants).
Le but de RMI est de permettre l'appel, l'exécution et le renvoi du résultat d'une méthode exécutée sur une machine virtuelle différente de celle de l'objet l'appelant. Cette machine virtuelle peut être sur une machine différente pourvu qu'elle soit accessible par le réseau.

La machine sur laquelle s'exécute la méthode distante est appelée serveur. L'appel côté client d'une telle méthode est un peu plus compliqué que l'appel d'une méthode d'un objet local mais il reste simple. Il consiste à obtenir une référence sur l'objet distant puis à simplement appeler la méthode à partir de cette référence.
La technologie RMI se charge de rendre transparente la localisation de l'objet distant, son appel et le renvoi du résultat. En fait, elle utilise deux classes particulières, le *stub* et le *skeleton*, qui doivent être générées avec l'outil *rmic* fourni par Java.
Le stub est une classe qui se situe côté client et le skeleton est son homologue coté serveur. Ces deux classes se chargent d'assurer tous les mécanismes d'appel, de communication, d'execution, de renvoi et de réception du résultat.

Java RMI permet donc de fournir un mécanisme uniforme de désignation d'objets permettant d'invoquer une méthode sur un objet donné, quelque soit sa localisation dans le système réparti, en utilisant une référence globale unique.
Une des caractéristiques principales de Java RMI est sa capacité de télécharger le bytecode (le code généré après la compilation) d'une classe d'un objet même si cette classe n'est pas définie sur la machine virtuelle réceptrice. Le type et le comportement de l'objet téléchargé, qui existait auparavant dans une seule machine virtuelle, peut être transmis à une autre machine virtuelle éventuellement distante.

RMI est une solution "tout Java", contrairement à la norme CORBA [OMGb] qui permet de manipuler des objets à distance avec n'importe quel langage. CORBA est toutefois beaucoup plus compliqué à mettre en œuvre que Java RMI [FHB00].

3.4.1.2 Agents Mobiles

Le concept d'agent mobile a été proposé comme alternative au modèle client-serveur qui est la base de la communication dans les applications distribuées.
Toute application mise en œuvre à partir d'agents mobiles peut aussi l'être avec des interactions client-serveur. Les bénéfices potentiels des agents mobiles sont liés principalement aux critères suivants :

1. Performance

   La performance se traduit par la possibilité d'optimiser l'utilisation des ressources du réseau et de réduire la quantité du trafic le traversant. Par ailleurs, la performance dépend du type de réseaux : plus le débit du réseau est faible, plus le coût des communications est élevé, et plus l'avantage est significatif. Le concept d'agent mobile s'adapte parfaitement aux applications qui évoluent dans des réseaux qui ne sont pas très rapides et assez chers (réseaux sans fil par exemple).

2. Asynchronisme

   L'asynchronisme se traduit par la possibilité de déléguer une tâche à un agent mobile sans rester en attente du résultat. Le concept d'agent mobile est ainsi adapté aux machines qui se connectent au réseau occasionnellement ou par intermittence.

3. Autonomie

   Un agent mobile contient suffisament d'informations (parcours prédéfini à l'avance par le développeur) pour qu'il puisse prendre des décisions de façon autonome.

4. Le temps de réponse

   Les agents mobiles permettent de réduire le temps de réponse de certaines applications. Cette réduction de temps est essentiellement due au déplacement de l'exécution vers les données. En effet, les agents vont se déplacer pour accéder localement aux données dont ils ont besoin plutôt que déplacer ces données vers eux, notamment si ces données sont très importantes.

La mise en œuvre des agents mobiles suppose que, sur chaque site susceptible d'accueillir des agents mobiles, il existe un support système pour l'exécution, la communication, la migration et la sécurité des agents mobiles [Ber00]. De nombreux systèmes agents mobiles ont été développés ces dernières années. Des exemples sont les systèmes AgentTCL [Gra95], Telescript [Gen95], Aglets [AGL] et Grasshopper [GRA]. La plupart de ces systèmes sont basés sur Java car ce langage est particulièrement adapté au développement de plates-formes d'agents mobiles. Il offre de nombreux avantages dont le principal est la portabilité, la mobilité du bytecode et l'indépendance vis-à-vis du système d'exploitation.

3.4.2 Étude comparative

Dans cette étude, nous cherchons à comparer le modèle client-serveur en géneral, et les invocations distantes en particulier, aux agents mobiles en terme de performance. Nous nous plaçons dans un cadre simplifié en ne considérant comme paramètre de performance que le volume de données à faire transiter sur un réseau.
Dans une organisation client-serveur classique, le choix du placement des éléments (client, services, etc.) sur les sites du réseau et leurs rôles influent particulièrement sur la performance.
Les agents mobiles peuvent réduire la consommation en bande passante en déportant les calculs vers les données (*i.e.* déplacer la logique des interactions vers la source) plutôt que de rapatrier ces données pour les traiter localement à la manière d'une invocation distante (par exemple : RPC ou RMI). Les agents peuvent alors interagir en local avec un service. Si le service impose des interactions multiples, le trafic réseau et la latence s'en retrouvent réduits.

Dans cette section, nous proposons une étude comparative entre Java RMI et les agents mobiles dont l'objectif est de mettre en évidence les avantages des agents pour la fourniture de services dans les environnements mobiles. Après une description du domaine d'application que nous adoptons, nous présentons le modèle analytique, les résultats des expériences ainsi que la conclusion sur l'évaluation de chacun des deux modèles.

3.4.2.1 Scénario d'application

Dans ce scénario, nous supposons que le client peut accéder à un service fourni par plusieurs opérateurs. Chaque opérateur propose une version différente de ce même service et chaque serveur proposant ce service possède des données spécifiques le décrivant.

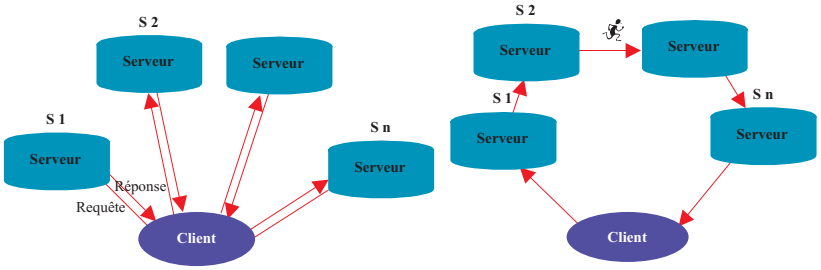


FIG. 3.1 – Modèle Java RMI et modèle AMs

Pour pouvoir récupérer un service, le client doit d'abord récupérer les données qui décrivent chaque version du service désiré. Le client traite ensuite ces données pour sélectionner la version du service la mieux adaptée à ses besoins (en terme de QoS, prix,

contenu, etc.). Une fois la version choisie, l'utilisateur peut télécharger le service sur son terminal.

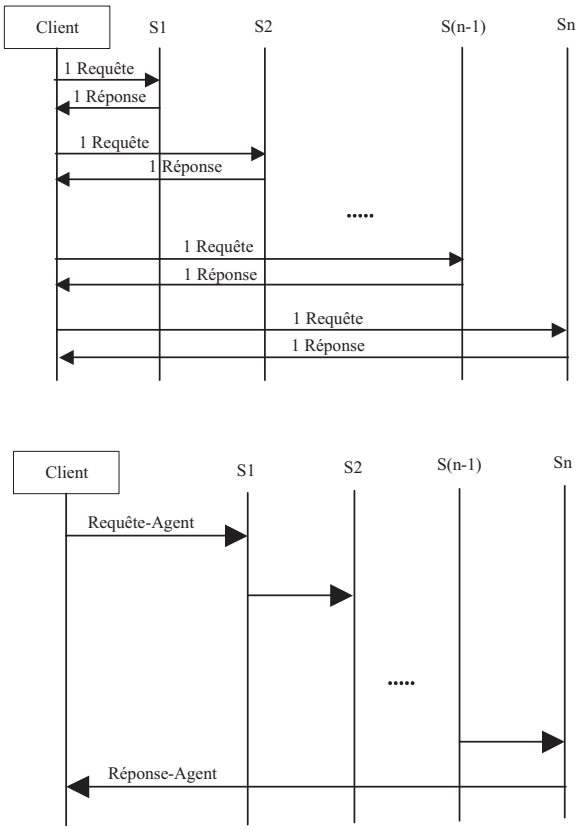


FIG. 3.2 – Diagrammes de communication dans les modèles Java RMI et AMs

Dans le cas du modèle Java RMI, la communication entre le client et le serveur se traduit par des appels de procédure à distance. Chaque message impliquant le client est soit une requête de recherche du service, soit la réponse à la requête incluant les données qui décrivent la version du service désiré. Le travail dans ce cas est orchestré par la machine du client et une interaction entre le client et chaque serveur nécessite une communication continue (une requête suivie de sa réponse) comme le montrent les figures 3.1 et 3.2). Ainsi, pour $\boldsymbol{n}$ serveurs à contacter, le client doit envoyer et recevoir $\boldsymbol{2n}$ messages.

Le modèle agent mobile nécessite uniquement l'envoi d'un message qui transporte l'agent mobile du client vers le premier serveur à visiter, ensuite entre les autres serveurs (figure 3.1). Le travail dans ce cas est orchestré par l'agent mobile et une interaction entre le client et chaque serveur ne nécessite pas une communication continue (pas de message

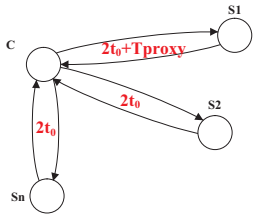
envoyé du serveur vers le client suite à une requête sauf pour le dernier serveur, figure 3.2). Ainsi, si n est le nombre de serveurs à contacter, $n+1$ messages seront échangés entre toutes les stations.

Notre modèle analytique est basé sur deux fonctions élémentaires : l'une évalue Java RMI et les AMs en terme de coût de communication, l'autre en terme de temps de téléchargement. Le but des expériences sur le coût de communication est de déterminer le temps moyen nécessaire pour qu'un client établisse une communication avec un nombre de serveurs pour effectuer une tâche spécifique sur chaque serveur. Dans ces expériences, le client ne récupère aucune donnée des serveurs. Le but des expériences concernant le temps de téléchargement est de déterminer le temps nécessaire au client pour télécharger un ensemble de données.

3.4.2.2 Le modèle Java RMI

En utilisant le modèle Java RMI, chaque serveur fournit une interface distante qui permet à l'utilisateur, en utilisant le nom du service, d'obtenir les données le décrivant.
RMI supporte deux classes qui implémentent une invocation de méthode distante. La première classe, qui est le comportement de la méthode distante, s'exécute sur le serveur. La deuxième, joue le rôle d'un mandataire ou intermédiaire (*proxy*) et s'exécute sur le terminal.
Pour pouvoir récupérer les données décrivant le service à télécharger, le client doit faire un appel distant à chaque serveur. Pendant cet appel, le client doit d'abord télécharger le proxy qu'il utilise ensuite pour invoquer la méthode distante.

1. Coût de communication
   Dans le modèle RMI, le temps pour obtenir la réponse finale d'un serveur est la somme du temps de l'établissement de la communication, du temps de téléchargement du proxy, du temps d'invocation de la méthode distante et du temps de réponse du serveur.
   Comme la même interface de la procédure distante est utilisée sur tous les serveurs, le proxy est téléchargé sur le client uniquement lors du premier appel distant (à partir du premier serveur).

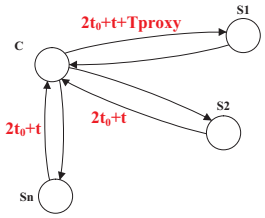


   Nous considérons le coût de communication le même pour tous les serveurs contactés par le client. Nous notons T_{proxy}, la durée du téléchargement du proxy.

Le temps de communication entre le client et le serveur, sans considérer la durée du téléchargement du proxy, est égal à $2t_0$, où t_0 représente le temps d'établissement de la communication entre le client et le serveur (ou entre le serveur et le client).
Pour n serveurs, le coût total de communication est donné par :

$$t^n_{ComRMI} = 2nt_0 + T_{proxy} \tag{3.1}$$

2. Temps de téléchargement
   Le client doit télécharger la même quantité de données décrivant le service depuis



chaque serveur. Le temps pour obtenir la réponse finale d'un serveur est égal au coût de communication, plus le temps t de téléchargement des données qui dépend évidemment de la taille des données à télécharger. Nous supposons que le temps de téléchargement des données est identique quelle que soit la localisation du serveur. Ainsi, la durée totale du téléchargement des données peut être exprimée par la formule suivante :

$$t^n_{RMI} = n(2t_0 + t) + T_{proxy} \tag{3.2}$$

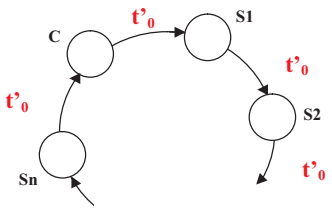
3.4.2.3 Le modèle agent mobile

En utilisant le modèle agent mobile, le client envoie un agent qui négocie avec chaque serveur pour récupérer le service le mieux adapté à ses préférences. Nous supposons ici que le temps de négociation entre l'agent et chaque serveur est négligeable.

1. Coût de communication
   Dans ce cas, on évalue le temps qu'il faut à un agent pour se déplacer d'un serveur à un autre sans récupérer les données. Ce temps est donné par :

$$t^n_{ComAM} = (n+1)t_0^{'} \tag{3.3}$$

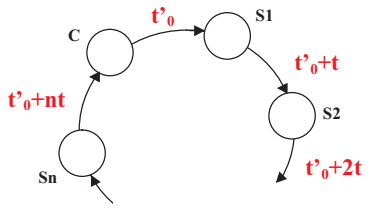
où, $t_0^{'}$ représente le temps de migration de l'agent. Pour simplifier, on admet que l'agent prend le même temps pour se déplacer entre deux machines (du client vers le serveur, puis d'un serveur à un autre).



2. Temps de téléchargement
   Commençons d'abord par le cas général. On suppose que l'agent migre d'un serveur à un autre et récupère la même quantité de données de chaque serveur.
   Si t est le temps de téléchargement de cette quantité de données, alors le temps total de téléchargement pour n serveurs est le suivant :

$$t_{AM}^{n} = t_0^{'} + (t_0^{'} + t) + \ldots + (t_0^{'} + nt) \tag{3.4}$$

$$t_{AM}^{n} = (n+1)\left(\frac{nt}{2} + t_0^{'}\right) \tag{3.5}$$



Nous nous plaçons dans un scénario où l'agent doit récupérer les données du meilleur service répondant à un critère particulier (prix, contenu, QoS offerte, etc.). Pour se faire, l'agent doit se déplacer d'un site à un autre à la recherche du service qui répond le mieux à ce critère. Une fois il le trouve, il récupère les données décrivant le service en question et il retourne à son site initial.
Dans ce scénario, l'agent mobile doit donc télécharger une seule fois les données qui se trouvent sur le dernier serveur. Le temps total est alors donné par la formule suivante :

$$t_{AM}^{n} = (n+1)t_0^{'} + t \tag{3.6}$$

3.4.2.4 Comparaison des deux modèles

Le modèle agent mobile est plus intéressant que le modèle RMI, si le temps de téléchargement des données dans RMI est plus grand que celui des agents mobiles. Selon les équations 3.2 et 3.6, cette condition peut s'exprimer par la relation suivante :

$$n(2t_0 + t) + T_{proxy} \geq (n+1)t_0^{'} + t \tag{3.7}$$

ou encore

$$n \geq \frac{t + (t_0^{'} - T_{proxy})}{t + (2t_0 - t_0^{'})} \tag{3.8}$$

Comme le temps de téléchargement (t) peut être exprimé en fonction de la taille des données ($DataSize$) et du débit du réseau (Th)[1], l'équation 3.8 devient :

$$n \geq \frac{DataSize + (t_0^{'} - T_{proxy}).Th}{DataSize + (2t_0 - t_0^{'}).Th} \tag{3.9}$$

De la même manière et à partir de l'équation 3.7, on peut obtenir les relations suivantes qui expriment le temps de téléchargement en fonction du nombre de serveurs :

$$t \geq \frac{(t_0^{'} - 2t_0)n + (t_0^{'} - T_{proxy})}{n-1} \tag{3.10}$$

$$DataSize \geq \left(\frac{(t_0^{'} - 2t_0)n + (t_0^{'} - T_{proxy})}{n-1} \right).Th \tag{3.11}$$

Il faut noter que dans ces équations, nous avons considéré que les paramètres $t_0^{'}$, t_0, T_{proxy} et Th sont constants. En effet, Th représente le débit du réseau et il est fixe. Les autres paramètres sont obtenus en faisant des tests de mesures du coût de communications sur deux machines pour les deux modèles RMI et agents mobiles.

Une première analyse de l'équation 3.11 montre que la taille des données converge vers une limite de petite valeur lorsque le nombre de serveurs est grand. Ainsi, pour une taille de données proche de cette limite, le modèle agent mobile est plus performant que le modèle RMI lorsque le nombre de serveurs est grand.

[1] $t = DataSize/Th$

3.4.2.5 Analyse du modèle analytique

Les expériences présentées dans cette section ont été réalisées sur des stations Sun Ultra-5 5.8 sous le système Solaris 3.6.1 en utilisant l'environnement Java (JDK1.2) et la plate-forme d'agents mobiles Grasshopper [GRA, BBCM99]. Nous utilisons un lien Ethernet 100 Mbits/s. Les mesures présentées sur chaque courbe ont été effectuées sur la station du client en répétant le même test 10 fois afin d'obtenir des valeurs moyennes..

Le modèle analytique présenté précédemment (équations 3.9 et 3.11) a été instantié dans cet environnement expérimental pour déterminer les valeurs des constantes des équations ($t_0^{'}$, t_0, T_{proxy} et Th).

En variant la taille des données dans l'équation 3.9, on obtient le nombre de serveurs pour lequel le modèle agent mobile est plus performant que le modèle RMI (voir la figure 3.3).

En variant le nombre de serveurs dans l'équation 3.11, on obtient la taille des données pour laquelle le modèle agent mobile est plus performant que le modèle RMI (voir la figure 3.4).

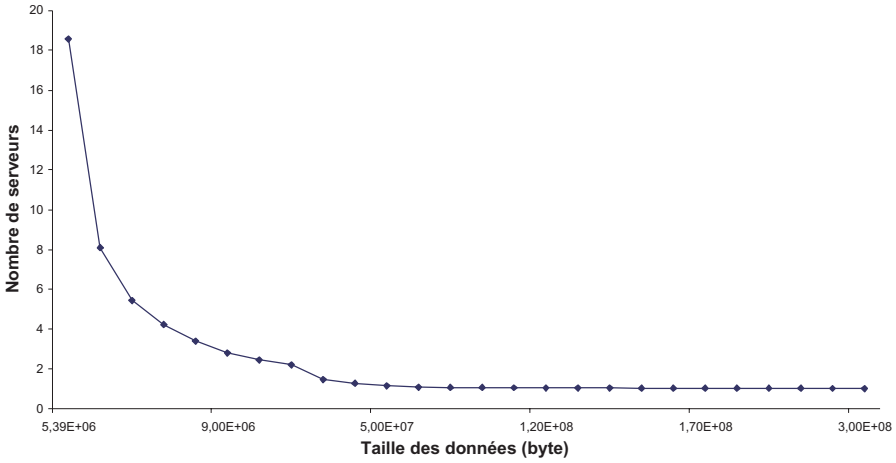


FIG. 3.3 – Nombre de serveurs en fonction de la taille des données pour que les AMs soient plus performants (selon le modèle analytique)

Les résultats présentés dans les figures 3.3 et 3.4 montrent que :

1. Pour une taille de données très réduite, l'utilisation des agents mobiles n'est interessante que s'ils doivent visiter un nombre important de serveurs. Sinon, Java RMI donne de meilleurs résultats.
2. Inversement, pour un nombre réduit de serveurs, les agents mobiles présentent de meilleurs résultats lorsque la taille des données à télécharger est très importante.

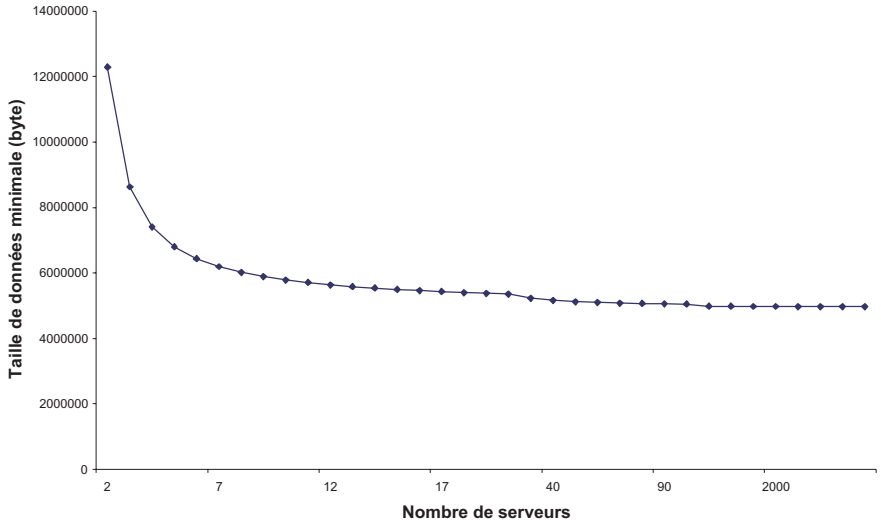


FIG. 3.4 – Taille des données en fonction du nombre de serveurs pour que les AMs soient plus performants (selon le modèle analytique)

3. Enfin, nous pouvons conclure d'après le modèle analytique, que les agents mobiles sont plus performants lorsque soit la taille des données est importante soit le nombre de serveurs est grand.

3.4.2.6 Résultats expérimentaux et analyses

Dans le reste de cette section, nous présentons les résultats expérimentaux dont l'objectif est de comparer les modèles agent mobile et RMI pour le téléchargement de services et la validation du modèle analytique décrit dans la section précédente.

Les figures 3.5, 3.6, 3.7 et 3.8 représentent respectivement la variation du temps de téléchargement en fonction de la taille des données à télécharger en considérant respectivement 1, 4, 6 et 9 serveurs.
Pour un seul serveur (voir figure 3.5), les résultats montrent que le modèle RMI est plus performant que les AMs pour les résultats théoriques et expérimentaux. La raison réside dans le fait que dans le modèle RMI, en plus des données décrivant le service, l'utilisateur doit télécharger le proxy de la méthode distante alors que dans le cas des AMs, le code de l'agent est transporté deux fois entre le client et les serveurs. Comme dans notre scénario, la taille du code de l'agent est largement plus grande que la taille du proxy, le modèle RMI est plus performant que les agents mobiles.
Selon notre modèle analytique et d'après les résultats expérimentaux, lorsque le nombre de serveurs dépasse les 4 serveurs, le modèle agent mobile devient plus intéressant que le modèle RMI si la taille des données à télécharger dépasse une valeur donnée (à peu près

0,12 Mbytes dans le cas de 6 serveurs). Ceci confirme les résultats obtenus dans le modèle analytique qui indiquent que les agents mobiles sont plus efficaces lorsque, soit la taille des données est importante, soit le nombre de serveurs est grand.
Cependant, on observe dans toutes les courbes, que les mesures expérimentales ne correspondent pas exactement au modèle théorique lorsque la taille des données est très grande. Ceci est dû à d'autres facteurs qui non pas été pris en compte dans notre modèle théorique. Ces facteurs sont essentiellement liés à la manière qu'utilise le système d'exploitation (Unix dans notre cas) pour gérer les fichiers et le transfert des données entre la mémoire et le disque local dans chaque machine visitée (la gestion du cache) [Kra88].

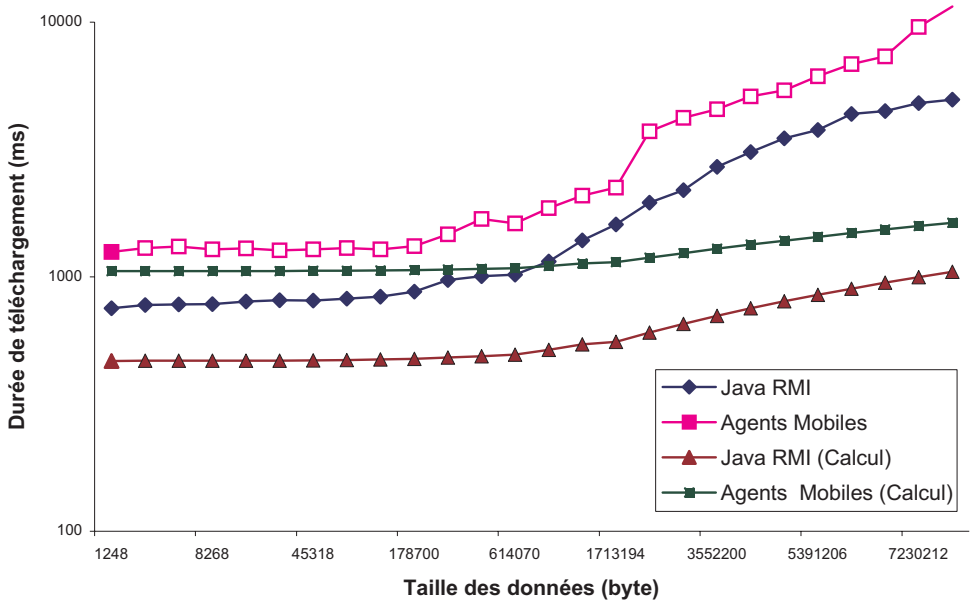


FIG. 3.5 – Comparaison entre Java RMI et AMs (1 Serveur)

Comme conclusion, nous pouvons dire que les agents mobiles sont beaucoup plus performants que Java RMI (et en général les modèles client-serveur) pour le téléchargement du code si le nombre de serveurs à visiter est grand ou lorsque les données à télécharger sont de taille très importante.

3.4.3 Travaux similaires

Peu de travaux ont proposé une évaluation quantitative de la technologie AMs par des mesures réelles.

Les auteurs dans [CPV97] expliquent comment choisir le paradigme de communication

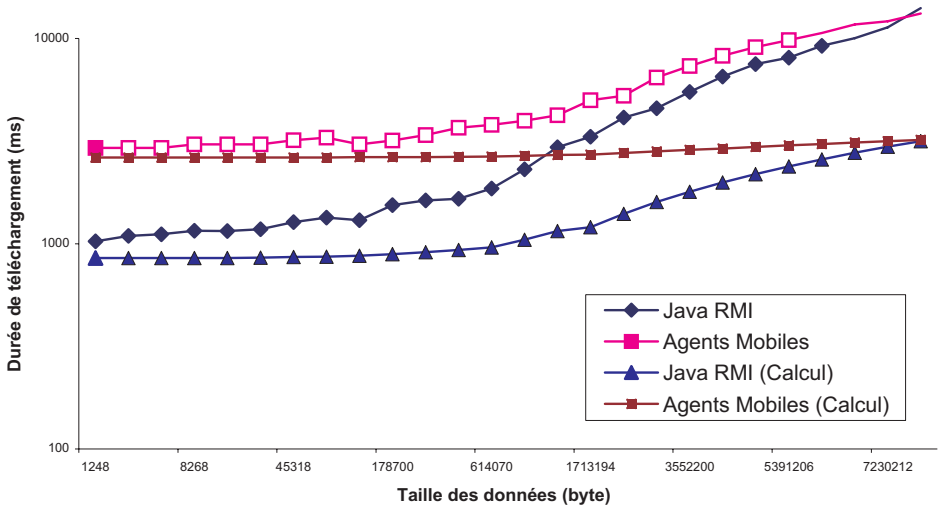


FIG. 3.6 – Comparaison entre Java RMI et AMs (4 Serveurs)

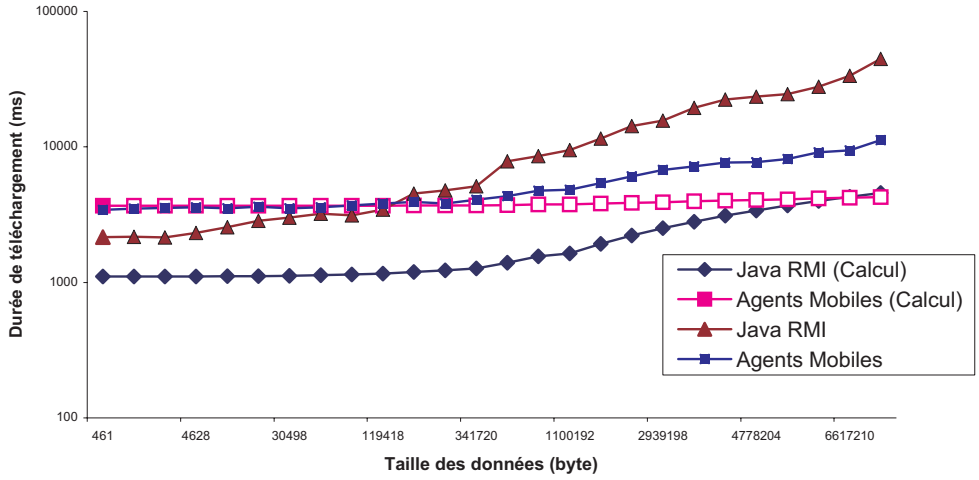


FIG. 3.7 – Comparaison entre Java RMI et AMs (6 Serveurs)

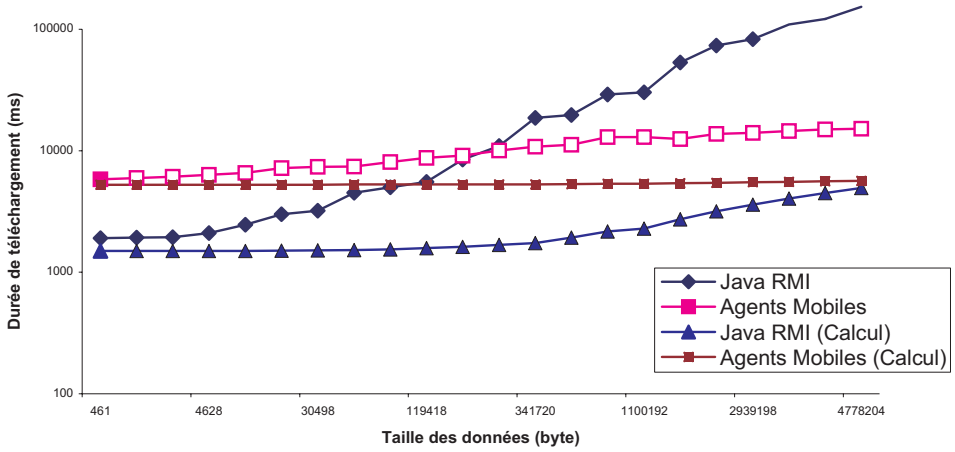


FIG. 3.8 – Comparaison entre Java RMI et AMs (9 Serveurs)

le plus approprié pour une application répartie donnée. Ils considèrent différents paradigmes de communication, notamment le client-serveur, l'évaluation à distance, le code à la demande et les agents mobiles. Ils étudient un modèle de performance simple basé sur une application de recherche documentaire. Les auteurs concluent que le choix d'un modèle de communication approprié pour la conception d'une application distribuée ne peut pas se faire dans l'absolu. En effet, ce choix dépend de la capacité du modèle à répondre aux besoins de l'application.

Dans [JAU00], les auteurs proposent un modèle analytique pour évaluer les performances des agents mobiles en terme de temps de réponse dans le domaines de la recherche d'informations pour les utilisateurs mobiles. Les résultats montrent que les agents mobiles ne sont pas toujours plus performants que le client-serveur et vice versa, en particulier si pour une requête donnée, la taille de l'agent mobile est plus grande que la requête du client-serveur. Les résultats montrent également que les agents mobiles présentent une meilleure performance lorsque la qualité du lien sans fil baisse.

Dans [IH99, HI00], les auteurs proposent une évaluation de performances du modèle agent mobile et du modèle client-serveur dans des conditions réelles sur Internet. Cette évaluation est effectuée dans un environnement Java en utilisant respectivement RMI, la plate-forme agents mobiles Aglets [AGL] et une plate-forme agents mobiles que les auteurs ont développée. Les mesures portent à la fois sur les coûts des mécanismes élémentaires de Java utilisés dans la conception d'un système agent mobile, et sur une évaluation comparative des deux modèles (client-serveur et agents mobiles) à travers deux scénarios d'application : la redirection de requêtes et la compression de données. Dans la première application, un agent mobile permet de remplacer des appels à distance par des appels de

méthodes locaux. Dans l'application de compression de données, un agent mobile permet de compresser les données à récupérer sur les serveurs avant de se déplacer.
Les résultats de l'évaluation montrent que les agents mobiles peuvent amener des gains significatifs lorsqu'ils permettent de tranformer les communications à distance en communications locales et de réduire le volume d'informations transféré sur le réseau. Ces gains sont d'autant plus importants que le réseau est lent, car on amortit alors plus rapidement le coût du déplacement de l'agent.

En conclusion, tous les travaux présentés dans cette section font une étude comparative entre les agents mobiles et le modèle client-serveur. Cependant, les performances mesurées dans chaque travail dépendent énormément du type de l'application pour laquelle l'étude a été faite. Une décision au sujet du choix du support approprié pour le téléchargement et la fourniture de services basée sur ces travaux n'est pas suffisante en soi. Ceci nous a amenés à réaliser notre propre étude comparative entre les agents mobiles et Java RMI.

3.5 Agents mobiles pour la fourniture de services dans les environnements mobiles

Le concept agent mobile a déja été utilisé comme élément de base dans les architectures des réseaux de la nouvelle génération. Dans l'état de l'art du chapitre 2, nous avons présenté plusieurs projets de recherche qui ont adopté ce concept pour la réalisation de certains aspects de la fourniture de services pour les environnements mobiles.

Dans le cadre du projet IST MOBIVAS, nous avons participé à la définition, la conception, et le développement d'une architecture de téléchargement de services à valeur ajoutée. Les résultats encourageants que nous avons obtenus de l'étude comparative entre Java RMI et les AMs nous ont poussés à proposer l'utilisation de la technologie agent mobile dans plusieurs aspects de la fourniture de services. Les aspects que nous avons traités sont la nomadicité, le téléchargement du code dans un environnement mobile, l'adaptabilité des services aux profils de l'utilisateur et la gestion de la mobilité du terminal et de la mobilité personnelle. Dans ce qui suit, nous présentons une infrastructure sur laquelle sera déployé un scénario de fourniture de services basé sur les agents mobiles.

3.5.1 Architecture générique pour la fourniture de services

Comme le montre la figure 3.9, l'architecture prévue pour supporter le scénario que nous proposons comprend trois entités de base : le terminal mobile (TM), l'opérateur réseau (OR) et les fournisseurs de services à valeur ajoutée (FSs). Le TM est le dispositif qui permet à l'utilisateur de communiquer avec le réseau et de télécharger les services. Les FSs offrent leurs services en connectant l'infrastructure de leur réseau à celle de l'opérateur réseau. Un nouveau composant pour la gestion de services appelé le Médiateur de Services (MS) est défini au niveau de l'opérateur réseau. C'est le composant le plus important dans cette architecture. Il gère la coordination des fonctions nécessaires pour une fourniture continue et efficace des services aux utilisateurs.

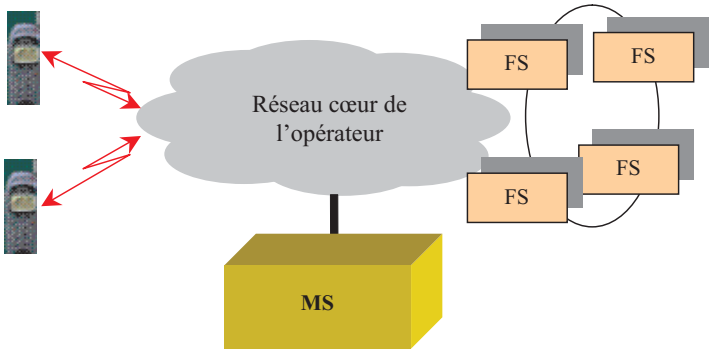


FIG. 3.9 – Architecture du réseau de fourniture de services

3.5.2 Scénario de fourniture de services basé sur les agents mobiles

Dans cette partie, nous présentons un scénario de fourniture de services basé sur les agents mobiles. Ce scénario permettra aux utilisateurs de terminaux mobiles d'accéder à des services selon leurs profils personnels même s'ils ne sont pas disponibles à partir de leur opérateur nominal (avec lequel l'utilisateur a une souscription pour un ou plusieurs services). Dans ce scénario, l'utilisation des agents mobiles à la place des mécanismes de communication client-serveur standards, permettra de réduire la charge du réseau et introduire la flexibilité pour la fourniture de services.

Un aspect très important dans la fourniture de services est d'offrir à l'utilisateur la possibilité de localiser le service désiré entre des milliers de services offerts par l'opérateur et les fournisseurs de services qui lui sont associés. Dans le cas de la nomadicité, ce problème devient plus complexe. Le souscripteur peut vouloir accéder à des services qui ne sont pas disponibles à partir de son opérateur (par exemple, des informations spécifiques au pays ou à la ville où il se trouve). De plus, il serait très souhaitable pour lui de trouver le service le plus adapté à ses besoins (en terme de QoS, prix, contenu, etc.) parmi ceux fournis, non seulement par un seul opérateur, mais par un groupe d'opérateurs (tous les opérateurs de son domaine de nomadicité). Dans ce cas, la communication entre l'opérateur nominal et l'opérateur visité est souvent coûteuse. Les agents mobiles peuvent jouer un rôle important pour minimiser la communication de bout en bout de l'opérateur visité à l'opérateur nominal. Certaines tâches spécifiques comme la gestion des profils de l'utilisateur chez l'opérateur visité, la recherche et le téléchargement de service lors de la nomadicité, la recherche des services les plus adéquats, peuvent être développées en utilisant les agents mobiles [FHB00, FBD01, FBD02a]. Le scénario suivant montre l'intérêt de l'utilisation des agents mobiles pour la fourniture et le téléchargement de services.

Afin d'offrir les fonctionnalités nécessaires à la fourniture de services, un médiateur de services (MS), est défini dans chaque opérateur réseau. Il contrôle le processus de fourniture de services entre le terminal mobile (TM) et les fournisseurs de service (FSs). L'utilisateur

peut aussi établir une session d'accès avec n'importe quel opérateur qui a une fédération avec son opérateur nominal.

Nous définissons quatre entités pour gérer la fourniture de services lors de la nomadicité :

- Médiateur de Service Principal (MSP) : un gestionnaire de services spécifique avec lequel l'utilisateur a une souscription pour un ou plusieurs services. Le MSP possède des informations sur le profil de l'utilisateur. Il peut déterminer les capacités du terminal et les préférences de l'utilisateur qui ont un impact important sur la façon dont les informations et les services sont présentés à l'utilisateur.
- Médiateur de Service par Défaut (MSD) : le premier point de contact du terminal avec la plate-forme dans un domaine visité. Le MSD peut aussi fournir des services à l'utilisateur.
- Médiateur de Service Candidat (MSC) : n'importe quel médiateur de services différent du Médiateur de Service Principal et qui peut fournir des services à l'utilisateur. Ceci implique que le MSC peut coopérer avec le Médiateur de Service Principal.
- Médiateur de Service Sélectionné (MSS) : le médiateur de services qui fournit à l'utilisateur un service suite à un processus de sélection de service.

Le processus de fourniture de services dans le cas de nomadicité nécessite les différentes phases suivantes :

- Phase d'établissement d'accès durant laquelle l'utilisateur demande un accès aux services et est authentifié. Dans cette phase, le médiateur de services visité récupère le profil de l'utilisateur du Médiateur de Service Principal.
- Phase de négociation de capacités durant laquelle le Médiateur de Service par Défaut sélectionne les services offerts à l'utilisateur en se basant sur les capacités du terminal et les préférences de l'utilisateur. La négociation des capacités est le mécanisme qui permet au terminal mobile de communiquer ses capacités (logicielles et matérielles) au médiateur de services.
- Phase de sélection de service durant laquelle l'utilisateur négocie, avec chaque médiateur de services pouvant lui offrir des services. La négociation est basée sur les préférences de l'utilisateur.
- Phase de téléchargement de service durant laquelle le téléchargement de service est établi entre l'utilisateur et le médiateur de services sélectionné. Les caractéristiques du service sélectionné sont récupérées pour être utilisées dans les phases suivantes.
- Phase d'utilisation du service durant laquelle l'utilisateur exécute le service téléchargé sur le terminal.
- Phase de terminaison d'accès durant laquelle la session d'accès est terminée. Les informations concernant les préférences de l'utilisateur collectées pendant cette session sont envoyées au Médiateur de Service Principal. Ce dernier les enregistre dans le profil de l'utilisateur pour les prochaines sessions de fourniture de services.

Les phases définies précédemment peuvent être implémentées en utilisant des agents mobiles. Des agents mobiles spécifiques peuvent migrer d'un médiateur de services à un autre pour accomplir chacune de ces phases.

Durant la phase d'établissement d'accès, l'utilisateur établit un accès avec un Médiateur

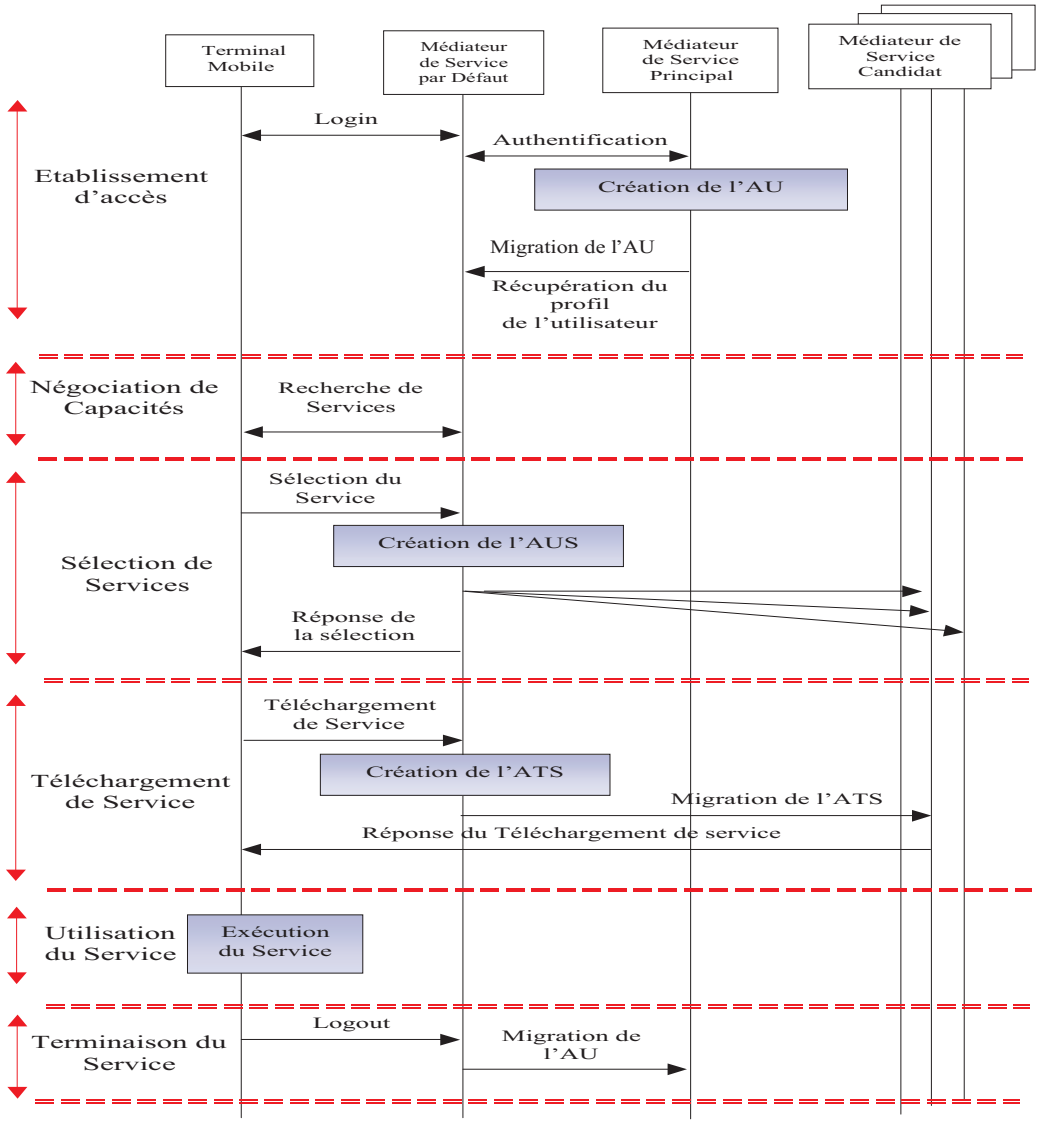


FIG. 3.10 – La fourniture de services basée sur les AMs dans le cas de nomadicité

de Service par Défaut qui contacte le Médiateur de Service Principal de l'utilisateur pour authentifier ce dernier. Après l'authentification, le Médiateur de Service Principal crée un agent mobile appelé Agent Utilisateur (AU) qui se déplace vers le Médiateur de Service par Défaut pour jouer le rôle de l'utilisateur dans le domaine local. À la fin de phase de terminaison d'accès, l'agent mobile retourne au Médiateur de Service Principal, où il met à jour le profil de l'utilisateur selon le choix et les préférences de l'utilisateur faits pendant les phases de sélection et d'utilisation du service.

Une fois que la phase d'établissement d'accès est terminée, la phase de négociation de capacités commence et l'utilisateur reçoit une liste de services qui peuvent être exécutés sur le terminal de l'utilisateur. Il choisit ensuite un service et une phase de sélection de services débute dont l'objectif est de choisir le médiateur de services qui fournit la meilleure offre pour le service sélectionné par l'utilisateur.

Durant la phase de sélection, l'utilisateur est représenté par un agent mobile appelé Agent Utilisateur Service (AUS) qui est créé par le AU dans le Médiateur de Service par Défaut. L'AUS récupére de l'AU toutes les informations concernant le profil de l'utilisateur. Son principal rôle est de négocier avec les différents médiateurs de services pour le compte de l'utilisateur. Afin d'accomplir sa tâche, l'AUS dans le Médiateur de Service par Défaut, envoie des clones (une copie de l'AUS) à tous les Médiateurs de Service Candidats qui offrent le service désiré par l'utilisateur. Chaque clone négocie localement, avec un ensemble de Médiateurs de Services Candidats, les meilleures offres, en utilisant les informations contenues dans le profil de l'utilisateur. Une fois que les négociations sont terminées, l'AUS dans le Médiateur de Service par Défaut récupère les résultats fournis par ses clones et prend la décision sur le meilleur service offert selon les préférences de l'utilisateur.

A la fin de la phase de sélection et durant la phase de téléchargement de service, l'AU crée l'Agent de Téléchargement de Service (ATS) dans le Médiateur de Service par Défaut. L'ATS est l'agent mobile qui migre vers le médiateur de services sélectionné pour récupérer le service en faveur de l'utilisateur.

Après la livraison du service à l'utilisateur et une fois que la phase d'utilisation est finie, l'ATS transmet à l'AU les informations sur les préférences de l'utilisateur obtenues pendant les phases de téléchargement et d'utilisation du service. Ensuite, il se termine.

Le scénario décrit ci-dessus est attrayant pour les raisons suivantes :

- Les agents mobiles sont utilisés pour assurer la fourniture de services dans les environnements mobiles. La plupart des phases requises pour la fourniture de services sont réalisées de façon asynchrone. Il n'y a pas besoin de maintenir une connexion sur un lien non fiable et coûteux pour toute la durée de chacune des phases (en particulier entre le TM et les autres entités de la plate-forme). Ceci aurait pu être réalisé par l'envoi de messages asynchrones. Cependant, dans ce cas, un degré raisonnable d'interactivité aurait nécessité un grand nombre de messages induisant un délai significatif pour l'accomplissement de la tâche ainsi qu'une augmentation du trafic sur le réseau (comme cela a été démontré dans la section 3.4). Dans le scénario à base d'agents mobiles, l'agent contient la logique nécessaire pour l'interaction avec les différentes entités de la plate-forme.

- Une interaction entre le Médiateur de Service Principal et le Médiateur de Service par Défaut durant les sessions d'établissement d'accès et de terminaison d'accès n'est pas nécessaire puisque l'agent mobile transporte avec lui toutes les informations nécessaires.
- Dans la phase de sélection de service, le scénario correspond exactement au scénario que nous avons développé dans la section 3.4 pour comparer les agents mobiles au Java RMI et dans lequel nous avons démontré que les agents mobiles sont plus performants que Java RMI lorsque le nombre de serveurs (correspond aux Médiateurs de Services Candidat dans ce scénario) ou la taille des données à télécharger sont importants.
- Durant la phase du téléchargement de service, le Médiateur de Service par Défaut délègue la tâche de téléchargement à un agent mobile sans rester en attente des réponses du fournisseur de services. Ce mécanisme permet de réduire la charge au niveau du Médiateur de Service par Défaut ainsi que le temps de réponse dans le cas où un grand nombre de requêtes de téléchargement sont envoyées simultanément au Médiateur de Service par Défaut. D'autres part, l'utilisation des agents mobiles pour le téléchargement des services peut être plus efficace encore si l'agent peut compresser les services sur le site du fournisseur de services avant de transporter le code au terminal [IH99, HI00]. Une opération de décompression sur le terminal permettra de reconstituer le service sur le terminal une fois l'agent arrive sur ce dernier.

3.5.3 Les agents mobiles pour une plate-forme de fourniture de services

Dans la section précédente, nous avons proposé un scénario de fourniture de services basé sur les AMs. Ce scénario montre que les agents mobiles peuvent être utilisés dans toutes les phases de fourniture de services et ils sont très adaptés aux environnements mobiles.
Toutefois, l'utilisation des agents mobiles comme plate-forme de fourniture de services peut soulever les problèmes suivants :

1. *Contraintes de développement :* Dans une solution agents mobiles, les interactions entre les différents composants de l'architecture de fourniture de services seront réalisées en utilisant une plate-forme agent mobile installée dans chaque composant. Ainsi, les fournisseurs de services et les opérateurs réseaux doivent non seulement accepter le déploiement des agents mobiles dans leurs domaines, mais ils doivent aussi développer leurs services dans le langage de programmation des agents mobiles (Java par exemple).
2. *Interopérabilité :* Les plates-formes agents mobiles disponibles actuellement ne sont pas interopérables entre elles. Des travaux de standardisation concernant les systèmes d'agents mobiles ont été proposés aussi bien par l'OMG (Object Management Group) [OMGa] que par la FIPA (Foundation for Intelligent Physical Agent) [FIP]. La FIPA centre son activité sur la standardisation des caractéristiques de base des agents telles que la communication, l'interaction, etc. L'OMG propose une base pour

l'interopérabilié des différentes plates-formes d'agents mobiles existantes en ajoutant à CORBA les faciliés MASIF (Mobile Agent System Interoperability Facility) [MAS]. L'interopérabilité dans MASIF est supportée principalement par les conventions de nommage, la représentation de la localisation et la définition d'un module IDL pour les types communs, ainsi que les structures et les opérations d'un système d'agents. Néanmoins FIPA et MASIF sont loin d'être complets et la plupart des plates-formes d'agents mobiles exisantes sont non conformes à ces standards [KG99]. De ce fait, l'interopérabilité entre les différentes plates-formes est non disponible [Bou99]. L'utilisation donc des agents mobiles dans chaque composant d'une plate-forme de fourniture de services peut compromettre son ouverture.

3. *Besoins de ressources :* Les avantages que les agents mobiles fournissent exigent des besoins supplémentaires en terme de mémoire et de puissance de traitement qui peuvent ne pas être disponibles sur un terminal mobile. Dans le cas de la fourniture de services dans un environnement mobile, la plate-forme agents mobiles peut être installée uniquement dans le réseau et éventuellement dans des classes de terminaux bien spécifiques qui possèdent des ressources suffisantes pour héberger des agents mobiles.

Pour des raisons liées à ces inconvénients et malgré son intérêt, la solution agents mobiles n'a pas été adoptée dans le cadre du projet MOBIVAS et donc le scénario présenté dans la section 3.5.2 n'a pas été évalué. Cette solution a été remplacée par une solution HTTP pour le téléchargement de services, comme nous le verrons dans le chapitre suivant.

3.6 Conclusion

Le travail présenté dans ce chapitre s'inscrit dans le cadre de la conception d'une plate-forme de fourniture de services pour un environnement mobile.
Un des besoins principaux que la plate-forme doit assurer est le téléchargement du service qui permet de transférer les services du fournisseur de services au terminal de l'utilisateur. La plate-forme choisie doit donc disposer d'un modèle d'exécution répartie qui assure la mobilité du code tout en considérant les contraintes et les limites imposées par les environnements mobiles.

Nous avons pensé d'emblée à l'utilisation des agents mobiles puisque ces derniers possèdent des caractéristiques les rendant très adaptés aux environnements mobiles.

Afin d'évaluer l'intérêt de ce modèle pour la fourniture de services dans les environnements mobiles, nous avons réalisé une étude de performance entre ce modèle et le modèle Java RMI. Les résultats montrent que les agents mobiles peuvent amener des gains significatifs en réduisant le volume d'interactions sur le réseau. Ces gains sont d'autant plus importants que le nombre de serveurs à visiter et la taille des données à récupérer sont importants.

Après avoir démontré quantitativement l'apport des agents mobiles pour le téléchargement du code et afin d'illustrer l'intêret de leur utilisation pour la fourniture de services, un scénario de services basé sur les AMs a été proposé en détails. Dans ce scénario, l'utilité

des agents mobiles se trouve efficace dans les différentes phases du processus de fourniture de services.

L'étude comparative présentée dans ce chapitre, a été effectuée au début de notre travail, au moment où l'approche agents mobiles était très utilisée pour les raisons suivantes : l'asynchronisme, l'autonomie et la performance des agents mobiles. Malgré son intérêt, la solution agents mobiles n'a pas été adoptée dans le projet MOBIVAS pour assurer la communication entre les différents acteurs de la fourniture de services. Ceci est dû à des raisons liées essentiellement à la non-interopérabilité et la lourdeur des plates-formes agents mobiles actuellement disponibles.

Aujourd'hui, on regarderait probablement des solutions type *Intergiciels Orientés Message MOM* (Messages Oriented Middleware) [JPK99] qui connaissent à l'heure actuelle, un nouveau essor. Ces infrastructures logicielles sont adaptées à des applications naturellement asynchrones, basées sur le modèle requête/réponse, comme c'est le cas de l'application de fourniture de services pour les environnements mobiles. En effet, les applications mobiles forment un domaine où les MOMs peuvent être déployés (JMS est déjà une implémentation utilisable [SUN99]). Les applications mobiles, par leurs spécificités, imposent de nombreuses contraintes sur le modèle de communication à utiliser que les MOMs semblent remplir. Il faudra cependant, attendre la généralisation des réseaux de communications 3G pour confirmer ces hypothèses.

Dans le chapitre suivant, nous présentons notre plate-forme de fourniture de services et nous verrons la solution qui a été retenue afin d'assurer la communication entre les différents composants de la plate-forme.

Chapitre 4

CASP : plate-forme de fourniture de services sensible au contexte

Dans cette thèse, nous proposons une plate-forme de fourniture de services sensible au contexte appelée CASP (*Context Aware Service Provision*) qui a été développée dans le cadre du projet Européen IST MOBIVAS [MOB]. Cette plate-forme apporte des solutions aux besoins soulevés dans le chapitre 1, c'est à dire la sensibilité au contexte, la personnalisation, la mobilité, la découverte et l'adaptabilité dynamique des services. L'approche que nous proposons permet aux utilisateurs mobiles, d'une part, de découvrir des services en prenant en considération leur contexte de découverte, et d'autre part, d'exécuter des services adaptés à leur environnement d'exécution.

L'innovation dans la plate-forme CASP repose essentiellement sur l'utilisation de la sensibilité au contexte. Un contexte est représenté par l'ensemble des éléments suivants : les préférences de l'utilisateur, les capacités du terminal, la localisation de l'utilisateur et les ressources réseau.
Dans CASP, le contexte est utilisé dans deux phases de la fourniture de services : dans la phase de découverte de services entre le terminal mobile (TM) et un Médiateur de Services (MS) et dans la phase d'exécution du service découvert entre le terminal mobile et le Fournisseur de Services (FS). La découverte de services sensibles au contexte permet à l'utilisateur mobile de ne découvrir que les services qui correspondent à ses préférences, aux capacités de son terminal et à sa localisation. Durant la phase d'exécution, le service est adapté dynamiquement aux changements du contexte.

Ce chapitre s'organise comme suit. Le paragraphe 4.1 rapelle les besoins auxquels répond la plate-forme de fourniture de services. Le paragraphe 4.2 définit quelques concepts relatifs à la sensibilité au contexte. Le paragraphe 4.3 présente brièvement la plate-forme CASP. Les paragraphes 4.4 et 4.5 s'intéressent à la gestion du contexte et à l'utilisation des profils dans CASP.

4.1 Introduction

La sensibilité au contexte est de plus en plus nécessaire dans les environnements mobiles [Mit02]. Ces derniers doivent assurer aux utilisateurs mobiles la possibilité d'accéder à tout type de service à partir de n'importe quel terminal, n'importe où et n'importe quand. De plus, les utilisateurs doivent pouvoir utiliser les mêmes services personnalisés dans toutes les situations possibles. Le contexte peut donc être utilisé pour déterminer les services qui répondent le mieux à la situation courante et qui peuvent être proposés à l'utilisateur.

L'un des objectifs de cette thèse est d'élaborer une plate-forme de fourniture de services pour environnements mobiles. Les principaux besoins qu'une telle plate-forme doit satisfaire sont : la personnalisation des services, la mobilité de l'utilisateur et du terminal, la découverte de services et l'adaptabilité des services (pour plus de détails, voir la partie 2.2, page 13).

Afin de satisfaire tous ces besoins, le système doit prendre en compte l'environnement où se situe le terminal mobile. En d'autres termes, il doit être *sensible au contexte*.
La sensibilité au contexte est donc la propriété centrale de tous ces besoins et l'une des composantes les plus innovantes de la plate-forme CASP.

Dans ce qui suit, nous définissons les concepts de base de la sensibilité au contexte. Nous présentons ensuite la plate-forme CASP.

4.2 Gestion de la sensibilité du contexte

La possibilité d'utiliser des informations sur le contexte est cruciale pour le développement d'applications mobiles adaptatives. Dans le cadre de la fourniture de services, les applications sensibles au contexte sont capables de fournir aux utilisateurs mobiles, des services sur mesure.

4.2.1 Sensibilité au contexte

L'informatique sensible au contexte est un paradigme dans lequel des applications peuvent découvrir et utiliser des informations contextuelles. Un système est sensible au contexte s'il emploie des informations sur l'environnement pour fournir des services appropriés à l'utilisateur.
Le contexte inclut toute information pouvant caractériser la situation d'une entité. Une entité peut être une personne, un endroit, ou un objet et qui est pertinente pour l'interaction entre l'utilisateur et le service. L'utilisateur et le service sont eux-mêmes des entités (voir partie 2.2.5, page 15).

4.2.2 Gestion du contexte

Afin qu'un système soit sensible au contexte, il est nécessaire de le munir d'outils dédiés à la perception du contexte. Cette perception, doit de plus, être accompagnée d'une structuration appropriée des informations captées afin qu'elles puissent être exploitées. La

détection et l'analyse des situations sont deux tâches inconscientes que nous (les humains) effectuons fréquemment tout au long d'une journée. Par exemple, en rentrant dans une pièce, nous pouvons sans effort découvrir combien de personnes s'y trouvent, leur identité si elle nous est connue, ou encore la fonction de la pièce (bureau, salle de réunion, etc.). Cependant, lorsque ces tâches doivent être effectuées par un ordinateur, cela devient plus délicat. En effet, la complexité du monde réel est telle qu'il est difficile pour un ordinateur d'identifier les sources d'informations pertinentes. Dans les systèmes informatiques sensibles au contexte, l'acquisition des informations contextuelles se fait au travers de **capteurs**.

La quasi-totalité des systèmes sensibles au contexte reposent sur un modèle de base dont les éléments centraux sont les capteurs d'informations contextuelles et les éléments périphériques sont les applications utilisant les informations délivrées par les capteurs ou délivrant des informations aux capteurs (voir figure 4.1).
Les capteurs représentent, dans une infrastructure sensible au contexte, les composants qui font le lien entre le monde physique (localisation, proximité d'autres composants, etc.) et le monde virtuel. En effet, ils possèdent (ou reçoivent) des informations contextuelles du monde physique (ou virtuel) qu'ils transmettent au monde virtuel.

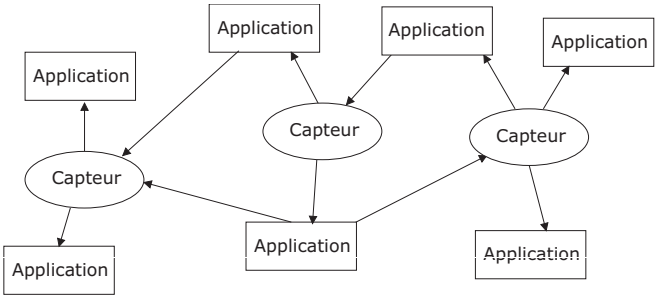


FIG. 4.1 – Consommation des informations contextuelles produites par les capteurs

On distingue deux catégories de capteurs :

1. **Les capteurs de localisation**

   La localisation est une information contextuelle importante qui varie avec les déplacements de l'utilisateur. Elle peut être récupérée automatiquement grâce à des capteurs type GPS (Global Positionning System [GPS]) ou Radio-Fréquence, ou à défaut transmise au système par l'utilisateur.

2. **Les capteurs d'informations contextuelles autres que la localisation**

   Dans le cadre de la fourniture de services, il existe des informations contextuelles autres que la localisation. Parmi ces informations, nous pouvons citer :
   - les ressources réseau disponibles : ce type d'informations permet d'adapter le comportement des applications selon la disponibilité de ces ressources (bande passante par exemple).

- les capacités du terminal qui décrivent les caractéristiques logicielles et matérielles du terminal.
- les "intérêts" de l'utilisateur qui décrivent les préférences de l'utilisateur.
- le temps qui est sans doute l'information contextuelle la plus facile à obtenir car la plupart des systèmes infomatiques sont dotés d'une horloge.

4.2.3 Représentation des informations contextuelles

Dans le cadre de la fourniture de services sensible au contexte, plusieurs éléments sont impliqués dans la récupération, la sauvegarde et l'utilisation du contexte. A cause de la nature distribuée d'une telle application, la standardisation du format, du contenu et d'échange des informations contextuelles est nécessaire.

Plusieurs standards, en cours de développement, sont concernés par la description des préférences de l'utilisateur, la personnalisation d'accès au contenu et l'échange des capacités du terminal et du réseau [LKP02]. Le standard le plus intéressant est le protocole CC/PP (W3C Composite Capabilities/Preference Profiles) [NHO00].
Basé sur le format XML et le format meta-données RDF (Resource Description Framework) [RDF], CC/PP permet de décrire les préférences de l'utilisateur et les caractéristiques logicielles et matérielles relatives au terminal.
Le *User Agent Profile* (UAProf) de WAP [UAP99] est une mise en œuvre de CC/PP destinée aux terminaux WAP pour décrire leurs capacités et les préférences de leurs utilisateurs.

4.3 Architecture de fourniture de services sensible au contexte

Dans le projet IST-MOBIVAS auquel nous avons participé, l'objectif était de permettre à un ensemble de prestataires de services de proposer des services à valeur ajoutée à des utilisateurs mobiles ([FBD02a, FBD02c, FBD02b, FHB00]). L'utilisateur se voit proposer un ensemble de services exécutables sur le terminal qu'il utilise, respectant ses préférences et prenant en compte sa localisation.
Dans le cadre de ce projet, notre travail a consisté à proposer et évaluer une solution architecturale permettant aux utilisateurs une découverte de services personalisée.

Nos propositions dans le cadre du projet MOBIVAS concernant la découverte personnalisée des services ont été intégrées dans la plate-forme CASP. En plus de la découverte des services, CASP assure l'adaptabilité des services, en cours d'exécution, aux changements du contexte de manière à garantir leurs fonctionnalités dans des conditions particulières ou nouvelles. Les différentes adaptations s'opèrent de manière dynamique.

Dans cette section, nous présentons la plate-forme CASP. L'architecture réseau prévue pour supporter cette plate-forme comprend trois entités de base : le terminal mobile (TM), l'opérateur réseau (OR) et les fournisseurs de services à valeur ajoutée (FSs).
Le terminal mobile est le dispositif qui permet à l'utilisateur de communiquer avec le réseau, de découvrir les services et de les télécharger. Les fournisseurs de services offrent leur services en connectant l'infrastructure de leur réseau à celle de l'opérateur réseau.

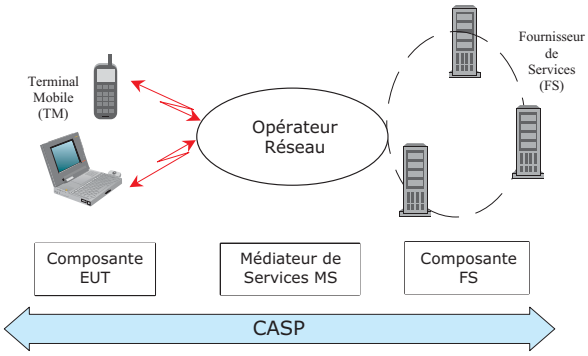


FIG. 4.2 – Architecture réseau supportant la plate-forme CASP

4.3.1 Composants de la plate-forme CASP

Comme le montrent les figures 4.2 et 4.3, la plate-forme CASP est constituée des trois composants suivants :

1. La *Composante FS* (installée chez les FSs), gère l'ensemble des services à proposer aux utilisateurs mobiles par le FS.
2. Le *Médiateur de Services (MS)* fait partie du réseau de l'opérateur. Il intervient dans le processus de découverte et de recherche de services. C'est le composant le plus important de la plate-forme CASP. Il gère la coordination des fonctions nécessaires pour une découverte et une fourniture de services personnalisés et sensibles au contexte. Deux bases de données sont associées au MS : la *BD_ Utilisateurs* et la *BD_ Services*. La première base de données contient des informations sur l'ensemble des abonnés de la plate-forme CASP attachée au MS. La deuxième contient les informations qui décrivent l'ensemble des services proposés par la plate-forme CASP.
3. Le dernier élément est localisé sur le terminal mobile. Il s'agit de *la Composante EUT* (*End User Terminal*). Elle représente l'environnement d'exécution des services CASP. Chaque EUT gère un ensemble de profils qui décrivent les capacités du terminal et les préférences de l'utilisateur.

Dans le chapitre 7, nous décrivons les interfaces définies entre les trois composants de la plate-forme CASP et nous présentons en détail la conception de chacun de ces composants ainsi que les différents modules les composant.

4.3.2 Gestion de l'utilisateur

Le MS possède des informations sur chaque abonné de la plate-forme CASP. Ces informations sont stockées dans une base de données appelée *BD_ Utilisateurs*. Les paramètres d'une entrée de la BD_Utilisateurs sont les suivants :

1. *Informations sur l'identification de l'utilisateur*
   Il s'agit de l'ensemble des informations qui identifient un utilisateur telles que : son

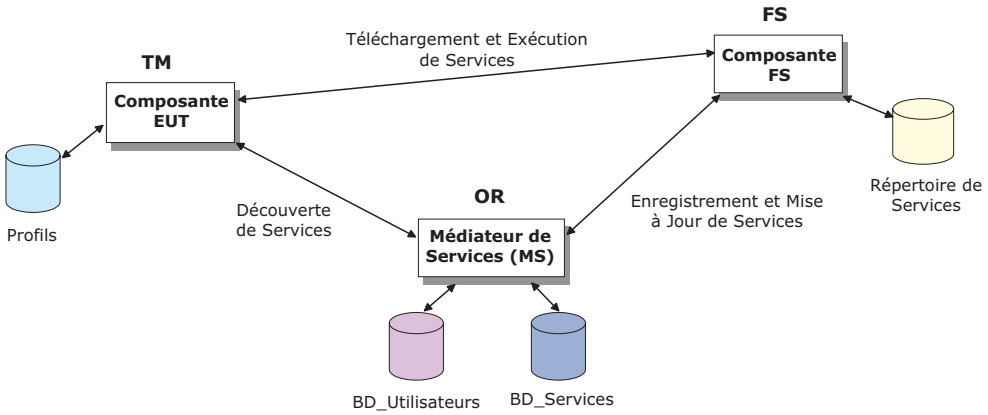


FIG. 4.3 – Les composants de la plate-forme CASP

nom, son adresse et son compte bancaire. Ces informations sont conservées par le MS, seule entité à pouvoir y accéder et modifier le contenu.

2. *Informations sur l'abonnement de l'utilisateur*
   L'utilisateur peut s'abonner à CASP selon trois types d'abonnement qui diffèrent par la qualité et le prix des services à proposer : *Gold*, *Standard* ou *Basic*. Si l'utilisateur choisit, par exemple, un abonnement de type *Basic*, le MS ne lui propose que les versions les moins chères de chaque service.
   Les informations sur l'abonnement des utilisateurs ne peuvent être modifiées que par le MS.

3. *Liste des services favoris*
   Il s'agit de l'ensemble des services favoris de l'utilisateur. Chaque élément de cette liste pointe une entrée dans la BD_Services.
   Les modifications de cette liste sont effectuées par le MS mais à la demande de l'utilisateur.

4. *Liste des catégories de services préférées*
   Cette liste contient la sélection de l'utilisateur des catégories de services favories disponibles dans la plate-forme.
   Cette liste est maintenue par l'utilisateur qui seul, peut y accéder et modifier le contenu.

5. *Préférences de configuration de services*
   Ces préférences ne sont pas liées à des services particuliers mais permettent au MS, suite à une requête de découverte de services envoyée par l'utilisateur, de sélectionner la version de chaque service la mieux adaptée aux besoins de l'utilisateur.

6. *Profils de l'interface graphique de l'utilisateur (copie de sauvegarde)*
   Normalement, les paramètres de configuration des interfaces graphiques de l'utilisateur sont stockés sur le terminal. Néanmoins, pour éviter à l'utilisateur de reconfigurer

son interface graphique dans le cas de perte ou d'endommagement de son terminal, une copie de ces paramètres est sauvegardée sur le MS.
L'utilisateur est la seule entité qui peut définir et modifier ces paramètres.

7. *Information de sécurité*
   Pour des besoins d'authentification des usagers et d'autorisation d'accès aux services, la plate-forme possède des clés de certification pour chaque utilisateur.

Le tableau 4.1 liste en détail ces paramètres. La figure 4.4 présente une entrée de la BD_Utilisateurs dans le format XML.

	Paramètres	**Description**	**Format**
Identificateur de l'abonné	UserNo	Numéro unique	Integer
	LoginID	Adresse mél	String
Type d'abonnement	TariffClass	Basic, standard ou gold	String
Info sur l'utilisateur	UserID		Integer
	LastName	Nom de l'utilisateur	String
	FirstName	Prénom de l'utilisateur	String
	Adress	Adresse de l'utilisateur	String
	City	Ville de l'utilisateur	String
	Zip	Code postal	Integer
	Country	Code du pays	Integer
	Phone	Numéro de téléphone	String
	Fax	Numéro de fax	String
Paramètres de Sécurité	SIM	Carte SIM	String
	SPKI	Certificat	String
	Passwd	Pour l'utilisation de CASP	String
Services Favoris	FavouriteServices	Liste des services favoris	String
	Preferred-ServiceCategory	Liste des Catégories de services	String
Profil de Configuration des Services	Quality	Basic, standard ou gold	String
	Culture	Langue : UK, FR, GER, ...	String
	Location	Localisation préférée	String
Profils des Interfaces Graphiques	TerminalProfiles	Liste des profils pour chaque classe de terminaux	String String
	Terminal	Classe du terminal	String

TAB. 4.1 – Les paramètres d'une entrée de la BD_Utilisateurs

4.4 Gestion du contexte dans CASP

Par contexte, nous faisons référence dans CASP, de manière minimale, mais néanmoins illustrative, à l'utilisateur et à son environnement physique.
Le contexte dans CASP est représenté par les différentes informations contextuelles suivantes :

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE User SYSTEM "c:\usermanagement\documents\UserDescription.dtd">
<User>
        <SubscriberIdentification>
                <UserNo>28</UserNo>
                <LoginID>praveen@yahoo.fr</LoginID>
        </SubscriberIdentification>
        <SubscriptionData>
                <TariffClass>Standard</TariffClass>
        </SubscriptionData>
        <UserInfo>
                <UserID>UserID</UserID>
                <LastName>Julia</LastName>
                <FirstName>Praveen</FirstName>
                <Street>44 rue des Cinq Diamants</Street>
                <Zip>75013</Zip>
                <City>Paris</City>
                <Country>Country</Country>
                <Phone>0033145817575</Phone>
                <Fax>0033145811213</Fax>
        </UserInfo>
        <SecurityParameters>
                <SIM>1234123412341234</SIM>
                <SPKI>SPKI</SPKI>
                <Passwd>pkb123</Passwd>
        </SecurityParameters>
        <FavouriteServices>Weather</FavouriteServices>
        <FavouriteServices>Stock</FavouriteServices>
        <FavouriteServices>XXX</FavouriteServices>
        <PreferredServicesCategory>Basic</PreferredServicesCategory>
        <PreferredServices>News</PreferredServices>
        <PreferredServices>Sports</PreferredServices>
        <ServiceConfigurationProfile>
                <Quality>Standard</Quality>
                <Culture>FR</Culture>
                <Location>France</Location>
        </ServiceConfigurationProfile>
        <TerminalConfigurationProfiles>
                <ProfileType>Class1</ProfileType>
                <ProfileConfiguration>MobileConfiguration</ProfileConfiguration>
        </TerminalConfigurationProfiles>
        <TerminalConfigurationProfiles>
                <ProfileType>Class2</ProfileType>
                <ProfileConfiguration>LaptopConfiguration</ProfileConfiguration>
        </TerminalConfigurationProfiles>
</User>
```

FIG. 4.4 – Exemple XML d'une entrée de la BD_Utilisateurs

1. *Préférences de l'utilisateur* : elles sont décrites dans un **Profil Utilisateur** qui sera présenté dans la section suivante.
2. *Capacités du terminal* : elles sont décrites dans un **Profil Terminal** qui sera présenté dans la section suivante.
3. *Localisation du terminal* : en théorie, CASP utilise le protocole GPS pour récupérer, à tout moment, la position du terminal (latitude, longitude). En pratique, l'interface qui s'occupe du transfert de la position du terminal à la plate-forme CASP, n'a pas été developpée. De ce fait, nous émulons cette interface en proposant à l'utilisateur d'introduire lui-même, les coordonnées de sa localisation à chaque nouvelle requête de découverte de services.
4. *Ressources réseau disponibles* : ces paramètres qui seront détaillés dans le chapitre 9, sont utilisés durant la phase d'exécution du service sur le terminal.

Dans CASP, le contexte est utilisé dans deux phases différentes de la fourniture de services :

1. Durant la phase de découverte de services afin de ne proposer à l'utilisateur que les services qui répondent à ses préférences, qui peuvent s'exécuter sur son terminal et qui utilisent éventuellement la localisation du terminal (s'il s'agit de services basés sur la localisation).
2. Durant l'exécution du service sur le terminal en adaptant dynamiquement le service aux changements qui interviennent dans le contexte afin de proposer la meilleure qualité possible du service à l'utilisateur.

4.5 Gestion des profils dans CASP

Afin de permettre la sensibilité au contexte, un modèle doit être défini pour l'acquisition, l'analyse, la sauvegarde et l'échange d'informations contextuelles. Le modèle utilisé dans CASP est basé sur les profils.

4.5.1 Intérêt des profils

Le *profil* est une notion centrale au concept de VHE qui préconise que les services et les préférences de l'utilisateur doivent toujours être présentés de la même manière quels que soient le réseau, la localisation de l'utilisateur et le terminal utilisé.

Le profil contient l'ensemble des informations qui peuvent être utilisées afin de personnaliser les services pour le compte d'un utilisateur. Il définit, pour un utilisateur particulier, l'environnement de fourniture et d'exécution des services en termes de ses préférences générales de communication, ses préférences concernant l'interface graphique et tout autre paramètre important pour cet utilisateur.

4.5.2 Les profils dans CASP

Dans la plate-forme CASP, quatre profils ont été définis : le Profil Service (*Service Profil*), le Profil Terminal (*Terminal Profil*), le Profil Utilisateur (*User Profil*) et le Profil Sécurité (*Security Profil*).

4.5.2.1 Profil Utilisateur

Selon les spécifications de MExE, chaque Profil Utilisateur est constitué de deux types d'informations :

1. **Profil de l'Interface Graphique de l'Utilisateur** : Ce sous-profil décrit la configuration de l'utilisateur de la partie CASP située sur le terminal. Il décrit également, comment l'utilisateur préfère agir avec les autres composants du système.
   Il est utilisé pour configurer l'interface graphique de l'utilisateur GUI (*Graphical User Interface*) selon les informations suivantes : la configuration des menus et des icônes, la taille et le type de la police, la configuration des couleurs, la langue à utiliser, etc.
2. **Profil de Configuration des Services**
   Ce sous-profil est utilisé pour personnaliser les services à télécharger sur le terminal mobile selon les préférences de l'utilisateur.
   Il contient par exemple des informations de pré-configuration liées aux services telles que : la langue préférée des services ou la qualité des services (*basic, standard ou gold*).

4.5.2.2 Profil Terminal

Ce profil décrit les capacités de chaque terminal. Les différents blocs qui constituent ce profil sont définis selon les spécifications de MExE et de CC/PP. Plusieurs autres attributs ont été rajoutés pour répondre aux besoins de CASP. Un exemple détaillé de ce profil est présenté dans l'annexe A.

4.5.2.3 Profil Service

Chaque service que la plate-forme CASP propose, est représenté par un *Profil Service* fourni par le FS, géré par le MS et stocké dans la BD_Services. Le profil service contient tous les attributs et les caractéristiques qui peuvent identifier le service et le décrire suffisamment pour être découvert par les utilisateurs.
Il peut y avoir plusieurs versions d'un même service. Celles-ci diffèrent entre elles par exemple, par le tarif de la version proposée ou par les capacités du terminal nécessaires à l'exécution de cette version sur le terminal.
Dans le profil service, certains attributs sont communs à toutes les versions du service (le nom du fournisseur par exemple). D'autres peuvent avoir différentes valeurs pour chaque version (le coût de la version du service par exemple).
Lors de la découverte et la recherche de services, l'utilisateur peut consulter certaines informations sur les services disponibles (telles que la description du service, sa version, son

coût, etc.). Ces informations, extraites du profil service, permettent à l'utilisateur de décider pour chaque service, si ce dernier l'intéresse ou pas.
Le tableau 4.2 décrit les attributs d'un profil service. Un exemple XML de ce profil est présenté dans la figure 4.5.

	Paramètres	Description	Format
VASGEN	VASName	Nom du service	String
	VASID	Identificateur du service (unique)	Integer
	VASVersion	Numéro de la version	Integer
	VASDescription	Description du service	String
	UpdateDescription	Changements par rapport à la version précédente	String
VASP	VASPName	Nom du FS	String
	VASPPublicKey	Identifiateur unique du FS	String
	VASPReference	Adresse du FS	String
VASDETAIL	Language	Langue utilisée	String
	SubscriptionType	Indique si une inscription est nécessaire	Boolean
	ValidLocation	Information sur la localisation	String
	Category	Catégorie du service	String
	Keywords	Mots-clés	String
	Availability	Service on-ligne/hors-ligne	String
SECURITY	Confidentiality	Utilise SSL pour le téléchargement ou pas?	Boolean
	VASConditionsOfUse	Conditions d'utilisation	String
PROXIES	Proxy		
	ProxyName	Nom du proxy	String
	ProxyDescription	Description du proxy	String
	TC-NC	Besoins du terminal	
	Hardware	Besoins matériels	String
	Software	Besoins logiciels	String
	NWCaract	Caractéristiques du réseau	String
	BrowserUI	Caractéristiques du navigateur	String
	Communication		
	TransportProtocol	Protocol de transport (e.g. TCP)	String
	QoSIndicator	Gold, basic, standard	Integer
	URL	URL du proxy	String
	IPAddr	Adresse IP du proxy	String
	IPPort	Port IP du proxy	String
	TariffClassNumber	Tarif de cette version	Integer

TAB. 4.2 – Les paramètres d'un Profil Service

4.5.2.4 Profil Sécurité

Ce profil décrit les différents paramètres de sécurité gérés par le terminal. L'utilisation de ce profil intervient, en particulier, dans le processus de téléchargement du service sur le terminal, afin de permettre au fournisseur du service de coder le service de telle sorte que le terminal puisse le décoder une fois téléchargé.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE VAS SYSTEM  VASDescription.dtd">

<VAS>
 <VASGEN>
  <VASName>Mountain Rescue</VASName>
  <VASID>3408923</VASID>
  <VASVersion>4</VASVersion>
  <VASDescription>Setup emergency call to mountain </VASDescription>
  <UpdateDescription>Pricing Model was changed. </UpdateDescription>
 </VASGEN>

 <VASP>
  <VASPName>Provider Company</VASPName>
  <VASPublicKey>1234</VASPublicKey>
  <VASPReference>http_server@135.143.53.234/9000;
  password_server@122.66.34.44/8000</VASPReference>
 </VASP>

 <VASDETAIL>
  <Language>english>/Language>
  <SubscriptionType>no</SubscriptionType>
  <ValidLocation>Alpen</ValidLocation>
  <Category>Tourism</Category>
  <keywords>Abenteuer, Notruf</Keywords>
  <Availability>Online</Availability>
 </VASDETAIL>

 <SECURITY>
  <IPRProtection>yes</IPRProtection>
  <Confidentiality>SSL</Confidentiality>
  <VasConditionsOfUse>XXXX</VasConditionsOfUse>
 </SECURITY>

 <PROXIES>
  <ProxyName>MR1</ProxyName>
  <ProxyDescription>palm proxy for mountain rescue</ProxyDescription>
  <TC-NC>
    <Hardware>hardwarerequirements</Hardware>
    <Software>softwarerequirements</Software>
    <NetworkCharacteristics>networkcaracteristics</NetworkCh aracteristics>
    <BrowserUA>navigatorrequirements</BrowserUA>
  </TC-NC>
  <Communication>
    <TransportProtocol>UDP</TransportProtocol>
    <QoSIndicator>5</QoSIndicator>
  </Communication>
  <URL>www.palm-service-mr.org</URL>
  <IPAddr>111.111.111.111</IPAddr>
  <IPPort>9090</IPPort>
  <PricingModelNumber>45</PricingModelNumber>
  <TariffClassNumber>23</TariffClassNumber>
 </PROXIES>
</VAS>
```

FIG. 4.5 – Exemple XML du Profil Service

Autres détails et quelques exemples des différents profils décrits ci-dessus sont présentés dans l'annexe A.

4.5.3 Hébergement des profils

Un des aspects les plus importants de la gestion du contexte est la sauvegarde des informations contextuelles. Dans le cadre de la plate-forme CASP, il s'agit de la localisation et de l'hébergement des profils.

Les profils liés à l'environnement d'exécution de CASP sur le terminal doivent être disponibles sur ce dernier dès qu'une session CASP débute.
Le *Profil Terminal*, le *Profil de l'Interface Graphique de l'Utilisateur* et le *Profil Sécurité* sont spécifiques à chaque terminal. Ils sont donc stockés et hébergés par le terminal.
Afin d'éviter à l'utilisateur de reconfigurer toute l'interface graphique dans le cas de perte ou d'endommagement du terminal, le MS sauvegarde localement une copie des différents Profils de l'Interface Graphique de l'Utilisateur.
Les informations valides pour tout type de terminal utilisé par le même usager sont hébergées dans le MS afin de supporter la mobilité de l'utilisateur. Elles sont téléchargées sur le terminal à chaque nouvelle session CASP. Ces informations constituent le *Profil de Configuration de Services*.
Les *Profils Services* liés aux services disponibles dans la plate-forme, sont sauvegardés dans le MS pour fournir le maximum de flexibilité.

La figure 4.6 décrit les différentes localisations de chaque profil utilisé dans CASP.

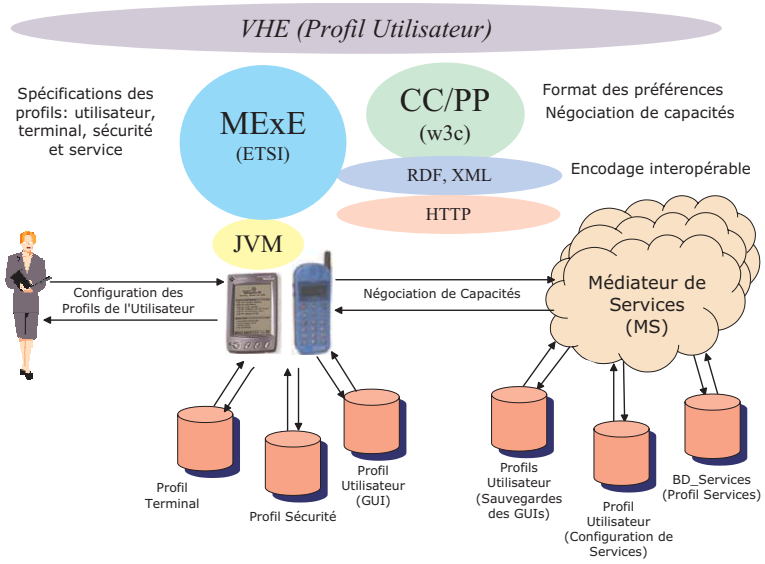


FIG. 4.6 – Gestion des profils dans CASP

4.5.4 La négociation des capacités et du contenu

L'un des principaux objectifs de la plate-forme CASP est la fourniture de services qui sont adaptés aux capacités des terminaux mobiles, c-à-d., qui peuvent s'exécuter sur le terminal de l'utilisateur. Pour ce faire, le terminal a besoin de transmettre ses capacités au MS qui les utilisera pour filtrer les services disponibles et pour ne proposer à l'utilisateur que les services que le terminal pourra supporter.
La négociation des capacités (ou la classification du terminal) est un mécanisme d'interaction entre le terminal et le MS qui permet, à chacune de ces deux entités, d'informer l'autre entité des mécanismes, des capacités et du support, que chacune peut fournir dans le cadre d'une interaction de découverte et de fourniture de services.
La négociation des capacités se fait avant tout processus de découverte de services par l'utilisateur. Elle permet ainsi au MS, en considérant les capacités du terminal, de filtrer les services disponibles dans la BD_Services pour composer la liste des services à proposer à l'utilisateur.
La négociation des capacités permet également aux services d'être développées indépendamment des plates-formes des différents terminaux. Les services peuvent ainsi viser une large gamme de terminaux de différentes capacités, allant d'un terminal de ressources très limitées (comme un PDA par exemple) à un terminal sophistiqué avec un environnement d'exécution complexe.
La négociation du contenu est un mécanisme d'interaction qui permet au MS et à l'utilisateur d'échanger des informations sur les différentes formes (versions) disponibles d'un même service afin de répondre au mieux aux préférences de l'utilisateur. Dans notre plate-forme, la négociation du contenu correspond à la sélection d'une des versions (proxies) du service personnalisé choisi par l'utilisateur. Elle se fait dans le cas où il existe plusieurs versions d'un même service dans la BD_Services (plusieurs implémentations qui diffèrent entre elles en terme d'interface graphique, langue utilisée, tarif associé, type de vidéo, caratéristiques de la vidéo, etc.).

La négociation des capacités et la négociation du contenu ont le même objectif : optimiser les réponses des requêtes de découverte de services suivant les capacités du terminal et les préférences de l'utilisateur.

4.6 Outils utilisés pour le développement de CASP

Les outils d'implémentation utilisés pour la réalisation de CASP que comme nous le verrons en détail au le chapitre 7, sont les suivants :

- XML et CC/PP pour la description des profils.
- Java pour le développement de CASP.
- HTTP pour la communication entre les différents composants de CASP et pour le téléchargement de services sur le terminal.

4.7 Conclusion

Dans ce chapitre, nous avons présenté brièvement la plate-forme CASP de découverte et de fourniture de services que nous avons développée dans le cadre du projet Européen MOBIVAS. Les solutions techniques choisies dans cette plate-forme ont pour but d'exploiter les paramètres qui définissent le contexte, afin d'assurer la meilleure qualité possible des services proposés aux utilisateurs mobiles.

Dans notre proposition, le contexte est défini par les paramètres suivants : les capacités du terminal, les préférences de l'utilisateur, la localisation du terminal et les ressources réseau. Il est basé sur l'utilisation des profils qui sont écrits en XML.
Comme nous allons le voir dans les chapitres suivants, le contexte intervient dans deux opérations principales de la fourniture de services : la découverte de services et l'adaptabilité des services en cours d'exécution.

Afin d'assurer la portabilité et l'intéropérabilité de la plate-forme CASP, le choix des outils de développement s'est porté sur l'utilisation des standards XML, MExE, CC/PP et HTTP. Une évaluation détaillée de la plate-forme CASP sera donnée dans le chapitre 7.

Dans les chapitres suivants, nous verrons comment la sensibilité au contexte est gérée dans la plate-forme CASP pour répondre aux besoins suivants : la découverte de services dans les environnements mobiles et l'adaptabilité dynamique des services en cours d'exécution. Pour chaque problématique, nous commençons d'abord, par dresser un état de l'art des solutions existantes ensuite, nous présentons la solution adoptée dans CASP et nous montrons ses apports.

Chapitre 5

La recherche et la découverte de services

L'évolution des services dans les réseaux est aujourd'hui incontestable. Un nombre grandissant de fournisseurs de services offrent des services de différents types et de différentes qualités. Il est donc nécessaire d'utiliser dans le réseau un mécanisme permettant le déploiement dynamique de ces services. D'autre part, ces services étant dynamiques, il est nécessaire de disposer de mécanismes de découverte qui permettent aux clients de s'informer sur les services proposés et de les découvrir de manière efficace et flexible, quelle que soit leur forme.
Ces mécanismes de découverte de services sont particulièrement indispensables pour les réseaux mobiles caractérisés par la mobilité et les fluctuations fréquentes dans la disponibilité des ressources. En effet, les terminaux dans ce type d'environnements peuvent se déplacer d'un réseau à un autre sans connaître les infrastructures utilisées dans chaque réseau. Cette transparence vis à vis de la nature des infrastructures utilisées et la possibilité de se connecter à des environnements initialement inconnus, empêchent les terminaux de tirer profit de certains services proposés et d'intéragir avec eux. Il est donc nécessaire de déployer des mécanismes qui permettent aux utilisateurs de découvrir automatiquement ces services sans pour autant reconfigurer leurs terminaux à chaque accès. De plus, ces mécanismes doivent prendre en considération le contexte de la découverte de services afin de ne proposer à l'utilisateur que les services qui répondent au mieux à ses besoins et qui peuvent s'exécuter sur son terminal (dans la limite des ressources disponibles : capacités du terminal et ressources réseau).

A ce jour, le support de la sensibilité au contexte dans les systèmes de découverte de services est peu fréquent.
Dans ce chapitre, nous identifions les principales fonctionnalités des protocoles de recherche et de découverte de services. Nous présentons ensuite, les principaux protocoles actuellement disponibles et nous les analysons par rapport à ces fonctionnalités. Enfin, nous montrons pourquoi ces protocoles ne peuvent pas être utilisés en l'état dans le cadre de la fourniture de services sensible au contexte.

5.1 Quelques concepts

La terminologie dans le domaine de la recherche et de la découverte de services n'est pas standardisée. Dans ce qui suit, nous adoptons celle utilisée dans [Rob00] et [OZO02].

1. ***Dispositifs et services***
   Les *dispositifs* et les *services* désignent les entités qui participent au processus de *recherche* ou de *découverte de services*.
   Les services regroupent l'ensemble des services réseau et des services à valeur ajoutée qui peuvent être mis à la disposition des entités. Les dispositifs incluent les ordinateurs classiques, les terminaux mobiles et d'autres terminaux spécifiques qui peuvent être utilisés dans un réseau tels que les caméras, les imprimantes ou les téléphones.
   Un dispositif connecté à un réseau peut proposer un ou plusieurs services. Une imprimante réseau, par exemple, peut fournir à la fois un service d'impression et un service fax.
   Pour les protocoles de recherche et de découverte de services, les dispositifs et les services sont équivalents (ce qui fonctionne pour l'un fonctionnera pour l'autre) et tous les deux représentent les *ressources du réseau*.
2. ***Recherche et découverte de services***
   La *recherche (lookup) de services* est le processus de localisation d'une ressource spécifique dans le réseau en utilisant son nom, son adresse ou une correspondance de certains critères. C'est une fonction passive dans la mesure où elle est déclenchée par une entité et elle nécessite l'existence d'un annuaire de services ou d'un autre élément pour répondre à la requête de l'entité.

   La *découverte de services* est un processus spontané dans lequel plusieurs ressources (non seulement l'annuaire de services comme dans le cas du lookup) découvrent d'autres ressources dans le réseau et se présentent elles-mêmes à d'autres ressources. L'objectif de la découverte de services et de faciliter la création et l'utilisation des services dans le réseau. Un mécanisme de découverte de services peut être utilisé pour la recherche de services mais plusieurs mécanismes de recherche ne supportent pas nécessairement la découverte de services.

5.2 Classification des systèmes de recherche et de découverte de services

Pour pouvoir utiliser des services dans un environnement mobile, les dispositifs ou les terminaux doivent être au courant de la présence de ces services ainsi que de leur mode d'exécution. *Les informations nécessaires pour l'utilisation d'un service sont collectées durant le processus de découverte de services.* Ce processus permet de fournir à l'utilisateur : la localisation du fournisseur du service, la description du service proposé, l'interface qui permet l'accès à ce service (proxy), etc.
La découverte de services peut être classée selon le niveau de connaissance initiale du client

sur le service, la relation entre le client et le fournisseur du service ainsi que les entités impliquées dans le processus de découverte de services [Pre02]. Les différentes classes forment une structure hiérarchique comme le montre la figure 5.1.

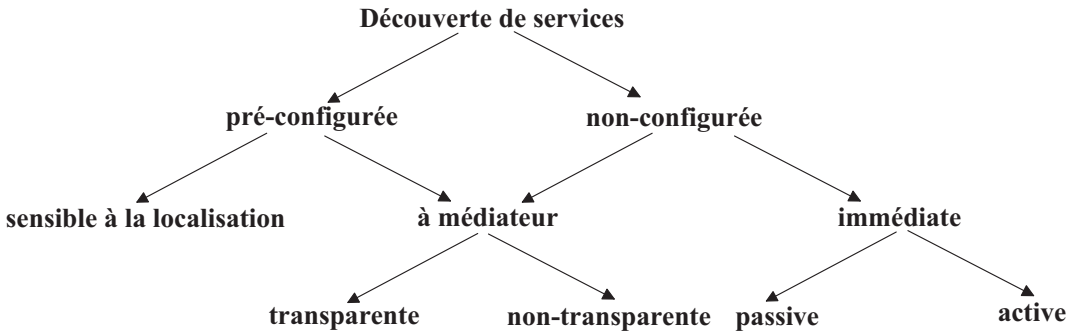


FIG. 5.1 – Classification de la découverte de services

La découverte de services peut être divisée en deux catégories principales. *Pré-configurée* où les entités qui découvrent les services possèdent des informations sur le fournisseur du service concerné ou sur l'entité qui permet de récupérer ce genre d'information. *Non-configurée* où les entités ignorent le contexte des services proposés. C'est la situation typique d'un réseau ad hoc [Per00].
Les deux catégories peuvent être, à leur tour, classées selon le nombre d'entités impliquées dans le processus de découverte de services. Les modes *sensible à la localisation* et *immédiat* sont caractérisés par une relation directe entre le client et le fournisseur du service. Le fournisseur lui-même propose au client toutes les informations nécessaires à l'utilisation de son service. Dans le mode *à médiateur*, un médiateur de services assure la découverte d'informations pour le compte des fournisseurs de services.
Dans la structure hiérarchique de la figure 5.1, on distingue au niveau des feuilles, cinq modes de découverte de services.
Dans le mode *sensible à la localisation*, les clients connaissent la localisation du service qu'ils veulent utiliser. De ce fait, la découverte du service se réduit à la recherche d'informations supplémentaires sur les caractéristiques du service ou sur l'interface d'accès au service.
Dans le mode *actif*, le client diffuse une requête dans le réseau pour retrouver un service. Les fournisseurs appropriés répondent par des informations sur la localisation du service désiré qu'ils proposent.
Dans le mode *passif*, les clients n'envoient pas de requêtes. Ils écoutent les messages d'annonce diffusés par les fournisseurs de services.
Les deux modes *transparent* et *non-transparent* utilisent un médiateur ou un annuaire d'information pour la découverte de services. Pour ce faire, les fournisseurs de services doivent enregistrer leurs services auprès du médiateur. Les deux modes diffèrent dans la conscience du client sur la présence du médiateur ou pas. Si le client utilise intentionnellement le mé-

diateur pour retrouver les services alors, il découvre de manière *non-transparente*. Le client découvre les services de manière *transparente* s'il pense qu'il interagit avec un fournisseur de services ordinaire alors qu'en réalité, il passe par un médiateur.

5.3 Principales fonctionnalités d'un protocole de recherche ou de découverte de services

La recherche ou la découverte de services permettent à chaque ressource dans le réseau de trouver les services disponibles de manière automatique sans avoir besoin d'une configuration manuelle. Les technologies de recherche et de découverte de services sont caractérisées par : la façon de décrire les services, la centralisation ou la décentralisation des descriptions des services, la déclaration et la découverte de ces descriptions ainsi que la gestion de la dynamicité des services.

5.3.1 Description du service

La description du service est l'une des principales caractéristiques de la recherche et de la découverte de services. Elle permet de décrire les fonctionnalités fournies par les services. Le service doit être décrit d'une façon assez expressive pour que n'importe quel client puisse appréhender clairement les caractéristiques du service et donc décider de l'utiliser ou pas. Le format de description de services doit être standardisé afin de permettre l'interopérabilité entre les différentes entités impliquées dans le processus de découverte de services [Rob00]. Les descriptions de services doivent se conformer à un certain schéma afin de maintenir la compatibilité entre les services et de permettre aux clients d'interpréter ces descriptions [Gur03].

5.3.2 Utilisation de répertoire ou d'annuaire central

Le répertoire ou l'annuaire central est un catalogue de descriptions de services publiés par les fournisseurs. Ces derniers y enregistrent leurs services et les clients le consultent pour rechercher et découvrir les services disponibles. L'annuaire peut être logiquement centralisé mais physiquement réparti.
Centraliser ou non les descriptions des services est un choix critique qui oppose la performance et le passage à l'échelle, à la tolérance aux fautes et la dépendance d'une infrastructure fixe.
L'existence d'un tel annuaire évite la nécessité de diffuser les descriptions de services et les requêtes de recherche d'un service en broadcast ou en multicast sur le réseau. Un point de consultation facilite les découvertes de services et augmente les performances. Néanmoins, un tel point peut rendre le système sensible aux pannes. D'autre part, il peut exister des situations où l'implémentation d'un tel annuaire n'est pas possible comme dans le cas des réseaux ad hoc.

5.3.3 Déclaration de services

La déclaration consiste à annoncer dans le réseau, la présence d'un nouveau service et la possibilité de l'utiliser. Elle peut se faire selon deux manières. La première, consiste à envoyer périodiquement des messages dans le réseau pour annoncer l'existence du service. La deuxième, consiste à enregistrer la description du service auprès d'un annuaire de service en précisant la durée de validité de cet enregistrement.
L'enregistrement génère moins de trafic que l'annonce périodique mais il est moins robuste. Des services peuvent tomber en panne avant l'expiration de la durée de validité de leur enregistrement au niveau des annuaires.

5.3.4 Découverte de services

Le processus de recherche ou de découverte de services est déclenché lorsque le client formule une requête dans laquelle il exprime les critères de choix des services qu'il veut récupérer. Il peut exister plus d'un service répondant à ces critères. Donc, plus la requête est expressive et bien formulée, plus le résultat de la recherche est pertinent pour le client. Si le protocole de découverte de services est sensible au contexte, la sensibilité au contexte doit être gérée durant la phase de découverte de services. Dans ce cas, la réponse à la requête du client doit, non seulement prendre en considération les critères du choix du client, mais aussi le contexte de la découverte de services tel que les capacités du terminal, sa localisation, les ressources réseau, etc.

5.3.5 Gestion de la dynamicité du système

Certains réseaux sont très dynamiques si bien que des mécanismes comme le *leasing* et/ou la *notification* sont nécessaires pour les gérer.
Le *leasing* est un paramètre qui indique la période de temps pendant laquelle le service reste valide. Avant l'expiration de cette période, le fournisseur peut renouveler le leasing. Si le leasing n'est pas renouvelé, le service est considéré non disponible.
La notification consiste à informer les clients de l'arrivée, du départ et des changements d'état d'un service.

5.4 Techniques de découverte ou de recherche de services

Dans ce qui suit, nous présentons les techniques de découverte de services qui sont également valables pour la recherche de services.
Les fonctionnalités citées précédemment fournissent les outils nécessaires pour la découverte de services. Ainsi, grâce à la description des capacités de chaque service, le protocole de découverte de services permet de comparer les besoins du client aux capacités offertes par les autres ressources dans le réseau. Cette comparaison peut être implantée de plusieurs façons.

La comparaison entre les besoins des clients et les fonctionnalités offertes par les services, peut être réalisée par le client. Dans ce cas, le protocole de découverte de services demande à chaque ressource du réseau, d'envoyer la description des fonctionnalités qu'elle propose. Dans cette approche, le client doit analyser chaque description pour retrouver celle qui correspond le mieux à ses besoins. De plus, une importante quantité d'information peut être envoyée par chaque ressource, ce qui peut surcharger le réseau.

La comparaison peut également être effectuée par le serveur. Dans ce cas, le protocole envoie à chaque ressource du réseau la liste des critères que le client exige. Chaque ressource consulte cette liste et répond positivement si elle satisfait les critères. Cette technique est adaptée aux environnements limités en terme de bande passante mais elle place la charge de la comparaison sur chaque serveur.

Une entité intermédiaire entre le client et les autres ressources peut participer à la découverte de services. L'annuaire de services assure un contact initial avec les ressources du réseau, collecte et enregistre les informations sur les fonctionnalités et les capacités de chaque ressource. Les clients envoient leurs requêtes à l'annuaire de services qui leur réalise la comparaison.
Cette approche permet de contôler l'utilisation de la bande passante puisqu'elle limite le nombre de requêtes. Cependant, elle ajoute une complexité au réseau parce qu'elle nécessite la présence d'un annuaire de services.

5.5 Systèmes de recherche de services

Il existe un grand nombre de systèmes de recherche (ou de lookup) de services. Dans ce qui suit, nous présentons brièvement les systèmes suivants : le DNS, le LDAP et le *Trader Service* de CORBA.

Le protocole DNS (Domain Name Service) [Moc87] est un standard IETF (Internet Engeineering Task Force) [IET] utilisé pour retrouver les adresses IP des machines dans Internet. Il fournit une base de données permettant la mise en correspondance entre des adresses physiques (adresses IP) et des adresses logiques (noms de machines) dans le réseau.
DNS est organisé sous forme hiérarchique et permet de garantir l'unicité d'un nom de machine. Lorsqu'il faut retrouver l'adresse physique d'une ressource, les serveurs qui gèrent le DNS s'envoient des requêtes de façon à remonter suffisamment dans la hiérarchie pour trouver l'adresse physique du correspondant. Ces requêtes sont effectuées par l'intermédiaire de petits messages qui portent la requête à l'aller et la réponse au retour.

LDAP (Lightweight Directory Access Protocol) [WHK97] est un protocole standard permettant de gérer des annuaires, c'est-à-dire d'accéder à des bases d'informations sur un réseau. Ces bases peuvent contenir toute sorte d'informations que ce soit des coordonnées de personnes ou des données système.
LDAP présente les informations sous forme d'une arborescence d'informations hiérarchiques appelée **DIT** (*Directory Information Tree*), dans laquelle les informations appelées **entrées** (ou encore DSE pour *Directory Service Entry*), sont présentées sous forme de

branches.
LDAP fournit un ensemble de fonctions pour effectuer des requêtes sur les données afin de rechercher, insérer, modifier ou effacer des entrées dans les répertoires.

Le *Trader Service* de CORBA [Gro97] est un service de "*pages jaunes*" pour les objets CORBA. Il leur permet de déclarer et de découvrir des services dans un système distribué. Le *Trader Service* joue le rôle de médiateur entre les serveurs qui proposent les services (exportateurs) et les utilisateurs de ces services (importateurs). Les exportateurs qui sont des objets CORBA, utilisent des interfaces fournies par le *Trader Service* pour enregistrer leurs services. Les informations sur les services enregistrés contiennent une référence de l'objet fournissant le service, des informations sur les opérations supportées par le service ainsi que d'autres propriétés décrivant les capacités du service. Le *Trader Service* sauvegarde ces descriptions dans un annuaire et maintient une base d'informations sur les objets. Les importateurs ou les clients, qui sont aussi des objets CORBA, peuvent envoyer des requêtes au *Trader Service* pour récupérer la liste des services disponibles ou pour récupérer un service particulier en spécifiant certaines critères.
Les *Trader Services* peuvent être associés pour former une fédération de *Trader Services*. Cette association permet à chaque *Trader Service* de proposer de manière transparente à ses propres clients les services offerts par les autres *Trader Services*.
Le *Trader Service* est défini comme une interface CORBA et toutes les déclarations, les requêtes et les réponses sont des objets CORBA. Comme le *Trader Service* dépend de CORBA, tous les participants au processus de recherche de services (clients et serveurs) doivent être des objets CORBA et doivent utiliser les protcoles de CORBA.

Les systèmes de recherche de services présentés dans cette section ne gèrent pas directement la découverte spontanée des services.

5.6 Protocoles de découverte de services

Les protocoles de découverte de services fournissent des mécanismes pour la découverte spontanée et dynamique de services disponibles dans le réseau. Dans ce qui suit, nous donnons un aperçu sur les protocoles les plus représentatifs et nous montrons comment chaque protocole assure les différentes fonctionnalités présentées dans la section 5.3.

5.6.1 Service Location Protocol (SLP)

Le protocole de localisation de services SLP (Service Location Protocol) [Gut99] de Sun Microsystems [SUN] est un standard de l'IETF [IET] pour la découverte de services spontanée dans les réseaux IP. Il est basé sur une infrastructure utilisant trois entités de base :

- L'**Agent Utilisateur** (UA pour *User Agent*) qui envoie les requêtes de recherche de services pour le compte d'un client (utilisateur ou application),
- L'**Agent Service** (SA pour *Service Agent*) qui annonce et affiche la localisation des services et leurs caractéristiques pour le compte des services,

- L'**Agent Répertoire** (DA pour *Directory Agent*) qui collecte dans sa base de données, les adresses des services et les informations reçues des SAs. Il répond également aux requêtes des UAs.

Lorsqu'un nouveau service se connecte au réseau, le SA contacte le DA pour annoncer son existence (*Enregistrement Service*). De même, si un client a besoin d'un service particulier, l'UA envoie une requête de recherche de services au DA (*Requête Service*).
Pour pouvoir contacter le DA, l'UA et le SA doivent retrouver son adresse. Pour cela, SLP propose trois méthodes différentes de découverte de DA : *statique*, *active* et *passive*.
Dans la découverte statique, les agents SLP obtiennent l'adresse du DA grâce au protocole DHCP (Dynamic Host Configuration Protocol) [Dro99]. Les options DHCP nécessaires pour l'utilisation de SLP sont définies dans [PG99].
Dans la découverte active, les UAs et les SAs envoient les requêtes à une adresse de diffusion allouée à SLP. Un DA en écoute sur cette adresse, répond en mode unicast à ses requêtes.
Dans le cas de la découverte passive, les DAs diffusent périodiquement des messages de présence sur le réseau. Ainsi, les UAs et les SAs récupèrent l'adresse des DAs et les contactent directement.

Voyons maintenant comment SLP assure les fonctionnalités que doit garantir un protocole de découverte de services.

1. *Description des services*

   SLP utilise une URL (*"Service URL"*) et un ensemble d'attributs *"Service Templates"* sous forme de texte pour décrire les services [GPK99]. Le *"Service URL"* contient le nom du service, le protocole que le client doit utiliser pour communiquer avec le serveur, l'adresse IP du service, le numéro de port et le chemin d'accès. Les *"Service Templates"*, standardisés par IANA (Internet Assigned Numbers Authority) [Gut99], sont des attributs qui décrivent le service et ses caractéristiques.
   Un exemple de description d'un service SLP est le suivant :

   URL = service :printer :lpr :hostname :port/path

   Attributs = (printer-name = lj4050), (printer-model = HPLJ4050),
   (printer-location = Room C214), (color-supported = false),
   (pages-per-minute = 9)

2. *Centralisation des descriptions*

   L'agent DA est utilisé pour centraliser les descriptions des services. Sa présence dans une architecture SLP est optionnelle. Il est particulièrement utilisé dans les réseaux à large échelle disposant de plusieurs services parce qu'il permet de les catégoriser. Dans les autres réseaux (comme les *"home networks"* et les *"car networks"*), SLP est déployé sans DA. SLP dispose donc de deux modes opérationnels selon qu'il possède un DA ou pas : *centralisé* avec un répertoire DA et *décentralisé* sans DA.

3. *Déclaration de services*

   La déclaration d'un nouveau service dans SLP se fait comme suit :

   – Dans le mode centralisé, le SA contacte le DA pour annoncer l'existence du nouveau service (Enregistrement Service). Pour ce faire, il utilise les *"Service URL"* et *"Service Templates"* qui décrivent le service pour enregistrer ce dernier auprès du DA.

   – Dans le mode décentralisé (sans répertoire), les SAs diffusent périodiquement des messages de présence en multicast pour mettre les UAs au courant de l'existence de leurs nouveaux services.

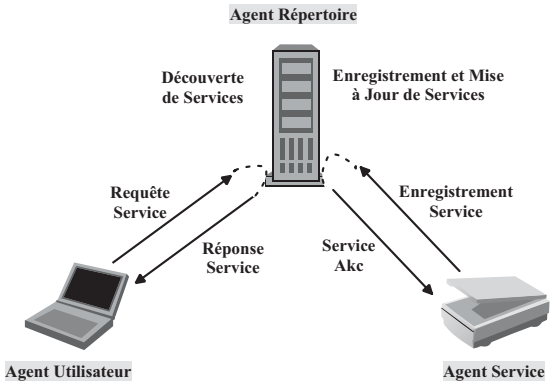


FIG. 5.2 – Agents SLP et leurs interactions

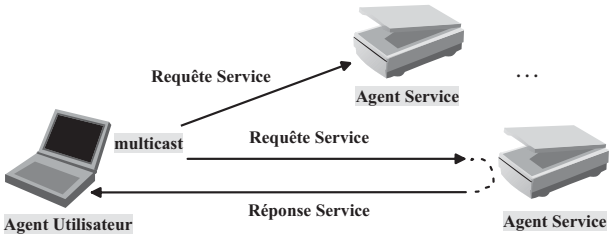


FIG. 5.3 – Découverte de services sans le DA

4. *Découverte de services*

   Dans le mode centralisé, les requêtes de recherche de services sont envoyées au DA qui les évalue. Dans le mode décentralisé, les UAs diffusent périodiquement en multicast les requêtes de recherche de services. Si les SAs trouvent que leurs services répondent à la requête, ils contactent le client.

   SLP supporte la recherche de services basée sur des mots-clés. Il compare ces derniers aux attributs de chaque service disponible dans le réseau. Si le service est conforme à la requête, l'URL du service est envoyée au client. SLP permet donc aux UAs de sélectionner le service le plus approprié parmi ceux disponibles dans le réseau.

SLP offre également aux UAs la possibilité de formuler des requêtes expressives en utilisant des opérateurs booléens (OR et AND), des comparateurs ($<$, $>$, $=$, ...) ou des sous-chaînes.

5. *Gestion de la dynamicité*
   Dans le mode centralisé, SLP exige un leasing au moment de l'enregistrement du service. Le fournisseur du service doit spécifier la durée de validité de cet enregistrement.

5.6.2 Le service lookup de Jini

Jini de Sun Microsystems est un environnement Java distribué qui supporte la découverte spontanée de services [Mic99]. Dans cet environnement, les services Jini peuvent être des équipements matériels, des applications et des programmes logiciels ou une combinaison des deux. Une collection de services Jini forme ce qu'on appelle *une fédération Jini*.
L'objectif de Jini est de transformer le réseau en une entité dynamique permettant d'une part, la découverte spontanée des services par les clients (services ou terminaux) et d'autre part, le rajout et la suppression de services du réseau et ceci de manière flexible.

Jini utilise trois entités de base : les *services* à proposer aux clients, les *Lookup Services (LSs)* qui cataloguent les services disponibles et, les *clients* qui utilisent les services. Jini ne spécifie pas le nombre de LSs dans un réseau.
Un service peut être à la fois un client ; par exemple, une caméra peut fournir des photos à un PDA en tant que service et peut utiliser un service d'impression en tant que client.

Pour enregistrer un nouveau service ou découvrir les services disponibles, les clients et les services doivent d'abord localiser un des LSs présents dans le réseau. Pour ce faire, ils utilisent un protocole de découverte basé sur le multicast. Ce protocole est encapsulé dans un ensemble de classes qu'offre Jini. Une fois le LS découvert, les clients et les services obtiennent des *Stub RMI* leurs permettant de communiquer avec le LS trouvé.

1. *Description des services*
   Les descriptions des services dans Jini sont présentées sous forme de proxy. Le proxy peut avoir des attributs qui sont des objets Java décrivant les caractéristiques du service. Jini ne spécifie pas les attributs d'un service. Les fournisseurs de services sont libres de décrire les services comme ils le souhaitent.
   Chaque proxy est une instance de la classe *ServiceItem* de Jini. Comme le montre l'exemple suivant, la classe *ServiceItem* définit le service sous forme d'un objet, son identificateur, ses attributs et ses interfaces.

```
public class ServiceItem implements Serializable {
          public ServiceItem(ServiceID serviceID,
                    Object service,
                    Entry[] attributeSets) {...}
          public ServiceID serviceID ;
          public Object service ;
```

```
                public Entry[] attributeSets;
    }
```

Les attributs sont décrits par la classe *Entry* de Jini qui regroupe les attributs de même caractère.

2. *Centralisation des descriptions*

   Tous les enregistrements des nouveaux services et les requêtes de localisation de services doivent passer par le LS en utilisant le protocole Java RMI [Ric00]. Les interactions entre les clients et les services sont basées sur les objets Java.

3. *Déclaration de services*

   Lorsqu'un service joint le réseau, il enregistre un *proxy* auprès du LS. La figure 5.4 présente un exemple de déclaration d'un service (service *"imprimante"*) dans Jini.

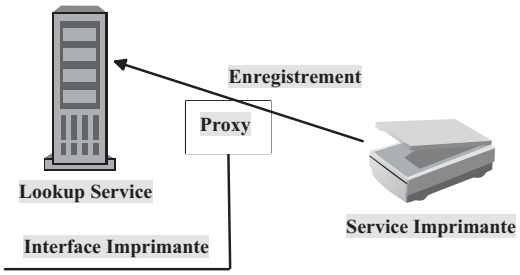


FIG. 5.4 – Enregistrement de services dans Jini

4. *Découverte de services*

   Une requête de recherche de services envoyée par le client est une instance de la classe *ServiceTemplate* de Jini. Le client peut spécifier l'identificateur du service à rechercher, son type ou les valeurs de ses attributs.
   L'évaluation des requêtes dans Jini est basée sur l'égalité et la correspondance exacte entre les paramètres de la requête et les attributs du service. Ainsi, Jini ne permet pas d'évaluer des requêtes compliquées avec des opérateurs booléens comme SLP par exemple.
   Une fois le service découvert et sélectionné, le LS retourne au client le *proxy* correspondant (voir figure 5.5). Le client télécharge ensuite le code du proxy si ce dernier n'existe pas sur sa machine. La communication entre le client et le service se fait par l'invocation d'une méthode spécifique du proxy (voir figure 5.6).

5. *Gestion de la dynamicité*

   Jini est basé sur l'utilisation des contrats (*leasing*) entre les LSs et les autres entités (clients et services). Tous les enregistrements des services se font pour une période déterminée de temps. Les clients et les services s'exécutant pour une longue période, doivent renouveler périodiquement leurs contrats. Lorsque leur contrat expire, le LS supprime automatiquement les descriptions des entités qui tombent en panne ou se

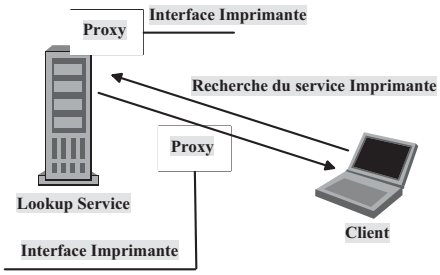


FIG. 5.5 – Découverte de services dans Jini

FIG. 5.6 – Utilisation du service dans Jini

déconnectent.
Jini permet également de notifier aux clients le départ et l'arrivée d'un service ainsi que les changements de ses valeurs d'attributs. Pour cela, le client doit s'inscrire auprès du LS pour de tels changements.

5.6.3 Universal Plug and Play (UPnP) et Simple Service Discovery Protocol (SSDP)

UPnP [CL99] est une architecture proposée par Microsoft pour déployer une infrastructure de réseau permettant la communication entre différents appareils de marques et de technologies différentes. UPnP vise à déployer un réseau de communication dans les maisons et dans les locaux d'entreprises où les membres communicant ne sont pas forcément des PCs.
UPnP permet aux dispositifs de joindre dynamiquement un réseau, obtenir une adresse IP, transmettre des services et connaître les équipements déjà connectés ainsi que leurs services.
La découverte de services dans UPnP est basée sur le protocole SSDP (*Simple Service Discovery Protocol*) [GCPLA99].

1. *Description des services*
   UPnP utilise XML (*eXensible Markup Language*) comme langage de description des dispositifs (terminaux) et des services qu'ils offrent. Ces descriptions contiennent des informations fournies par le constructeur du dispositif (type de dispositif, nom du constructeur, identificateur du dispositif, etc.), une liste de services que le dispositif fournit et des URLs pour récupérer les descriptions des services.

Chaque description de service contient une liste de méthodes invocables par un client et une liste de variables dont les changements de valeur peuvent être notifiés aux clients.

2. *Centralisation des descriptions*
   UPnP fonctionne en mode décentralisé et sans annuaire principal.

3. *Déclaration de services*
   Lorsque un nouveau service veut joindre le réseau, il diffuse une annonce en multicast dans le réseau pour indiquer sa présence. L'annonce est effectuée périodiquement. Elle inclut l'URL où les dispositifs peuvent récupérer le document de description du dispositif et ses services.

4. *Découverte de services*
   Lorsqu'un client veut découvrir un service, il peut soit envoyer un message en multicast sur le réseau soit, contacter directement le service via son adresse URL récupérée suite à l'annonce de sa présence. Dans le premier cas, la réponse à la requête est envoyée en unicast par le service lui-même.
   UPnP ne permet pas de formuler des requêtes sur des attributs. Il restreint aux clients la connaissance du service à son nom, son type ou son adresse. Un client UPnP qui a récupéré la description du service désiré, doit explicitement demander les valeurs des attributs en envoyant un message pour chaque attribut. L'évaluation de la requête est donc effectuée par le client.

5. *Gestion de la dynamicité*
   UPnP assure le leasing et la notification. En effet, chaque nouveau service doit s'annoncer en multicast en précisant la durée de sa disponibilité dans le réseau. Cette annonce est renouvelée quand le service étend sa durée de vie.
   En ce qui concerne la notification, la description UPnP d'un service inclut une liste de variables qui définissent l'état du service. Le dispositif peut diffuser leur mise à jour lorsque ces variables changent et les clients peuvent souscrire au service afin de recevoir cette information.

5.6.4 Salutation

Salutation est un protocole de découverte de services développé par le consortium Salutation [Con99]. C'est un standard ouvert indépendant des systèmes d'exploitation, des protocoles de communication et des plates-formes physiques.

Salutation a été créé pour résoudre les problèmes de découverte et d'utilisation de services dans les environnements caractérisés par l'hétérogénéité et la mobilité de leurs dispositifs. Salutation assure l'interopérabilité entre des entités réseau ayant des environnements d'exécution hétérogènes et utilisant des protocoles de transport différents.

Salutation fournit des moyens aux applications, aux services et aux dispositifs pour décrire et annoncer leurs capacités aux autres entités. Il leur permet également de chercher

d'autres entités d'une capacité particulière et d'établir des sessions avec elles pour utiliser leurs capacités.

1. *Description des services*

   Salutation définit un format spécifique pour la description et la localisation des services. Chaque service est décrit avec un *Functional Unit Description Record*. Cet enregistrement contient le nom du service, son identificateur et une liste d'attributs : *Attribute Record*. Chaque Attribut Record contient l'identificateur de l'attribut, l'identificateur de la fonction de comparaison qui servira à comparer la valeur de cet attribut en cas de requête et finalement, la valeur de l'attribut.

2. *Centralisation des descriptions*

   Salutation est composé de deux principaux composants : le SLM (*Salutation Manager*) et le TM (*Transport Manager*). Le SLM est le cœur de l'architecture. C'est l'annuaire des descriptions de services disponibles. Il fonctionne au dessus du TM qui assure des communications fiables indépendamment des protocoles de transport réseau utilisés.

   Un même SLM peut présenter un ou plusieurs clients et/ou services. Il peut découvrir d'autres SLMs et les services qu'ils proposent. La coopération entre les SLMs forme conceptuellement un service lookup similaire à celui de Jini. La seule différence est qu'il est distribué dans le réseau. La communication entre les SLMs se fait via RPC (*Remote Procedure Call*) qui permet, même à des clients et à des services ne possédant pas de SLM, d'utiliser des SLMs distants.

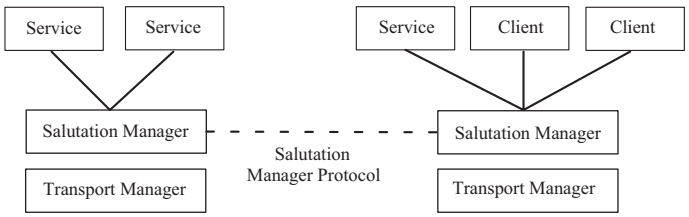


FIG. 5.7 – Architecture de Salutation

3. *Déclaration de services*

   Chaque client peut enregistrer son service auprès de son SLM local (si le SLM existe sur le même dispositif que le client) ou auprès du SLM le plus proche (en utilisant le protocole RPC).

4. *Découverte de services*

   Les clients localisent les services en envoyant des requêtes de recherche de services au SLM local qui coopère ensuite avec les autres SLMs pour accomplir cette tâche. Le client peut demander soit la liste de tous les services disponibles, soit la liste des services d'un type particulier, soit la liste des services qui répondent à un attribut particulier. En réponse, le SLM envoie l'adresse et la description du service.

Chaque requête de recherche de service est présentée sous forme de *Service Description Record*. Elle contient une liste d'attributs avec les valeurs désirées par le client. Grâce aux fonctions de comparaisons de Salutation (*strEqualTo, intEqualOrGreaterThan, setIntIntersect, etc.*), Salutation permet aux clients de faire des requêtes très expressives. Ces requêtes sont évaluées par le SLM qui retourne les descriptions des services conformes aux besoins du client.

5. *Gestion de la dynamicité*

   Salutation n'implémente pas le mécanisme de *leasing* mais un client Salutaion peut demander à son SLM de contrôler la disponibilité d'un service en envoyant périodiquement des messages de vérification.
   Les clients Salutation peuvent également s'inscrire pour être informés des changements des variables dynamiques d'un service.

5.6.5 Secure Service Discovery Service (SSDS)

Le protocole SSDS [CZH+99] a été proposé dans le cadre du projet de recherche Ninja [NIN] de l'Université de Californie, Berkeley. SSDS est une architecture de déploiement et de découverte de services à grande échelle. Elle inclut principalement des serveurs, un gestionnaire de certificats et un gestionnaire de capacités. Chaque serveur a pour rôle de sauvegarder des informations sur les services disponibles dans les domaines qu'il couvre.
Les principaux composants du système SSDS et leur rôle dans le processus de découverte de services sont les suivants :

- *Serveurs SDS (Service Discovery Service)* : les serveurs SDS jouent le rôle d'annuaires de services. Ils sont organisés en domaines hiérarchiques où chaque domaine spécifie une étendue dans le réseau. Les serveurs SDS disponibles diffusent périodiquement en multicast des messages authentifés contenant leur adresse URL.
- *Services* : les services écoutent les messages des serveurs SDS pour récupérer leurs adresses. Ils diffusent ensuite, en multicast, les descriptions des services chiffrées et authentifiées.
- *Clients* : les clients découvrent le serveur SDS de leur domaine en écoutant sur une adresse SDS connue. Ils utilisent un protocole RMI authentifié [CZH+99, Wel99] pour contacter le serveur SDS et pour envoyer les requêtes de découverte de services en format XML.
- *Autorité du certificat CA (Certificate Authority)* : le SDS utilise des certificats signés par le CA pour l'authenticité des liens entre les composants du système SSDS et leurs clés publiques.
- *Gestionnaire des capacités CM (Capability Manager)* : le SDS utilise les capacités (des clés privées appropriées) comme mécanisme de contrôle d'accès pour permettre aux services de contrôler l'ensemble des utilisateurs autorisés à découvrir leur existence. A la demande du service, le CM génère et distribue les capacités à tous les utilisateurs ayant le droit d'utiliser ce service. Seuls les clients possédant des capacités peuvent accéder au service.

Dans SSDS, la découverte de services est gérée comme suit :

1. *Description des services*
   Les services dans SSDS sont décrits sous forme de documents XML.
2. *Centralisation des descriptions*
   SSDS peut opérer avec ou sans le SDS qui joue le rôle d'annuaire central des descriptions de services.
3. *Déclaration de services*
   Lorsqu'un nouveau service veut s'associer au réseau, il envoie une annonce dans le réseau pour indiquer sa présence aux différents dispositifs. L'annonce peut être envoyée soit en unicast au SDS soit, en multicast dans le réseau. Dans le dernier cas, l'annonce peut être interceptée par tous les dispositifs du réseau.
4. *Découverte de services*
   La formulation des requêtes de recherche de services est basée sur XML et leur évaluation est réalisée en comparant les balises XML des requêtes avec celles des descriptions des services. Lorsque le client veut découvrir un service, il peut soit contacter le service directement à travers l'URL enregistrée suite à la déclaration du service soit, il envoie une requête en multicast dans le réseau. Dans le dernier cas, la réponse à cette requête peut être envoyée soit par le SDS, soit par le service lui-même.
   Si le SDS existe, il utilise son parseur XSet XML [ZJ00] pour rechercher les descriptions de services qui répondent aux besoins du client.
5. *Gestion de la dynamicité*
   Le service indique au SDS, au moment de son déclaration, la période de temps pour laquelle le service reste valide. Après l'expiration de cette période, le service est considéré comme indisponsable par le SDS.

Comparé aux autres protocoles de découverte de services, SSDS a apporté des améliorations considérables par rapport à la fiabilité, le passage à l'échelle et la sécurité. SSDS est implanté en Java. Il utilise Java RMI pour les appels distants et XML pour la description et la localisation des services (au lieu d'utiliser des objets Java comme c'est le cas de Jini) [CZH+99].
SSDS assure une sécurité extrêmement forte. Toutes les parties du système sont authentifiées et tous les messages échangés à travers le réseau sont chiffrés. Les aspects de sécurité supportés comprennent : l'authenticité du service découvert et de sa description, l'authenticité de tous les composants du système et enfin l'authenticité et le chiffrement des invocations distantes. L'organisation hiérarchique des serveurs SDS assure la mise à l'échelle du système, la détection des pannes dans le système et le redémarrage automatique des serveurs en panne.

5.6.6 Bluetooth-SDP

Bluetooth [Con01] est une technologie sans fil qui permet aux utilisateurs de faire des connexions faciles, sans fil et instantanées entre des dispositifs de communication divers

comme les téléphones portables, les ordinatuers et les PDAs.

La pile protocolaire de Bluetooth contient le protocole SDP (Service Discovery Protocol) de découverte de services fournis par les terminaux Bluetooth [SDP99].

SDP permet à un client SDP d'accéder à des informations sur des services proposés par des serveurs SDP. Un serveur SDP peut être n'importe quel terminal Bluetooth qui fournit un service à utiliser par un autre terminal Bluetooth. Un client SDP peut être n'importe quel terminal Bluetooth pouvant utiliser les services. Un terminal Bluetooth peut être à la fois, un client SDP et un serveur SDP.

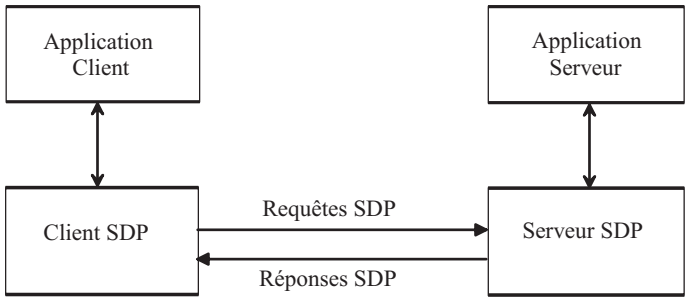


FIG. 5.8 – Protocole SDP dans Bluetooth

Contrairement aux autres protocoles de découverte de services, le SDP de Bluetooth est spécifique aux terminaux Bluetooth. Il ne fournit pas de mécanismes permettant d'accéder aux services et d'annoncer ou d'enregistrer un nouveau service. Pour pouvoir utiliser un service découvert par le protocole SDP, le terminal doit faire appel à un autre protocole de haut niveau pour effectuer toutes les tâches nécessaires à l'accès et à l'utilisation du service.

1. *Description des services*

   Les services SDP sont classés selon une structure hiérarchique de *classes de services*. Un service est une instance d'une *classe de services*. Il est représenté par un *Service Record* qui est composé d'un ensemble d'attributs. Chaque attribut a un identificateur ID et une valeur de type spécifique de données. Les *classes de services* définissent les catégories des terminaux (*Printer, ColorPrinter, PostScriptPrinter, etc.*). Les attributs permettent de décrire le service en détail et fournissent des informations sur son utilisation [BS01].
   Certains attributs dans Bluetooth, dits standards, sont communs à tous les *Service Record* et possèdent un identificateur de type UUID (*Universally Unique IDentifier*).

2. *Centralisation des descriptions*

   Il n'existe pas d'annuaire central de descriptions de tous les services disponibles dans le réseau, comme c'est le cas de Jini par exemple. Chaque serveur SDP maintient

une liste de descriptions qui décrivent les services qu'il propose lui-même (services locaux).

3. *Déclaration de services*
   SDP ne fournit pas de mécanisme pour la déclaration de nouveaux services dans le réseau.

4. *Découverte de services*
   SDP propose aux terminaux Bluetooth une API pour découvrir les terminaux voisins et les services disponibles qu'ils proposent. SDP supporte la recherche de services par classe, par attributs ou par navigation. La recherche par attributs s'effectue uniquement sur les attributs dont les identificateurs sont de type UUID. La recherche par navigation est utilisée lorsqu'un client Bluetooth ne possède aucune connaissance préalable sur les services disponibles. Elle permet ainsi de retrouver la liste de tous les services dans le voisinage.

5. *Gestion de la dynamicité*
   Bluetooth ne propose pas de mécanisme de notification qui permet d'informer les utilisateurs lorsque le service devient indisponible.

5.6.7 MOCA

MOCA d'IBM [BGI99] est une infrastructure de services écrite entièrement en Java et conçue pour les terminaux d'environnements mobiles de capacités limitées. Elle est basée sur la notion de *service* et *d'application*. Une application est composée d'un ensemble de services coopératifs. Un service est un composant logiciel qui assure une fonction particulière.
Le *Service Registry* est un élément central de l'architecture MOCA. Il joue le rôle de gestionnaire de descriptions de services locaux et assure leur recherche. Il gère également le cycle de vie des différents services en assurant l'enregistrement, la mise à jour et, la suppression des descriptions de services.

Les services peuvent être utilisés par d'autres applications et services s'exécutant sur d'autres terminaux mobiles. L'accès à un service peut être limité par une politique de sécurité définie par le fournisseur du service. Les services et les applications utilisés par un terminal, peuvent être locaux ou téléchargés dynamiquement à travers le réseau.

Le mécanisme de découverte de services de MOCA est implanté par deux services optionels sur chaque terminal : le *Broadcaster* et le *Listener*. Le Broadcaster maintient la liste des services offerts par le terminal et transmet cette information périodiquement sur le réseau. Le Listener écoute les annonces de services et accepte ou rejette les services offerts par les terminaux voisins.

1. *Description des services*
   Chaque service est représenté par une description contenant trois entités : le nom de l'interface du service (*InterfaceName*), le nom de l'implémentation du service

(*ImplementationName*), et une URL. Cette dernière indique la localisation de l'implémentation du service. La valeur de ce champ est optionnelle. Si elle n'est pas spécifiée, le *Service Registry* prend une localisation par défaut.

2. *Centralisation des descriptions*
   MOCA ne possède pas d'annuaire central de descriptions de services.

3. *Déclaration de services*
   La déclaration de services dans MOCA est basée sur un modèle multicast. Les terminaux mobiles envoient périodiquement une liste de services à travers le réseau. Chaque service est représenté par sa description. Les terminaux voisins écoutent les messages d'annonce sur le réseau pour découvrir les nouveaux services. Un terminal peut accepter ou rejeter un service. Lorsque le terminal accepte un service, ce dernier est rajouté à son Service Registry et mis à la disposition des applications locales (application ou service). Le Service Registry gère les services locaux et distants de la même manière, assurant ainsi une transparence de localisation des services.
   Les nouveaux services découverts ne sont pas téléchargés immédiatement. Le téléchargement est différé jusqu'à l'invocation du service par une application locale. Les services découverts qui sont enregistrés et qui n'ont pas été utilisés, ne seront jamais téléchargés sur le terminal.

4. *Découverte de services*
   Pour utiliser un service, le client demande à son Service Regsitry de réaliser une recherche de services. Le client doit au moins fournir le nom de l'interface du service. Fournir le nom de l'implémentation du service est optionnel. Le Service Registry retourne la référence de l'objet implémentant l'interface demandée. Ainsi, le client peut accéder au service via les méthodes de l'interface du service.
   Conceptuellement, le Service Registry ne fait pas de distinction entre les services locaux et les services distants. Dans le cas de services distants, l'implémentation du service fournit un proxy qui assure la communication entre le client et le service distant en utilisant le protocole RMI.

5. *Gestion de la dynamicité*
   A chaque description de service, le Service Registry associe une période de validité. Une fois cette période expirée, le service est supprimé du Service Registry. Les périodes de validité peuvent être renouvelées avant l'expiration du leasing.

5.6.8 Universal Description Discovery and Integration (UDDI)

UDDI [UDD] est une spécification qui s'articule autour des standards HTTP, SOAP [BEK+00] et XML. Elle permet aux entreprises de publier, de découvrir et d'accéder à des informations sur les fournisseurs et sur les Web Services [UDD] qu'ils proposent.

1. *Description des services*

La publication d'un service dans UDDI requiert la création d'un fichier XMl qui décrit le Web Service suivant les spécifications UDDI [CCMW01]. Les principaux éléments de ce fichier sont :

- *businessEntity* : chaque service appartient à une organisation identifiée par un nom, une description, des personnes de contact, une ou plusieurs catégories, etc.
- *businessService* : inclut les informations non techniques relatives au service comme son nom et sa description.
- *bindingTemplate* : définit où (URL) et comment (protocole) accéder au service.
- *tModel* : comprend des liens vers les informations techniques du service, comme par exemple un fichier WSDL (Web Service Description Language) décrivant les différentes méthodes disponibles pour l'utilisation du service.

2. *Centralisation des descriptions*

   La spécification UDDI adopte une approche basée sur un annuaire distribué de descriptions de Web Services, comportant une interface programmable permettant de publier ou de chercher de manière logicielle des Web Services. L'annuaire UDDI est composé d'un ensemble de *nœuds* distribués dans le réseau.
   Les nœuds fonctionnent à la manière de DNS. Ils répliquent leurs informations dans un délai de 24 heures en s'authentifiant par un mécanisme de certificats et en échangeant ensuite les données de manière sécurisée.

3. *Déclaration de services*

   L'API UDDI est divisée en une interface de programmation pour l'enregistrement de Web Services dans l'annuaire UDDI et une interface de programmation pour la recherche d'informations dans l'annuaire. L'API UDDI est concrétement constituée de deux grandes bibliothèques d'appels : l'API de requête et l'API de publication.
   En utilisant l'API UDDI de publication, l'utilisateur peut stocker les descriptions des services qu'ils propose dans un nœud. Elles sont ensuite répliquées de nœud en nœud par une technique appelée *UDDI Cloud Services* [CDK+02]. Le Web Service peut alors être connu par tous ceux qui le recherchent.

4. *Découverte de services*

   La recherche des Web Services dans l'annuaire se fait par des requêtes XML (définies dans le schéma fourni par UDDI) avec une couche SOAP pour le transport. Ces requêtes peuvent être faites via le web ou de manière logicielle.
   La recherche de services est encore limitée. En effet, même dans la dernière spécification en date, une recherche ne peut se faire que sur le nom d'un service, et non pas sur sa description par exemple. UDDI offre cependant la possibilité de catégoriser un service, ce qui permet d'effectuer un premier tri.

5. *Gestion de la dynamicité*

   Le protocole UDDI supporte le mécanisme de leasing qui permet de préciser la durée de disponibilité d'une description de service dans l'annuaire UDDI.

5.7 Comparaison entre les différents protocoles

Les protocoles de découverte de services présentés dans ce chapitre sont proposés pour faciliter la coopération dynamique entre des services et des dispositifs distribués dans le réseau avec le minimum possible de configuration humaine.
Pratiquement, tous les protocoles cités précédemment fournissent des mécanismes similaires. A la base, des clients doivent retrouver des descriptions de services pertinents, comprenant des informations suffisantes pour établir le contact avec les services. Il existe deux mécanismes de base : l'annonce de services et la recherche de services.
Les services annoncent leur disponibilité, leur adresse et d'autres informations nécessaires. Les clients qui reçoivent ces annonces, peuvent contacter directement les services qu'ils veulent ou envoyer des requêtes de recherche de services. Les services ou d'autres éléments reçoivent ces requêtes et répondent en conséquence. Chaque protocole implémente l'un ou les deux concepts même si les détails d'implémentation diffèrent considérablement d'un protocole à l'autre. Le tableau 5.1 montre une comparaison entre ces protocoles et révèle les avantages et les inconvénients de chaque protocole. Dans ce qui suit, nous détaillons cette comparaison.

5.7.1 Descriptions de services

La description du service est le moyen qui permet au serveur de décrire le service qu'il propose ainsi que ses caractéristiques. Grâce à cette description, le client peut être renseigné sur le service et donc décider de l'utiliser ou pas. Plus la description du service est riche, plus le client est bien informé.
La description du service est une composante importante du protocole de découverte de services. L'expression d'une requête, la recherche d'une réponse ainsi que l'évaluation de la requête sont toutes basées sur la description du service.

Les protocoles étudiés dans ce chapitre offrent différents types de description de services. UPnP, UDDI et SSDS utilisent le langage XML. SDP, Salutation et MOCA proposent des formats qui leurs sont propres. Jini utilise des objets Java (proxy) et SLP décrit les services sous forme de texte.
Mis à part ceux qui sont basés sur XML, les protocoles de découverte de services maintiennent une interopérabilité très stricte entre les développeurs de services. Ces derniers doivent utiliser le schéma de description de services imposé par le protocole. Ce schéma ne leurs permet pas de décrire de manière détaillée les différentes fonctionnalités du service à proposer.

Le choix du formalisme de description de services est très important en particulier pour les environnements mobiles qui sont caractérisés par la mobilité, la dynamicité et l'hétérogénéité des différentes entités. Dans de tels environnements, les clients et les services doivent partager des sémantiques communes pour pouvoir établir la communication entre eux et négocier les interfaces et les paramètres du service. De plus, les descriptions de services doivent être standardisées pour assurer l'interopérabilité entre les différentes entités et pour permettre aux clients de formuler des requêtes sur ces descriptions.

	description des services	répertoire central	déclaration des services	découverte des services	requêtes de recherche de services	gestion de la dynamicité
SLP	service URL + service template	DA (optionnel)	enregistrement auprès du DA ou annonce en multicast	requête au DA ou découverte en multicast	requêtes expressives, opérateurs <,>,=,...	leasing
Jini	proxy (interface + attributs)	Lookup Service (LS)	enregistrement auprès du LS	requêtes pour le LS	attributs = valeurs	leasing + notification
UPnP (SSDP)	document XML	Non	annonce en multicast	écoute d'annonces ou découverte en multicast	recherche par nom, par type ou par adresse	leasing + notification
UDDI	document XML	UDDI registry (nœuds distribués)	enregistrement dans un nœud	requêtes XML pour l'UDDI registry	recherche basée sur le nom	leasing
Salutation	functional unit description record	SLM	enregistrement auprès du SLM	requêtes au SLM	requêtes expressives, opérateurs prédéfinis	notification
SSDS	document XML	SDS (optionnel)	annonce unicast (SDS) ou multicast (réseau)	écoute d'annonces ou découverte en multicast	comparaison balises des XML	leasing
Bluetooth (SDP)	service record	Non	Non	API spéciale	par attributs, par classe ou par navigation	Non
MOCA	description de services	non	annonce en multicast	requête au service registry local	basée sur le nom de l'interface	leasing

TAB. 5.1 – Comparaison des principaux protocoles de découverte de services

XML est la technologie de choix parce qu'elle permet d'exprimer toutes les fonctionnalités des services. Elle rend les descriptions de services fortement extensibles et interopérables. De plus, gâce à XML, le client peut utiliser le service même s'il le rencontre pour la première fois.

5.7.2 Découverte de services

Les répertoires centraux sont utilisés pour centraliser les descriptions des services afin de faciliter la découverte de services. Dans le cas de l'existence d'un répertoire de services, un serveur peut publier son service en enregistrant sa description auprès du répertoir et un client peut consulter ce répertoire pour trouver un service de son choix (c'est le cas de SLP, Jini, UDDI, Salutation et SSDS). Si un tel répertoire n'existe pas, alors les serveurs doivent diffuser (annoncer) des descriptions afin d'informer les clients de la présence de leurs services (comme MOCA), ou bien les clients doivent diffuser leurs requêtes de découverte et de recherche de services, ou bien les deux mécanismes à la fois (comme SLP, UPnP et SSDS).

Les protocoles qui offrent une solution de découverte de services avec un répertoire central semblent plus économes pour les environnements mobiles. Au lieu d'envoyer un message en multicast dans le réseau pour localiser un service, un client peut envoyer un message unicast pour découvrir les services. Les découvertes de services en multicast ou en broadcast peuvent causer un excès de messages dans le réseau. De plus, elles peuvent provoquer un grand nombre de messages et donc une grande consommation de ressources de la part des terminaux qui doivent traiter tous ces messages.

5.7.3 Gestion et évaluation des requêtes

Lorsque le client veut découvrir un service, il exprime son besoin sous forme d'une requête. La requête est soit envoyée au répertoire central ou diffusée dans le réseau. Elle est évaluée par le répertoire, par le client ou par le fournisseur du service.
La requête précise le genre de services que le client recherche ainsi que les valeurs de leurs attributs. Elle doit contenir des opérateurs booléens et des opérateurs logiques pour être assez expressive et donc permettre au client de trouver les services pertinents en une seule requête. En effet, le protocole doit éviter de présenter au client les descriptions de tous les services diponibles et de les interroger une à une pour trouver celle qu'il peut utiliser.

SLP et Salutation permettent de faire des requêtes plus sophistiquées et plus expressives que celles des autres protocoles. UPnP et MOCA ne permettent pas du tout des requêtes basées sur des attributs du service. Les autres protocoles proposent plusieurs manières de formuler la requête mais ils restent assez limités en terme de richesse d'expression de la requête.

5.7.4 Gestion de la dynamicité

Les environnements mobiles sont des environnements très dynamiques. Les unités mobiles peuvent entrer dans le réseau instantanément et se déconnecter fréquemment. En conséquence, la disponibilité des entités et des services qu'elles proposent ne peut pas être toujours assurée. Les mécanismes de leasing et de notification ont été proposés pour gérer une telle dynamicité.

Tous les protocoles de découverte de services présentés précédemment, sauf Salutation et Bluetooth, implémentent le mécanisme de leasing qui permet de détecter l'indisponibilité des services.
Jini, UPnP et Salutation implémentent le mécanisme de notification qui permet aux systèmes dynamiques d'informer les clients des changements qui se produisent sur les services (arrivée, départ et mise à jour du service).

5.8 Limites des systèmes existants pour les environnements mobiles

Nous pensons que même si les protocoles de découverte de services présentés dans les sections précédentes, sont bien utilisés dans les systèmes distribués, ils possèdent des limitations qui les rendent non exploitables dans les environnements mobiles. Dans ce qui suit, nous présentons les principales limitations qui nous intéressent.

5.8.1 Non exploitation du contexte

Bien qu'elles constituent les deux principales caractéristiques des environnements mobiles, la mobilité des dispositifs et l'hétérogénéité de leurs capacités ne sont pas prises en considération par ces protocoles conçus initialement pour opérer dans des environnements fixes.
Un autre problème majeur de ces protocoles est qu'ils n'exploitent pas les informations concernant le contexte de découverte de services telles les préférences de l'usager et les capacités des ressources disponibles (réseau et dispositif). De plus, il est quasiment impossible de retrouver un service qui possède un attribut de valeur pouvant changer selon le contenu dynamique de l'environnement comme la bande passante du réseau par exemple.

5.8.2 Manque de description détaillées des services

Les services proposés dans un système de découverte et de recherche de services sont représentés par des descriptions composées d'un ensemble d'attributs. Chaque attribut décrit une fonctionnalité que le service peut offrir. Ces descriptions sont utilisées par les clients pour découvrir les services et sélectionner ceux qui répondent au mieux à leurs besoins.
Si les descriptions des services sont mal faites, les fonctionnalités et les capacités des services seront mal représentées. Par conséquent, les clients seront mal informés sur les services

proposés et donc auront du mal à retrouver les services qui répondent au mieux à leurs besoins.
Les protocoles de découverte de services existants n'utilisent pas des langages expressifs de description de services qui permettent de représenter convenablement les différentes fonctionnalités d'un service. Ils manquent également d'outils de filtrage de services qui permettent de filtrer les services selon leurs fonctionnalités. En effet, ils se contentent de retrouver les services disponibles dans le réseau mais ils ne vérifient pas si ces services répondent correctement aux besoins de la requête ou pas.

5.8.3 Absence de formalismes communs de description des services

Pour permettre à un service proposé par un dispositif d'interagir avec les autres composants du système (autres services et clients), la description de ses capacités et ses fonctionnalités doit être comprise par tous les composants du système. Dans d'autres termes, un formalisme commun bien défini (ou une ontologie) doit être adopté par tous ces composants avant qu'un processus de découverte ou de recherche de services n'ait lieu.
Les infrastructures de découverte de services existantes n'utilisent pas de formalismes ou de standards pour la description de leurs services sauf UPnP et SSDS qui sont basés sur le langage XML. Cependant, l'utilisation de XML dans ces deux protocoles ne joue pas un rôle important dans le processus de découverte de services car les descriptions des services ne sont pas basées sur des formalismes bien définis.

5.9 Conclusion

Les problèmes de découverte et de recherche de services ont attiré l'attention de plusieurs organismes de recherche et d'industrie. Les protocoles présentés dans ce chapitre en témoignent.

Bien que ces protocoles semblent fournir une bonne solution aux problèmes de découverte et de recherche de services, ils n'adressent pas un des besoins les plus importants de la fourniture de services dans les environnements mobiles. Il s'agit de la sensibilité au contexte.
Les protocoles étudiés dans ce chapitre n'utilisent pas les informations contextuelles sur l'environnement de découverte de services. Ils ne peuvent donc pas proposer aux clients les services pertinents les mieux adaptés à leurs besoins et aux besoins de leur environnement. De plus, ces protocoles ne se basent pas sur des formalismes de description de services. Ils n'assurent pas ainsi l'interopérabilité entre tous les fournisseurs de services et les clients qui veulent utiliser les services proposés.

Dans le chapitre suivant, nous présentons l'approche de découverte de services proposée dans CASP, dans laquelle nous fournissons un support pour la prise en compte du contexte.

Chapitre 6

Découverte de services sensibles au contexte

Les protocoles présentés dans le chapitre précédent, fournissent une variété de solutions au problème de recherche et de découverte de services. Néanmoins, ils n'adressent pas le besoin de découverte de services sensibles au contexte. Pour remédier à cet inconvénient, nous avons développé une nouvelle approche pour la découverte et la recherche de services dans laquelle, nous fournissons un support pour la personnalisation des services, la sensibilité au contexte et la gestion de la mobilité.

Dans l'approche que nous proposons, la sensibilité au contexte est réalisée via les paramètres suivants : les capacités du terminal, les préférences de l'utilisateur et la localisation du terminal. Cette solution permet aux services personnalisés les plus appropriés, d'être découverts par des utilisateurs mobiles, quel que soit le réseau utilisé (principal ou étranger) et quel que soit le terminal utilisé (ordinateur de bureau ou terminal de capacités limitées).

Les principaux apports de la plate-frome CASP sont :

1. CASP permet la découverte et la recherche de services (pas de découverte spontannée). Par rapport à la classification des systèmes de découverte de services présentée dans la section 5.2 (page 72), cette approche est préconfigurée, à médiateur et non-transparente.
2. Elle intègre la découverte, la recherche et la sélection de services pour les environnements mobiles.
3. Elle permet la personnalisation de services et la sensibilité au contexte.
4. Elle permet la mobilité personnelle et la mobilité du terminal (nomadicité) et implémente le concept VHE.
5. CASP est une approche conforme aux standards MExE et CC/PP. Elle est basée sur l'utilisation des profils écrits en XML (profil terminal, profil utilisateur et profil service).

Dans ce chapitre, nous montrons comment la plate-forme CASP assure les fonctionnalités de base que doit garantir un protocole de découverte de services (voir section 5.3 page 74). Nous verrons également, comment la sensibilité au contexte et la nomadicité sont utilisées dans le processus de découverte et de recherche de services.

6.1 Découverte de services sensibles au contexte

Nous présentons dans cette section le système de recherche et de découverte de services utilisé dans CASP. Nous montrons en particulier, comment ce système assure la description des services, la gestion de l'annuaire de services, l'annonce des nouveaux services dans CASP, la découverte de services sensibles au contexte et la gestion de la dynamicité des services.
Les spécifications détaillées de l'ensemble des messages relatifs à la fourniture et la découverte de services dans CASP sont présentées dans l'annexe B.

6.1.1 Description des services

Chaque service proposé par un FS est décrit par le *Profil Service* que nous avons présenté dans la partie 4.5.2.3 du chapitre 4 (page 64). Ce profil est fourni par le FS, géré par le MS et stocké dans la BD_Services. Le profil service contient tous les attributs et les caractéristiques qui peuvent identifier le service et le décrire suffisamment pour être découvert par les utilisateurs.
Le service dans CASP peut avoir plusieurs versions (*proxy*). Celles-ci diffèrent entre elles par exemple, par le tarif de la version proposée ou par les capacités du terminal nécessaires à l'exécution de cette version sur le terminal.
Dans le profil service, certains attributs sont communs pour toutes les versions du service (le nom du fournisseur du service, par exemple). D'autres peuvent avoir différentes valeurs pour chaque version (le coût de la version du service, sa qualité, son langage, etc.).

Le tableau 4.2 du chapitre 4 (page 65) décrit les attributs d'un profil service dans CASP. Il est composé des quatre parties suivantes :

- La partie **VASGEN** contient des informations générales sur le service.
- La partie **VASP** contient des informations sur le fournisseur du service.
- La partie **VASDETAIL** décrit les paramètres valides pour toutes les versions (proxy) d'un même service.
- La partie **SECURITY** est dédiée aux paramètres de sécurité.
- La partie **PROXIES** contient des paramètres spécifiques à chaque version. Les attributs suivants sont définis pour chaque version : les capacités du terminal nécessaires à l'exécution de cette version sur le terminal, le coût de la version et enfin les caractéristiques et les protocoles de communication à utiliser pour télécharger et exécuter le service sur le terminal.

Le choix du langage de description des services s'est porté sur le langage XML pour les raisons suivantes :

- XML permet de valider le contenu d'une description de service par rapport à un schéma de description de services (grammaire) bien défini.
- Il assure l'interopérabilité et facilite l'échange des descriptions de services entre le MS et les fournisseurs de services.
- Il prend en charge une fonction de recherche très avancée car la structure et le sens des descriptions sont bien définis dans une grammaire. Ainsi, la recherche par nom de balises, attributs de balises, contenu des données et emplacement dans la description, sont des critères de recherche dont les documents XML facilitent l'implémentation.

Un exemple XML du profil service est présenté dans la figure 4.5 du chapitre 4 (page 66).

6.1.2 Utilisation d'annuaire

La plate-forme CASP utilise un annuaire pour gérer toutes les descriptions de services disponibles et toutes les informations d'identification des utilisateurs abonnés.
Chaque annuaire est associé à un opérateur réseau. Il gère deux types de bases de données : une base de données des utilisateurs et une base de données des descriptions de services qu'offre cet annuaire.
L'annuaire dans CASP peut être conçu sous forme d'un seul MS ou d'une fédération de MSs.

6.1.2.1 Répartition de l'annuaire

Comme le montre la figure 6.1, l'annuaire de services peut être implémenté sous forme de site central unique (un seul référentiel) composé d'un seul MS et de deux bases de données (BD_Utilisateurs et BD_Services). Néanmoins, faire confiance à un annuaire physiquement centralisé n'est pas souhaitable dans le cadre de la fourniture de services, car un accès unique et centralisé peut constituer un goulot d'étranglement surtout, si le nombre d'utilisateurs est très grand. De plus, le système peut être vulnérable (si une panne survient dans l'annuaire centralisé, l'accès à l'ensemble des services deviendrait impossible).

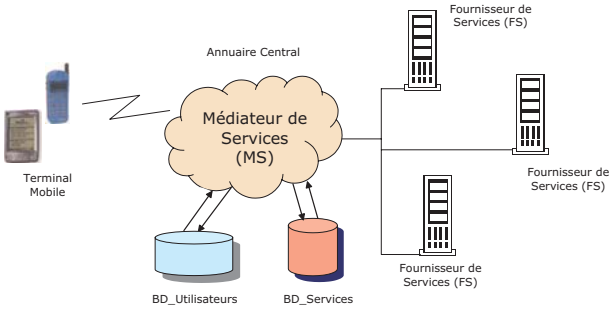


FIG. 6.1 – Annuaire central dans CASP

Implémenter l'annuaire sous forme de répertoire réparti sur plusieurs référentiels hébergés par des sites différents est sans doute très intéressant en matière de disponibilité de

données. De plus, la répartition de l'annuaire permet de prendre en compte la répartition géographique des données.
L'annuaire réparti est constitué d'un ensemble de MSs interconnectés par un réseau de communication comme le montre la figure 6.2. Chaque MS gère les abonnés d'une même zone géographique par exemple. Dans ce cas, le système doit fournir aux utilisateurs mobiles des services personnalisés requis, indépendamment de leur point d'accès à l'annuaire (à partir de n'importe quel MS associé à l'annuaire). Pour ce faire, le système doit prévoir des mécanismes de répartition et d'extraction des informations des deux bases de données concernant les utilisateurs et les services associés à ce même annuaire. La répartition des bases de données peut être gérée de deux manières : la **réplication** et le **partitionnement** [GG96].

La réplication permet de gérer des copies de l'information détenue par l'annuaire sur un ou plusieurs MSs. Cette technique peut s'appliquer sur la base de données des services afin d'augmenter les performances et d'assurer une meilleure disponibilité des descriptions de services. Ainsi, la même BD_Services est mémorisée sur chaque MS (voir la figure 6.2).

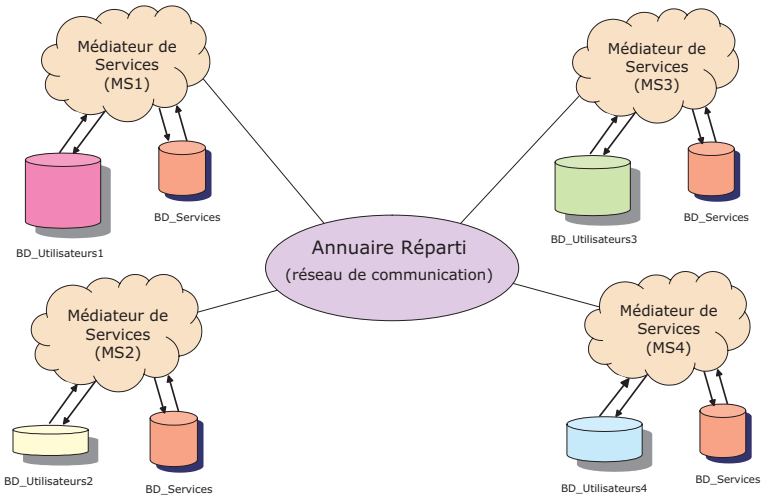


FIG. 6.2 – Annuaire réparti dans CASP

Le partitionnement permet de répartir l'information sur plusieurs sites, chacun contenant une partie de la base de données. Cette technique peut s'appliquer sur la base de données globale des utilisateurs associés à un même annuaire. Ainsi, chaque MS gère dans une BD_Utilisateurs locale, une partie des utilisateurs abonnés. La base de données globale des utilisateurs associés à l'annuaire, est l'union de toutes les BD_Utilisateurs locales et disjointes. Les avantages du partitionnement sont alors évidents : l'organisation répartie combine l'efficacité de l'évaluation (les données sont mémorisées à proximité du MS où elles sont le plus souvent utilisées) avec des capacités d'accès améliorées (il est possible d'accéder à des données sur un MS, depuis un autre MS via le réseau de communication).

6.1.2.2 Gestion de la nomadicité

La plate-forme CASP peut assurer la fourniture de services même dans le cas de nomadicité. Pour ce faire, deux types de MS ont été définis :

1. *Médiateur de Services Principal (MS Principal) :* il s'agit d'un des MSs associés à l'annuaire avec lequel l'utilisateur a une souscription pour un ou plusieurs services. Le MS Principal peut récupérer, à partir de la BD_Utilisateurs, les informations sur l'abonnement de l'utilisateur ainsi que ses préférences pour assurer la personnalisation des services.
2. *Médiateur de Services Visité (MS Visité) :* il s'agit de n'importe quel MS, différent du MS Principal, qui peut fournir des services à l'utilisateur mobile dans un réseau étranger (géré par un autre annuaire). Le MS Visité récupère les profils et les préférences de l'utilisateur de son MS Principal afin de personnaliser les services qui sont offerts à l'utilisateur.

6.1.3 Déploiement et annonce des services

Le processus de déploiement et d'annonce de services englobe l'ensemble des messages nécessaires pour l'introduction et la suppression des services de la plate-forme CASP. Les composants impliqués dans ce processus sont le MS et les FSs. Ils interagissent pour l'accomplissement des tâches suivantes :

6.1.3.1 Enregistrement des services

Les nouveaux services sont enregistrés par leur FS auprès du MS. Les informations sur le nouveau service (profil service) sont stockées dans la BD_Services. Une fois l'enregistrement terminé, le service est mis à la disposition des utilisateurs, c.-à-d., il peut apparaître dans la liste des services disponibles que les utilisateurs obtiennent, suite à une requête de découverte et de recherche de services.

6.1.3.2 Désenregistrement des services

Chaque FS peut désenregistrer son service en supprimant de la BD_Services les informations qui lui sont associées. Après son désenregistrement, le service devient inaccessible pour les utilisateurs. Le MS envoie alors, un message de notification à tous les terminaux en cours d'accès au service pour informer les utilisateurs de la non disponibilité du service.

6.1.3.3 Mise à jour des services

Il s'agit de mettre à jour les descriptions de services dans la BD_Services. Si l'opération comporte une modification du tarif du service ou de sa localisation dans le réseau (adresse IP par exemple), le MS doit envoyer une notification aux utilisateurs mobiles qui utilisent ce service.

6.1.4 Découverte de services sensibles au contexte

Grâce à une interface graphique personnalisable que propose la plate-forme CASP sur le terminal, les utilisateurs mobiles peuvent découvrir et sélectionner des services enregistrés dans la BD_Services. Cette interface offre aux utilisateurs les différentes possibilités (types) de découverte de services suivantes :

- Découverte de tous les services disponibles dans le réseau : formulation du Menu *LS (Lookup Services)*.
- Recherche des services favoris de l'utilisateur : formulation du *Menu des Services Favoris*. A chaque utilisation d'un service sur le terminal, la plate-forme CASP propose à l'utilisateur la possibilité de le rajouter à une liste de services favoris associée à ce même utilisateur. L'utilisateur peut à tout moment récupérer cette liste sur son terminal pour pouvoir demander le téléchargement de ses services favoris.
- Découverte de services basée sur des mots clés introduits par l'utilisateur.

6.1.4.1 Gestion de la sensibilité au contexte

Le contexte que nous utilisons dans le cadre de la découverte de services est représenté par les paramètres suivants : les capacités du terminal, les préférences de l'utilisateur et la localisation du terminal. Le MS utilise ces informations pour filtrer la BD_Services et pour ne proposer à l'utilisateur que les services personnalisés qui répondent à ses besoins et qui peuvent s'exécuter sur son terminal.

La personnalisation permet de fournir des services adaptés aux préférences de l'utilisateur. Elle est réalisée grâce à l'utilisation du Profil Utilisateur, dont une partie est stockée dans la BD_Utilisateurs (au niveau du MS). L'utilisateur peut ainsi, accéder à des services personnalisés à partir de n'importe quel terminal (fixe ou mobile, limité ou puissant). En permettant aux utilisateurs d'utiliser des services personnalisés independamment du terminal utilisé, CASP assure la **mobilité personnelle** et réalise ainsi le concept **VHE**.

Les informations sur les capacités du terminal et sa localisation sont envoyées au MS avec chaque requête de découverte de services. La localisation du terminal, introduite par l'utilisateur, permet au MS de proposer à l'utilisateur des services qui sont basés sur ce type d'informations (service météo par exemple). Les capacités du terminal, envoyées sous forme de Profil Terminal, permettent au MS de ne proposer à l'utilisateur que des services qui peuvent s'exécuter sur son terminal.

6.1.4.2 Déroulement d'une session de découverte de services

Lorsqu'un utilisateur mobile entre dans le réseau, une procédure d'enregistrement ou d'authentification de l'utilisateur est lancée entre le TM et MS afin de permettre à l'utilisateur d'accéder à la plate-forme CASP en toute sécurité.

Le processus de découverte de services commence par le téléchargement sur le terminal, d'une interface graphique adaptée aux capacités du terminal. Il s'agit du Proxy Lookup Service (Proxy LS).

Le Proxy LS permet aux utilisateurs mobiles de choisir un des trois types de la découverte de services proposée par CASP : la formulation du Menu LS, la recherche de services à base de mots-clés, la formulation du "Menu des Services Favoris" ou la mise à jour de ce dernier.
Dans la requête de récupération d'un Proxy LS, le terminal envoie au MS ses capacités sous forme de profil terminal et la localisation de son terminal introduite par l'utilisateur.

A partir du type de la découverte de services, des capacités du terminal et de sa localisation (déjà récupérées de la requête du Proxy LS) et des préférences de l'utilisateur (Profil Utilisateur), le MS filtre la BD_Services en utilisant un parser XML, il formule ensuite la liste des services sous forme d'un fichier XML et l'envoie enfin à l'utilisateur.
Le contenu de cette liste est composé principalement des descriptions de services qui correspondent au contexte de la découverte de services.
L'utilisateur choisit un service et envoie sa sélection au MS. Ce dernier répond en envoyant une liste des différentes versions disponibles du service sélectionné qui répondent aux caractéristiques du contexte : capacités du terminal, préférences de l'utilisateur et localisation du terminal. L'utilisateur peut ainsi choisir la version la mieux adaptée à ses besoins et demander son téléchargement sur le terminal.

A chaque service et version de service proposés à l'utilisateur, le MS associe une description et des informations indicatives sur le tarif, afin de permettre à l'utilisateur de choisir la version du service la mieux adaptée à ses besoins.

6.1.4.3 Interactions associées à la découverte de services

1. **Fourniture du Menu LS**
   Le schéma 6.3 décrit l'ensemble des opérations à effectuer par le MS, suite à une requête de fourniture du Menu LS.
   Le Menu LS demandé est une liste de descriptions de services disponibles dans la BD_Services qui correspondent à la combinaison : capacités du terminal, préférences de l'utilisateur et localisation du terminal.
2. **Fourniture du "Menu des Services Favoris"**
   Les services favoris ne sont pas liés à un terminal particulier, car l'utilisateur peut y accéder quel que soit le terminal utilisé (dans la limite de ses capacités).
   Le "Menu des Services Favoris" fourni à l'utilisateur, est une liste de services favoris qui répond à la combinaison : capacités du terminal, préférences de l'utilisateur et localisation du terminal.
   Le procédure suivie pour la formulation du "Menu des Services Favoris" est similaire à celle utilisée pour la formulation du LS, à qui il faut rajouter un filtrage final du contenu du Menu LS en se basant sur la liste des services favoris de l'utilisateur.
   La figure 6.4 montre l'ensemble des opérations à effectuer par le MS pour répondre à la requête du "Menu de Services Favoris".
3. **Sélection du service**

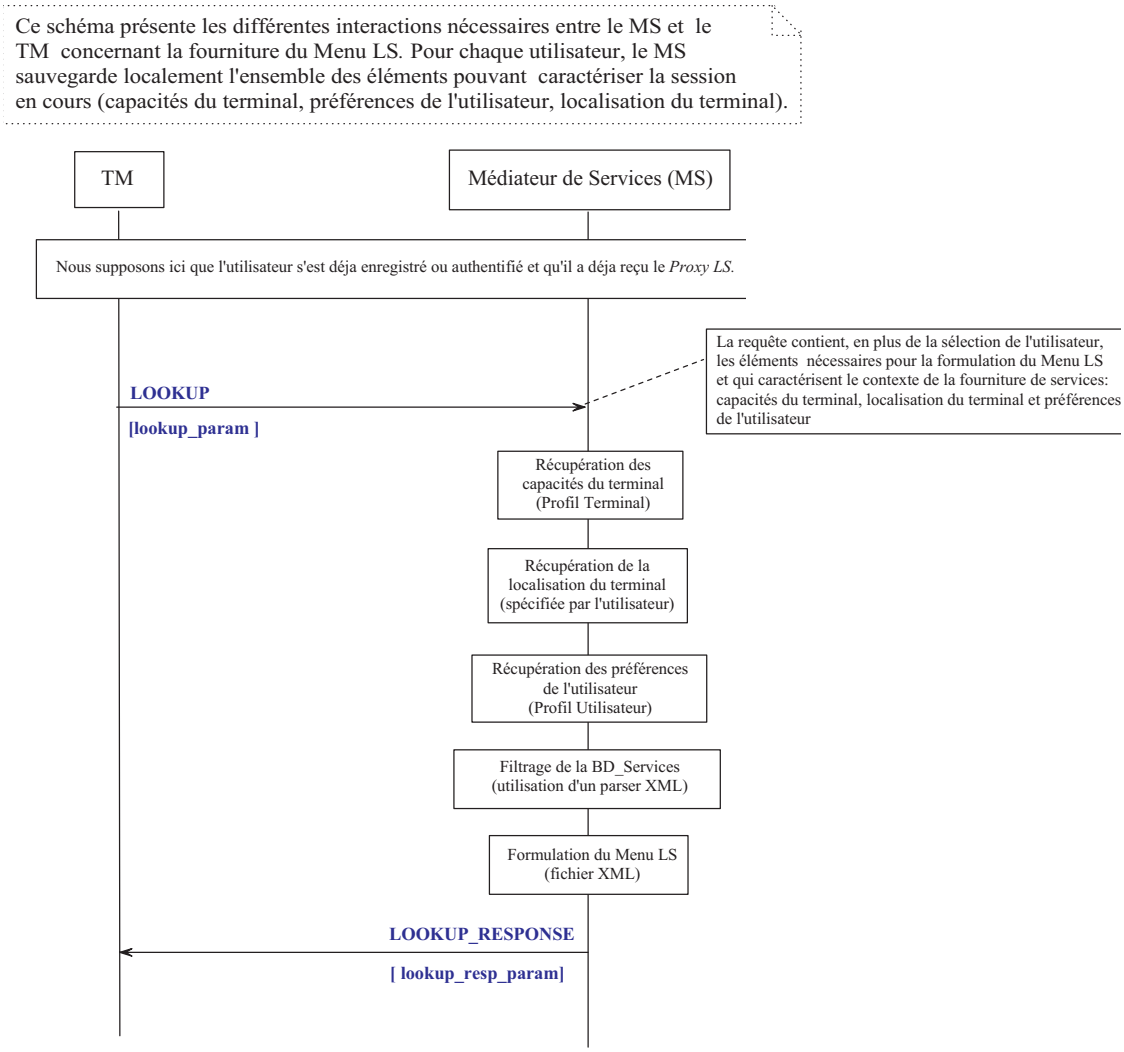


FIG. 6.3 – Interactions nécessaires pour la formulation du Menu LS

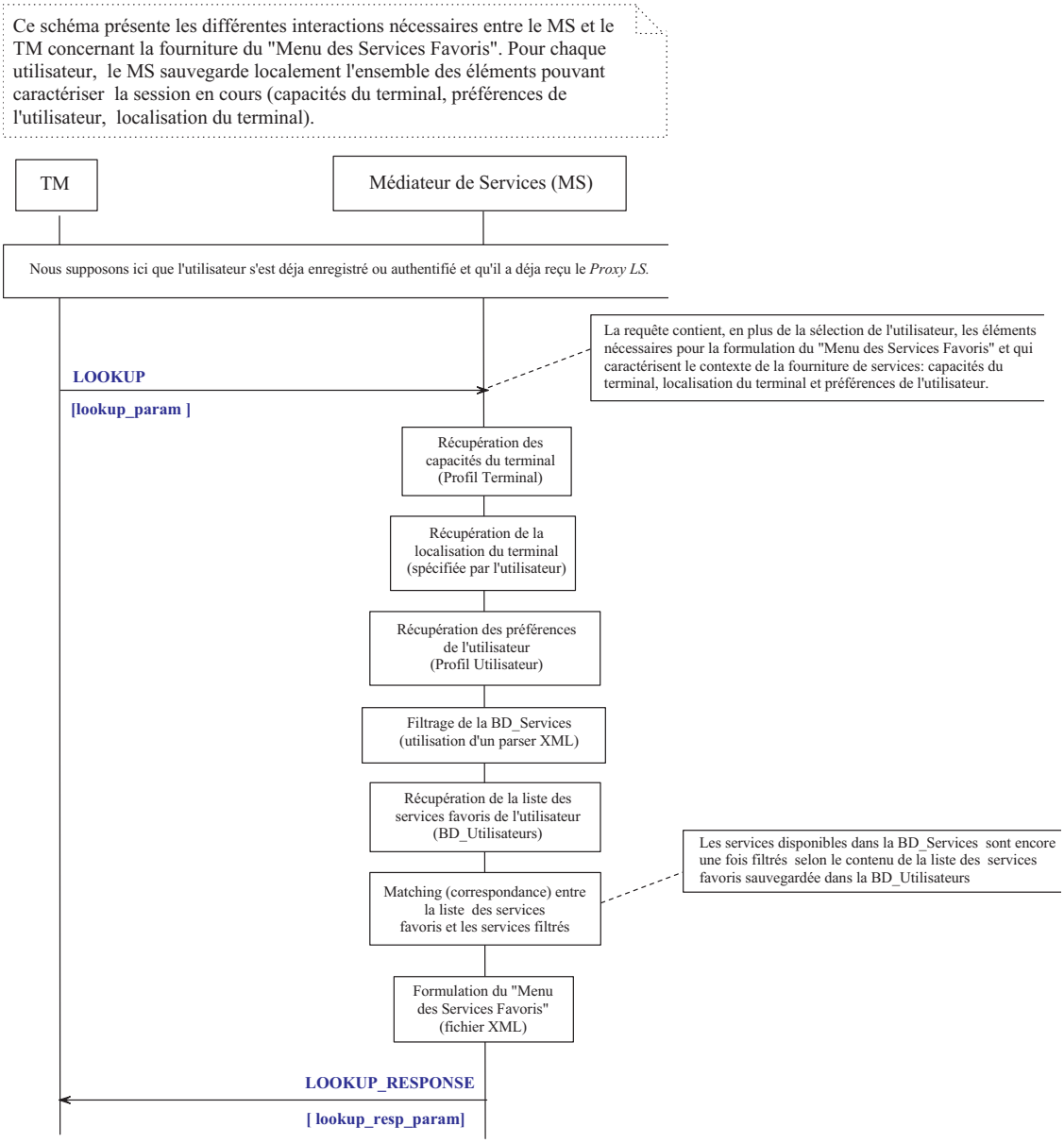


FIG. 6.4 – Interactions nécessaires pour la formulation du “Menu des Services Favoris”

Après la phase de découverte de services et des versions de services, l'utilisateur peut sélectionner la version qu'il souhaite utiliser. Le MS lui fournit alors l'adresse URL de cette version que le terminal utilisera par la suite, pour initialiser le téléchargement du service sur le terminal à partir du FS.

Les spécifications des différentes interactions entre les composants de CASP (TM, MS et FSs) ainsi que les paramètres qui leurs sont associés, sont détaillés dans l'annexe B.

6.1.5 Gestion de la dynamicité

CASP n'implémente pas le mécanisme de leasing. Une fois enregistré, le service reste disponible dans la BD_Services jusqu'à ce que son FS demande sa suppression.
CASP en revanche, utilise le mécasnisme de notification pour notifier les modifications d'un service aux utilisateurs qui l'utilisent.

6.2 Conclusion

Dans ce chapitre, nous avons présenté en détail, le système de découverte de services sensibles au contexte utilisé dans CASP.
La contribution principale de ce système, par rapport aux solutions existantes, est sa capacité à assurer une découverte et une recherche de services personnalisés et sensibles au contexte, notamment dans la cas de la nomadicité.
CASP permet à la fois, la découverte, la recherche et la sélection des services selon les capacités du terminal, les préférences de l'utilisateur et la localisation du terminal. Grâce à l'utilisation des profils basés sur XML et à la conformité aux standards MExE et CC/PP, CASP propose une description riche des services et une recherche rapide et efficace des services selon plusieurs types de découvertes de services.

Dans le chapitre suivant, nous présentons en détail, les différents composants de CASP et nous verrons comment cette plate-forme a été implémentée et testée dans le cadre du projet Européen MOBIVAS.

Chapitre 7

Conception et mise en œuvre de la plate-forme CASP

Dans ce chapitre, nous présentons l'environnement de développement de la plate-forme CASP et nous proposons des scénarios qui illustrent le fait que les services proposés par CASP aux utilisateurs mobiles, diffèrent d'un contexte à un autre. Afin de démontrer l'aptitude de CASP à prendre en compte cet aspect, nous avons utilisé deux versions différentes de CASP : une, basée sur Java et destinée aux terminaux puissants et, une autre, basée sur MIDP et destinée aux terminaux de capacités limitées.
Deux services, développés dans la cadre du projet MOBIVAS, ont également été utilisés : un service lecteur multimédia et un service de localisation des utilisateurs proches du terminal [RQF01b, RQF01a]. Deux versions différentes ont été utilisées pour chaque service : une basée sur Java pour les terminaux puissants, et une autre basée sur MIDP pour les terminaux de faibles capacités.

Dans un premier temps, nous décrivons en détail les différents composants de la plate-forme CASP ainsi que le prototype utilisé pour la valider. Nous présentons ensuite, l'ensemble des scénarios utilisés pour démontrer l'abilité de la plate-forme CASP à supporter la découverte et la fourniture de services sensibles au contexte. Nous terminons ce chapitre par une évaluation des performances de la plate-forme CASP.

7.1 Les composants de la plate-forme CASP

Rappelons que la plate-forme CASP est composée des trois éléments suivants :

- La composante EUT (*End User Terminal*) installée dans le terminal
- Le MS (Médiateur de Services) installé dans la partie fixe du réseau
- La composante FS installée chez le fournisseur de services

La figure 7.1 est une représentation globale de la plate-forme CASP. Dans ce qui suit, nous présentons en détail, la conception de chacun de ces composants ainsi que les différents modules qui les composent.

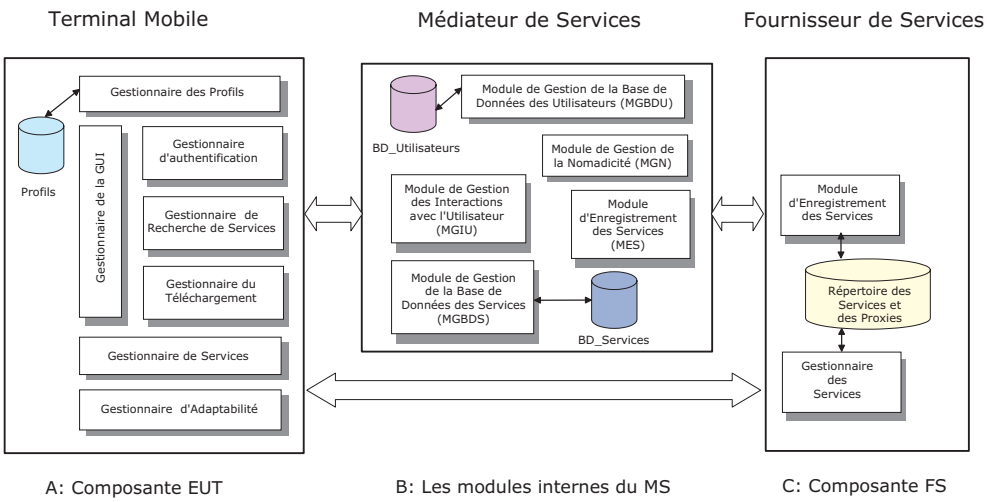


FIG. 7.1 – Les composants de la plate-forme CASP

7.1.1 Médiateur de Services

Le Médiateur de Services (MS) est l'élément central de la plate-forme CASP. C'est le coordinateur des différents composants de la plate-forme.

Le MS est responsable des fonctionnalités suivantes :

- Enregistrement et désenregistrement des fournisseurs de services.
- Authentification des fournisseurs de services.
- Gestion des descriptions de services (enregistrement, suppression et mise à jour).
- Authentification et enregistrement des utilisateurs.
- Gestion des profils des utilisateurs.
- Classification des terminaux mobiles et négociation de leurs capacités.
- Découverte et fourniture de services sensibles au contexte.

La figure 7.2 décrit les principaux modules internes du MS.

1. *Module de Gestion des Interactions avec l'Utilisateur (MGIU)*
   Il gère toutes les interactions avec l'utilisateur : l'authentification et l'enregistrement des utilisateurs mobiles, la classification des terminaux mobiles, la gestion des profils de l'utilisateur, la découverte et la sélection des services, la localisation des terminaux mobiles et enfin, la notification aux utilisateurs des mises à jour des services.
2. *Module de Gestion de la Base de Données des Utilisateurs (MGBDU)*
   Il gère les données et les informations concernant les utilisateurs abonnés au MS (préférences des utilisateurs, les informations authentifiants les utilisateurs, etc.). L'ensemble de ces informations est sauvegardé, sous forme de Profils Utilisateurs, dans une base de données locale (BD_Utilisateurs) associée au MS.

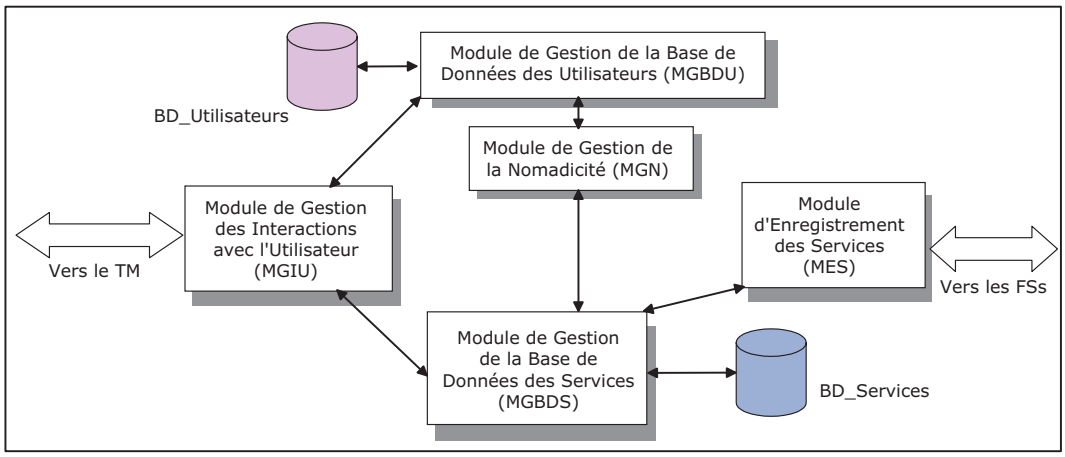


FIG. 7.2 – Modules internes du MS

3. *Module de Gestion de la Base de Données des Services (MGBDS)*
   Il est responsable de la gestion des informations décrivant les services disponibles. Ces informations sont sauvegardées dans une base de données de services (BD_Services) associée au MS.
4. *Module de Gestion de la Nomadicité (MGN)*
   Il gère toutes les interactions concernant les utilisateurs qui se connectent à des MSs, autres que leur MS Principal (chez qui ils s'enregistrent) et qui ont un accord de nomadicité avec le MS Principal. Il leur fournit toutes les informations concernant l'utilisateur et ses préférences.
5. *Module d'Enregistrement de Services (MES)*
   Il gère toutes les interactions entre le MS et les FSs qui concernent l'enregistrement des nouveaux services, le désenregistrement des services déjà enregistrés et, la mise à jour des services disponibles.

7.1.2 Composante FS

Les FSs sont les entités qui proposent, via le MS, des services aux utilisateurs mobiles. La composante FS est la partie de CASP installée chez le fournisseur de services. Elle est responsable des tâches suivantes :

- Enregistrement, désenregistrement et mise à jour des services.
- Gestion de l'accès et du téléchargement des services, à la demande des utilisateurs.

Les services proposés par les FSs, possèdent plusieurs versions (proxy) qui diffèrent entre elles par le coût, la langue utilisée ou les capacités du terminal nécessaires pour l'exécution du service sur le terminal.
Les services et les versions proposés sont localisés dans un répertoire géré par le FS.

La figure 7.3 présente les différents composants de la plate-frome FS :

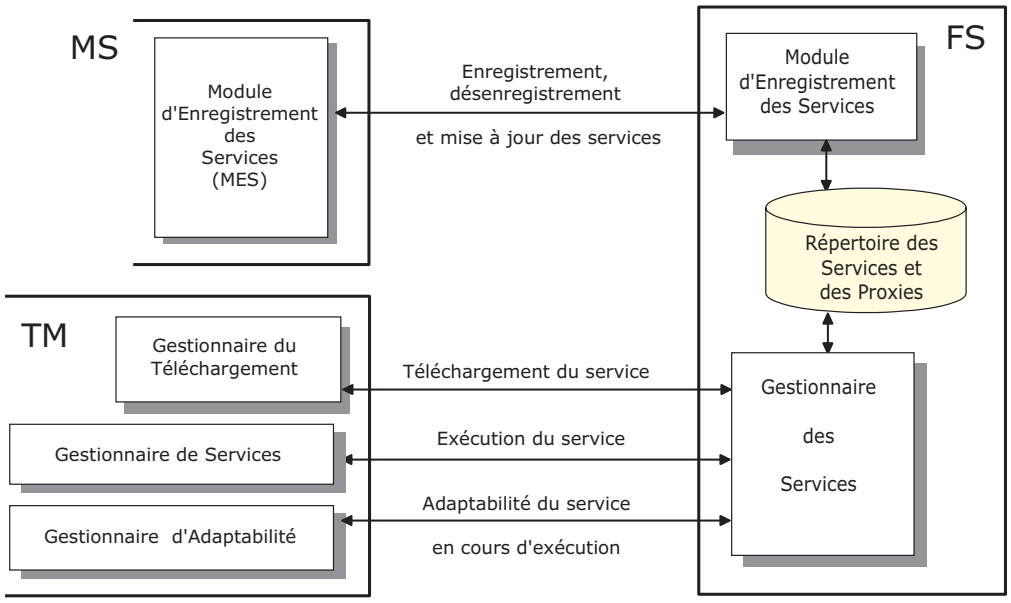


FIG. 7.3 – Modules internes de la composante FS

1. *Module d'Enregistrement des Services (MES)*

   Ce module gère l'enregistrement, le désenregistrement et la mise à jour des services.

2. *Gestionnaires des Services (GS)*

   Ce module assure l'accès aux services. L'interface d'accès d'un proxy est identifiée par une adresse IP et un numéro de port. Cette adresse est déclarée par le FS lors de l'enregistrement du service. Elle est utilisée ensuite par le terminal pour accéder au service.

7.1.3 Composante EUT

L'EUT (*End User Terminal*) est la partie de CASP installée sur le terminal. Elle est responsable des fonctionnalités suivantes :

- Gestion de l'interface graphique de l'utilisateur qui lui permet de découvrir, télécharger et exécuter les services.
- Gestion de l'authentification et de l'enregistrement des utilisateurs de manière securisée.
- Gestion des différents profils (terminal, utilisateur, sécurité).
- Gestion de la découverte et la recherche de services.
- Gestion du téléchargement de services.

– Gestion de l'exécution et de l'adaptabilité des services sur le terminal.

La figure 7.4 présente les différents modules qui constituent l'EUT (il s'agit de la partie A de la figure 7.1) :

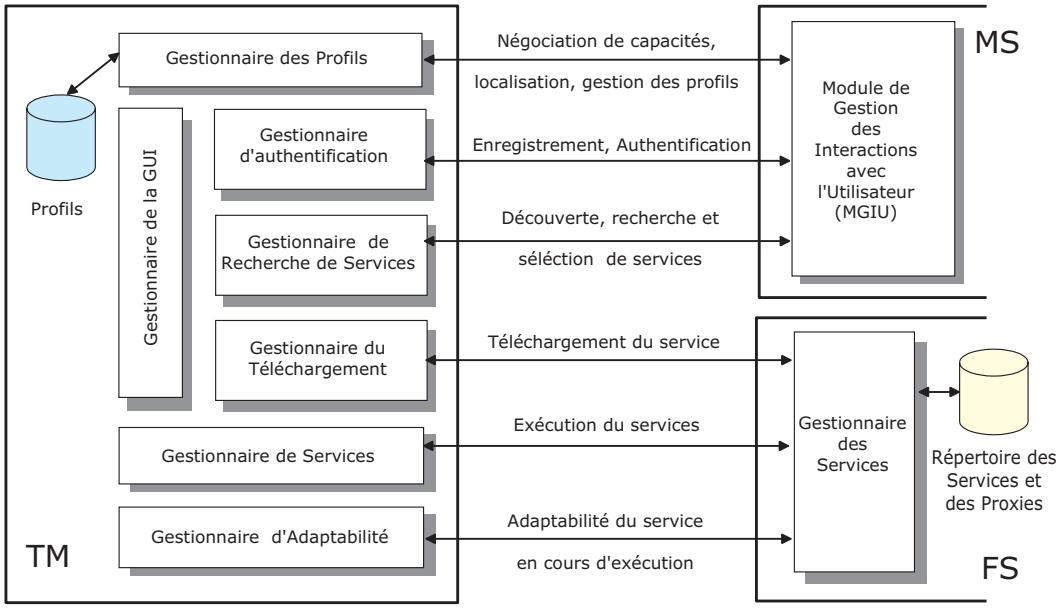


FIG. 7.4 – Modules internes de l'EUT

1. *Gestionnaire de la GUI*
   Il est responsable de la présentation, la gestion et la configuration de l'interface graphique de CASP sur le terminal.
2. *Gestionnaire d'Authentification (GA)*
   Il est responsable de l'enregistrement et de l'authentification de l'utilisateur (*Login*). L'enregistrement de l'utilisateur est effectué une seule fois, lorsque l'utilisateur signe un contrat d'abonnement avec le MS. Le Login de l'utilisateur est réalisé chaque fois que l'utilisateur ouvre une nouvelle session de découverte et de fourniture de services avec le MS.
3. *Gestionnaire des Profils (GP)*
   Il est responsable de la gestion des différents profils localisés sur le terminal. Il peut également interagir avec le MS pour l'échange de ces profils.
4. *Gestionnaire de Recherche de Services (GRS)*
   Il est responsable de la découverte de services sensibles au contexte pour le compte de l'utilisateur.

5. *Gestionnaire du Téléchargement de Services (GTS)*
   Il gère le téléchargement des services sur le terminal.

6. *Gestionnaire de Services (GS)*
   Il est responsable de l'exécution et du contrôle du service téléchargé sur le terminal. Il démarre le service téléchargé sur le terminal et contrôle son exécution.

7. *Gestionnaire d'Adaptabilité du Terminal (GAT)*
   Il est responsable de la gestion de l'adaptabilité des services à leur contexte d'exécution. Une étude détaillée sur le fonctionnement de ce module sera donnée dans les chapitres 8 et 9.

7.1.4 Fourniture de services dans le cas de nomadicité

CASP assure la fourniture de services aux utilisateurs mobiles même dans le cas de nomadicité (*roaming*). En effet, l'utilisateur peut se connecter à un MS Visité, autre que son MS Principal (avec qui il s'est enregistré) et découvrir des services personnalisés et sensibles au contexte.

Lorsque l'utilisateur se connecte à un MS Visité, le module MGIU (Module de Gestion des Interactions avec l'Utilisateur) de ce dernier, détecte que l'utilisateur est un utilisateur étranger (abonné à un autre MS) et identifie son MS Principal. Il contacte alors le module MGN (Module de Gestion de la Nomadicité) du MS Principal, pour récupérer le Profil Utilisateur et les préférences de l'utilisateur.

Le MGIU du MS Visité, à partir des capacités du terminal, de sa localisation et du Profil Utilisateur (envoyé par le MGN du MS Principal) filtre la base de données des services et propose à l'utilisateur des services personnalisés et sensibles au contexte.

CASP permet également à un utilisateur nomade de récupérer des services proposés par son MS Principal, même dans le cas du roaming. De plus, l'utilisateur peut effectuer des mises à jour sur son Profil Utilisateur et sur ses données d'enregistrement. Le MGIU du MS Visité se charge, à la fin de la session de découverte de services, de transmettre ces modifications au MS Principal pour qu'il les prenne en considération.
CASP assure ainsi la mobilité du terminal qui correspond à la possibilité pour le terminal d'accéder aux services quel que soit l'endroit où se trouve le terminal et quel que soit le point d'accès à la plate-forme CASP (MS Visité ou MS Principal).

En permettant une découverte de services personnalisés et sensibles au contexte même dans le cas du roaming, CASP assure à la fois la **mobilité du terminal** et la **mobilité personnelle**. Elle réalise ainsi les principes du concept **VHE**.

7.2 Mise en œuvre de CASP

Dans le prototype que nous avons développé, l'annuaire de CASP est central. Il est conçu sous forme d'un seul MS qui gère les deux bases de données suivantes :

– une base de données locale *BD_ Services* qui regroupe les descriptions des services disponibles dans CASP.
– une base de données locale *BD_ Utilisateurs* qui regroupe les informations sur les abonnés de CASP.

7.2.1 Environnement de développement

Afin d'assurer la portabilité de la plate-forme CASP, quels que soient l'environnement et le terminal utilisés, le choix s'est porté sur la langage Java sous l'environnement Linux. Les profils utilisés dans la plate-forme CASP sont décrits en XML, selon les spécifications de CC/PP et de MExE.
CASP utilise le protocole CC/PP pour la négociation des capacités entre le terminal et le MS. Elle utilise également, le protocole HTTP pour l'échange de messages entre les différents composants de la plate-forme, comme le montre la figure 7.5.

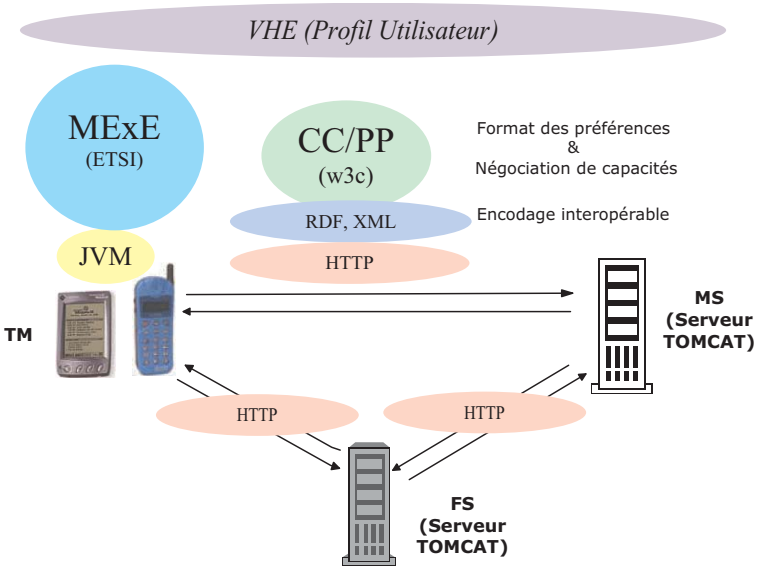


FIG. 7.5 – Environnement de développement de la plate-forme CASP

Le choix des serveurs pour le MS et le FS s'est porté sur TOMCAT parce qu'il supporte l'API Java et la sécurité. De plus, il est *"open source"*.

Les outils utilisés pour le développement de la plate-forme CASP sont :
– Java de Sun Microsystems : JDK 1.2.2
– Java Servlet JSDK 2.2.b
– Tomcat version 3.2.2
– Linux : Mandrake (version 7.2), noyau 2.4
– MIDP (Mobile Information Device Profile) : un profil qui fait partie de J2ME (Java 2 Micro Edition). Il identifie les APIs nécessaires pour le développement d'applications pour les terminaux mobiles de capacités limitées.

7.3 Scénarios d'évaluation de CASP

Les scénarios présentés dans cette section ont été développés pour démontrer d'une part, la capacité de la plate-forme CASP à assurer la découverte et la fourniture de services sensibles au contexte, et d'autre part, la possibilité d'utiliser CASP par des terminaux de capacités différentes.
L'évaluation du système consiste à démontrer que CASP supporte l'adaptabilité des services au profil de l'utilisateur, aux capacités du terminal et à sa localisation.

Plusieurs contraintes fonctionnelles ont été prises en considération :

- CASP doit gérer plusieurs terminaux de capacités différentes.
- Un même client peut utiliser plusieurs terminaux mobiles différents, avec un seul abonnement à la plate-forme CASP.
- Quel que soit le terminal utilisé, l'utilisateur doit retrouver un environnement personnalisé de découverte et de fourniture de services, aussi familier que possible (dans la limite des capacités du terminal).
- Les services proposés à l'utilisateur doivent correspondre aux capacités du terminal utilisé, aux préférences de l'utilisateur et à la localisation du terminal.

7.3.1 Environnement d'évaluation

Pour tester la plate-forme CASP, deux terminaux ont été utilisés :

- Un terminal puissant basé sur un ordinateur portable et utilisant la plate-forme Java.
- Un terminal limité basé sur la plate-forme Java J2ME qui émule les terminaux de capacités limitées.

Deux types de services ont été utilisés : l'un utilise la localisation du terminal et l'autre ne l'utilise pas :

- Un service Lecteur Multimedia appelé *Mobiplay* qui permet de télécharger des fichiers multimedia du réseau et de les exécuter sur le terminal.
- Un service basé sur la localisation du terminal, appelé *Locator*. Il s'agit d'un service d'envoi de messages qui permet de localiser la position géographique d'autres utilisateurs qui se trouvent dans le voisinage du terminal.

Deux versions de chaque service ont été utilisées : une basée sur MIDP pour le terminal de capacités limitées, et une autre, basée sur Java pour le terminal puissant.

7.3.2 Tests et Scénarios

Plusieurs scénarios ont été réalisés pour valider et démontrer les fonctionnalités suivantes de la plate-forme CASP :

- Gestion des profils
- Négociation des capacités du terminal et du contenu des services
- Découverte de services sensibles au contexte
- Téléchargement et exécution des services sur le terminal

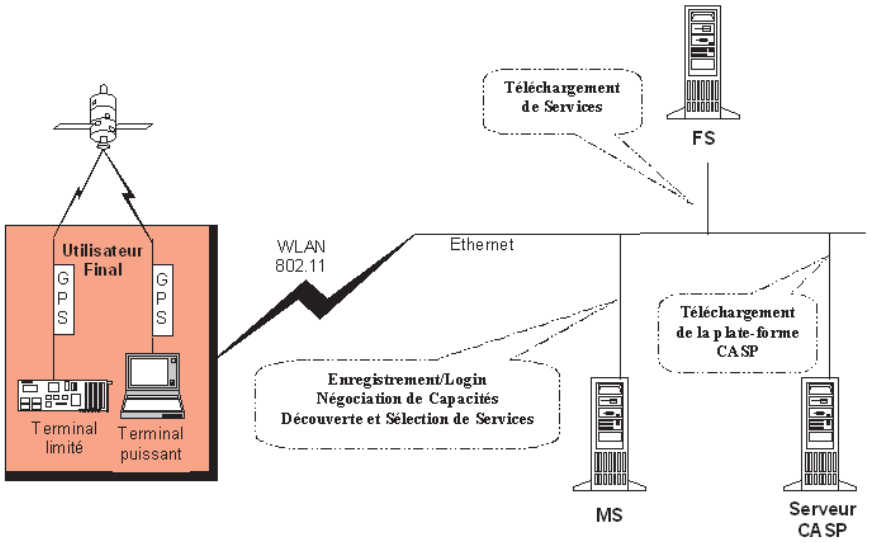


FIG. 7.6 – Plate-forme de test

7.3.2.1 Scénario 1

- *Objectif : Gestion des profils*
- *Acteurs : un utilisateur, un MS*
- *Test :*
  * Login.
  * Accès au profil.
  * Modification des profils.
  * Vérifier ensuite, si les modifications ont été prises en compte ou pas.

Ce scénario s'intéresse à la gestion des profils. L'objectif est de démontrer que la plate-forme CASP garantit que les utilisateurs retrouvent systématiquement le même environnement personnalisé, interfaces graphiques et services personnalisés, quelles que soient les capacités du terminal utilisé.

Dans ce scénario, l'utilisateur peut modifier son profil utilisateur, le profil de son interface graphique ainsi que le profil de la configuration du terminal utilisé. L'idée est de montrer que tous les changements effectués sur les profils, à la demande de l'utilisateur, sont pris en considération par la plate-forme CASP.

7.3.2.2 Scénario 2

- *Objectif : Négociation des capacités du terminal et du contenu des services*
- *Acteurs : deux utilisateurs, deux types de terminaux, un FS, un MS*

- *Test :*
  * Login du terminal TM1.
  * Découverte du service Mobiplay par le terminal TM1.
  * Sélection, téléchargement et exécution de Mobiplay par le terminal TM1.
  * Login du terminal TM2.
  * Découverte du service Mobiplay par le terminal TM2.
  * Sélection, téléchargement et exécution de Mobiplay par le terminal TM2.
  * Vérifier si les services proposés dépendent effectivement des capacités du terminal utilisé ou pas.

Ce scénario a démontré que l'utilisateur ne peut découvrir que les services qui sont sensibles au contexte. La liste des services proposés à l'utilisateur correspond à la combinaison : capacités du terminal, préférences de l'utilisateur et localisation du terminal.
Deux utilisateurs, ayant deux terminaux de capacités différentes, demandent le même service. Les résultats montrent que le MS fournit à chaque utilisateur, selon les capacités du terminal, une version différente du service.

7.3.2.3 Scénario 3

- *Objectif : Hététérogéinité des terminaux*
- *Acteurs : deux utilisateurs, deux types de terminaux, un MS, un FS*
- *Test :*
  * Négociation de capacités et contenu.
  * Découverte et sélection des services sensibles au contexte.
  * Téléchargement du service.
  * Gestion des services.
  * Gestion des profils.

L'objectif principal de ce scénario est de démontrer que la plate-forme CASP peut utiliser plusieurs types de terminaux, allant d'un téléphone mobile de très faibles capacités, à un ordinateur de bureau puissant. Toutes les fonctionnalités ont été testées pour démontrer la facilité et la flexibilité de la découverte et la fourniture de services sensibles au contexte.

7.4 Evaluation des performances de CASP

Les différents modules et toutes les interactions entre les composants de la plate-forme CASP ont été implémentés et validés avec succès dans le cadre du projet MOBIVAS. Toutes les fonctionnalités ont été testées pour illustrer le mécanisme de découverte et de fourniture de services sensibles au contexte.

Le tableau 7.1 donne un aperçu sur des mesures de temps effectuées sur les principales opérations d'un processus de découverte de services à savoir, l'enregistrement de l'utilisateur, le login de l'utilisateur, la sélection et le téléchargement du service [RQF01b].
Les tests ont été effectués sur des machines de type : Pentium III, CPU cadensé à 650MHz avec 128 M de RAM. La figure 7.6 représente l'environnement d'évaluation de CASP.

Fonctionnalités	Temps en s
Enregistrement	16
Login	1,2
Sélection et téléchargement du service	
Téléchargement de Mobiplay (taille du service : 311.4 ko)	46
Téléchargement de Locator (taille du service : 576 ko)	54
Opérations de sécurité	Mobiplay : 43 Locator : 46

TAB. 7.1 – Evaluation du temps dans CASP (un seul utilisateur)

Les valeurs des mesures temporelles obtenues pour un seul utilisateur sont considérables et empêchent ainsi, d'obtenir des résultats satisfaisants pour un nombre important d'utilisateurs.

L'argument principal derrière l'obtention des ces résultats est lié d'une part, à la multitude des opérations de sécurité effectuées à chaque étape de la découverte et la fourniture de services, et d'autre part, à la lourdeur de leur exécution sur la machine virtuelle Java [YHF+03]. En effet, dans chacune des ces étapes (enregistrement de l'utilisateur, login de l'utilisateur, sélection et téléchargement du service), plusieurs opérations cryptographiques (opérations de sécurité) sont effectuées sur chaque composant de CASP impliqué dans cette étape, afin d'assurer l'authentification des entités et la confidentialité des données. Ceci alourdit considérablement le temps de réponse aux requêtes de l'utilisateur.
Le tableau 7.1 montre également que le temps de téléchargement du service Mobiplay est autours de 46 secondes dont 43 secondes sont utilisées juste pour les opérations cryptographiques.

Le protocole de sécurité utilisé dans CASP a été développé par d'autres partenaires du projet MOBIVAS et intégré dans notre système de découverte de services. Il est basé sur une méthode de chiffrement (protocole RSA [RSA]) qui consomme beaucoup de temps, en particulier pour les terminaux à faibles ressources système [BS03a].

7.5 Discussion

Les performances obtenues ne mettent en cause, en aucun cas, le choix des concepts proposés pour assurer la découverte et la fourniture de services à savoir : la sensibilité au contexte, l'utilisation des profils basés sur XML, le VHE, la description standard des services, la négociation des capacités du terminal et du contenu des services. En effet, les résultats des tests de performance sont biaisés par la multitude d'opérations cryptographiques réalisées, en particulier sur les serveurs (MS et FS). De plus, les résultats montrent

que, si les opérations de sécurité ne sont pas réalisées, le temps de téléchargement du service Mobiplay ne dépasserait pas les 3 secondes (3s = 46s - 43s).

Afin de rendre notre proposition adaptée à une grande échelle d'utilisateurs, plusieurs solutions sont possibles :

1. Remplacer le protocole de sécurité utilisé dans CASP par une autre solution standard plus rapide, adaptée aux environnements mobiles et basée sur des opérations cryptographiques moins coûteuses que le protocole développé dans MOBIVAS.
2. Les protocoles de recherche et de découverte de services actuellement utilisés et présentés dans le chapitre 5, ne sont pas sensibles au contexte. Ainsi, une autre solution serait de porter les concepts et les idées développés dans CASP (sensibilité au contexte, description détaillée des services, utilisation de profils, XML, CC/PP, etc.) sur un des protocoles de découverte de services disponibles et adressant le passage à l'échelle, tels que SLP, UDDI, etc.

7.6 Conclusion

Ce chapitre décrit la mise en œuvre de notre plate-forme CASP et des services qui lui sont associés. Les scénarios décrits dans ce chapitre ont pour but de valider la capacité de la plate-forme à assurer la découverte et la fourniture de services sensibles au contexte.
Nous avons démontré, par l'utilisation de divers terminaux, services et versions de services, l'accès à notre plate-forme et l'utilisation des services par des terminaux de différentes capacités. Nous avons montré, également, que la plate-forme permet d'assurer la découverte et la fourniture de services sensibles au contexte. En effet, CASP ne fournit aux utilisateurs mobiles que les services qui respectent la combinaison : préférences de l'utilisateur, capacités du terminal et localisation de l'utilisateur.
Les scénarios présentés dans ce chapitre ont été exécutés, testés et validés dans le cadre du projet MOBIVAS. Néanmoins, les résultats des évaluations de CASP effectuées pour un seul utilisateur, montrent que la plate-forme ne peut pas passer à l'échelle dans sa version actuelle. Les performances sont biaisées par le protocole de sécurité développé dans MOBIVAS et intégré dans CASP.
Les évaluations temporelles ne mettent pas en cause les concepts de notre système de découverte de services sensibles au contexte. Une solution serait de tester la validité de ces concepts sur n'importe quel système de découverte de services qui passe à l'échelle.

Chapitre 8

L'adaptabilité dans les environnements mobiles

Avec l'introduction des systèmes informatiques mobiles, tout ce qui est dans l'environnement de l'utilisateur est susceptible de changer. En effet, en plus des changements dans les préférences de l'utilisateur et la localisation du terminal, l'environnement mobile peut changer dans le temps. Des terminaux mobiles peuvent être connectés ou déconnectés du réseau et peuvent commuter entre les différents points d'accès. De plus, les ressources du réseau et des terminaux sont sujets à des variations dans leur disponibilité.

Les applications conçues pour ces environnements doivent être capables de s'adapter pour fournir l'information appropriée face aux changements constants de l'environnement. Les applications traditionnelles ne sont pas conçues pour prendre en compte de telles variations qui résultent d'un comportement généralement imprévu. La solution est donc de permettre l'**adaptabilité** de l'application et des mécanismes système.

Ce chapitre présente une analyse détaillée des problèmes relatifs à l'adaptabilité des systèmes informatiques. Cette analyse tente de définir clairement les enjeux de l'adaptabilité dans les environnements mobiles et de dégager les fonctionnalités nécessaires à tout système prenant en charge cette problématique. Dans le chapitre suivant, nous présentons une infrastructure de fourniture de services adaptables pour les environnements mobiles.

8.1 Introdution

Les avancées dans le domaine de l'informatique mobile ont apporté de nouveaux défis qui dépassent le cadre traditionnel des systèmes distribués [Sat96]. En effet, la mobilité des terminaux et des utilisateurs, ainsi que les contraintes matérielles, conduisent à une variation importante des ressources qui doit être prise en compte pour conserver, voire améliorer la qualité du service rendu à l'utilisateur.
Les variations concernent aussi bien les ressources système (latence, bande passante disponible, mémoire, CPU disponible, etc.) que les ressources de niveau applicatif (localisation

du terminal et préférences de l'utilisateur). Leur détection nécessite la coopération du système utilisé en support et de l'application elle-même.

L'adaptabilité suppose qu'un certain nombre de décisions d'implémentation, traditionnellement prises pendant le codage de l'application, doivent être reportées jusqu'au moment où les informations nécessaires sont réellement connues. De plus, ces décisions qui dépendent des conditions d'exécution, doivent pouvoir être constamment remises en question si les conditions correspondantes évoluent.

Dans la suite de ce chapitre, nous présentons les principes généraux de l'adaptabilité. Ensuite, nous proposons une synthèse des différentes propositions et solutions permettant la mise en œuvre de l'adaptabilité dans les environnements mobiles.

8.2 Analyse conceptuelle de la notion d'adaptabilité

Nous consacrons cette partie aux différents concepts et aspects liés à l'adaptabilité.

8.2.1 Qu'est ce l'adaptabilité ?

Les termes *"adapter"* et *"adaptabilité"* peuvent être définis de la façon suivante [Hac] :

- **Adapter** : rendre apte à assurer des fonctions dans des conditions particulières ou nouvelles.
- **Adaptabilité** : qualité de qui peut être adapté, de ce qui peut s'adapter.

Suivant [SC01] :

On appelle *"adaptation"* ou *"adaptabilité"* d'un système, un changement dans le système pour prendre en compte un changement dans l'environnement du système.

L'adaptation d'un système logiciel (S) est causée par un changement (δ_E) d'un environnement initial (E) à un nouvel environnement ($E\prime$), et produit un nouveau système ($S\prime$) qui satisfait les besoins de l'environnement ($E\prime$). L'adaptation peut donc être vue comme une fonction :

$$E \text{ x } E\prime \text{ x } S \longrightarrow S\prime, \text{ où } satisfait(S\prime, besoin(E\prime))$$

Un système est *"adaptable"* ou *"adaptatif "* si l'on peut trouver une fonction d'adaptation.

L'adaptation implique trois tâches :

1. Capacité d'identifier les changements (δ_E).
2. Capacité de déterminer les changements (δ_S) à effectuer sur le système (S) selon les changements (δ_E).
3. Capacité de réaliser les changements (δ_S) afin de produire le nouveau système ($S\prime$).

Ces tâches peuvent être exprimées sous forme de fonctions comme suit [SC01] :

$IdentChangEnvironnement : E\prime - E \longrightarrow \delta_E$
$IdentChangSystem : \delta_E \text{ x } S \longrightarrow \delta_S$
$ChangSystem : \delta_S \text{ x } S \longrightarrow S\prime$, où $satisfait\ (S\prime, besoin(E\prime))$.

La fonction *satisfait* implique deux tâches : la validation et la vérification qui confirment que le système ($S\prime$) satisfait réellement les besoins de l'environnement changé ($E\prime$).

Il faut préciser que la généralité de la notion d'adaptabilité implique que, selon les différentes interprétations que l'on choisit, selon les besoins de l'application considérée, selon les systèmes et la culture des programmeurs, il est possible d'avoir de nombreuses approches différentes à la réalisation de l'adaptabilité. Ce sont les différentes approches que l'on présentera dans la section 8.4.

8.2.2 Fonctionnalités requises pour l'adaptation

Le schéma de mise en œuvre de l'adaptation est toujours le même et peut être séparé en plusieurs phases :

1. Détection d'un changement significatif.
2. Analyse de l'impact de ce changement sur le système et détermination de la réaction appropriée sous forme de modifications à appliquer sur le système.
3. Applications des modifications décidées.

Pour modéliser les trois étapes de mise en œuvre d'une adaptation, le système doit assurer les trois fonctionnalités suivantes [Dav01] :

1. **Déclenchement**

   Le système doit être capable de découvrir quelles sont les ressources disponibles, à la fois physiques et logicielles, ainsi que de détecter les variations dans la disponibilité de ces ressources. Ce sont ces variations qui seront la cause première du déclenchement des adaptations. Ce déclenchement peut être effectué par différents acteurs (par ex. utilisateur ou sonde logicielle).
   Il est nécessaire d'avoir un moyen pour observer le comportement d'un système en cours d'exécution. Sans une telle information, il n'est pas possible de déterminer si un changement doit être fait ou pas.

2. **Décision**

   L'étape de décision consiste à déterminer les modifications qui doivent être effectuées pour réagir aux changements significatifs détectés dans l'étape précédente. Certains changements ne donneront lieu à aucune modification.

3. **Réalisation**

   Une fois que le système a détecté une modification significative de l'environnement d'exécution et qu'il a décidé que celle-ci devait entraîner une réaction de sa part, il doit effectuer les adaptations identifiées dans la phase de décision.

8.2.3 Dynamisme de l'adaptabilité

Le critère de classification que nous considérons dans cette partie est le dynamisme de l'adaptation. Trois solutions possibles ont été définies [Mar98] : l'adaptation statique, l'adaptation spécifiée statiquement mais effectuée dynamiquement et l'adaptation spécifiée et effectuée dynamiquement.

8.2.3.1 Adaptation statique

Ce cas correspond à une adaptation effectuée **avant** l'exécution de l'application, en fonction des connaissances détenues de l'environnement de déploiement, ou à une adaptation faite pendant une phase d'initialisation de l'application.

8.2.3.2 Adaptation spécifiée statiquement mais effectuée dynamiquement

Cette solution consiste à prendre en compte, pendant la construction de l'application, les différentes variations de l'environnement et de définir les actions d'adaptation (la stratégie) en conséquence. La stratégie est ainsi spécifiée de manière statique (avant le lancement) mais les adaptations sont effectuées dynamiquement. C'est l'approche utilisée dans le cas des applications réparties classiques où les constructeurs décident d'utiliser un protocole donné pour la gestion d'un problème relié aux fluctuations de l'environnement. Le choix est statique alors que le fonctionnement de ce protocole prend en compte des informations dynamiques.

8.2.3.3 Adaptation spécifiée et effectuée dynamiquement

Cette dernière approche va encore plus loin et permet, non seulement d'effectuer de l'adaptation pendant l'exécution mais aussi la définition et la mise en place dynamique de la stratégie d'adaptation.

8.3 Fourniture adaptative des services

Nous nous intéressons dans cette section à l'adaptation des services dans le cadre de leur fourniture dans les environnements mobiles.

8.3.1 Pourquoi adapter les services ?

Pour illustrer les besoins en adaptabilité d'une application de fourniture de services dans un environnement mobile, nous allons étudier l'utilisation d'une application de flux vidéo sur un assistant personnel avec une connexion sans-fil. Ce scénario a été présenté dans [Led01].
À la conception de cette application, le programmeur décide de prévoir et d'implanter deux adaptations possibles. La première, consiste à diminuer le nombre d'images transmises par seconde, et la seconde consiste à dégrader la qualité de l'arrière plan de la vidéo.

Ensuite, tout en se déplaçant, l'utilisateur exécute son application et plusieurs adaptations sont alors envisageables. Lors d'une diminution de la bande passante due aux interférences de la connexion sans-fil, une possibilité d'adaptation est d'adopter le mode prévu de dégradation de l'arrière plan de la vidéo. Lors d'une congestion entre le serveur vidéo et l'assistant, une autre possibilié est de diminuer le nombre d'images transmises par seconde. Des adaptations non prévues peuvent également avoir un sens. La localisation de l'utilisateur peut être utilisée pour choisir le serveur vidéo le plus proche.
Enfin, l'application peut également évoluer en adaptant ces fonctionnalités ou en intégrant de nouvelles. Si le client utilise un décodeur MPEG-2, lors de la connexion à un serveur vidéo utilisant un encodage MPEG-4, celui-ci doit pouvoir charger et installer le décodeur correspondant. L'intégration de nouvelles fonctionnalités peut se faire à l'initiative de l'utilisateur, par exemple, si celui-ci veut intégrer un module effectuant des retours en arrière, ou à l'initiative d'un tiers, par exemple, si le fournisseur de flux vidéo souhaite ajouter un module de télépaiement.
Ce scénario montre qu'il existe plusieurs types d'adaptation de services pouvant intervenir à différents moments et à l'initiative de différents acteurs.

8.3.2 Adapter les services à quoi ?

Plusieurs paramètres du contexte peuvent changer pendant l'exécution d'un service. Parmi ces paramètres, nous pouvons citer :

- Les préférences de l'utilisateur (besoins de la QoS, sécurité, performance).
- La localisation du terminal.
- L'hétérogénéité des terminaux, des réseaux, des systèmes d'exploitation, des langages de programmation, etc.
- La variabilité des ressources (bande passante du réseau, batterie du terminal, puissance du CPU, mémoire du terminal, etc.).

8.3.3 Adaptation des services

Nous avons vu dans les chapitres précédents que la découverte de services sensibles au contexte joue un rôle important dans la recherche des services qui répondent au mieux aux besoins de l'utilisateur. Une fois ces services sélectionnés et lancés, ils ont besoin d'être adaptés aux contexte de leur exécution. Une telle adaptation peut avoir plusieurs formes, et peut avoir lieu soit avant de démarrer le service, soit durant son utilisation.
On distingue deux types d'adaptation selon *ce qui est adapté* :

- ***Adaptation du contenu*** : Il s'agit de l'ensemble des formes d'adaptabilité qui choisissent le changement du contenu fourni par le service selon le contexte. Une forme typique de l'adaptation du contenu est le changement de l'encodage d'un flux multimédia ou le changement de la présentation d'un service selon le contexte.
- ***Adaptation logique*** : Un autre type d'adaptation consiste à changer la logique d'un service. Elle concerne l'occurrence et l'ordre des étapes d'exécution d'un service, ainsi que les relations logiques entre ces différentes étapes.

Un autre facteur important de l'adaptabilité d'un service est le moment de l'adaptation. On distingue deux catégories principales.

- ***Adaptation statique*** : Elle consiste à configurer les paramètres et réaliser l'adaptation au lancement du service. C'est la forme d'adaptation la plus basique et la plus commune. Si le contexte est relativement stable, ce type d'adaptation suffit. Néanmoins, dans les contextes dynamiques des utilisateurs mobiles, un autre type d'adaptation, moins statique, est nécessaire.
- ***Adaptation dynamique*** : Ceci implique l'adaptation du service ou de son contenu au moment de l'exécution du service. Un exemple de ce type d'adaptation est le changement de l'encodage d'un flux vidéo lorsque l'utilisateur mobile passe d'une bande passante suffisante à une bande passant réduite.

8.4 Les systèmes adaptables dans les environnements mobiles

Plusieurs architectures et systèmes supportant la mobilité et l'adaptabilité ont été présentés dans les domaines de l'informatique mobile et l'informatique omniprésente (*Ubiquitous Computing*) [Dem94]. Alors que les approches de l'informatique omniprésente visent à adapter les environnements informatiques au contexte des utilisateurs [CK99], les approches proposées dans le domaine de l'informatique mobile tentent de s'adapter aux variations des ressources et d'étendre les applications conçues spécialement pour les environnements fixes, pour supporter cette adaptabilité.

8.4.1 L'adaptabilité dans l'informatique mobile

De nombreux travaux ont été menés pour adapter les environnements informatiques mobiles aux contraintes de la mobilité. Ces approches s'étendent du niveau système, en masquant la mobilité à l'utilisateur, jusqu'au niveau applicatif, où l'application elle-même met en œuvre l'adaptation à la mobilité.

8.4.1.1 Approches masquant la mobilité aux applications

Les approches masquant la mobilité à l'utilisateur visent à fournir au terminal mobile les mêmes services que sur sa machine de bureau. Elles permettent aux applications qui ne sont pas conçues explicitement pour la mobilité, de s'exécuter sans modification dans les environnements mobiles. Ces approches se placent au niveau système ou réseau et consistent en une émulation et une adaptation des protocoles des systèmes statiques [Mou99, JHE99]. Les variations que l'on peut observer dans un environnement mobile sont des phénomènes de bas niveau. Elles influent donc directement sur les couches protocolaires utilisées par les applications. Pour rendre transparents les déplacements et les déconnexions, il s'avère nécessaire d'adapter les protocoles aux environnements mobiles. [Mou03] propose un état de l'art détaillé sur les approches masquant la mobilité aux applications. Il classifie ces dernières comme suit :

1. *Transparence de l'adressage IP :*

   Lorsque un terminal portable change de réseau, son adresse IP change. Les applications utilisant le protocole IP ne sont pas conçues pour gérer dynamiquement ce changement dynamique d'adresse. Des approches masquant l'adressage IP [DH98], par exemple les solutions Mobile-IP [Per96], permettent aux terminaux mobiles de garder leurs propres adresses IP malgré leurs déplacements dans le réseau et n'exigent aucune modification des applications.

2. *Transparence au niveau du protocole TCP :*

   Les terminaux portables peuvent se déplacer fréquemment tout en communiquant. Pendant ces déplacements, les données envoyées ou à recevoir par le terminal portable peuvent être perdues à cause d'interférences ou de changements de cellules. Le protocole TCP [Pos81] interprète ces pertes comme une situation de congestion et active alors les mécanismes appropriés. Ces mécanismes induisent alors une dégradation significative des performances.
   Plusieurs travaux optimisent le protocole TCP pour les environnements mobiles en utilisant des techniques différentes. Certaines, comme Snoop [BSAK95], masquent complètement le lien sans-fil à l'émetteur et considèrent que le problème est local au lien sans fil et qu'il doit donc être résolu localement sans modifier la couche TCP de l'émetteur. Dans d'autres travaux comme Mobile TCP [SB98], l'émetteur est conscient du lien sans-fil et arrive à distinguer les pertes de transmission dues au lien sans fil de celles dues à une congestion. Dans ce cas, l'émetteur peut décider de l'activation des mécanismes de congestion.

3. *Transparence des systèmes de fichiers :*

   L'accès aux fichiers en environnements mobiles présente principalement deux inconvénients : (i) les pertes sur le lien sans fil sont considérées comme une situation de congestion et entraînent une dégradation des performances et, (ii) les déconnexions ne sont pas gérées et entraînent l'impossibilité d'accès aux fichiers.
   La dégradation des performances et les déconnexions ont été traitées avec des mécanismes de cache, comme l'illustrent les systèmes NFS/M [LOT98], MFS [AS99] et Coda [SMKO90] qui permettent aux utilisateurs de fonctionner, de manière transparente, en mode déconnecté.

4. *Transparence au niveau du protocole HTTP :*

   Le WWW est constitué d'un ensemble de données accessibles à distance en utilisant l'infrastructure de communication Internet. Ces données sont organisées en pages contenant du texte, des images, du son et de la vidéo. Elles sont publiées sur des serveurs Web et l'accès depuis les clients s'effectue selon le protocole HTTP.
   L'utilisation du protocole HTTP en environnements mobiles pose deux problèmes principaux : (i) l'encodage de données d'une requête HTTP s'effectue selon différents langages (HTML, XML, etc.) qui sont conçus pour une réalisation simple et facile par l'utilisateur mais qui ne sont pas optimisés pour le transfert de données, (ii) chaque requête HTTP correspond à une connexion TCP entre le client et le serveur

Web, ce qui entraîne un surcoût à l'établissement de la connexion et la possibilité de déclenchement des mécanismes de congestion liés à TCP.
Le masquage de la mobilité a été donc utilisé dans des systèmes d'accès au Web, tels que WebExpress [HL96a] et Mowgli [KRA94], qui utilisent des mécanismes d'interception et d'optimisation du protocole HTTP afin de réduire le volume du trafic sur le lien sans fil et le temps de latence des communications entre le terminal et le serveur.

Dans les approches masquant la mobilité, l'adaptation est portée entièrement par le système qui fournit le point focal pour l'arbitrage et la commande des ressources. Ces approches permettent d'adapter les applications existantes aux contraintes imposées par la mobilité. Elles ne nécessitent aucune modification de l'application et ne prennent pas en compte sa sémantique particulière. L'inconvénient de cette approche, est qu'il peut y avoir des situations dans lesquelles, l'adaptation réalisée par le système, est inadéquate pour certaines applications.

8.4.1.2 Approches intégrant l'adaptabilité aux applications

Les approches intégrant l'adaptation à l'application permettent de prendre en compte des informations liées à l'environnement et de réaliser une adaptation du comportement de l'application aux conditions changeantes. Une manière pour réaliser cette adaptation consiste en la collaboration entre le système et l'application. Le système surveille l'état des ressources, informe les applications des changements appropriés, et impose des décisions d'attribution des ressources. Une fois ces décisions annoncées, chaque application décide, de manière indépendante, de la meilleure façon de s'adapter. Dans une application de flux vidéo, par exemple, une telle adaptation permet à l'application, lors d'une baisse des ressources réseau, de réajuster la qualité de la vidéo et/ou du son.

Selon l'entité où réside la logique d'adaptation de l'application, les approches intégrant l'adaptabilité aux applications peuvent être classées en trois catégories :

- Approches utilisant une adaptation basée sur le client comme Odyssey [NSN+97]. Ces approches permettent aux applications de réagir aux changements environnementaux sur les clients (terminaux mobiles).
- Approches utilisant une adaptation basée sur le client-serveur (projet Rover Toolkit [JTK96] par exemple) qui permettent à des applications de s'adapter sur le client et sur le serveur.
- Approches utilisant une adaptation basée sur le proxy telles que BARWAN [BKC+98]. L'adaptation de l'application est assurée sur un serveur proxy spécifique à l'application et situé dans le réseau fixe. Le serveur proxy joue le rôle d'intermédiaire entre des serveurs ordinaires et des clients mobiles hétérogènes.

Les approches intégrant l'adaptabilité aux applications permettent des adaptations plus appropriées que les approches basées sur le masquage. Néanmoins, en raison de l'intégration de l'adaptation aux applications, ces approches ne sont pas flexibles et sont souvent

limitées à la dégradation du comportement des applications alors que des améliorations sont également possibles. De plus, elles ne traitent que le contrôle des ressources réseau de bas niveau telles que la bande passante, et ne considèrent pas toutes les informations contextuelles.

8.4.2 L'adaptabilité dans l'informatique omniprésente

Contrairement aux approches conçues pour les environnements mobiles, les solutions proposées dans le domaine de l'informatique omniprésente profitent de l'avantage de la mobilité. Des systèmes sensibles au contexte tels que : ParcTab [SAG+93], Mobisaic [VB95], Dataman [AIB94] et VESPER [LYBP02b] ont été développés dans ce cadre et exploitent l'environnement local pour adapter les services existants selon les ressources disponibles. Néanmoins, dans ces approches, les informations contextuelles telles que la localisation des personnes ou des ressources disponibles dans le voisinage (une imprimante par exemple) sont étroitement liées à l'environnement physique.

8.5 Conclusion

Ce chapitre a pour but de faire un état de l'art sur l'adaptabilité dans les environnements mobiles. Ainsi, dans un premier temps, nous avons défini un certain nombre de principes généraux et critères de l'adaptabilité. Puis, nous avons présenté brièvement les différents travaux menés autour de cette problématique dans les domaines de l'informatique mobile et omniprésente.

Dans le chapitre suivant, nous verrons comment la plate-forme CASP assure une fourniture de services adaptatifs dans les environnements mobiles. La solution proposée permet de préserver, d'améliorer ou de dégrader la qualité du service en cours d'exécution sur le terminal de l'utilisateur, selon les variations du contexte.
La solution proposée évite la transparence complète des approches basées sur le masquage de l'adaptabilité afin de tirer bénéfice de l'aspect mobile de l'environnement considéré et d'améliorer ainsi le niveau du service. La solution intègre l'adaptation à la plate-forme CASP pour pouvoir se comporter correctement dans un environnement mobile en exploitant au maximum l'avantage du contexte.
Dans notre solution, la réaction aux changements du contexte se fait non seulement par des mécanismes de dégradation, mais aussi par des mécanismes qui permettent d'exploiter les ressources supplémentaires pour fournir aux utilisateurs mobiles des services de haute qualité.

Chapitre 9

Fourniture adaptative des services multimédia

Nous nous intéressons dans ce chapitre à l'adaptation des services multimédia qui, contrairement aux autres services, se caractérisent par une utilisation intensive des ressources et nécessitent le transfert d'importants volumes de données soumises à un respect des délais d'acheminement. Ce chapitre propose des solutions aux contraintes imposées par ces services et par les conditions variables du contexte dans lequel ils s'exécutent. L'objectif est de préserver, d'améliorer ou de dégrader la qualité du service fourni selon les variations de l'environnement. L'approche proposée consiste à adapter dynamiquement le contenu des services multimédia sur des nœuds intermédiaires, placés entre les fournisseurs qui proposent leurs services et les terminaux mobiles qui exécutent ces services. Sur ces nœuds, des mécanismes d'adaptation sont appliqués aux données multimédia et permettent de prendre en considération les variations dans le contexte.

9.1 Fourniture adaptative des services

Comme nous l'avons vu précédemment, l'adaptabilité des services dans CASP est gérée lorsque le service, une fois découvert et sélectionné par l'utilisateur, est téléchargé sur le terminal mobile pour y être exécuté. Le processus d'adaptation est partagé entre le fournisseur de services qui offre le service, et le terminal mobile qui télécharge et exécute ce service.
La figure 9.1 présente les différents composants de la plate-forme CASP qui ont été détaillés dans le chapitre 7. Les composants qui sont chargés de la gestion de l'adaptation sont les modules qui apparaissent en dessous de la ligne pointillée de la figure 9.1.
Dans ce qui suit, nous présentons en détail, ces modules et nous décrivons comment ils gèrent l'adaptation dynamique des services en cours d'exécution sur le terminal mobile.

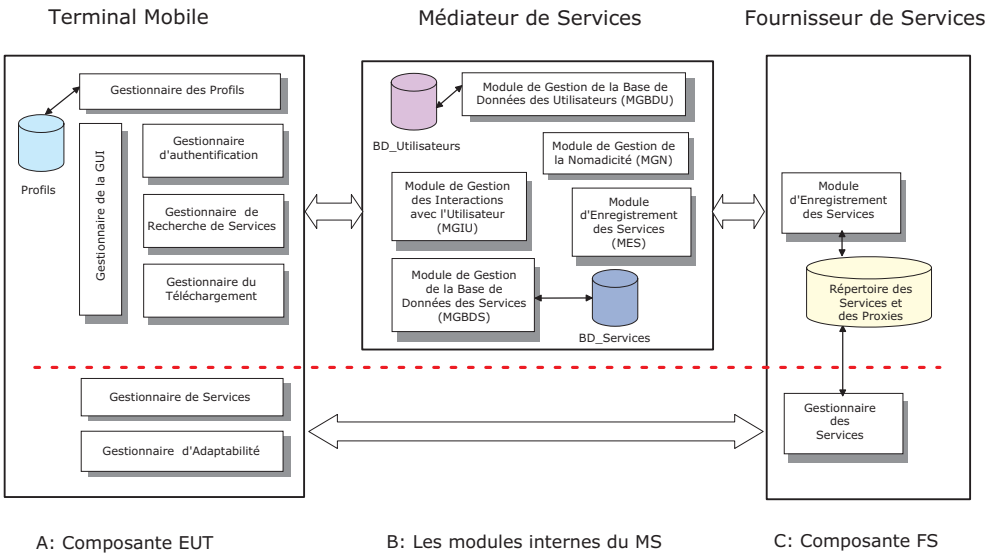


FIG. 9.1 – Gestion de l'adaptabilité dans la plate-forme CASP

9.1.1 Concept de service téléchargeable

Dans CASP, les services fournis par les fournisseurs de services FSs sont composés de deux parties : une partie mobile (téléchargeable) et une partie stationnaire (optionnelle). Cette dernière reste au niveau du FS et implante les fonctionnalités du service qui nécessitent des ressources non disponibles sur le terminal mobile. La partie téléchargeable est constituée d'une interface graphique du service et des fonctionnalités de base qui permettent à l'utilisateur d'accéder au service et d'assurer la communication avec la partie stationnaire, si cette dernière existe.

Deux types de services sont définis dans CASP :

1. *Services Autonomes* : ce sont des services qui n'ont pas besoin d'une partie stationnaire sur le réseau et qui sont téléchargés entièrement sur le terminal mobile lorsqu'ils sont fournis. Une fois que leur exécution commence sur le terminal mobile, les services autonomes n'ont besoin d'aucune interaction avec le réseau.
   Les services autonomes sont composés seulement d'une partie téléchargeable adaptée au contexte au moment du téléchargement du service sur le terminal mobile.
2. *Services Distribués* : contrairement aux services autonomes, les services distribués utilisent une partie stationnaire résidante sur le réseau.
   Lorsque l'utilisateur commence l'exécution d'un service distribué, la partie téléchargeable interagit avec la partie stationnaire sur le réseau pour le transfert du contenu. Les services multimédia sont un bon exemple de cette classe de services.
   L'adaptation au contexte des services distribués est réalisée au moment du téléchargement du service sur le terminal mobile et pendant son exécution (transfert du

contenu entre les deux parties stationnaire et téléchargeable).

9.1.2 Gestion de l'adaptabilité

Dans CASP, deux types d'adaptabilité sont supportés :

1. *Adaptabilité Statique :* assurée entre le terminal mobile et le Médiateur de Services (MS) durant les phases de découverte, de sélection et du téléchargement du service. Le service est adapté aux profils et aux préférences de l'utilisateur, aux capacités du terminal (concept de VHE) et à la localisation de l'utilisateur.
2. *Adaptabilité Dynamique :* assurée entre le terminal mobile et le fournisseur de services durant la phase d'exécution du service sur le terminal mobile. Le service est alors adapté aux ressources et aux capacités du terminal, aux ressources réseau, aux préférences de l'utilisateur et à la localisation du terminal.

L'adaptabilité statique est assurée pour n'importe quel type de service, qu'il soit distribué ou autonome. L'adaptabilité dynamique est assurée seulement pour les services distribués car la partie stationnaire, qui réside sur le réseau, interagit avec la partie téléchargeable sur terminal mobile pendant l'exécution du service. Dans ce cas, les changements dans le contexte, en particulier, les fluctuations dans les ressources réseau, peuvent considérablement affecter le transfert du contenu entre les deux parties du service.

Adapter les services multimédia, en particulier la vidéo, constitue une tâche difficile. Effectuer les adaptations en temps réel ajoute une nouvelle contrainte qui exige qu'une attention particulière soit prêtée à leur efficacité. L'approche proposée dans ce chapitre exploite la possibilité d'effectuer des adaptations sur des nœuds intermédiaires dans le réseau, appelés Serveurs Proxies.

9.1.3 Description de l'architecture

Les Serveurs Proxies sont placés sur des emplacements appropriés dans le réseau pour offrir différents niveaux de qualité qui correspondent à différents besoins contraints par le contexte.
Lorsque l'utilisateur choisit un service à exécuter, le terminal mobile se connecte au fournisseur de services FS à travers un Serveur Proxy, qui reçoit les données du FS, effectue une adaptation selon le contexte et transmet les données transformées au terminal. Les deux principales considérations dans le développement d'un schéma d'adaptation sur des Serveurs Proxies sont la conception de méthodes d'adaptation efficaces et la sélection du Serveur Proxy qui accomplit les adaptations.

Le traitement effectué par le Serveur Proxy consiste à adapter les services multimédia en temps réel aux contraintes imposées par le contexte. D'un point de vue général, la nature de cette opération comprend les fonctions suivantes :

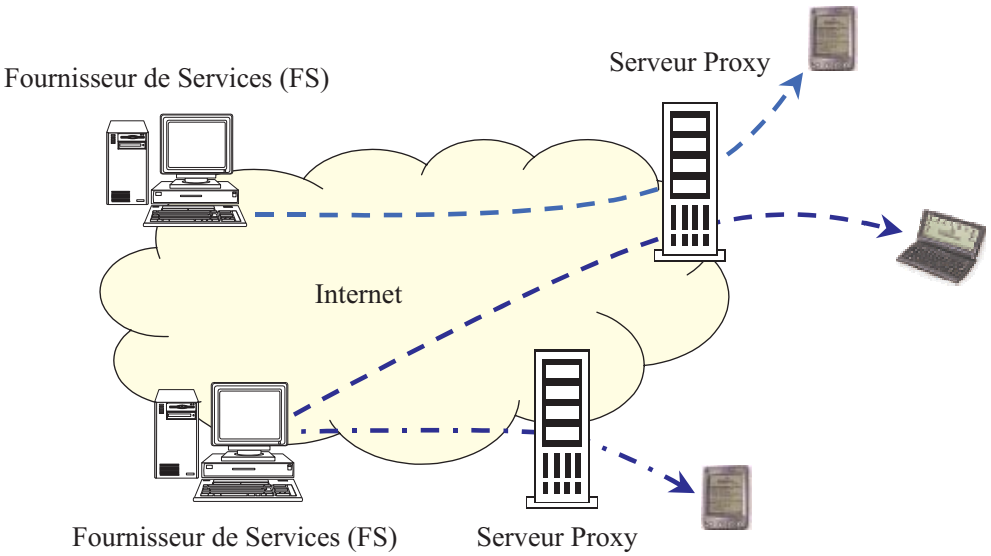


FIG. 9.2 – Utilisation des Serveurs Proxies

- Un Serveur Proxy reçoit un service multimédia à partir d'un fournisseur de services, adapte le contenu et envoie les données adaptées au terminal. L'adaptation des données s'effectue à partir d'un format arbitraire en entrée vers un autre en sortie.
- Durant l'exécution du service sur le terminal mobile, des traitements additionnels peuvent être appliqués pour effectuer des transformations sur le contenu du service afin de prendre en compte les changements dans le contexte. Par exemple, un service vidéo couleur est converti en monochrome.

L'approche proposée permet de traiter les services multimédia d'une manière généralisée, indépendamment de leur format de codage original. Elle offre aussi les méthodes d'adaptation qui permettent de prendre en compte les changements dans le contexte.

La figure 9.3 décrit globalement l'architecture d'adaptation dynamique. Elle présente les modules nécessaires sur le terminal mobile, sur le Serveur Proxy et sur le fournisseur de services, pour assurer la gestion de l'adaptation des services et le contrôle du contexte.

9.1.3.1 Terminal mobile

Lorsque le contexte change, le terminal mobile peut lui-même adapter le service, en cours d'exécution, au contexte, si ses ressources (mémoire, CPU, batterie, etc.) le permettent. Dans le cas contraire, il demande au Serveur Proxy d'effectuer l'adaptation à sa place.

Pour gérer l'adaptation des services, quatre modules ont été définis sur le terminal mobile :

1. *Moniteur d'Environnement* : c'est le module qui contrôle la disponibilité des ressources et leurs variations sur le terminal mobile. Il transmet périodiquement ces

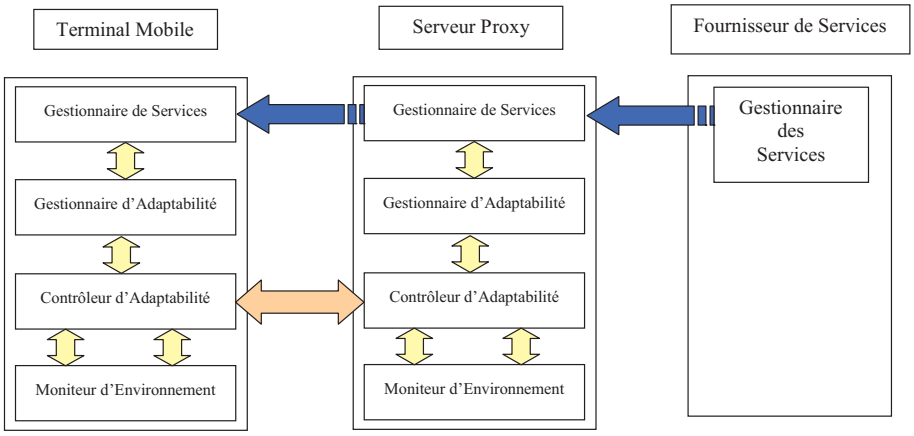


FIG. 9.3 – Composants responsables de la gestion de l'adaptation dynamique dans CASP

informations contextuelles au *Contrôleur d'Adaptabilité* ou à la demande de ce dernier.

2. *Contrôleur d'Adaptabilité* : c'est le module qui, selon la disponibilité des ressources et le type du service en cours d'exécution sur le terminal mobile, évalue si une adaptation du service est nécessaire ou pas et si elle doit s'effectuer sur le terminal mobile (adaptation locale) ou sur le Serveur Proxy.
3. *Gestionnaire d'Adaptabilité* : dans le cas d'une adaptation locale, le Gestionnaire d'Adaptatbilité choisit la stratégie d'adaptation à appliquer et effectue les changements nécessaires sur le service en cours d'exécution.
4. *Gestionnaire de Services* : c'est le module responsable de l'exécution et de la diffusion du service multimédia sur le terminal mobile.

9.1.3.2 Serveur Proxy

Sur le Serveur Proxy, on place quatre éléments, similaires à ceux présents sur le terminal mobile : un *Moniteur d'Environnement* pour contrôler les ressources contextuelles, un *Contrôleur d'Adaptabilité* pour analyser la nécessité d'adaptation des services, un *Gestionnaire d'Adaptabilité* pour gérer l'adaptation du service, et enfin, un *Gestionnaire de Services*, dont le rôle est la récupération du service du FS et son envoi au terminal mobile.

Le Contrôleur d'Adaptabilité sur le Serveur Proxy contrôle, à la fois, les ressources réseau et les autres informations contextuelles (capacités du terminal, localisation du terminal, préférences de l'utilisateur, etc.).
Dans le cas où l'adaptation du service ne peut pas se faire localement sur le terminal mobile à cause d'un manque de ressources par exemple, le Contrôleur d'Adaptation sur le terminal mobile envoie une requête au Contrôleur d'Adaptation sur le Serveur Proxy, dans laquelle il décrit l'ensemble des informations contextuelles disponibiles sur le terminal mo-

bile. Le Contrôleur d'Adaptation détermine ensuite, suivant les informations contextuelles, la qualité du service à appliquer.

9.1.3.3 Fournisseur de Services (FS)

Un seul module présent sur le FS intervient dans le processus d'adaptation de services en cours d'exécution sur le terminal mobile. Il s'agit du *Gestionnaire des Services* qui envoie le service demandé par le terminal mobile au Serveur Proxy. Ce dernier s'occupe, par la suite, de l'adaptation du service au contexte et de son transfert sur le terminal mobile.

9.2 Mise en œuvre de l'application MADSSERV

L'application MADSSERV (pour Media ADaptive Streaming Services) que nous allons présenter est une implémentation de l'architecture présentée dans ce chapitre. Elle consiste en la réalisation d'une adaptation au contexte d'un contenu multimédia demandé par un terminal mobile et sa diffusion sur celui-ci. L'adaptation consiste à modifier sa qualité afin d'assurer une fourniture de services sensible au contexte.

9.2.1 Scénario d'application

L'utilisateur d'un terminal mobile cherche à télécharger et à visionner un service qu'il a découvert grâce au mécanisme de découverte de services de CASP. Le service en question correspond à une séquence vidéo disponible chez un fournisseur de services FS. L'utilisateur effectue alors une requête de diffusion auprès du Serveur Proxy le plus proche du terminal mobile. Au cours de sa diffusion, les ressources dans le contexte changent. Pour fournir à l'utilisateur une qualité de diffusion correspondant aux ressources dont il dispose, le Serveur Proxy adapte la séquence vidéo au contexte (ex : dégrader la couleur, diminuer la fréquence d'affichage, supprimer l'image, améliorer la couleur, etc.).

9.2.2 Description de la plate-forme développée

MADSSERV est une application de type client/serveur pour la diffusion et l'adaptation de services multimédia qui utilise un programme indépendant permettant de choisir le niveau de qualité du service multimédia diffusé. On émule ainsi les Moniteurs d'Environnement et les Contrôleurs d'Adaptation, sur le terminal et sur le Serveur Proxy, qui n'ont pas été développés dans notre application.

MADSSERV se décompose donc de quatre éléments : le fournisseur de services, le Serveur Proxy, le terminal et un utilitaire de choix de niveau de qualité qui simule les composants de CASP manquants (voir figure 9.4).

9.2.2.1 Simulateur des composants de CASP manquants

La simulation permet de transmettre de manière interactive des changements de niveau de qualité. Ces changements n'ont aucune relation avec l'état des ressources dans

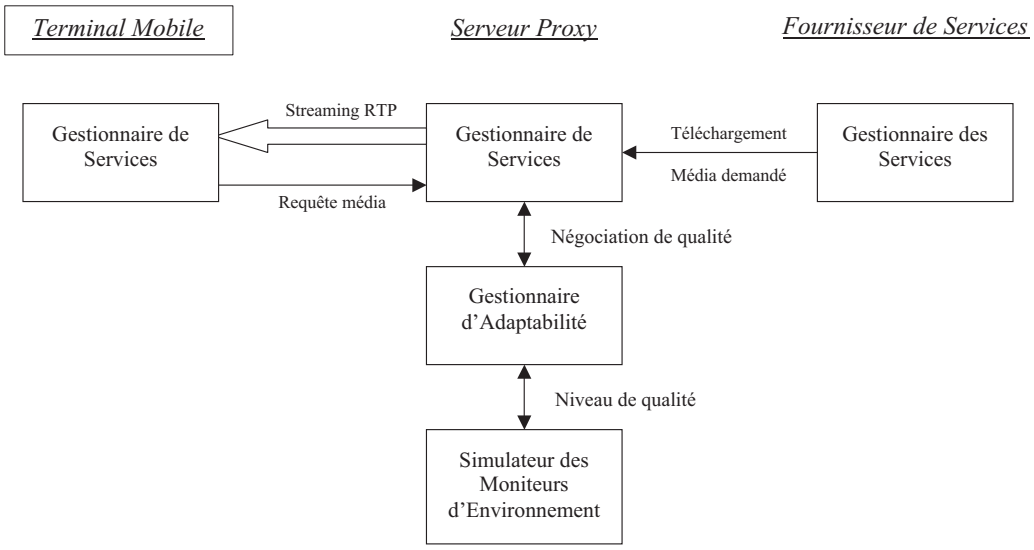


FIG. 9.4 – Modules développés pour assurer l'adaptation des services

l'environnement, ceci est dû à l'absence des Moniteurs d'Environnement et de Contrôleurs d'adaptation dont le rôle serait d'introduire une corrélation entre le niveau de qualité et l'état du contexte. Cette simulation est réalisée par un utilitaire possédant une interface graphique pour sélectionner le niveau de qualité à appliquer sur le service en cours d'exécution sur le terminal mobile (voir figure 9.5).

9.2.2.2 Serveur Proxy

Les modules développés sur le Serveur Proxy sont les suivants :

9.2.2.3 Gestionnaire de Services

Le Gestionnaire de Services réalise l'ensemble des opérations de traitement sur les services multimédia puis, les diffuse en assurant des transitions fluides lors du changement de niveau de qualité. Il utilise les bibliothèques JMF pour réaliser les quatre principales fonctions suivantes :

- Gestion des connexions : gère l'établissement et le suivi des connexions avec les terminaux mobiles et traite leur requêtes.
- Traitement de la conversion du contenu multimédia : convertit le fichier multimédia demandé par le terminal au format final de diffusion transmis par le *Gestionnaire d'Adaptabilité*.
- Gestion des transitions de format : gère le passage d'un format de diffusion à un autre (c'est-à-dire d'un niveau de qualité à un autre).

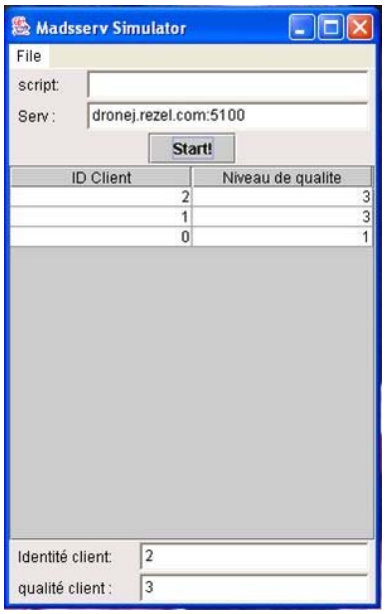


FIG. 9.5 – Simulateur (3 clients connectés)

– Streaming RTP : gère la diffusion du contenu multimédia vers le terminal.

9.2.2.4 Gestionnaire d'adaptabilité

Le Gestionnaire d'Adaptabilité a pour fonction d'imposer au Gestionnaire de Services le format final de diffusion du contenu multimedia. Pour cela, il prend en paramètres d'entrée le niveau de qualité (transmis par l'utilitaire de choix de niveau de qualité, Simulateur des composants CASP manquants) ainsi que le format original du service multimédia.
A partir de ces informations, le Gestionnaire d'Adaptabilité détermine le niveau de qualité optimal et le transmet au Gestionnaire de Services. Ce processus de décision inclut la vérification de l'intégrité de la conversion (disponibilité dans la JMF des outils de décodage et d'encodage nécessaires).

Le Gestionnaire d'Adaptabilité travaille selon plusieurs niveaux de qualité. Les niveaux suivants ont été identifiés (à chaque niveau est associé un format de média) :

– Niveau 1 : visualiser la séquence vidéo en couleur à une fréquence d'affichage f1.
– Niveau 2 : visualiser la séquence vidéo en couleur à une fréquence d'affichage f2.
– Niveau 3 : visualiser la séquence vidéo en noir et blanc.
– Niveau 4 : n'envoyer que le son avec des images extraites de la vidéo à des intervalles de temps réguliers.
– Niveau 5 : n'envoyer que le son.
– Niveau 6 : envoyer du texte si la transcription de la piste sonore est disponible.

L'association ressources disponibles/niveau de qualité est effectuée à terme par les Contrôleurs d'Adaptation (sur le terminal et sur le Serveur Proxy) dont le développement ne fait pas partie de cette application.

9.2.2.5 Terminal Mobile

Le terminal mobile permet à un utilisateur de demander la diffusion d'un service multimédia dont il spécifie l'adresse. Les requêtes sont gérées par le Serveur Proxy qui se charge de la diffusion du service. De façon transparente pour l'utilisateur, la qualité du service multimédia est modifiée au cours de sa diffusion pour s'adapter aux changements du niveau de qualité.

FIG. 9.6 – Madsserv terminal

9.3 Implémentation de MADSSERV

L'application MADSSERV décrite dans ce chapitre a été développée mais indépendamment du système de découverte de services de CASP.

9.3.1 Définition des niveaux de qualité

L'adaptation du service multimédia est basée sur des niveaux de qualité discrets. Le Simulateur permet d'associer un niveau de qualité à un terminal et de l'envoyer au Serveur Proxy. Ce dernier réalise l'association entre ce niveau de qualité et un format video-audio

à diffuser vers le terminal. Quatre niveaux de qualité ont été définis, le niveau 1 est associé à la qualité optimale (en général, la qualité la plus proche de l'original) et les niveaux suivants à des qualités progressivement dégradées. Etant donné que les formats de réencodage supportés dépendent des formats originaux du fichier demandé, il n'y a pas d'association systématique entre niveau de qualité et formats audio et vidéo. Le choix se fait à partir d'une table de charge réseau et système des formats de diffusion RTP[1].

9.3.2 Ressources utilisées

L'application d'adaptabilité dynamique de services multimédia MADSSERV a été développée intégralement en Java. Elle fait appel aux fonctionnalités multimédia de l'API JMF (Java Media Framework) de Sun [JMF]. La diffusion des services à travers le réseau est assurée par le protocole RTP (implémentation fournie par la JMF).

9.3.2.1 Ressources utilisées du côté terminal mobile

- Linux.
- JMF 2.1.1c Cross Platform (version générique pour toute plate-forme Java avec un support de formats vidéo limité).

9.3.2.2 Ressources utilisées du côté Serveur Proxy

- Sun UltraSparc.
- Solaris 9.
- JMF 2.1.1c Solaris Performance Pack (version optimisée pour les plate-formes Solaris assurant un support très large de formats vidéo pour l'encodage et le décodage).

9.4 Expérimentation

L'objectif principal de cette expérimntation est de vérifier la faisabilité de l'adaptation des services multimédia développée grâce à JMF. Afin d'évaluer les performances de MADSSERV, plusieurs tests ont été implémentés. Ces tests portent sur les informations suivantes :

- *Temps de transition* : il s'agit d'évaluer les temps de transitions entre les différents formats de diffusion.
- *Taux d'utilisation des ressources système* : il s'agit de vérifier si la conversion de plusieurs flux multimédia simultanément est une lourde charge pour les capacités de calcul du Serveur Proxy ou pas.

[1] http ://java.sun.com/products/java-media/jmf/2.1.1/formats.html#RTPFormats

9.5 Évaluation de performances

Dans le cadre du développement de MADSSERV, nous avons effectué une série de tests sous diverses conditions :

1. Lancement du terminal.
2. Changement de qualité sur un terminal.
3. Lancement de plusieurs terminaux et changements de qualité.
4. Surcharge du Serveur Proxy avec 13 terminaux connectés simultanément.
5. Surcharge du Serveur Proxy avec 40 terminaux connectés simultanément.

Dans ce qui suit, nous allons faire l'analyse de chacun des premiers résultats que nous avons, par commodité, mis sous forme de tableau. Les informations sont extraites de fichiers de *logs* répartis dans tout le programme.

NB : *Les terminaux et le Simulateur se trouvent sur une machine différente de celle du Serveur Proxy.*

9.5.1 Lancement d'un terminal (client)

1. *Côté Client*

   Ici le client est lancé, il envoie l'URL du service à exécuter au Serveur Proxy qui lui répond en diffusant le service demandé au bout de 7 secondes. Le player vidéo met 4 secondes à s'initialiser et à diffuser. Il faut donc 11 secondes entre le moment d'envoyer la requête de diffusion d'un service multimédia et le moment de le visualiser sur le terminal mobile.

Fri Mar 21 00:51:51 CET 2003	0	Client	Client demarre	
Fri Mar 21 00:51:55 CET 2003	0	DialogServer	Recu Ports : 40000 40002	
Fri Mar 21 00:51:55 CET 2003	0	DialogServer	Envoie de l'URL	file:///c:\temp\lco.avi
Fri Mar 21 00:52:02 CET 2003	0	DialogServer	Recu message de debut de diffusion	2
Fri Mar 21 00:52:06 CET 2003	0	ClientRTPManager	player Video realise	
Fri Mar 21 00:53:50 CET 2003	0	DialogServer	Recu message de deconnexion	Deconnexion du client 0
Fri Mar 21 00:53:50 CET 2003	0	ClientRTPManager	Fin de session ClientRTPManager	
Fri Mar 21 00:53:55 CET 2003	0	Client	Fermeture du Client	

information démarrage/fermeture du client

information graphique réalisée par le client

information de dialogue avec le serveur

9.5.2 Changement de qualité sur un client

1. *Côté client*

   Même procédure que précédemment jusqu'au premier changement de qualité. 7 s plus tard, le précédent player est arrêté. Après remplissage du buffer du client (8 s), celui-ci diffuse le nouveau flux multimédia encodé dans la nouvelle qualité.
   Il faut donc 15 s entre le moment de détection d'un changement dans le contexte et le moment de visualiser le service adapté sur le terminal.

Fri Mar 21 01:56:54 CET 2003	0	Client	Client demarre	
Fri Mar 21 01:56:59 CET 2003	0	DialogServer	Recu Ports : 40000 40002	
Fri Mar 21 01:56:59 CET 2003	0	DialogServer	Envoie de l'URL	file:///c:\temp\lco.avi
Fri Mar 21 01:57:05 CET 2003	0	DialogServer	Recu message de debut de diffusion	2
Fri Mar 21 01:57:14 CET 2003	0	ClientRTPManager	playeur Video realise	
Fri Mar 21 01:57:32 CET 2003	0	DialogServer	changement de qualite 0	
Fri Mar 21 01:57:32 CET 2003	0	DialogServer	Temps actuel de diffusion 17.816000000000003	
Fri Mar 21 01:57:39 CET 2003	0	DialogServer	Recu message de debut de diffusion	2
Fri Mar 21 01:57:39 CET 2003	0	ClientRTPManager	Fin de session ClientRTPManager	
Fri Mar 21 01:57:47 CET 2003	0	ClientRTPManager	playeur Video realise	
Fri Mar 21 01:58:12 CET 2003	0	DialogServer	changement de qualite 0	
Fri Mar 21 01:58:12 CET 2003	0	DialogServer	Temps actuel de diffusion 24.665000000000003	
Fri Mar 21 01:58:14 CET 2003	0	DialogServer	Recu message de debut de diffusion	2
Fri Mar 21 01:58:14 CET 2003	0	ClientRTPManager	Fin de session ClientRTPManager	
Fri Mar 21 01:58:23 CET 2003	0	ClientRTPManager	playeur Video realise	
Fri Mar 21 01:59:16 CET 2003	0	DialogServer	Recu message de deconnexion	Deconnexion du client 0
Fri Mar 21 01:59:16 CET 2003	0	ClientRTPManager	Fin de session ClientRTPManager	
Fri Mar 21 02:01:48 CET 2003	0	Client	Fermeture du Client	

Affichage du nouveau player media

Requête d'un changement de qualité pour le client 0

2. *Côté Serveur Proxy*

   Nous avons constaté que jusqu'au premier changement de qualité, il n'y a aucune différence par rapport au cas précédent. A chaque demande de changement de format imposé par le Simulateur, le Serveur Proxy choisit un nouveau format adapté et demande la conversion dans le nouveau format. Le temps de réaction varie entre 3 s et 9 s (selon le type de codage du service à adapter).

9.5.3 Lancement de plusieurs clients et changements de qualités

1. *Côté Client*

   Dans ce test, trois clients se connectent simultanément au Serveur Proxy.

2. *Côté Simulateur*

   Le Simulateur est l'interface qui impose les changements de qualité au Serveur Proxy. On peut voir les connexions au Serveur Proxy de 3 clients (0, 1 et 2) avec le message

"ajout d'un nouveau client".
Le Serveur Proxy met entre 4s et 8 s à prendre en compte l'ordre de changement de qualité.

Fri Mar 21 02:35:36 CET 2003	Simulator	Ouverture du simulateur		
Fri Mar 21 02:36:49 CET 2003	DialogServer	ajout d'un nouveau client	0	1
Fri Mar 21 02:36:51 CET 2003	DialogServer	ajout d'un nouveau client	1	1
Fri Mar 21 02:36:55 CET 2003	DialogServer	ajout d'un nouveau client	2	1
Fri Mar 21 02:37:21 CET 2003	Simulator	Changement de Qualite	1	3
Fri Mar 21 02:37:25 CET 2003	DialogServer	mise à jour du client	1	3
Fri Mar 21 02:37:55 CET 2003	Simulator	Changement de Qualite	2	3
Fri Mar 21 02:38:03 CET 2003	DialogServer	mise à jour du client	2	3
Fri Mar 21 02:38:39 CET 2003	DialogServer	effacement Client : 2		
Fri Mar 21 02:38:40 CET 2003	DialogServer	effacement Client : 1		
Fri Mar 21 02:38:46 CET 2003	DialogServer	effacement Client : 0		

3. *Côté Serveur Proxy*
   Les temps de réaction au changement de qualité sont équivalents au cas précédent (de 4 à 8s). Il semble que 3 clients connectés n'altèrent pas les temps de traitement sur le Serveur Proxy.

9.5.4 Surcharges du serveur avec 13 clients connectés simultanément

Dans cette configuration, nous n'avons pas détecté de problème de ressources systèmes à 13 clients connectés simultanément. La connexion réseau était de 100Mbit/s au départ du Serveur Proxy. Plusieurs clients étaient implémentés sur différentes machines distantes avec des capacités hétérogènes (100 ou 10Mbit/s).
Les Résultats montrent que le temps d'adaptation de services varie entre 4 s et 13 s.

9.5.5 Surcharges du serveur avec 40 clients

Lorsque le nombre de clients est égal à 40 clients, le temps nécessaire pour adapter un service varie entre 4s et 20 s (selon le type de codage de la vidéo et le rang du service dans la liste des services à adapter par le Serveur Proxy) .

9.5.6 Discussion

D'après ces premiers résultats, nous pouvons constater que le temps d'adaptation d'un service multimédia dépend enormément du type de codage utilisé. En outre, plus le nombre de clients augmente, plus le temps de transition et d'adaptation des services augmente.

Si le contexte est très fluctuant et le nombre de clients est très grand, le système effectuera des adaptations consécutives sur chaque service et pour chaque client. Ce qui peut augmenter considérablement le temps de prise en compte des changements dans le contexte et donc, l'efficacité de l'adaptation elle-même.
Le système doit être capable d'assurer l'adaptation dynamique des services, si cette dernière n'influe pas sur le bon fonctionnement du système. La ***stabilité du système*** doit donc être une priorité majeure dans un système d'adaptation dynamique de services et doit être introduite comme paramètre d'entrée (comme toute autre information contextuelle) qui influe sur la décision d'adaptation et sur le choix de la stratégie d'adaptation à appliquer, lorsque le contexte change.
L'étude de la stabilité d'un système d'adaptation de services constitue une de nos principales perspectives de recherche.

9.6 Travaux similaires

L'idée d'employer des Serveurs Proxies n'est pas nouvelle. Elle a été introduite dans la spécification du protocole HTTP pour des raisons de performance. Depuis, elle a été utilisée dans plusieurs applications pour des besoins d'adaptation. La plupart des travaux effectués s'intéressent principalement à l'adaptation des médias discrets tels que les pages Web et les images. Dans ces travaux, des Serveurs Proxies HTTP interceptent les requêtes des clients et adaptent le contenu selon leurs besoins. Les adaptations communes à ces travaux consistent en des transformations de documents textuels (du format *postscript* au format *pdf*, par exemple) [HL96b, ZD97, KPMM01], des images [FGBA96, HBL+98] et des pages Web [FKN98] selon des formats supportés par les applications clientes.

Il existe aussi dans cette approche quelques travaux visant à adapter les flux continus comme l'audio et la vidéo en utilisant des techniques de transcodage qui consistent à transformer les données multimédia, à partir de leur format d'encodage original en entrée, vers un autre format en sortie. Un Serveur Proxy de transcodage reçoit les requêtes, et interagit avec le serveur vidéo pour fournir un contenu adapté aux clients.
Certaines adaptations consistent à remplacer la vidéo par un flux audio, ou une description textuelle de son contenu en fonction des capacités du client [SML98, MSL99].
D'autres travaux s'intéressent à des formats d'encodage particuliers et offrent des méthodes de transcodage spécifiques à ces formats qui permettent d'effectuer des adaptations sur la vidéo. En général, leurs objectifs consistent à adapter ces flux aux capacités réseau, soit par l'élimination des images dans une vidéo, soit en réduisant la qualité de la vidéo à l'encodage. La limitation de ces travaux est que les mécanismes employés restent liés à un seul format et ne peuvent pas être généralisés à tous les formats existants [AMZ95].

Peu de travaux ont utilisé la technique d'adaptation sur des nœuds intermédiares pour implémenter des adaptations dynamiques sur les flux continus comme l'audio et la vidéo [ALDPH03], [STS03] et [CZMN04]. Dans ces travaux, l'adaptation est basée sur une technique de transcodage de services multimédia. Elle consiste à dégrader le service pour répondre aux capacités du terminal et/ou aux ressources réseau. Les autres informations

contextuelles, telles que la localisation du terminal et les préférences de l'utilisateur, ne sont pas utilisées.

Pour finir, la plupart des travaux conduits n'offrent pas de solutions générales à tous les types de média utilisés et n'exploitent pas toutes les informations contextuelles pour adapter les services multimédia. En effet, ces derniers sont souvent adaptés aux ressources réseau ou/et aux capacités du terminal. De plus, ils utilisent souvent des techniques pour juste dégrader la qualité du service et ne proposent pas de solutions pour l'améliorer lorsque le contexte le permet.

9.7 Conclusion

Ce chapitre a présenté une architecture à base de Serveurs Proxies pour l'adaptation dynamique des services multimédia. Contrairement aux travaux existants, qui focalisent sur certains formats de codage et ne traitent que l'adaptation aux capacités réseau, l'approche proposée permet de traiter les services multimédia d'une manière généralisée et offre des méthodes d'adaptation qui permettent de prendre en compte les variations dans le contexte.
Les premières expérimentations effectuées avec cette architecture, montre qu'une attention particulière doit être prêtée à l'efficacité de l'adaptation dans un environnement très fluctuant et la nécessité de préviligier la stabilité du système à l'adaptation des services dans ce cas.
L'approche proposée dans ce chapitre, ne traite que l'adaptation du contenu des services et non pas leur logique. De plus, elle ne peut être utilisée que pour adapter des services multimédia. Dans le chapitre suivant, nous proposons une extension de la plate-forme CASP pour supporter une adaptation au contexte basée sur la composition des services.

Chapitre 10

Vers l'adaptabilité dynamique des services basée sur la composabilité

Dans le chapitre précédent, nous avons montré comment la plate-forme CASP assure l'adaptabilité dynamique des services multimédia. La solution proposée utilise des mécanismes de base de JMF pour assurer le transcodage du service afin de répondre aux changements du contexte.
La solution proposée s'intéresse à un type particulier d'adaptation. Il s'agit de l'adaptation du contenu de service. Pour pouvoir réaliser tout type d'adaptation (contenu de service ou logique de service), nous proposons d'étendre la plate-forme CASP avec une approche à base de *composants* qui offre une vue modulaire des services. Chaque service est fait d'un ensemble de composants interconnectés, capables d'être supervisés et reconfigurés si nécessaire. Un composant est une unité d'encapsulation des fonctions d'un module logiciel autonome qui offre une interface d'accès à son traitement.
Cette approche de conception de services modulaire permet d'adapter dynamiquement n'importe quel type de service (qu'il soit multimédia ou pas) en cours d'exécution. Une réalisation monolithique de ces services ne permettrait pas de prendre en compte tous les contextes possibles, et il serait irréaliste de vouloir générer une version de service pour chacun de ces contextes.
Dans ce chapitre, nous présentons les perspectives de notre travail de recherche qui concerne l'adaptabilité dynamique de services basée sur la composabilité. Nous donnons un bref apperçu des concepts de base relatifs à la notion de la composabilité. Puis, nous présentons une extension de la plate-forme CASP pour supporter la composabilité des services. Enfin, un scénario simple d'adaptation dynamique de services est présenté afin d'illuster notre approche.

10.1 Définitions et concepts de base

Un composant est, avant tout, une unité d'encapsulation de fonctionnalités et de déploiement (ce dernier point représente une différence majeure avec les objets) qui offre une,

ou plusieurs, interfaces pour exploiter ses fonctionnalités et pour exprimer ses dépendances fonctionnelles (i.e. les fonctionnalités fournies par le composant, et les fonctionnalités requises ou exigées par le composant et qui sont offertes par des composants tiers). Chaque interface doit refléter le plus précisément possible le comportement du composant. Les composants peuvent être personnalisés et assemblés avec d'autres composants à travers des *connecteurs* sans modification de leur code [RPPM00a, KAFFD03].

La figure 10.1 montre la composition de deux composants. Le *"Connecteur"* établit le lien entre l'interface fournie du *"Composant 1"* et l'interface exigée du *"Composant 2"*.

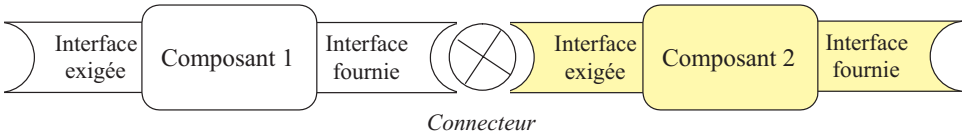


FIG. 10.1 – Assemblage des composants

Il est nécessaire d'utiliser le composant adéquat en fonction du contexte et de configurer les composants en fonction des besoins de l'application. Pour cela, il faut disposer des informations relatives à l'implantation d'un composant. Ces informations doivent, d'une part, préciser pour quelle plate-forme est destinée l'implantation du composant, et d'autre part décrire les besoins systèmes de l'implantation ainsi que leur configuration. Toutes ces informations sont regroupées dans le *descripteur du composant* permettant de choisir un composant en fonction du contexte.
Pour faciliter le déploiement et l'utilisation des composants, l'ensemble des caractéristiques de chaque composant doit être rendu disponible dans un format diffusable. Pour cette raison, les composants sont souvent archivés sous forme de paquetage contenant à la fois une forme binaire de l'implantation du composant, ses interfaces (son mode d'emploi), et son descripteur [RPPM00a].

10.2 Composition de services dans le cadre de la fourniture de services

Dans notre approche, la composition dynamique des services est basée sur l'utilisation de plusieurs composants qui sont assemblés pour fournir un service unique et accessible à travers une seule interface commune. Le choix des composants se fait au moment de la composition du service et ceci de manière transparente à l'utilisateur [Fer02, KAFFD03].

L'adaptation dynamique du service au contexte de son exécution consiste en général, à remplacer un composant par un autre qui peut éventuellement fournir des fonctionnalités différentes du composant initial. L'adaptation revient à déconnecter le composant du

service, à transférer son état courant vers un nouveau composant et à connecter ce dernier au service. L'opération d'adaptation requiert généralement l'arrêt du service et oblige à reprendre le cycle de vie à partir de la phase d'assemblage.

La création ou l'adaptation d'un service composé passe par les étapes suivantes :

- Localisation des composants nécessaires à la création d'un nouveau service composé ou à l'adaptabilité d'un service en cours d'exécution. Ces composants sont stockés dans une base de données où chacun possède une description claire et détaillée des différentes opérations qu'il peut effectuer, des différentes interfaces qu'il propose ainsi que leurs paramètres d'entrée et de sortie.
- Définition de la composition dynamique à exécuter qui doit être basée sur les besoins du contexte et les conditions d'efficacité du service résultant.

10.3 Architecture pour l'adaptation dynamique des services basée sur la composabilité

La figure 10.3 présente les différents éléments nécessaires, sur le terminal mobile et le Serveur Proxy, pour assurer l'adaptabilité dynamique des services basée sur la composabilité. L'architecture proposée est composée essentiellement de la plate-forme CASP à laquelle nous rajoutons un *Gestionnaire de Composabilité* pour gérer la composabilité des services.

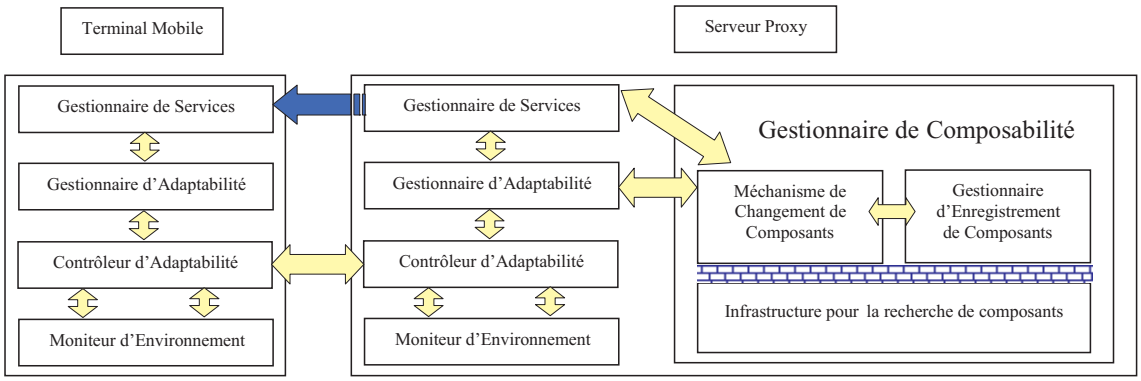


FIG. 10.2 – Architecture globale d'adaptation dynamique

Sur le terminal mobile, on retrouve les quatres éléments principaux définis pour gérer l'adaptabilité dans CASP. Le *Moniteur d'Environnement* (ME) contrôle la disponibilité des ressources sur le terminal mobile. Il transmet périodiquement ou à la demande du *Contrôleur d'Adaptabilité* (CA) les informations contextuelles.
Selon les ressources disponibles et selon le type du service en cours d'exécution, le Contrôleur d'Adaptabilité évalue la nécessité d'adaptation du service sur le terminal et informe le *Gestionnaire d'Adaptabilité* (GA) de la nécessité d'une adaptation locale (sur le terminal

mobile) du service, si celle-ci peut se faire sur le terminal. Cette adaptation est effectuée sur le Serveur Proxy si elle nécessite plus de ressources non disponibles sur le terminal ou requiert une transformation dans la structure et la composition du service. Dans ce cas, le Contrôleur d'Adaptabilité du terminal envoie un rapport sur les ressources disponibles sur le terminal mobile au Contrôleur d'Adaptabilité du Serveur Proxy.

Le Gestionnaire d'Adaptabilité sur le terminal mobile gère et contrôle l'adaptation du service si cette dernière est effectuée sur le terminal. Dans ce cas, il demande au Gestionnaire de Services (GS) d'effectuer les changements nécessaires sur le service tels que le téléchargement des plug-in nécessaires pour l'adaptation du service au contexte.

Sur le Serveur Proxy, on place quatre éléments similaires à ceux présents sur le terminal mobile. Il s'agit du Moniteur d'Environnement, du Contrôleur d'Adaptabilité, du Gestionnaire d'Adaptabilité, du Gestionnaire de Services ainsi que du Gestionnaire de Composabilité (GC). Le rôle de ce dernier consiste à gérer la composition des services pour les adapter dynamiquement au contexte.
Le Gestionnaire de Composabilité est lui-même constitué de trois éléments : Mécanisme de Changement de Composants (MCC), Gestionnaire d'Enregistrement de Composants (GEC) et une infrastructure pour la recherche de composants. Le MCC s'occupe du choix et de la composition des services. Le GEC se charge de l'enregistrement des composants et de la gestion de leur cycle de vie (mise à jour, modification et suppression des composants) par les fournisseurs des services. L'infrastructure de recherche de composants est responsable de la découverte et de la recherche des composants pertinents.

Sur le Serveur Proxy, les variations dans le contexte d'exécution du service sont notifiées au Contrôleur d'Adaptabilité par le Moniteur d'Environnement. Le Contrôleur d'Adaptabilité décide si une adaptation du service est nécessaire ou pas et envoie sa décision au Gestionnaire d'Adaptabilité. Ce dernier décide alors soit, de charger des nouveaux composants afin de les inclure dans la composition du service en cours d'exécution et de répondre ainsi aux nouvelles fonctionnalités, soit d'effectuer des modifications dans la composition courante du service, en omettant certains composants ou en assemblant autrement les composants si nécessaire (voir figure 10.3).

Le Mécanisme de Changement de Composants change alors la composition du service et les interactions entre les composants impliqués. Selon la décision du Gestionnaire d'Adaptabilité, plusieurs stratégies d'adaptation peuvent être effectuées :

1. Permutation des versions d'un composant pour activer la plus appropriée (c.-à.-d. la version que le service en cours d'exécution exige pour être adapté aux nouveaux changements). Chaque composant peut posséder plusieurs versions qui toutes, effectuent les mêmes fonctionnalités mais chacune est destinée à s'exécuter dans un contexte particulier.
2. Intégration de nouveaux composants.
3. Suppression de composants existants.

Toute combinaison de ces différentes stratégies peut également se faire pour adapter dynamiquement le service.

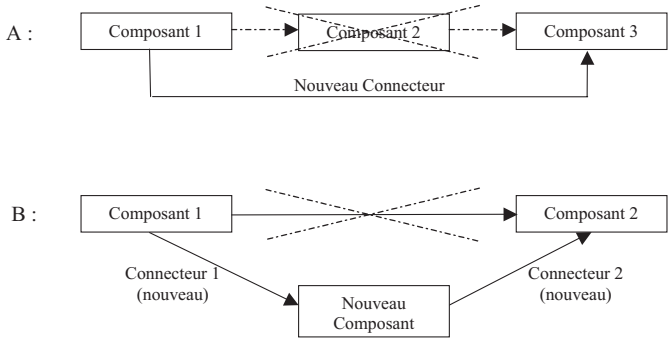


FIG. 10.3 – Exemple de composition de service

10.4 Scénario d'adaptation dynamique de services

Dans ce qui suit, nous illustrons grâce à un scénario, les différentes étapes d'adaptabilité dynamique basée sur la composabilité d'un service.

Dans ce scénario, le terminal mobile exécute un service qui consomme beaucoup de ressources. Une des ressources critiques qui nécessitent une adaptation continue du service est la charge de la batterie du terminal. Les étapes suivantes correspondent aux numéros indiqués sur la figure 10.4 et qui tracent le processus d'adaptation d'un service suite à une baisse de ressources sur le terminal :

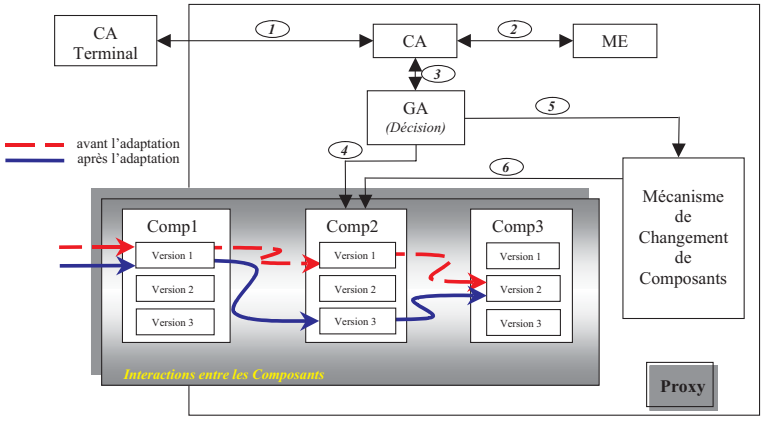


FIG. 10.4 – Scénario d'adaptation dynamique de services

1. Une baisse importante de la charge de la batterie du terminal est détectée par une sonde et transmise au Contrôleur d'Adaptabilité (CA sur le terminal mobile). Ce der-

nier détecte que le service doit être recomposé et ne peut donc pas être adapté sur le terminal mobile. Il envoie alors une requête d'adaptation au Contrôleur d'Adaptabilité (CA) du Serveur Proxy.

2. Les informations sur les ressources réseau sont ensuite fournies par le Moniteur d'Adaptabilité (MA) au Contrôleur d'Adaptabilité (CA) sur le Serveur Proxy.
3. Selon les informations contextuelles récupérées du terminal et du réseau, le Contrôleur d'Adaptabilité (CA) décide si le service doit être adapté ou pas et envoie une notification au Gestionnaire d'Adaptabilité (GA).
4. Le Gestionnaire d'Adaptabilité (GA) décide quelles sont les modifications à effectuer sur le service en cours d'exécution afin d'être adapté.
5. Le Gestionnaire d'Adaptabilité (GA) envoie au Mécanisme de Changement de Composants (MCC) les modifications à effectuer sur le service. Le MCC décide alors d'activer des nouveaux composants (dans notre scénario, il s'agit de la *version 3* du *component 2* : Comp2/Version3) et de désactiver ou de remplacer d'autres composants (dans notre scénario, il s'agit de la *version 1* du *component 2* : Comp2/Version1).
6. Le Mécanisme de Changement de Composants (MCC) change alors la composition du service en modifiant les versions du *composant 2* (la *Version1* par la *Version3*). La recherche et la récupération des nouveaux composants peuvent être effectuées en utilisant un des protocoles de découverte de services présentés dans le chapitre 5.

10.5 Travaux similaires et conclusion

Dans ce chapitre, nous avons proposé une extension de la plate-forme CASP afin de supporter la fourniture de services dynamiquement adaptables et composables. Plusieurs architectures basées sur la composabilité, telles que ICARIS [Men00], EVOLVE [SC00] et YASMIN [Der97], ont été proposées. Ces architectures ont étudié la composition dynamique des services. Elles se sont principalement intéressées à la façon dont les composants peuvent être assemblés en temps réel pour former des services composés. Leur objectif était donc de trouver et de concevoir des moyens appropriés pour le stockage des composants, des méthodes efficaces pour le choix et la recherche des composants et enfin, des techniques pour la composition et l'assemblage de ces composants en temps réel.
Ces travaux couvrent une partie seulement de notre travail puisque l'architecture CASP propose d'utiliser la composition dynamique des services pour assurer leur adaptabilité au contexte. Dans ce cadre, d'autres travaux plus récents ont été proposés pour résoudre la problématique de l'adaptation dynamique des applications dans les environnements mobiles en se basant sur des techniques de composition. Parmi ces travaux, nous pouvons citer les projets suivants : BASE [BS03b], ADAPT[ADA], ACEEL [CA03], Aura [CGS+02], CESURE [RPPM00b], etc.
En se basant sur les résultats des travaux actuellements disponibles et en les adaptant à notre problématique, l'implémentation de l'extension de CASP présentée dans ce chapitre, pourrait sans doute être un point de départ pour résoudre de manière complète et intégrale

la problématique très complexe de la découverte et la fourniture de services adaptatifs dans les environnements mobiles.

Chapitre 11

Conclusions et perspectives

Cette thèse traite de la découverte et la fourniture de services adaptatifs dans les environnements mobiles. Elle a pour but de proposer une architecture permettant aux usagers mobiles de découvrir, de télécharger et d'éxecuter des services sensibles au contexte de leur environnement. Les services proposés sont adaptés aux préférences de l'usager, aux capacités de son terminal, à la localisation de l'usager, et enfin, aux ressources réseau disponibles.

Les contributions de cette thèse sont les suivantes :

Analyse de l'existant.

Nous avons commencé cette thèse par l'identification des besoins imposés par la fourniture de services dans les environnements mobiles. Ces besoins sont essentiellement liés, d'une part, à la mobilité et à l'hétérogénéité des environnements mobiles, et d'autre part, à la personnalisation des services pour le compte de l'usager. Trois besoins ont été étudiés en détail le long de ce mémoire. Il s'agit de la sensibilité au contexte, de la découverte de services sensibles au contexte et de l'adaptabilité des services au contexte de leur exécution. Ensuite, nous avons présenté un panorama des travaux de recherche, de standardisation et d'industrie sur la fourniture de services. Nous avons discuté des avantages et des limites de ces travaux et nous avons montré en quoi ces solutions répondent ou non aux besoins que nous avons soulevés dans cette thèse. Le constat est que la plupart de ces travaux traitent partiellement l'ensemble des besoins nécessaires à la fourniture adaptative des services dans les environnements mobiles.

Étude de l'applicabilité des agents mobiles pour la fourniture de services dans les environnements mobiles.

Pour étudier l'intérêt des agents mobiles pour la fourniture de services, nous avons comparé les agents mobiles au modèle Java RMI. Les résultats de cette étude montrent que la solution à base d'agents mobiles s'avère intéressante, en particulier, dans les applications qui entraînent la consultation de plusieurs services successifs sur le réseau (ex : visite de plusieurs prestataires de services dans le cadre d'une découverte de services). Malgré leur

intérêt, les agents mobiles n'ont pas été utilisés dans le développement de notre solution pour des raisons liées à la non-interopérabilité et la lourdeur des plates-formes agent mobile actuellement disponibles.

Présentation d'une plate-forme de fourniture de services sensibles au contexte.

Nous avons proposé la plate-forme CASP de fourniture de services pour environnements mobiles qui répond aux besoins étudiés dans cette thèse. Elle est basée sur les concepts des futures générations de réseaux mobiles tels que VHE, la sensibilité au contexte, la mobilité, la personnalisation, l'adaptabilité, etc. La sensibilité au contexte est utilisée dans CASP dans la phase de découverte de services et dans la phase d'exécution du service sur le terminal mobile.

Proposition d'un mécanisme de découverte de services sensibles au contexte.

Nous avons proposé dans CASP un mécanisme de découverte de services qui permet aux fournisseurs de services de déployer et d'introduire facilement leurs services ; et aux usagers mobiles de personnaliser la fourniture de services et de l'adapter à leurs exigences. Ce mécanisme, mis en œuvre par un composant médiateur entre l'usager mobile et les fournisseurs de services, permet de ne proposer à l'usager que les services qui sont adaptés à ses préférences, aux capacités de son terminal et à sa localisation. La liste des services proposés à l'usager contient d'autres informations qui permettent de décrire suffisamment le service (description des fonctionnalités offertes par le service, tarifs indicatifs du service, etc.).
L'implémentation de notre mécanisme de découverte de services est basée sur l'utilisation des profils et d'outils standard tels que XML et CC/PP (Composite Capabilities/Preference Profiles).

Proposition d'une solution architecturale pour l'adaptabilité des services.

CASP permet également l'adaptation des services, une fois découverts par le terminal, au contexte de leur exécution (besoins des usagers, caractéristiques techniques du terminal, localisation du terminal et ressources du réseau). Cette adaptation est réalisée au moment de l'exécution du service sur le terminal mobile. La solution proposée est basée sur l'utilisation de serveurs intermédiaires (Serveur Proxy). Un service multimédia a été développé pour valider les concepts proposés dans cette thèse.

Proposition d'une solution pour la composition de services.

Afin d'assurer l'adaptabilité dynamique des services, un même service (qu'il soit multimédia ou pas) doit supporter plusieurs configurations possibles. Chaque configuration correspond à un état particulier du contexte d'exécution. Pour ce faire, nous avons proposé, pour la conception des services, une approche à base de composants qui offre une vue modulaire des services. Chaque service est constitué d'un ensemble de composants interconnectés, pouvant être supervisés et reconfigurés si nécessaire. Un composant est une unité autonome d'encapsulation des fonctions qui offre une interface d'accès à son traite-

ment.
Cette approche de conception de services modulaire permet de prendre en compte les différents contextes d'exécution possibles, en permettant de composer dynamiquement à chaque fois, les composants applicatifs les mieux adaptés au contexte d'exécution.

Cette solution n'a pas été implémentée dans le cadre de notre thèse mais elle est proposée comme perspective de notre travail de recherche.

Les perspectives de nos travaux sont nombreuses. Les plus importantes concernent la gestion de la sensibilité au contexte, la gestion de l'adaptabilité dynamique et l'adaptabilité des services basée sur la composabilité.

Dans le premier objectif visé, il s'agit d'enrichir le mécanisme de gestion de la sensibilité au contexte dans CASP, en définissant d'autres paramètres qui font partie du contexte et qui peuvent être importants dans le cadre de la fourniture de services (la connectivité réseaux, le niveau de batterie du terminal, la proximité d'autres utilisateurs, les informations d'historique d'activité, etc.).
La sensibilité au contexte suppose tout d'abord, le choix d'un modèle de contexte ainsi qu'un ensemble de dispositifs capteurs nécessaires pour acquérir les informations du contexte. Il faut ensuite disposer d'un environnement d'exécution qui permette d'utiliser les informations contextuelles pour adapter le traitement au contexte.
Nous avons proposé dans cette thèse, un environnement d'exécution qui permet d'utiliser des informations contextuelles (capacités du terminal, localisation du terminal, préférences de l'utilisateurs, etc.) pour adapter les services découverts et les services en cours d'exécution. Les informations contextuelles utilisées se limitent à des informations textuelles qui ne sont pas forcément introduites, de manière automatique et dynamique, par le système. Il serait donc intéressant de pouvoir utiliser des sondes et des capteurs pour contrôler ces informations contextuelles et pour détecter automatiquement et en temps réel, les changements dynamiques du contexte [GSC01].

Le second objectif concerne l'adaptabilité dynamique des services multimédia qui doit être étudié de manière plus profonde. La solution proposée dans le chapitre 9, n'a été évaluée que sur quelques exemples, et en utilisant un prototype très simplifié. Il serait donc utile de compléter ce prototype et ces évaluations, en intégrant un mécanisme de validation qui permet de vérifier et de valider l'adaptation d'un service [NSN+97]. En effet, il ne suffit pas d'adapter le service au contexte et d'appliquer un algorithme d'adaptation, mais il faut être sûr que l'algorithme appliqué répond bien aux besoins du contexte.
En outre, il serait intéressant d'étudier la stabilité du système lorsque le contexte est très fluctuant. Dans ce cas, le système ne doit pas passer son temps à réaliser des adaptations de manière continue en dépit de sa fonction initiale qui est la fourniture des services aux utilisateurs mobiles avec la meilleur qualité de service possible. La stabililité doit donc être une priorité dans le système.

Le dernier objectif, l'adaptabilité dynamique basée sur la composabilité des services, nous semble aussi une voie intéressante à explorer. Nous proposons donc pour le développement de services une approche à base de composants qui offre une vue modulaire du

service [CA03, RPPM00b].
Cette approche de conception de services modulaires, permet d'une part, de composer n'importe quel type de service au moment de sa sélection et de son exécution avec un ensemble de composants adaptés au contexte, et d'autre part, de faire coopérer différents composants. Une réalisation monolithique des services ne permet pas de prendre en compte tous les contextes d'exécution possibles et il n'est pas réaliste de générer une version de service pour chacun de ces contextes.
L'extension présentée à la fin de cette thèse, pourrait sans doute être un point de départ intéressant pour résoudre la problématique de l'adaptabilité dynamique des services.

Enfin, d'autres points, partiellement traités dans notre thèse, devraient être étudiés de manière plus détaillée. C'est notamment le cas de nombreux problèmes concernant la description et le découverte des services. Dans cette thèse, nous avons défini des outils qui permettent à des clients de découvrir des services sensibles au contexte. L'utilisation des standards comme CC/PP et MExE, d'une part, et celle du langage XML, d'autre part, constituent les deux caractéristiques fondamentales de notre proposition. Par conséquent, les différentes descriptions des services, dans le cadre de la découverte de services, peuvent être facilement traitées par tous les composants de CASP (terminal mobile, médiateur de services et fournisseurs de services). Cependant, le consensus sur la mécanique des interactions (format des messages, types des données et protocoles d'échanges) n'est pas suffisant en soi pour permettre une découverte de services sensibles au contexte. L'absence d'une sémantique explicite de descriptions de services limite les possibilités d'automatisation d'une découverte sensible au contexte. Pour pallier cette limitation, il y a clairement un besoin de langages de descriptions des services qui permettent de conférer une signification explicite et non ambiguë aux descriptions des services. Dans ce contexte, la notion d'*ontologie* en tant que conceptualisation formelle et consensuelle des services, peut jouer un rôle important pour associer une sémantique formelle à la description d'un service [CFJ03, Ous03].
L'utilisation des ontologies pour la description des services pourait être une bonne direction pour des recherches futures dans le domaine de la découverte et la fourniture de services dans les environnements mobiles.

Bibliographie

[3GP] 3GPP, Third Generation Partnership Project home page. http ://www.3gpp.org.

[3GP01] 3GPP. USIM/SIM Application Toolkit (USAT/SAT) ; Service description ; Stage 1 (Release 5), June 2001.

[ACT] ACTS home page. http ://www.cordis.lu/acts/.

[ADA] ADAPT : Middleware Technologies for Adaptive an Composable Distributed Components. Homepage at : http ://adapt.ls.fi.upm.es/adapt.htm.

[AGL] Aglets Agent Platform home page. http ://www.trl.ibm.com/aglets.

[AIB94] Arup Acharya, T. Imielinski, and B. R. Badrinath. DATAMAN project : Towards a mosaic-like location dependant information service for mobile clients. Technical Report TR-320, 1994.

[ALDPH03] Slim Ben Atallah, Oussama Layaida, Noel De-Palma, and Daniel Hagimont. Dynamic Configuration of Multimedia Applications. In *In Proceedings of the 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services(MMNS'03)*, Belfast, North Irland, Spetember 6-10 2003.

[AMZ95] Elan Amir, Steve McCanne, and Hui Zhang. An application level video gateway. *In Proc. ACM Multimedia '95, San Francisco, CA*, 1995.

[arc02] Etude technique, économique et réglementaire de l'évolution vers les réseaux de nouvelle génération (NGN, Next Generation Networks). Technical report, Etude résalisée par le cabinet Arcome pour le compte de l'Autorité de Régulation des Télécommunications (ART), Septembre 2002.

[AS99] F. André and M.T. Segarra. On Building a File System for Mobile Environments Using Generic Services. In *12th International Conference on Parallel and Distributed computing Systems (PDCS99)*, Fort Lauderdale, Florida, USA, August 1999.

[BBCM99] C. Bäumer, M. Breugst, S. Choy, and T. Magedanz. Grasshopper- A Universal Agent Platform based on OMG MASIF and FIPA Standrads. In *In mata'99, First International Workshop on Mobile Agents for Telecommunication Applications*, pages 66–81, 1999.

[BBX97] P.J Brown, J.D. Bovey, and C. Xian. Context-aware applications : from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5) :58–64, October 1997.

[BCS00] P. Bellavista, A. Corradi, and C. Stefanelli. A mobile agent infrastructure for terminal, user and resource mobility. In *Network Operations and Management Symposium, 2000 (NOMS 2000)*, pages 877–890. IEEE/IFIP, 10-14 April 2000.

[BCS02] P. Bellavista, A. Corradi, and C. Stefanelli. The ubiquitous provisioning of internet services to portable devices. *Pervasive Computing, IEEE*, 1(3) :81 –87, July-Sept 2002.

[BEK+00] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman AMD, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winner. Simple Object Access Protocol (SOAP) version 1.1. W3C Note. Technical report, May 2000.

[Bel01] E. Belloni. A Multi-paradigm Approach for Mobile Agents Development. *Journal of Computer Science and Technology*, 1(4) :12–17, March 2001. Special Issue on Computer Science Research.

[Ber00] G. Bernard. Apport des agents mobiles à léxécution répartie. In *ISYPAR'00, 4^{me} Ecole d'Informatique des Systèmes Parallèles et Rèpartis*, Toulouse, France, Février 2000.

[BGI99] James Beck, Alain Gefflaut, and Nayeem Islam. Moca : A service framework for mobile computing devices. In *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access, Seattle, WA, USA*, pages 62–68. ACM, August 1999.

[BGP97] Mario Baldi, Silvano Gai, and Gian Pietro Picco. Exploiting Code Mobility in Decentralized and Flexible Network Management. In *Proceedings of the First International Workshop on Mobile Agents*, pages 13–26, Berlin, Germany, 1997.

[BKC+98] E.A. Brewer, R.H. Katz, Y. Chawathe, S.D. Gribble, T. Hodes, G. Nguyen, M. Stemm, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V.N. Padmanabhan, and S. Seshan. A network architecture for heterogeneous mobile computing. *IEEE Personal Communications*, 5(5) :8–24, October 1998.

[BN83] A. D. Birrell and B. J. Nelson. Implementing Remote Procedure Calls. In *Proceedings of the ACM Symposium on Operating System Principles*, page 3, Bretton Woods, NH, 1983. Association for Computing Machinery.

[Bob01] J.F. Bobier. Microsoft .NET : Architectures et Services. Master's thesis, Département Informatique et Réseaux, École Nationale Supérieure de Télécommunications (ENST Paris), Juillet 2001.

[Bou99] N. Boukhatem. Les agents mobiles et applications. In *DNAC'99, De Nouvelles Architectures pour les Communications, La mobilié dans les réseaux*, Paris, France, Décembre 1999.

[BS01] Jennifer Bray and Charles F Struman. *Bluetooth Connect Without Cables.* 2001.

[BS03a] M. Badra and A. Serhrouchni. A New Secure Session Exchange Key Protocol for Wireless Communications. In *The 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC2003*, pages 2765–2769, China, September 7-10 2003.

[BS03b] Christian Becker and Gregor Schiele. Middleware and application adaptation requirements and their support in pervasive computing. In *Proceedings of 3rd International Workshop on Distributed Auto-adaptive and Reconfigurable Systems (DARES)*, pages 98–103, Providence, USA, May 19-22 2003.

[BSAK95] H. Balakrishnan, S. Seshan, E. Amir, and R.H. Katz. Improving TCP/IP Performance over Wireless Network. In *Proceedings of Mobicom*, November 1995.

[CA03] D. Chefrour and F. André. Auto-adaptation de composants ACEEL coopérants. In *Troisième Conférence Française sur les Systèmes d'Exploitation(CFSE'3)*, La Colle sur Loup, France, October 2003.

[CAM] ACTS CAMELEON project home page. http ://www.comnets.rwth-aachen.de/~cameleon/.

[CAM01] 3GPP TS 22.078 v5.5.0. Customised Applications for Mobile network Enhanced Logic (CAMEL). Service description, Stage 1, December 2001.

[CCMW01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language 1.1, W3C Note. Technical report, March 2001.

[CDK+02] F. Curbera, M. Duftler, R. Khalaf, N. Mukhi, W. Nagy, and S. Weerawarana. Unraveling the Web Services Web, An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 6(2) :86–93, March-April 2002.

[CFJ03] Harry Chen, Tim Finin, and Anupam Joshi. An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, November 2003.

[CGS+02] Shang-Wen Cheng, David Garlan, Bradley Schmerl, João Pedro Sousa, Bridget Spitznagel, Peter Steenkiste, and Ningning Hu. Software Architecture-Based Adaptation for Pervasive Systems. *Lecture Notes in Computer Science*, 2299, 2002.

[CK99] P. Couderc and A.M Kermarrec. Improving level of service for mobile users using context-awareness. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, pages 24–33, 19-22 October 1999.

[CL99] B. Christensson and O. Larsson. Universal Plug and Play Connects Smart Devices. WinHEC 99, 1999.

[CLD] Sun Microsystems Computer Corporation. CLDC Specifications.

[CLI] CLIMATE ACTS Cluster home page.
http ://www.fokus.gmd.de/research/cc/ecco/climate/.

[Con99] Salutation Consortium. Salutation Architecture : Overview. White Paper, 1999. http ://www.salutation.org/whitepaper.

[Con01] Bluetooth Consortium. Bluetooth Protocol Architecture. Bluetooth White Paper, August 2001. http ://www.bluetooth.com.

[CPV97] Antonio Carzaniga, Gian Pietro Picco, and Giovanni Vigna. Designing Distributed Applications with a Mobile Code Paradigm. In *Proceedings of the 19th International Conference on Software Engineering*, Boston, MA, 1997.

[CZH+99] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz. An architecture for a secure service discovery. In *Mobile Computing and Networking*, pages 24–35, 1999.

[CZMN04] Jinwei Cao, Dongsong Zhang, K. M. McNeill, and Jay. F. Nunamaker. An overview of network-aware applications for mobile multimedia delivery. In *37th Hawaii International Conference on System Sciences (HICSS-37 2004)*, 5-8 January 2004.

[Dav01] Pierre-Charles David. Une infrastructure pour middleware adaptable. Rapport de DEA, École des Mines de Nantes, Université de Nantes, Septembre 2001.

[Dem94] Alan J. Demers. Research issues in ubiquitous computing. In *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*, pages 2–8. ACM Press, 1994.

[Der97] Luca Deri. *A Component-based Architecture for Open, Independently Extensible Distributed Systems*. PhD thesis, 1997.

[Dey00] Anind K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

[DH98] S. Deering and R. Hinde. Internet protocol, version 6 (IPv6), Specification. IETF RFC, No. 2460, December 1998.

[Dro99] Ralph Droms. Automated configuration of TCP/IP with DHCP. *IEEE Internet Computing*, 3(4) :45–53, July 1999.

[FBD01] O. Fouial, N. Boukhatem, and I. Demeure. Mobile agents : a suiatable support for service provision in mobile computing environments. In *OPODIS'2001, 5th International Conference On Principles Of DIstributed Systems*, Manzanillo, Mexico, December 2001.

[FBD02a] O. Fouial, N. Boukhatem, and I. Demeure. Mobile agents : a suitable support for service provision in mobile computing environments. *Studia Informatica Universalis*, pages 33–54, 2002. Special Issue : OPODIS'2001.

[FBD02b] O. Fouial, N. Boukhatem, and I. Demeure. Plate-forme de fourniture de services pour les environnements mobiles. In *MCSEAI 2002, Seventh Maghrebian Conference On Computer Sciences*, Annaba, Algérie, May 2002.

[FBD02c] O. Fouial, N. Boukhatem, and I. Demeure. Service provision based on mobile agents in mobile computing environments. In *MWCN 2002, Fourth Conference on Mobile and Wireless Communications Networks*, Stockholm, Sweden, September 2002.

[Fer02] Salim Ferraz. An adaptive service provision architecture for mobile computing environments. Master's thesis, Ecole Nationale Supérieure des Télécommunications, Novembre 2002.

[FGB00] P. Farjami, C. Gorg, and F. Bell. Advanced service provisioning based on mobile agents. *Computer Communications, Special Issue : Mobile Software Agents for Telecommunication Applications*, 23 :754–760, April 2000.

[FGBA96] Armando Fox, Steven D. Gribble, Eric A. Brewer, and Elan Amir. Adapting to network and client variability via on-demand dynamic distillation. In *Proceedings of the seventh international conference on Architectural support for programming languages and operating systems*, pages 160–170, 1996.

[FHB00] O. Fouial, N. Houssos, and N. Boukhatem. Software Downloading Solutions for Mobile Value-Added Service Provision. In *IST Mobile Communications Summit 2000*, Galway, Ireland, October 2000.

[FIP] The Foundation for Intelligent Physical Agents (FIPA) web site. http ://www.fipa.org.

[FKN98] C. Freytag, C. Kumpf, and L. Neumann. Utilisation of User and Device Profiles for Adaptive WWW Access. In *Proceedings of the Fifth International Workshop on Mobile Multimedia Communication*, pages 283–294, Berlin, October 1998.

[FLP98] T. Finin, Y. Labrou, and Yun Peng. Mobile Agents Can Benefit from Standards Efforts on Interagent Communication. *IEEE Communications Magazine*, 36(7) :50–56, July 1998.

[For98] WAP Forum. Wireless Application Protocol : Wireless Markup Language Specification, 1998.

[GBF+03] V. Gurbani, A. Brusilovsky, I. Faynberg, H.L. Lu, M. Unmehopa, K. Vemuri, and J. Gato. The SPIRITS Protocol. IETF Internet-Draf, April 2003. http ://search.ietf.org/internet-drafts/draftgurbani-spirits-protocol-03.txt.

[GCPLA99] Y. Goland, T. Cai, P.and Y. Gu P. Leach, and S. Albright. Simple Service Discovery Protocol. IETF Draft, Draft-cai-ssdp-v1-03, 1999.

[Gen95] October General. Telescript language reference, 1995.

[GG96] Georges Gardarin and Olivier Gardarin. *Le Client Serveur*. Eyrolles, 1996.

[GKN+96] Robert S. Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile agents for mobile computing. Technical Report TR96-285, 1996.

[GPa] Groupe PARLAY. Groupe de Standardisation. http ://www.parlay.org.

[GPK99] E. Guttman, C.E. Perkins, and J. Kempf. Service Templates and Service : Scheme. Internet RFC 2609, June 1999.

[GPS] International GPS Service. http ://igscb.gnutella.com.

[GRA] Grasshopper Agent Platform. http ://www.grasshopper.de.

[Gra95] R. S. Gray. Agent Tcl : A transportable agent system. In *CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95)*, December 1995.

[Gro97] Object Management Group. CORBA Services, Trader Service (Chapter 16). 97-12-23. http ://www.omg.org, 1997.

[GSB+01] S. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, and R. Neyama. *Building Web Services with Java : Making Sense of XML, SOAP, WSDL, and UDDI.* Sams, 2001.

[GSC01] D. Garlan, B. R. Schmerl, and J. Chang. Using Gauges for Architecture-Based Monitoring and Adaptation. In *Working Conference on Complex and Dynamic System Architecture*, Brisbane, Australia, December 2001.

[Gur03] Levent Gurgen. Découverte de données dans les réseaux mobiles. Mémoire de DEA D'informatique, Université Joseph Fourier, Juin 2003.

[Gut99] Erik Guttman. Service Location Protocol : Automatic Discovery of IP Network Services. *IEEE Internet Computing*, 3(4) :71–80, 1999.

[Hac] Dictionnaire Hachette universel francophone en ligne. http ://www.francophonie.hachette-livre.fr.

[HAK01] H. Harroud, M. Ahmed, and A. Karmouch. Agent-based Personalized Services in a Mobile Computing Environment. In *2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'01)*, pages 728–731, Victoria Canada, August 26-28 2001.

[HBL+98] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas. Dynamic adaptation in an image transcoding proxy for mobile Web browsing. *IEEE Personal Communications*, 5(6) :8–17, December 1998.

[HI00] D. Hagimont and L. Ismail. Agents mobiles et client/serveur : évaluation de performance et comparaison. *Technique et Science Informatiques (TSI)*, 19(9), 2000.

[HL96a] Barron C. Housel and David B. Lindquist. WebExpress : A system for optimizing Web browsing in a wireless environment. In *Second Annual International Conference on Mobile Computing and Networking*, pages 108–116. ACM, November 1996.

[HL96b] Barron C. Housel and David B. Lindquist. WebExpress : a system for optimizing Web browsing in a wireless environment. In *Proceedings of the 2nd annual international conference on Mobile computing and networking (MOBICOM)*, pages 108–116, November 1996.

[IET] IETF : Internet Engeineering Task Force web site. http ://www.ietf.org/.

[IH99] Leila Ismail and Daniel Hagimont. A Performance Evaluation of the Mobile Agent Paradigm. In *Conference on Object-Oriented*, pages 306–313, 1999.

[JAU00] R. Jain, F. Anjum, and A. Umar. A comparaison of mobile agent and client-server paradigms for information retrieval tasks in virtual entreperises. In *Academia and Industry Working Conference on*, pages 209–213, 2000.

[JAV] Sun Microsystems Computer Corporation. JavaPhone Specifications.

[JHE99] Jin Jing, Abdelsalam Sumi Helal, and Ahmed Elmagarmid. Client-server computing in mobile environments. *ACM Computing Surveys*, 31(2) :117–157, 1999.

[JJ02] Jorma Jormakka and Henryka Jormakka. State of the Art of Service Creation Technologies in IP and Mobile Environments. In *Communication Systems : The State of the Art (IFIP World Computer Congress)*, pages 147–166, 2002.

[JK98] A. Joshi and A. Krishna. Mobile Access To WEB Resources (Guest Editorial). *IEEE Personal Communications*, 5(5) :6–7, October 1998.

[JMF] Java Media Framework Home Page. java.sun.com/products/java-media/jmf/index.html.

[JPK99] D. G. Jung, K. J. Paek, and T. Y. Kim. Design of MOBILE MOM : Message oriented middleware service for mobile computing. In *International Workshops on Parallel Processing*, pages 434–439, 21-24 Sept 1999.

[JTK96] Anthony D. Joseph, Joshua A. Tauber, and M. Frans Kaashoek. Building reliable mobile-aware applications using the rover toolkit. In *Second ACM International Conference on Mobile Computing and Networking (MobiCom'96)*, 1996.

[JVP+99] H. Jormakka, K. Valtari, D. Prevedourou, A. Kind, and K. Raatikainen. Agent-based TINA Access Session Supporting Retailer Selection in Personal Mobility Context. In *TINA 99, Telecommunication Information Networking Architecture Conference*, Hawaii, April 1999.

[KAFFD03] Z. Kazi-Aoul, S. Ferraz, O. Fouial, and I. Demeure. CAAS : an architecture for component-based adaptable service provision. In *2nd ANWIRE Workshop on Wireless, Mobile and Always Best Connected*, Mykonos, Greece, September 2003.

[KG99] David Kotz and Robert S. Gray. Mobile code : The future of the Internet. In *Proceedings of the Workshop "Mobile Agents in the Context of Competition and Cooperation (MAC3)" at Autonomous Agents '99*, pages 6–12, May 1999.

[KPM+98] B. Kreller, A.S.B. Park, J. Meggers, G. Forsgren, E. Kovacs, and M. Rosinus. UMTS : a middleware architecture and mobile API approach. *Personal Communications, IEEE*, 5(2) :32–38, April 1998.

[KPMM01] Jari Korva, Johan Plomp, Petri Määttä, and Maija Metso. On-line service adaptation for mobile and fixed terminal devices. In *Proceedings of MDM : Mobile Data Management, Second International Conference*, Lecture Notes in Computer Science, Hong Kong, China, January 2001. Springer.

[Kra88] S. Krakowiak. *Principles of operating systems.* MIT Press, 1988.

[KRA94] M. Kojo, K. Raatikainen, and T. Alanko. Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network. Technical report, Helsinki, Finland, 1994.

[Kre01] H Kreger. Web Services Conceptual Architecture. Technical report, IBM, WCSA 1.0, 2001.

[KRS99] E. Kovacs, K. Rohrle, and B. Schiemann. Adaptive mobile access to context-aware services. In *Agent Systems and Applications, 1999 and Third International Symposium on Mobile Agents. Proceedings. First International Symposium on*, pages 190–201, 1999.

[Led01] Tomas Ledoux. État de l'art sur l'adaptabilité. Délivrable D1.1, Projet RNTL ARCAD, Décembre 2001.

[Les00] P. Lescuyer. *UMTS : Les origines, l'architecture et la norme*. Paris, 2000.

[LKP02] M.M. Lankhorst, H.V. Kranenburg, and A.J.H Peddemors. Enabling technology for personalizing mobile services. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 1464–1471, January 2002.

[LOT98] J.C.S. Lui, K.Y. Oldfield, and T. S. Tam. NFS/M : An Open Platform Mobile File System. In *International Conference on Distributed Computing Systems*, pages 488–495, 1998.

[LYBP02a] A. Liotta, A. Yew, C. Bohoris, and G. Pavlou. Delivering Service Adaptation with 3G Technology. In *13th IFIP/IEEE International Workshop on Distributed Systems : Operations and Management (DSOM'2002)*, pages 108–120, Montreal, Canada, October 2002. Springer.

[LYBP02b] Antonio Liotta, Alvin Yew, Chris Bohoris, and George Pavlou. Supporting adaptation-aware services through the virtual home environment middleware. In *Proceeding of HP-OVUA, The Hewlett-Packard Openview University Association Plenary Workshop 2002*, June 11-13 2002.

[Mar98] V. Maranzova. Qualité de service et adaptabilité. Rapport de Magistère Informatique, 3^{e}année, Université Joseph Fourier, Septembre 1998.

[MAS] MASIF, Agent Work Group. http ://www.fipa.org.

[Men00] D. Mennie. *An Architecture to Support Dynamic Composition of Service Components*. PhD thesis, University of Carleton, 2000.

[MEX] 3G TS 23.057 : "3rd Generation Partnership Project ; Technical Specification Group Terminals ; Mobile Station Application Execution Environment (MExE) ; Functional description ; Stage 2 (3G TS 23.057 version 3.0.0)".

[MG97] Michael Meyer and Eckhard Geulen. The Onthemove Concept for Mobile Middleware, 1997. http ://citeseer.nj.nec.com/meyer97onthemove.html.

[Mic99] Sun Microsystems. Jini Architecture Overview. Technical White Paper. http ://www.sun.com/jini/, 1999.

[Mit02] Keith Mitchell. *Supporting the Development of Mobile Context-Aware Computing*. PhD thesis, Lancaster University, January 2002.

[MOB] MOBIVAS : Downloadable MOBile Value-Added Services through Software Radio and Switching Integrated Platforms. http ://mobivas.cnl.di.uoa.gr. Europeen IST Project.

[MOB00] Definition, Identification and Requirements of Downloadable VAS. Deliverable D-2.1.1, MOBIVAS Project (IST-1999-10206), April 2000.

[Moc87] Paul Mockapetris. Domain names, implementation and specification. Available on the World Wide Web at ftp ://ftp.internic.net/rfc/rfc1035.txt, November 1987.

[mon] ACTS MONTAGE project home page. http ://montage.ccrle.nec.de/.

[Mou99] F. Le Mouël. Amélioration du niveau de service par la distribution adaptative d'applications dans un environnement mobile. In *Journée des Jeunes Chercheurs en Systèmes (JCS'99), dans les actes de la 1re Conférence Française sur les Systèmes d'Exploitation (CFSE'1)*, pages 229–232, Rennes, France, 8-11 Juin 1999.

[Mou03] Frédéric Le Mouël. *Environnement adaptatif d'exécution distribuée d'applications dans un contexte mobile*. PhD thesis, Université de Rennes 1, IRISA, Décembre 2003.

[MSL99] Rakesh Mohan, John R. Smith, and Chung-Sheng Li. Adapting multimedia internet content for universal access. *IEEE Transactions on Multimedia*, 1(1) :104–114, 1999.

[Ngu00] V. Nguyen. Mobile computing and disconnected operation : A survey of recent advances. http ://www.cis.ohio-state.edu/~jain/cis788-95/mobile_comp/, 2000.

[NHO00] M. Nilsson, J. Hjelmcand, and H. Ohto. Composite Capabilities/Preference Profiles : Requirements and Architecture. W3C Working Draft, 21 July 2000.

[NIN] The Ninja Project . http ://ninja.cs.berkeley.edu/.

[NSN+97] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. In *Sixteen ACM Symposium on Operating Systems Principles*, pages 276–287, Saint Malo, France, 1997.

[OMGa] Object Management Group(OMG) web site. http ://www.omg.org.

[OMGb] OMG. The Common Object Request Broker : Architecture and Specification (CORBA). Report 91.12.1, The Object Management Group, 1991.

[OSA01] Services and System Aspects, Service Aspects, Stage1 : Service Requirement for the Open Service Access (OSA). Technical Report TS 22.127 v4.1.0, 3rd Generation Partnership Project, Mars 2001.

[Ous03] Oussama Kassem Zein and Yvon Kermarrec. An Approach for Describing User Service Interfaces in Distributed Systems. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA03*, Las Vegas, Nevada, USA, 23-26 June 2003.

[OZO02] Ozone technology inventory. Deliverable D2A. IST-2000-30026. OZONE, May 2002.

[Par00a] PARLAY Group. Parlay API Business Benefits. http ://www.parlay.org. Janvier 2000.

[Par00b] PARLAY Group. Parlay Specification 2.1, Core Specification Document. http ://www.parlay.org. Juin 2000.

[PC00] S. Petrack and L. Conroy. The PINT Service Protocol : Extensions to SIP and SDP for IP Access to Telephone Call Services. RFC 2848, June 2000.

[PER] Sun Microsystems Computer Corporation. PersonalJava Specifications.

[Per96] C.E. Perkins. IP Mobility Support. Request For Comments (RFC) 2002, October 1996.

[Per00] Charles E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2000.

[PG99] C. E. Perkins and E. Guttman. DHCP Options for Service Location Protocol. Internet RFC 2610, June 1999.

[PK98] V. A. Pham and A. Karmouch. Mobile software agents : an overview. *IEEE Communications Magazine*, 36 :26–37, July 1998.

[Pon01] C Ponsioen. Study of customisation techniques in UMTS. Deliverable 2.5, GigaMobile Project (TI/RS/2001/099). Telematica Institut, 2001.

[Pos81] J. Postel. Transmission Control Protocol. Standard (STD) 0007, Request For Comments (RFC) 793, 1 September 1981.

[PPM+00] Marie Claude Pellegrini, Olivier Potonniée, Raphaël Marvie, Sébastien Jean, and Michel Riveill. Cesure : une plate-forme d'applications adaptables et sécurisées pour usagers mobiles. *Calculateurs parallèles, Evolutions des plates-formes orientées objets répartis*, 12(1) :113–120, 2000.

[Pre02] Stephan Preuß. JESA Service Discovery Protocol. In E. Gregori, M. Conti, A. T. Campbell, G. Omidyar, and M. Zukerman, editors, *Proceedings of Networking 2002*, volume 2345 of *LNCS*, pages 1196–1201, Pisa, Italy, May 2002. Springer-Verlag.

[RCCC01] Herman Chung-Hwa Rao, Di-Fa Chang, Yih-Farn Chen, and Ming-Feng Chen. iMobile : a proxy-based platform for mobile services. In *Wireless Mobile Internet*, pages 3–10, 2001.

[RDF] Ressource Description Framework. http ://www.w3.org/RDF/.

[Ric00] Colden G. Richard. Service Advertisement and Discovery : Enabling Universal Device Cooperation . *IEEE Internet Computing*, 4(5) :18–26, Sep/Oct 2000.

[RMI99] Java Remote Method Invocation Specification, December 1999. Revision 1.7.Java 2 SDK, Standard Edition, v 1.3.0.

[Rob00] M. Robert. Discovery and its Discontents : Discovery Protocols for Ubiquitous Computing. Technical Report UIUCDCS-R-99-2132, National Center for Supercomputing Applications, Department of Computer Science, University of Illinois Urbana-Champaign, Urbana, April 2000.

[Rou02] S. Rouvrais. *Utilisation d'agents mobiles pour la construction de services distribués.* Thèse de doctorat en informatique, Université de Rennes I, Juillet 2002.

[RPPM00a] M. Riveill, M. C. Pellegrini, O. Potonniée, and R. Marvi. Adaptabilité des applications pour des usagers mobiles. In *OCM 2000*, Nantes, Mai 2000.

[RPPM00b] Michel Riveill, Marie Claude Pellegrini, Olivier Potonniée, and Raphaël Marvi. Adaptabilité et disponibilité des applications pour des usagers mobiles. In *Objets, Composants et Modèles (OCM'2000)*, Nantes, France, May 2000.

[RQF01a] P. Royard, S. Quesnel, and O. Fouial. Deliverable D-4.4.1 : Implementation of the terminal software, infrastructure and interface. IST 1999-1O206 MOBIVAS, October 2001.

[RQF01b] P. Royard, S. Quesnel, and O. Fouial. Deliverable D-5.1.1 : Small scale pilot 3. IST 1999-1O206 MOBIVAS, December 2001.

[RSA] RSA based Cryptographic Schemes. http ://www.rsasecurity.com.

[Sab03] Nawel Sabri. *Une Architecture à base de Composants CORBA pour des Services Personnalisés.* PhD thesis, Université d'Evry, Val d'Essonne, Juin 2003.

[SAG+93] Bill N. Schilit, Norman Adams, Rich Gold, Michael M. Tso, and Roy Want. The PARCTAB mobile computing system. In *Workshop on Workstation Operating Systems*, pages 34–39, 1993.

[Sat96] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Symposium on Principles of Distributed Computing*, pages 1–7, 1996.

[SB98] M. Stangel and V. Bharghavan. Improving TCP performance in mobile computing environments. *International Conference on Communications 98, Atlanta, GA*, 1998.

[SC00] O. Stiemerling and A. Cremers. The EVOLVE project : Component based tailorability for CSCW applications. AI and Society, 14 :120-141, 2000.

[SC01] N. Subramanian and L. Chung. Software Architecture Adaptability : An NFR Approach. In *International Workshop on Principles of Software Evolution (IWPSE 2001)*, pages 52–61, Vienna, September 2001. ACM Press.

[Sch95] Bill N. Schilit. *A Context-Aware System Architecture for Mobile Distributed Computing.* PhD thesis, Columbia University, 1995.

[SDP99] Bluetooth Specification Part E, Service Discovery Protocol (SDP). http ://www.bluetooth.com, November 1999.

[SMKO90] M. Satyanarayanan, .J. MKistler, P. Kumar, and M. Okasaki. Coda : a highly available file system for a distributed workstation environment. *IEEE Transactions on Computers*, 39(4) :447–459, April 1990.

[SML98] J. Smith, R. Mohan, and C. Li. Transcoding internet content for heterogeneous client devices, 1998.

[SMS00] Short Message Service (SMS). http ://www.mobileSMS.com. September 2000.

[SOA] SOAP web site, http ://www.w3.org/TR/SOAP.

[SPW+02] J. Stewart, L. Pitt, M. Winskel, R. Williams, I. Graham, J. Aguiar, L.M. Correia, B. Hunt, T. Moulsley, F. Paint, S. Svaet, B. Michael, A. Burr, T. G. Eskedal, V. Yin, and C. Stimming. FLOWS Scenarios and Definition of Services. IST FLOWS Project Deliverable D1, European Commission IST Office, Brussels, Belgium, December 2002. http ://www.flows-ist.org/.

[STS03] Jun-Zhao Sun, J. Tenhunen, and J. Sauvola. CME : a Middleware Architecture for Network-Aware Adaptive Applications. In *Proceedings on 14th IEEE Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2003)*, volume 1, pages 839–843, September 7-10 2003.

[SUN] Sun Microsystems web site. http ://java.sun.com/.

[SUN99] SUN. Java Message Service (JMS), 1999.

[SWA] SWARM home page. http ://www.swarm.org/.

[Tho97] Tommy Thorn. Programming Languages for Mobile Code. Technical Report 3134, INRIA, Mars 1997.

[TIN] TINA-C home page. http ://www.tinac.com/.

[UAP99] Wireless Application Group, User Agent Profile Specification. WAP Forum Approved Specification WAP-174, 10 November 1999.

[UDD] UDDI web site, http ://www.uddi.org/about.html.

[VB95] G. M. Voelker and B. N. Bershad. Mobisaic - an information system for a mobile wireless computing environment. Technical Report TR-95-04-01, 1995.

[VHE] 3G TS 23.127 3.0.0 (2000-03). "3^{rd} Generation Partnership Project ; Technical Specification Group Services and System Aspects ; Virtual Home Environment/Open Service Architecture".

[VOY] Voyager Agent Platform home page. http ://www.recursionsw.com/osi.asp.

[Wel99] Matt Welsh. NinjaRMI : A Free Java RMI. http ://www.cs.berkeley.edu/ mdw/proj/ninja/ninjarmi.html, 1999.

[WHK97] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol, version 3, ietf, rfc 2251. Available at http ://www.rfc-editor.org/rfc/rfc2251.txt, December 1997.

[Win01] Terry Winograd. Interaction Spaces for 21st Century Computing. *Human-Computer Interaction in the New Millennium, Addison-Wesley, (in press)*, 2001. John Carroll, Editor.

[WSD] WSDL web site, http ://www.w3.org/TR/wsdl.

[XML98] Extensible Markup Language (XML) 1.0, W3C Recommendation. http ://www.w3.org/TR/1998/REC-xml-19980210, Ferbruary 1998.

[YHF+03] Yan Yan, Yi Huang, Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Ali Kaplan, and Ahmet Topcu. Implementing a Prototype of the Security Framework for Distributed Brokering Systems. In *Proceedings of the 2003 International Conference on Security and Management*, volume I, pages 212–218, 2003.

[ZD97] Bruce Zenel and Dan Duchamp. A general purpose proxy filtering mechanism applied to the mobile environment. *Mobicom'97*, October 1997.

[ZJ00] B. Zhao and A. Joseph. Xset : A lightweight database for internet applications, 2000.

[Zui02] Johan Zuidweg. *Next Generation Intelligent Networks*. Artech House, Boston, MA, 2002.

Annexe A

Gestion des profils dans CASP : exemples de profils utilisés

Dans cette annexe, nous détaillons le Profil Terminal et le Profil Sécurité. Les deux autres profils utilisés dans CASP (Profil Utilisateur et Profil Service) ont été présentés dans le chapitre 4.

A.1 Profil Terminal

Les blocs descriptifs suivants constituent le *Profil Terminal* qui décrit les capacités d'un terminal générique selon les spécifications de MExE. D'autres attributs ont été rajoutés pour répondre aux besoins de la plate-forme CASP.

A.1.1 Plate-forme matérielle

La plate-forme matérielle est une collection de propriétés qui décrit les caractéristiques matérielles du terminal . Elle est définit par les attributs suivants :

- **BitsPerPixel** : Nombre de bits de couleur ou de niveaux de gris par pixel.
- **ColorCapable** : Indique si l'affichage du terminal supporte les couleurs.
- **CPU** : Nom et référence du CPU.
- **ImageCapable** : Indique si le terminal supporte l'affichage d'images.
- **InputCharSet** : Indique les jeux de caractères supportés par le terminal pour les entrées de type texte.
- **Keyboard** : Type du clavier supporté par le terminal.
- **MaxScreenChar** : Taille de la page virtuelle dans laquelle un document est affiché, en unités de caractères.
- **Model** : Numéro du modèle attribué au terminal par son constructeur.
- **OutputCharSet** : Liste des jeux de caractères supportés par le terminal pour la sortie d'affichage.
- **PointingResolution** : Type de résolution supportée par le terminal.
- **ScreenSize** : La taille de l'écran du terminal, en unité de pixels.
- **ScreenSizeChar** : La taille de l'écran du terminal, en unité de caractères.
- **SoftKeysCapable** : Indique si le terminal supporte les touches programmables.
- **SoundOutputCapable** : Indique si le terminal supporte le son en sortie.

- **TextInputCapable** : Indique si le terminal supporte l'entrée de texte alphanumérique.
- **Vendor** : Le nom du fabriquant du terminal.
- **VoiceInputCapable** : Indique si le terminal supporte une entrée de voix.

A.1.2 Plate-forme logicielle

Ce bloc décrit les propriétés de l'environnement logiciel du terminal, telles que le système d'exploitation, les logiciels installés, les codecs audio et vidéo, etc. Ces propriétés sont décrites par les attributs suivants :

- **AcceptDownloadableSoftware** : Indique si l'utilisateur accepte le téléchargement des logiciels sur le terminal ou pas.
- **AudioInputEncoder** : Liste des encodeurs d'entrée supportés par le terminal.
- **DownloadableSoftwareSupport** : Liste des types de contenus exécutables supportés par le terminal.
- **VideoInputEncoder** : Liste des encodeurs d'entrée vidéo supportés par le terminal.
- **JVMVersion** : Indique la version de la machine virtuelle Java installée sur le terminal.
- **JavaPlatform** : Liste des plates-formes Java installées sur le terminal.
- **MexeClassmark** : Indique la classe du terminal (*Classmark* selon les concepts de MExE).
- **MexeAcceptPlatform** : Liste des plate-formes d'exécution présentes sur le terminal.
- **OSName** : Nom du système d'exploitation installé sur le terminal.
- **OSVendor** : Fournisseur du système d'exploitation installé sur le terminal.
- **OSVersion** : Version du système d'exploitation installé sur le terminal.

A.1.3 Caractéristiques du réseau

Les caractéristiques du réseau représentent l'ensemble des informations liées à l'infrastructure réseau disponible. Elles sont décrites par les attributs suivants :

- **CurrentBearerService** : Le support sur lequel la session courante a été initiée.
- **SupportedBearers** : Liste des supports acceptés par le terminal.

A.1.4 Navigateur

Ce bloc décrit le logiciel navigateur disponible sur le terminal. Il est composé des attributs suivants :

- **BrowserName** : Nom du navigateur associé à la session courante.
- **DownloadableBrowserApps** : Liste des contenus exécutables supportés par le navigateur.
- **FramesCapable** : Indique si le navigateur est capable d'afficher les frames HTML.
- **HTMLVersion** : Version du HTML supportée par le navigateur.
- **JavaScriptVersion** : Version du langage JavaScript supportée par le navigateur.
- **PreferenceForFrames** : Indique les préférences de l'utilisateur.
- **TablesCapable** : Indique si le navigateur est capable d'afficher les tables HTML.
- **XhtmlVersion** : Version du XHTML supportée par le navigateur.
- **Xhtmlmodules** : Liste des modules XHTML supportés par le navigateur.

A.2 Profil Sécurité

Ce profile décrit les attributs de sécurité qui peuvent être supportés par le terminal.

```
<?xml version="1.0"?>
        <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:prf="http://www.w3.org/TR/WD-profile-vocabulary#">
         <rdf:Description about="HardwarePlatform">
           <prf:Defaults
            Vendor="Nokia"
            Model="2160"
            Type="PDA"
            ScreenSize="800x600x24"
            CPU="PPC"
            Keyboard="Yes"
            Memory="16mB"
            Bluetooth="YES"
            Speaker="Yes" />
          <prf:Modifications
            Memory="32mB" />
         </rdf:Description>
         <rdf:Description about="SoftwarePlatform">
          <prf:Defaults
           OS="EPOC1.0"
           HTMLVersion="4.0"
           JavaScriptVersion="4.0"
           WAPVersion="1.0"
           WMLScript="1.0" />
          <prf:Modifications
           Sound="Off"
           Images="Off" />
         </rdf:Description>
         <rdf:Description about="EpocEmail1.0">
           <prf:Defaults
           HTMLVersion="4.0" />
         </rdf:Description>
          <rdf:Description about="EpocCalenda r1.0">
          <prf:Defaults
           HTMLVersion="4.0" />
         </rdf:Description>
          <rdf:Description about="UserPreferences">
         <prf:Defaults
              Language="English"/>
         </rdf:Description>
        </rdf:RDF>
```

FIG. A.1 – Exemple de Profil Terminal utilisé dans CASP

- **SecurityAlgo** : Mécanisme de sécurité supporté par le terminal.
- **HashingFunction** : Fonction de hachage supportée par le terminal.
- **ProtocolAuthentication** : Identité du protocole d'authentification.
- **KeyType** : Type des clés de sécurité.
- **SecurityCertificateType** : Type des certificats de sécurité (SPKI, X509, etc).

Annexe B

Spécifications détaillées des différentes interfaces de la plate-forme CASP

Cette annexe présente les spécifications détaillées de toutes les interfaces définies entre les composants de la plate-forme CASP.

La figure B.1 présente les interactions logiques entre les trois composants principaux de la plate-forme CASP (TM, MS et FS). Le rôle de ces interactions est défini comme suit :

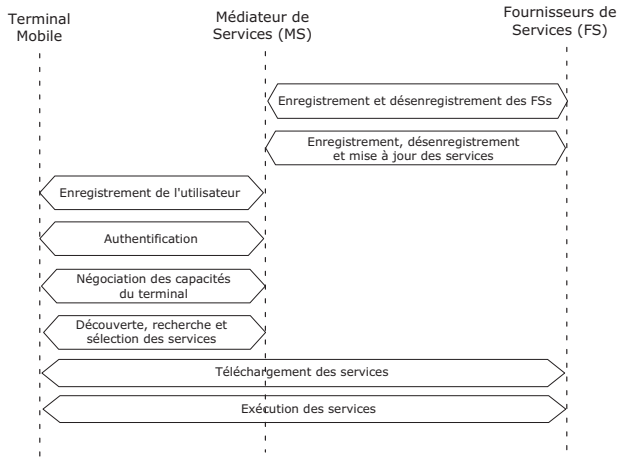


FIG. B.1 – Interactions entre les composants de la plate-forme CASP

1. *Interactions entre le TM et le MS* : elles assurent les fonctionnalités suivantes :
   - L'enregistrement et l'authentification de l'utilisateur
   - La gestion des profils
   - La négociation des capacités du terminal
   - La découverte, la recherche et la sélection des services

2. *Interactions entre le MS et les FSs* : elles assurent les fonctionnalités suivantes :
   - L'enregistrement et le désenregistrement des FSs auprès du MS
   - L'enregsitrement et le désenregistrement des services par les FSs
   - La mise à jour des services disponibles dans CASP
3. *Interactions entre le TM et le FS* : elles assurent le téléchargement, l'exécution et l'adaptabilité des services.

Ces interactions sont effectuées gâce à un ensemble d'interfaces définies entre les différents modules internes des composants de la plate-forme CASP. Ces modules ont été présentés dans la section 7.1 du chapitre 6 (voir les figures : 7.1, 7.2, 7.3 et 7.4 pour plus de détail). Les spécifications détaillées de ces interfaces sont présentées dans les sections suivantes.

B.1 Interfaces impliquées dans la fourniture et la découverte de services

Dans cette section, nous nous intéressons à la fourniture et à la découverte de services par des utilisateurs qui se connectent à leur MS Principal (pas de nomadicité). La figure B.2 présente les différentes interfaces impliquées dans cette situation.

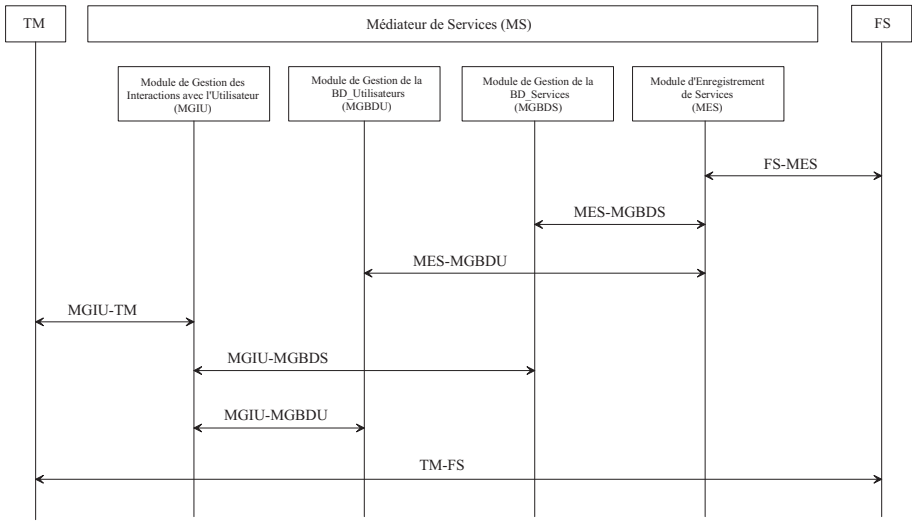


FIG. B.2 – Interfaces impliquées dans la fourniture et la découverte de services (pas de nomadicité)

B.1.1 L'interface FS-MES

Cette interface gère, à la demande des FSs, l'ensemble des requêtes concernant l'insertion, la suppression et la mise à jour des services dans CASP. Les messages qui lui sont associés sont :

- ***INSERT_REQUEST :*** Utilisé par le FS pour initialiser l'enregistrement d'un nouveau service dans CASP.
- ***INSERT_REQUEST_CONFIRMATION :*** La réponse du MS au message précédent.

- ***DELETE_REQUEST :*** Utilisé par le FS pour lancer le désenregistrement (suppression) d'un service de CASP.
- ***DELETE_REQUEST_CONFIRMATION :*** La réponse du MS au message précédent.
- ***UPDATE_REQUEST :*** Utilisé par le FS pour lancer la mise à jour d'un descripteur de services (Profil Service) dans la BD_Services.
- ***UPDATE_REQUEST_CONFIRMATION :*** La réponse du MS au message précédent.

B.1.2 L'interface MES-MGBDS

Cette interface est utilisée pour effectuer les opérations nécessaires dans la base de données des services (BD_Services), suite à une opération d'enregistrement, de désenregistrement ou de mise à jour d'un service. Les messages définis pour cette interface sont les suivants :

- ***VAS_DB_UPDATE_REQUEST (vas_db_update_request_param) :*** Le MS effectue des modifications dans le BD_Services (insertion, suppression ou mise à jour).

  NEWTYPE vas_db_update_request_param **STRUCT**
  id Integer ; // identificateur du message
  requestID Integer ; // identificateur du type de la requête (insert, delete, modify)
  vas_id Charstring optional ; // l'ID du service dans la BD_Services
  newRecord XMLDocType optional ; // document XML représentant le nouveau Profil Service
  ENDNEWTYPE vas_db_update_request_param ;

- ***VAS_DB_UPDATE_RESPONSE (vas_db_update_response_param) :*** La réponse de la BD_Services au message précédent.

  NEWTYPE vas_db_update_response_param **STRUCT**
  id Integer ; // identificateur du message
  requestID Integer ; // identificateur du type de la requête (insert, delete, modify)
  returnCode Integer ; // code de retour indiquant les résultats de l'opération
  ENDNEWTYPE vas_db_update_response_param ;

B.1.3 L'interface MGIU-TM

L'interface MGIU-TM gère l'ensemble des messages concernant l'authentification, l'enregistrement, la découverte et la recherche de services, la négociation des capacités, etc. Les messages gérés par cette interface sont :

- ***LOOKUP_PROXY (lookup_proxy_param) :*** La requête de l'utilisateur pour un Proxy LS.

  NEWTYPE lookup_proxy_param **STRUCT**
  id Integer ; // l'ID de ce message
  user_id IMSI_TYPE ; // l'ID de l'utilisateur
  ms_id MS_ID ; // l'ID du terminal
  prof_class PROFILE_CLASS ; // la classe du terminal (*terminal classmark* selon MExE)
  ENDNEWTYPE lookup_proxy_param ;

- ***LOOKUP_PROXY_RESPONSE (lookup_proxy_resp_param) :*** La réponse du MS au message précédent.

  NEWTYPE lookup_proxy_resp_param **STRUCT**
  id Integer ; // le même ID que l'ID du message requête
  rc1 RCODE1 ; // l'état de la réponse (OK ou Error)
  proxy_software PROXY_IE optional ;// le module Proxy LS
  ENDNEWTYPE lookup_proxy_resp_param ;

- ***LOOKUP (lookup_param) :*** La requête de l'utilisateur pour la fourniture du LS.

  NEWTYPE lookup_param **STRUCT**
  id Integer ; // l'ID de ce message
  user_id IMSI_TYPE ; // l'ID de l'utilisateur
  ms_id MS_ID ; // l'ID du terminal
  term_capab TERM_CAPABILITIES optional ; // l'ensemble des capacités du terminal (document XML)
  location LOCATION_ID optional ; // la localisation du terminal
  selection Charstring ; // la sélection de l'utilisateur (e.g. catégorie des services, mots-clés)
  index INDEX ; // sélection entre le LS et le "Menu des Services Favoris"
  oper_ls_selec OPERATOR_LS_SELECTION optional ; // utilisé par les utilisateurs en cas de nomadicité. Il exprime les préférences de l'utilisateur pour le contenu du LS demandé. Il prend la valeur "HOME" (pour un LS avec des services du MS principal), "VISITED" (pour un LS avec des services de l'opérateur visité) ou "BOTH" (pour un LS qui combine les deux types de services)
  ENDNEWTYPE lookup_param ;

- ***LOOKUP_RESPONSE (lookup_resp_param) :*** La réponse du MS à la requête LS de l'utilisateur.

  NEWTYPE lookup_resp_param **STRUCT**
  id Integer ; // l'ID de ce message
  rc2 RCODE2 ; // l'état de la réponse (OK ou Error)
  ls_content LS_MENU optional ; // le menu LS demandé
  ENDNEWTYPE lookup_resp_param ;

- ***VAS_SELECTION (vas_selection_param) :*** Pour envoyer au MS le service sélectionné par l'utilisateur.

  NEWTYPE vas_selection_param **STRUCT**
  id Integer ; // l'ID de ce message
  user_id IMSI_TYPE ; // l'ID de l'utilisateur
  ms_id MS_ID ; // l'ID du terminal
  term_capab TERM_CAPABILITIES optional ; // l'ensemble des capacités du terminal (document XML)
  location LOCATION_ID optional ; // la localisation du terminal
  vas_id Charstring ; // l'ID du service sélectionné
  ENDNEWTYPE vas_selection_param ;

- ***VAS_SELECTION_RESPONSE (vas_selection_resp_param) :***

  NEWTYPE vas_selection_resp_param **STRUCT**

id Integer ; // le même ID que l'ID du message requête
ms_id MS_ID ; // l'ID du terminal
rc3 RCODE3 ; // l'état de la réponse (OK ou Error)
vas_uri VAS_URI optional ; // l'URL du service sélectionné

ENDNEWTYPE vas_selection_resp_param ;

- ***TERM_CAPABILITIES (term_capabilities_param) :*** La requête du MS au terminal pour lui demander ses capacités.

NEWTYPE term_capabilities_param **STRUCT**
id Integer ; // l'ID du message
ms_id MS_ID ; // l'ID du terminal
ENDNEWTYPE term_capabilities_param ;

- ***TERM_CAPABILITIES_RESPONSE (term_capabilities_resp_param) :*** La réponse du terminal mobile au message précédent.

NEWTYPE term_capabilities_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
ms_id MS_ID ; // l'ID du terminal
rc4 RCODE4 ; // l'état de la réponse (OK ou Error)
term_capab TERM_CAPABILITIES optional ; // les capacités du terminal
ENDNEWTYPE term_capabilities_resp_param ;

- ***ABORT_LOOKUP (abort_lookup_param) :*** Utilisé par le terminal mobile pour abandonner le processus de découverte de services LS.

NEWTYPE abort_lookup_param **STRUCT**
id Integer ; // l'ID du message
user_id IMSI_TYPE ; // l'ID de l'utilisateur
ms_id MS_ID ; // l'ID du terminal
ENDNEWTYPE term_capabilities_param ;

- ***ABORT_LOOKUP_RESPONSE (abort_lookup_resp_param) :*** La réponse du MS au message précédent.

NEWTYPE abort_lookup_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
ms_id MS_ID ; // l'ID du terminal
rc5 RCODE5 ; // l'état de la réponse (OK ou Error)
ENDNEWTYPE abort_lookup_resp_param ;

- ***LOG_OFF (log_off_param) :*** Utilisé par le terminal mobile pour terminer la session de fourniture de services.

NEWTYPE log_off_param **STRUCT**
id Integer ; // l'ID du message
user_id IMSI_TYPE ; // l'ID de l'utilisateur
ms_id MS_ID ; // l'ID du terminal
ENDNEWTYPE log_off_param ;

- ***LOG_OFF_RESPONSE (log_off_resp_param) :*** La réponse du MS au message précédent.

NEWTYPE log_off_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
ms_id MS_ID ; // l'ID du terminal
rc6 RCODE6 ; // l'état de la réponse (OK ou Error)

ENDNEWTYPE log_off_resp_param ;

- ***VAS_NOTIFICATION (vas_notification_param) :*** C'est un message utilisé par le MS pour informer les utilisateurs, en cours d'exécution d'un service, que ce dernier est actuellement mis à jour ou supprimé de la BD_Services.

  NEWTYPE vas_notification_param **STRUCT**

id	Integer ; // l'ID de ce message
user_id	IMSI_TYPE ; // l'ID de l'utilisateur
ms_id	MS_ID ; // l'ID du terminal
vas_notification	NOTIFICATION_IE ; // une table d'actions à effectuer par l'utilisateur, suite à la mise à jour du service

  ENDNEWTYPE vas_notification_param ;

- ***VAS_NOTIFICATION_RESPONSE (vas_notification_resp_param) :*** Pour la réponse du MS impliqué dans une opération d'accès au service modifié.

  NEWTYPE vas_notification_resp_param **STRUCT**

id	Integer ; // le même ID que l'ID du message requête
user_id	IMSI_TYPE ; // l'ID de l'utilisateur
ms_id	MS_ID ; // l'ID du terminal
rc7	RCODE7 ; // l'état de la réponse (OK ou Error)
notification_resp	NOTIFICATION_IE_RESP optional ; // une table de réponses pour chaque action

  ENDNEWTYPE vas_notification_resp_param ;

- ***UPDATE_USER_ENTRY (update_user_entry_param) :*** Utilisé par les utilisateurs pour modifier les données concernant leur enregistrement ou leur profil (e.g. l'ajout d'un nouveau service dans leur "Menu des Services Favoris").

  NEWTYPE update_user_entry_param **STRUCT**

id	Integer ; // l'ID de ce message
user_id	IMSI_TYPE ; // l'ID de l'utilisateur
ms_id	MS_ID ; // l'ID du terminal
update_info	UPDATE_IE ; // contient les éléments qui vont être ajoutés, supprimés ou qui vont remplacer d'autres informations dans la BD_Services
index	INDICATION ; // indique si le paramètre "update_info" concerne un ajout au menu des services favoris, une mise à jour du profil de l'utilisateur ou une mise à jour des données de l'enregistrement de l'utilisateur

  ENDNEWTYPE update_user_entry_param ;

- ***UPDATE_USER_ENTRY_RESPONSE (update_user_entry_resp_param) :*** La réponse du MS au message précédent.

  NEWTYPE update_user_entry_resp_param **STRUCT**

id	Integer ; // le même ID que l'ID du message requête
user_id	IMSI_TYPE ; // l'ID de l'utilisateur
ms_id	MS_ID ; // l'ID du terminal
rc8	RCODE8 ; // l'état de la réponse (OK ou Error)

  ENDNEWTYPE update_user_entry_resp_param ;

- ***AUTHENTICATE, AUTHENTICATE_RESPONSE :*** Pour l'authentification mutuelle entre l'utilisateur et le MS.

– ***REGISTER, REGISTER_RESPONSE :*** Pour l'enregistrement d'un nouveau utilisateur à la plate-forme CASP.

B.1.4 L'interface MGIU-MGBDS

Cette interface assure les requêtes de filtrage de la BD_Services et les requêtes de notification. Les messages définis pour cette interface sont les suivants :

– ***FILTER_VAS_DB (filter_vas_db_param) :*** Utilisé par le module MGIU du MS pour demander le filtrage de la BD_Services.

NEWTYPE filter_vas_db_param **STRUCT**
id Integer ; // l'ID du message
filter_param FILTERING_PARAMETERS ; // structure ou table contenant les paramètres de filtrage
ENDNEWTYPE filter_vas_db_param ;

– ***FILTER_VAS_DB_RESPONSE (filter_vas_db_resp_param) :*** La réponse du module MGBDS du MS à la requête précédente.

NEWTYPE filter_vas_db_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
rc9 RCODE9 ; // l'état de la réponse (OK ou Error)
nrecords Integer optional ; // le nombre d'entrées dans l'ID du service
vas_ie VAS_IE optional ; // structure ou table contenant les informations demandées de la BD_Services
ENDNEWTYPE filter_vas_db_resp_param ;

– ***VAS_DB_UPDATED (vas_db_updated_param) :*** Utilisé par le module MGBDS pour informer le MGIU, chaque fois qu'une action de mise à jour (ajout, suppression ou modification) est effectuée sur les entrées de la BD_Services.

NEWTYPE vas_db_updated_param **STRUCT**
id Integer ; // l'ID du message
updated_actions UPDATED_ACTIONS ; // une table d'actions de mise à jour réalisés
ENDNEWTYPE vas_db_updated_param ;

– ***VAS_DB_UPDATED_RESPONSE (vas_db_updated_resp_param) :*** La réponse du module MGIU à la requête précédente.

NEWTYPE vas_db_updated_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
rc10 RCODE10 ; // l'état de la réponse (OK ou Error)
ENDNEWTYPE vas_db_updated_resp_param ;

B.1.5 L'interface MGIU-MGBDU

Cette interface gère la requêtes de récupération et de mise à jour des Profils Utilisateurs dans la BD_Utilisateurs. Les messages qui lui sont associés sont :

– ***USER_PROFILE (user_profile_param) :*** Utilisé par le module MGIU pour demander la récupération d'une information spécifique du profil de l'utilisateur (Profil Utilisateur).

NEWTYPE user_profile_param **STRUCT**
id Integer ; // l'ID du message
user_id IMSI_TYPE ; // l'ID de l'utilisateur
domain PROFILE_DOMAIN ; // l'information du profil de l'utilisateur ou de ses données d'enregistrement qui doit être récupérée (e.g., les préférences de l'utilisateur, sa liste de services favoris, ...)
ENDNEWTYPE user_profile_param ;

- ***USER_PROFILE_RESPONSE (user_profile_resp_param) :*** La réponse du module MGBDU au message précédent.

NEWTYPE user_profile_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
rc11 RCODE11 ; // l'état de la réponse (OK ou Error)
user_profile_ie USER_PROFILE_IE optional ; // l'information demandée du profil de l'utilisateur ou de ses données d'enregistrement
ENDNEWTYPE user_profile_resp_param ;

- ***UPDATE_USER_DB (update_user_db_param) :*** Utilisé par le module MGIU pour modifier les données concernant l'enregistrement de l'utilisateur ou son profil (e.g., l'ajout d'un nouveau service dans la liste des services favoris).

NEWTYPE update_user_db_param **STRUCT**
id Integer ; // l'ID de ce message
user_id IMSI_TYPE ; // l'ID de l'utilisateur
update_info UPDATE_IE ; // contient les éléments qui vont être ajoutés, supprimés ou qui vont remplacer d'autres informations dans la BD_Utilisateurs
index INDICATION ; // indique si le paramètre "update_info" concerne un ajout à la liste des services favoris, une mise à jour du profil de l'utilisateur ou une mise à jour des données de l'enregistrement de l'utilisateur
ENDNEWTYPE update_user_db_param ;

- ***UPDATE_USER_DB_RESPONSE (update_user_db_resp_param) :*** La réponse du module MGBDU au message précédent.

NEWTYPE update_user_db_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
user_id IMSI_TYPE ; // l'ID de l'utilisateur
rc12 RCODE12 ; // l'état de la réponse (OK ou Error)
ENDNEWTYPE update_user_db_resp_param ;

B.1.6 L'interface MES-MGBDU

Cette interface gère la notification des mises à jour des services aux utilisateurs concernés. Les messages définis pour cette interface sont :

- ***NOTIFY_USER (notify_user_param) :*** Envoyé par le module MES pour notifier à un groupe d'utilisateurs, la mise à jour ou la suppression du service qu'ils sont en train d'utiliser.

NEWTYPE notify_user_param **STRUCT**

id	Integer ; // l'ID de ce message
user_group_ids	USER_GROUP_IDS ; // les USERS_IDs et respectivement les MS_IDs des utilisateurs de ce groupe
vas_notification	NOTIFICATION_IE ;// une table d'actions que les utilisateurs doivent effectuer suite à une mise à jour ou une suppression d'un service

ENDNEWTYPE notify_user_param ;

- ***NOTIFY_USER_RESPONSE (notify_user_resp_param) :*** La réponse du module MGIU au message précédent.

NEWTYPE notify_user_resp_param **STRUCT**

id	Integer ; // le même ID que l'ID du message requête
rc13	RCODE13 ; // l'état de la réponse (OK ou Error)
nrecords	Integer optional ; // le nombre d'entrées dans le paramètre "group_notification_resp"
group_notification_resp	GROUP_NOTIFICATION_IE_RESP optional ; // une table de l'ensemble des réponses des terminaux

ENDNEWTYPE notify_user_resp_param ;

B.1.7 L'interface TM-FS

Cette interface gère les requêtes de téléchargement et d'exécution de services entre le FS et le TM. Les messages qui lui sont associés sont :

- ***INITIATE_DOWNLOAD (initiate_download_param) :*** Utilisé par le terminal pour initialiser le processus de téléchargement du service. Suite à ce message, le FS vérifie si l'utilisateur est authorisé à utiliser son service ou pas.

NEWTYPE initiate_download_param **STRUCT**

id	Integer ; // l'ID de ce message
user_id	IMSI_TYPE ; // l'ID de l'utilisateur
ms_id	MS_ID ; // l'ID du terminal
vas_id	Charstring ; // l'ID du service sélectionné

ENDNEWTYPE initiate_download_param ;

- ***INITIATE_DOWNLOAD_RESPONSE (initiate_download_resp_param) :*** La réponse du FS au message précédent. Si la réponse est positive, le terminal peut continuer le téléchargement.

NEWTYPE initiate_download_resp_param **STRUCT**

id	Integer ; // le même ID que l'ID du message requête
vas_components	SERVICE_COMPONENTS_IE optional ; // une table de fichiers constituants le service avec leurs tailles
rc14	RCODE ; // l'état de la réponse (OK ou Error)

ENDNEWTYPE initiate_download_resp_param ;

- ***SERVICE_DOWNLOAD (service_download_param) :*** Utilisé pour commencer le transfert et le téléchargement de l'ensemble des fichiers constituant le service vers le terminal.

NEWTYPE service_download_param **STRUCT**

id	Integer ; // l'ID de ce message
user_id	IMSI_TYPE ; // l'ID de l'utilisateur
ms_id	MS_ID ; // l'ID du terminal

vas_id Charstring ; // l'ID du service sélectionné
ENDNEWTYPE service_download_param ;

– ***SERVICE_DOWNLOAD_RESPONSE (service_download_resp_param) :*** La réponse du FS au message précédent.

NEWTYPE service_download_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
vas_software SERVICE_IE ; // le code du service
rc15 CODE15 ; // l'état de la réponse (OK ou Error)
ENDNEWTYPE service_download_resp_param ;

– ***STOP_SERVICE_DOWNLOAD (stop_service_download_param) :*** Message utilisé par le terminal pour arrêter momentanément le processus de téléchargement du service.

NEWTYPE service_download_param **STRUCT**
id Integer ; // l'ID de ce message
user_id IMSI_TYPE ; // l'ID de l'utilisateur
ms_id MS_ID ; // l'ID du terminal
vas_id Charstring ; // l'ID du service sélectionné
ENDNEWTYPE service_download_param ;

– ***STOP_SERVICE_DOWNLOAD_RESPONSE (stop_service_download_resp_param) :*** La réponse du FS au message précédent.

NEWTYPE stop_service_download_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
ms_id MS_ID ; // l'ID du terminal
vas_id Charstring ; // l'ID du service sélectionné
load_level LOAD_LEVEL_IE ;// spécifie le niveau d'arrêt du téléchargement dans le fichier
rc16 RCODE16 ; // l'état de la réponse (OK ou Error)
ENDNEWTYPE stop_service_download_resp_param ;

– ***CONTINUE_SERVICE_DOWNLOAD (continue_service_download_param) :*** Utilisé par le terminal pour reprendre le téléchargement d'un service déjà interrompu. Le téléchargement continue à partir du point d'interruption provoqué par le message *STOP_SERVICE_DOWNLOAD*.

NEWTYPE continue_service_download_param **STRUCT**
id Integer ; // l'ID de ce message
user_id IMSI_TYPE ; // l'ID de l'utilisateur
ms_id MS_ID ; // l'ID du terminal
vas_id Charstring ; // l'ID du service sélectionné
load_level LOAD_LEVEL_IE ;// spécifie le niveau dans le fichier à partir duquel le téléchargement doit reprendre
ENDNEWTYPE continue_service_download_param ;

– ***CONTINUE_SERVICE_DOWNLOAD_RESPONSE (continue_service_download_resp_param) :*** La réponse du FS au message précédent.

NEWTYPE continue_service_download_resp_param **STRUCT**
id Integer ; // le même ID que l'ID du message requête
ms_id MS_ID ; // l'ID du terminal
rc17 RCODE17 ; // l'état de la réponse (OK ou Error)

ENDNEWTYPE continue_service_download_resp_param);

- ***ABORT_SERVICE_DOWNLOAD (abort_service_download_param) :*** Utilisé pour abandonner le processus de téléchargement du service. Si ce message est lancé, l'invocation du message *SERVICE_DOWNLOAD* ou du message *CONTINUE_SERVICE_DOWNLOAD* va échouer.

NEWTYPE abort_service_download_param **STRUCT**
id Integer; // l'ID de ce message
user_id IMSI_TYPE; // l'ID de l'utilisateur
ms_id MS_ID; // l'ID du terminal
vas_id Charstring; // l'ID du service sélectionné
ENDNEWTYPE abort_service_download_param;

- ***ABORT_SERVICE_DOWNLOAD_RESPONSE (abort_service_download_resp_param) :*** La réponse du FS au message précédent.

NEWTYPE abort_service_download_resp_param **STRUCT**
id Integer; // le même ID que l'ID du message requête
ms_id MS_ID; // l'ID du terminal
rc18 RCODE18; // l'état de la réponse (OK ou Error)
ENDNEWTYPE abort_service_download_resp_param;

B.2 Interfaces impliquées dans le cas de nomadicité

Dans cette section, nous nous intéressons à la fourniture et à la découverte de services dans le cas de nomadicité. L'utilisateur se connecte à un MS Visité, autre que son MS Principal (qui détient ses données d'enregistrement et son Profil Utilisateur).
Le module MGIU du MS Visité communique alors avec le MGN (Module de Gestion de la Nomadicité) du MS Principal pour récupérer les informations d'enregistrement de l'utilisateur ainsi que son Profil Utilisateur. L'utilisateur peut ainsi découvrir les mêmes services personnalisés même dans le cas du roaming.

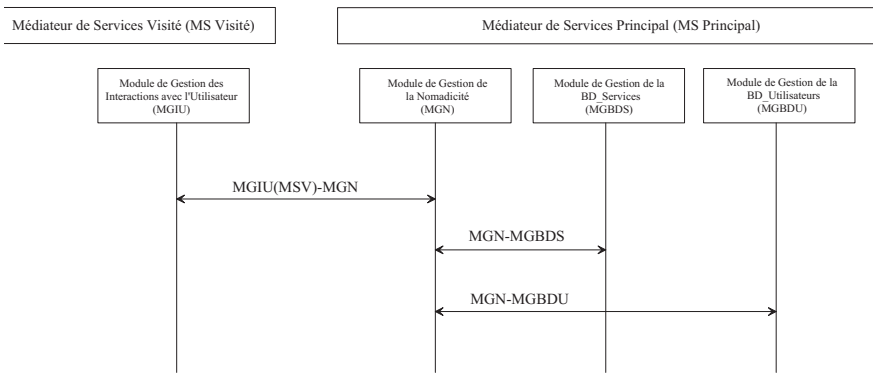


FIG. B.3 – Interfaces impliquées dans le cas de nomadicité

B.2.1 L'interface MGIU (MSV)-MGN (MSP)

Cette interface permet de transmettre les informations de personnalisation de l'utilisateur, de son MS Principal aux MSs Visités. Elle permet également de mettre à jour le Profil Utilisateur, même si les requêtes de modification sont envoyées directement à un MS Visité.
Les messages gérés par cette interface sont :

- ***GET_VAS_DB_INFO (filter_vas_db_param) :*** Message utilisé par le module MGIU du MS Visité pour demander au module MGN du MS Principal (*Home*), le filtrage de la BD_Services de l'opérateur principal.
  La structure du paramètre *filter_vas_db_param* a été définie dans l'interface MGIU-MGBDS.
- ***GET_VAS_DB_INFO_RESPONSE (filter_vas_db_resp_param) :*** La réponse du module MGN au message demandeur.
  La structure du paramètre *filter_vas_db_resp_param* a été définie dans l'interface MGIU-MGBDS.
- ***GET_USER_PROF (user_profile_param) :*** Message utilisé par le module MGIU du MS Visité pour demander au module MGN du MS Principal (*Home*) la récupération d'informations spécifiques sur le profil de l'utilisateur de la BD_Utilisateurs de l'opérateur principal.
  La structure du paramètre *user_profile_param* a été définie dans l'interface MGIU-MGBDU.
- ***GET_USER_PROF_RESPONSE (user_profile_resp_param) :*** La réponse du module MGN au message demandeur.
  La structure du paramètre *user_profile_resp_param* a été définie dans l'interface MGIU-MGBDU.
- ***PERFORM_UPDATE_USER_DB (update_user_db_param) :*** Message utilisé par le module MGIU du MS Visité pour demander au module MGN du MS Principal (*Home*), la mise à jour du Profil Utilisateur dans BD_Utilisateurs de l'opérateur principal.
  La structure du paramètre *update_user_db_param* a été définie dans l'interface MGIU-MGBDU.
- ***PERFORM_UPDATE_USER_DB_RESPONSE (update_user_db_resp_param) :*** La réponse du module MGN au message demandeur.
  La structure du paramètre *update_user_db_resp_param* a été définie dans l'interface MGIU-MGBDU.
- ***SELECT_LS_PROXY (lookup_proxy_param) :*** Message utilisé par le module MGIU du MS Visité à l'attention du module MGN du MS Principal (*Home*). Il permet la sélection d'un Proxy LS approprié pour chaque utilisateur de l'opérateur principal qui est en état de nomadicité chez l'opérateur visité.
  La structure du paramètre *lookup_proxy_param* a été définie dans l'interface MGIU-MT.
- ***SELECT_LS_PROXY_RESPONSE (lookup_proxy_resp_param) :*** La réponse du module MGN au message précédent.
  La structure du paramètre *lookup_proxy_resp_param* a été définie dans l'interface MGIU-MT.
- ***CLEAR_HOME_VAS_INFO (vas_db_updated_param) :*** Message utilisé par le module MGN du MS Principal (*Home*) pour informer le MGIU du MS Visité des différentes actions de mise à jour effectuées au niveau de la BD_Services de l'opérateur principal.
  La structure du paramètre *vas_db_updated_param* a été définie dans l'interface MGIU-MGBDS.

- ***CLEAR_HOME_VAS_INFO_RESPONSE (vas_db_updated_resp_param) :*** La réponse du module MGIU Visité au message précédent.
  La structure du paramètre *vas_db_updated_resp_param* a été définie dans l'interface MGIU-MGBDS.

B.2.2 L'interface MGN-MGBDU

Lorsque les requêtes de mise à jour du Profil Utilisateur ou des données d'enregistrement de l'utilisateur, sont reçues par un MS Visité (dans le cas du roaming), ce dernier les transmet au MGN du MS Principal. A son tour, le MGN les envoie au MGBDU, à travers l'interface MGN-MGBDU. Les modifications sont ainsi prises en compte par le MS Principal.
Les différents messages associés à l'interface MGN-MGBDU sont :

- ***USER_PROFILE (user_profile_param) :*** Utilisé par le MGN pour demander la récupération d'une information spécifique du profil de l'utilisateur (Profil Utilisateur).
- ***USER_PROFILE_RESPONSE (user_profile_resp_param) :*** La réponse du module MGBDU au message demandeur.
- ***UPDATE_USER_DB (update_user_db_param) :*** Utilisé par le module MGN pour demander des modifications des données concernant l'enregistrement de l'utilisateur ou son profil (e.g., l'ajout d'un nouveau service dans la liste des services favoris).
- ***UPDATE_USER_DB_RESPONSE (update_user_db_resp_param) :*** La réponse du module MGBDU au message précédent.

B.2.3 L'interface MGN-MGBDS

Cette interface assure la fourniture de services proposés par le MS Principal dans le cas du romaing. Les messages gérés par l'interface MGN-MGBDS sont :

- ***FILTER_VAS_DB (filter_vas_db_param) :*** Utiliser par le module MGN pour demander le filtrage de la BD_Services.
- ***FILTER_VAS_DB_RESPONSE (filter_vas_db_resp_param) :*** Pour la réponse du module MGBDS au message demandeur.
- ***VAS_DB_UPDATE_REQUEST (vas_db_updated_param) :*** Utilisé par le module MGBDS pour informer le module MGN chaque fois qu'une action de mise à jour (ajout/suppression/modification) est effectuée sur une entrée de la BD_Services. Ainsi, le MGIU du MS Visité (à travers les interactions avec le MGN) est mis au courant.
- ***VAS_DB_UPDATE_RESPONSE (vas_db_updated_resp_param) :*** La réponse du module MGN au message précédent.

Annexe C

Publications associées à cette thèse

C.1 Article dans une revue

1. *Mobile agents : a suitable support for service provision in mobile computing environment.* Studia Informatica Universalis, Hors Série OPDIS'2001, pp.33-54, 2001. N° ISSN Collection Informatique : 1621-0875. O. Fouial, N. Boukhatem and I. Demeure.

C.2 Communications internationales avec comité de lecture

2. *Fourniture de Services Adaptables dans les Environnements Mobiles.* NOTERE 2004 : conférence sur les Nouvelles Technologies de la Répartition. 27-30 Juin, Saidia, Maroc. O. Fouial et I. Demeure.
3. *CAAS : an architecture for component-based adaptable service provision.* Second ANWIRE Workshop on Reconfigurability. Mykonos, Greece, 25-26 Septembre 2003. Z. I. Kazi-Aoul, S. Ferraz, O. Fouial and I. Demeure.
4. *Service provision based on mobile agents in mobile computing environments.* Fourth IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002), Stockholm, Sweden, 9-11 September 2002. O. Fouial, N. Boukhatem and I. Demeure.
5. *Adaptative Service Provision in Mobile Computing Environments.* Second IEEE Workshop on Applications and Services in Wireless Networks (ASWN 2002), Paris, France, 3-5 July 2002. O. Fouial, K. Abi Fadel and I. Demeure.
6. *Advanced Service Provision Architecture for Mobile Computing Environments.* IST Mobile & Wireless Telecommunications Summit 2002, Thessaloniki, Greece, 16-19 June 2002. O. Fouial, S. Quesnel, P. Royard and I. Demeure.
7. *Mobile agents : a suitable support for service provision in mobile computing environment.* In OPODIS'2001, 5th International Conference On Principles Of DIstributed Systems, Manzanillo Mexico, December 2001. O. Fouial, N. Boukhatem and I. Demeure.

8. *Software Downloading Solutions for Mobile Value Added Service Provision.* IST Mobile Communications Summit 2000, Galway, Ireland, 1-4 October 2000. O. Fouial, N. Houssos and N. Boukhatem.

C.3 Conférences et colloques avec actes à diffusion restreinte

9. *Fourniture de Services Adaptatifs dans les Environnements Mobiles.* Journées sur les systèmes à composants adaptables et extensibles, Poster, Grenoble, France, 17-18 Octobre 2002. O. Fouial et I. Demeure.
10. *Plate-forme de fourniture de services pour les environnements mobiles.* In MCSEAI 2002, Seventh Maghrebian Conference On Computer Sciences, Annaba, Algeria, 6-8 May 2002. O. Fouial, N. Boukhatem et I. Demeure.

C.4 Participation aux livrables de projets

11. *D-6.1.1 Evaluation Report.* IST-1999-10206 "MOBIVAS", September 2002. All MOBIVAS members.
12. *D-5.2.1 Report on the field trial.* IST-1999-10206 "MOBIVAS", September 2002. All MOBIVAS members.
13. *D-5.1.1 Small scale pilot 3.* IST-1999-10206 "MOBIVAS", December 2001. O. Fouial, et al (Thales).
14. *D-4.4.1 Implementation of the terminal software, infrastructure and Interface.* IST-1999-10206 "MOBIVAS", October 2001. O. Fouial, et al (Thales).
15. *D-3.4.1 Design of the End User Terminal Platform.* IST-1999-10206 "MOBIVAS", December 2000. O. Fouial, N. Boukhatem, I. Demeure.
16. *D-2.1.3 Description of the pilot requirements and the user scenarios.* IST-1999-10206 "MOBIVAS", June 2000. O. Fouial, N. Boukhatem, B. Dupouy, I. Demeure, G. Mouret, et al.
17. *D-2.1.2 Requirements for general architecture and interface specification..* IST-1999-10206 "MOBIVAS", May 2000. All MOBIVAS members.
18. *D-2.1.1 Definition, Identification and Requirements of Downloadable VAS.* IST-1999-10206 "MOBIVAS", April 2000. All MOBIVAS members.