Environnement adaptatif d'exécution distribuée d'applications dans un contexte mobile

Frédéric Le Mouël

N° d'ordre: 2661

THÈSE

Présentée devant

l'Université de Rennes 1

pour obtenir

le grade de : DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention INFORMATIQUE

par

Frédéric LE MOUËL

Équipe d'accueil : IRISA
École Doctorale : Mathématiques, Informatique, Signal et Électronique et Télécommunications (MATISSE)
Composante universitaire : IFSIC

TITRE DE LA THÈSE :

Environnement adaptatif d'exécution distribuée d'applications dans un contexte mobile

soutenue le 1*er* décembre 2003 devant la commission d'examen

COMPOSITION DU JURY :

M. :	Daniel	HERMAN	Président
Mme :	Isabelle	DEMEURE	Rapporteurs
M. :	Hervé	GUYENNET	
Mme :	Françoise	ANDRÉ	Examinateurs
M. :	Bertil	FOLLIOT	
Mme :	Maria-Teresa	SEGARRA	

« C'est l'imagination qui étend pour nous la mesure des possibles, soit en bien, soit en mal, et qui, par conséquent, excite et nourrit les désirs par l'espoir de les satisfaire. »
Jean-Jacques Rousseau.

À mes grands-parents, Joachim et Renée,
à mes parents, Gilbert et Jacqueline,
à mon frère, Julien,
à mon étoile, Dubhe.

Remerciements

Je remercie Daniel HERMAN, Professeur à l'Université de Rennes 1, qui m'a fait l'honneur de présider ce jury.

Je remercie Isabelle DEMEURE, Maître de Conférences au GET / ENST Paris, et Hervé GUYENNET, Professeur à l'Université de Franche Comté d'avoir accepté d'évaluer ma thèse. J'ai particulièrement apprécié leurs rapports détaillés et approfondis témoignant de l'intérêt porté à mon travail.

Bertil FOLLIOT, Professeur à l'Université Pierre et Marie Curie, et Maria-Teresa SEGARRA, Maître de Conférences au GET / ENST Bretagne ont également accepté de juger ce travail. Leurs questions m'ont permis de défendre et valoriser mon travail. Je les en remercie.

J'exprime ma plus sincère reconnaissance à Françoise ANDRÉ, Professeur à l'Université de Rennes 1, qui a dirigé ma thèse. La justesse de ses conseils, la motivation et la confiance qu'elle m'a prodigué me furent très précieuses pour mener à bien cette thèse.

Je tiens aussi à remercier les personnes avec lesquelles j'ai travaillé, je travaille ou je continue de partager des bouts de vie, Maria-Teresa SEGARRA pour nos longues discussions et émulsions d'idées, notre soutien mutuel en période de doute et les fameuses soirées Paella-Mecano, Erwan SAINT POL pour son inépuisable énergie et bonne humeur et pour m'avoir fait découvrir le bruit (*La Noise !*).

Merci également à toutes les personnes dont l'amitié m'a apporté les moments de réconfort et distraction nécessaires lors du déroulement d'un tel projet, notamment, le Club Chicogné, pour sa disponibilité à toutes heures, et la communauté hispanophone de Rennes (Mexicaine en particulier), pour ses dégustations et soirées Salsa !

Je dédie cette thèse à ma famille. J'y puise une ouverture d'esprit, un soutien, une confiance et un amour indéfectibles que j'espère rendre pareillement et transmettre à mon tour. Parrain et mémé, j'aurais aimé que vous assistiez à ma soutenance, mais je sais que vous êtes, de toutes façons, avec moi. Papa, maman, Juju, merci de votre amour et d'être toujours là quand j'en ai besoin. Dubhe, merci pour ton sourire éclatant, tes yeux pétillants et ton amour qui me font oublier tout le reste.

Table des matières

Table des figures

Liste des tableaux

Introduction

Motivations

Depuis quelques années, les nombreuses évolutions réalisées dans les domaines des terminaux portables et des réseaux sans-fil suscitent un intérêt croissant pour l'informatique mobile. L'utilisation de ces nouveaux environnements introduit de nouvelles problématiques [Forman et Zahorjan 1994] et créé de nouveaux besoins.

Ces environnements présentent, en effet, une hétérogénéité importante et une grande variabilité aussi bien au niveau des moyens d'exécution que des moyens de communication. Les ressources offertes par un terminal portable sont, en général, bien moins importantes que celles que l'on peut trouver dans une station fixe. Ces ressources peuvent également être extrêmement disparates selon que l'on utilise un assistant personnel, un ordinateur de poche ou un ordinateur portable. De plus, la disponibilité de ces ressources n'est pas figée et peut varier en fonction d'ajout ou de suppression à chaud de périphériques ou de limitations imposées par des politiques d'économie de la batterie. Les réseaux de communications sans-fil utilisés par ces terminaux présentent également des différences notoires par rapport aux réseaux filaires. La mobilité et les performances espérées dépendent grandement de la structure du réseau et notamment de l'infrastructure déployée et de la portée de communication. Ces paramètres sont complètement différents selon que l'on utilise un réseau ad-hoc ou un réseau à point d'accès local, téléphonique ou satellitaire. De plus, les performances observées sur le lien sans-fil sont soumises à d'importantes variations occasionnées par l'environnement proche comme les interférences et déconnexions dues à des éléments physiques, des changements de cellules ou de réseau.

Ces différences que présentent les environnements mobiles par rapport aux environnements fixes nous amènent à reconsidérer les applications à exécuter dans ce genre d'environnement. En effet, en environnement fixe, les applications réservent d'importantes ressources et supposent que celles-ci sont disponibles jusqu'à la fin de l'exécution. Au lancement ou en cours d'exécution, une seule ressource brutalement non disponible conduit à une terminaison non prévue de l'application. Hors les environnements mobiles disposent de ressources moindres et celles-ci peuvent varier. On ne peut donc considérer les changements d'état des ressources comme des erreurs fatales. Il s'avère alors nécessaire de pouvoir *adapter* le comportement des applications aux environnements mobiles.

De nouveaux besoins d'adaptabilité peuvent ainsi être identifiés au niveau applicatif. Ces besoins se situent tout au long du cycle de vie d'une application. À la conception de l'application, différentes adaptations peuvent être décidées selon la sémantique de l'application. Une application de flux vidéo peut, par exemple, vouloir s'adapter aux variations de la bande passante en augmentant ou diminuant le nombre d'images transmises par seconde. Cette décision d'adaptation dépend uniquement de l'application. À l'exécution de l'application, les adaptations peuvent porter sur tout ce qui peut varier au cours de cette exécution comme les ressources de l'environnement, les performances du réseau, la localisation ou les préférences de l'utilisateur. Par exemple, une application de type météo doit adapter son contenu à la ville ou la région où se trouve l'utilisateur. Selon une prévision de déplacement de l'utilisateur, elle peut aussi afficher la météo de l'endroit où va prochainement se déplacer l'utilisateur. Enfin, l'évolution de l'application peut aussi donner lieu à des adaptations avec l'intégration de nouvelles technologies ou l'ajout de nouvelles fonctionnalités qui n'avaient pas lieu d'être à la conception. Par exemple, une application de type consultation de données conçue à l'origine avec un accès total et gratuit peut évoluer en intégrant de nouveaux modules de contrôle d'accès et de tarification.

Les approches existant dans le domaine de l'informatique mobile ne couvrent pas toutes ces formes d'adaptations. Les adaptations misent en place lors de la conception d'une application sont, en général, figées et ne peuvent évoluer. Lors de l'exécution de l'application, les adaptations peuvent être statiques ou dynamiques, selon qu'elles prennent ou non en compte les conditions d'exécutions. Les adaptations statiques permettent à l'application de bien s'exécuter dans des conditions particulières préalablement définies. Ces adaptations pèchent néanmoins en présence de variations. Une application de système de fichiers peut, par exemple, avoir été optimisée pour réduire le taux de retransmission sur un lien sans-fil. Mais, en présence d'une déconnexion, cette application sera de toute manière bloquée. Les adaptations dynamiques prennent en compte ces variations pour adapter le comportement du système ou de l'application aux conditions d'exécution courantes. Dans l'application de système de fichiers, des recopies locales en mémoire des fichiers les plus utilisés permettent de passer en mode de travail local lors d'une déconnexion. Enfin, l'évolution des applications se limite actuellement à l'arrêt de l'application, la modification de celle-ci et sa recompilation. La seule évolution dynamique possible est la reparamétrisation de l'application quand celle-ci en offre la possibilité.

L'étude de ces différentes approches nous a permis de mettre en évidence l'importance de l'environnement d'un terminal portable. En effet, les terminaux portables étant limités, la plupart des approches proposent l'optimisation d'un critère unique à l'aide de ressources extérieures au terminal portable. Dans l'application de système de fichiers, du code intermédiaire (*proxy*) peut être déployé sur le réseau fixe supportant le terminal portable pour mettre en cache les fichiers du client. Cette solution permet d'avoir une bonne optimisation de lien sans-fil avec une anticipation des accès aux données et un préchargement en vue de déconnexions. Cette solution utilise les ressources mémoire et disque de la station supportant le code intermédiaire. Cette utilisation n'est pas forcément la plus adéquate et n'est pas non plus optimisée. La station *proxy* peut très bien être en surcharge si tous les terminaux portables décident d'y mettre leur propre code intermédiaire. Cette constatation nous a amené à examiner les approches existant dans le domaine des systèmes distribués et de la répartition de charge. Certaines de ces approches s'avèrent intéressantes en environnements mobiles parce qu'elles proposent un mo-

dèle global de gestion des ressources, un support pour les ressources hétérogènes, la possibilité d'optimisation multi-critères et un placement dynamique selon ces critères. Néanmoins, ces approches ayant été conçues pour des environnements filaires, plusieurs aspects ne coïncident pas avec les priorités des environnements mobiles. Une profusion d'algorithmes de distribution existent, mais, en environnements mobiles, aucun n'est totalement approprié et n'offre de performances optimales dans toutes les situations possibles. De plus, ces algorithmes ne prennent pas en compte les nouveaux critères introduits par les environnements mobiles comme la localisation, le mouvement, la possibilité de déconnexion ou la consommation énergétique. Ces algorithmes ne supportent également pas les différentes formes d'adaptations précitées et que l'on souhaite autoriser en environnements mobiles.

Objectifs

L'objectif de cette thèse est d'élaborer un système adaptatif de distribution d'applications en environnements mobiles. Cette approche intègre aussi bien des aspects venant de l'informatique mobile que des systèmes de distribution. Il nous est donc paru important que le système vérifie un ensemble de propriétés correspondant à ces deux domaines :

Généricité : les différents éléments du système doivent être suffisamment génériques pour que chaque application puisse les utiliser.

Modularité : les différents éléments du système doivent être suffisamment bien découpés et découplés pour que, si une application le souhaite, elle puisse utiliser seulement certains de ces éléments.

Adaptabilité : les différents éléments du système doivent pouvoir être adaptés ou s'adapter eux-mêmes de manière à assurer leurs fonctionnalités dans des conditions particulières ou nouvelles. Ces adaptations doivent respecter deux sous-propriétés :

- **Prise en compte de l'environnement :** les adaptations doivent tenir compte de l'environnement où se situe le terminal portable.
- **Prise en compte de l'application :** les adaptations doivent tenir compte des besoins des applications. Une application doit également pouvoir spécialiser certains éléments du système en y incluant ses propres implantations spécifiques.

Évolutivité : de nouveaux éléments, correspondant à de nouvelles technologies ou de nouvelles fonctionnalités qui n'existaient pas ou n'avaient pas lieu d'être à la conception, doivent pouvoir être ajoutés au système.

Dynamicité : les différentes adaptations et évolutions doivent s'opérer de manière dynamique. Cette dynamicité est importante afin de réagir aux changements dans les ressources de l'environnement ou dans les besoins des applications, et à l'introduction de nouvelles technologies sans pour autant devoir arrêter et remodifier le système.

Efficacité : les environnements mobiles étant fortement contraints par le matériel, l'introduction de la généricité, de la modularité et de l'adaptabilité ne doit pas se faire au détriment de l'efficacité. Deux critères principaux sont alors importants :

Performances : l'exécution d'une application doit s'effectuer de la manière la plus performante possible en essayant de tirer le meilleur parti des ressources du terminal portable comme de celles de l'environnement. Plusieurs optimisations sont alors indispensables : le temps d'exécution de l'application, la consommation énergétique des ressources du terminal portable, les données transférées sur le lien sans-fil et l'utilisation des ressources de l'environnement.

Stabilité : dans un système distribué filaire, la stabilité du système est assurée si la différence de taux d'utilisation entre deux machines est fixe, si le temps de réponse de n processus est borné et s'il n'y a pas d'écroulement du système [Bernard et Folliot 1996]. Dans un environnement mobile, le système peut essayer d'optimiser les deux premiers critères mais il ne peut pas les assurer du fait qu'une déconnexion peut toujours survenir inopinément. Par contre, l'introduction de différentes adaptations ne doit pas conduire à l'écroulement du système, comme avec les effets « boomerang[1] » ou « domino[2] ».

Organisation du document

Ce document est décomposé en trois parties.

La première partie est consacrée au contexte de notre étude. Le premier chapitre présente une comparaison entre les environnements mobiles et les environnements fixes. Il met en évidence les besoins d'adaptations et propose quelques pistes d'adaptations auxquelles les systèmes doivent répondre. Le deuxième et troisième chapitre répertorient les solutions existantes. Ils examinent l'adéquation de ces solutions avec les besoins d'adaptations envisagés.

La deuxième partie du document présente l'ensemble de nos propositions. Le premier chapitre présente l'organisation générale du système d'adaptation que nous proposons pour les environnements mobiles. Il s'appuie principalement sur un découpage en cadre de conception et boîte à outils. Le deuxième et troisième chapitre présentent des fonctionnalités particulières que nous avons mis en exergue : la fonctionnalité d'adaptation et de réaction dynamique, et les fonctionnalités de gestion des ressources et distribution des applications.

La troisième partie développe le prototype AeDEn que nous avons réalisé, avec principalement l'utilisation du système MolèNE et les expérimentations avec une application de type navigateur.

Nous concluons ce document en rappelant les apports de cette thèse et en dressant une liste des perspectives de recherche envisageables.

[1] L'adaptation d'un élément `A` entraîne l'adaptation d'un élément `B` qui réentraîne de nouveau l'adaptation de l'élément `A`, etc.

[2] L'adaptation d'un élément `A` entraîne l'adaptation d'un élément `B` qui entraîne l'adaptation de `C`, `D`, `E`, etc et qui finalement réentraîne l'adaptation de l'élément `A`, etc.

Première partie

Le contexte de la thèse

Chapitre 1

Du fixe vers le mobile ...

De récentes percées technologiques dans le domaine des terminaux portables et dans celui des réseaux sans-fil ont conduit à une forte évolution des environnements mobiles.

Les environnements mobiles se caractérisent par la présence d'un ou de plusieurs terminaux portables ayant chacun un ou plusieurs moyens de communication sans-fil. Ces interfaces de communication sans-fil permettent aux terminaux, tout en se déplaçant, de communiquer entre eux ou avec des stations fixes. Ces environnements présentent de grandes différences par rapport aux environnements traditionnels ou fixes [Satyanarayanan 1996]. Pour des raisons de taille et de poids, les terminaux portables disposent de ressources moins importantes par rapport à celles qu'offrent des stations fixes. De plus, l'utilisation de ces ressources est limitée dans le temps puisqu'elle dépend d'une source d'énergie limitée, la batterie. En ce qui concerne les réseaux de communication sans-fil, ils offrent une bande passante beaucoup plus faible et variable que les réseaux filaires. En effet, ces communications sont soumises à de fortes variations résultant des interférences du signal avec les éléments physiques alentours. Ces variations conduisent, dans le cas extrême, à la déconnexion lorsque le signal ne parvient plus au terminal portable.

Ce chapitre fait un état des technologies actuelles des environnements mobiles. Le paragraphe 1.1 détaille les terminaux portables et fait une comparaison avec les stations fixes. Le paragraphe 1.2 présente les différents réseaux sans-fil et les compare avec les réseaux filaires. Enfin, le paragraphe 1.3 présente un éventail des différentes possibilités d'adaptation auxquelles le système ou les applications doivent répondre.

1.1 Les terminaux

1.1.1 Caractéristiques des terminaux portables

Les terminaux portables peuvent être décrits par trois caractéristiques : *les ressources*, *l'encombrement* et *l'autonomie*. Ces différentes caractéristiques ne sont pas indépendantes les unes des autres. En effet, réduire grandement la taille et le poids du terminal portable ne permet pas d'avoir d'importantes ressources mais permet d'avoir une bonne autonomie. À l'inverse, un terminal portable plus volumineux peut offrir plus de ressources, mais il consomme alors plus

d'énergie et bénéficie donc d'une autonomie moindre. Différents compromis existent pour satisfaire aux exigences des utilisateurs (voir la synthèse dans le tableau 1.1).

Les assistants personnels numériques (*Personal Digital Assistants*) sont conçus pour tenir dans la main ou être mis dans une poche. Ils sont dépourvus de clavier et la saisie s'effectue au moyen d'un stylet et d'un écran tactile. En ce qui concerne les ressources et l'autonomie, deux compromis existent : (i) les organiseurs (*Organizers, Palmtops*) [Palm, Handspring] possèdent des ressources moindres (processeur entre 16 et 33 MHz et mémoire entre 8 et 16 Mo) pour une excellente autonomie (entre 2 et 8 semaines) ; ces restrictions limitent grandement leurs fonctionnalités aux applications de consultation et saisie d'informations personnelles (agenda, répertoire, liste de tâches, mémentos, etc), et (ii) les assistants personnels de type *Pocket PC, Palm-size PC* [Casio, Compaq, HP] possèdent de plus grandes ressources (processeur entre 131 et 400 MHz et mémoire de 16 à 64 Mo) pour une autonomie moindre (de 6 à 15 heures) ; les fonctionnalités offertes permettent d'implanter les systèmes d'exploitation (Windows CE [Microsoft], Linux [LoL]) avec la plupart des applications prévues pour ces systèmes (Internet Explorer, Netscape, etc).

Les ordinateurs de poche (*Handheld Computers*) [Psion] sont un peu plus grands que les assistants personnels. En plus de l'écran tactile et du stylet, ils disposent d'un petit clavier pour la saisie de texte. Au niveau des ressources, la puissance du processeur varie entre 90 et 350 MHz, la mémoire varie entre 16 et 32 Mo. Du fait de l'écran, l'autonomie diminue encore et varie entre 6 et 10 heures. Les fonctionnalités disponibles sont similaires à celles des ordinateurs de poche, si ce n'est que le clavier rend plus facile l'utilisation des applications de type saisie et traitement de texte.

Les ordinateurs portables (*Notebooks, Laptops*) [Dell, IBM] nécessitent l'utilisation d'une sacoche pour le transport. La saisie peut se faire aisément à l'aide du clavier et la souris est remplacée par une tablette sensitive (*Touchpad*) ou un ergot de pointage (*Trackpoint*). Les ressources sont comparables à celles d'une station de travail fixe : la puissance du processeur varie de 600 MHz à 2.2 GHz et la mémoire de 128 à 512 Mo. Ces ressources permettent d'utiliser les mêmes systèmes d'exploitation et les mêmes applications que sur une station fixe, mais, par la même, limitent à une autonomie entre 1.5 et 3.5 heures.

De plus, quel que soit le terminal portable, de nouveaux périphériques peuvent être insérés en court d'exécution ("à chaud") en utilisation les possibilités d'extension qu'offrent les standards PC Card[1] [PCMCIA 2001], USB (*Universal Serial Bus*) [USB 2000] et FireWire [IEEE 1995, IEEE 2000]. Ainsi, il est possible d'ajouter aux terminaux portables des périphériques permettant d'améliorer (i) le confort d'utilisation des entrées/sorties avec, par exemple, le branchement possible d'un clavier et/ou d'une souris, l'utilisation d'un écran extérieur, ou le branchement d'une imprimante, (ii) les possibilités de stockage avec de la mémoire supplémentaire ou des disques durs, (iii) l'autonomie avec l'ajout de batteries.

1.1.2 Comparaisons avec les stations fixes

Les stations fixes sont conçues pour offrir à l'utilisateur le meilleur confort d'utilisation possible. Elles n'ont pas de contraintes de portabilité et d'autonomie et peuvent donc se per-

[1] Le standard était préalablement appelé PCMCIA mais ce terme désigne maintenant l'association qui définit le standard : *Personal Computer Memory Card International Association* [PCMCIA].

	Ressources				Encombrement		Autonomie
	Taille de l'écran (pouces) Résolution maximale (pixels) Nombre de couleurs	Processeur (MHz)	Mémoire (extensible à) (Mo)	Disque dur (Go)	Dimensions (L x l x e cm)	Poids (kg)	
Assistants personnels (*Personal Digital Assistants*)	3.6 - 3.8			-	15 x 10 x (1 - 2.5)	0.1 - 0.25	
↪ Organiseurs (*Organizers, Palmtops*)	160 x 160 16 (derniers modèles à 65536)	16 - 33	8 - 16 (64)				2 - 8 semaines
↪ *Pocket PC, Palm-size PC*	340 x 320 65536	131 - 400	16 - 64 (128)				6 - 15 h
Ordinateurs de poche (*Handheld Computers*)	6.5 - 10 640 x 480 65536	90 - 350	16 - 32 (96)	-	(18 - 25) x (10 - 20) x (2 - 5)	0.25 - 1	6 - 10 h
Ordinateurs portables (*Notebooks, Laptops*)	10 - 15.1 1400 x 1050 16 millions	600 - 2200	128 - 512 (1024)	10 - 60	(26 - 33) x (22 - 28) x (2 - 5)	1.5 - 4	1.5 - 3.5 h
Stations fixes	17 - 22 2048 x 1536 16 millions	800 - 3000	256 - 512 (2048)	20 - 80	-	-	-

TAB. 1.1 – Caractéristiques physiques des terminaux

mettre d'utiliser des ressources importantes (voir leurs caractéristiques dans le tableau 1.1).

Les ordinateurs portables sont les seuls terminaux portables pouvant offrir un confort d'utilisation et des ressources proches de celles des stations fixes. Cependant, si l'on compare les meilleurs ordinateurs portables actuels avec les meilleurs stations fixes actuelles, on constate que ces dernières sont, quand même, plus puissantes que leurs homologues portables : résolution 45% plus grande, processeurs 40% plus puissants, mémoire pouvant aller jusque 100% plus importante et disques durs pouvant aller jusque 300% plus grands. De plus, les ordinateurs portables sont contraints par une source d'énergie limitée et des restrictions de taille et de poids, limitant ainsi les utilisations possibles. Ces différences ne sont pas nouvelles mais elles ne sont pas non plus appelées à disparaître car les évolutions technologiques bénéficient aussi bien aux terminaux portables qu'aux stations fixes [Imielinski et Korth 1996].

1.2 Les réseaux

Les réseaux sans-fil peuvent être examinés selon deux aspects : l'architecture du réseau et la technologie utilisée. L'architecture définit les différents éléments d'un réseau sans-fil et comment ceux-ci communiquent et réagissent à la mobilité. La technologie utilisée permet de caractériser différents paramètres de ces communications.

1.2.1 Topologies des réseaux sans-fil

Il existe principalement deux architectures de réseaux sans-fil. L'architecture la plus répandue est celle des réseaux à point d'accès. L'architecture en réseau ad hoc est en train d'émerger (voir la synthèse dans le tableau 1.2).

1.2.1.1 Réseau sans-fil à point d'accès

La figure 1.1 présente l'architecture d'un réseau sans-fil à point d'accès. On peut y distinguer deux sortes d'entités [Helal et al. 1999] : les stations fixes et les terminaux portables. Les stations fixes sont interconnectées entre elles à l'aide d'un réseau classique de communication filaire, comme un réseau Ethernet local. Certaines de ces stations, appelées *stations de base*, jouent le rôle d'infrastructure de communication pour le réseau sans-fil. Elles possèdent une interface de communication sans-fil leur permettant de communiquer avec les terminaux portables se trouvant à portée de communication. Cette portée de communication définit une zone géographique appelée *cellule*.

La station de base est le passage obligé pour toute communication sans-fil entre terminaux portables ou avec l'extérieur. Par exemple, dans la figure 1.1, le `Terminal portable 1` doit s'adresser à la `Station de base A` pour communiquer avec le `Terminal portable 2` et avec les stations fixes.

Les terminaux portables étant par nature mobiles, ceux-ci peuvent changer de cellule. Lorsqu'un terminal portable a établi une communication, différents protocoles de changement de cellule (*Handoff Protocols*) permettent de changer de cellule tout en conservant cette connexion. Ces protocoles assurent automatiquement le transfert de prise en charge d'une station de base à une autre. Bien que des recouvrements soient possibles entre cellules contiguës,

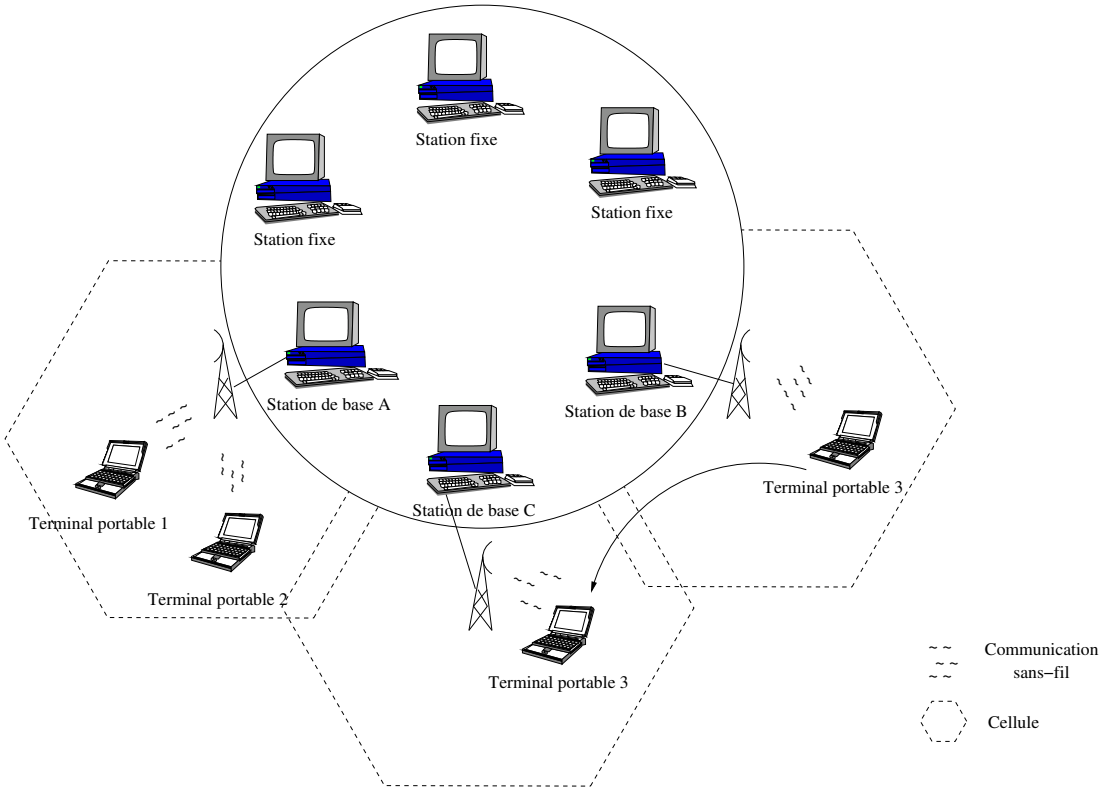


FIG. 1.1 – Architecture d'un réseau sans-fil à point d'accès

ces protocoles assurent qu'un terminal portable n'est, à un instant donné, rattaché qu'à une seule station de base. Dans la figure 1.1, le `Terminal portable 3` passe, par exemple, de la `Station de base B` à la `Station de base C`.

1.2.1.2 Réseau sans-fil ad hoc

À l'inverse des réseaux sans-fil à point d'accès, les réseaux sans-fil ad hoc ne nécessitent pas d'infrastructure de communication. La figure 1.2 présente l'architecture générale d'un réseau sans-fil ad hoc ou MANET (*Mobile Ad hoc NETworking*) [Corson et Macker 1999]. Elle est constituée d'un ensemble autonome de *noeuds*. Chaque noeud est muni d'un moyen de communication sans-fil et est capable de router les paquets lui arrivant. À un instant donné, en fonction de la position des noeuds, de la configuration de leur émetteur-récepteur (niveau de puissance de transmission) et des interférences, il y a une connectivité sans-fil qui existe entre les noeuds, sous forme de graphe multi-saut.

Comme nous allons considérer l'existence simultanée de plusieurs réseaux ad hoc, par analogie avec les réseaux sans-fil à point d'accès, nous allons appeler *cellule* chaque réseau ad hoc. Ce terme s'applique bien ici puisqu'il correspond également à la zone géographique délimitée par la portée de communication.

La communication entre noeuds d'une même cellule s'effectue en utilisant des protocoles

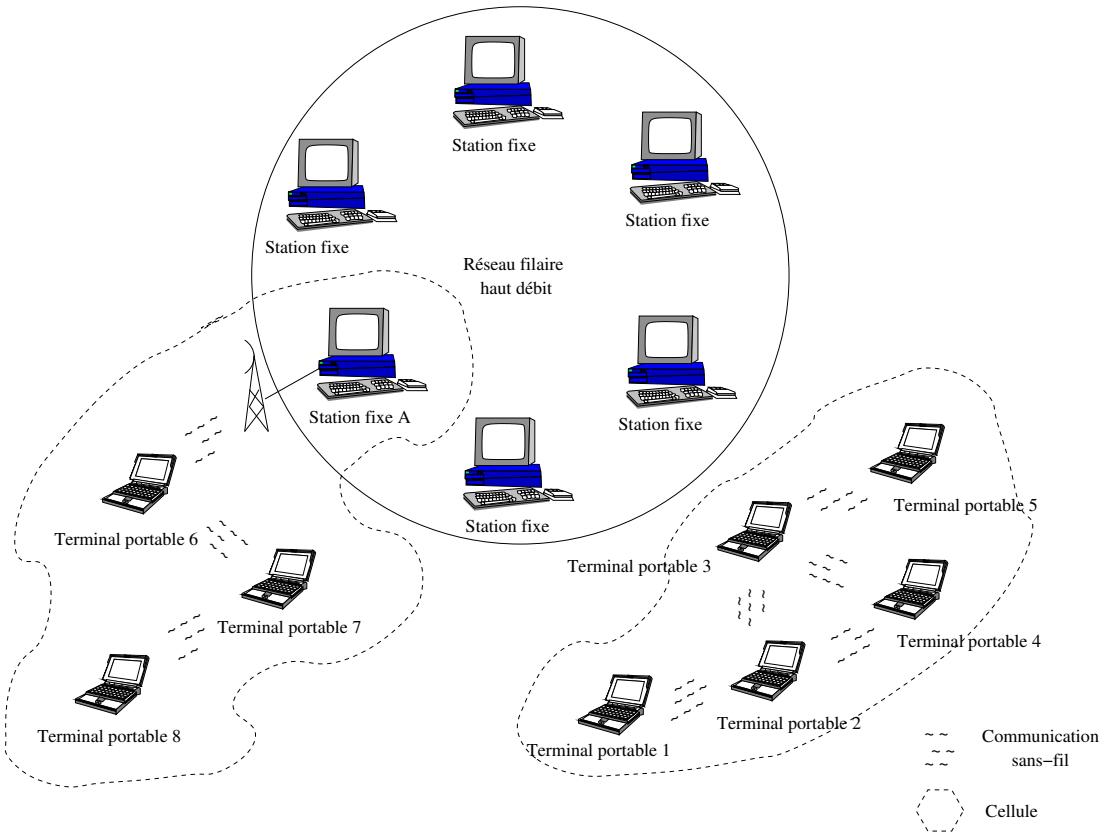


FIG. 1.2 – Architecture d'un réseau sans-fil ad hoc

de routage par voisinage. Différents protocoles (proactifs, réactifs, etc) font l'objet de recherches en ce sens [IETF]. Une cellule peut être isolée et donc n'avoir aucune possibilité de communication avec l'extérieur, comme pour les `Terminaux portables 1` à `5`, mais elle peut aussi avoir des interfaces la reliant à un réseau fixe, comme la `Station fixe A`.

Dans un réseaux sans-fil à point d'accès, les cellules sont figées et sont attachées à la zone géographique de la station de base. Dans un réseau sans-fil ad hoc, la topologie et l'emplacement des cellules peut changer avec le temps en fonction du mouvement des noeuds ou de l'ajustement de leurs paramètres d'émission-réception. La mobilité des noeuds entraîne donc plusieurs cas possibles : (i) aucune incidence : par exemple, la sortie du `Terminal portable 4` de sa cellule a pour seule incidence une mise à jour des tables de routage, (ii) la scission de cellules : si le `Terminal portable 7` s'éloigne du `Terminal portable 6` et perd la connexion avec celui-ci (tout en gardant sa connexion avec le `Terminal portable 8`), la cellule se divise en deux, ou (iii) la fusion de cellules : l'arrivée d'un `Terminal portable` qui serait à portée de communication des `Terminaux portables 1` et `7` permettrait de fusionner ces deux cellules. Ces différents changements sont assurés par les protocoles de routage, chaque noeud connaissant à tout moment ses noeuds voisins.

	Réseau sans-fil à point d'accès	Réseau sans-fil ad hoc
Cellule		
↪ Portée de communication	fixe	selon les noeuds présents dans la cellule
↪ Zone géographique	fixe	selon les déplacements des noeuds de la cellule (fusion et scission possibles)
Communications		
↪ À l'intérieur de la cellule	en passant par la station de base	par voisinage
↪ Vers l'extérieur de la cellule	toujours (en passant par la station de base)	possible si un des noeuds de la cellule est connecté à l'extérieur

TAB. 1.2 – Caractéristiques des architectures des réseaux sans-fil

1.2.2 Technologies des réseaux sans-fil

Plusieurs technologies existent pour implanter les différentes architectures de réseaux sans-fil. Celles-ci se caractérisent par deux critères : la *bande passante* disponible et la *portée de transmission*. Ces deux critères varient de manière opposée. Plus la portée de transmission est importante, plus la puissance du signal reçu est faible et sujette à interférences et, donc, plus la bande passante est faible. Différents compromis existent et chacun possède ses avantages (voir la synthèse dans le tableau 1.3).

Les réseaux à ondes infrarouges privilégient la bande passante à la portée de transmission. Deux technologies de réseaux infrarouges existent : les réseaux à ondes infrarouges directes et les réseaux à ondes infrarouges diffuses. Les réseaux à ondes infrarouges directes sont normalisés par les standards IrDA Data [IrDA 1996a, IrDA 1996b, IrDA 2001] et IrDA Control [IrDA 1998]. La communication s'effectue en plaçant les entités communiquantes sur une même ligne de vue et à une distance maximale de 2 m. Les échanges se font entre 1 et 4 Mb/s. Les réseaux à ondes infrarouges diffuses ne sont pas très répandus bien que certaines expérimentations [Harter et Hopper 1994] et réalisations commerciales [Spectrix] montrent pourtant qu'ils offrent des perspectives intéressantes. La bande passante offerte est toujours de 4 Mb/s, mais une antenne à ondes infrarouges diffuses couvre une zone de 100 m^2. Les ondes infrarouges ne traversent pas les murs, donc ce type d'antenne est plutôt installé en intérieur pour couvrir une pièce.

Les réseaux radio à courte portée se positionnent sur le même créneau que les réseaux à ondes infrarouges. Par rapport aux réseaux à ondes infrarouges directes, les antennes radio présentent l'avantage de communiquer non pas sur une ligne mais sur une cellule. De plus, les ondes radio traversent les objets qui peuvent faire obstacle aux ondes infrarouges. Ces caractéristiques font que ce type de réseau est en plein émergence, notamment avec le standard Bluetooth [Bluetooth]. Ce standard offre la possibilité à un terminal portable de communiquer dans un rayon de 10 m et avec une bande passante de 1 Mb/s.

Les réseaux locaux sans-fil utilisent également les ondes radio pour les communications sans-fil. Les antennes sont, par contre, beaucoup plus puissantes et permettent d'augmenter la portée de transmission. Deux familles de standards existent pour la mise en place de tels réseaux, ceux définis par l'IEEE [IEEE WLAN] et ceux définis par l'ETSI [ETSI]. Les premiers (IEEE 802.11 [IEEE 1999c], 802.11b [IEEE 1999b] et 802.11a [IEEE 1999a]) proposent différentes bandes passantes selon la distance de portée : 54 Mb/s sur 80 m, 48 Mb/s sur 120 m, 36 Mb/s sur 160 m, 24 Mb/s sur 210 m, 12 Mb/s sur 400 m. Ces standards sont actuellement très utilisés par les constructeurs [Orinoco, Proxim]. Les seconds (HiperLAN/1 [ETSI 1998] et HiperLAN/2 [ETSI 2000]) offrent une bande passante de 54 Mb/s pour une portée de communication de 30 m (intérieur) à 150 m (sans obstacle). Ceux-ci permettent l'assignation d'une priorité ou d'une qualité de service à chaque connexion.

Les réseaux de téléphonie cellulaire initialement conçus pour la transmission de la voix [Ames et Gabor 2000] permettent maintenant la transmission de donnés. Les réseaux actuellement en service sont de la génération 2.5. Les exemples les plus probants sont : IS-95-B (64 Kb/s) ou IS-136-B (64-115 Kb/s) pour D-AMPS, GPRS (*General Packet Radio Service*) (115-170 Kb/s) ou EDGE (*Enhanced Data rates for GSM Evolution*) (384 Kb/s) pour le GSM, avec des portées entre 500 m et 10 km. Le groupe de travail IMT-2000 [ITU] a défini les caractéristiques des réseaux de $3^{ème}$ génération à l'aide de cinq recommandations pour les standards (IMT-MC DS TC SC FT). Le standard UMTS (*Universal mobile Telecommunications System*) [3GPP 2001] respecte ces recommandations. La bande passante autorisée y varie entre 144 Kb/s pour les véhicules et environnements extérieurs ruraux, 384 Kb/s pour les environnements extérieurs urbains et 2 Mb/s pour les environnements urbains denses et les environnements intérieurs. La zone de couverture doit être calculée pour chaque antenne. Elle dépend du facteur de service que l'on veut offrir ainsi que du nombre d'utilisateurs [Schueller et al. 2000]. La portée maximale de transmission est de l'ordre de 50 km.

Les réseaux satellitaires sont constitués de constellations de satellites qui offrent une couverture presque totale de la planète [Globalstar, Skybridge, Teledesic]. Le nombre de satellites de ces constellations peut varier de 48 à 288 et la couverture offerte par chaque satellite peut donc varier de 3 à 3000 km. Deux cas se présentent pour la bande passante : elle peut aller jusque 64 Mb/s en réception et jusque 2 Mb/s en émission. Du fait de l'éloignement terre-satellite, les délais de propagation sont néanmoins importants et les communications souffrent donc de délais de latence de bout en bout importants (entre 50 et 500 ms).

1.2.3 Comparaison avec les réseaux filaires

Les réseaux filaires sont très utilisés pour permettre des communications à haut débit notamment dans des réseaux locaux ou grappes de stations. Ces réseaux offrent une bande passante bien supérieure à celle que l'on peut obtenir dans les réseaux sans-fil : ATM (*Asynchronous Transfert Mode*) [ATM] (622 Mb/s), Gigabit Ethernet (IEEE 802.3z [IEEE LAN]) (1 Gb/s), SCI (*Scalable Coherent Interface*) [SCI] (1.33 Gb/s), Myrinet [Myrinet] ($\sim$2 Gb/s), SONET (*Synchronous Optical NETwork*) [SONET] (2.5 Gb/s), standard 10-Gigabit Ethernet (IEEE 802.3ae [10GEA]).

Par rapport aux réseaux filaires, les réseaux sans-fil souffrent également d'importantes variations de la bande passante. Ces variations proviennent de plusieurs phénomènes comme

Type de réseau	Technologie	Portée de transmission	Bande passante	Mobilité supportée	Topologie possible
Réseau infrarouge					
↪ direct	ondes infrarouges directes	2 m	4 Mb/s	statique	ad hoc
↪ diffus	ondes infrarouges diffuses	10 m	4 Mb/s	statique - piéton	point d'accès
Réseau radio courte portée	ondes radio	10 m	1 Mb/s	statique - piéton	ad hoc
Réseau local sans-fil	ondes radio	150 m	54 Mb/s	piéton	ad hoc & point d'accès
Réseau de téléphonie	ondes radio	50 km	2 Mb/s	véhicule	point d'accès
Réseau satellitaire	ondes radio	3000 km	2 Mb/s (émission) 64 Mb/s (réception)	véhicule	point d'accès

TAB. 1.3 – Caractéristiques des technologies des réseaux sans-fil

l'atténuation ou le bruit des signaux. L'atténuation du signal survient lorsque le signal doit passer à travers un objet, comme un mur, ou lorsqu'il doit parcourir une distance plus importante que prévue, par exemple, en se réfléchissant sur un mur. Le bruit peut être causé par des interférences avec le signal lui-même, par exemple lorsqu'il se réfléchit plusieurs fois, ou avec des signaux extérieurs générés, par exemple, par un poste radio ou une antenne de télévision. Ces interférences provoquent des pertes de données qui doivent ensuite être retransmises, diminuant ainsi la bande passante.

Le cas extrême d'une variation est la déconnexion. Trois types de déconnexions peuvent se produire : les déconnexions de courte durée, les déconnexions de durée indéterminée et les déconnexions volontaires. Les déconnexions de courte durée se produisent lors des interférences ou lors d'un changement de cellule. La mobilité du terminal permet en général de rétablir rapidement la connexion. Les déconnexions de durée indéterminée se produisent lorsqu'un terminal portable sort en dehors de toutes zones de couverture. Dans ce cas, la reconnexion ne surviendra que lorsque le terminal rentrera dans une zone de couverture. Enfin, l'utilisateur peut vouloir économiser sa batterie et éteindre son portable. La reconnexion ne se produira alors que selon le bon vouloir de l'utilisateur.

1.3 Discussion

En environnement mobile, exécuter une application conçue pour un environnement fixe peut s'avérer délicat. Comme l'ont montré les paragraphes précédents, des différences importantes existent entre ces deux types d'environnements et elles ne sont pas, en général, prises en compte par les applications. En effet, la conception d'une application se base la plupart du temps sur un environnement connu, statique et suffisant pour les besoins de l'application. Cela n'est plus vrai dans un environnement mobile où des terminaux portables peuvent arriver et repartir, disposer de ressources faibles à un instant donné, puis en avoir suffisamment l'instant d'après. Afin de prendre en compte ces changements possibles, il est nécessaire que les applications puissent s'*adapter*. Trois axes principaux d'adaptations sont définis ci-dessous :

Adaptation au terminal portable

Ressources restreintes : la restriction des ressources sur un terminal portable peut empêcher l'exécution d'une application sous des conditions optimales. Le mécanisme d'allocation des ressources doit pouvoir informer une application qu'une ressource dont elle a besoin est temporairement indisponible ou réduite. L'application pourrait alors changer son comportement pour utiliser moins de ressources ou des ressources différentes. Par exemple, une application de chargement n'ayant plus assez de mémoire pour ses données pourrait décider de stocker ses données sur le disque dur, ou bien de ne charger que les données les plus utilisées.

Autonomie limitée : la durée de vie de la batterie est un paramètre limitant incontournable. Actuellement, les mécanismes d'économie de la batterie sont principalement matériels et n'interviennent que lorsque la batterie est très basse ou lorsque le terminal n'est pas utilisé. Avertir les applications du niveau de la batterie pourrait leur permettre d'adop-

ter des modes dégradés adéquats et d'économiser encore un peu plus la batterie. Une application pourrait, par exemple, passer d'un affichage graphique à un affichage textuel.

Ajout de ressources possible : l'ajout en cours d'exécution de différents périphériques doit aussi être notifié aux applications pour qu'elles puissent en profiter. Des changements de comportement inverses des deux précédents pourraient alors être envisagés.

Adaptation au réseau sans-fil

Variations de la bande passante : les variations de la bande passante peuvent introduire des délais inacceptables pour certaines applications ayant, par exemple, des contraintes d'interactivité. Avertir ces applications pourrait leur permettre de changer la qualité de service en accord avec l'utilisateur. Lors d'une variation de bande passante, une application de visio-conférence pourrait, par exemple, décider de réduire le nombre de couleurs ou la taille de l'image.

Déconnexions réseau : les déconnexions peuvent totalement paralyser une application ayant besoin d'un accès distant. L'anticipation des déconnexions en utilisant des techniques de préchargement pourrait permettre à l'utilisateur de travailler en mode déconnecté. Une application de base de données pourrait, par exemple, précharger la partie de la base de données sur laquelle travaille l'utilisateur.

Adaptation à l'environnement

Localisation géographique : la mobilité des terminaux portables permet une utilisation depuis n'importe quel endroit. De nouvelles applications utilisent maintenant cette localisation pour rendre leurs services. Elles doivent pouvoir être notifiées des déplacements pour adapter leur contenu. Une application de presse électronique doit, par exemple, pouvoir afficher l'actualité et la météo de l'endroit où se trouve l'utilisateur.

Utilisation des ressources de l'environnement : les terminaux portables doivent également pouvoir profiter des ressources offertes par l'environnement local. Par exemple, un terminal portable peut être intéressé de savoir qu'une imprimante est disponible dans la pièce où il vient d'entrer, ou bien que le réseaux local où se trouve sa station de base offre, en accès libre, un serveur d'exécution d'applications.

Chapitre 2

Systèmes d'adaptation aux environnements mobiles

La forte évolution des technologies des environnements mobiles se retrouve dans le nombre très important de travaux traitant des spécificités de ces environnements. Ces travaux sont très diversifiés et abordent l'adaptation aux environnements mobiles selon des angles différents. Certains s'appliquent aux plus bas niveaux avec la mise en place de systèmes d'exploitation ou de protocoles réseaux adaptés. D'autres s'appliquent à plus haut niveau et concernent l'adaptation générique d'applications, ou l'adaptation spécifique de certaines applications comme les applications transactionnelles ou de visio-conférence.

Ce chapitre fait un état de ces travaux de manière à mettre en évidence les différentes formes d'adaptations possibles décrites dans le chapitre précédent.

Le paragraphe 2.1 introduit quelques définitions importantes permettant de faire le lien entre environnements mobiles, applications et systèmes d'adaptation.

La mobilité des terminaux portables peut être gérée selon deux approches décrites dans [Satyanarayanan 1996] : les approches masquant la mobilité aux applications (*application-transparent approaches*) et les approches montrant la mobilité aux applications (*application-aware approaches*). Les premières approches, décrites dans le paragraphe 2.2, sont assez attrayantes car elles permettent d'utiliser les applications existantes sans aucune modification. Elles présentent néanmoins le désavantage de ne pas tirer profit de la mobilité. Au contraire, les approches montrant la mobilité aux applications, détaillées dans le paragraphe 2.3, permettent à celles-ci de s'adapter aux variations selon la stratégie la plus adéquate.

Bien que ne traitant pas forcément d'environnements mobiles, les systèmes adaptatifs se basant sur une approche réflexive proposent différentes techniques d'adaptation qui pourraient s'appliquer aux environnements mobiles. Ces approches seront examinées dans le paragraphe 2.4.

2.1 Définitions

Ce paragraphe définit le vocabulaire que nous utiliserons dans la suite de cette étude. Il reprend et complète les concepts introduits dans [Segarra 2000].

Environnement mobile : un environnement mobile est constitué d'un ensemble de terminaux portables se déplaçant librement et communiquant au moyen d'un réseau sans-fil.

Système adaptatif ou d'adaptation : un système adaptatif ou d'adaptation est une entité logicielle qui prend en compte les besoins des applications et les caractéristiques de l'environnement mobile pour exécuter les applications. Les besoins des applications sont spécifiés dans le *comportement d'exécution* et les caractéristiques de l'environnement mobile sont prises en compte par la *stratégie d'adaptation*.

Application : une application est une entité logicielle qui utilise un système d'adaptation pour son exécution dans un environnement mobile.

Les interactions entre ces différents éléments sont décrites dans la figure 2.1. Pour s'exécuter, l'application utilise un système d'adaptation. Selon le comportement d'exécution et la stratégie d'adaptation, celui-ci modifie l'exécution de l'application en agissant sur l'environnement mobile, l'application ou le système d'adaptation lui-même.

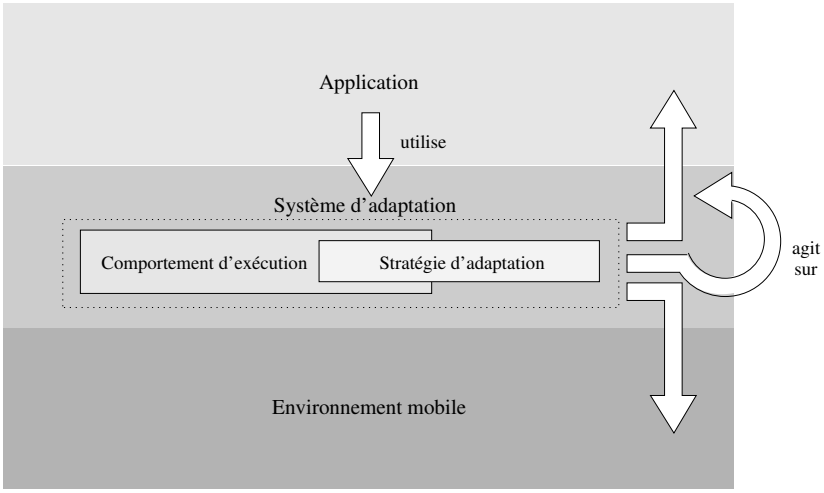


FIG. 2.1 – Interactions entre application, système d'adaptation et environnement mobile

Les systèmes d'adaptation peuvent être classés selon deux familles de comportements d'exécution et deux familles de stratégies d'adaptation :

Système adaptatif à comportement spécialisable (CS) (resp. non spécialisable (CNS)) : un système adaptatif à comportement spécialisable offre (resp. n'offre pas) la possibilité aux applications de spécifier leurs besoins au sein du système d'adaptation.

Système adaptatif à stratégie contextuelle (sc) (resp. non contextuelle (snc)) : un système adaptatif à stratégie contextuelle prend en compte (resp. ne prend pas en compte) les caractéristiques des environnements mobiles.

La prise en compte des besoins des applications ou des caractéristiques de l'environnement peut s'effectuer statiquement ou dynamiquement et constitue donc deux sous-familles :

Système adaptatif à comportement spécialisable statiquement (CSS) : un système adaptatif à comportement spécialisable statiquement prend seulement en compte les besoins des applications au lancement de celles-ci. Ces besoins sont supposés ne plus changer au cours de l'exécution.

Système adaptatif à comportement spécialisable dynamiquement (CSD) : un système adaptatif à comportement spécialisable dynamiquement tient compte des besoins des applications au cours de l'exécution de celles-ci. Ces besoins peuvent changer au cours de l'exécution.

Système adaptatif à stratégie contextuelle statique (scs) : un système adaptatif à stratégie contextuelle statique prend en compte les caractéristiques des environnements mobiles mais suppose que les conditions d'exécution associées à ces environnements sont invariables.

Système adaptatif à stratégie contextuelle dynamique (scd) : un système adaptatif à stratégie contextuelle dynamique prend en compte les caractéristiques des environnements mobiles ainsi que les variations dans les conditions d'exécution associées à ces environnements.

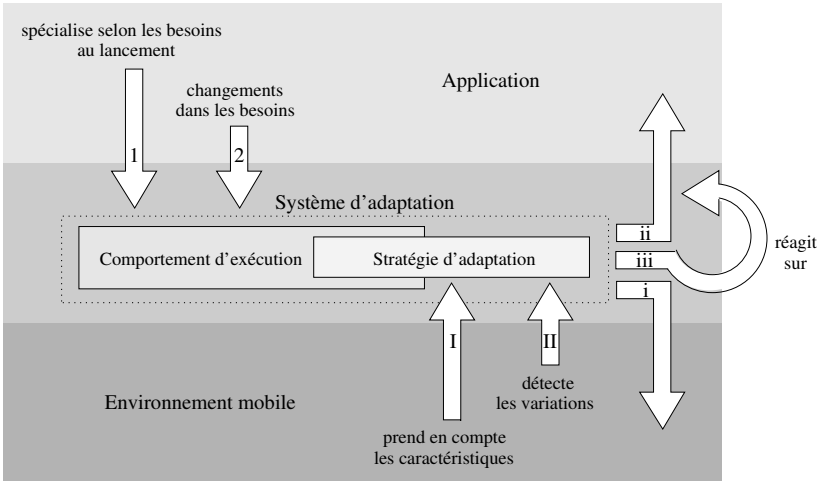


FIG. 2.2 – Système adaptatif à comportement spécialisable dynamiquement et à stratégie contextuelle dynamique (CSD-scd)

La figure 2.2 illustre un système adaptatif à comportement spécialisable dynamiquement et à stratégie contextuelle dynamique (CSD-scd). L'application peut spécifier ses besoins en spécialisant le comportement d'exécution du système d'adaptation. Cette spécialisation peut, par exemple, concerner les besoins en ressources de l'application, mais cela peut aussi permettre à l'application de définir sa propre stratégie d'adaptation. Cette spécialisation se fait au lancement de l'application (étape 1 de la figure 2.2) mais peut changer au cours de l'exécution (étape 2). Dans la figure, la stratégie d'adaptation prend en compte les caractéristiques

de l'environnement mobile (étape I) et détecte toutes les variations qui pourraient y survenir (étape II). Tout changement intervenant dans le comportement d'exécution (étape 2) ou toute variation détectée par la stratégie d'adaptation (étape II) peut alors produire une réaction d'adaptation agissant sur l'environnement mobile, l'application ou le système d'adaptation lui-même (étapes i, ii et iii).

2.2 Transparence de la mobilité

Les variations que l'on peut observer dans un environnement mobile sont des phénomènes de bas niveau. Elles influent directement sur les couches protocolaires utilisées par les applications. Pour rendre transparents les déplacements et les déconnexions, il s'avère donc nécessaire d'adapter les protocoles aux environnements mobiles. L'adressage IP [Deering et Hinden 1998] et TCP [Postel 1981], NFS [Shepler et al. 2003] et HTTP [Fielding et al. 1999, Khare et Lawrence 2000] sont les principaux protocoles ayant fait l'objet de recherches en ce sens.

2.2.1 Transparence de l'adressage IP

Actuellement, quand un terminal portable change de réseau, son adresse IP change. Les applications utilisant le protocole TCP/IP ne sont pas conçues pour gérer dynamiquement un changement d'adresse IP. De nombreux travaux ont proposés des solutions à ce changement dynamique d'adresse [Sunshine et Postel 1980, Ioannidis et al. 1991, Teraoka et al. 1992, Wada et al. 1993, Perkins et al. 1994]. Mobile-IPv6 [Perkins et al. 2003] est un travail en cours proposé par l'IETF permettant de prendre en compte la mobilité des stations au sein du protocole IPv6 [Deering et Hinden 1998].

2.2.1.1 Mobile-IP

Mobile-IP se base sur un procédé à deux adresses. Le terminal portable (*Mobile Node*) possède une adresse IP permanente (*home address*) correspondant à son réseau d'attache (*home network*) et une adresse IP temporaire (*care-of address*) correspondant au réseau où il se trouve actuellement (*foreign network*).

Quand le terminal portable se trouve dans son réseau d'attache, les paquets qui lui sont adressés sont routés de manière conventionnelle en utilisant les protocoles de routage Internet. Ce routage se fait de manière triviale puisque le préfixe de l'adresse IP permanente correspond au préfixe du réseau d'attache.

Quand le terminal portable se déplace dans un autre réseau, il obtient une adresse IP temporaire en utilisant, par exemple, le protocole DHCPv6 [Thomson et Narten 1998, Droms et al. 2002]. À un instant donné, un terminal portable peut avoir plusieurs adresses IP temporaires. Il en choisit une comme étant son adresse primaire (*primary address*) et l'enregistre auprès d'un agent se trouvant sur son réseau d'attache (*Home Agent*) au moyen d'un datagramme spécifique (*Binding Update*) (étape 1 sur la figure 2.3). L'agent met à jour sa table de routage et joue alors temporairement le rôle de routeur en redirigeant et en encapsulant les communications vers la nouvelle adresse IP temporaire (étapes 2 et 3 sur la figure 2.3). Le

terminal portable peut alors avertir ses correspondants de sa nouvelle adresse et établir une connexion directe (étape 4 sur la figure 2.3).

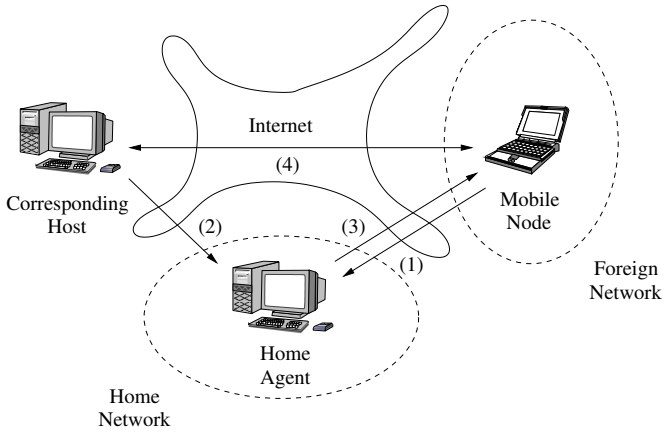


FIG. 2.3 – Support de la mobilité dans Mobile-IP

Mobile-IP permet de résoudre le problème de la mobilité en utilisant une stratégie contextuelle dynamique qui se base sur le passage d'un réseau à un autre. Il présente cependant l'inconvénient de devoir passer par l'agent du réseau d'attache. Dans un réseau sans-fil avec des cellules couvrant de petites zones géographiques, un trafic important, pouvant entraîner des situations de congestion, est généré à chaque changement de localisation. De plus, lors d'un déplacement géographiquement important, le temps d'établissement de la connexion avec l'agent du réseau d'attache peut se révéler plus important que le temps d'établissement de la connexion avec les correspondants. Pour pallier ces inconvénients, différentes techniques peuvent être utilisées telles que la réplication des informations de localisation [Jannink et al. 1997] ou une organisation hiérarchique en régions comme dans Cellular IP [Valkó 1999], HAWAII [Ramjee et al. 1999] ou HMIPv6 [Soliman et al. 2002].

2.2.2 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 [Postel 1981] interprète ces pertes comme une situation de congestion et active alors les mécanismes appropriés. Ces mécanismes induisent alors une dégradation significative des performances [Cáceres et Iftode 1994]. D'autres mécanismes, comme la mise à jour de la localisation ou le recalcul des tables de routage, peuvent également produire des dégradations des performances [Holland et Vaidya 1999]. Plusieurs travaux optimisent le protocole TCP pour les environnements mobiles en utilisant des techniques différentes. Ils peuvent être classés en deux familles [Balakrishnan et al. 1997] : (i) les protocoles masquant complètement le lien sans-fil à l'émetteur (*TCP-unaware*) et (ii) les protocoles informant l'émetteur du lien sans-fil (*TCP-aware*). Les approches de la première famille se basent sur le fait 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 les approches de la seconde famille, l'émetteur est conscient du lien sans-fil et est capable de distinguer les pertes de transmission dues au lien sans-fil de celles dues à une congestion. Dans ce cas, l'émetteur peut décider à bon escient de l'activation des mécanismes de congestion. Ces deux familles sont illustrées ci-dessous respectivement par I-TCP, Snoop et Mobile TCP, WTCP.

2.2.2.1 I-TCP

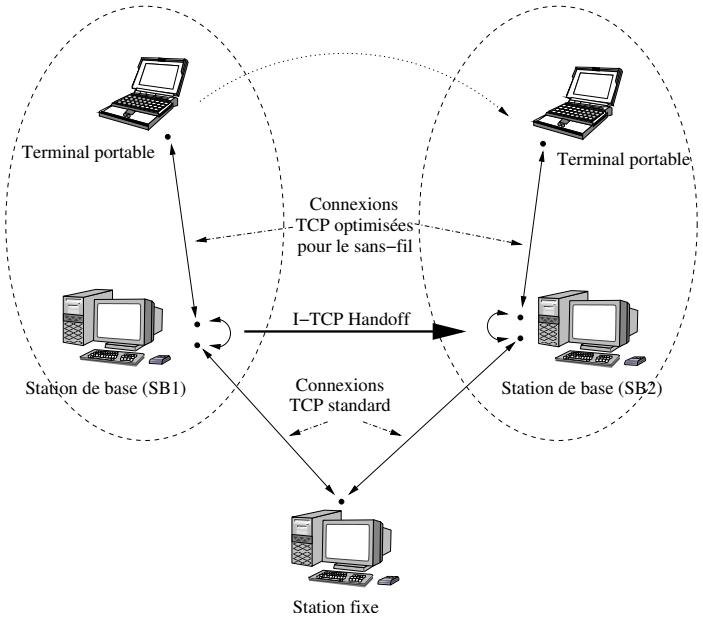


FIG. 2.4 – Indirection d'une connexion TCP dans I-TCP

I-TCP [Bakre et Badrinath 1995a, Bakre et Badrinath 1997] décompose la connexion entre une station fixe et un terminal portable en deux connexions. La première se situe entre la station fixe et la station de base et la seconde entre la station de base et le terminal portable. La figure 2.4 explique le fonctionnement du protocole. Lorsqu'une connexion est demandée entre le terminal portable et une station fixe, la station de base (SB1) reçoit une requête d'ouverture. Elle ouvre alors une autre connexion TCP avec le récepteur. Les connexions ouvertes entre la station de base et les terminaux portables utilisent un protocole TCP optimisé pour le sans-fil [Yavatkar et Bhagwat 1994], tandis que les connexions avec le réseau fixe utilise le protocole TCP standard. Les données envoyées sont d'abord reçues par la station de base qui acquitte l'émetteur et les achemine ensuite au récepteur. Lorsque le terminal portable change de cellule, l'état de la connexion est transféré de SB1 vers SB2. L'indirection est transparente pour la station fixe et le terminal portable comme leurs ports de communication restent inchangés. Ce découpage de la connexion est repris par d'autres protocoles comme Amigos [Hansen et al. 1996, Hansen et Reich 1996] ou [Brown et Singh 1997, Haas 1997].

Cette approche présente néanmoins deux inconvénients : (i) les stations de base doivent être plus complexes et posséder d'importants tampons en cas de trafic important, (ii) lors de

fréquents changements de cellule, le surcoût relatif à la transmission de l'état de la connexion peut être important et introduire des délais.

2.2.2.2 Snoop

Dans cette approche, la couche réseau n'est modifiée nulle part sauf sur la station de base. Son code de routage utilise un module, *the Snoop module* [Balakrishnan et al. 1995a, Balakrishnan et al. 1995b], qui examine les paquets TCP. Les actions de la station de base lorsqu'elle reçoit un paquet sont décrites dans la figure 2.5. À la réception d'un paquet envoyé par la station fixe, le module Snoop copie ce paquet dans un cache et le transmet au terminal portable. Si ce paquet est perdu au cours de la transmission sans-fil, la station de base reçoit des acquittements dupliqués pour le paquet précédent. Le protocole d'acquittement de la station de base est expliqué dans la figure 2.6. Lorsque le module Snoop reçoit des acquittements dupliqués, si le paquet perdu est dans son cache, il le retransmet alors et n'envoie pas les acquittements dupliqués à la station fixe. Si le paquet n'est pas dans son cache, il achemine les acquittements dupliqués à la station fixe qui est alors en charge de revenir sur la perte du paquet.

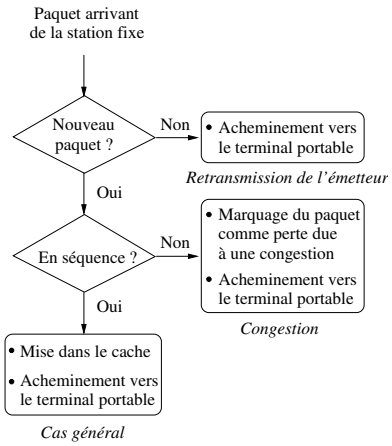


FIG. 2.5 – Protocole de réception de paquets dans le module Snoop

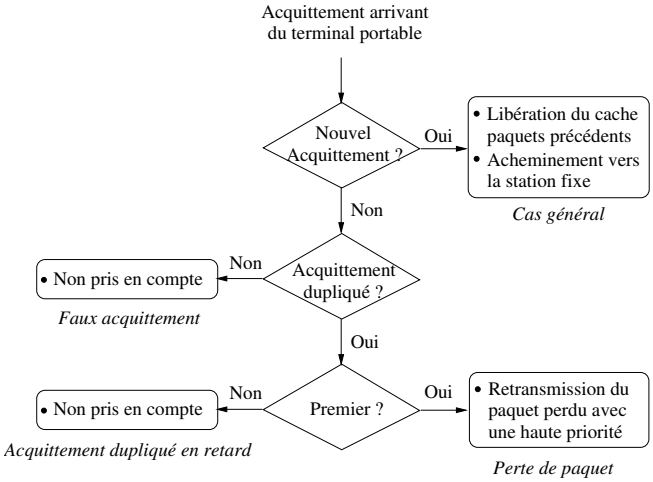


FIG. 2.6 – Protocole de réception d'acquittements dans le module Snoop

Cette approche identifie aussi un groupe de stations de base vers lesquelles le terminal portable est susceptible de se déplacer. Ces stations reçoivent (en *broadcast*) et mettent en cache tous les paquets destinés au terminal portable. Si un terminal portable se déplace vers une station de base du groupe, les paquets sont déjà disponibles dans le cache, évitant un coût de transfert d'état pendant le changement de cellule et permettant ainsi une récupération rapide sur perte. Si le terminal portable se déplace vers une station de base ne faisant pas partie du groupe, reconstruire le cache prend alors un temps important. Cette méthode induit néanmoins un coût important sur le réseau fixe. Ce système de cache est repris par d'autres approches en version "*unaware*" [Chan et al. 1997], comme en version "*aware*" [Vaidya et al. 1999].

2.2.2.3 Mobile TCP

Avec TCP standard, toutes les pertes de paquet sont interprétées par la station fixe émettrice comme une situation de congestion impliquant des méthodes de contrôle comme la réduction de la fenêtre d'émission. Mobile TCP [Stangel et Bharghavan 1998] permet de distinguer les différentes pertes de paquets. La station de base peut informer la station fixe de pertes de paquets dues à (i) une congestion, (ii) un changement d'interface ou (iii) un changement de cellule. Dans le premier cas, elle peut activer les mécanismes appropriés à la congestion. Dans le second cas, le terminal portable peut se déplacer dans un réseau possédant des caractéristiques totalement différentes du précédent. La station fixe effectue alors une remise aux valeurs par défaut de la taille de la fenêtre d'émission (*window size*), du seuil de l'algorithme *slow start* (*ssthresh*), du temps estimé d'aller-retour (*RTT : Round Trip Time*) et du temporisateur de retransmission (*RTO : Retransmission Time Out*), et redémarre l'algorithme *slow start*. Enfin, dans le troisième cas, la retransmission ne se fait que quand le terminal portable a fini de changer de cellule. La valeur de la taille de la fenêtre d'émission et celle du seuil de l'algorithme *slow start* sont divisées par deux, tandis que le temps estimé d'aller-retour reste le même.

Cette approche est reprise par d'autres protocoles [Jin et al. 1999, Ramakrishnan et al. 2001] et peut se révéler non triviale à mettre en œuvre [Biaz et Vaidya 1998, Biaz et Vaidya 1999]. Elle permet de bien gérer les différents *handoffs* grâce aux informations sur la cause de la perte des paquets. Néanmoins, comme beaucoup d'autres approches [Cáceres et Iftode 1995, Samaraweera 1999, Goff et al. 2000], elle n'agit que sur les paramètres du protocole TCP et ne permet pas de prendre en compte les caractéristiques intrinsèques au lien sans-fil.

2.2.2.4 WTCP

Les approches précédentes se fondent principalement sur le taux de perte du lien sans-fil pour réémettre ou pour adapter les paramètres d'émission. Elles font la différence entre pertes dues au lien sans-fil et pertes dues à une congestion, et laissent donc intact les mécanismes de contrôle de congestion de TCP. WTCP [Sinha et al. 1999] propose une couche de transport permettant des communications de bout en bout. Cette couche possède ses propres mécanismes de fiabilité et de contrôle de congestion. La fiabilité est assurée par des acquittements sélectifs [Mathis et al. 1996, Floyd et al. 2000], l'absence de temporisateur de retransmission est remplacée par un contrôle de la fréquence des acquittements. Le contrôle de congestion se base sur le taux de transmission et sur le délai entre les paquets. Ces valeurs sont rapidement calculées à l'établissement de la connexion en utilisant l'algorithme *packet-pair* [Keshav 1991] plutôt que l'algorithme *slow start*. En maintenant un historique de ces valeurs, WTCP arrive à faire la distinction entre les pertes de paquets. Il réagit alors en adaptant le taux de transmission (i) d'une manière agressive s'il s'agit d'une congestion et (ii) d'une manière plus modérée s'il s'agit de pertes liées au lien sans-fil. Cette approche avec mécanismes intégrés permet même de développer des mécanismes de différentiation de services [Nandagopal et al. 1999]. L'inconvénient majeur est que les stations voulant communiquer doivent implanter cette couche de communication.

2.2.3 Transparence des systèmes de fichiers

NFS [Shepler et al. 2003] est un système distribué de fichiers. Il implémente un protocole permettant à des clients d'accéder à des fichiers se trouvant sur des serveurs distants. L'utilisation de ce protocole 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. Dans la version 2 de NFS, les données sont transférées par blocs de taille fixe de 8 Ko. Depuis la version 3, cette limite a été abrogée et des blocs de 32 Ko sont utilisés pour des réseaux entre 10 et 100 Mb/s et de 48 Ko pour des réseaux supérieurs à 100 Mb/s. Ces blocs sont ensuite divisés en paquets pour l'envoi sur le réseau. La perte d'un seul de ces paquets nécessite la réémission du bloc entier, ce qui produit une nette dégradation sur un lien à faible débit. Cette dégradation peut être réduite en adaptant la taille des blocs, comme dans MFS, ou en adaptant le mécanisme de contrôle de congestion. Ce mécanisme utilise un intervalle de réémission de durée exponentielle qui n'est pas approprié aux courtes coupures d'un lien sans-fil. [Dube et al. 1997] proposent un algorithme qui permet une incrémentation linéaire de l'intervalle de retransmission et, si les pertes se poursuivent, permet alors une incrémentation exponentielle. Les déconnexions peuvent également être traitées avec des mécanismes de cache, comme l'illustrent les systèmes NFS/M, MFS et Coda.

2.2.3.1 NFS/M

Le protocole NFS est particulièrement sensible aux déconnexions. En effet, il effectue sur le client une copie des blocs qui sont en cours d'utilisation. Cette copie ne reste valide que pendant une courte période de temps (*lease*) [Gray et Cheriton 1989, Kon et Mandel 1995] et, lorsque cette période expire, une requête de vérification de la validité de la copie (*renew operation*) est envoyée au serveur. Si le serveur n'est pas accessible, la copie est invalidée, le bloc demeure inaccessible et tout traitement est impossible. NFS/M [Lui et al. 1998] permet de masquer ces périodes de déconnexions grâce à l'utilisation d'un cache sur le client. La figure 2.7 présente les différents modules du client. Le *Prefetcher* s'occupe de charger à l'avance les blocs qui pourraient être accédés. Ce préchargement peut se faire selon deux techniques. La première repose sur le fait que lorsqu'un bloc d'un fichier est accédé, les autres blocs de ce fichier seront également bientôt accédés. La seconde se base sur une liste de fichiers définie par l'utilisateur. En mode connecté, la cohérence des copies avec le serveur est assurée par le *Cache Manager* avec le même système d'échéances que NFS. Lors d'une déconnexion, le *Prefetcher* et le *Cache Manager* sont désactivés et le *Proxy Server* mémorise les modifications effectuées sur les copies locales. À la reconnexion, le *Reintegrator* utilise cet historique pour répercuter les modifications sur le serveur. Les éventuels conflits sont réglés en utilisant la dernière date de modification d'une copie. Tout réintégration de données peut poser des problèmes de conflits plus complexes qui seront traités dans le paragraphe 3.1.3.

2.2.3.2 MFS

MFS [André et Segarra 1999, Segarra et André 1999] introduit un intermédiaire (*Proxy*) sur le réseau fixe entre le client mobile et le serveur NFS. La figure 2.8 présente l'architec-

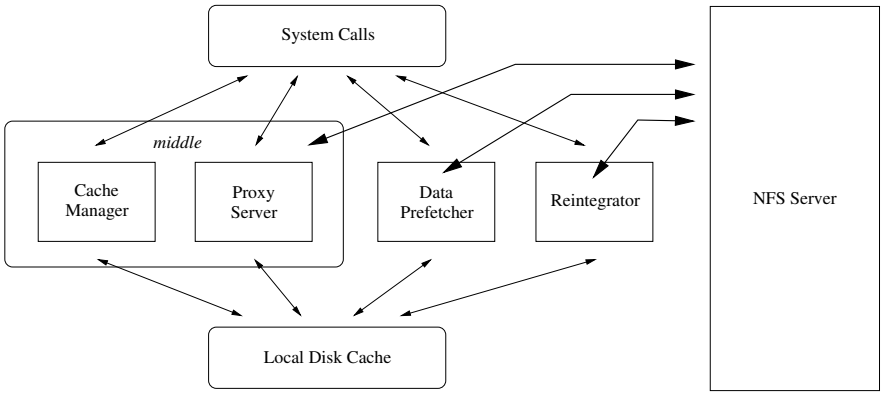


FIG. 2.7 – Architecture d'un client NFS/M

ture du système MFS. L'intermédiaire communique avec le serveur NFS à l'aide du protocole standard, mais il utilise un protocole optimisé pour dialoguer avec le client mobile. Ce protocole optimisé consiste en (i) l'utilisation de taille de blocs adaptée à la bande passante et (ii) la réduction du trafic grâce à la suppression des requêtes de vérification de validité. Le client mobile accède à ses copies locales sans vérification de validité. Cette vérification est effectuée par l'intermédiaire.

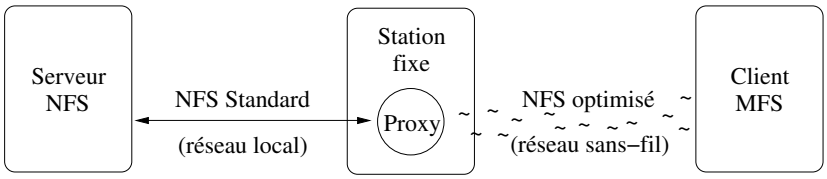


FIG. 2.8 – Architecture du système MFS

Les demandes d'accès aux blocs sont envoyées par le client à son intermédiaire. Celui-ci connaît donc le contenu de la mémoire du client et peut alors vérifier périodiquement la validité de ces blocs auprès du serveur. En mode connecté, si certaines données du serveur sont plus récentes que celles sur le client mobile, il demande au client d'invalider celles-ci et de créer des versions à jour. Le nombre d'invalidations envoyées étant dépendant du degré de partage des données par les applications, le trafic sur le réseau sans-fil est diminué lorsque ce partage est rare. En mode déconnecté, comme dans NFS/M, les modifications sont mémorisées en vue de la réintégration dans le serveur lors de la reconnexion.

2.2.3.3 Coda

Le système Coda[1] [Mummert et al. 1995, Kumar et Satyanarayanan 1995] n'utilise pas le système de fichiers NFS mais le système de fichiers AFS (*Andrew File Sys-*

[1]La version de Coda présentée correspond à la version 2. Les différents modes de fonctionnement de la version 1 [Satyanarayanan et al. 1990, Kistler et Satyanarayanan 1992] se rapprochent de ceux du système NFS/M et ne sont donc pas détaillés. Les particularités de la version 1 sont, de toute manière, reprises dans la version 2.

tem) [Satyanarayanan 1990]. À l'inverse de NFS, les serveurs AFS conservent un état de l'utilisation des fichiers et ce n'est plus aux clients de vérifier si leurs copies locales sont valides mais c'est le serveur qui met à jour les copies locales des clients à l'aide de messages d'invalidation (*callback breaks*).

Sur chaque client, un processus utilisateur appelé *Venus* gère le cache de fichiers sur le disque local. Ce processus réagit en fonction des déconnexions en adoptant trois différents modes de fonctionnement décrits dans la figure 2.9. En mode *Hoarding*, le client est considéré comme fortement connecté (*strong connection*) et le processus Venus (i) charge les données nécessaires au fonctionnement courant, (ii) précharge les données nécessaires pendant une déconnexion et (iii) applique normalement le protocole AFS de gestion de cache par invalidation. En mode *Emulating*, le client est déconnecté et le processus Venus (i) effectue alors les modifications sur les objets locaux et en garde l'historique pour la réintégration sur le serveur et (ii) assure la persistance des objets locaux pour que l'utilisateur puisse reprendre son travail même après avoir éteint son portable. Le mode *Write Disconnected* est un mode intermédiaire entre les précédents et correspondant à une connexion faible du portable (*weak connection*). Dans ce cas, les actions du processus Venus sont de (i) réaliser les modifications sur les objets locaux avec conservation de l'historique de ces modifications et (ii) continuer le préchargement de données nécessaires à une déconnexion. Ce mode permet aux clients faiblement connectés de ne pas pénaliser les clients fortement connectés en les empêchant de mettre à jour des objets en cours de réintégration.

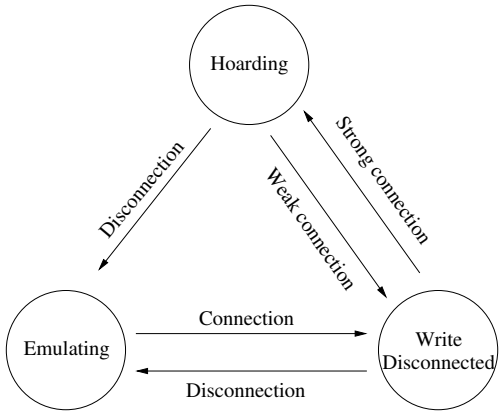


FIG. 2.9 – États et transitions du processus Venus

La réintégration sur le serveur des modifications effectuées localement ne correspond pas à un mode de fonctionnement du processus Venus, mais s'effectue de manière asynchrone en tâche de fond. Cette réintégration est active en mode *Hoarding*, active mais non primordiale en mode *Write Disconnected* (comme les modifications sont sauvegardées localement) et inactive en mode *Emulating*.

2.2.4 Transparence au niveau du protocole HTTP

Le *World Wide Web* 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 [Fielding et al. 1999, Khare et Lawrence 2000]. L'utilisation du protocole HTTP en environnements mobiles pose deux problèmes principaux : (i) l'encodage des données d'une requête HTTP s'effectue selon différents langages (SGML *Standard Generalized Markup Language* [ISO 1986] : HTML [W3C 1999a], XHTML [W3C 2000], XML [W3C 2002], 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 dû à l'établissement de la connexion et la possibilité de déclenchement des mécanismes de congestion liés à TCP. Pour surmonter ces problèmes et réduire le temps de latence des communications, différentes approches [Liljeberg et al. 1996, Housel et Lindquist 1996, Nielsen et al. 1998, Nielsen et al. 2000] proposent des mécanismes tels que des connexions persistantes, le multiplexage des requêtes ou un encodage optimisé. Ci-dessous, WebExpress illustre ces mécanismes à travers un modèle d'interception et d'optimisation du protocole HTTP.

2.2.4.1 WebExpress

Dans le but de réduire et d'optimiser le protocole de communication, WebExpress [Housel et Lindquist 1996, Chang et al. 1997, Housel et al. 1998] utilise un intermédiaire pour intercepter et contrôler les communications sur le lien sans-fil. Deux composants sont introduits entre le client Web et le serveur Web (voir figure 2.10) : le processus *Client Side Intercept (CSI)* qui s'exécute sur le terminal portable et le processus *Server Side Intercept (SSI)* qui s'exécute sur le réseau fixe.

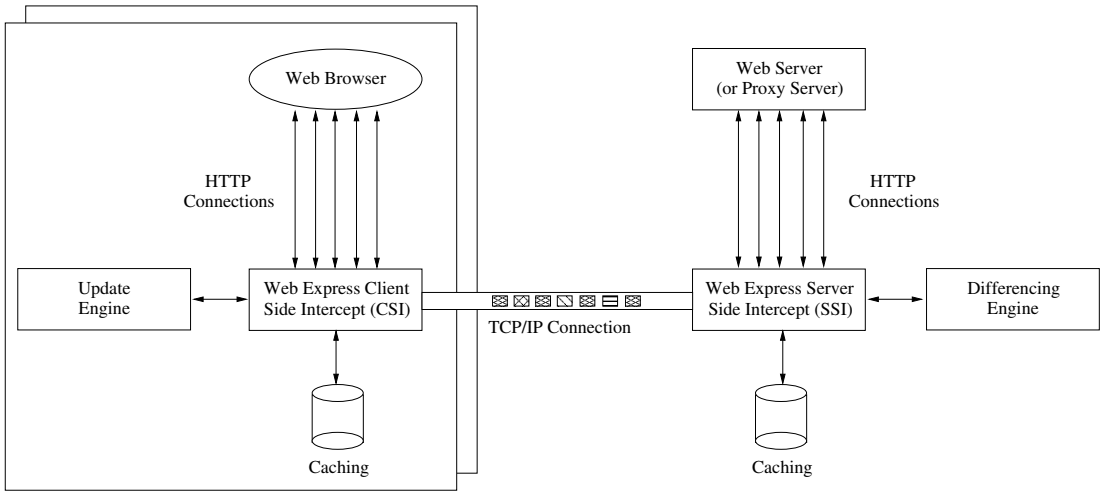


FIG. 2.10 – Modèle d'interception des requêtes HTTP par WebExpress

Du point de vue du butineur, le CSI apparaît comme un *proxy Web* et donc aucune modification du butineur n'est nécessaire[2]. Le CSI communique avec le SSI en utilisant une seule connexion TCP et un protocole HTTP optimisé. Le SSI reconstitue les requêtes HTTP provenant du CSI et les soumet au serveur Web comme si elles venaient directement du butineur. À l'inverse, les réponses renvoyées par le serveur sont reconstituées par le CSI qui les soumet au butineur comme si elles provenaient directement du serveur Web. Ces deux composants intermédiaires permettent à WebExpress (i) d'offrir la transparence aux clients et serveurs et (ii) d'effectuer les optimisations suivantes :

Techniques de cache (*Caching*) : le CSI et le SSI mettent en cache les objets graphiques et les objets HTML des requêtes les plus fréquentes. Si une requête HTTP spécifie un objet se trouvant dans le cache, celui-ci est immédiatement retourné comme réponse. L'intégrité du cache du client est assurée par invalidation avec intervalle de temps spécifié par l'utilisateur.

Techniques de différenciation (*Differencing*) : les réponses à différentes requêtes HTTP peuvent avoir un ensemble d'informations ou d'objets communs. Ceci s'observe facilement dans un même site Web où les différentes pages HTML ont une mise en page commune, ou dans un même formulaire CGI (*Common Gateway Interface*) où l'objet de base est le même et seules les informations saisies par l'utilisateur sont différentes. Le CSI et le SSI effectuent une différenciation des requêtes HTTP qui consiste à mettre en cache les objets de base communs aux requêtes. Les différences dans les réponses sont calculées par le SSI et envoyées au CSI qui les réintègre avec l'objet de base.

Réduction de protocole (*Protocol reduction*) : chaque CSI est connecté à son SSI en utilisant une seule connexion TCP/IP. Toutes les requêtes et réponses sont routées et multiplexées sur cette connexion pour éviter les surcoûts liés à l'établissement d'une connexion TCP/IP.

Réduction d'entête (*Header reduction*) : le protocole HTTP est sans état, ce qui oblige chaque requête à contenir les paramètres propres à chaque butineur. Pour un butineur donné, ces informations sont les mêmes pour toutes les requêtes. Quand une connexion est établie entre un CSI et un SSI, ces paramètres sont envoyés uniquement à la première requête et mémorisés par le SSI pour toute la durée de la connexion. Le SSI inclut ensuite ces informations dans toutes les requêtes qu'il doit remettre au serveur Web.

2.2.5 Résumé

Après avoir détaillé ces approches, il est important de voir comment elles répondent aux propriétés que l'on souhaite obtenir de notre système d'adaptation à la mobilité.

Ces approches ne sont pas génériques et sont, au contraire, spécialisées pour un type d'application bien précis. Elles obtiennent toutes de bonnes performances en optimisant un critère relatif à la liaison sans-fil (approches à stratégie contextuelle (sc)) mais de manière transparente et donc non spécialisable par l'application (approches à comportement non spécialisable (CNS)). Les stratégies contextuelles utilisées sont, pour certaines, statiques (scs),

[2]Les options de configuration du butineur doivent juste être modifiées pour inclure l'adresse IP locale du CSI comme *proxy Web*.

comme WebExpress qui optimise la taille des données transférées sur le lien sans-fil mais ne réagit à aucune variation, et, pour d'autres, dynamiques (scd), comme Mobile TCP qui modifie les paramètres de l'algorithme *slow start* dans TCP selon les pertes sur le lien sans-fil.

Ces approches étant non génériques et souvent non modulaires, elles ne sont pas facilement évolutives. Le seul moyen de les faire évoluer est de reprendre la phase de conception et de développement. Aucune évolution dynamique n'est possible. De plus, la stabilité et la prise en compte de critères relatifs aux ressources du terminal portable ou aux ressources de l'environnement ne sont pas traitées dans ces approches. Ceux-ci devraient pourtant être considérés, comme, par exemple, WebExpress qui introduit des codes intermédiaires qui effectuent des traitements utilisant des ressources (mémoire et puissance de calcul) du terminal portable et du serveur sur le réseau fixe.

2.3 Le besoin de spécialisation

Dans de nombreux cas, une adaptation n'a de sens que vis à vis de l'application. En effet, l'application est la seule à connaître la sémantique de ses données et à savoir ce qu'elle désire comme qualité de service (QdS) au niveau des ressources. Lors d'une variation, l'application est alors la plus à même pour choisir la réaction à adopter. Par exemple, face à une baisse de la bande passante, une application de flux vidéo peut décider de renégocier ses allocations de ressources, de diminuer le nombre d'images transmises par secondes ou de dégrader la qualité des images transmises. Comme décrit dans le paragraphe 2.1, cette spécialisation du comportement d'exécution peut concerner les besoins des applications mais peut aussi permettre à l'application de définir sa propre stratégie d'adaptation. Le paragraphe 2.3.1 présente les approches permettant aux applications de spécifier une qualité de service en environnements mobiles et le paragraphe 2.3.2 présente les approches permettant aux applications de spécifier leurs propres stratégies d'adaptation.

2.3.1 Spécialisation de la qualité de service

Différents systèmes permettent aux applications de spécifier leurs besoins en terme de qualité de service [Chalmers et Sloman 1999]. Cette spécification peut (i) s'effectuer dans les couches basses du système, comme dans la couche réseau pour MosquitoNet, (ii) s'effectuer dans les couches hautes, comme dans les couches application et session pour WAP, ou (iii) être fournie dans l'intégralité de la pile des protocoles, comme pour le système MOST.

2.3.1.1 MosquitoNet

Par rapport aux approches qui proposent un routage global dans Mobile IP [Montenegro 2001, Perkins et Johnson 2001], MosquitoNet [Baker et al. 1996, Zhao et al. 1998, Zhao et al. 2001] propose un routage adapté et optimisé selon la qualité de la connexion demandée et selon les caractéristiques des réseaux disponibles. Pour cela, une table de routage avec de nouveaux champs correspondant à des choix possibles de politiques (*Mobile Policy Table*) est ajoutée en complément dans la couche réseau. Ces champs

permettent d'associer l'interface de communication à utiliser ainsi que la méthode d'émission et de réception des paquets à une adresse IP et à un port.

La figure 2.11 présente les différentes méthodes d'émission et de réception des paquets. Le choix s'effectue ici selon que la mobilité est transparente ou non et selon que la communication doit traverser ou non des routeurs filtrants[3] (*ingress filtering*) [Ferguson et Senie 2000].

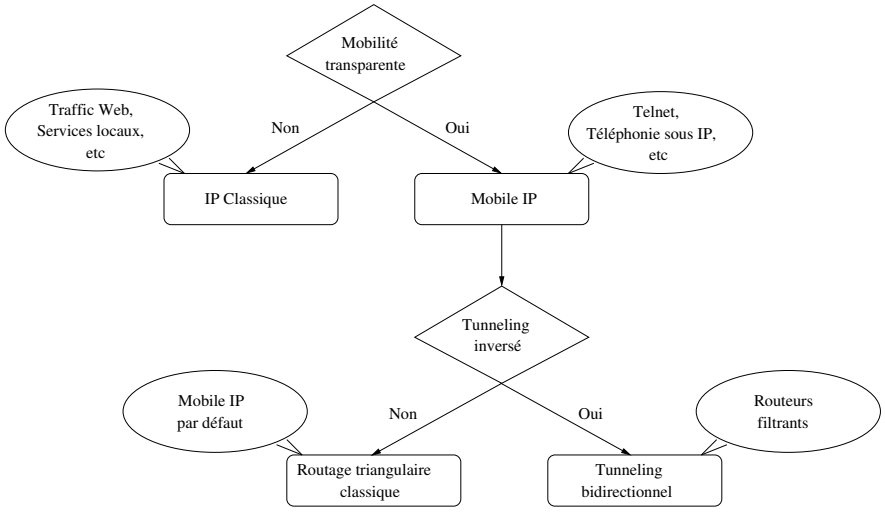


FIG. 2.11 – Méthodes d'émission de paquets supportées dans MosquitoNet

	Destination	Passerelle	Masque réseau	Numéro de port	Mobilité	Tunneling	Métrique	Interface
1	a.b.0.0	0.0.0.0	255.255.0.0	0	Non	N/A	0	eth0
2	c.d.0.0	0.0.0.0	255.255.0.0	0	Oui	Non	0	st0
3	c.d.0.0	0.0.0.0	255.255.0.0	80	Non	N/A	0	st0
4	0.0.0.0	a.b.0.1	0.0.0.0	0	Non	N/A	1	eth0
5	0.0.0.0	c.d.0.1	0.0.0.0	0	Oui	Oui	100	st0
6	0.0.0.0	c.d.0.1	0.0.0.0	80	Non	N/A	100	st0

TAB. 2.1 – Exemple de table de routage dans MosquitoNet

Le choix de l'interface de communication s'effectue selon une métrique associée à chaque interface. Cette métrique est définie ou calculée pour chaque interface suivant différents paramètres : qualité de service, coût du *handoff*, lien symétrique ou non, coût monétaire, anonymat et sécurité. Le tableau 2.1 montre un exemple de table de routage possible [Zhao et Baker 1997]. Tout trafic destiné au terminal portable peut passer par une interface

[3] De nombreux routeurs effectuent un filtrage sur les adresses sources. Ils rejettent les paquets dont l'adresse ne peut être identifiée comme correcte, c.a.d. dont le réseau d'origine ne peut pas être identifié comme étant l'adresse source.

ethernet (eth0) ou radio (st0). Dans le cas de communications locales (lignes 1 et 2), IP classique est utilisé pour l'interface ethernet et mobile IP classique est utilisé pour l'interface radio. Dans le cas de communications distantes (lignes 4 et 5), le tunneling bidirectionnel est mis en place pour l'interface radio mais cela est plus coûteux que d'utiliser l'interface ethernet. Tout trafic utilisant le port 80 (trafic Web) sur l'interface radio (lignes 3 et 6) ne nécessite pas la mise en place du support de la mobilité (la spécification du port est prioritaire sur la destination).

Lorsqu'une application veut utiliser une méthode d'émission ou de réception, elle peut utiliser les entrées génériques (*generic entries*) définies dans la table de routage, mais elle peut également définir ses propres entrées (*per socket entries*) qui passeront alors outre les entrées génériques. Une application peut aussi vouloir un accès spécifique aux interfaces de communication. En émission, une application peut lier une socket précise avec une interface définie (*bind-to-device socket*). Ainsi, différentes applications s'exécutant simultanément peuvent choisir des interfaces différentes pour envoyer des paquets. En réception, une application peut recevoir un même flot de données simultanément sur plusieurs interfaces (*flow-to-interface binding*) en s'enregistrant auprès de son agent du réseau d'attache (*Home Agent*).

Ces différentes modifications de la table de routage peuvent s'effectuer dynamiquement à l'exécution selon deux techniques. La première consiste à intervertir complètement la table de routage courante avec la nouvelle table à mettre en place. Cette technique est particulièrement utilisée lors d'un changement total d'environnement, comme un changement de réseau d'attache pour lequel des fichiers de configuration prédéfinis existent. La seconde technique permet de modifier seulement une entrée de la table. Il est, par exemple, possible de passer dynamiquement de la méthode de routage triangulaire (plus efficace) à la méthode bidirectionnelle (plus robuste) si un certain nombre d'envois se soldent par des échecs.

2.3.1.2 WAP

Soutenu par un consortium d'industriels, le WAP (*Wireless Application Protocol*) [WAP 2000, WAP 2001b] est un standard qui permet la présentation et l'accès à des informations et des services distants depuis un terminal portable sans-fil. Il repose sur une pile de protocoles légers [WAP 2001a] présentés dans la figure 2.12 :

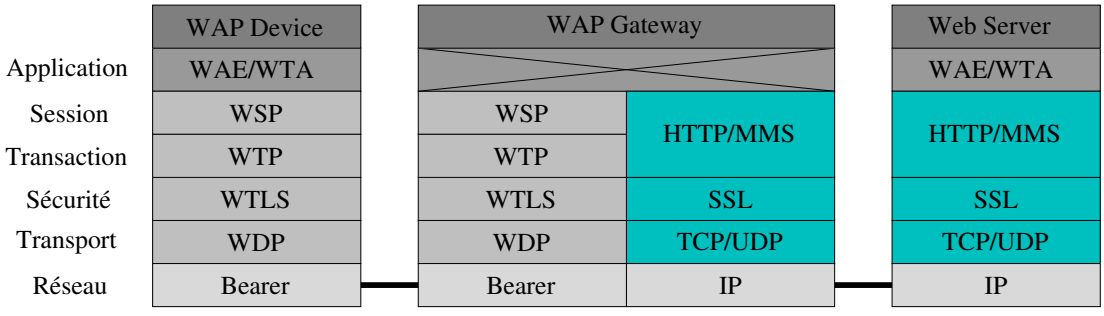


FIG. 2.12 – Pile des protocoles WAP et interconnexions avec les protocoles existants

Au niveau applicatif, le WAE (*Wireless Application Environment*) et le WTA (*Wireless Telephony Application*) fournissent un environnement de développement comprenant des interfaces

de programmation, des langages, des scripts, des styles de mise en page et la possibilité d'accès à des services existants. Ils permettent aux concepteurs de pages Web de spécifier, de manière adaptée, les informations à envoyer aux terminaux portables.

Au niveau session, le WSP (*Wireless Session Protocol*) optimise le protocole HTTP avec un protocole moins coûteux d'établissement et de rétablissement des connexions ainsi qu'un encodage plus compact des données échangées (encodage en binaire des requêtes et suppression des éléments redondants). Il étend également les fonctionnalités du protocole HTTP en offrant un support pour la qualité de service incluant : (i) des mécanismes d'exceptions permettant aux applications d'enregistrer leurs intérêts et d'être notifiées en cas de changement, (ii) des mécanismes de négociation des paramètres du protocole selon le format des données et la capacité d'affichage du client, (iii) la possibilité d'interruption d'une transaction en cours et (iv) la possibilité d'affichage d'informations de manière synchrone ou asynchrone et/ou à l'initiative du client ou du serveur (*pull/push*).

WTP (*Wireless Transaction Protocol*) est le coeur de l'architecture puisque c'est la couche qui se charge d'assurer une transmission efficace et fiable. Des mécanismes similaires à ceux de [Dube et al. 1997] y sont utilisés pour gérer les retransmissions, les acquittements et les duplications en cas de perte. La couche WTLS (*Wireless Transport Layer Security*) assure les mêmes fonctionnalités de sécurité que TLS 1.0 [Dierks et Allen 1999] avec, en plus, un protocole de négociation et un rafraîchissement dynamique des clefs de cryptage. La couche WDP (*Wireless Datagram Protocol*) assure l'indépendance du reste de la pile WAP vis à vis des divers types de réseaux de communication considérés (SMS, USSD, GHOST, etc).

Comme le montre la figure 2.12, ces différentes couches peuvent également interagir avec les protocoles existants mais avec une perte des fonctionnalités introduites dans les couches WAP.

2.3.1.3 MOST

Le projet MOST s'attaque aux problèmes de fluctuations de la qualité de service dans un environnement constitué de réseaux de communication hétérogènes (GSM, WaveLAN, Ethernet, ATM, etc). La plateforme distribuée développée [Davies et al. 1994, Friday et al. 1996, Friday et al. 1999] offre (i) la possibilité de développement de services applicatifs adaptables à l'aide d'une spécification de QdS et (ii) une adaptabilité dynamique du réseau de communication courant à l'aide de liens explicites basés sur la QdS (*QoS based explicit bindings*). L'interopérabilité est assurée par l'utilisation du système distribué ANSAware [Mayers 1995] parce que celui-ci a eu une grande influence sur le standard RM-ODP (*Reference Model for Open Distributed Processing*) [ISO 1998a, ISO 1996a, ISO 1996b, ISO 1998b].

Les mécanismes de l'adaptation se situent à tous les niveaux de l'architecture de la plateforme (voir figure 2.13) et reposent sur des mécanismes d'enregistrement et de collecte d'informations sur la QdS. Au niveau système, S-UDP (*Serial UDP*) fournit des fonctionnalités de transport similaires à celles d'UDP. Il fournit aussi la possibilité d'enregistrer des intérêts de changement sur la connexion (absence de réponse, technologie utilisée, coûts) et la surveillance et la notification (*callback address*) de ceux-ci aux applications. Au niveau de l'intergiciel lui-même, les liens explicites permettent d'établir une connexion entre deux interfaces opérationnelles avec un flot de communication respectant une QdS donnée. Chaque lien possède une

interface de contrôle (i) vérifiant les préconditions de QdS, (ii) surveillant les changements possibles de QdS et notifiant les applications dans ce cas et (iii) pouvant modifier les paramètres de QdS appliqués à ce lien si l'application le demande. Les paramètres de QdS concernés sont le débit disponible, le délai de propagation (protocole QeX [Davies et al. 1996]), le taux d'inactivité et l'atteignabilité. Dans l'application testée dans la plateforme, l'adaptabilité remonte jusqu'à l'utilisateur. La couleur de fond de l'interface graphique de certains modules applicatifs varie selon la QdS du lien explicite associé. Ce retour permet à l'utilisateur de modifier les paramètres pour obtenir la couleur désirée.

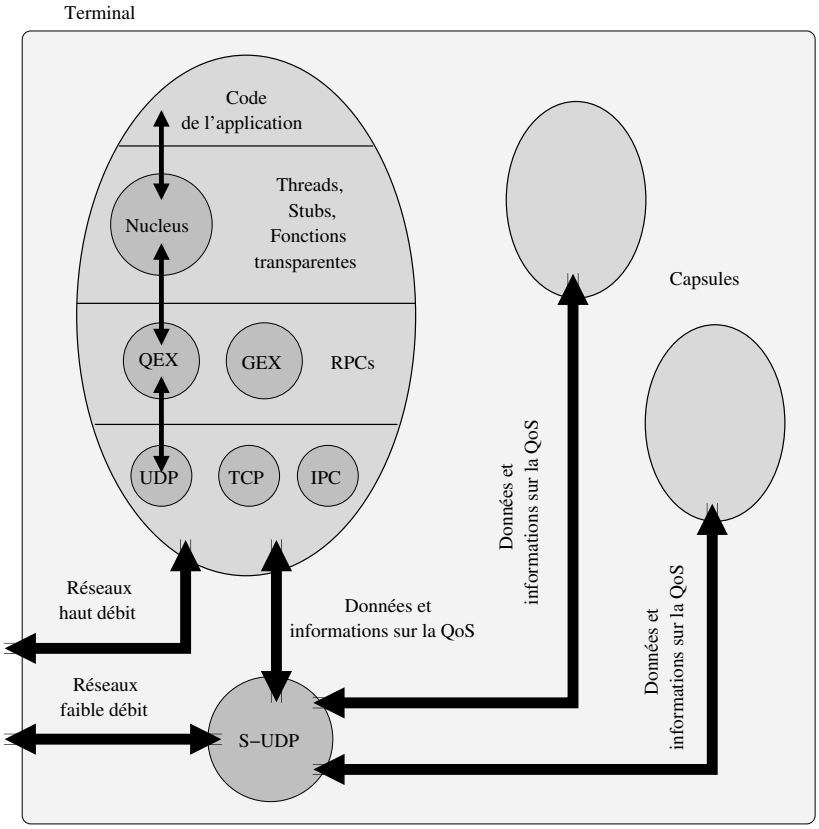


FIG. 2.13 – Architecture de la plateforme du projet MOST

2.3.2 Spécialisation de comportement d'exécution incluant des stratégies d'adaptation spécifiques

La spécialisation du système d'adaptation peut également permettre aux applications de spécifier leurs propres stratégies d'adaptation. Ces stratégies peuvent être incluses dans les couches basses du système, comme dans la couche transport pour Mobiware, ou plutôt dans les couches applicatives comme dans le système Odyssey.

2.3.2.1 Mobiware

Mobiware [Campbell 1997, Angin et al. 1998, Campbell et al. 1999] est un intergiciel qui permet la conception de services de type multimédia et Internet avec une gestion et un contrôle de la qualité de service des flots entre ces services. Il fournit principalement (i) un ensemble d'interface et d'objets CORBA qui représentent et fournissent une abstraction du réseau de communication et (ii) un ensemble de contrôleurs qui interagissent avec le médium de communication et les services distribués pour maintenir et adapter la QdS spécifiée par les applications.

Les applications utilisent une spécification de la QdS définie au niveau de la couche transport et s'appuyant sur un modèle d'interface spécifique (*Adaptive-QoS API and service model*) introduit dans [Bianchi et al. 1998]. Ces interfaces permettent à l'application de définir :

Une fonction de satisfaction selon la bande passante (*bandwidth utility function*) **:** cette fonction permet à l'application de définir une courbe exprimant son degré de satisfaction en fonction de la bande passante observable. Différents exemples de courbes sont présentés dans la figure 2.14. Cette fonction est prise en compte dans les optimisations des algorithmes d'allocation de bande passante [Bianchi et al. 1998].

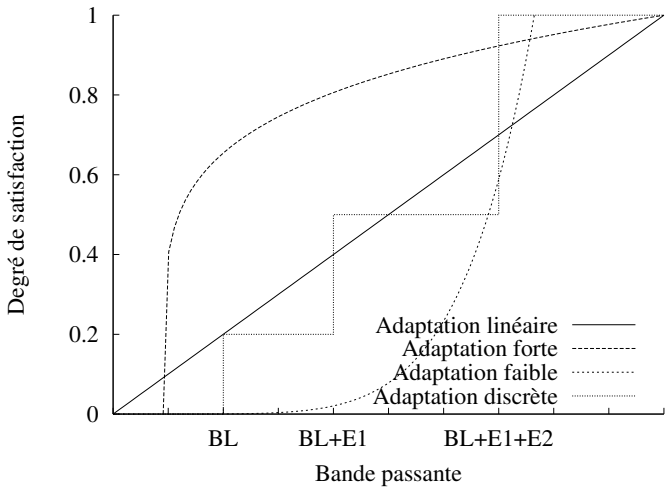


FIG. 2.14 – Fonction de satisfaction selon la bande passante dans Mobiware

Une politique d'adaptation (*adaptation policy*) **:** la politique d'adaptation permet de capturer le dynamisme d'adaptation spécifique à chaque application. Cette politique permet à l'application de contrôler le changement du degré de satisfaction (c.a.d. les déplacements le long de la courbe de satisfaction) selon les variations de la disponibilité de la bande passante. En effet, certaines applications, comme des applications vidéo multi-résolutions, peuvent vouloir limiter la fréquence des adaptations et ainsi choisir une politique d'adaptation conservatrice. Au contraire, d'autres applications, comme des applications temps-réel, peuvent vouloir saisir toutes les opportunités d'adaptation et ainsi choisir une politique d'adaptation instantanée. Cette définition de la politique d'adaptation s'effectue par la création d'un *adaptation handler* qui implante la politique propre

à l'application, ou par l'utilisation d'une des quatre politiques prédéfinies (*fast, smooth, handoff, never*).

Des préférences sur les sessions (*session preferences*) : les préférences sur les sessions permettent à l'utilisateur de classer les sessions (c.a.d. les flots audio, vidéo, etc.) entrant et sortant du terminal portable. Si la bande passante est insuffisante, cette classification est utilisée pour dégrader les flots de moindre priorité. La méthode appliquée est une dégradation par filtres actifs (*Active Filters*) [Balachandran et al. 1997]. La transformation des données sera détaillée dans le paragraphe 3.1.1.

Cette qualité de service est ensuite assurée par deux sortes d'objets déployés sur l'ensemble des terminaux portables et des points d'accès : les *Active transport objects*, qui se chargent des adaptations sur les données (dégradations selon les priorités, etc), et les *Adaptation proxies*, qui se chargent des adaptations du réseau (passages d'une interface réseau à une autre, etc). Ces deux types d'objets possèdent des contrôleurs qui, selon la bande passante observée, se chargent d'effectuer les adaptations nécessaires pour satisfaire la QdS spécifiée.

2.3.2.2 Odyssey

Odyssey [Satyanarayanan et al. 1994, Noble et al. 1997, Noble 2000] est une plateforme d'accès à des données depuis un terminal mobile. L'adaptation y est définie comme une négociation de la qualité des données en fonction des ressources disponibles. Les mécanismes proposés par cette plateforme cherchent à respecter deux propriétés : la fidélité et l'agilité.

La fidélité est le degré de correspondance entre une donnée originale et une copie de cette donnée. Elle peut avoir plusieurs dimensions qui dépendent uniquement du type de la donnée considérée. Une vidéo a, par exemple, deux dimensions : le nombre d'images par seconde et la qualité de l'image. La fidélité s'associe donc naturellement au type de la donné et les adaptations peuvent s'effectuer selon les dimensions de la fidélité. L'agilité est la capacité du système à détecter les changements de l'environnement et à les notifier aux applications, ainsi que la capacité des applications à prendre en compte ces changements pour les répercuter sur le système.

Différents composants sont mis en place au sein de la plateforme pour respecter ces propriétés (voir figure 2.15). Les composants *Wardens* implantent les méthodes d'accès aux objets d'un type défini. Ils permettent également à l'application de définir les mécanismes de fidélité associés à ce type : niveaux de fidélité et politiques d'adaptation de la donnée pour chaque niveau de fidélité (interfaces détaillées dans [Noble et al. 1995]). Les composants *Wardens* sont subordonnés au composant *Viceroy* dont la tâche la plus importante est d'effectuer la gestion des ressources du système. Le *Viceroy* reçoit toutes les requêtes d'accès aux données (interceptées par l'*Interceptor*) et, selon le niveau de fidélité des *Wardens* et la disponibilité des ressources, il peut notifier l'application que la QdS n'est plus satisfaite. Cette notification est réalisée par le composant *Upcall*. Ce composant est spécifié par l'application qui y définit : les seuils de tolérance, qui permettent de filtrer les notifications non pertinentes, et le nom de la politique d'adaptation de la fidélité, qui sera appelée dans le cas de notifications pertinentes. Dans ce dernier cas, la politique d'adaptation de la fidélité peut alors décider de modifier la fidélité en agissant sur les composants *Wardens*.

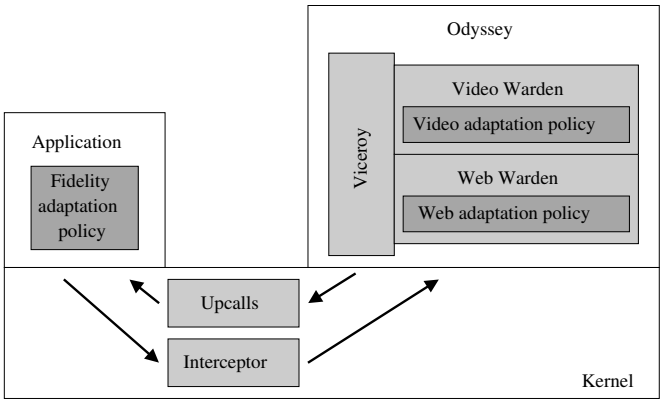


FIG. 2.15 – Architecture du système Odyssey

On peut remarquer que toutes les politiques d'adaptation ne correspondent pas à une spécialisation du système. La politique d'adaptation de la fidélité se trouve, en effet, dans l'application. Dans ce cas, la frontière qui sépare l'application du système d'adaptation est toutefois difficile à délimiter comme le montre l'exemple d'adaptation de Netscape [Noble et al. 1997, Noble et Satyanarayanan 1999]. Dans cet exemple, un élément d'interposition entre Netscape et Odyssey implante la politique d'adaptation de la fidélité. Cet élément pourrait très bien être inclus dans le système d'adaptation.

2.3.3 Résumé

Ces approches permettent aux applications d'avoir conscience et de pouvoir réagir à la mobilité. Elles ne permettent toutefois pas de couvrir tous les objectifs fixés.

La généricité est assurée par la spécification d'une qualité de service. Cette spécification rend le système d'adaptation indépendant des applications qui peuvent l'utiliser[4]. Elle intègre différents paramètres caractérisant l'environnement mobile (interface de communication, *handoff*, bande passante, etc) (approches à stratégie contextuelle (sc)) et peut être spécifiée par l'application au sein de stratégies d'adaptation (approches à comportement spécialisable (CS)). Pour pouvoir assurer cette QdS spécifiée par l'application, les approches présentées possèdent toutes un système de détection et de notification (scd) permettant à l'application de réajuster dynamiquement la QdS selon les besoins. Cette spécialisation dynamique du comportement (CSD) ne concerne que la qualité de service, la spécialisation de stratégies d'adaptation est statique (CSS).

Ces approches ayant une certaine généricité et modularité, elles peuvent assez aisément évoluer. Néanmoins, les adaptations sont prévues statiquement dans le système et ne peuvent facilement évoluer dynamiquement. Par exemple, dans Odyssey, l'ajout d'un nouveau type de données (avec ses niveaux de fidélité et sa politique d'adaptation) nécessite une refonte importante du système. Comme pour les approches transparentes, la prise en compte de critères

[4]Dans le cas de spécialisation de stratégies, cette généricité ne peut plus être totale puisqu'une partie du système d'adaptation dépend de l'application. La généricité peut toutefois s'appliquer à des classes d'applications ou de données, comme dans Odyssey.

relatifs aux ressources du terminal portable ou aux ressources de l'environnement n'est pas abordée. Par contre, certaines de ces approches laissent à l'application le soin de définir la stabilité des adaptations face aux changements. Les techniques utilisées (fonctions, seuils) endiguent l'effet « boomerang » mais uniquement pour les adaptations au sein de l'application. La stabilité n'est pas gérée de manière globale et, donc, une adaptation au sein d'une application peut très bien être considérée comme stable par cette application mais induire des perturbations sur des applications concurrentes ou sur le système.

2.4 Adaptation et réflexivité

La *réflexivité* est la capacité d'un système à se représenter, s'observer et à agir sur lui-même [Smith 1982, Maes 1988, Maes et Nardi 1988]. La représentation du système se fait par le processus de *réification*. Il consiste à faire correspondre les entités fonctionnelles du système (niveau de base) avec des entités chargées de décrire et de gérer le fonctionnement lui-même (niveau méta). L'*introspection* donne la possibilité au système de connaître son état interne, lui permettant de pouvoir raisonner et de prendre des décisions consécutivement. L'*intercession* est le mécanisme permettant au système d'adapter son comportement en modifiant son propre fonctionnement.

Le niveau méta s'occupant de la gestion du fonctionnement du système, plusieurs aspects non fonctionnels peuvent y être implantés. La persistance [Paepcke 1991, Stroud et Wu 1994], la localisation d'objets [Okamura et Ishikawa 1994], l'atomicité [Stroud et Wu 1995], la réplication [Kleinöder et Golm 1996b], la tolérance aux fautes [Fabre et Pérennou 1998, Killijian et Fabre 2000], la composition [Olivia et Buzato 1998], la sécurité [Welch et Stroud 1998, Welch et Stroud 2000], les mécanismes d'exceptions [Welch et al. 2001] sont notamment des aspects faisant l'objet de recherches en ce sens.

Les approches réflexives assument qu'il y a un but à l'adaptation (pourquoi ?), comme de s'adapter aux variations de l'environnement d'exécution, mais elles ne s'y attachent pas autant qu'aux techniques elles-mêmes : sujet (qui ?), moment (quand ?), mécanismes (comment ?) de l'adaptation. Une adaptation peut alors porter sur différents sujets :

Entités : comme une méthode dans [Blair et al. 1998, Costa 2001], un objet dans Aperios[5] [Yokote 1992, Itoh et al. 1995, Yokote 1999] ou DART [Raverdy et Lea 1998, Raverdy et al. 1998] ou un composant dans OOPP (*Open-ORB Python Prototype*) [Andersen et al. 2000].

Liaison entre entités : comme celles entre entités du niveau de base dans mChaRM [Cazzola et Ancona 2000, Cazzola 2000] ou MICADO [Berger 2001], ou comme celles entre entités du niveau de base et entités du niveau méta dans OpenCorba [Ledoux 1999],

Un ensemble d'entités : comme dans Jonathan [Dumant et al. 1998] ou 2K/DynamicTAO [Kon et al. 2000b].

Elle peut également avoir lieu à différents moments et selon différents mécanismes :

[5] Préalablement nommé Apertos.

À la compilation : tout comme chaque sujet (entité, liaison ou ensemble) possède son code source, la structure elle-même du sujet est décrite à l'aide d'un code source. Par exemple, en Java, les classes et les méthodes sont décrites par d'autres classes (métaclasses). La compilation utilise cette description pour la génération du code compilé. CLOS [Bobrow et al. 1988, Kiczales et al. 1991], OpenC++ [Chiba 1995, Chiba 1998b], Iguana [Gowing et Cahill 1996], OpenJava [Chiba et Tatsubori 1998, Tatsubori et al. 2000] sont des exemples de langages réflexifs.

Au chargement : dans le cas d'une entité ou de liaison, les outils de chargement sont modifiés pour autoriser une altération du code compilé, comme dans Kava [Welch et Stroud 1999, Welch et Stroud 2001] et Javassist [Chiba 1998a, Chiba 2000]. Dans le cas d'un ensemble d'entités, c'est le déploiement qui est modifié, comme dans Jonathan [Dumant et al. 1998].

À l'exécution : l'accès dynamique au niveau méta peut être réalisé de deux manières. L'interception peut s'opérer à l'aide de *proxies* mis en place à la création des sujets, comme dans Reflective Java [Wu et Schwiderski 1997], Proactive [Caromel et al. 1998], RAM [Bouraqadi-Saâdani et al. 2001a] ou Reflex [Tanter et al. 2001]. L'interception peut également être faite par le support d'exécution, comme dans MetaXa [Kleinöder et Golm 1996a, Golm et Kleinöder 1997, Golm 1998], Guaraná [Olivia et al. 1998, Olivia et Buzato 1999], Iguana/J [Redmond et Cahill 2000] ou Correlate [Truyen et al. 2001].

Les systèmes RAM et 2K/DynamicTAO sont examinés dans de plus amples détails ci-dessous parce qu'ils proposent des mécanismes qui peuvent être utilisés pour implanter des stratégies de gestion des ressources (voir paragraphe 3.2). RAM propose une infrastructure pour une mobilité forte du code. 2K/DynamicTAO est un intergiciel de communication qui offre des mécanismes permettant la (re)configuration dynamique de l'architecture des composants d'une application.

2.4.0.1 RAM

Pour la construction d'applications distribuées, RAM (*Reflection for Adaptable Mobility*) [Bouraqadi-Saâdani et al. 2001a] considère la mobilité du code d'une application comme un aspect non fonctionnel qui peut donc se trouver au niveau méta et être greffé sur une application.

Les objets sont groupés en *cluster* ce qui constitue l'unité de mobilité. Comme le montre la figure 2.16, en plus des objets de base, trois méta-objets principaux sont associés à chaque cluster :

Une méta-façade (*meta-facade*) : la méta-façade initialise et permet l'accès à l'infrastructure du cluster et, en particulier, aux méta-objets décrivant les politiques attachées à ce cluster (file d'attente des messages, politique de migration, etc).

Des références instanciées (*reified references*) : les références instanciées sont des méta-objets qui représentent des références locales ou distantes à d'autres clusters. Elles sont créées par la méta-façade et peuvent donc être retrouvées et mises à jour par celle-ci.

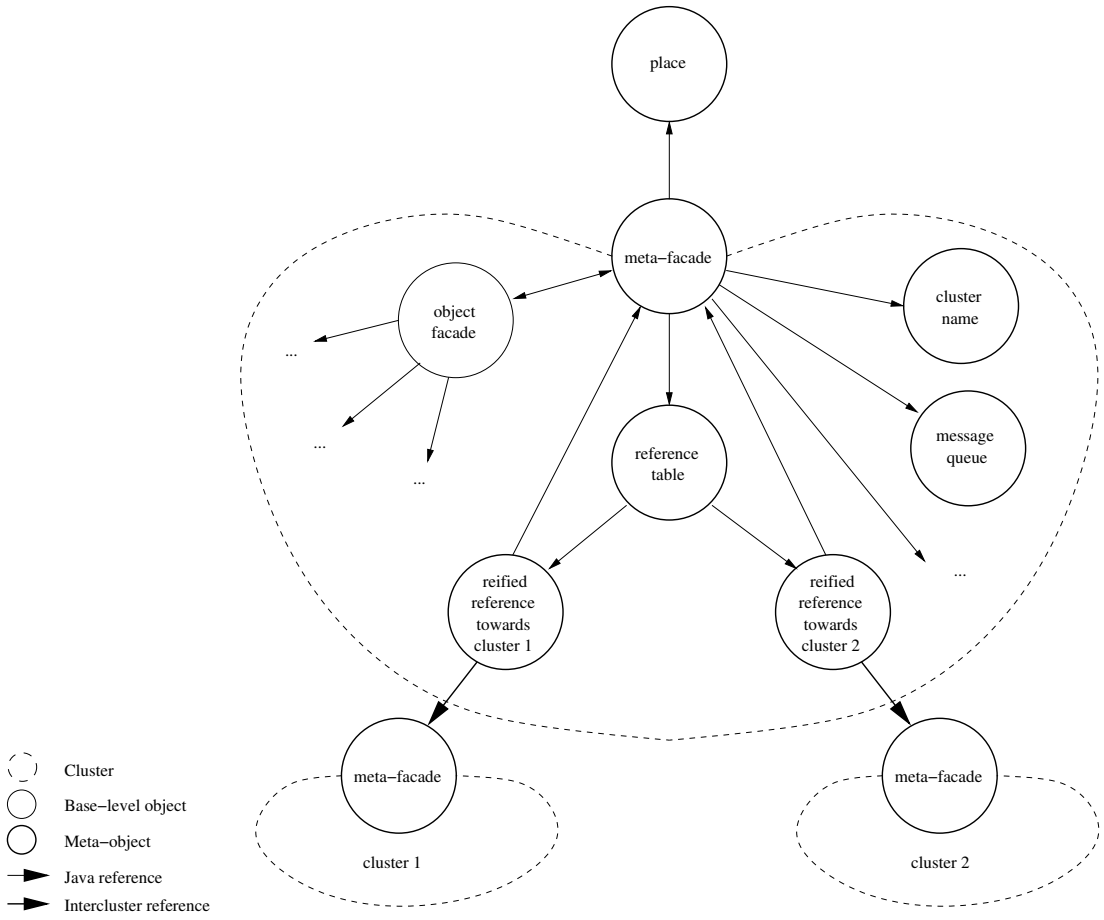


FIG. 2.16 – Description d'un cluster dans RAM

La table de référence (*reference table*) : toutes les références instanciées sont répertoriées dans la table de référence qui assure alors l'unicité d'une référence dans un cluster. Cette table est utilisée par la méta-façade pour retrouver et mettre à jour les références (par exemple lors d'une migration).

Un certain nombre de politiques sont instanciées par défaut lors de la création de la méta-façade. La politique par défaut de migration est d'effectuer une migration systématique quand la communication entre deux clusters est distante. La politique par défaut de mise à jour des références est de trouver et de mettre à jour les références par le réseau. La politique par défaut de gestion des processus d'exécution est la mobilité forte et la politique par défaut de communication est l'échange synchrone de messages. Ces politiques peuvent être spécialisées différemment par une spécialisation différente de la méta-façade.

Le prototype implanté en Java présente également la propriété intéressante de ne pas modifier la machine virtuelle Java mais d'utiliser des mécanismes de réflexivité et de transformation de code pour assurer la mobilité. La portabilité de Java est ainsi préservée.

2.4.0.2 2K/DynamicTAO

2K/DynamicTAO [Kon et al. 2000b] est un intergiciel de communication réflexif (*reflective Object Request Broker*) s'appuyant sur CORBA. Il exporte une interface permettant (i) le transfert de composants dans un système distribué, (ii) le chargement et le déchargement d'un composant sur l'intergiciel pendant l'exécution, (iii) la surveillance (par introspection) et la modification (par intercession) de l'état de l'intergiciel lui-même.

La réflexivité est mise en place dans 2K/DynamicTAO par un ensemble de *Component Configurators* [Kon et Campbell 2000]. Un *Component Configurator* maintient les dépendances entre un composant et les autres composants du système. Comme le montre la figure 2.17, chaque composant s'exécutant sur 2K/DynamicTAO contient donc une instance d'un *Component Configurator* appelé *Domain Configurator* chargé de maintenir les références entre l'intergiciel et les différents composants dont celui dépend (*Servant Configurators*). L'intergiciel possède lui-même une instance de *Component Configurator* appelé *TAO Configurator* qui maintient les stratégies de l'intergiciel de communication.

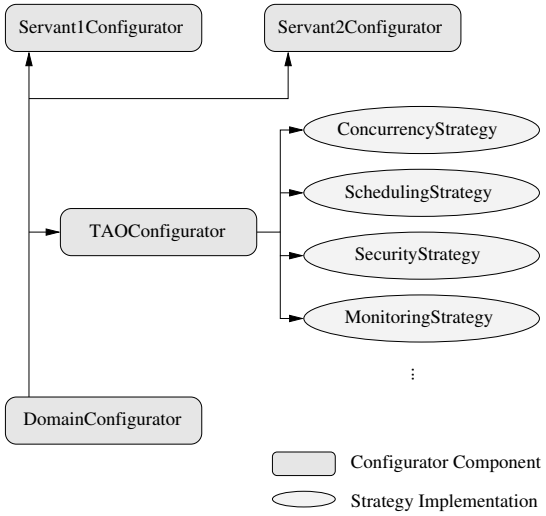


FIG. 2.17 – Structure de réification dans 2K/DynamicTAO

Le *TAO Configurator* contient des points d'interception (*mounting points*) qui permettent de dériver les appels effectués par les composants vers les implantations spécifiques des stratégies de l'intergiciel. Les implantations des stratégies peuvent être changées pendant l'exécution en utilisant un mécanisme de chargement d'implantation (méthode *load_implementation*) et d'attachement de cette implantation à la stratégie voulue (méthode *hook_implementation*). Cet attachement s'effectue en respectant le pattern *Memento* [Gamma et al. 1995] qui permet de récupérer l'état de l'ancienne implantation et d'instancier la nouvelle implantation avec celui-ci.

Différents exemples de stratégies ont été illustrés à l'aide de cette structure. Ces exemples sont eux-mêmes implantés comme des composants greffés sur l'intergiciel de communication à l'aide de l'interface. Les *Reconfiguration Agents* [Kon et al. 1999] listent les implantations possibles, construisent le graphe de dépendances et permettent sa reconfiguration.

Le *2K Monitoring Service* [Mao 1999] permet de spécifier les composants qui doivent être surveillés. Cette spécification inclue des paramètres comme les requêtes entrantes et sortantes de ce composant mais aussi des paramètres comme la fréquence de ces requêtes. Le *SecureAgentBroker* [Kon et al. 2000a] fournit le contrôle d'accès, l'authentification et le cryptage selon le modèle RBAC (*Role-Based Access Control*) [Sandhu et al. 1996].

2.4.1 Résumé

Les approches réflexives s'attachent plus aux mécanismes d'adaptation en eux-mêmes qu'aux bonnes stratégies d'un environnement particulier, comme les environnements mobiles. Elle ne satisfont donc pas toutes les propriétés que l'on souhaite obtenir de notre système.

Ces approches offrent aux applications des mécanismes d'introspection et d'intercession. Ceux-ci sont indépendants de la classe d'application ou du type des données. Les mécanismes d'introspection sont généralement utilisés pour implanter des stratégies contextuelles dynamiques (scd). Les parties du système modifiables par intercession définissent la granularité de la modularité (approches à comportement spécialisable (CS)). Cette granularité peut être fine, comme avec la modification d'une méthode d'un objet, ou à gros grain, comme avec l'ajout ou le retrait dans une architecture d'objets. Selon le but recherché, cette spécialisation peut être réalisée statiquement (CSS), comme avec les optimisations de compilation, ou dynamiquement (CSD), comme avec la modification de méthodes au chargement ou pendant l'exécution.

Ces systèmes étant grandement génériques et modulaires, ils peuvent facilement évoluer. Cette évolution est facilitée par le découpage entre niveau de base et niveau méta. En effet, la modification de la description du système peut alors s'effectuer indépendamment de la manière dont celui-ci est implanté (par exemple pour ajouter de nouveaux aspects non-fonctionnels).

Toutefois, ces approches ne s'attachent pas à un environnement particulier et ne proposent donc pas de mécanismes de gestion des environnements mobiles. Les stratégies proposées sont donc contextuelles, mais avec un contexte non défini ne permettant pas d'augurer de leurs performances. La gestion de la stabilité n'est également pas complète. Par exemple, dans 2K/DynamicTAO, le transfert d'informations d'une implantation à l'autre des stratégies du *TAO Configurator* s'effectue de manière cohérente en respect le pattern *Memento*. Grâce aux dépendances entre *Component Configurator*, le changement peut être propagé, si nécessaire, aux composants adéquats. Cette propagation ne permet de régler que des problèmes de synchronisation d'adaptations et ne permet pas de répondre aux effets « boomerang » et « domino ».

2.5 Discussion

Le tableau 2.2 résume les propriétés liées aux différentes approches précédemment énoncées. Globalement, à partir de ces propriétés, ces approches peuvent être caractérisées par rapport à la puissance d'expression de l'adaptation et par rapport à la facilité et l'adéquation de leur utilisation en environnements mobiles, comme le présente la figure 2.18.

Les approches transparentes proposent des stratégies d'adaptation parfaitement optimisées aux environnements mobiles mais celles-ci ne peuvent être exprimées que de manière statique et non évolutive. Dans les approches applicatives, les stratégies d'adaptation sont proposées par le système et peuvent être spécialisées par les applications. Elles sont donc optimisées dans

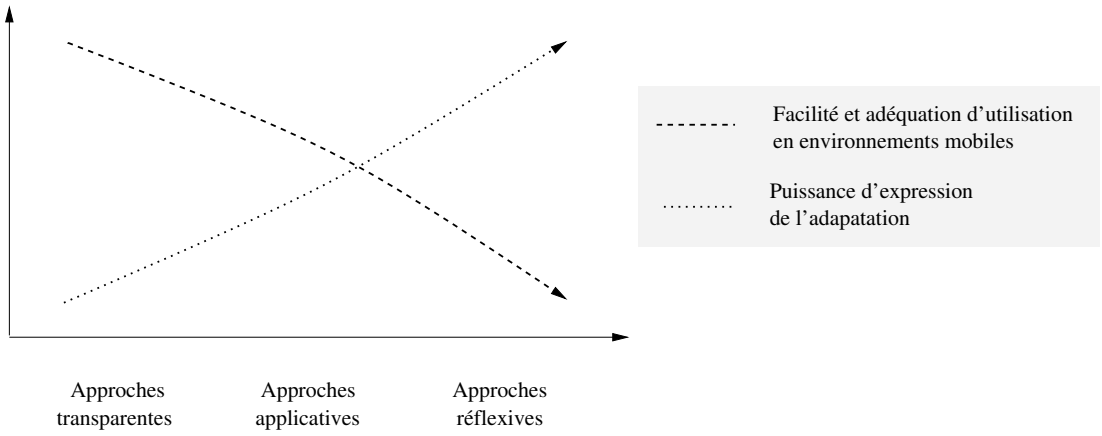


FIG. 2.18 – Comparatif des approches transparentes, applicatives et réflexives selon la puissance d'expression de l'adaptation et la facilité d'utilisation en environnements mobiles

la mesure où l'application connaît bien l'environnement mobile. La puissance d'expression de l'adaptation permet de modifier dynamiquement certains aspects comme la qualité de service mais le changement de stratégies reste statique. Les approches réflexives offrent une grande puissance d'expression de l'adaptation en permettant d'en définir le grain, de modifier tout le système dynamiquement. Par contre, elles n'offrent aucune prise en charge de l'environnement mobile et les différents mécanismes de gestion des environnements mobiles doivent être complètement spécialisés.

Chacune de ces approches possède des caractéristiques intéressantes vis à vis des propriétés souhaitées mais aucune n'est pleinement satisfaisante du point de vue de l'adaptation ou de la gestion des environnements mobiles. Un système d'adaptation en environnement mobile doit posséder des caractéristiques appartenant à ces différentes approches (cellules grisées du tableau 2.2) :

Généricité : *un système d'adaptation doit offrir des points d'entrée permettant à n'importe quelles applications de l'utiliser*. Ces points d'entrée peuvent être mis en place dans un cadre de conception qui peut combiner des interfaces de qualité de service avec des mécanismes d'introspection et d'intercession.

Modularité : un système d'adaptation doit être suffisamment bien découpé pour qu'une application puisse utiliser ou spécialiser la partie souhaitée. Le minimum à offrir est donc la possibilité de spécialisation des stratégies. Cela n'offre toutefois la possibilité aux applications de ne spécialiser que les stratégies prévues. À l'opposé, les approches réflexives permettent de modifier entièrement le système d'adaptation. *Un système d'adaptation doit donc définir sa granularité d'adaptation*.

Adaptabilité :

Prise en compte de l'environnement : les stratégies d'adaptation du système doivent tenir compte de l'environnement mobile. Chaque approche (de la catégorie transparente ou applicative) examine un paramètre bien particulier de l'environnement

mobile comme le changement de cellule, les pertes sur le lien sans-fil ou les déconnexions, mais aucune ne prend en compte la globalité des paramètres. *Une caractérisation de l'ensemble des paramètres définissant un environnement mobile est nécessaire. Une caractérisation de l'utilisation de ces paramètres dans les stratégies d'adaptation est également nécessaire*.

Prise en compte de l'application : les stratégies d'adaptation du système doivent tenir compte de l'application. Cette spécialisation possible doit utiliser les mécanismes génériques définis dans le système pour permettre à l'application de définir sa propre qualité de service ou d'insérer ses propres stratégies d'adaptation. *Une caractérisation des actions que peut effectuer l'application pour spécialiser le système est nécessaire*.

Évolutivité : la généricité et la modularité doivent être suffisamment importantes pour permettre des modifications non prévues. Le seul moyen pour effectuer cela dynamiquement est d'avoir une description du système modifiable dynamiquement. *Une description du système ainsi qu'une caractérisation des actions que l'on peut effectuer sur celle-ci est nécessaire*.

Dynamicité : à chaque fois qu'un utilisateur se déplace dans un nouvel endroit[6], ou installe une nouvelle application[7], il n'est pas envisageable qu'il doive arrêter application et système, modifier ceux-ci et les relancer pour prendre en compte ces nouveaux éléments. *Les mécanismes implantant les différentes actions d'adaptation doivent donc être dynamiques*.

Efficacité :

Performances : les performances de ces différentes approches résultent directement des stratégies choisies et implantées. Ces stratégies optimisent (i) uniquement les paramètres du terminal portable et non les paramètres de tout l'environnement mobile et (ii) uniquement en effectuant des réductions de coûts et non en effectuant une répartition des coûts sur l'environnement. *Un système d'adaptation en environnement mobile doit donc posséder un système de gestion des données et des ressources ayant une stratégie (i) globale (par opposition à locale), pour mettre à profit l'environnement et (ii) spécialisée, pour prendre en compte les spécificités de l'environnement mobile*.

Stabilité : la stabilité de ces différentes approches résulte également des stratégies choisies et implantées. Ces stratégies s'appliquent uniquement aux adaptations au sein d'une application et ne permettent pas de stabiliser l'ensemble des adaptations du système. *Un système d'adaptation doit posséder un système de coordination des adaptations ayant une stratégie (i) globale, pour permettre d'endiguer les effets « boomerang » et « domino » et (ii) spécialisée, pour permettre de limiter les adaptations en fonction des paramètres de l'environnement mobile et d'éviter ainsi d'avoir un nombre important d'adaptations nuisant aux performances du système*.

[6] Un nouvel environnement mobile présente des caractéristiques différentes de ce que le système d'adaptation courant peut gérer.

[7] Une nouvelle application utilise différemment le système d'adaptation par rapport aux applications courantes.

	Approches transparentes	Approches applicatives	Approches réflexives
Généricité	non génériques	génériques (en utilisant les interfaces de Qualité de Service)	génériques (en utilisant les mécanismes de la réflexivité : interfaces d'introspection et d'intercession)
Modularité	non modulaires	modulaires (au niveau des stratégies)	modulaires (pour l'ensemble du système et selon la granularité définie)
Adaptabilité			
↪ *Prise en compte de l'environnement*	adaptées aux environnements mobiles (sc)	adaptées aux environnements mobiles (sc)	adaptables à différents environnements (sc) mais non utilisées en environnements mobiles
↪ *Prise en compte de l'application / Spécialisation*	non adaptées aux applications (CNS)	adaptées aux applications qui tiennent compte des environnements mobiles (CS)	adaptables aux applications (CS) mais non utilisées par des applications tenant compte des environnements mobiles
Évolutivité	non évolutives	statiquement évolutives	dynamiquement évolutives
Dynamicité	stratégies contextuelles statiques (scs) ou dynamiques (scd)	stratégies contextuelles statiques (scs) ou dynamiques (scd) à comportement spécialisable statiquement (QdS et stratégies) (CSS) ou dynamiquement (QdS uniquement) (CSD)	stratégies contextuelles statiques (scs) ou dynamiques (scd) à comportement spécialisable statiquement (CSS) ou dynamiquement (CSD)
Efficacité			
↪ *Performances*	optimisées mais uniquement pour les ressources du terminal portable et non pour celles de l'environnement mobile	optimisées mais uniquement pour les ressources du terminal portable et non pour celles de l'environnement mobile	non optimisées
↪ *Stabilité*	non stables	stables mais uniquement localement (application) et non globalement (système)	mécanismes permettant une cohérence globale des adaptations mais non utilisés pour une stabilité globale du système

TAB. 2.2 – Comparatif des propriétés des approches transparentes, applicatives et réflexives

Chapitre 3

Stratégies d'adaptation en environnements mobiles

Les performances d'une application utilisant un système d'adaptation dépendent des stratégies d'adaptation implantées dans celui-ci. Ces stratégies peuvent agir sur deux aspects d'une application : les données manipulées et les ressources utilisées.

Le paragraphe 3.1 détaille les stratégies de gestion des données et le paragraphe 3.2 détaille les stratégies de gestion des ressources. Les systèmes présentés dans ces paragraphes illustrent ces stratégies de manière indépendante mais celles-ci ne sont pas exclusives et sont implantées simultanément dans certains systèmes.

3.1 Stratégies de gestion des données

Les stratégies de gestion de données considèrent les données manipulées par les applications. Elles leur appliquent différentes techniques prenant en compte les paramètres de l'environnement mobile, permettant ainsi d'en optimiser les performances. Les paragraphes suivants détaillent ces techniques : le paragraphe 3.1.1 examine les transformations de données, le paragraphe 3.1.2 le préchargement des données et le paragraphe 3.1.3 la réplication et la cohérence des données.

3.1.1 Transformation des données

On dit qu'il y a transformation d'une donnée lorsqu'il y a une différence entre la donnée initialement demandée et la donnée reçue. Les deux techniques de transformations appliquées en environnement mobile ont pour but de diminuer la taille des données :

Compression : on dit qu'il y a compression lorsque le contenu de la donnée reçue est le même que celui de la donnée demandée mais qu'il y a un changement dans la structure de la donnée (encodage) induisant une diminution de taille. Les algorithmes LZ* [Ziv et Lempel 1977, Welch 1984] sont des exemples d'algorithmes de compression.

Dégradation : on dit qu'il y a dégradation lorsque la structure de la donnée reçue est la même que celle de la donnée demandée mais que le contenu de la donnée reçue est un sous-ensemble du contenu de la donnée demandée. Transformer une image couleur en image noir et blanc est un exemple de dégradation.

Ces deux techniques ne sont pas incompatibles et peuvent être appliquées simultanément. L'algorithme JPEG [ISO 1994] est un exemple d'algorithme de compression avec perte.

Ces techniques sont appliquées dans plusieurs systèmes de gestion de la mobilité cités précédemment (WebExpress, Odyssey, Mobiware, etc) et sont illustrées ci-dessous par les systèmes Mowgli pour la compression, BARWAN pour la dégradation, MOWSER et l'approche de Chalmers et al. pour une dégradation négociée.

3.1.1.1 Mowgli

Mowgli [Kojo et al. 1996, Alanko et al. 1997] part du même principe que I-TCP et découpe la connexion en deux parties : une pour la communication filaire et l'autre pour la communication sans-fil (voir figure 3.1).

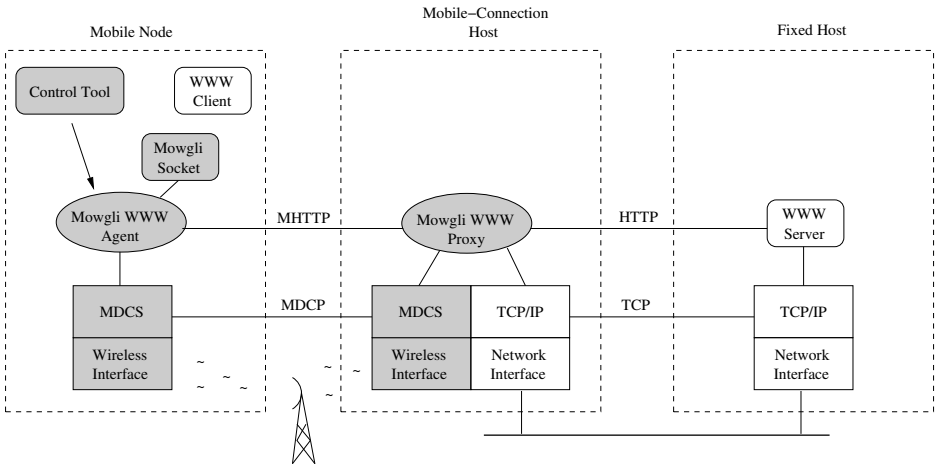


FIG. 3.1 – Architecture du système Mowgli

Deux médiateurs se chargent de la communication avec le niveau applicatif : *The Mowgli Agent* sur le terminal mobile et *The Mowgli Proxy* sur la station fixe établissant la communication sans-fil. Ils fournissent des services similaires à TCP et UDP. Ils utilisent tout deux *The Mowgli Data Channel Service (MDCS)*. Le MDCS offre deux types de canaux bidirectionnels pour le transfert de données sur le lien sans-fil : les *stream channels*, avec les mêmes fonctionnalités que TCP, et les *message channels*, avec les mêmes fonctionnalités qu'UDP. Tous les canaux sont indépendants et chaque canal possède ses propres mécanismes de multiplexage et d'ordonnancement.

La compression est appliquée au niveau du protocole de communication MHTTP entre *The Mowgli Agent* et *The Mowgli Proxy* [Liljeberg et al. 1996] et porte sur les données propres au protocole (entêtes) et sur les données véhiculées par le protocole. À la différence du protocole HTTP, MHTTP n'est plus codé en ASCII (lisible et compréhensible par un programmeur)

mais est codé en binaire. Cet encodage permet de diviser par cinq la taille des entêtes. Les données véhiculées sont également compressées de manière générique ou selon le type de la donnée. Pour les données génériques, l'algorithme *splay-prefix* [Grinberg et al. 1995] est appliqué parce qu'il est simple, rapide et qu'il permet de compresser des flux. Pour chaque donnée d'un type défini, un algorithme de compression spécifique est appliqué. Par exemple, pour les images, l'algorithme GIF peut être appliqué. L'utilisateur a toujours la possibilité d'activer ou non les algorithmes de compression.

3.1.1.2 BARWAN / Pythia / GloMop

Le système BARWAN [Katz et al. 1996, Katz et Brewer 1996] propose une architecture permettant aux applications de s'adapter à la qualité de service des réseaux sans-fil sous-jacents (principalement en terme de bande passante et de latence). Cette approche se base sur le fait qu'il existe des réseaux sans-fil très hétérogènes en terme de taille et de caractéristiques (*regional-area*, *wide-area*, *campus-area*, *in-building*, *in-room*) et que c'est le passage d'un réseau à un autre auquel il faut s'adapter.

Le système BARWAN est centré sur les points d'accès (*gateway-centric*). De la même manière que dans Mowgli, chaque mobile dialogue avec un *proxy*, responsable des communications entre le réseau sans-fil et le réseau fixe. Celui-ci défini les paramètres de la transmission des données. Cette transmission est fortement typée, ce qui permet au *proxy* d'associer un *distiller* à chaque type de donnée. Plusieurs implantations de *proxy* (Pythia [Fox et Brewer 1996], GloMop [Fox et al. 1996]) ont permis de tester plusieurs techniques :

Distillation et raffinement (*distillation and refinement*) : la distillation et le raffinement sont deux termes introduits par cette approche. La distillation est une compression avec perte préservant la sémantique de la donnée. Le raffinement est la capture de bonne qualité d'une partie de la donnée. Ces deux types de dégradation s'effectuent suivant différents axes caractérisant la donnée (voir ci-dessous).

Axes de variations et axes de dégradation : pour déclencher la distillation et le raffinement, plusieurs axes de variations sont pris en compte : (i) les paramètres de la connexion réseau (obtenus par l'utilisateur, par un profil réseau ou par une surveillance automatisée), (ii) les caractéristiques de l'affichage du terminal (en terme de résolution et de couleurs) et (iii) le type de données requis par l'application. Selon ces axes de variations, la distillation et le raffinement peuvent avoir lieu selon différents axes de dégradation. Ceux-ci sont propres aux données, comme le montre le tableau 3.1. Pour une image, la distillation et le raffinement peuvent, par exemple, être combinés et s'effectuer selon la résolution et la couleur : une image GIF couleur en 800x600 peut être dégradée en une image GIF en 320x200 et 16 niveaux de gris (distillation) avec zoom sur une inscription de l'image source (raffinement).

Dégradation "temps réel" (*"on the fly" degradation*) : pour pouvoir correspondre à toutes les demandes, BARWAN ne précalcule aucune dégradation mais les effectue à la demande. Pour cela, il utilise une modélisation du temps d'exécution d'une dégradation. Cela permet de déterminer quels sont les meilleurs axes de dégradation à utiliser pour se conformer aux axes de variations.

Type de la donnée	Encodage spécifique	Axes de dégradation
Image	GIF, JPEG, PPM, Figure PostScript	Résolution, Nombre de couleurs, Palette des couleurs
Texte	ASCII, HTML, PostScript, PDF	Mise en forme : lourde (fontes, couleurs, etc), simple (marqueurs), aucune
Vidéo	NV, H.261, VQ, MPEG	Résolution, Nombre d'images/s, Nombre de couleurs, Limites de progression

TAB. 3.1 – Types de données et axes de dégradation dans BARWAN

3.1.1.3 MOWSER

MOWSER [Joshi et al. 1996, Joshi et al. 1997] est un *proxy HTTP* qui (i) permet à l'utilisateur de définir ses préférences selon la connexion réseau ou les ressources disponibles et (ii) effectue des transformations actives (*active transcoding*) des données selon ces préférences. Ce *proxy* est introduit sur la station supportant la connexion sans-fil et joue le rôle de serveur pour le mobile et celui de client pour les serveurs Web.

La première fonction du *proxy* est de stocker, en fonction de leur adresse IP, les préférences utilisateurs ainsi que les paramètres acceptables des requêtes HTTP. Ces paramètres acceptables sont déduits des préférences utilisateurs et des capacités du terminal portable. Ils concernent notamment le nombre de couleurs, la résolution, la possibilité de jouer du son, la taille autorisée des fichiers et les techniques de réduction de la taille des fichiers.

La seconde fonction du *proxy* est d'effectuer des transformations actives [Bharadvaj et al. 1998]. Dans la plupart des approches, le processus de transformation est unidirectionnel : la requête d'un client est donné à un serveur et la réponse de ce serveur est altérée. MOWSER propose d'altérer également la requête en utilisant les mécanismes de négociation du protocole HTTP/1.1. L'idée est que les serveurs Web actuels proposent différentes représentations d'une même donnée. Ils peuvent automatiquement choisir la bonne donnée si le client envoie des préférences dans sa requête. Lorsqu'une requête est effectuée sur le terminal mobile, celle-ci est transmise au *proxy HTTP*. Avant de l'envoyer au serveur Web, celui-ci prend en compte les préférences et paramètres acceptables et modifie la requête pour (i) supprimer les parties de la requête ne pouvant être satisfaites (fichier son supprimé si le terminal n'a pas de possibilité d'écoute), (ii) inclure les paramètres de négociation (par exemple : `Accept : image/x-sgif video/x-rmpg` pour *small gif* et *representative mpg*). Ce mécanisme peut s'appliquer à tout type de donnée, y compris à du code exécutable[1]. Dans une requête HTTP, si le terminal portable ne possède pas beaucoup de puissance de calcul, il est donc possible de spécifier que l'on préfère un script CGI plutôt qu'un script ou une applet Java.

Par ailleurs, MOWSER peut effectuer les mêmes dégradations sur les données reçues que les approches précédentes. Ces dégradations sont réalisées dans le *proxy* en utilisant les tech-

[1] Un programme n'est qu'une donnée pour un système d'exécution.

niques de réduction de fichiers préalablement spécifiées par l'utilisateur.

3.1.1.4 Approche de Chalmers et al.

L'approche de Chalmers et al. [Chalmers et al. 2001] montre que les spécifications de l'accès aux données dans les systèmes mobiles sont assez restreintes. Elles se limitent à (i) la définition des types de données (images/pas d'images, GIF/JPG, etc), (ii) la définition de préférences des utilisateurs (taille maximale des images = 5 Ko, taux d'encodage JPEG = 0.75, langue = français, etc) et (iii) la mise en place de mécanismes d'altération des données (diminution de la résolution, de la qualité, etc). Ces mécanismes ne sont pas suffisants dans de nombreux cas car ils ne prennent pas en compte la sémantique des données. Par exemple, pour des cartes routières, diminuer la taille ou la qualité pour un affichage sur un terminal portable ne peut que rendre les cartes moins lisibles. La solution est plutôt de n'afficher que les informations pertinentes pour l'utilisateur, comme les routes principales et le trafic, et de supprimer les informations non essentielles, comme la topographie ou la végétation.

Pour pouvoir définir la sémantique d'une donnée, une description de la donnée est nécessaire. Plusieurs méta-données sont ajoutées à la donnée elle-même pour décrire sa structure et caractériser ses éléments. Une donnée est une unité structurée d'éléments pouvant chacun avoir plusieurs variantes. La structuration s'effectue par agrégation (un carte contient un ensemble de routes, noms, etc) ou spécialisation (une route départementale est un sous-type de route, etc). Chaque élément possède un ensemble d'attributs le décrivant (`name = M1 motorway`, `type = major-road`, `scale = 1 :10000`, `size = 20 Ko`, etc).

Le processus de sélection d'une variante d'une donnée utilise ensuite un gestionnaire de qualité de service (*QoS Manager*), comme le montre la figure 3.2. Lorsqu'un accès à une donnée est effectué par l'application, le gestionnaire de qualité de service demande au serveur les méta-données associées. À l'aide de celles-ci, le gestionnaire utilise un algorithme de sélection des variantes prenant en compte les préférences de l'utilisateur et les contraintes de l'environnement mobile. Les préférences de l'utilisateur sont exprimées à l'aide d'une *utility function* (similaire à celle de [Walpole et al. 1999]). La bande passante est le seul paramètre examiné à l'aide d'un model simple à quantum (*time-sliced*). La sélection des variantes est transmise à l'application qui peut les charger depuis le serveur.

Chacun de ces envois peut passer par un *proxy* qui peut effectuer (i) les opérations préalablement citées d'altération des données et/ou (ii) des opérations de modifications liées à la sémantique des données. Ces modifications sémantiques peuvent, par exemple, s'effectuer sur les données ou les méta-données pour prendre en compte un environnement temporel. Dans la figure 3.2, la variante "traffic1.ntf", correspondant au trafic routier actuel pour la carte "roads1.ntf", est générée automatiquement par le *proxy*.

3.1.2 Préchargement

Le préchargement des données est une technique d'anticipation d'utilisation et de mise en cache des données. Utilisée dans de nombreux domaines comme dans les caches processeurs [Jouppi 1990], les systèmes de stockage [Patterson et al. 1995], les systèmes de fichiers [Lei et Duchamp 1997] ou les accès au Web [Padmanabhan et Mogul 1996], le préchar-

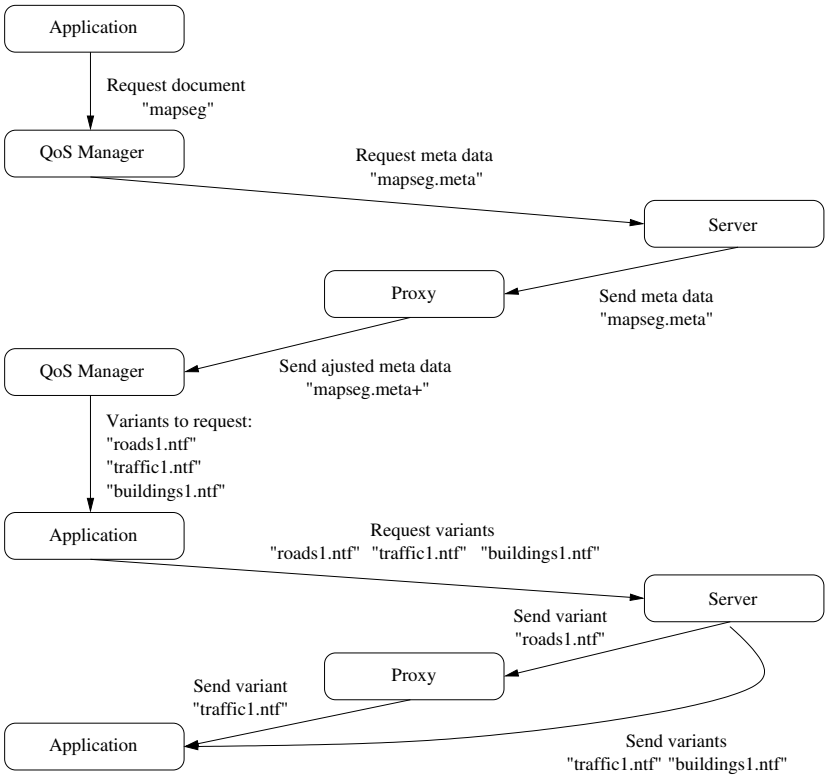


FIG. 3.2 – Processus de sélection des variantes d'une donnée dans l'approche de Chalmers et al.

gement est particulièrement utilisé en environnement mobile (*hoarding techniques*). Il permet en effet de (i) réduire le temps d'accès aux données à l'aide d'un accès en mémoire ou à un disque (plus rapide qu'un accès réseau), ce qui est d'autant plus important que la connexion sans-fil possède un faible débit, (ii) réduire les défauts d'accès aux données (*Data Miss*) lors de déconnexions en permettant de travailler localement sur le cache.

Le succès du préchargement repose sur le fait que le cache doit répondre à la majorité des accès aux données. Pour y précharger les données adéquates, plusieurs stratégies ont été testées dans différents systèmes. Ces stratégies peuvent utiliser des informations fournies par l'utilisateur ou bien être entièrement automatisées et calculer les données à précharger. Ce calcul peut prendre en compte (i) les accès et relations entre données, comme, par exemple, une analyse d'exécution de programme dans Spy Utility [Tait et al. 1995] ou une distance sémantique entre fichiers dans Seer [Kuenning et Popek 1997], (ii) les relations entre les données et l'environnement, comme les prédictions de mouvements dans [Liu et al. 1995, Chim et al. 1998, Bhattacharya et Das 1999] ou l'utilisation de la localisation dans [Long et al. 1996, dNitto Persone et al. 1998, Cheverst et al. 2000]. Ci-dessous, le système Coda illustre le préchargement "utilisateur", l'approche de Saygin et al. illustre le préchargement automatique à partir d'analyses d'historiques et le système Nexus illustre le préchargement en fonction de la localisation et du mouvement.

3.1.2.1 Coda

Le fonctionnement du système Coda [Mummert et al. 1995, Kumar et Satyanarayanan 1995] est détaillé dans le paragraphe 2.2.3.3. En mode *Hoarding* et *Write Disconnected*, le système utilise la connexion réseau pour effectuer des préchargements et, en mode *Emulating*, le système est déconnecté et travaille sur les données mises en cache lors des préchargements.

La stratégie de préchargement est fixe et utilise des préférences définies par l'utilisateur. Celui-ci doit spécifier les fichiers et répertoires candidats au préchargement. Les fichiers à précharger sont ensuite choisis selon une modélisation logarithmique du seuil de patience de l'utilisateur. Ce seuil peut être modifié par l'utilisateur.

Lors d'une déconnexion, les modifications des fichiers du cache sont mémorisées dans un historique. Celui-ci est optimisé de manière à occuper le moins de place possible sur le portable et à réduire le volume de données lors de la réintégration [Noble et Satyanarayanan 1994].

La gestion de cache par invalidation s'appuie sur l'observation que la majorité des objets est valide lors d'une reconnexion. Deux niveaux de granularité sont introduits : le volume et l'objet. Quand un objet est mis à jour, la version de l'objet ainsi que la version du volume le contenant sont incrémentés. Lors d'une vérification de validité, si la version d'un volume est à jour, les versions de tous les objets s'y trouvant le sont également ; si la version d'un volume n'est pas à jour, la vérification s'effectue au niveau de chaque objet du volume.

La réintégration en une seule fois de toutes les modifications effectuées peut s'avérer coûteuse en terme de bande passante et peut, par exemple, handicaper le préchargement en mode *Write Disconnected*. L'impact de cette réintégration est réduit par un découpage de l'ensemble des modifications en paquets de taille adaptée à la bande passante. L'atomicité de la réintégration est préservée car l'ensemble des modifications ne sont prises en compte du côté du serveur que lorsque celui-ci a reçu tous les paquets. La réintégration est également non-bloquante et d'éventuelles nouvelles modifications peuvent être effectuées sur les fichiers en cache, à l'exception du volume en cours de réintégration.

3.1.2.2 Approche de Saygin et al.

L'approche de Saygin et al. [Saygin et al. 2000] propose une technique de préchargement automatique, générique et indépendante des applications. Elle s'effectue selon trois phases : (i) construction de règles d'association à partir de techniques de *data-mining* sur les historiques d'accès, (ii) construction de l'ensemble des données candidates au préchargement et (iii) construction de l'ensemble des données préchargées.

Lors de la première phase, la notion de session est introduite pour partitionner l'historique des accès. Une session est un groupe continu de requêtes sur une période d'intérêt pour l'utilisateur. Ce mécanisme est générique et les bornes de ces sessions peuvent être déterminées par différents algorithmes (*sliding windows, flat, user-based, gap-based, cluster-based algorithms*). À partir de ces sessions, des règles d'association sont déduites par inférence sur les schémas d'accès. Par exemple, pour l'ensemble de sessions suivant : `{1,2} {1,3,2} {1,3} {5,6} {2,7} {2,5} {3,8} {3,7,9}`, les règles suivantes sont déduites : `1→2, 1→3` avec un support de 25% et une confiance de 66,6%.

La seconde phase consiste ensuite à construire, lors d'une session, l'ensemble des candidats au préchargement. Par exemple, lors de la session `{6,7,1,5}`, à partir des règles précédentes, les données `2` et `3` seraient mises dans l'ensemble des candidats au préchargement. Ces données sont mises dans cet ensemble selon une métrique dépendante du pourcentage support et confiance.

La troisième phase consiste enfin à choisir les données à précharger dans l'ensemble précédent. Ces deux ensembles sont séparés pour pouvoir prendre en compte la taille du cache du terminal portable. Le client peut effectuer ses choix selon des heuristiques à base de priorités qui prendront en compte ses besoins spécifiques (choix des données les plus petites, pas plus de X Ko, etc).

3.1.2.3 Nexus

Nexus [Hohl et al. 1999] est une plateforme qui fournit une infrastructure pour les applications orientées localisation. Cette approche s'appuie des représentations de régions du monde physique, augmentées par des objets virtuels (*Augmented Areas*). Un des axes de recherche de cette plateforme est de fournir un mécanisme de préchargement universel (*Universal Hoarding*) [Kubach et Rothermel 2001]. L'idée principale en est que l'accès à l'information est dépendant de la localisation, c.a.d. que la probabilité qu'un utilisateur accède à une certaine information est fonction de la localisation géographique de celui-ci.

Ce mécanisme de préchargement s'appuie sur une infrastructure de stations d'information avec un *proxy server* associé à chacune d'entre-elles. Chaque *proxy server* réalise un cycle de trois phases : *Download*, *Disconnected Operation*, *Upload*. Pendant la phase de chargement, l'utilisateur se trouve dans la région de la station d'information (c.a.d. le terminal portable est relié au *proxy server* par une liaison sans-fil). Le *proxy server* détermine les informations que l'utilisateur est le plus susceptible d'accéder avant la prochaine station d'information et les charge sur le terminal. Lorsque l'utilisateur quitte la région, il travaille en mode déconnecté sur les informations mises en cache. Chaque requête (satisfaite ou non) est stockée dans un fichier avec la position géographique où celle-ci a été réalisée. La troisième phase est déclenchée lorsque l'utilisateur arrive à la prochaine station d'information. Le fichier avec les requêtes est envoyé au *proxy server*. Celui-ci diffuse ces informations aux différents *proxy server* pour une mise à jour des accès aux informations.

Lors de la première phase, la décision de préchargement prise par le *proxy server* est probabiliste. Deux variantes sont possibles : une variante à gros-grain ou une variante à grain fin.

Dans la première variante, chaque *proxy server* maintient une table de probabilités d'accès aux informations de la région (*APT : Access Probability Table*). Elle se présente sous la forme de doublets $(i, a(i))$ (information i, probabilité $a(i)$ d'accès à l'information i). La mise à jour de cette table lors de la troisième phase s'effectue en donnant un certain poids α au passé $(a'(i) = \alpha \cdot a(i) + (1-\alpha) \cdot a_{\Delta}(i))$.

La deuxième variante prend en compte les déplacements futurs de l'utilisateur. Chaque région est découpée en zones, qui peuvent être homogènes (carrés) ou qui peuvent refléter la géométrie du monde réel (rues, bâtiments, etc). Le *proxy server* maintient des tables APT séparées pour chacune des zones de sa région (les mécanismes de la première variante s'appliquent de manière identique). Additionnellement, chaque *proxy server* maintient une table de

probabilité de visite de ses zones. Ces probabilités de visite sont mises à jour (i) en comptant le nombre de visites dans cette zone et (ii) à partir d'informations externes. L'utilisateur peut modifier les probabilités de visite en fournissant un chemin ou une direction plus ou moins précise qu'il souhaite emprunter. L'interface permet également à d'autres systèmes ou applications de calculer et fournir ces informations (systèmes de calcul de trajectoire, etc).

3.1.3 Réplication et cohérence des données

Lors d'un préchargement, des copies des données (*replicas*) sont stockées sur le terminal portable. La problématique du préchargement se combine alors avec les problématiques de réplication et de cohérence [Anderson et al. 1998] et plusieurs stratégies doivent être mises en place (voir figure 3.3).

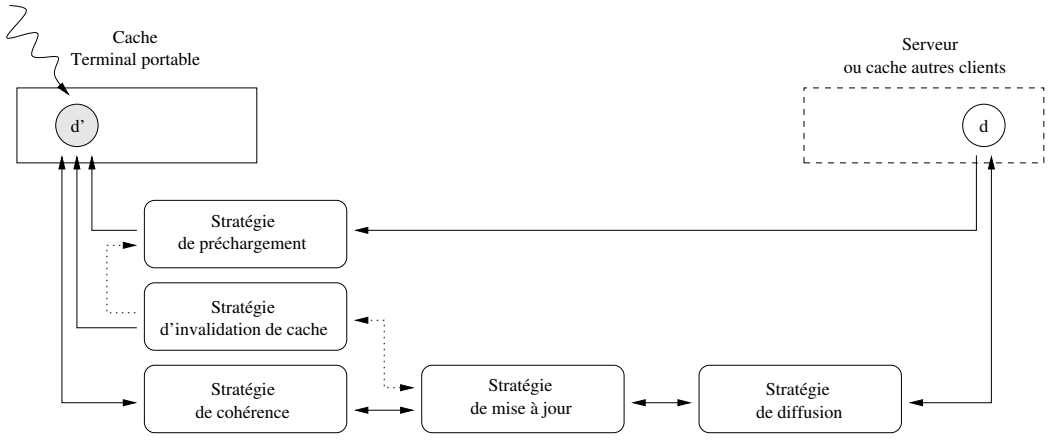


FIG. 3.3 – Relations entre stratégies de préchargement, d'invalidation, de cohérence, de mise à jour et de diffusion

Lorsque ces copies ne sont accédées qu'en lecture, comme pour une navigation Web, des problèmes de cohérence peuvent apparaître si les données originales changent. Dans ce cas, il est donc nécessaire d'avoir au minimum une stratégie d'invalidation de cache qui permette de "nettoyer" le cache et éventuellement d'initier un (re)(pré)chargement de données plus récentes. Celle-ci peut décider uniquement localement (données les moins utilisées, selon les contraintes matérielles, etc) ou faire des vérifications par rapport aux données originales.

Lorsque ces copies sont accédées en écriture (ce qui est souvent le cas pour permettre le travail en mode déconnecté), les modifications doivent être répercutées sur les données originales ainsi que sur les autres copies. Cela nécessite d'avoir trois stratégies : (i) une stratégie de cohérence qui va décider du niveau de cohérence à appliquer (pessimiste vs optimiste), (ii) une stratégie de diffusion qui va décider de qui initie et de qui reçoit (centralisée vs distribuée) et (iii) une stratégie de mise à jour qui va décider de quand concilier (immédiatement vs périodiquement). Les stratégies de mise à jour et de diffusion peuvent également être utilisées pour l'invalidation.

Ci-dessous, trois systèmes sont examinés : Bayou (stratégie de diffusion client/serveurs, stratégie de cohérence optimiste vis à vis du client, pessimiste entre les server), Rumor / Roam (stratégie de diffusion distribuée respectivement en anneau et selon un *Ward*, stratégie de cohérence optimiste, stratégie de mise à jour périodique) et DeNO (stratégie de diffusion distribuée par épidémie, stratégie de cohérence optimiste et stratégie de mise à jour par vote).

3.1.3.1 Bayou

Bayou [Demers et al. 1994] est un système qui offre à un ensemble d'utilisateurs mobiles le partage de données se trouvant sur plusieurs serveurs répliqués. Les applications concernées par ce partage ne nécessite qu'une connexion intermittente comme les agendas partagés, les bases de données partagées, etc [Edwards et al. 1997].

Bayou met en place une stratégie de réplication optimiste de type *read-any/write-any*. Cette stratégie autorise les clients à lire et à écrire sur n'importe quels serveurs. Les serveurs propagent ensuite ces modifications entre-eux suivant un protocole de réconciliation incrémental (*anti-entropy protocol*) [Petersen et al. 1996] prenant en compte d'éventuelles corruptions des serveurs [Spreitzer et al. 1997, Spreitzer et al. 1999].

Bayou est un système orienté applications. Il offre des mécanismes grâce auxquels l'application peut spécifier la résolution de conflits[2] [Terry et al. 1995, Terry et al. 1998] et paramétrer le protocole de réconciliation entre les serveurs [Edwards et al. 1997, Petersen et al. 1997].

La détection de conflits s'effectue lors des opérations d'écriture. À chaque opération d'écriture est associée une précondition définie par l'application (*dependy check*). Si celle-ci n'est pas satisfaite, il y a conflit et le serveur invoque alors un procédure de résolution de conflit (*merge procedure*) également définie par l'application.

L'application peut agir sur plusieurs paramètres du protocole de réconciliation : (i) le niveau de cohérence (*session guarantees : Read Your Writes, Monotonic Reads, Write Follow Reads, Monotonic Writes*), (ii) le niveau de validation d'une écriture (*stable, tentative*), (iii) le choix du moment de la réconciliation (de manière périodique, à l'initiative de l'utilisateur, lors de conditions système satisfaites), (iv) le choix des copies à réconcilier (copie primaire, copies les plus à jour, etc), (v) le niveau d'agressivité de la réconciliation (troncation plus ou moins importante du journal des écritures pour une stabilisation plus ou moins rapide) et (vi) le choix des serveurs pour la création de nouvelles copies.

3.1.3.2 Rumor / Roam

Le projet Travler [Travler] propose plusieurs systèmes pour les environnements mobile : un système de préchargement de fichiers, Seer, utilisé par deux systèmes de réplication, Rumor[3] et Roam.

[2] Deux type de conflits sont possibles : (i) les conflits *write/write* lorsque deux clients mettent à jour la même donnée de manières différentes et (ii) les conflits *read/write* lorsqu'un client met à jour une donnée en se basant sur la lecture d'une donnée en train d'être mise à jour.

[3] Descendant de Ficus [Reiher et al. 1994, Page et al. 1998].

Seer [Kuenning 1997] définit une distance sémantique qui quantifie l'affinité entre deux fichiers. Entre deux évènements d'ouverture de fichier, celle-ci décompte (i) le temps passé et (ii) le nombre de références à d'autres fichiers utilisés (*lifetime semantic distance*) [Kuenning et Popek 1997]. Cette distance est ensuite utilisée par un algorithme spécifique de groupement de fichiers (*agglomerative algorithm*) dont les tests montrent l'efficacité [Kuenning et al. 1997].

Ce groupement de fichiers est ensuite donné comme information aux systèmes de réplication. Rumor [Guy et al. 1998] met en œuvre une stratégie de cohérence optimiste, une stratégie de mise à jour périodique et une stratégie de diffusion *peer-to-peer*. Pour ne pas dépendre d'un système, Rumor est construit au niveau applicatif[4]. Chaque copie est fragmentée en volumes, portions continues du système de fichiers, permettant d'exploiter la localité des fichiers. Le stockage d'un système de fichiers entier nécessite énormément d'espace disque, ce que ne peuvent fournir des terminaux portables. Rumor fournit un mécanisme de réplication sélective (*selective replication*) [Ratner 1995] applicable par fichier (*per-file reconciliation*), permettant ainsi à certaines copies de ne stocker qu'une portion des volumes et de réconcilier sur les fichiers plutôt que sur le volume entier. La détection et la résolution de conflits s'effectue selon des vecteurs de version [Ratner et al. 1997] et se propage par paire sur un anneau adaptatif. La propagation par pair permet de ne bloquer simultanément que deux terminaux lors de la réconciliation, le reste des terminaux peut ne pas être disponible. Par contre, la propagation sur un anneau ne permet pas une extensibilité aisée.

Roam [Ratner et al. 1999, Ratner et al. 2001] est une extension de Rumor destinée à améliorer l'extensibilité. Il se base sur une stratégie de diffusion utilisant *The Ward Model* [Ratner et al. 1996]. Un *Ward* est une collection dynamique de machines géographiquement proches avec l'une d'entre-elles jouant le rôle de *Ward Master*. Celui-ci peut être comparé à un serveur dans le modèle client/serveur avec quelques différences : (i) chaque membre du *Ward* est un noeud du modèle *peer-to-peer* et peut donc se réconcilier avec n'importe quel noeud (le modèle client/server n'autorise pas de communication client/client), (ii) tout les membres du *Ward* étant égaux, ils peuvent tous tenir le rôle de *Ward Master* ; des mécanismes d'élection et de reconfiguration sont mis en place si le *Ward Master* est défaillant. Dans les modèles *peer-to-peer* classiques, chaque noeud connaît l'existence de tous les autres. Ici, le *Ward Master* est le seul lien entre les *Wards*, il est le point d'accès unique permettant de connaître toutes les copies. Il peut donc appliquer un algorithme de cohérence à plus gros grain, entre *Wards* uniquement.

3.1.3.3 DeNO

DeNO [Keleher 1999] est un protocole de réplication pour les environnements mobiles et peu connectés. Il combine une stratégie de mise à jour par vote pondéré avec une stratégie de diffusion par épidémie. L'avantage de la diffusion épidémique est qu'elle ne pose aucune contrainte structurelle ou topologique. Un terminal portable a juste à maintenir ses voisins et, lors d'une diffusion, à établir une communication par paire. Les avantages de la validation par vote sont que (i) le protocole est totalement décentralisé et aucune donnée n'est primaire : les

[4] La différence principale entre Ficus et Rumor est leur niveau d'implantation. Ficus est implanté à l'intérieur du noyau du système (SunOS) alors que Rumor est implanté au niveau applicatif.

terminaux jouent tous le même rôle, (ii) la disponibilité est augmentée : il n'est pas besoin que tous les terminaux soient disponibles, un certain nombre de votes suffit, (iii) la réconciliation n'est plus un problème : entre deux modifications de copies, il y a un vote pour choisir.

Deux problèmes se posent pour effectuer un vote en environnement mobile : le maintient d'un groupe de votants et la non terminaison d'un vote. Chacun des participants doit maintenir l'ensemble du groupe des participants (ou son nombre) pour savoir qui a voté et si une décision est possible : le surcoût occasionné par la maintenance du groupe est d'autant plus important que la mobilité est forte. Les déconnexions peuvent également (i) fausser un vote : un vote peut échouer parce que certains terminaux ne peuvent voter positivement, (ii) être bloqué : un vote peut être dans l'attente de certaines voix.

DeNO résout ces problèmes avec une technique de crédit ou porte-monnaie (*Currency Mechanisms*) [Keleher et Çetintemel 2000, Çetintemel et Keleher 2000a]. Chaque votant possède un montant de monnaie par fichier. Ce montant reflète la capacité du terminal à maintenir celui-ci. Lorsqu'un vote est propagé, chaque votant peut décider de mettre un montant de monnaie dans le porte-monnaie. Lorsque le porte-monnaie atteint un certain montant prédéfini, le vote est entériné. Si le porte-monnaie ne peut plus être propagé, le vote échoue. Pour ne pas fausser le vote lors de déconnexions, des *proxies* sont déployés et votent de manière transparente selon des comportements définis par les terminaux portables [Çetintemel et Keleher 2000b].

Dans le protocole de base, la stratégie de cohérence assure une exécution faiblement cohérente. Une extension [Çetintemel et al. 2001] permet une cohérence forte en fournissant une exécution globalement sérialisable et un ordre unique de validation des transactions de mises à jour. Ces deux variantes autorisent les requêtes et validations entièrement en local sur le terminal portable, sans aucun blocage.

3.1.4 Résumé

Niveau d'adaptation : une des conclusions du chapitre 2 est qu'un système d'adaptation doit être modulaire et définir sa granularité d'adaptation. Les approches présentées dans les paragraphes précédents montrent que les stratégies de gestion des données s'applique à des niveaux différents : une stratégie de préchargement peut s'attacher à une entité cache, une stratégie de transformation de données peut s'attacher à une entité ou aux liaisons que celle-ci possède, une stratégie de réplication peut s'attacher à une architecture d'entités distribuées. *Un système d'adaptation générique doit donc définir une granularité variable d'adaptation.*

Adaptabilité dynamique : l'examen des approches présentées dans les paragraphes précédents rejoint les conclusions du chapitre 2 sur l'adaptabilité et le dynamisme. En effet, dans ces approches, pour un même niveau d'adaptation, plusieurs stratégies différentes peuvent s'appliquer. Toutefois, aucune de ces stratégies n'offre de solution idéale. Selon les contraintes de l'environnement ou les besoins de l'application, une stratégie est plus appropriée qu'une autre dans une situation donnée. *Un système d'adaptation doit donc dynamiquement pouvoir changer de stratégie d'adaptation.*

Gestion des données / gestion des ressources : les stratégies de gestion des données utilisent des ressources et le gain engendré pour l'optimisation de certains paramètres a forcément un coût sur d'autres paramètres de l'environnement mobile. Par exemple, la compression

d'une donnée réduit la bande passante utilisée pour le transfert vers le terminal portable, mais elle utilise du temps processeur sur un serveur de l'environnement effectuant la compression et sur le terminal portable effectuant la décompression. Le paragraphe 3.2 examine plus en détails les stratégies de gestion des ressources en environnements mobiles.

3.2 Stratégies de gestion des ressources

Les stratégies de gestion de ressources considèrent les ressources allouées par les applications sur le terminal portable et aussi dans l'environnement. Pour optimiser cette allocation en fonction des paramètres de l'environnement mobile, différentes techniques doivent être examinées. Les solutions aux problèmes de découverte et localisation de ressources sont présentées dans le paragraphe 3.2.1. Les différents modèles d'utilisation distribuée de ces ressources sont présentés dans le paragraphe 3.2.2. L'équité de cette utilisation par placement et équilibrage de charge est examinée dans le paragraphe 3.2.3.

3.2.1 Nommage, découverte et localisation des ressources

Pour accéder à une ressource, il est nécessaire de connaître son nom (ou d'avoir sa description) et d'avoir un moyen de découvrir où celle-ci se situe. Ce problème est bien connu dans les systèmes distribués et à donné lieu à de nombreuses normes ou conventions (i) de nommage, comme URN (*Uniform Resource Names*) [Moats 1997], CORBA *Naming Service* [OMG 2002b], nommage dans Globe [Ballintijn et al. 2000] ou (ii) de description de ressources, comme RDF (*Resource Description Framework*) [W3C 1999b] ou WSDL (*Web Services Description Language*) [Chinnici et al. 2003].

Plusieurs protocoles de découverte ont également été inventés pour répertorier les ressources et en donner la localisation (GNS (*Global Name Service*) [Lampson 1986], DNS (*Domain Name System*) [Mockapetris 1987a, Mockapetris 1987b], X.500 [Weider et al. 1992, Wahl et al. 1997], CORBA *Trader Service* [OMG 2000a], UDDI (*Universal Description, Discovery and Integration*) [UDDI 2000], découverte dans Globe [Baggio et al. 2001], etc).

La mobilité apporte de nouvelles dimensions à ces protocoles. En effet, lorsqu'un terminal portable arrive dans un nouvel endroit, celui-ci peut dynamiquement amener de nouvelles ressources. Lorsqu'il quitte cet endroit, ces ressources deviennent non disponibles alors qu'elles étaient peut-être en cours d'utilisation. Différents systèmes de découverte peuvent être utilisés en environnements mobiles, comme Jini, UPnP [Microsoft 2000], SLP, SDS [Czerwinski et al. 1999] ou INS, et certains sont présentés ci-dessous.

3.2.1.1 Jini

Jini [Sun 2001a] est une architecture qui permet de créer des services orientés-réseaux (matériels ou logiciels). Pour permettre aux clients de découvrir ce que propose les fournisseurs de services, Jini fournit le *Lookup Service* [Sun 1999].

Lorsqu'un nouveau service se connecte au système Jini, il localise le *Lookup Service* à l'aide d'un protocole d'annonce *multicast* et de réponse *unicast* (*discovery*). Pour s'enregistrer,

il envoie au *Lookup Service* un objet Java implantant son interface (*join*).

Lorsqu'un nouveau terminal se connecte au système Jini, il localise de la même manière le *Lookup Service*. Pour accéder à un service, il interroge le *Lookup Service* qui utilise les interfaces des services enregistrés (*lookup*). Le mécanisme de recherche de Jini s'appuie sur une correspondance au niveau du mécanisme de sérialisation de JavaSpace [Sun 2002a]. L'avantage est de réaliser les correspondances entre deux services de même sous-type et l'inconvénient est de ne pas faire correspondre deux mêmes classes de versions différentes. Une fois trouvé, le *Lookup Service* renvoie ensuite au client un objet à exécuter localement. Cet objet est une copie du service demandé ou un *proxy* qui redirige les appels vers le service original.

L'accès aux services utilise un mécanisme de bail (*lease*). Un bail est une garantie d'accès pendant une période de temps donnée. Chaque bail est négocié entre le client et le fournisseur de service. Si un bail n'est pas renouvelé avant expiration, le service est considéré comme libéré pour le client comme pour le fournisseur. L'expiration peut résulter (i) de la fin du besoin du service, (ii) d'une défaillance du client, du serveur ou du réseau ou (iii) d'un refus de renouvellement. Les baux sont exclusifs ou non-exclusifs et permettent donc d'implanter l'exclusion mutuelle ou le partage d'un service.

Le passage à l'échelle n'est pas beaucoup abordé et fait d'ailleurs l'objet de recherches [GGF]. Jini introduit la notion de fédérations qui correspondent à des domaines d'administration locaux. Les interactions entre ces fédérations ne sont pas bien définies : enregistrement possible d'un *Lookup Service* par un autre *Lookup Service*, mention d'un *Inter-Lookup Service* entre fédérations.

3.2.1.2 SLP

SLP (*Service Location Protocol*) [Guttman et al. 1999, Guttman 1999] introduit la notion d'étendue (*scope*) comme étant un domaine non administratif (par exemple, un département dans une entreprise). Trois types d'entités permettent de gérer une étendue : (i) les *Service Agents (SAs)* qui correspondent chacun à une ressource (par exemple, un espace mémoire, une imprimante ou un logiciel), (ii) les *User Agents (UAs)* qui sont les utilisateurs des services et (iii) les *Directory Agents (DAs)* qui répertorient les différents SAs d'une étendue.

Trois moyens de découverte sont utilisables dans une étendue : (i) une découverte active (*Active discovery*), les SAs et UAs envoient des requêtes *multicast* pour trouver le DA, (ii) une découverte passive (*Passive discovery*), les DAs envoient périodiquement des messages d'annonce en *multicast* pour signaler leur présence et (iii) les UAs et SAs peuvent connaître la localisation des DAs par des moyens externes, comme un fichier de configuration.

Les UAs accèdent ensuite aux SAs de leur étendue selon deux modes possibles : (i) comme pour Jini, les SAs enregistrent leurs services dans le DA local et les UAs envoient des requêtes de demande de services au DA, ou (ii) pour garder la compatibilité avec les versions précédentes du protocole, les SAs attendent les requêtes sur un certain port de communication et les UAs utilisent le *broadcast* ou IP *multicast* pour envoyer leurs requêtes. Le désenregistrement d'un service dans un DA se fait de manière implicite par mécanisme d'expiration. Pour avoir une idée du trafic engendré par ces mécanismes d'enregistrement, [Govea et Barbeau 2001] présente une comparaison de l'utilisation de la bande passante et de la latence entre les systèmes Jini et SLP.

Un des aspects intéressants de SLP est la structure de description des services. Les services sont organisés par types et chaque type est associé avec un modèle décrivant les attributs requis pour ce type de service. L'expressivité de ces différentes fonctionnalités est assurée par XML DTD (*Document Type Definition*).

Pour le passage à l'échelle, une extension a été proposée [Rosenberg et al. 1997]. L'approche suggérée est d'avoir un *Advertising Agent (AA)* dans chaque étendue. Les différents AAs envoient ensuite, par *multicast*, les informations sur leurs services locaux. Un *Brokering Agent (BA)* dans chaque étendue se charge ensuite (i) d'écouter les informations envoyés par les AAs (ii) de les enregistrer dans le DA local comme s'il était un SA.

3.2.1.3 INS

INS (*Intentional Naming System*) [Adjie-Winoto et al. 1999] est en même temps un système de nommage et de découverte de services.

Le nommage est intentionnel, c.a.d. qu'une application décrit les ressources ou données dont elle a besoin sous forme de propriétés et d'attributs. Les noms intentionnels sont des expressions appelées *name-specifiers*. Ils s'organisent comme une hiérarchie de paires `(attribut,valeur)` ayant une relation de dépendance fils-père (par exemple, la paire `building=whitehouse` est dépendante de son père `city=washington`). La correspondance avec une spécification arbitraire d'un nom par une application se fait de manière exacte ou par des opérateurs à intervalles.

Chaque domaine (de nommage) possède des *Intentional Name Resolvers (INRs)* pour résoudre les noms. Ceux-ci contiennent les méta-données sur les noms et la localisation des objets correspondants. Pour échanger ces informations, les INRs sont interconnectés au niveau applicatif et enregistrés dans le *Domain Space Resolver (DSR)*. Le protocole d'échange entre INRs est auto-configurable (démarrage, terminaison automatique, maintenance des voisins, mises à jour périodiques, équilibrage selon la bande passante, etc).

INS intègre simultanément la résolution de nom et l'envoi des messages à l'aide d'une liaison retardée (*late binding*). Lorsqu'une application demande une ressource, le *name-specifier* ainsi que les messages destinés à cette ressource sont transmis à l'INR le plus proche du domaine. Celui-ci choisit de les transmettre à tels ou tels autres INRs jusqu'à celui désiré. Les INRs affectent donc les décisions de routage et permettent un traçage efficace des changements (mobilité utilisateur, modification des attributs, etc). L'application peut donc également influer sur les décisions de routage en spécialisant les INRs.

Pour l'extensibilité et le passage à l'échelle, INS étend les mécanismes précédents : plusieurs DSRs et un partitionnement de l'espace de nommage. Pour résoudre un nom qui n'appartient pas au domaine de nommage local, le DSR local doit maintenir un ensemble de DSRs externes. Des échanges d'informations, comme l'espace de nommage cible, permettent de rediriger les résolutions de nom externes vers le bon DSR. Comme les noms intentionnels nécessitent une organisation hiérarchique, chaque espace de nommage peut être partitionné de manière hiérarchique entre les INRs. Ainsi chaque INR n'a plus qu'un sous-ensemble de l'espace de nommage à résoudre.

3.2.2 Modèles de conception de systèmes distribués

L'utilisation d'une ressource distribuée passe par trois niveaux d'abstraction [Mattern et Sturm 2003]. La figure 3.4 présente les différents paradigmes correspondant à ces niveaux :

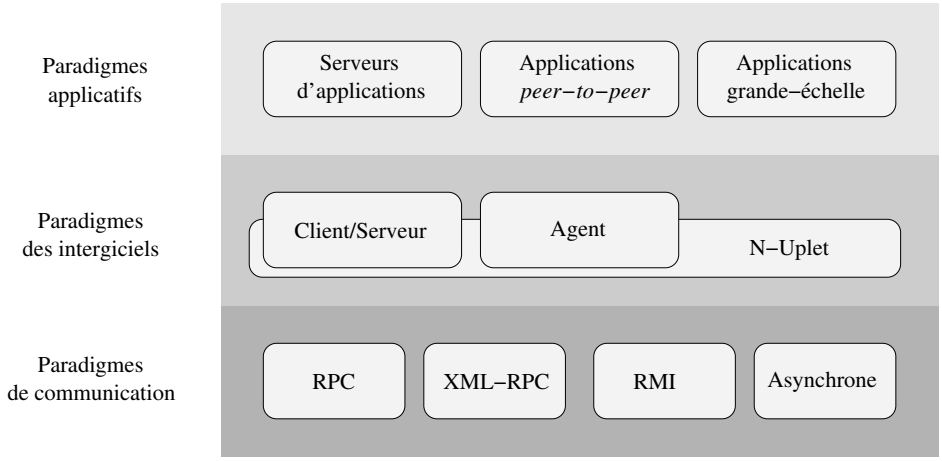


FIG. 3.4 – Paradigmes des systèmes distribués

(i) Paradigmes de communication : pour accéder à une ressource distante, il est nécessaire d'avoir un mécanisme de communication et de demande auprès de la machine où se trouve cette ressource. RPC (*Remote Procedure Call*) [Srinivasan 1995], RMI (*Remote Method Invocation*) [Sun 2002b], SOAP (*Simple Object Access Protocol*) [Box et al. 2000]) et les protocoles asynchrones (à évènements [Bacon et al. 2000]) en sont des exemples.

(ii) Paradigmes des intergiciels : les architectures de conception avec leurs environnements d'exécution associés permettent ensuite l'utilisation des ressources distantes. Différents modèles existent avec plusieurs implantations : (i) **modèle client/serveur** avec, par exemple, COM+ (*Microsoft Object Component Model*) [Kirtland 1997], CORBA (*Common Object Request Broker Architecture*) [OMG 2002a], J2EE (*Java 2 Platform, Enterprise Edition*) [Sun 2001c] ou .NET [Obermeyer et Hawkins 2001], (ii) **modèle agent** avec, par exemple, TACOMA [Johansen et al. 2002], ARA (*Agents for Remote Action*) [Peine et Stolpmann 1997], Aglets [Aridor et Oshima 1998] (voir un état de l'art dans [Gray et al. 2001a, Bradshaw], des listes de plateformes sur [MAL, AgentBuilder] et une réflexion sur le futur des agents dans [Kotz et al. 2002]) et (iii) **modèle d'espace de n-uplets (*Tuple Space*)** avec, par exemple, TSpaces [Lehman et al. 1999], JavaSpace [Sun 2002a] ou Ruple [Rogue 2002].

(iii) Paradigmes applicatifs : les couches précédentes sont utilisées pour construire des applications de différents types : (i) **serveurs d'applications** avec, par exemple, EJB (*Entreprise Java Beans*) [Sun 2001b], MTS (*Microsoft Transactions Server*) [Chappell 1997], Serveurs .NET [Microsoft 2001]), (ii) **applications *peer-***

to-peer avec, par exemple, Napster [Napster] ou Gnutella [Gnutella] et (iii) **applications grande-échelle** avec, par exemple, Seti@Home [Sullivan*III* et al. 1997] ou Distributed.net [d.net]. Plusieurs intergiciels sont également spécialisés pour ces deux derniers types d'applications comme Globus [Foster et Kesselman 1997] ou JXTA [Gong 2001].

Différentes approches proposent une modification des mécanismes de communication pour prendre en compte les déconnexions (M-RPC [Bakre et Badrinath 1995b], *wireless Java RMI* [Campadello et al. 2000]). Mais, les systèmes d'adaptation qui nous intéressent constituent un intergiciel entre l'environnement mobile et les applications. Plusieurs intergiciels récents, comme *Telecom Wireless CORBA* [OMG 2003b], *Java 2 Platform, Micro Edition* [Sun 2001d] ou *The .NET Compact Framework* [Microsoft 2002], commencent à proposer des solutions commerciales à la problématique des intergiciels pour environnements mobiles. Les paragraphes ci-dessous montrent cette évolution des intergiciels.

Modèles Client/Serveur étendus. Dans un modèle client/serveur classique, le découpage entre la fonctionnalité du client et celle du serveur est fixe. En environnements mobiles, cette frontière peut temporairement se déplacer pour répondre à différentes contraintes [Jing et al. 1999]. Ce déplacement des fonctionnalités peut s'effectuer dans le sens *serveur vers client* ou *client vers serveur*.

Lorsqu'un client souhaite avoir la possibilité de travailler en mode déconnecté, il doit pouvoir émuler le comportement du serveur. Une partie de la fonctionnalité du serveur doit donc se trouver sur le client. Dans le cas extrême (*full client architecture*), le client possède en local une copie ou une version légère du serveur, comme, par exemple, dans WebExpress ou Coda.

Lorsqu'un client est limité par ses ressources, certaines opérations qui devraient normalement s'exécuter sur le client peuvent l'être sur le serveur. Dans le cas extrême (*thin client architecture*), le client ne possède localement qu'un système minimal et toutes les exécutions se font de manière distante, comme dans Infopad [Le et al. 1994]. Certaines opérations peuvent également être exécutées sur un serveur *délégué* faisant office de serveur principal. Différentes manières d'implanter des serveurs délégués existent, comme avec des *proxies* [Hokimoto et al. 1996, Zenel et Duchamp 1997a] ou des serveurs répliqués (voir paragraphe 3.1.3).

Ci-dessous, Rover présente ces deux aspects avec des objets dynamiquement chargeables du serveur sur le client et vice-versa. L'approche de Zenel et al. présente un serveur délégué par *proxy*.

3.2.2.1 Rover

Rover [Joseph et al. 1997, Joseph et Kaashoek 1997] offre aux applications un support d'exécution en environnements mobiles basé sur un mécanisme d'appel de procédure avec liste d'attente QRPC (*Queued Remote Procedure Call*) et des objets dynamiquement transférables RDOs (*Relocatable Dynamic Objects*). La figure 3.5 présente l'architecture du système.

Dans Rover, les décisions liées à la gestion de la mobilité sont entièrement laissées à l'application. Le concepteur d'une application doit développer les parties client/serveur avec

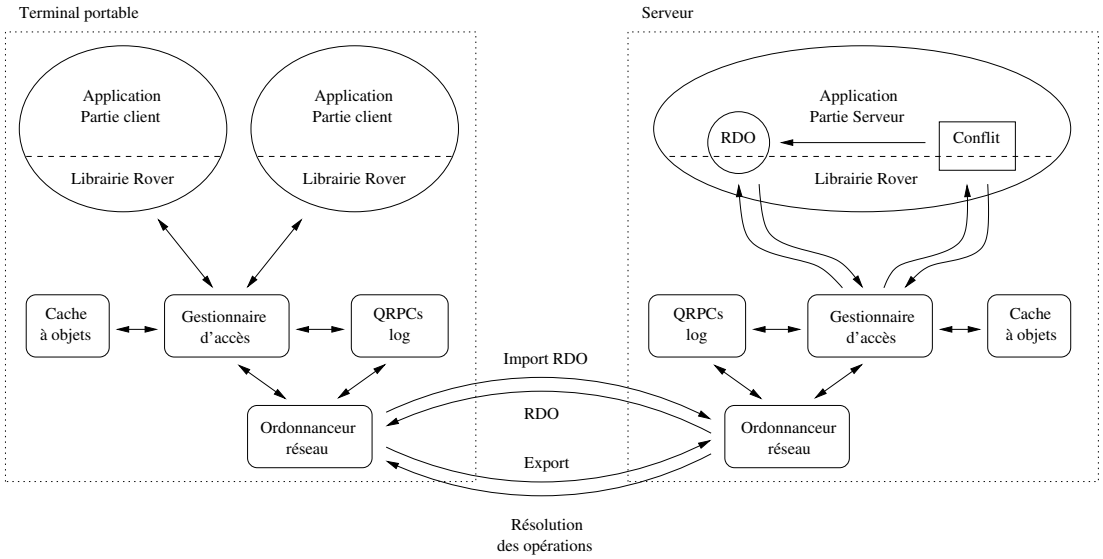


FIG. 3.5 – Architecture du système Rover

des RDOs. L'interface de programmation permet à l'application de gérer la mobilité des RDOs avec les primitives suivantes : (i) `create session` authentifie et établit une connexion avec le gestionnaire d'accès, (ii) `import` importe une copie d'un RDO dans le cache à objets, (iii) `invoke` invoque une méthode d'un RDO et (iv) `export` exporte les modifications effectuées sur un RDO.

À l'exécution, l'accès à un RDO peut se faire (i) de manière distante par un appel de procédure s'il ne se trouve pas dans le cache à objets ou (ii) de manière locale en important le RDO et exportant les modifications effectuées. Tous les accès réseau (appel de procédure, importation, exportation) utilisent QRPC qui applique la politique suivante : (i) enregistrement de l'appel dans un historique, (ii) retour du contrôle à l'application, (iii) si le terminal portable est déconnecté, essai d'envois périodiques et, lors du rétablissement de la liaison, envoie des appels stockés dans l'historique et (iv) lors de la réception d'un résultat, avertissement de l'application si celle-ci a enregistrée une demande de retour (*callback*).

3.2.2.2 Approche de Zenel et al.

Dans l'approche de Zenel et al. [Zenel et Duchamp 1997b, Zenel 1999], un *Proxy Server* est introduit entre le serveur et le client mobile. Celui-ci a pour mission d'appliquer des filtres pouvant effectuer des actions de transformations (similaires à celles décrites dans le paragraphe 3.1.1) ou des choix de protocoles pertinents. Le protocole PMICP (*Proxy Mobile Internetworking Control Protocol*) garanti que le trafic réseau passe par le *Proxy Server* malgré les déplacements du terminal portable.

La plupart des systèmes d'exploitation faisant la différence entre les protocoles de niveau réseau/transport (IP, ICMP, TCP, UDP) et les protocoles de niveau applicatif (MPEG, SMTP, HTTP), le *Proxy Server* est découpé en un *Low Level Proxy* et un *High Level Proxy*.

Le *High Level Proxy* travaille au niveau de chaque connexion et utilise la migration des *sockets* d'I-TCP (voir paragraphe 2.2.2.1 et figure 3.6). Le filtre est une *applet* en code natif ou interprété. Il est créé par l'application et, lorsque le terminal portable arrive dans un nouvel environnement, est migré et exécuté sur le *High Level Proxy* (voir figure 3.6-III).

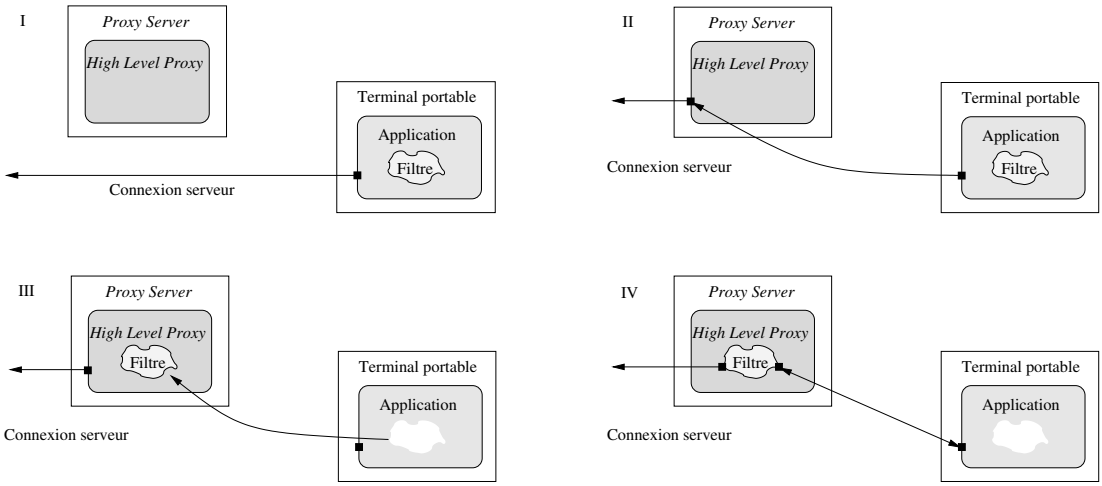


FIG. 3.6 – *High Level Proxy* dans l'approche de Zenel et al.

Le *Low Level Proxy* travaille directement au niveau de l'interface réseau, des ports de communication (voir figure 3.7). Tous les paquets sont stockés dans une queue d'attente (*LPP Packet Queue*) avant d'être filtrés. Le filtre est créé par le *Low Level Proxy* et peut seulement être paramétré par l'application. Celle-ci spécifie les critères de correspondance (adresse IP source, destination, etc) pour les paquets à envoyer au terminal portable.

Le déclenchement des filtres s'effectue par l'application (contrôle interne) ou par l'environnement (contrôle externe) (voir figure 3.8). Ces notifications sont faites par évènements consécutifs aux changements dans la bande passante estimée, la disponibilité de l'interface réseau, la batterie ou la localisation.

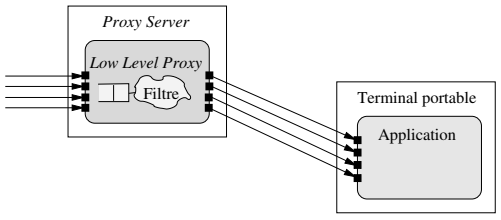


FIG. 3.7 – *Low Level Proxy* dans l'approche de Zenel et al.

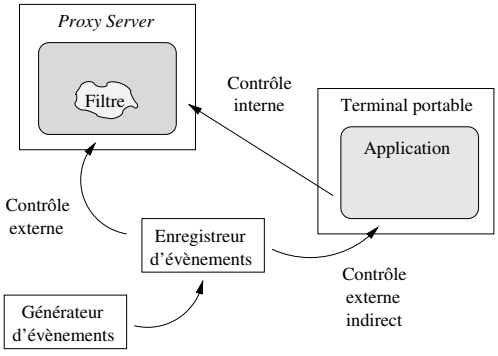


FIG. 3.8 – Contrôle interne et externe des filtres par gestion à évènements

Modèle Agent. Le concept d'agent a déjà été défini de multiples manières [Franklin et Graesser 1996, Jennings et al. 1998]. Plusieurs propriétés en donnent une classification : (i) *réactivité (perception+action) :* capacité à répondre de manière appropriée à un changement dans l'environnement, (ii) *autonomie :* capacité à exercer le contrôle de ses propres actions, (iii) *possession d'une ou plusieurs tâches :* capacité à ne pas seulement réagir à l'environnement mais à agir selon ses propres buts, (iv) *sociabilité :* capacité à communiquer avec d'autres agents ou des personnes, (v) *adaptabilité :* capacité à modifier ses buts ou son comportement et (vi) *mobilité :* capacité à se déplacer d'un endroit à un autre.

Les propriétés (i,v,vi) sont particulièrement intéressantes en environnements mobiles [Kotz et Gray 1999]. La réactivité permet de répondre à la grande variabilité des environnements mobiles. L'adaptabilité permet de répondre à l'imprévisibilité et à l'évolutivité des environnements mobiles. Lorsque cela s'y prête [Gray et al. 2001b], la mobilité permet de continuer les tâches malgré les déconnections, d'effectuer des optimisations sur les communications (bande passante, latence) et sur le temps d'exécution [Bandyopadhyay et Paul 1999, Helin et al. 1999].

Ci-dessous, D'Agents illustre les propriétés d'autonomie, d'adaptabilité et de mobilité et LEAP illustre les propriétés de réactivité, d'adaptabilité et de sociabilité.

3.2.2.3 D'Agents

D'Agents[5] [Kotz et Gray 1999] est un système à agents se focalisant principalement sur l'autonomie et la mobilité des agents en environnements mobiles. Ce cadre a particulièrement été testé avec des applications militaires dans le projet ActComm [Gray 2000].

Comme le montre la figure 3.9, l'architecture du système D'Agent se décompose en quatre couches. La couche de base gère toutes les interactions entre le réseau et le serveur D'Agents. Chaque site acceptant des agents exécute un serveur D'Agents. Celui-ci est en charge (i) de garder la liste et le statut des agents s'exécutant sur la machine, (ii) d'assurer une migration forte avec authentification, (iii) de fournir un stockage non volatile pour l'état des agents en cas de défaillance et (iv) d'assurer les communications entre agents avec un espace de nommage hiérarchique, des messages synchrones (*connection message*) et asynchrones (*event message*), des tampons d'émission et de réception et un mécanisme de sélection de l'interface de transport.

Tout les autres services (découverte de ressources, contrôle d'accès, communication de groupe, etc) sont fournit par des agents. En particulier, les *docking agents* permettent de supporter le mode déconnecté [Gray et al. 1996]. Si un agent est dans l'impossibilité de migrer à cause d'une défaillance du réseau ou d'une machine, celui-ci est ajouté dans une liste d'attente (*dock*) sur le réseau. La migration deviendra effective lorsque la machine cible redeviendra disponible (détection effectuée par les *traffic monitor agents*). Les *navigator agents* maintiennent un annuaire virtuel non exhaustif des services existants. Ainsi, un agent, qui recherche le service *S* et ne le trouve pas auprès du *navigator agent* de sa machine, peut dynamiquement modifier son planning de migration pour aller consulter d'autres *navigator agents* qui pourront peut-être lui indiquer la localisation du service *S*.

[5] Préalablement nommé AgentTcl [Gray 1997].

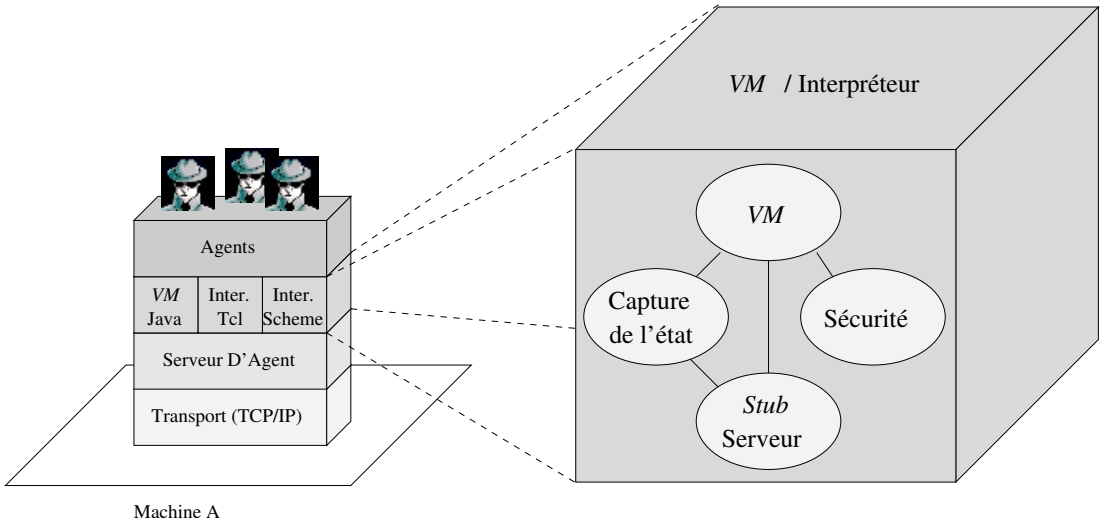


FIG. 3.9 – Architecture du système D'Agents

La troisième couche propose un interpréteur (ou une machine virtuelle) pour chaque langage. D'Agents supporte des agents programmés en Scheme, Tcl et Java. Chaque interpréteur est constitué (i) de l'interpréteur lui-même, (ii) d'un module de sécurité empêchant les agents d'effectuer des actions malveillantes, (iii) d'un module d'état qui capture et restaure l'état interne d'un agent et (iv) d'une interface d'accès aux fonctionnalités du serveur D'Agents.

La dernière couche est constituée des agents eux-mêmes. L'interface de programmation propose des primitives permettant aux agents de contrôler leur exécution : (i) `new Agent()`[6] permet la création d'un agent, (ii) `agent.begin(...)` et `agent.end()` permettent à l'agent de s'enregistrer et de se désenregistrer auprès du serveur D'Agents, (iii) `agent.send(...)` et `agent.receive(...)` permettent à l'agent d'envoyer et de recevoir des messages, (iv) `agent.jump(...)` permet à l'agent de changer de machine et (v) `agent.submit(...)` permet à l'agent de créer un agent fils.

3.2.2.4 LEAP

LEAP (*Lightweight Extensible Agent Platform*) [Bergenti et Poggi 2001] est une plateforme à agents intelligents pour terminaux portables. Cette plateforme est générique et a été testée avec une application de gestion d'équipes virtuelles : gestion de connaissances communes, travail coopératif, gestion d'emploi du temps ou de déplacements. Cette plateforme est la première à respecter les spécifications de la FIPA[7] [FIPA 2002].

[6] Exemples donnés avec la notation Java. L'interface de programmation diffère évidemment selon le langage de programmation utilisé, mais les primitives gardent une correspondance une à une.

[7] La FIPA (*Foundation for Intelligent Physical Agents*) est une organisation à but non lucratif visant à produire des standards pour gérer l'interopérabilité de systèmes à agents. Différentes spécifications en définissent plusieurs aspects : le transport des messages, le langage de communication entre agents, le langage de description sémantique d'un agent, le protocole d'interaction entre agents, etc.

LEAP est une branche indépendante du développement de la plateforme JADE [Bellifemine et al. 1999] et en reprend donc l'architecture (voir figure 3.10) [Berger et al. 2001]. L'agent AMS (*Agent Management System*) fournit la gestion du cycle de vie des agents (création, suppression, etc) ; l'agent DF (*Directory Facilitator*) fournit un service de pages jaunes ; le composant ACC (*Agent Communication Channel*) gère les communications externes (avec d'autres plateformes à agents) et le composant MD (*Message Dispatcher*) gère les communications internes (entre environnements d'exécution propres à la plateforme).

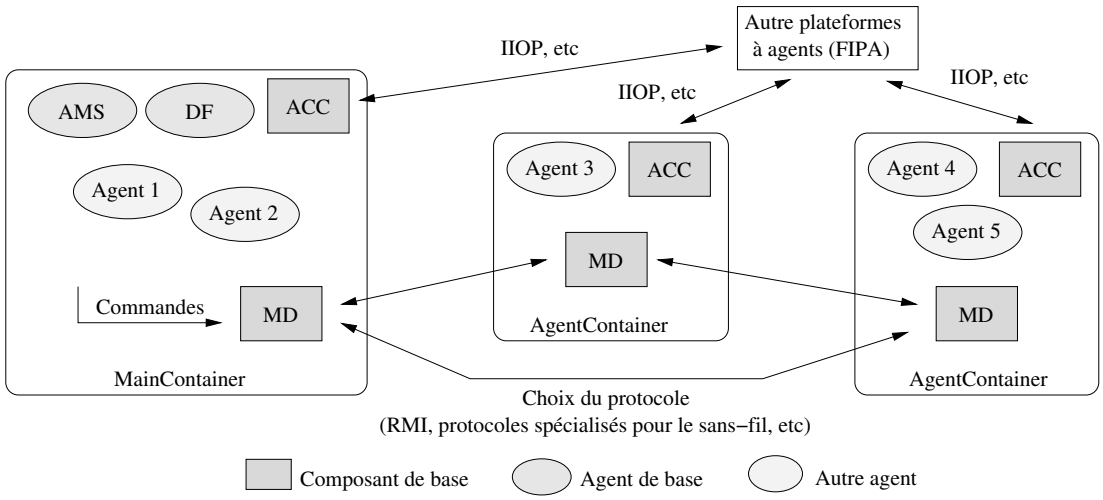


FIG. 3.10 – Architecture du système LEAP

Pour être adapté aux environnements mobiles, LEAP restructure de corps de JADE de manière à ce qu'il soit :

Léger : pour pouvoir s'exécuter sur des terminaux portables pauvres en ressources, la traditionnelle machine virtuelle Java est remplacée par la machine virtuelle KVM (*K Virtual Machine*) [Sun 2000] avec J2ME/CLDC (*Java 2 Platform, Micro Edition / Connected Limited Device Configuration*) [Sun 2001d]. Le corps de la plateforme LEAP est également définit comme un ensemble extrêmement minimal de composants et d'agents.

Indépendant du protocole de transport : JADE ne propose que l'utilisation de RMI pour les accès distants. LEAP remplace ces invocations par des appels à des commandes génériques. Le composant MD est ensuite responsable (i) du choix du protocole selon l'environnement mobile, (ii) de la transformation des appels génériques en appels spécifiques au protocole choisi et (iii) de l'envoie des informations (éventuellement en utilisant le mécanisme de sérialisation).

Extensible : à partir du cœur minimal de LEAP, deux types d'extensions sont possibles : l'intégration de fonctionnalités et les fonctionnalités optionnelles. LEAP offre aux applications la possibilité d'intégrer des fonctionnalités générales, comme un nouveau protocole de transport dans le composant MD, ou des fonctionnalités spécifiques aux applications, comme des profils dans l'agent DF. Selon la puissance de la machine, des

fonctionnalités optionnelles peuvent être activées. La mobilité et/ou la surveillance des ressources peuvent, par exemple, être activées dans les agents et prises en compte dans les agents AMS et DF.

Modèle N-Uplet. Le modèle d'espace de n-uplets (*Tuple Space*) est décrit pour la première fois dans le langage Linda [Carriero et Gelernter 1989]. L'idée est d'avoir un ensemble de programmes distribués, n'ayant aucune connaissance les uns des autres, mais capables de communiquer. La communication s'effectue par publication des données (n-uplet) dans un espace de n-uplets. L'accès aux n-uplets s'effectue à l'aide de primitives (i) de lecture : `in` lit et retire un n-uplet, `rd` lit un n-uplet sans le retirer, (ii) d'écriture : `out` écrit un n-uplet, `eval` écrit un n-uplet *actif* (un ensemble de fonctions doit être exécuté sur ce n-uplet avant qu'il ne soit accessible). Ces interactions sont orientées sur les données et ne fonctionnent que par correspondances. Pour avoir plus de flexibilité, [Cabri et al. 1998] et [Omicini et Zambonelli 1998] proposent des espaces programmables de n-uplets. Ceux-ci fournissent la possibilité aux applications d'introduire des entités exécutables (*computational activities*) dans les espaces de n-uplets. Celles-ci peuvent, par exemple, être activées lors de l'arrivée d'un certain n-uplet dans l'espace.

Un espace de n-uplets fournit une abstraction qui, par rapport aux deux modèles précédents, peut être examinée selon deux angles. Du point de vue de l'implantation, le stockage et la propagation des modifications des n-uplets peuvent s'effectuer selon un modèle *n* clients/un serveur centralisé [Silva et al. 1994] ou *n* clients/*n* serveurs [Rowstron et Wood 1996] ou selon le modèle agent [Silva et al. 2001]. Du point de vue de l'utilisation, un espace de n-uplets peut être utilisé par des plateformes client/serveur ou agents pour régler des problèmes d'exécution parallèle [Carriero et al. 1995], de persistance et de tolérance aux fautes [Jeong 1996] ou de coordination [Cabri et al. 1998].

Ci-dessous, L^{2}imbo et Lime proposent le découpage de l'espace global de n-uplets en différents espaces adaptés à l'environnement mobile. L^{2}imbo a été mis en œuvre à partir du modèle client/serveur et est utilisé par différents agents. Lime utilise exclusivement des agents.

3.2.2.5 L^{2}imbo

La plateforme L^{2}imbo [Davies et al. 1997, Wade 1999] reprend le modèle de Linda en y incluant des extensions pour prendre en compte les aspects spécifiques aux environnements mobiles : espaces multiples de n-uplets, n-uplets avec attributs de qualité de service, n-uplets typés hiérarchiquement et gestion des espaces de n-uplets par des agents système.

Le modèle de Linda implique une vue consistante d'un espace global et unique de n-uplets. Dans un environnement distribué, cette approche s'applique difficilement en terme d'extensibilité et de performances [Hupfer 1990]. Dans un environnement mobile, ce problème s'accentue encore du fait du coût des communications. L^{2}imbo partitionne l'espace global de n-uplets en espaces multiples. Un espace de n-uplets universel (*Universal Tuple Space*) se charge de la gestion des espaces multiples : `out(create_tuple_space, QoS, characteristics, request_id)`, `in(tuple_space, ?QoS, ?characteristics, ?handle, request_id)`, etc. L'implantation de cet espace de n-uplets universel est mis en œuvre de manière centralisée ou distribuée [Davies et al. 1998].

Les paramètres `QoS` et `characteristics` permettent la création d'espaces de n-uplets spécialisés, par exemple, pour la persistance ou la sécurité. Pour pouvoir optimiser l'utilisation de la bande passante, L^{2}imbo associe une date limite à chaque n-uplet. Dans le cas d'une écriture, la date limite correspond au temps que le n-uplet est autorisé à rester dans l'espace de n-uplets avant d'être effacé. Dans le cas d'une lecture, la date limite correspond au temps maximum de recherche avant expiration.

L'organisation hiérarchique des n-uplets établit une relation de sous-typage entre eux. En plus des bénéfices liés à une signature typée (vérification, etc), la relation de sous-typage peut être utilisée dans les primitives de recherche, comme `in` qui recherche si un n-uplet d'un type donné correspond à un n-uplet existant. Dans deux espaces distincts, deux n-uplets de types différents peuvent correspondre par sous-typage.

Des agents sont ajoutés aux espaces de n-uplets. Ceux-ci sont classés dans deux catégories : (i) les agents système fournis par L^{2}imbo et (ii) les agents applicatifs fournis par les applications. Cette classification n'est pas rigide car une application est libre d'introduire des agents système additionnels. L^{2}imbo définit trois agents système principaux : les *Type Management Agents*, les *QoS Monitoring Agents* et les *Bridge Agents*.

Les *Type Management Agents* déterminent les relations de typage entre espaces de n-uplets. Ils fournissent des fonctionnalités (i) de recherche permettant à un espace de n-uplets de trouver des sous-types donnés dans d'autres espaces de n-uplets et (ii) de ratification autorisant ou non la création d'une relation de sous-typage. Les *QoS Monitoring Agents* (*Connectivity Monitors*, *Power Monitors*, *Cost Monitors*, etc) sont inclus dans un espace de n-uplets spécifique présent sur chaque machine (*Local Management Tuple Space*). Ceux-ci injectent dans cet espace des n-uplets représentant l'état courant du système. Les *Bridge Agents* fournissent les moyens (i) de lier de manière arbitraire deux espaces de n-uplets et (ii) de contrôler la propagation des n-uplets entre ces espaces.

Pour réaliser l'adaptation, une des techniques principales de L^{2}imbo consiste à utiliser des *Filtering Agents*. Un *Filtering Agent* est un *Bridge Agent* spécialisé pour la transformation de n-uplets selon différents niveaux de qualité de service. Par exemple, un *Filtering Agent* peut agir entre deux espaces de n-uplets contenant un flux vidéo MPEG de manière à ne propager que les trames de type I (voir figure 3.11). Ces agents autorisent la construction en parallèle d'espaces de n-uplets offrant le même service mais selon une qualité différente.

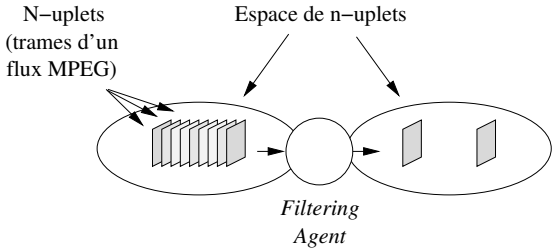


FIG. 3.11 – *Filtering Agent* dans L^{2}imbo

3.2.2.6 Lime

Lime [Picco et al. 1999, Picco et al. 2000] propose un espace de n-uplets gérant *la mobilité physique*, correspond aux déplacements d'un terminal portable, et *la mobilité logique*, correspond à la migration de code. Pour cela, Lime introduit la notion d'espace fluctuant de n-uplets (*Transiently Shared Tuple Space*).

Comme le montre la figure 3.12, cette notion est implantée à l'aide d'un modèle agent. Chaque agent mobile possède, de manière permanente, son propre espace de n-uplets (*Interface Tuple Space*) contenant les informations que celui-ci souhaite partager avec d'autres agents. Pour un agent, l'action de rendre accessible son propre espace de n-uplets consiste à s'enregistrer dans un espace fluctuant. Deux types d'espaces fluctuants existent : un espace de n-uplets propre à chaque machine (*Host-Level Tuple Space*) qui permet le partage entre agents locaux et un espace de n-uplets entre machines (*Federated Tuple Space*) qui permet le partage en réseau.

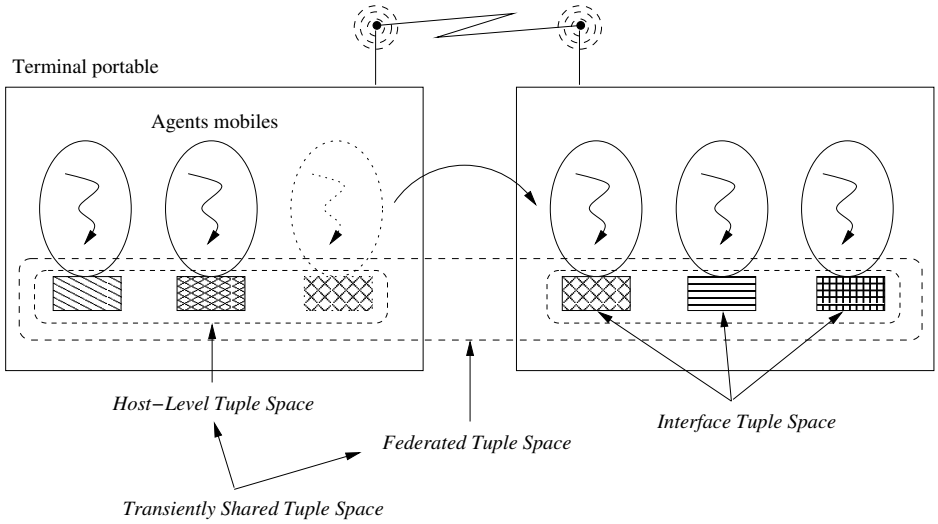


FIG. 3.12 – Architecture des espaces de n-uplets dans Lime

La mobilité physique est gérée par modification de l'espace de n-uplets entre machines à l'aide de protocoles de gestion de groupe [Murphy et al. 2001]. La mobilité logique est assurée par la migration des agents entre espaces de n-uplets propres à chaque machine.

Pour pouvoir prendre en compte ces mobilités, deux modifications principales sont ajoutées par rapport au modèle Linda : l'ajout de la localisation et d'un système de réaction.

L'ajout d'un n-uplet se fait par défaut dans l'espace de n-uplets de l'agent effectuant l'ajout. Or celui-ci peut très bien vouloir écrire dans l'espace d'un autre agent, par exemple, pour assurer la persistance. La primitive d'écriture `out[λ](n)` prend ainsi en compte la localisation d'un agent λ.

Les évènements de changements sont naturellement modélisés par des n-uplets. Ceux-ci sont de différents types : arrivée et départ d'un agent, changement de qualité de service, etc et sont insérés par des agents de surveillance dans un espace de n-uplets particulier *LimeSystem ITS* présent sur chaque machine. Le modèle de Linda est étendu pour pou-

voir ajouter des réactions `reactsTo(s,p)` à ces évènements (le code `s` est exécuté lorsqu'un n-uplet correspondant à `p` est trouvé). Ces réactions peuvent s'appliquer de manière synchrone (*StrongReaction*) ou asynchrone (*WeakReaction*), et être locales (*LocalizedReaction*) ou distribuées sur un espace fluctuant donné (*UbiquitousReaction*).

3.2.3 Partage et équilibrage de charge

La problématique du partage et de l'équilibrage de charge consiste à mettre à profit, de la meilleure façon, les possibilités d'utilisation des ressources distantes. Le partage de charge vise à garantir qu'aucune machine n'est sous-chargée ou surchargée. L'équilibrage de charge cherche à instaurer une charge uniforme sur toutes les machines.

Pour assurer cette répartition de la charge, différents choix et actions doivent être réalisés. Ceux-ci sont traditionnellement mis en place au sein de trois politiques [Zhou 1988] :

Politique d'information : la politique d'information définit la nature des informations à collecter, comme l'état d'utilisation des ressources ou les besoins des application. Elle définit également la manière dont ces informations sont collectées et mises à jour.

Politique d'élection : la politique d'élection utilise ces informations ainsi qu'une métrique de charge pour décider si une machine est en sur(sous)charge. Elle décide particulièrement quelles sont les entités applicatives[8] en cause.

Politique de placement : la politique de placement décide des actions à appliquer aux entités applicatives élues. Ces actions peuvent être (i) un placement : choix initial d'une machine d'exécution, ou (ii) une migration : transfert en cours d'exécution d'une machine à une autre. Pour choisir la machine cible de l'exécution, les informations collectées et la métrique de charge peuvent (i) ne pas être utilisées, comme dans le cas d'un placement prédéfini, (ii) être utilisées d'une manière différente de celle de la politique d'élection, comme en choisissant d'élire localement mais de placer globalement, ou (iii) être utilisées de manière similaire à la politique d'élection.

Une des premières cibles de cette problématique a été la mise en commun des ressources d'un réseau de stations de travail (NoW [Anderson et al. 1995]). Pour chacune de ces politiques, de nombreux algorithmes existent et sont classés dans la figure 3.13 [Bernard et Folliot 1996, Guyennet et al. 1997]. Ils ont été expérimentés dans différents systèmes comme Radio [Bernard et al. 1991] et Utopia [Zhou et al. 1992] pour les algorithmes non-adaptatifs et non-préemptifs, Condor [Litzkow et Livny 1990] et Sprite [Douglis et Ousterhout 1991] pour les algorithmes préemptifs, GatoStar [Folliot et Sens 1994] pour le placement et la migration, PM^2 [Namyst et Méhaut 1996] et Stardust [Cabillic et Puaut 1996] pour les algorithmes adaptatifs.

La problématique de répartition de charge s'est ensuite développée selon deux axes : (i) l'élargissement des critères de distribution afin de prendre en compte de nouveaux environnements (Internet, etc) et de nouvelles applications (applications multimédias, applications

[8] Les entités applicatives dépendent du choix du modèle de conception de système distribué : processus, objets, composants, agents, etc.

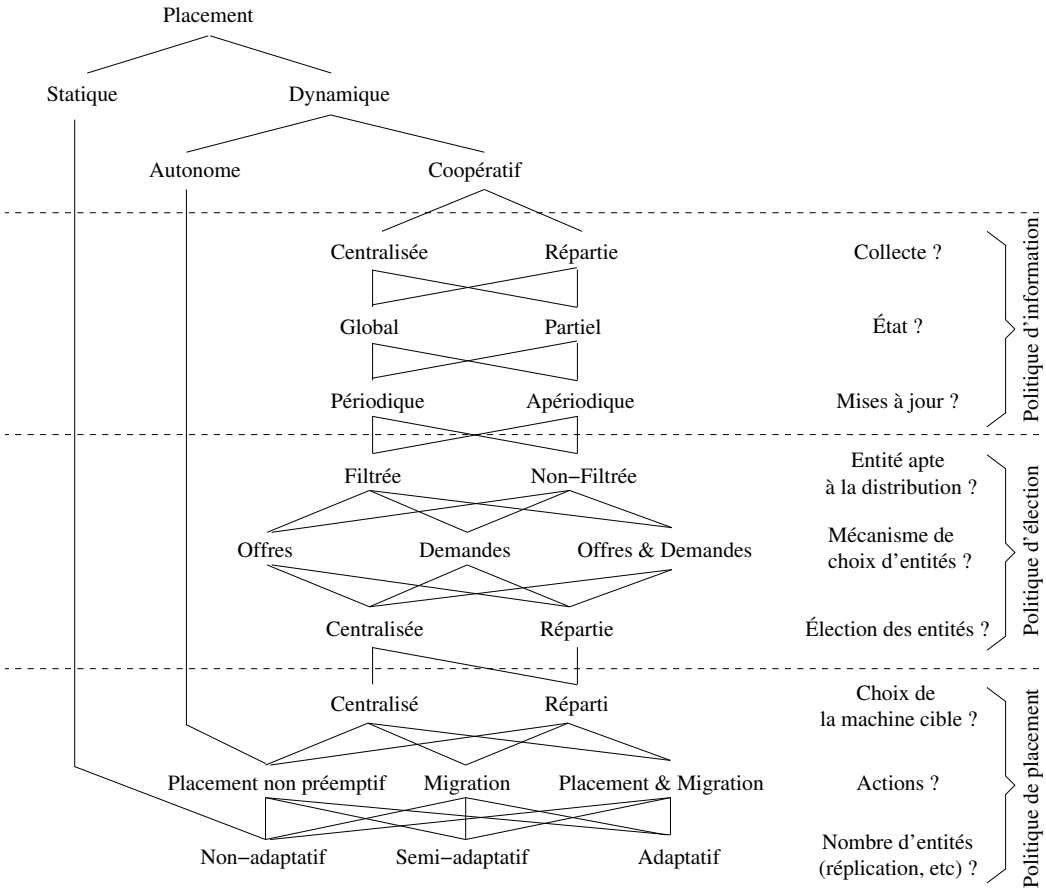


FIG. 3.13 – Classification des algorithmes de placement

à grande échelle, etc) et (ii) l'augmentation du niveau d'abstraction permettant une utilisation plus simple.

Dans les approches précédentes, les informations collectées et l'indice de charge portent principalement sur le critère d'utilisation du processeur. Pour optimiser l'ensemble des ressources qu'une application peut utiliser, les approches multi-critères considèrent simultanément plusieurs nouveaux critères [Folliot 1992, Chatonnay et al. 2000, Gomoluch et Schroeder 2001] : (i) liés aux ressources : taux d'utilisation de la mémoire, nombres d'entrées/sorties des périphériques (disques durs, etc), taux d'utilisation du réseau (bande passante, délai, pertes, etc), etc ou (ii) liés aux applications : structure [Decker 2000], variabilité [Corradi et al. 1999], nombre d'utilisateurs, etc. Pour répondre de manière encore plus adéquate aux besoins des applications, certains approches permettent aux applications de spécifier leurs propres critères. Ainsi, dans [Cen et al. 1995] et [Nee et al. 1997], la réservation des ressources est modifiée en fonction de critères propres aux flux vidéo (fréquence et type des trames audio et vidéo). Certaines approches proposent une spécification de critères de qualité de service, comme, par exemple, par le biais de *distributed QoS adapters* dans [Campbell et Coulson 1997]. Ceux-ci se combinent aisément avec les cri-

tères de charge [Bom et al. 1998] et les informations locales [Silaghi et Keleher 2001] d'une station finale.

Afin d'abstraire les différents concepts des systèmes de répartition de charge, plusieurs approches proposent des cadres de conception offrant des propriétés d'abstraction, de structuration, de flexibilité de spécialisation et d'extension pour les différentes politiques. SOS [Shapiro et al. 1989] est un système orienté-objet dont chaque objet est constitué de fragments (interface client, interface de groupe, objet de connection) [Makpangou et al. 1994]. SOS n'inclue pas de mécanisme automatique de migration mais offre des politiques paramétrables pour la minimisation des communications. Legion [Grimshaw et al. 1997] et Globe [vSteen et al. 1999] sont des méta-systèmes conçus pour des applications à grande-échelle. Legion propose un cadre de conception dans lequel le programmeur peut spécialiser la politique de placement des objets [Chapin et al. 1999]. Globe reprend le modèle d'objets fragmentés de SOS : chaque objet est constitué de sous-objets (sémantique, communication, réplication, contrôle) dont chaque algorithme, notamment le placement et la migration, peut être spécialisé par l'application [Jansen et al. 2001].

Par rapport aux environnements statiques, les environnements mobiles introduisent de nouveaux paramètres hautement variables liés au terminal portable, au réseau sans-fil et à l'environnement d'exécution (voir paragraphe 1.3). Pour effectuer une répartition de charge efficace dans de tels environnements, il est nécessaire de les prendre dynamiquement en compte comme critères de distribution. Les trois approches présentées ci-dessous offrent des solutions s'appuyant sur des modèles de conception différents : Daedalus / TACC pour le modèle client/serveur, SOMA pour le modèle à agents et MARS pour le modèle n-uplets.

3.2.3.1 Daedalus / TACC

Daedalus est la suite du projet BARWAN et reprend tous les aspects de transformations de données de celui-ci (voir paragraphe 3.1.1.2). Daedalus propose le modèle de programmation TACC (*Transformation, Aggregation, Caching, Customization*) pour construire des services de type Internet. Ceux-ci ont particulièrement été testés avec TranSend (réimplantation du serveur Pythia avec une grappe de PCs) et Top Gun Wingman (navigateur pour 3Com PalmPilot) [Fox et al. 1998a, Fox et al. 1998b].

Dans le modèle TACC, une application est construite à partir de blocs interconnectés par des interfaces simples. Chaque bloc, ou *Worker*, est spécialisé pour une tâche particulière comme une conversion entre deux formats de données. La connection des blocs se fait par composition en chaîne (comme un pipeline Unix) ou par appel de méthode.

À partir de cette construction, la partie minimale d'une application s'exécute sur le terminal portable et les autres blocs s'exécutent sur un serveur TACC. La figure 3.14 présente l'architecture de celui-ci. Les *Front-ends* reçoivent les requêtes des clients mobiles. Le *Load Balancing/Fault Tolerance Manager* est ensuite responsable (i) du bon fonctionnement des *Workers* : lancement, détection et redémarrage sur pannes, etc et (ii) de la transmission des requêtes aux *Workers* : choix du *Worker* approprié selon la spécification du profil utilisateur et selon des critères d'équilibrage de charge.

Dans l'application Top Gun Wingman, l'optimisation porte sur le délai entre le terminal portable et le serveur. Dans ce cas, l'amélioration ne porte pas sur la liaison sans-fil mais sur le

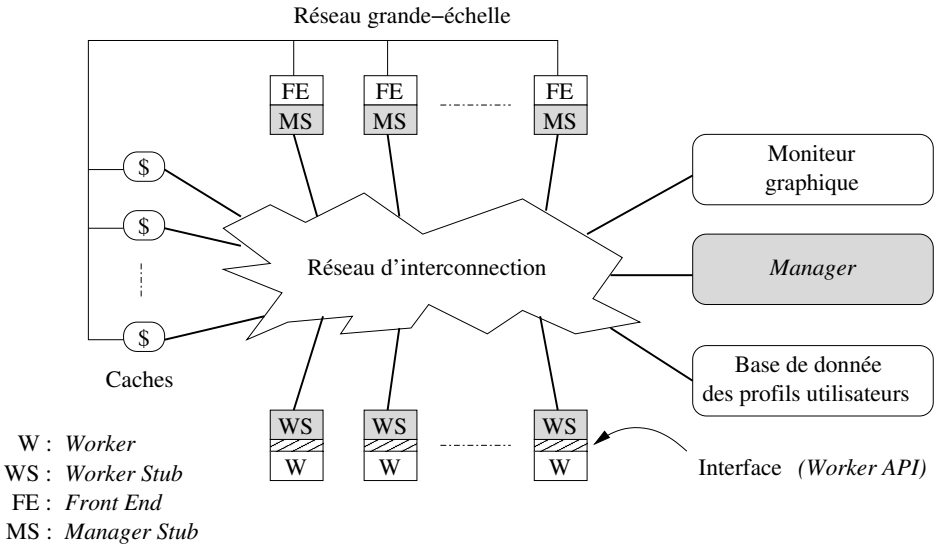


FIG. 3.14 – Architecture d'un serveur TACC

temps de réponse des composants de l'application qui s'exécutent sur le serveur. La stratégie employée est de type *overflow pool* [Fox et al. 1997]. L'indice de charge est calculé à partir du nombre de requêtes se trouvant dans les files d'attentes des *Workers*. Lorsque celui-ci dépasse un seuil H correspondant au délai maximum spécifié par l'utilisateur, le *Manager* exécute un nouveau *Worker* pour absorber la charge. Cette approche est extensible puisque, face à un flux important de requêtes, il suffit de rajouter des *Front-ends* et des *Workers*.

Cette approche présente néanmoins le défaut de n'avoir qu'un seul serveur et donc de ne pouvoir utiliser que les ressources définies par cet environnement. D'autres approches, comme Conductor [Yarvis et al. 2000] ou Panda [Reiher et al. 2000], reprennent les mêmes principes mais à plus grande échelle, en ayant, par exemple, un ensemble de *proxies* présents sur le réseau et permettant le déploiement d'*adaptors*.

3.2.3.2 SOMA

SOMA (*Secure and Open Mobile Agent*) [Bellavista et al. 1999] est un cadre de conception s'appuyant sur des agents mobiles Java et visant à assurer des propriétés de sécurité et d'interopérabilité. Pour protéger les agents entre eux ainsi que vis à vis de l'environnement d'exécution, SOMA suit des modèles de sécurité et propose un ensemble d'outils permettant la mise en place de politiques flexibles de sécurité. SOMA autorise l'interopérabilité entre différents composants d'une même application, ou d'applications différentes, en suivant les standards OMG CORBA et MASIF [OMG 2000b].

En plus des services implantant les propriétés précédentes, SOMA propose quatre services additionnels pour supporter la mobilité des utilisateurs et des terminaux [Bellavista et al. 2001b] (voir figure 3.15). Ceux-ci sont tous implantés avec des agents mobiles.

Le service MEN (*Mobility-enabled Naming*) est capable de traquer des entités se déplaçant

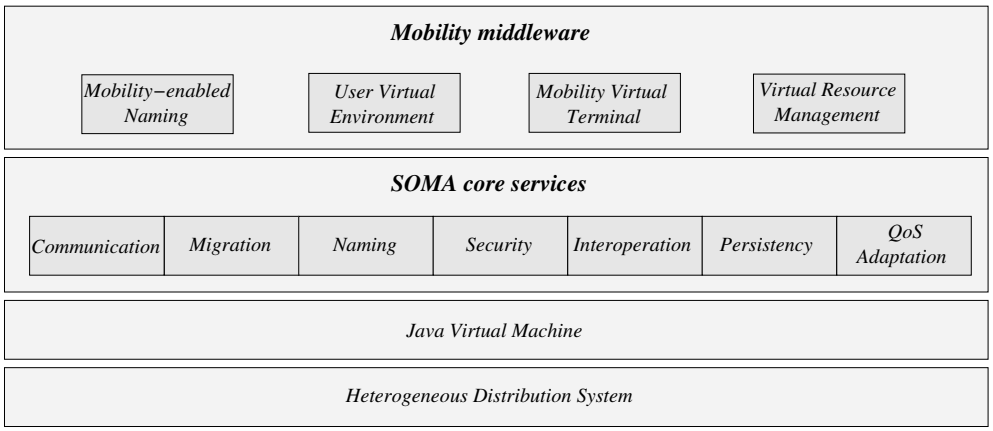


FIG. 3.15 – Architecture du système SOMA avec les services additionnels pour la mobilité

dans un environnement à grande échelle (de type Internet). Il offre les fonctionnalités de découverte et de répertoire. Celles-ci différent dans leur visibilité : la découverte est uniquement locale et est mise en œuvre à l'aide de protocoles de diffusion (*broadcast*) et d'agents distribués localement ; le répertoire est global et est mis en œuvre à l'aide d'une hiérarchie d'agents distribués sur l'ensemble du réseau.

Le service UVE (*User Virtual Environment*) fournit la possibilité aux utilisateurs de se connecter à différents endroits et sur différents terminaux tout en conservant leur configuration personnelle. Le profil utilisateur spécifie des informations sur les données propres à l'utilisateur, comme l'arrangement des icônes et menus de son affichage ou comme les clefs de cryptage pour son authentification. Il comporte également des informations relatives aux ressources et services utilisés, comme les caractéristiques requises pour une utilisation ordinaire ainsi que les contraintes d'adaptations pour des environnements donnés. Ce service est mis en œuvre de manière centralisé sur une station propre à l'utilisateur (*user home*). Pour des raisons de performance, ce profil est répliqué. Les agents replicas utilisent justement la politique de placement définie par l'utilisateur dans son profil. Celle-ci peut dépendre de critères comme la distance réseau, le temps de réponse ou le taux d'utilisation des machines.

Le service MVT (*Mobile Virtual Terminal*) assure la continuation de l'exécution des applications lors de la mobilité d'un terminal. Celle-ci consiste à utiliser le mécanisme de persistance, effectuer une requalification des ressources et services (nouvelle allocation dans le nouvel environnement en fonction de l'ancienne allocation), effectuer le transfert ou continuer l'exécution en distant ou hors-ligne. Ce service travaille en conjonction avec les services MEN et VRM : le changement de localisation est enregistré dans le premier service en cas de requalification positive, sinon l'exécution est enregistrée comme étant distante dans le second.

Le service VRM (*Virtual Resource Management*) assure la gestion des ressources et des services. Il en conserve les propriétés (localisation $\leftrightarrow$ identifiant, état, etc) et contient également une politique d'allocation des ressources et services. Cette dernière permet, par exemple, aux administrateurs de favoriser l'accès aux ressources locales et d'équilibrer la charge du système par une redistribution des agents. Différentes politiques d'optimisation de critères ont été testées pour différents types d'applications : le taux d'utilisation des

processeurs et des liens réseau avec une application de contrôle des ressources et des services [Bellavista et al. 2001a], la localisation géographique avec une application de recherche d'informations distribuées [Bellavista et al. 2000] et des critères de qualité de service vidéo avec une application de distribution d'application multimédia [Bellavista et Corradi 2002]. Ces politiques sont néanmoins spécialisées pour une application donnée et ne peuvent être modifiées que par les administrateurs et non par une application voulant introduire ses propres critères.

3.2.3.3 MARS

MARS (*Mobile Agent Reactive Spaces*) [Cabri et al. 2000c] est un système de coordination d'interactions entre agents mobiles ou entre agents mobiles et ressources de l'environnement. Il définit un espace de n-uplets qui peut être programmé avec des réactions spécifiques.

Pour régler le problème d'allocation compétitive des ressources, une solution implantée dans MARS consiste à utiliser une politique à enchères [Cabri et al. 2000a]. Les avantages de MARS pour la mise en œuvre d'une telle politique sont que (i) le modèle Linda permet d'implanter les services vendeurs et acheteurs de manière simple et uniforme et (ii) la propriété de programmabilité de MARS permet de découpler les politiques à enchères, contenues dans les agents mobiles, des mécanismes de l'enchère qui se trouvent dans l'espace de n-uplets.

La figure 3.16 présente le déroulement d'une enchère. L'agent vendeur enregistre d'abord une offre de vente comprenant la description de la ressource à vendre, une limite de temps, un prix de départ et le type de l'enchère. Le type de l'enchère correspond à une spécialisation des mécanismes de la politique à enchères dans l'espace de n-uplets (agent *Reaction* jouant le rôle de commissaire-priseur). Chaque agent acheteur peut ensuite enregistrer un n-uplet comme enchère. L'agent *Reaction* (i) surveille ces n-uplets, (ii) décide de l'agent acheteur remportant l'enchère lorsque celle-ci est terminée et (iii) écrit un n-uplet pour informer tous les participants.

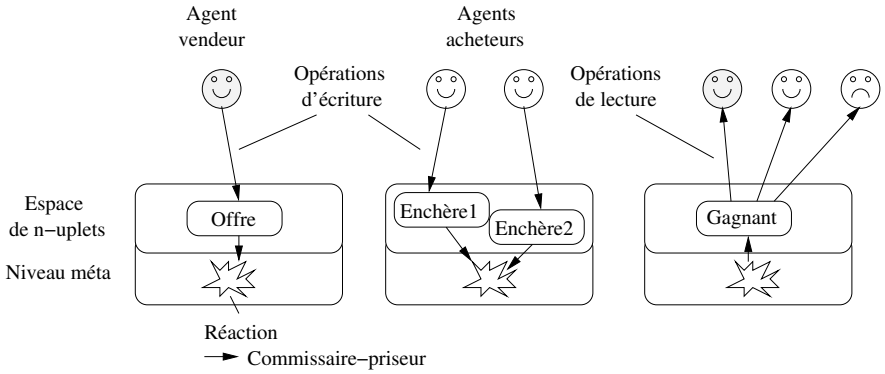


FIG. 3.16 – Allocation de ressources par politique à enchères dans MARS

MARS fournit un système de surveillance des ressources, et donc n'importe quel critère peut être pris en compte dans une politique à enchères. Dans [Cabri et al. 2000a], les critères de prix et d'identité de l'acheteur (enchère publique ou privée) sont testés dans deux politiques

à enchères (*First-Price Auction, Vickery Auction*). Dans [Cabri et al. 2000b], les mobilités logique et physique sont prises en compte pour effectuer des coordinations dépendantes de l'environnement d'exécution (notamment, utilisation de la localisation dans [Bonfigli et al. 2001]).

3.2.4 Résumé

Spécialisation de l'adaptation en fonction de l'implantation : le paragraphe 3.2.2 conforte l'idée qu'il est nécessaire d'avoir plusieurs niveaux d'adaptation selon que l'on souhaite adapter au niveau des moyens de communications, des intergiciels ou de l'application. Il démontre également que, pour un même niveau d'adaptation, plusieurs mises en œuvre sont possibles (clients/serveurs, agents, espaces de n-uplets) et que chacune d'elles possèdent des adaptations propres. Une des adaptations intrinsèque à un agent mobile est de se déplacer, cette adaptation n'a aucun sens pour un espace de n-uplets. *Un système d'adaptation doit proposer un cadre de conception permettant de définir une adaptation de manière générique et que celle-ci puisse être spécialisée en fonction de l'implantation souhaitée*.

Répartition et environnement : une stratégie de gestion des ressources travaille en fonction de définitions des ressources et de critères de répartition. Dans un environnement mobile, ces définitions peuvent être amenées à changer : déplacement dans un nouveau lieu possédant une ressource non définie jusqu'alors, nouveau critère de répartition à prendre en compte (proximité de bornes multimédia, etc). *un système de gestion de ressources en environnements mobiles doit donc caractériser de manière générique ce qu'est (i) un environnement mobile : éléments, variabilité, etc et (ii) les services de gestion de cet environnement : critères, stratégies de répartition, etc*.

Stratégie de répartition dynamiquement adaptable : l'adaptabilité dynamique est encore plus cruciale pour les stratégies de gestion des ressources que pour les stratégies de gestion des données. En effet, l'emploi d'une stratégie de gestion des données inappropriée peut amener des pertes de performance mais cela ne change pas le comportement de l'application. Par contre, si une stratégie de gestion des ressources gère de manière inadéquate l'arrivée ou le départ de ressources et services, les applications les utilisant peuvent se terminer brutalement. *La stratégie de gestion des ressources d'un système d'adaptation en environnements mobiles doit donc dynamiquement pouvoir changer pour s'adapter aux changements de conditions de l'environnement ou aux changement de caractérisation des ressources ou de l'environnement*.

Deuxième partie

Généricité, gestion des ressources et distribution des applications en environnements mobiles

Chapitre 4

Organisation générale du système d'adaptation

Ce chapitre présente l'organisation générale du système d'adaptation proposé pour les environnements mobiles. Celle-ci se présente sous la forme d'un *cadre de conception* et d'une *boîte à outils*. Le cadre de conception offre un ensemble de *fonctionnalités* communément utilisées pour la gestion des environnements mobiles. La boîte à outils comprend un ensemble d'*implantations* correspondant à des mises en œuvre des fonctionnalités du cadre de conception. La généralité du cadre de conception et la diversité des implantations de la boîte à outils garantissent l'indépendance par rapport aux domaines des applications. Une application trouvera donc toutes les fonctionnalités dont elle a besoin dans le cadre de conception, et, par spécialisation, les instanciera par une implantation de la boîte à outils ou par une implantation qu'elle pourra fournir.

Ce chapitre s'organise comme suit. Le paragraphe 4.1 rappelle les propriétés que doit respecter notre système. Le paragraphe 4.2 définit quelques concepts de la méthodologie objet, notamment les notions de cadre de conception et de boîte à outils. Le paragraphe 4.3 présente l'architecture du système d'adaptation et, plus particulièrement, le paragraphe 4.3.1 décrit les fonctionnalités faisant parties du cadre de conception et le paragraphe 4.3.2 présente les principales implantations se trouvant dans la boîte à outils.

4.1 Objectifs

L'objectif de cette étude est la conception d'un système adaptatif de distribution d'applications en environnements mobiles. Il nous est paru important que ce système vérifie un ensemble de propriétés répondant aux caractéristiques des systèmes de l'informatique mobile et des systèmes de distribution : la *généricité* pour l'utilisation par différentes familles d'applications, la *modularité* pour un découpage et un découplage adéquat, l'*adaptabilité* avec la *prise en compte de l'environnement et des applications*, l'*évolutivité* pour correspondre à de nouvelles technologies ou de nouvelles fonctionnalités, la *dynamicité* afin de réagir aux changements sans pour autant arrêter le système et l'*efficacité* en terme de *performances* et de *stabilité*. Ces propriétés sont les mêmes que celles énoncées dans l'introduction du document.

La suite du chapitre détaille l'architecture du système d'adaptation que nous proposons et comment celle-ci satisfait les propriétés de généricité, de modularité, d'adaptabilité et d'extensibilité à l'aide d'un découpage en cadre de conception (paragraphe 4.3.1) et boîte à outils (paragraphe 4.3.2). Le chapitre 5 présente de manière plus approfondie les propriétés d'adaptabilité et de dynamicité. Le chapitre 6 s'attache à la gestion de l'environnement, de l'application et à la distribution. Enfin, le chapitre 7 revient sur les problèmes d'efficacité associés.

4.2 Définitions

Quelques termes, empruntés à la terminologie du domaine de la conception par objets, sont utilisés par la suite et nécessitent donc d'être définis. Les trois termes ci-dessous décrivent des outils de conception d'applications. Ils sont présentés du niveau le plus concret vers le plus abstrait :

Boîte à outils (*toolkit*) **:** *« A toolkit is a set of related and reusable classes designed to provide useful, general-purpose functionnality. An example of a toolkit is a set of collection classes for lists, associate tables, stacks and the like. »* [Gamma et al. 1995].

Une boîte à outils n'impose aucune particularité de conception d'une application. Elle permet juste la réutilisabilité en fournissant des implantations de fonctionnalités générales répondant aux besoins spécifiques de conception d'une application. Une boîte à outils peut être comparée aux bibliothèques de procédures ou d'appels.

Cadre de conception (*framework*) **:** *« A framework is a set of cooperating classes that make up a reusable design for a specific class of software. »* [Gamma et al. 1995].

Un cadre de conception définit le squelette d'une famille d'applications. Ce squelette décrit une partie de l'architecture des applications impliquant les fonctionnalités utilisées ou offertes (classes, objets, services, etc) et les interactions entre celles-ci (appels, dépendances, contrôle, etc). À la différence d'une boîte à outils, un cadre de conception n'est pas utilisable "en l'état" (exécutable). Il doit être complété, spécialisé (voir définition ci-après) pour la création d'une application. Un cadre de conception offre donc l'aide à la conception, la réutilisabilité et la maintenabilité pour une famille d'applications.

Patron de conception (*design pattern*) **:** *« [A design pattern is] the description of communicating objects and classes customized to solve general design problem in a particular context. [...] Each design pattern lets some aspect of system structure vary independently of other aspects, thereby making a system more robust to a particular kind of change. »* [Gamma et al. 1995].

Un patron de conception est une solution identifiée et réutilisable à un problème récurrent de conception dans un contexte précis. Cette solution résulte de, et renferme, l'expérience de concepteurs sur un point précis de conception. Celle-ci offre alors un vocabulaire défini permettant une meilleure réutilisabilité par les concepteurs. Un patron de conception est plus abstrait qu'un cadre de conception puisqu'il existe en tant qu'entité logique alors qu'un cadre de conception possède une représentation physique (logicielle). Un patron de conception est également plus générique et présente une architecture beaucoup plus réduite qu'un cadre de conception puisqu'il doit répondre à un problème précis et non s'adapter à une classe d'applications.

Les deux termes ci-dessous décrivent des techniques utilisées lors de l'exécution d'applications :

Instanciation (*instantiation*) **:** *« Every object is an instance of a class. [...] A class specifies the structure and the behavior of all its instances. [...] Instances of a class share the same behavior and have a specific state. »* [Goldberg et Robson 1989].

De manière plus générale, l'instanciation est l'action de créer une entité exécutable (objet, composant, service, application, etc) à partir de sa description (classes pour les objets, etc). Chaque entité s'exécute selon un comportement défini dans sa description mais possède un état interne propre. L'instanciation d'une application s'effectue à partir d'un cadre de conception ayant été spécialisé.

Spécialisation (*specialization*) **:** *« A framework defines a high-level language with which applications within a domain are created through specialization. »* [Pree 1997].

La spécialisation consiste à compléter et/ou à altérer un cadre de conception afin de finaliser son comportement. Cette action rend instanciables les applications décrites par ce cadre. La spécialisation s'effectue en insérant du code à certains endroits du cadre de conception (décrits sous différents noms dans la littérature : *variability points*, *hooks*, *hotspots*, etc).

4.3 Architecture du système d'adaptation

Dans le chapitre 2, nous avons présenté les approches transparentes, applicatives et réflexives et avons mis en exergue qu'aucune d'entre-elles n'est pleinement satisfaisante vis à vis de la puissance d'expression de l'adaptation et de la facilité et de l'adéquation de l'utilisation en environnements mobiles.

Le système d'adaptation que nous proposons reprend plusieurs points positifs de ces approches. Il consiste en une approche réflexive proposant un grand nombre de fonctionnalités dédiées aux environnements mobiles. Le choix de l'approche réflexive donne à notre système une grande capacité d'expression de l'adaptation. Les implantations optimisées des fonctionnalités le rendent approprié aux environnements mobiles.

Notre système s'appuie sur un découpage entre la structure abstraite des fonctionnalités et leur mise en œuvre concrète au sein d'implantations. Les fonctionnalités sont regroupées dans un cadre de conception et les implantations se trouvent dans une boîte à outils. L'instanciation du système d'adaptation s'effectue par spécialisations de notre cadre de conception. Ces spécialisations peuvent s'appliquer soit (i) statiquement (avant l'instanciation) par les concepteurs lors du développement, soit (ii) dynamiquement (à l'instanciation ou pendant l'exécution) par tout intervenant "autorisé" à modifier le comportement du système d'adaptation. Les intervenants pouvant effectuer ces modifications sont détaillés ci-dessous, mais les notions de sécurité liées aux autorisations d'effectuer les modifications ne seront pas traitées dans cette thèse.

Lors du développement d'une application devant s'exécuter en environnement mobile, les concepteurs de celle-ci sont donc amenés à faire des choix de spécialisation de notre cadre de conception. Ces choix sont illustrés dans la figure 4.1. Ils s'apparentent à différentes techniques de spécialisation :

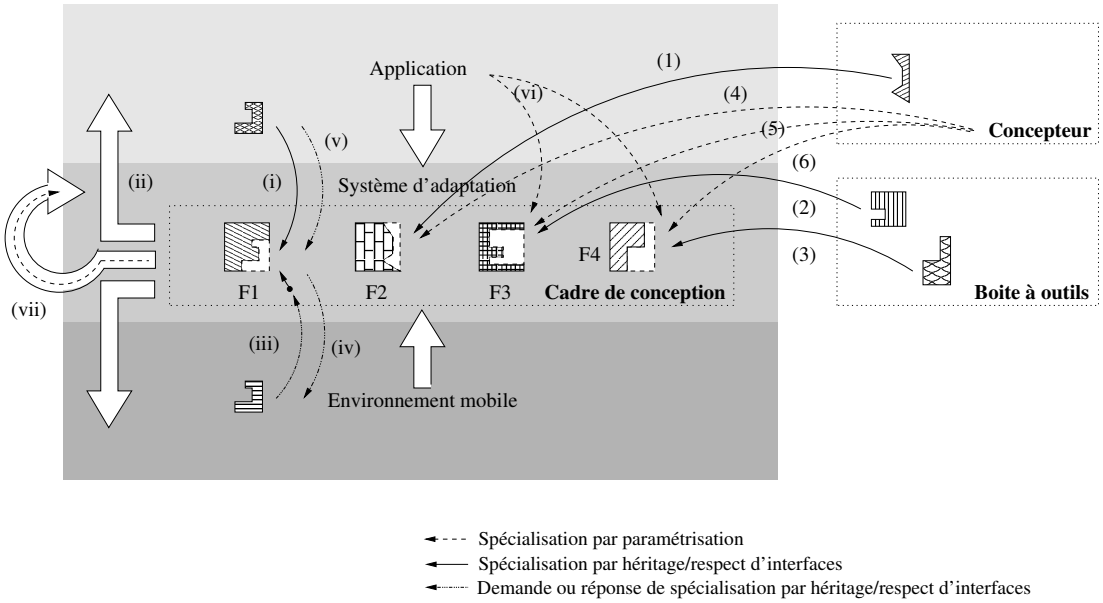


FIG. 4.1 – Spécialisation et instanciation (ou spécialisation et modification d'instance) du système d'adaptation

(i) La composition

statique : une des premières étapes consiste à choisir les fonctionnalités parmi la diversité offerte par le cadre de conception. Ce choix est laissé aux concepteurs puisque ce sont eux qui connaissent le mieux les besoins de leur application et sont alors à même de connaître les fonctionnalités les plus appropriées. Ceux-ci vont alors créer le système d'adaptation par composition des fonctionnalités choisies. Dans la figure 4.1, le système d'adaptation est composé des fonctionnalités F1, F2, F3 et F4.

dynamique : la composition dynamique ne peut se faire que dans le cas où l'application elle-même est adaptative. En effet, par rapport au moment de sa déclaration de besoins en fonctionnalités, celle-ci doit avoir un changement de comportement et déclarer de nouveaux besoins en fonctionnalités. Lors de l'instanciation de ce nouveau comportement, le système d'adaptation se recomposera alors avec les nouvelles fonctionnalités.

(ii) L'héritage/respect d'interfaces

statique : une des étapes suivantes consiste à compléter les fonctionnalités (structures abstraites) du cadre de conception par des implantations (mises en œuvre concrètes). Cette complétion peut s'effectuer en utilisant des mécanismes d'héritage ou tout simplement en construisant des implantations qui respectent les interfaces de la structure. Comme le montre la figure 4.1, l'implantation d'une fonctionnalité peut entièrement être effectuée par un concepteur (étape 1). Notre boîte à outils fournit également des implantations communément utilisées dans les environnements mobiles (étapes 2 et 3).

dynamique : la complétion dynamique des structures du cadre de conception par des implantations peut toujours être effectuée par les concepteurs de l'application à partir de

leurs propres implantations (étape 1) ou celles de notre boîte à outils (étapes 2 et 3). Cette spécialisation peut également être effectuée par d'autres intervenants "humains" comme les administrateurs (de l'application, du système d'adaptation, etc). Deux intervenants logiciels peuvent aussi déclencher et réaliser la spécialisation : l'application et le système d'adaptation lui-même. Ainsi, l'application peut avoir été programmée non seulement pour modifier son comportement d'exécution mais aussi les stratégies d'adaptation du système (voir figure 2.1 du chapitre 2.1). L'application peut alors spécialiser une fonctionnalité du système d'adaptation avec une implantation lui convenant (étape i). L'application prend sa décision de spécialisation selon ses propres critères, comme, par exemple, en réaction à un évènement de changement d'état de l'environnement mobile fourni par le système d'adaptation (étape ii). L'implantation d'une fonctionnalité peut encore être fournie par l'environnement mobile, par exemple, à l'aide d'un serveur (étape iii). Dans ce cas, c'est soit le système d'adaptation lui-même (étape iv puis iii) soit l'application par l'intermédiaire du système d'adaptation (étapes v, iv puis iii) qui est à l'origine et réalise la spécialisation. Notre système n'offre pas la possibilité à l'environnement d'effectuer les changements, ce domaine de recherche est plutôt abordé dans les systèmes d'informatique de proximité et systèmes d'informations spontanés [Touzet et al. 2001].

(iii) La paramétrisation (ou configuration)

statique : une des étapes suivantes consiste à paramétrer les fonctionnalités précédemment spécialisées. Il peut s'agir, pour les concepteurs, de modifications de l'état interne d'une ou plusieurs implantations, ou de modifications des paramètres à prendre en compte lors des interactions entre les fonctionnalités. Ces paramétrisations peuvent concerner les implantations provenant des concepteurs (étape 4) comme celles de la boîte à outils (étapes 5 et 6).

dynamique : de la même manière que statiquement, la spécialisation par paramétrisation est possible dynamiquement par les concepteurs (ou les administrateurs) de l'application (étapes 4, 5 et 6). Cette paramétrisation peut également être réalisée par l'application (étape vi) ou par le système d'adaptation (étape vii).

(iv) L'ajout avec altération

statique : notre système d'adaptation étant ouvert, les concepteurs peuvent accéder aux sources du cadre de conception et de la boîte à outils. Ils peuvent ainsi (i) modifier les interfaces et le corps des fonctionnalités et implantations existantes ou (ii) inclure de nouvelles fonctionnalités ou de nouvelles implantations avec leurs propres interfaces. Toutefois, notre système n'offre aucun outil d'aide ou d'automatisation de cette tâche. Des outils extérieurs, comme OpenJava [Chiba et Tatsubori 1998, Tatsubori et al. 2000], peuvent être alors utilisés. Dans ce cas, la cohérence de notre système, ainsi que celle de l'application construite, sont laissées à la charge des concepteurs. Nous entendons ici par cohérence, la cohérence structurelle du système et de l'application. Garantir cette cohérence assure qu'à la compilation ainsi qu'à l'exécution, il n'y aura pas de terminaison non prévue du système ou de l'application due à un comportement ou une interaction inappropriés entre fonctionnalités.

dynamique : notre système d'adaptation n'offre pas d'outils de modification du code compilé ni d'interception des appels. Des outils extérieurs, comme

Javassist [Chiba 1998a, Chiba 2000] ou Iguana/J [Redmond et Cahill 2000], peuvent être utilisés. Comme dans le cas statique, la cohérence structurelle du système est alors laissée aux "modificateurs". De plus, la cohérence fonctionnelle de certaines implantations de la boîte à outils ne peut plus être assurée. Par exemple, les implantations de la fonctionnalité de communication garantissent la non-perte d'un message sur le lien sans-fil. Cette garantie ne peut plus être assurée si, par exemple, les appels effectuant la mise en tampon des messages sont interceptés.

Notre approche présente plusieurs avantages. Le découpage en cadre de conception et boîte à outils favorise (i) la réutilisation : les fonctionnalités proposées dans le cadre de conception sont suffisamment génériques pour pouvoir être utilisées par un grand nombre d'applications et les implantations proposées dans la boîte à outils sont suffisamment "standards" pour s'appliquer à un grand nombre d'environnements mobiles, (ii) la facilité de développement : notre système traitant des aspects sans-fil, les concepteurs d'une application peuvent se concentrer pleinement sur les aspects purement applicatifs ; cette séparation est d'autant plus effective que nous proposons ces facilités au sein de fonctionnalités permettant de ne pas entremêler le code applicatif et le code de gestion du sans-fil.

Les techniques de spécialisation contribuent, de plus, à l'amélioration de (i) la facilité de développement : la spécialisation par composition offre la flexibilité du choix, de l'ajout, de retrait de fonctionnalités ; la spécialisation par héritage/respect d'interfaces offre la facilité de complétion avec de nombreuses alternatives dans la boîte à outils, (ii) la maintenabilité : les possibilités de spécialisations en cours d'exécution permettent aux concepteurs ou aux administrateurs de faire évoluer dynamiquement le système d'adaptation.

4.3.1 Le cadre de conception

Notre cadre de conception est constitué de fonctionnalités répondant aux besoins des applications et aux contraintes des environnements mobiles. La figure 4.2 présente ces principales fonctionnalités. Leur structure est conçue de manière générique afin d'obtenir une réutilisabilité maximale. Leur utilisation est illustrée par plusieurs applications sans-fil dans [André et Segarra 1999, André et Saint Pol 2000, André et al. 2000, Le Mouël et al. 2002] et dans la chapitre 7.

Gestion des communications. Cette fonctionnalité permet à des entités distantes d'établir une connexion fiable à travers un lien sans-fil. Ici, une connexion fiable s'entend par la non-perte et la non-altération des données transmises malgré des déconnexions possibles. La garantie de non-famine ne peut être assurée du fait qu'une déconnexion peut se prolonger à l'infini. Les concepteurs peuvent ainsi spécialiser cette fonctionnalité en fournissant des implantations qui garantissent le niveau de fiabilité désiré.

Gestion des données. La fonctionnalité de gestion des données se découpe en deux sous-fonctionnalités : le stockage et la cohérence des données.

Le stockage est réalisé par un Système de Gestion des Données (SGD). Celui-ci peut être implanté de diverses manières comme par des Systèmes de Gestion de Base de Données (SGBDs) (*SQL, Oracle, etc) ou par des systèmes de fichiers (locaux : ext2, FAT, etc,

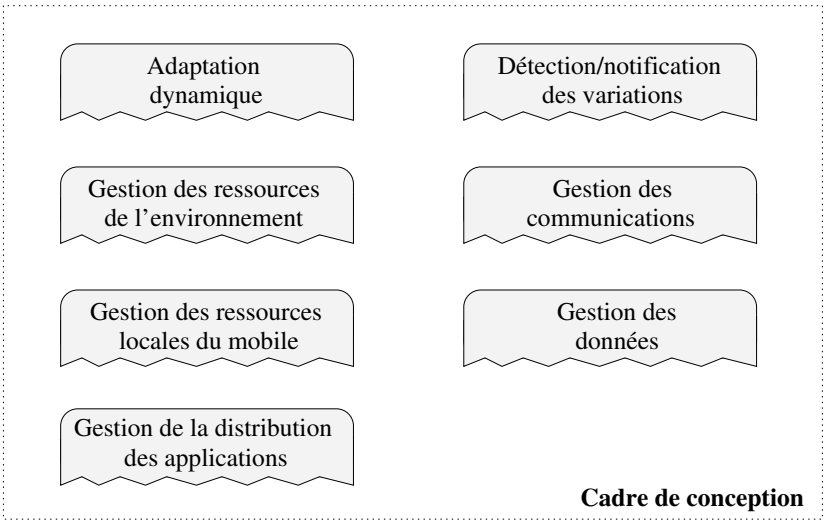


FIG. 4.2 – Fonctionnalités du cadre de conception

ou distants : NFS, Samba, etc). Notre SGD fournit deux abstractions : les entités *Data* et *MetaData*. La première permet de transformer les accès génériques au SGD en accès spécifiques au SGBD ou au système de fichiers. Cette entité contient principalement la donnée elle-même et les opérations réalisables sur celle-ci. L'entité *MetaData* permet d'obtenir des informations pertinentes sur la donnée, comme sa taille, le temps d'accès, etc. Celles-ci sont notamment utiles pour le paramétrage de fonctionnalités faisant des transferts sur le lien sans-fil.

Le stockage peut être réalisé en tout ou partie sur le terminal portable. Comme cela est précédemment présenté dans le chapitre 3.1.3, ce stockage sur le terminal portable introduit le problème de cohérence des données. Notre système résout ce problème à l'aide d'une fonctionnalité de gestion de la cohérence des données. Celle-ci se trouve sur chaque machine (serveur ou terminal portable) où des données sont répliqués. Ainsi, elle actualise les données locales avec les modifications provenant de machines possédant des copies, et propage, également, vers ces mêmes machines, toutes modifications sur les données locales. Les concepteurs peuvent spécialiser cette fonctionnalité de manière à mettre en place les stratégies de cohérence, de diffusion et de mise à jour qu'ils désirent.

Détection et notification des variations. Cette fonctionnalité offre la possibilité de surveiller les changements intervenant dans l'environnement mobile. Celle-ci en détecte les variations et notifie alors le système en charge de la réaction (système réactif) qui peut être un ensemble d'applications ou le système d'adaptation.

Elle se présente sous la forme d'un ensemble de moniteurs et d'un gestionnaire de moniteurs. Le gestionnaire de moniteur gère la durée de vie de ceux-ci (création, (ré)utilisation, destruction) en fonction des attentes du système réactif. Les moniteurs peuvent être de deux natures : les moniteurs de base (MBs) ou les moniteurs de haut niveau (MHNs). Les MBs surveillent une ressource basique unique, comme le processeur, la mémoire, etc. Les MHNs permettent d'exprimer des conditions de surveillance plus complexes à partir des informations

fournies par les MBs. Ainsi, par exemple, un MHN de qualité de service d'une connexion réseau peut être défini à partir des MBs surveillant la bande passante, le temps de latence et le taux de perte de la liaison sans-fil.

Les concepteurs peuvent spécialiser le gestionnaire de moniteurs pour permettre la création de nouveau type de moniteurs ou définir la politique de réutilisation des moniteurs existants. Ils peuvent également spécialiser les MBs et les MHNs pour être au plus près des caractéristiques physiques du système d'exploitation et du réseau sous-jacents au système d'adaptation[1].

Les fonctionnalités de gestion des communications, gestion des données et détection et notification des variations ont été traitées plus en détails dans [Segarra 2000]. Leurs implantations sont brièvement décrites dans le paragraphe ci-dessous, mais elles ne seront pas plus explicitées dans la suite du document. Les fonctionnalités restantes constituent les points importants qui seront développés dans cette thèse et font donc l'objet de chapitres à part entière. La fonctionnalité d'adaptation dynamique est décrite dans le chapitre 5. Les fonctionnalités de gestion du terminal portable, de l'environnement et de la distribution des applications sont plus amplement expliquées dans le chapitre 6.

4.3.2 La boîte à outils

Notre boîte à outils fournit des implantations pleinement adaptées à certaines caractéristiques des environnements mobiles. Les concepteurs peuvent les utiliser pour spécialiser les fonctionnalités du cadre de conception. La figure 4.3 présente synthétiquement quelques-unes des principales implantations disponibles pour chacunes des fonctionnalités :

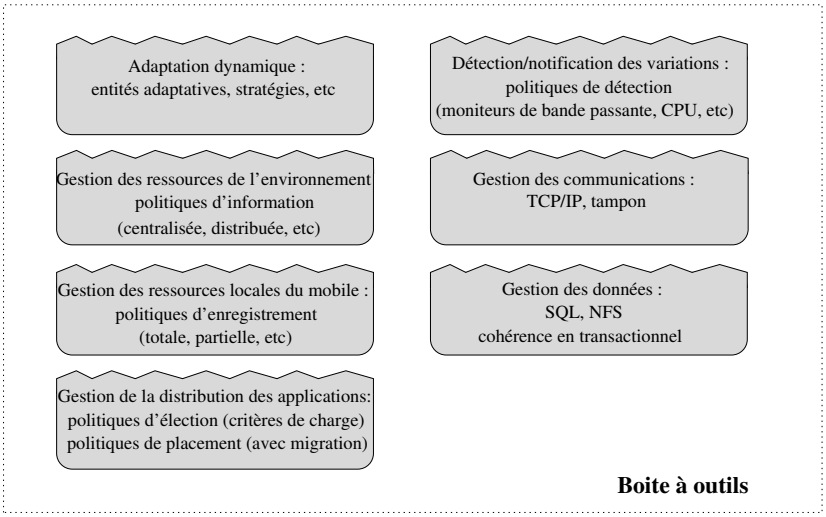


FIG. 4.3 – Principales implantations de la boîte à outils

[1] Les performances d'un système de détection et notification dépendent, en effet, grandement de l'adéquation entre les algorithmes de surveillance utilisés et les caractéristiques physiques du système d'exploitation et du réseau sous-jacents [Hollingsworth et Miller 1993].

Gestion des communications. L'implantation proposée pour la fonctionnalité de gestion des communications assure la fiabilité des connexions logiques en dépit de déconnexions physiques pouvant provisoirement survenir sur le lien sans-fil. Celle-ci s'appuie sur la couche de communication TCP/IP. Lors du bon fonctionnement du lien sans-fil, la non-perte et la non-altération sont assurés par les mécanismes d'acquittements et de contrôle des erreurs de TCP/IP. Lors d'une déconnexion, les connexions logiques sont maintenues mais les connexions TCP/IP sont fermées. Les messages envoyés à partir de ce moment (ainsi que ceux n'ayant pas été acquittés) sont stockés dans un tampon et leur ordre d'émission est conservé dans un historique. Lors de la reconnexion, de nouvelles connexions TCP/IP sont établies et les messages stockés sont réémis dans l'ordre de l'historique.

Gestion des données. Pour chacune des sous-fonctionnalités de la gestion des données, le stockage et la cohérence, deux implantations sont proposées. Le stockage peut s'effectuer par accès au système de fichiers NFS ou par accès à une base de donnée SQL ou Oracle.

Pour l'implantation utilisant NFS, les lectures et écritures sont effectuées localement sur des copies maintenues sur le terminal portable. Les modifications sont réintégrées par NFS sur les données du serveur (suivant la procédure implantée dans la fonctionnalité de cohérence décrite ci-dessous). Dans cette implantation, l'entité *Data* est spécialisée en *Bloc* autorisant l'accès à un bloc d'un fichier NFS.

Pour l'implantation accédant aux SGBDs, les transactions s'effectuent sur des bases de données SQL ou Oracle locales au terminal portable. Celles-ci sont ensuite répercutées sur les bases de données se trouvant sur les serveurs ou les autres terminaux portables (toujours suivant la procédure implantée dans la fonctionnalité de cohérence décrite ci-dessous). Dans cette implantation, l'entité *Data* est spécialisée en *Row* et *Table* autorisant l'accès à une ligne et à une table des bases de données.

Deux implantations sont également proposées pour la sous-fonctionnalités de gestion de la cohérence : cohérence par invalidation périodique ou cohérence transactionnelle.

La cohérence par invalidation est assurée par un intermédiaire sur le réseau fixe. Celui-ci reçoit les demandes d'accès aux fichiers NFS. Il peut donc vérifier périodiquement si les données présentes sur les autres stations ne sont pas plus récentes que celles maintenues sur le terminal portable ($si \exists\ m \in \{serveurs,\ terminaux\ avec\ la\ donnée\ d\}\ tq\ mtime_m(d) > mtime_{terminal\ portable}(d)$). Dans le cas où une donnée plus récente est trouvée, une demande d'invalidation est envoyée au terminal portable qui charge alors la nouvelle version de cette donnée. Le nombre d'invalidations (et consécutivement le trafic sur le lien sans-fil) dépend alors du taux de partage des données entre les applications.

La cohérence transactionnelle consiste à permettre d'exécuter des transactions localement sur le terminal portable. Ces transactions sont numérotées à l'aide d'estampilles (produites à partir de la fonction *mtime*). Elles sont ensuite réintégrées périodiquement sur le serveur ou les autres terminaux portables en suivant l'ordre des estampilles. Cet ordre suppose qu'il y ait peu de conflits permettant ainsi d'entrelacer l'exécution des transactions. Les conflits détectés sont de type écriture/écriture et écriture/lecture ; ils invalident alors les transactions incriminées et obligent leurs réexécutions.

Détection et notification des variations. Les implantations des moniteurs doivent tenir compte au plus près des caractéristiques des systèmes et réseaux sous-jacents. Dans la boîte à outils, nous proposons des implantations qui adressent périodiquement les commandes d'un système Linux sous-jacent.

Les moniteurs de surveillance de la mémoire et du fichier d'échange (*swap*) mesurent la disponibilité à l'aide de la commande *free*. Les moniteurs de surveillance du processeur mesurent la charge et le taux d'utilisation de celui-ci à l'aide de la commande *top*. Les moniteurs de surveillance de la batterie utilisent *« The Linux APM Daemon (Advanced Power Management) »* (commandes : *apmd*, *apm*) [Pennarun] pour mesurer le pourcentage de charge, le temps de vie restant et le taux de décharge de la batterie. Les moniteurs de surveillance du réseau sans-fil utilisent *« The Wireless tools for Linux »* (commandes : *iwconfig*, *iwspy*, *iwlist*, *iwpriv*) [Tourrilhes] pour mesurer la qualité du lien, la force du signal et le niveau du bruit. Le moniteur de surveillance du débit de la bande passante ne fonctionne pas uniquement à partir d'informations locales. Le concepteur spécifie l'adresse d'une station avec laquelle il souhaite calculer le débit de bout en bout ; le moniteur lui envoie périodiquement des messages d'une taille définie ; la station réceptrice calcule alors le nombre de bits reçus par unité de temps.

Comme pour le cadre de conception, les implantations des fonctionnalités d'adaptation dynamique, de gestion du terminal portable, de l'environnement et de distribution des applications ne sont pas détaillées ci-dessus, mais leurs mises en œuvre sont présentés dans le chapitre 7.

4.4 Conclusion

Travaux similaires. Par rapport aux approches présentées dans le chapitre 2, nous avons essayé de reprendre les points positifs de chacune (que nous avions listés dans le tableau 2.2). Pour la puissance d'expression de l'adaptation, notre système repose sur une approche réflexive. Il reprend la modularité des approches applicatives en introduisant plusieurs découpages. Et, enfin, il tente d'être aussi performant que les approches transparentes avec des mises en œuvre optimisées pour les environnements mobiles.

Respects des objectifs. La généricité de notre système est assurée par le découpage en cadre de conception et boîte à outils. Cette modularité, fonctionnalités et implantations, donne aux concepteurs (i) une vision synthétique de ce qu'offre notre système, ainsi qu'(ii) une facilité d'utilisation reposant sur la diversité des choix possibles.

Toutes les fonctionnalités proposés traitent d'un point bien particulier de l'environnement mobile. Certaines s'attachent plus au terminal portable, comme la gestion des communications, la gestion des ressources locales et la détection des variations. Et les autres s'intéressent plus aux paramètres externes influant sur le terminal portable, comme la gestion des ressources de l'environnement, la gestion de la distribution des données et des applications. Grâce aux techniques d'introspection et d'intersession utilisés, ces fonctionnalités peuvent être dynamiquement spécialisés selon différentes techniques : choix des fonctionnalités par composition, modification de la configuration des fonctionnalités par paramétrisation et incorporation de ses propres implantations par héritage/respect des interfaces. Efin, pour chacune de ces fonctionnalités, des implantations optimisées sont également proposées.

Chapitre 5

Conception d'un système d'adaptation et de réaction dynamique à granularité variable

Les fonctionnalités présentées dans le chapitre 4 peuvent toutes changer de comportement (peuvent être spécialisées par différentes implantations). Ces changements ou *adaptations* peuvent être dynamiques, consécutivement, par exemple, aux variations observées par la fonctionnalité de détection et notification.

Ce chapitre présente le cœur de notre système, sa partie adaptation et réaction. Le paragraphe 5.1 reprend les propriétés requises pour notre système et les affine pour la partie adaptation et réaction. Le paragraphe 5.2 décrit le modèle de notre système d'adaptation et de réaction. Le paragraphe 5.2.1 définit notamment notre unité logicielle de base, *l'entité*. Les paragraphes suivants détaillent ensuite les solutions apportées aux différentes problématiques : comment adapter l'entité ? par la définition d'*entité adaptative* (paragraphe 5.2.2), comment la rendre réactive ? par la mise en place de *stratégies d'adaptation* (paragraphe 5.2.3), comment effectuer de multiples adaptations ? en proposant la synchronisation des adaptations (paragraphe 5.2.4). Dans le paragraphe 5.3, nous effectuons les relations entre notre modèle de conception et les modèles de conception existants, principalement par une hiérarchisation de représentation des abstractions de conception (paragraphe 5.3.1). Enfin, nous concluons en revenant sur les travaux similaires et sur les objectifs fixés.

5.1 Propriétés du système d'adaptation et de réaction

La fonctionnalité d'adaptation et de réaction dynamique fait partie intégrante de notre cadre de conception et doit donc satisfaire les mêmes propriétés :

Généricité : les concepteurs, les administrateurs, les applications et le système d'adaptation ont la possibilité de changer le comportement des fonctionnalités par différentes spécialisations. Pour pouvoir autoriser cela, le modèle d'adaptation proposé doit donc pouvoir s'appliquer à toutes les fonctionnalités du cadre de conception.

Modularité : le découpage en fonctionnalités et implantations constitue une granularité facile d'utilisation pour les concepteurs. Toutefois, si ceux-ci souhaitent effectuer des traitements plus fins (par exemple, au niveau d'un objet) ou plus grossiers (par exemple, au niveau d'une application grande-échelle), notre système doit proposer une échelle de granularité d'adaptation.

Adaptabilité :

Prise en compte de l'environnement : pour pouvoir réagir et s'adapter aux variations de l'environnement, notre système d'adaptation doit (i) interagir avec la fonctionnalité de détection et de notification, et (ii) permettre de spécifier, au sein d'une stratégie d'adaptation, les choix d'adaptations qui doivent alors être appliqués.

Prise en compte de l'application : tout comme l'application peut spécialiser les fonctionnalités avec ses propres implantations, l'application doit pouvoir spécialiser les adaptations possibles et les stratégies d'adaptation.

Évolutivité : la spécification des mécanismes d'adaptation doit être suffisamment souple pour pouvoir ajouter ou retirer facilement des adaptations possibles ou des stratégies d'adaptation.

Dynamicité : la dynamicité de ces spécialisations est essentielle pour (i) ne pas avoir à arrêter le système pour le faire évoluer, mais surtout (ii) parce que les choix d'adaptations à un instant donné ne sont plus forcément pertinent à un autre instant (une fonctionnalité que l'on souhaitait adaptable à un instant ne doit plus l'être à un autre ; une fonctionnalité que l'on souhait adapter d'une certaine manière à un certain instant ne doit plus l'être de la même manière à un autre instant).

Efficacité :

Stabilité : une variation dans l'environnement induit souvent plus d'un seul changement, se traduisant par l'application de plusieurs adaptations. Dans ce cas d'adaptations multiples, deux hypothèses sont possibles : (i) le système se trouve dans un état e qui déclenche n adaptations sur une fonctionnalité et amène le système dans l'état e', (ii) le système se trouve dans un état e qui déclenche la $i^{ème}$ adaptation sur une fonctionnalité et amène le système dans l'état e', cet état déclenche la $i+1^{ème}$ adaptation sur une autre fonctionnalité et amène le système dans l'état e'', etc. Le premier cas correspond à des problèmes de synchronisation d'adaptations. Il sera traité dans ce chapitre. Le second cas correspond aux effets « boomerang » et « domino » et sera traité dans le chapitre 6.

5.2 Modèle du système d'adaptation et de réaction

La définition du modèle de notre système d'adaptation nécessite, en premier lieu, d'identifier les acteurs auxquels va s'appliquer l'adaptation. La figure 5.1 présente les différents éléments du système. Les terminaux mobiles et les stations fixes exécutent différentes *entités communicantes*. Ces entités appartiennent à notre système d'adaptation ou aux applications.

Dans le premier cas, elles correspondent à des fonctionnalités ayant été spécialisées et instanciées. Dans la suite du document, nous les appellerons *services*. Par exemple, le point (i) de la figure 5.1 représente ainsi le service de gestion des communications ; le point (ii) montre le service de détection et notification en train de surveiller les ressources d'une station. Ces entités sont reliées par des dépendances de communication (point iii) ou des dépendances de fonction (point iv). Un dépendance de fonction peut, par exemple, s'illustrer par un service de décompression qui dépendrait du service de compression bien qu'il n'y ait pas de communications directes entre eux, les données pouvant être échangées de manière sous-jacente par un système de fichier ou un SGBD.

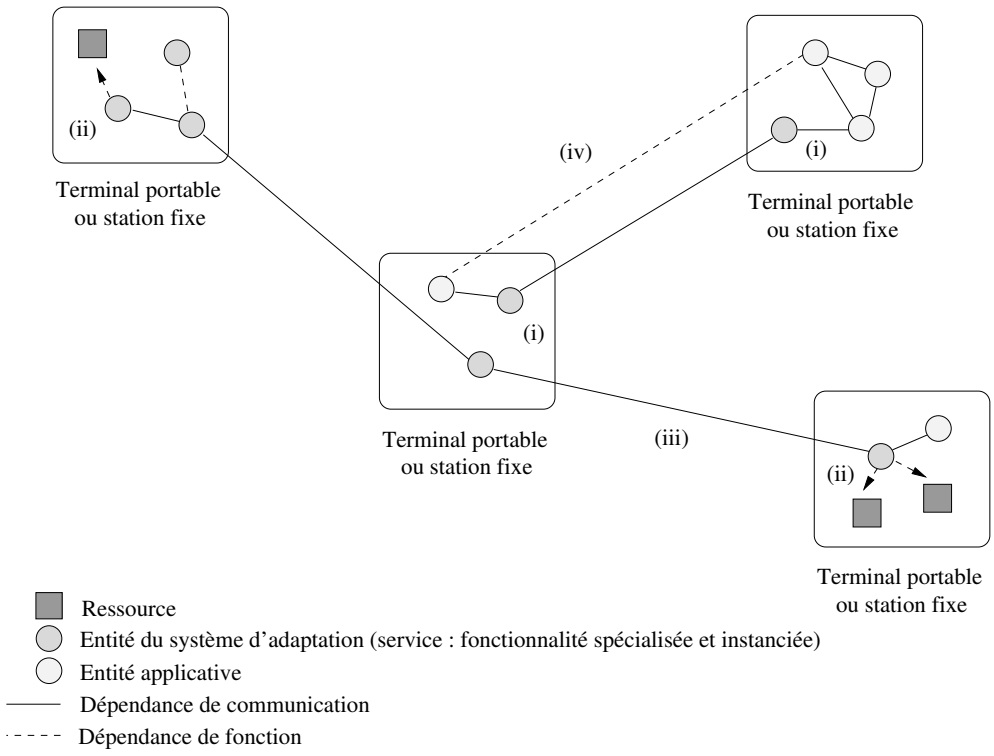


FIG. 5.1 – Acteurs du système d'adaptation et de réaction

5.2.1 Définition d'une entité

Une entité est l'unité logicielle de conception de notre système d'adaptation. Celle-ci est abstraite et spécialisable, ce qui lui permet de s'abstraire des modèles de conception distribués (nous reviendrons sur les interactions de notre modèle et des modèles de conception distribués dans le paragraphe 5.3). Chaque fonctionnalité de notre cadre de conception correspond à une entité. Chaque service de notre système correspond donc à une entité ayant été spécialisée et instanciée.

Comme le montre la figure 5.2, la structure d'une entité est partagée selon trois aspects :

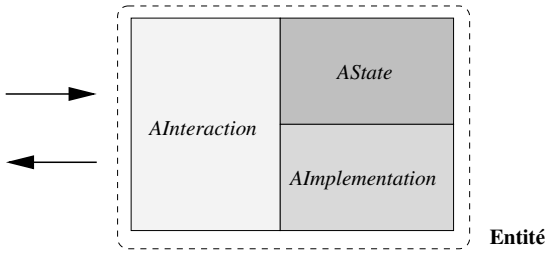


FIG. 5.2 – Structure d'une entité

L'aspect "communication" : la partie communication (type *AInteraction*) se charge des interactions avec les autres entités.

L'aspect "métier" : la partie métier (type *AImplementation*) effectue les traitements relatifs à la fonction attendue de l'entité.

L'aspect "état" : la partie état (type *AState*) conserve l'état courant interne de l'entité. Bien que les parties communication et métier possèdent aussi un état interne, l'état d'une entité porte uniquement sur les informations à conserver de la partie métier. Pour illustrer la différence entre état interne des parties et état de l'entité, la modification des paramètres d'une connexion entre entités ne doit pas, par exemple, se faire sur les états internes des parties de communication mais par une paramétrisation du service de communication utilisé.

La figure 5.3 présente les relations entre ces trois aspects ainsi que leurs possibilités de spécialisation :

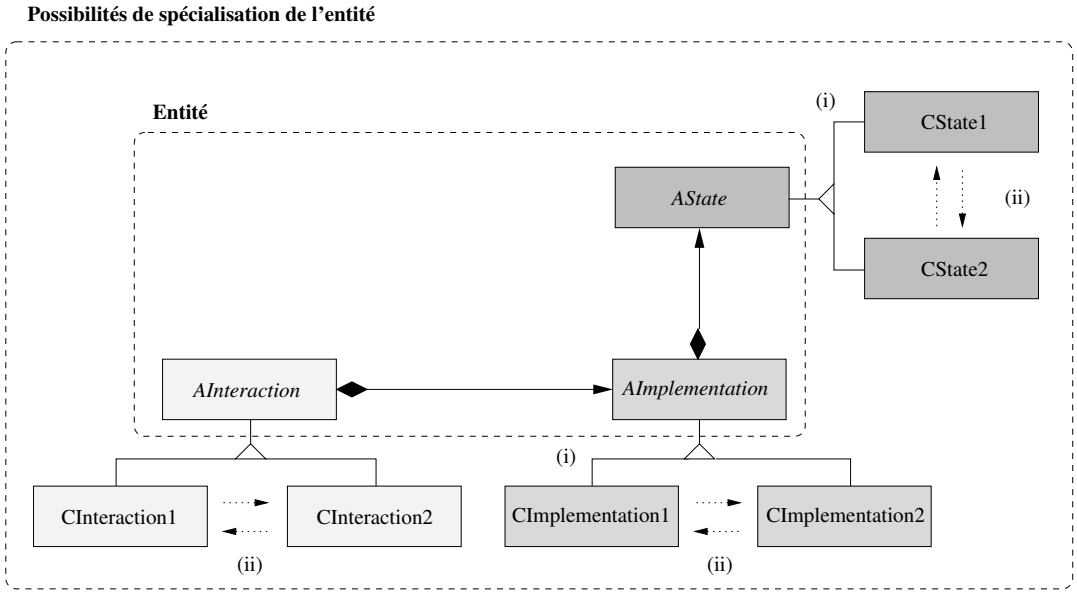


FIG. 5.3 – Relations entre éléments de la structure d'une entité et spécialisations possibles de ces éléments

Le type *AInteraction* maintient une relation de composition avec le type *AImplementation* (au sens UML (*Unified Modeling Language*) du terme composition [OMG 2003a]). Le type *AImplementation* maintient également une relation de composition avec le type *AState*. Ces relations de composition signifient qu'il y a une agrégation forte entre ces types impliquant (i) qu'il ne peut y avoir, à un instant donné, qu'une seule instance du type *AImplementation* lié avec une instance du type *AInteraction* (idem entre *AState* et *AImplementation*), (ii) que leurs cycles de vie sont liées ; la destruction d'une instance du type *AInteraction* entraînerait la destruction de l'instance de type *AImplementation* qui entraînerait la destruction de l'instance de type *AState*.

Les spécialisations suivent les patrons de conception *Strategy* (point i de la figure 5.3) et *State* (point ii de la figure 5.3) [Gamma et al. 1995]. Ainsi, le type abstrait *AInteraction* peut être spécialisé par héritage en types concrets CInteraction1 ou CInteraction2 (de même pour les types *AImplementation* et *AState* respectivement en CImplementation1 ou CImplementation2 et CState1 ou CState2). La spécialisation selon le patron de conception *State* ne peut être réalisée que lorsqu'une spécialisation par héritage a déjà eu lieu. Elle permet à un type concret de permuter avec un autre type concret héritant du même type abstrait. Lorsque cette spécialisation s'effectue dynamiquement entre instances, cela entraîne alors la destruction de l'instance initiatrice puisque, par composition, il ne peut y avoir qu'une instance liée à un instant donné.

5.2.2 Définition d'une entité adaptative

Une entité adaptative est l'unité logicielle dont le comportement peut être modifié par l'application d'adaptations. Par définition, une entité adaptative est donc une entité à laquelle est rajoutée la description des adaptations possibles sur celle-ci. La figure 5.4 présente la structure d'une entité adaptative :

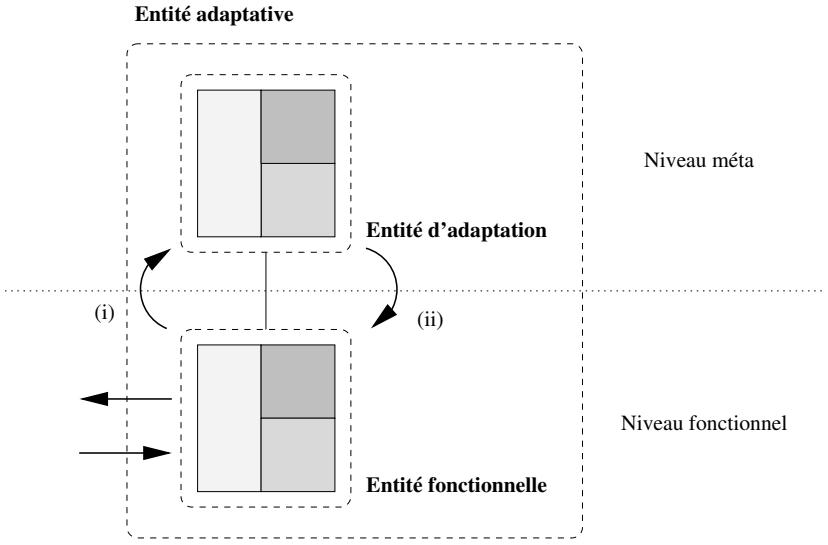


FIG. 5.4 – Structure d'une entité adaptative

Au niveau fonctionnel, une entité adaptative comprend une *entité fonctionnelle* qui effectue les traitements attendus de l'entité adaptative. Au niveau méta, la description des possibilités d'adaptations est matérialisée par une *entité d'adaptation* liée à l'entité fonctionnelle. Nous avons choisi de modéliser la description des adaptations par une entité pour obtenir une dynamicité maximale. La description des adaptations peut, en effet, être dynamiquement modifiée par (i) la permutation de l'entité d'adaptation avec une autre entité d'adaptation (en modifiant le lien entre entité fonctionnelle et entité d'adaptation), ou (ii) la mise en œuvre de l'entité d'adaptation par une entité elle-même adaptative. Ce dernier cas permet aux concepteurs de créer des entités adaptatives possédant plusieurs catégories d'adaptations. Une entité adaptative peut alors, par exemple, avoir les adaptations $\{a_1, a_2, a_3\}$ applicables dans la situation S_1 et permuter pour les adaptations $\{a_4, a_5\}$ dans la situation S_2.

Les deux paragraphes suivants présentent les interactions entre le niveau fonctionnel et le niveau méta. Le paragraphe 5.2.2.1 détaille le parcours des demandes d'adaptations (étape i de la figure 5.4) et le paragraphe 5.2.2.2 explique la manière d'application de celles-ci (étape ii de la figure 5.4).

5.2.2.1 Demandes d'adaptations du niveau fonctionnel vers le niveau méta

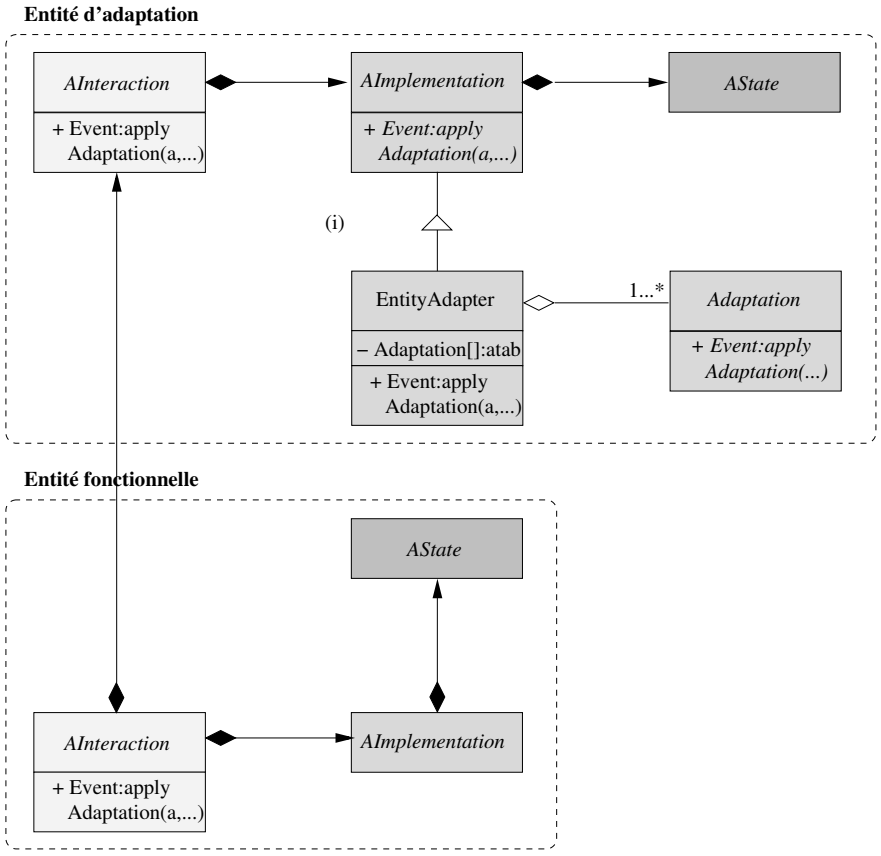


FIG. 5.5 – Relations entre éléments de la structure d'une entité adaptative

Comme le montre la figure 5.5, l'entité d'adaptation est liée à l'entité fonctionnelle par une relation de composition entre le type *AInteraction* de l'entité fonctionnelle et le type *AInteraction* de l'entité d'adaptation. Comme pour la structure interne d'une entité, cela signifie qu'il ne peut y avoir qu'une seule entité d'adaptation liée à l'entité fonctionnelle et que la destruction d'une instance de l'entité fonctionnelle entraîne la destruction de l'instance d'entité d'adaptation liée.

La spécialisation de l'entité d'adaptation suit le patron de conception *Adapter* (point i de la figure 5.5) [Gamma et al. 1995]. Celui-ci a été modifié pour permettre, non plus l'interfaçage entre un client et un adapté, mais entre l'entité fonctionnelle et les *n* adaptations de l'entité d'adaptation. Le type *AInteraction* de l'entité fonctionnelle fournit la méthode `Event:applyAdaptation(a,args[])` chargée d'appliquer l'adaptation `a` à l'entité fonctionnelle (la réception de l'évènement `Event` confirme sa réalisation ou son échec). Celle-ci est implantée par une redirection de l'appel vers la même méthode dans le type *AInteraction* de l'entité d'adaptation :

```
abstract class AInteraction { // Interaction de l'entité fonctionnelle

   protected AInteraction adaptationEntityInteraction; // Lien vers l'entité d'adaptation
   ...
   public Event applyAdaptation(String adaptationName, String[] args) {
      return(adaptationEntityInteraction.applyAdaptation(adaptationName, args));
   }
   ...
}
```

FIG. 5.6 – Redirection de l'appel `applyAdaptation(a,args[])` de l'entité fonctionnelle vers l'entité d'adaptation

La méthode du type *AInteraction* de l'entité d'adaptation redirige également l'appel vers la méthode identique dans le type *AImplementation*. Le type EntityAdapter spécialise le type *AImplementation*. Il répertorie toutes les adaptations applicables à l'entité fonctionnelle et fournit une implantation de la méthode `Event:applyAdaptation(a,args[])` permettant le routage de l'appel vers la bonne adaptation :

```
class EntityAdapter { // Adaptateur routant les appels vers la bonne Adaptation

   protected Adaptation[] atab; // Références vers les adaptations applicables
   ...
   public Event applyAdaptation(String adaptationName, String[] args) {
      return(atab[adaptationName].applyAdaptation(args));
   }
   ...
}
```

FIG. 5.7 – Routage de l'appel `applyAdaptation(a,args[])` dans le type EntityAdapter vers l'*Adaptation* adéquate

Les méthodes[1] d'accès, d'ajout et de retrait d'adaptations (`get|add|removeAdaptation`) sont conçues de la même manière que la méthode d'application de l'adaptation. Le transit des appels suit le même chemin du niveau fonctionnel jusqu'au niveau méta. La seul différence réside dans le type EntityAdapter qui met alors à jour ses références vers les adaptations.

5.2.2.2 Application des adaptations du niveau méta au niveau fonctionnel

Avant d'appliquer des adaptations, il est nécessaire de caractériser les adaptations qu'il est possible d'appliquer à l'entité fonctionnelle. La figure 5.8 répertorie les trois catégories d'adaptations que nous avons identifiées :

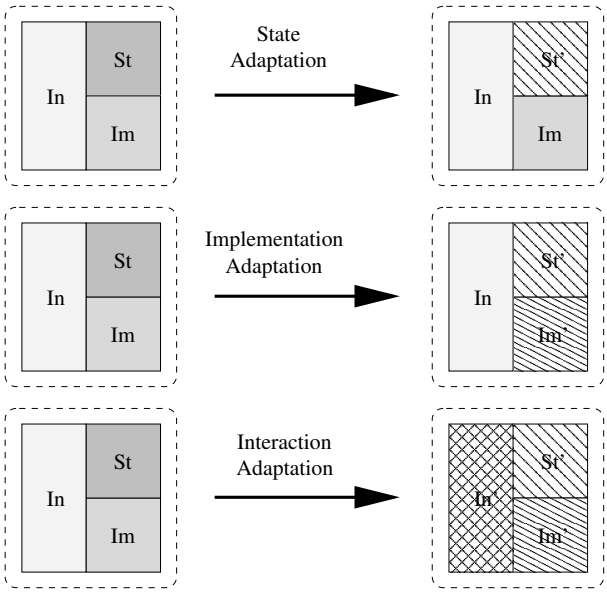


FIG. 5.8 – Possibilités d'adaptation d'une entité

La première forme consiste à modifier l'état de l'entité fonctionnelle. Cette modification peut être sans conservation d'état (St' ne dépend pas de St) ou avec transformation d'état (St' $= Tr_{AState}$(St)). La deuxième forme consiste à modifier l'implantation de l'entité fonctionnelle. De la même manière que pour l'état, la modification de l'implantation peut se faire sans conservation de son état interne (Im' ne dépend pas de Im) ou avec transformation de l'état interne (Im' $= Tr_{AImplementation}$(Im)). Le type *AImplementation* étant lié par composition au type *AState*, cette modification de l'implantation est suivie d'une modification de l'état de l'entité (même si l'état reste le même, celui-ci se présentera sous une nouvelle instance). La troisième forme consiste à modifier l'interaction de l'entité fonctionnelle. Cette modification se présente de la même manière que la précédente : (i) sans conservation de son état interne (In' ne dépend pas de In) ou avec transformation de l'état interne (In' $= Tr_{AInteraction}$(In)), et (ii) suivie des modifications de l'implantation et de l'état.

[1] Par manque de place, nous n'avons pas fait apparaître toutes les méthodes sur la figure 5.5.

L'avantage d'appliquer une transformation lors d'une adaptation est de pouvoir conserver une cohérence interne et externe de l'entité. La cohérence interne est conservée en récupérant les informations pertinentes d'exécution, par exemple, les données à sauvegarder dans le type *AState*, l'état du flot d'exécution dans le type *AImplementation* ou l'état des messages reçus et acquittés dans le type *AInteraction*. La cohérence externe est conservée en informant des changements les entités dépendantes. Par exemple, lors du changement d'interaction, les entités ayant des dépendances de communication peuvent être informées de manière à s'adresser à la nouvelle instance de la partie communication. Ou encore, lors du changement d'implantation, les entités ayant des dépendances de fonction peuvent être informées de manière à accorder leur partie métier.

La figure 5.9 présente ces trois formes d'adaptations ainsi que leurs relations avec les différents éléments de l'entité fonctionnelle :

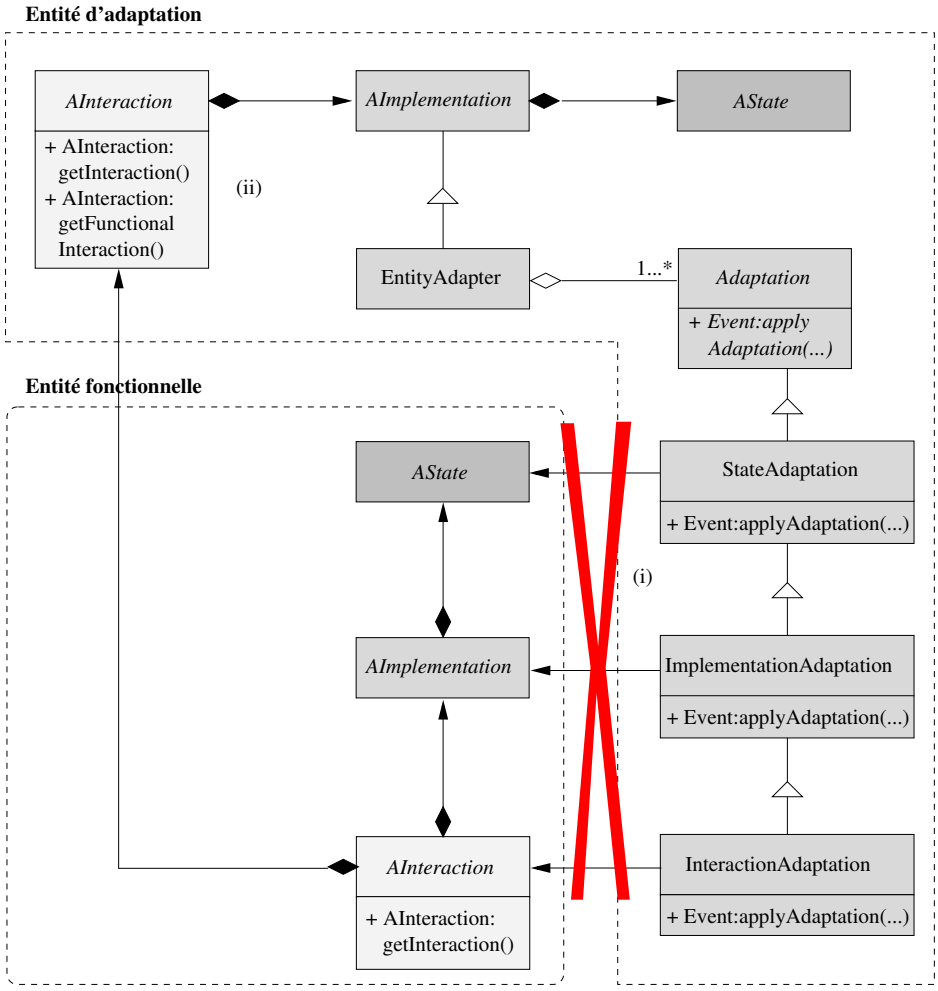


FIG. 5.9 – Relations entre adaptations et éléments de la structure de l'entité fonctionnelle

Chacune de ces formes est mise en œuvre par un type spécialisant le type *Adaptation*. StateAdaptation spécialise directement *Adaptation*, ImplementationAdaptation spécialise StateAdaptation et InteractionAdaptation spécialise ImplementationAdaptation.

Ces adaptations ont besoin d'accéder aux différents éléments de l'entité fonctionnelle (i) pour en récupérer les instances et les états internes et effectuer les transformations, et (ii) pour appliquer les nouvelles instances et les nouveaux états. Deux solutions s'offraient pour ces accès.

La première solution (point i de la figure 5.9) consistait à maintenir un lien de dépendance entre chaque adaptation et l'élément concerné par l'adaptation. Cette solution présentait l'inconvénient majeur de rendre l'entité d'adaptation totalement dépendante de l'entité fonctionnelle. Ainsi, les liens entre toutes les adaptations et les éléments à adapter auraient du être actualisés lors de la permutation d'entités d'adaptation, de l'ajout d'une adaptation et de l'application d'une adaptation. De fait, cette solution n'a pas été retenue.

La seconde solution, que nous avons choisie (point ii de la figure 5.9), consiste à définir des propriétés d'introspection et d'intercession pour une entité et pour une entité adaptative. Ainsi chaque type *AInteraction*, *AImplementation*, *AState* d'une entité comporte les méthodes suivantes[2] :

```
public AInteraction:getInteraction(); // retourne l'interaction de l'entité courante

public AImplementation:getImplementation(); // retourne l'implantation de l'entité courante

public AState:getState(); // retourne l'état de l'entité courante

public Event:setInteraction(AInteraction in); // positionne l'interaction de l'entité
                                              // courante à in; retourne Event lorsque
                                              // cela est fait

public Event:setImplementation(AImplementation im); // positionne l'implantation de
                                                    // l'entité courante à im; retourne
                                                    // Event lorsque cela est fait

public Event:setState(AState st); // positionne l'état de l'entité courante à st; retourne
                                  // Event lorsque cela est fait
```

FIG. 5.10 – Méthodes d'introspection et d'intercession d'une entité

Néanmoins, ces méthodes ne suffisent pas puisque, si on les utilise dans l'entité d'adaptation, elles accèdent aux instances des types *AInteraction*, *AImplementation*, *AState* de l'entité d'adaptation et non aux instances des types *AInteraction*, *AImplementation*, *AState* de l'entité fonctionnelle. Nous avons donc également défini des méthodes supplémentaires[2] présentes dans chaque type *AInteraction*, *AImplementation*, *AState* de l'entité d'adaptation :

[2]Par manque de place, nous n'avons pas fait apparaître toutes les méthodes sur la figure 5.9.

```
abstract class AInteraction { // Interaction de l'entité d'adaptation

   protected AInteraction functionalEntityInteraction; // Lien vers l'entité fonctionnelle
   ...
   public AInteraction:getFunctionalInteraction() { // retourne l'interaction de
                                                    // l'entité fonctionnelle
      return(functionalEntityInteraction.getInteraction());
   }
   ...
   // Idem pour getFunctionalImplementation() et getFunctionalState()
   ...
   public Event:setFunctionalInteraction(AInteraction in) { // positionne l'interaction de
                                                            // l'entité fonctionnelle à in;
                                                            // retourne Event lorsque cela
                                                            // est fait.
      return(functionalEntityInteraction.setInteraction(in));
   }
   ...
   // Idem pour setFunctionalImplementation(AImplementation) et setFunctionalState(AState)
   ...
}
```

FIG. 5.11 – Méthodes d'introspection et d'intercession de l'entité adaptative (illustrées par le type *AInteraction* de l'entité d'adaptation)

Comme dans le paragraphe précédent pour la redirection des demandes d'adaptations, les méthodes d'accès `set|getFunctional*` redirigent les appels vers les méthodes d'introspection et d'intercession de l'entité fonctionnelle. Les types *Adaptation* peuvent ensuite utiliser ces méthodes pour définir la méthode d'application de l'adaptation (`Event:applyAdaptation(args[])`) et celle de transformation de l'élément concerné de l'entité (par exemple, `Aimplementation:implementationTransformation(im, st, args[])`). La figure 5.12 l'illustre ci-dessous avec une adaptation d'implantation :

```
class ImplementationAdaptation extends StateAdaptation { // Adaptation d'implantation

   // héritage de la déclaration de variable
   // protected EntityAdapter entityAdapter;
   ...
   // héritage de la déclaration de méthode
   // public AState stateTransformation(AState st, String[] args);
   ...
   public Event applyAdaptation(String[] args) { // méthode d'application de l'adaptation

      AState cfs; // current functional state
      AState tfs; // transformed functional state
      AImplementation cfi; // current functional implementation
      AImplementation tfi; // transformed functional implementation
      ...
```

```
      // récupération des éléments de la structure de l'entité fonctionnelle
      cfs = entityAdapter.getFunctionalState();
      cfi = entityAdapter.getFunctionalImplementation();
      ...
      // transformations de ces éléments
      tfs = stateTransformation(cfs, args);
      tfi = implementationTransformation(cfi, tfs, args); // on reprend l'état transformé
      ...
      return(entityAdapter.setFunctionalImplementation(tfi));
   }
   ...
   public AImplementation implementationTransformation(AImplementation im,
                                                       AState st,
                                                       String[] args) { // transformation
                                                                        // d'une implantation
      AImplementation tim; // transformed implementation
      ...
      // Création d'une nouvelle instance d'implantation
      tim = new Implementation(im, st);
      // Ici, modifications de l'implantation tim
      ...
      // On prévient, par exemple, les entités ayant des dépendances fonctionnelles
      ...
      return(tim);
   }
}
```

FIG. 5.12 – Utilisation des méthodes d'introspection et d'intercession pour appliquer une adaptation d'implantation

On peut noter dans la figure 5.12 que l'on peut interagir, dans la procédure de transformation, avec les entités présentant des dépendances de communication ou de fonction. Ces notifications d'adaptations sont données à titre informatif aux entités dépendantes. Les entités réceptrices des notifications d'adaptations peuvent, en effet, très bien choisir (i) de modifier leurs comportements pour être en conformité avec les adaptations réalisées, ou (ii) d'entamer une procédure de négociation pour trouver un compromis d'adaptation satisfaisant les deux parties, ou encore (iii) de rompre les dépendances n'étant pas satisfaites ou ne pouvant s'adapter au nouveau comportement.

Cette solution de définition de méthodes d'introspection et d'intercession procure une grande indépendance et une grande généricité à nos entités adaptatives. L'indépendance est garantie par le fait que le seul lien qui subsiste entre entité fonctionnelle et entité d'adaptation est le lien de composition entre les interactions. La généricité est procurée par les faits que (i) les méthodes d'introspection et d'intercession s'appliquent à toutes les entités et à toutes parties de leur structure, (ii) les adaptations ne sont pas conçues en fonction d'une entité fonctionnelle particulière mais sont applicables à toute entité fonctionnelle courante à un instant t.

5.2.3 Entité adaptative et réaction

Dans les paragraphes précédents, une entité adaptative peut changer de comportement "si on le lui dit". La notion de décision de l'adaptation n'y est pas incluse et est donc laissée à la charge d'acteurs extérieurs comme le concepteur, l'administrateur ou l'application. Une des attentes de notre système est de pouvoir réagir à différents évènements comme, notamment, les variations observables dans l'environnement mobile. La décision d'adaptation doit donc pouvoir être prise à l'intérieur d'une entité. Cela nous amène à la définition d'une entité auto-adaptative.

Une entité auto-adaptative est l'unité logicielle pouvant décider de son changement de comportement par l'application d'adaptations. Par définition, une entité auto-adaptative est une entité adaptative à laquelle est rajoutée la description des décisions d'adaptations que celle-ci peut prendre. La figure 5.13 présente la structure d'une entité auto-adaptative :

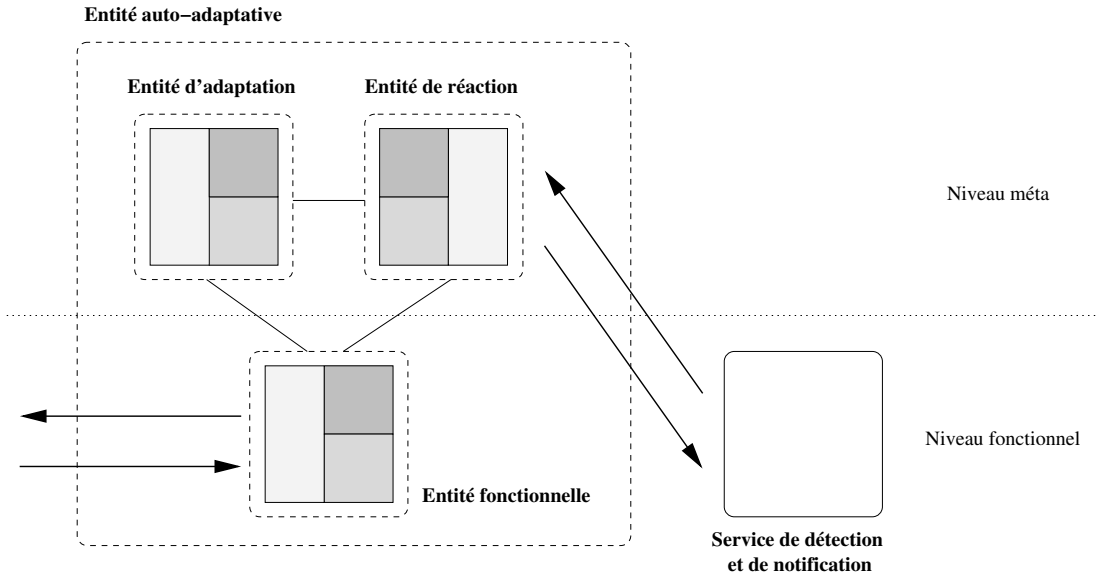


FIG. 5.13 – Structure d'une entité auto-adaptative

Au niveau fonctionnel, une entité auto-adaptative comporte toujours l'entité fonctionnelle qui effectue les traitements attendus de l'entité auto-adaptative. Au niveau méta, la description des possibilités d'adaptations est toujours à la charge de l'entité d'adaptation. La nouveauté réside au niveau méta dans l'*entité de réaction* qui matérialise la description des décisions possibles d'adaptations. Nous avons choisi de modéliser l'entité de réaction par une entité pour les mêmes raisons de dynamicité évoquées dans le paragraphe 5.2.2. L'entité de réaction est liée à l'entité fonctionnelle puisque l'entité fonctionnelle peut vouloir changer à tout moment les décisions à prendre. Elle est également liée à l'entité d'adaptation puisque, sans adaptations à appliquer, les décisions d'adaptations seraient sans objet. De manière à prendre ses décisions, l'entité de réaction interagit avec le service de détection et notification afin d'être informée des variations de l'environnement mobile.

Au sein de l'entité de réaction, les décisions d'adaptations sont maintenues dans une *stratégie d'adaptation*. Celle-ci est conçue comme un automate dont les états représentent l'état d'exécution courant de l'entité fonctionnelle et les transitions représentent les conditions sur les variations de l'environnement mobile et la décision de l'adaptation à appliquer dans ce cas. La figure 5.14 donne l'exemple d'une stratégie d'adaptation pour un service de transmission d'images compressées :

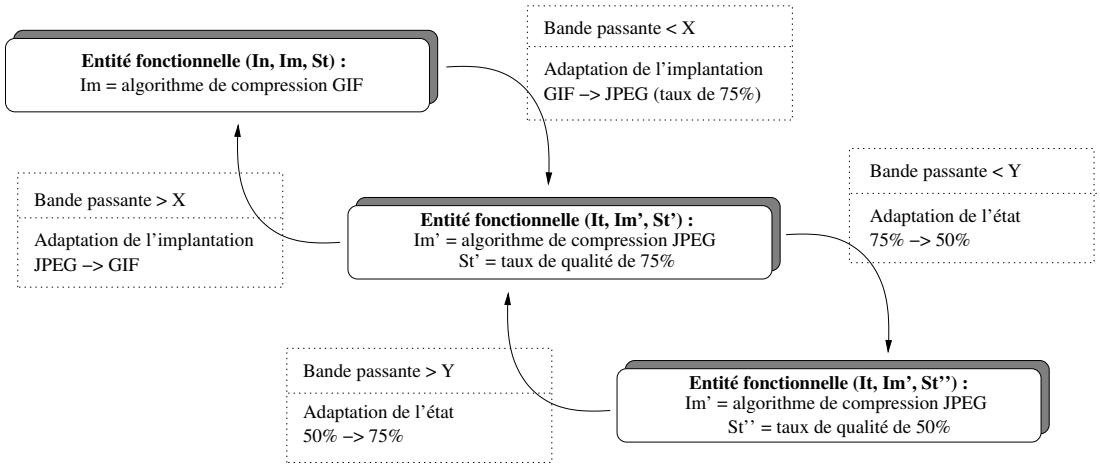


FIG. 5.14 – Stratégie d'adaptation pour un service de transmission d'images compressées

Dans cet exemple, si la bande passante se trouve être supérieure à une valeur X, l'implantation de l'entité fonctionnelle est un algorithme de compression au format GIF. Si la bande passante diminue mais est toujours supérieure à une valeur Y, une adaptation de changement d'implantation est appliquée permutant ainsi l'implantation de l'entité fonctionnelle vers un algorithme de type JPEG. Une adaptation de changement d'état[3] est également appliquée fixant le taux de qualité à 75%. Si la bande passante se trouve être inférieure à la valeur Y, une adaptation de changement d'état est appliquée fixant le taux de qualité à 50%. Dans cette stratégie d'adaptation, les transitions inverses appliquent les adaptations inverses.

La figure 5.15 présente la structure d'une entité adaptative, et, notamment, la manière dont est implantée la stratégie d'adaptation.

L'entité de réaction est liée à l'entité fonctionnelle et à l'entité d'adaptation par des relations de composition entre les types *AInteraction*. Comme pour l'entité adaptative, cela signifie qu'il ne peut y avoir qu'une seule entité de réaction liée à l'entité fonctionnelle. La destruction d'une instance de l'entité fonctionnelle entraîne la destruction de l'instance d'entité de réaction liée. Mais, aussi, la destruction d'une instance de l'entité d'adaptation entraîne la destruction de l'instance d'entité de réaction liée.

La spécialisation de l'entité de réaction suit le patron de conception *Adapter* (point i de la figure 5.15) [Gamma et al. 1995]. Celui-ci permet l'interfaçage entre l'entité fonctionnelle et la stratégie d'adaptation de l'entité d'adaptation. La stratégie d'adaptation de

[3] L'adaptation de changement d'état n'est pas notée dans l'action à effectuer lors de la transition puisqu'elle est implicite du fait qu'une adaptation d'implantation implique une adaptation d'état.

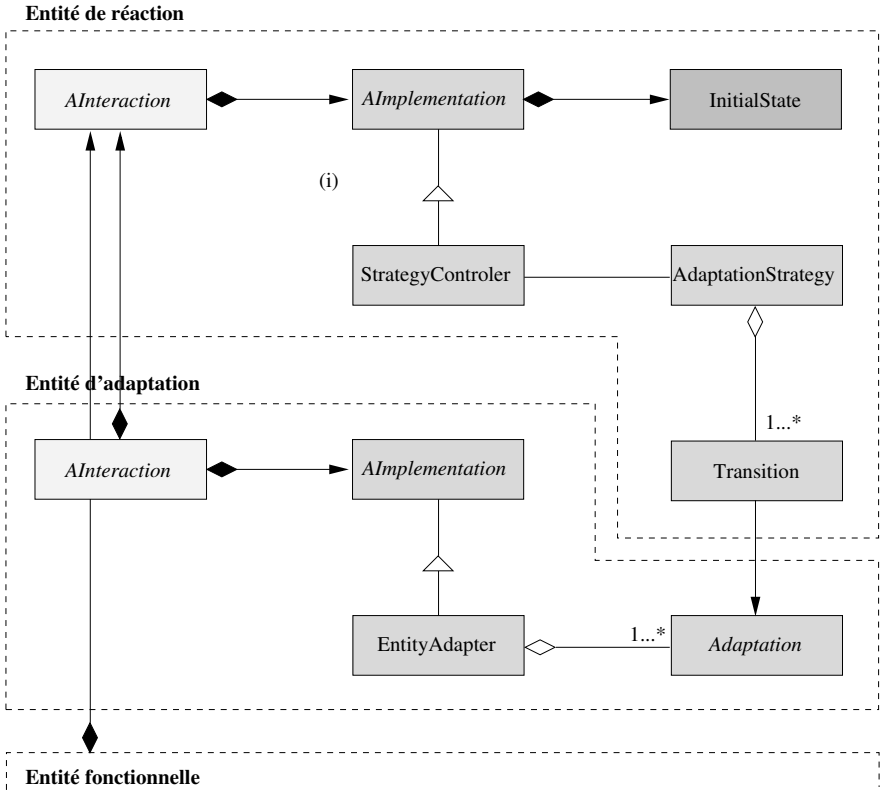


FIG. 5.15 – Relations entre éléments de la structure d'une entité auto-adaptative

type AdaptationStrategy est composée d'un ensemble de transitions de type Transition. Une Transition maintient une dépendance directe avec l'Adaptation qu'elle doit appliquer. Nous avons ici fait le choix de maintenir des dépendances directes car l'entité de réaction ne peut intrinsèquement pas être indépendante de l'entité d'adaptation. En effet, toutes modifications dans les adaptations possibles doivent se répercuter sur la stratégie d'adaptation. Par exemple, le retrait d'une adaptation référencée dans une transition conduit à l'invalidation de cette transition.

Les méthodes[4] d'accès à la stratégie d'adaptation (`get|setAdaptationStrategy`) ainsi que les méthodes[4] d'accès, d'ajout et de retrait de transitions (`get|add|removeTransition`) sont conçues de la même manière que la méthode d'application de l'adaptation (dans le paragraphe 5.2.2.1). Le transit des appels suit le même chemin du niveau fonctionnel jusqu'au niveau méta, mais en passant par l'entité de réaction. Et cette fois-ci, c'est le type StrategyControler qui met à jour sa référence vers la stratégie d'adaptation et le type AdaptationStrategy qui met à jour ses références vers les transitions.

[4]Par manque de place, nous n'avons pas fait apparaître toutes les méthodes sur la figure 5.15.

5.2.4 Entité adaptative et synchronisation d'adaptations multiples

Dans le paragraphe précédent, les entités auto-adaptatives réagissent aux variations de l'environnement. Seulement, une variation dans l'environnement induit souvent plus d'un seul changement, se traduisant par l'application de plusieurs adaptations. Dans ce cas d'adaptations multiples, deux hypothèses sont possibles :

(i) Les conditions de l'environnement mobile se trouvent dans un état c qui déclenche la $i^{ème}$ adaptation a_i sur l'entité e et amène les conditions de l'environnement mobile dans un état c'. Cet état c' déclenche la $i+1^{ème}$ adaptation a_{i+1} sur l'entité e' et amène les conditions de l'environnement mobile dans un état c'', etc $\left(\forall i \quad \{c_i\} \quad a_{i,\, c_i \rightarrow c_{i+1}}(e_i) \quad \{c_{i+1}\}\right)$. Ce cas correspond aux effets « boomerang » et « domino » et sera traité dans le chapitre 6.

(ii) Les conditions de l'environnement mobile se trouvent dans un état c qui déclenche n adaptations a sur l'entité e et amène les conditions de l'environnement mobile dans un état c' $\left(\{c\} \quad \forall i \quad a_{i,\, c \rightarrow c'}(e) \quad \{c'\}\right)$. Ce cas correspond à des problèmes de synchronisation d'adaptations et peut être réglé par la mise en place d'adaptations particulières.

La figure 5.16 présente la manière dont nous avons mis en place les adaptations multiples. Une adaptation multiple est définie comme un type *MultipleAdaptations* spécialisant le type *Adaptation*. Il contient les références des différentes adaptations à appliquer (`Adaptation[] atab`). Il définit également les méthodes[5] de manipulation de l'ensemble des adaptations (`set|get|add|removeAdaptations`). Par contre, il ne définit pas la méthode d'application des adaptations `Event:applyAdaptation(args[])`.

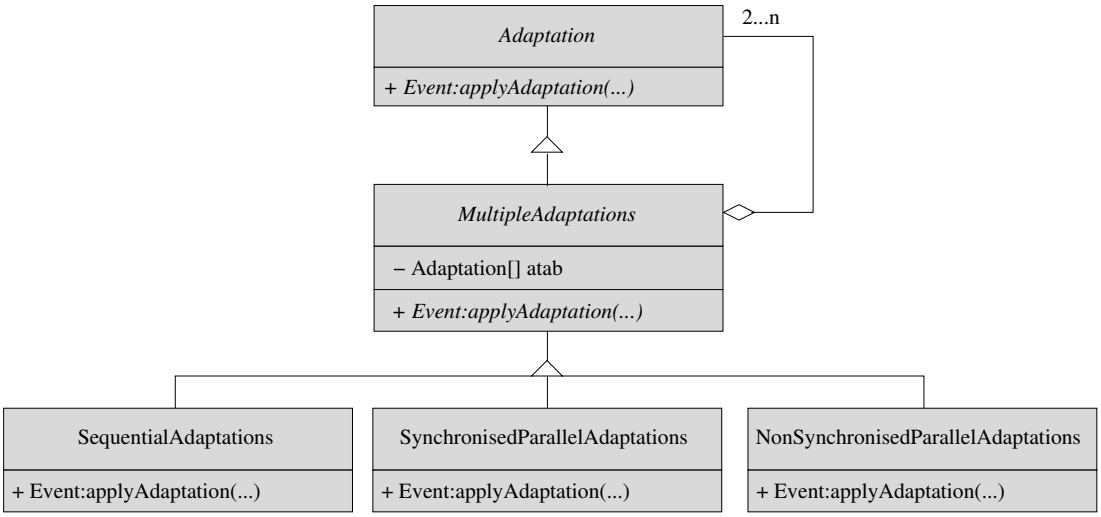


FIG. 5.16 – Synchronisation des adaptations multiples

Un des avantage de nos *MultipleAdaptations* est qu'elles sont elles-mêmes des *Adaptation*. À partir de quelques définitions d'adaptations multiples basiques, il est ainsi pos-

[5] Par manque de place, nous n'avons pas fait apparaître toutes les méthodes sur la figure 5.16.

sible de les combiner pour définir des séquences complexes de synchronisation. Les types SequentialAdaptations, SynchronisedParallelAdaptations et NonSynchronisedParallelAdaptations spécialisent le type *MultipleAdaptations* et définissent dans la méthode `Event:applyAdaptation(args[])` des manières basiques d'application de l'ensemble des adaptations. Les figures 5.17 et 5.18 présentent, par exemple, respectivement une manière séquentielle et une manière parallèle synchronisée d'appliquer l'ensemble des adaptations.

```
class SequentialAdaptations extends MultipleAdaptations { // Adaptations séquentielles

   // héritage de la déclaration de variable
   // protected Adaptation[] atab;
   ...
   public Event applyAdaptation(String[] args) { // méthode d'application séquencielle
                                                 // de l'ensemble ordonné des adaptations
      int i = 0; // compteur pour la séquentialité
      Event currentAdaptationResult; // résultat de l'adaptation courante
      ...
      repeat
         currentAdaptationResult = atab[i].applyAdaptation(args); // ième adaptation
         i++;
      until ( isOK(currentAdaptationResult) and i<=atab.length);
      ...
      return(currentAdaptationResult);
   }
}
```

FIG. 5.17 – Adaptations appliquées de manière séquentielle

```
class SynchronisedParallelAdaptations extends MultipleAdaptations { // Adaptations parallèles
                                                                    // et synchronisées
   // héritage de la déclaration de variable
   // protected Adaptation[] atab;
   ...
   public Event applyAdaptation(String[] args) { // méthode d'application de l'ensemble
                                                 // des adaptations
      Event[] currentAdaptationResults; // ensemble des résultats des adaptations
      Event   currentAdaptationResult;  // résultat final
      ...
      forall i in atab do
         currentAdaptationResults[i] = atab[i].applyAdaptation(args);
      done
      // on ne sort de la boucle parallèle que lorsque l'on a tous les résultats d'adaptations
      ...
      // on construit un évènement relatant le succès des adaptations ayant réussies
      currentAdaptationResult = new Event(all i which isOK(currentAdaptationResults[i]));
      ...
      return(currentAdaptationResult);
   }
}
```

FIG. 5.18 – Adaptations appliquées de manière parallèle synchronisée

L'évènement résultat fournit par l'application d'une adaptation multiple nécessite quelques explications. Dans le cas de l'application d'une unique adaptation, l'évènement résultat ne doit pas être interprété comme "j'ai réussi à appliquer l'adaptation ou non" mais comme "je suis dans un état cohérent résultant de l'adaptation, ou sinon je suis dans l'état cohérent précédent". Dans le cas d'adaptations multiples, l'évènement résultat ne doit pas être interprété comme "j'ai réussi à appliquer en entier tout l'ensemble des adaptations ou bien aucune adaptation" mais comme "je suis dans l'état cohérent résultant de l'application de tout l'ensemble des adaptations, ou sinon je suis dans un état cohérent résultant de l'application d'un sous-ensemble des adaptations". Ainsi dans la figure 5.17, l'évènement résultat retourne l'adaptation jusqu'à laquelle la séquence s'est correctement déroulée. Dans la figure 5.18, l'évènement résultat comprend les identifications des adaptations s'étant correctement déroulées. Cette manière d'implanter les évènements résultats est plus souple que la méthode du "tout ou rien" puisqu'elle offre à l'invocateur de l'adaptation multiple la possibilité de décider si cette adaptation multiple partielle lui convient ou non. De plus, cette manière d'implanter les évènements résultats peut toujours être utilisée pour implanter le "tout ou rien" si celui-ci est vraiment requis. Dans les figures 5.17 et 5.18, il est toujours possible soit (i) d'appliquer les adaptations dans le sens inverse (après vérification de la possibilité de retour arrière dans l'automate), soit (ii) d'appliquer les adaptations sur une entité copie qui ne sera validée que si toutes les adaptations se déroulent correctement.

5.3 Définition des modèles de conception par spécialisation d'entité

Comme le montre la figure 5.19, lorsque l'on examine un système, on s'aperçoit que celui-ci se compose d'un grand nombre d'entités. Cette constatation s'accentue en examinant les systèmes distribués et à grande échelle. Lorsque l'on examine un service d'un système, on s'aperçoit également que celui-ci peut ne pas se composer uniquement d'une entité mais en comporter de nombreuses (point i de la figure 5.19), et également que certaines entités peuvent ainsi être regroupées pour former un service (point ii).

La composition et la représentation d'une entité par un ensemble d'entités introduit la notion de *niveau d'abstraction*. Ainsi, dans la figure 5.19, l'entité E représente les entités E1, E2, E3 et E4. Tout changement de comportement dans les entités représentées E1, E2, E3 et E4 entraîne une modification de comportement de l'entité représentatrice E, et c'est pour cela que l'entité E doit maintenir des dépendances de niveau d'abstraction avec les entités représentées E1, E2, E3 et E4.

L'application d'adaptations dans ces conditions posent deux problèmes : (i) une adaptation donnée est-elle pertinente pour une entité, un ensemble d'entités comme un service ou un système ? (ii) à qui dois-je m'adresser pour appliquer cette adaptation ? Pour résoudre ces problèmes, nous proposons une échelle de représentation des abstractions de conception (paragraphe 5.3.1). Nous illustrons ensuite la propagation des adaptations au sein de l'échelle de niveaux (paragraphe 5.3.2), et donnons, enfin, quelques exemples concrets de spécialisation (paragraphe 5.3.3).

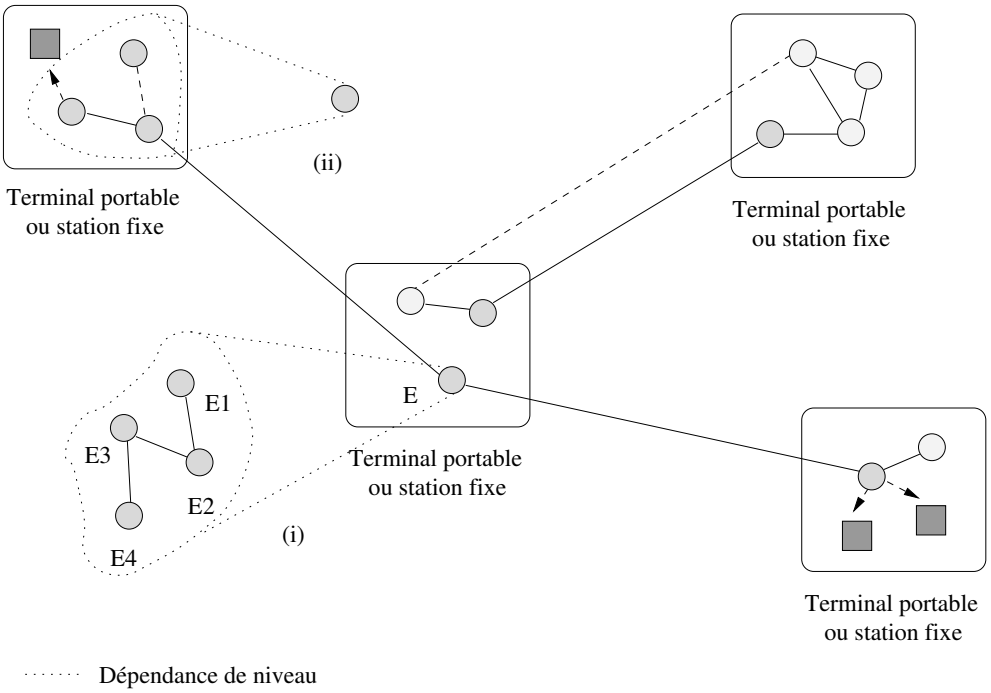


FIG. 5.19 – Regroupement des entités du système d'adaptation et de réaction

5.3.1 Hiérarchie de représentation des abstractions de conception

En reprenant le découpage des modèles de conception introduit dans le paragraphe 3.2.2, nous avons répertorié plusieurs niveaux d'abstractions de conception. Ceux-ci sont présentés dans la figure 5.20.

Ainsi, en partant du niveau le plus abstrait, un modèle représente une famille d'applications ; une application peut utiliser plusieurs intergiciels ; un intergiciel distribué peut être implanté de différentes manières (Client/Serveur, Agents, N-uplets) ; ceux-ci peuvent utiliser différents services (de communication, de stockage, etc) ; chaque service peut comprendre de nombreux composants qui peuvent eux-mêmes comporter plusieurs objets.

Notre système d'adaptation propose de représenter chaque abstraction de conception par une entité adaptative spécialisée. Nous avons utilisé, par extension, la notation d'héritage entre les niveaux d'abstractions et l'entité adaptative. Cette notation doit être comprise dans le sens de spécialisation et non d'héritage.

Une abstraction de conception maintient une relation d'utilisation avec les abstractions des niveaux inférieurs. Ces relations ne sont pas exclusives et peuvent court-circuiter des niveaux. Une application peut, par exemple, utiliser des composants et des objets sans passer par un intergiciel (point i de la figure 5.20). Notre échelle n'est évidemment pas exhaustive et de nombreuses abstractions peuvent y être ajoutées.

Cette représentation en niveaux d'abstractions apporte principalement deux avantages :

(i) Pour les différentes interactions avec une instance d'un certain niveau d'abstraction, l'in-

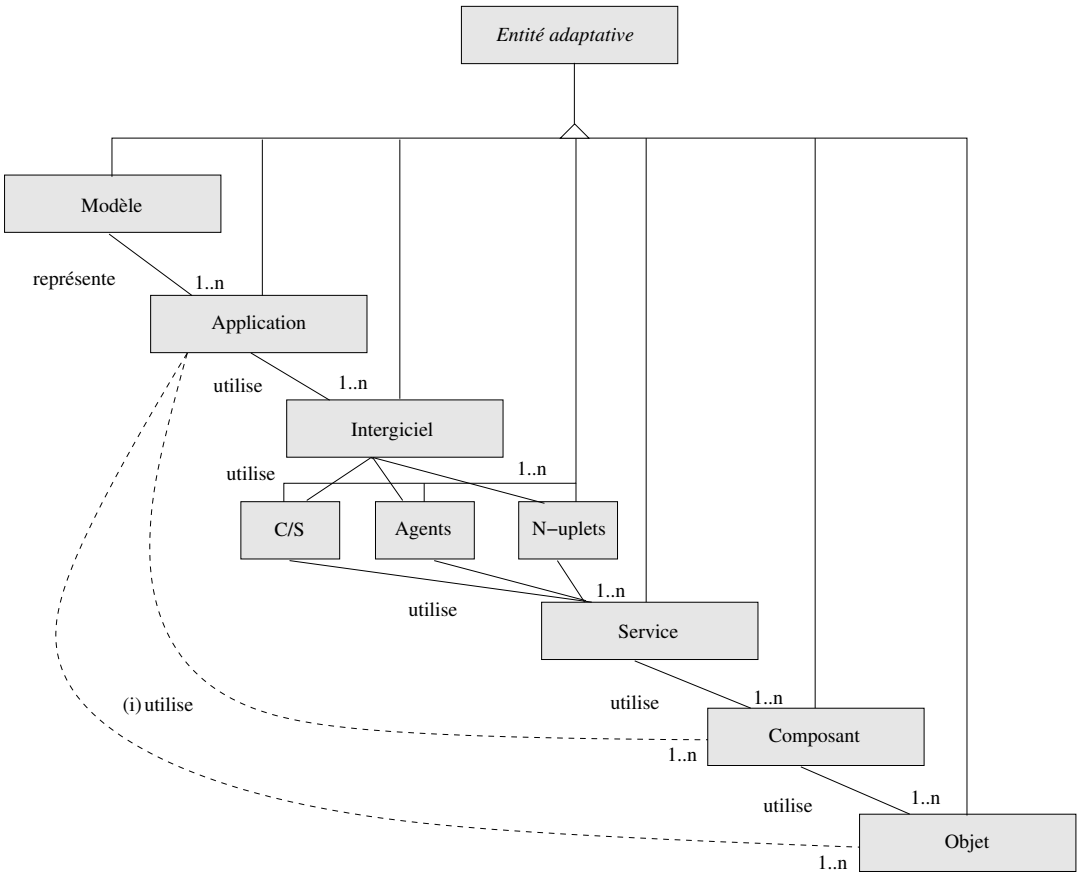


FIG. 5.20 – Échelle de représentation des abstractions de conception

terlocuteur est bien identifié "en la personne" de l'entité fonctionnelle de l'entité adaptative. Par exemple, pour un système construit à partir de nombreux services, il n'est plus besoin de connaître parfaitement les services auxquels s'adresser, il suffit de s'adresser à l'entité adaptative représentant le système.

(ii) Les adaptations pertinentes pour une instance d'un certain niveau d'abstraction sont concentrées au sein de l'entité d'adaptation de l'entité adaptative. Par exemple, pour une application construite à partir de composants, l'ajout d'un composant est traité comme une adaptation de l'application. En restant au niveau des composants, un composant particulier n'aurait pu décider de cet ajout, ne connaissant pas les dépendances (et leurs impacts) que ce composant doit instaurer sur l'ensemble des composants.

Cette échelle offre donc à notre système une granularité variable d'adaptation selon le niveau d'abstraction auquel on se place. Ainsi, un concepteur peut définir des adaptations s'appliquant du niveau le plus fin, comme la transformation du code d'un objet, au niveau le plus global, comme la transformation de modèle d'application [Bézivin 2001].

5.3.2 Propagation des adaptations au sein de l'échelle de niveaux d'abstractions

Chaque abstraction étant le résultat de la spécialisation d'une entité adaptative, l'application d'adaptations s'effectue conformément aux méthodes présentées dans le paragraphe 5.2.2. Avec cette méthode, l'application d'adaptations peut donc également influer sur les entités dépendantes de celle-ci. Ici, toutefois, les dépendances sont d'une nature impliquant des répercutions différentes. Dans le cas des dépendances de communication et de fonction, les notifications d'adaptations sont données à titre informatif aux entités dépendantes. Dans le cas de dépendances de niveau d'abstraction, toute adaptation de l'entité représentatrice entraîne des adaptations sur les entités représentées. Et, de la même manière, toute adaptation d'une entité représentée entraîne des adaptations sur l'entité représentatrice.

La figure 5.21 présente ci-dessous la manière dont sont propagées les adaptations au sein de l'échelle de niveaux d'abstractions :

```
...
public AImplementation implementationTransformation(AImplementation im,
                                                     AState st,
                                                     String[] args) { // transformation
                                                                      // d'une implantation
   AImplementation tim; // transformed implementation
   Ainteraction e1Interaction; // interaction de l'entité fonctionnelle de e1
   Ainteraction e2Interaction; // interaction de l'entité fonctionnelle de e2
   Ainteraction e3Interaction; // interaction de l'entité fonctionnelle de e3
   Ainteraction e4Interaction; // interaction de l'entité fonctionnelle de e4
   ...
   // Création d'une nouvelle instance d'implantation
   tim = new Implementation(im, st);
   // Ici, modifications de l'implantation tim
   ...
   // Application de différentes adaptations aux entités avec lesquelles
   // l'entité E présente des dépendances de niveau d'abstraction
   e1Interaction.applyAdaptation(e1Adaptation1, args);
   e2Interaction.applyAdaptation(e2Adaptation7, args);
   e3Interaction.applyAdaptation(e3Adaptation0, args);
   e4Interaction.applyAdaptation(e4Adaptation1, args);
   ...
   return(tim);
}
...
```

FIG. 5.21 – Exemple de propagation des adaptations au sein de l'échelle de niveaux d'abstractions

L'exemple ci-dessus correspond à l'application d'une adaptation sur l'entité représentatrice E. Ici, la procédure de transformation, appelée lors de l'application de l'adaptation, maintient les dépendances de niveau d'abstraction sous forme de liens vers les interactions des entités fonctionnelles. Lors de cette adaptation de E, les entités représentées E1, E2, E3 et E4 sont modifiées en conséquence par l'application d'adaptations (les adap-

tations `e1Adaptation1`, `e2Adaptation7`, `e3Adaptation0` et `e4Adaptation1` sont appliquées respectivement sur E1, E2, E3 et E4).

L'exemple présenté illustre une propagation descendante des adaptations (du niveau le plus abstrait vers le plus concret). La propagation peut aussi être ascendante. Par exemple, l'application d'une adaptation sur E1 peut très bien déclencher une adaptation sur E. Le fait de pouvoir propager de manière descendante et ascendante pose le problème de la présence de boucles : l'adaptation sur E1 déclenche une adaptation sur E, l'adaptation sur E déclenche une adaptation sur E1, etc. Ce problème correspond aux effets « boomerang » et « domino » et sera traité dans le chapitre 6.

5.3.3 Exemples de spécialisations de modèles de conception

Les figures 5.22, 5.23 et 5.24 reprennent la notation UML caractérisant une spécialisation contrainte. Le cadre de droite explicite cette contrainte. La partie supérieure du cadre montre à quoi correspondent les différentes parties de l'entité fonctionnelle. La partie inférieure du cadre détaille les adaptations possibles sur l'entité fonctionnelle.

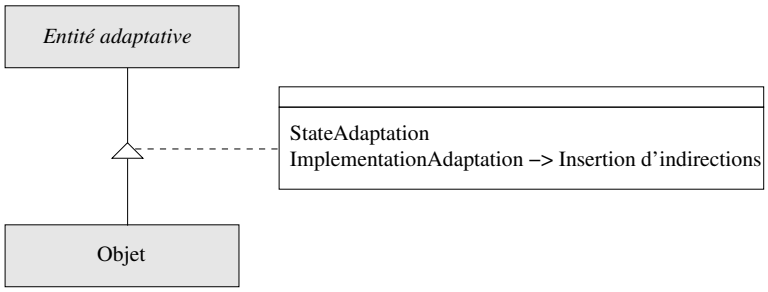


FIG. 5.22 – Modèle objet par spécialisation d'entité adaptative

La première figure présente le modèle objet. Celui-ci est un des plus simples possibles. Il ne possède aucune contrainte sur les parties de l'entité fonctionnelle. Les adaptations possibles sont aussi assez sommaires puisqu'il est principalement possible de changer d'état. Une adaptation de changement d'implantation par transformation du code de l'objet (insertion d'indirections, etc) est également possible sous réserve d'avoir accès à des outils de modification de code.

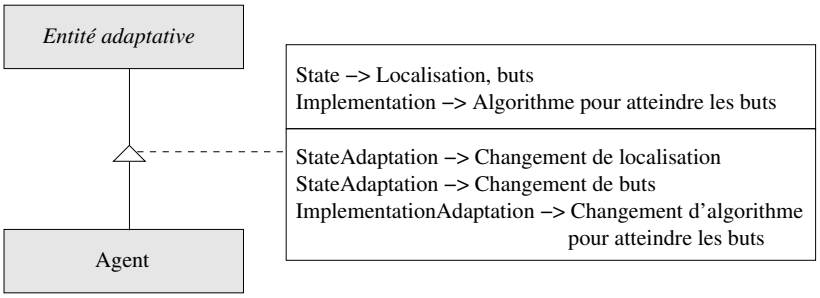


FIG. 5.23 – Modèle agent par spécialisation d'entité adaptative

La seconde figure présente le modèle agent. L'état de l'entité fonctionnelle doit comporter la localisation et les buts de l'agent. L'implantation de l'entité fonctionnelle doit effectuer les démarches nécessaires pour atteindre les buts. Les adaptations possibles sur l'agent sont donc soit (i) un changement de localisation (par une adaptation de l'état), soit (ii) un changement de buts à atteindre (également par une adaptation de l'état), soit (iii) un changement de l'algorithme de résolution des buts (par une adaptation de l'implantation).

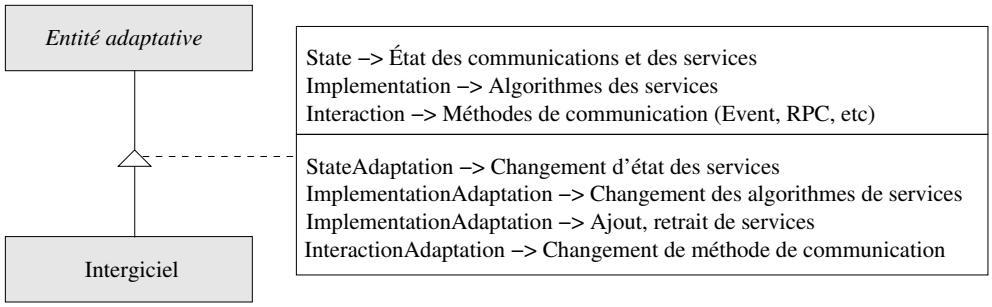


FIG. 5.24 – Modèle intergiciel par spécialisation d'entité adaptative

La troisième figure présente le modèle intergiciel. Celui-ci est un peu plus complexe. L'état de l'entité fonctionnelle est composé des états internes et des états de communication des services. L'implantation correspond à ce que peuvent faire les services. Et enfin, afin de pouvoir interagir avec un maximum d'entités extérieures, l'interaction peut disposer de plusieurs méthodes de communication (RPC, par évènements, etc). Les adaptations applicables à l'intergiciel sont alors soit (i) un changement d'état des services (par une adaptation de l'état), soit (ii) un changement d'implantation des services (par une adaptation de l'implantation), soit (iii) un ajout ou un retrait de services (par une adaptation de l'implantation), soit (iv) un changement de méthode de communication (par une adaptation de l'interaction).

Toutes les abstractions présentées dans l'échelle de niveaux peuvent ainsi résulter de la spécialisation d'une entité adaptative. Il suffit, pour cela, de prendre les spécificités propres à chaque abstraction de conception.

5.4 Conclusion

Travaux similaires. Les systèmes présentés dans les paragraphes 2.2 et 2.3 s'attellent aux problèmes de variations en environnements mobiles, et, pour cela, offrent des solutions ad hoc au problème de l'adaptation. Par rapport à ces approches, notre système propose une approche générique de gestion de l'adaptation.

Pour ce qui est des approches réflexives, plusieurs systèmes proposent l'application dynamique d'adaptations mais aucun ne tient compte d'un quelconque environnement mobile. ASM [Bruneton et al. 2002] permet d'effectuer dynamiquement n'importe quelles transformations de classes Java. Pour gagner en temps d'exécution et taille mémoire, il n'effectue aucune interposition, ne génère aucun nouveau code, ne garde aucune représentation en mémoire des adaptations et des objets à modifier, mais travaille directement sur les classes elles-mêmes.

Cette approche est extrêmement performante mais ne peut être appliquée que pour des transformations simples et ne permet pas la gestion des adaptations (comme, par exemple, pour revenir en arrière après l'application d'une transformation).

Jabyce [Lenglet 2002] propose un cadre d'adaptation découplant le modèle logique et le modèle physique. Au niveau logique, le cadre comprend des composants adaptateurs s'appliquant à des entités logiques. Au niveau physique, Jabyce utilise BCEL [Dahm 1999] (et s'applique donc au niveau des classes Java) et définit 21 types d'entités logiques (`Class`, `Method`, `Instruction`, etc) et un type de composant adaptateur pour chaque entité. Dans cette approche, le cadre d'adaptation permet de générer automatiquement les interfaces et les classes de base mais l'assemblage de composants adaptateurs est statique, ne gère pas de dépendances et peut s'appliquer à une seule classe à la fois.

[Bobeff et Noyé 2003] approfondit les techniques de spécialisation en décrivant l'évaluation partielle, la fragmentation de programme, la fusion et la fission de programme. Cette approche définit un modèle de composant prenant en compte les contraintes de ces différentes techniques de spécialisation : conditions d'utilisation, dépendances intra- et inter-composants. Notre système d'adaptation étant suffisamment générique, chacune des techniques utilisées dans ces systèmes peut être intégrée dans le nôtre.

De plus, ces approches se placent toujours à un niveau d'abstraction bien défini comme les composants [Cervantes et Ketfi 2002], les connecteurs [Vadet et Merle 2002, Budau et Bernard 2003], les aspects [Beauvois et Coupaye 2002], les intergiciels [Quéma et al. 2002] ou les applications [Roose et al. 2002]. Aucune approche ne propose de modèle d'adaptation pour différents niveaux d'abstractions.

[Berger 2002, Blay-Fornarino et al. 2002] présente une approche intéressante et complémentaire de la nôtre. En effet, dans notre système, lors d'une adaptation, les entités dépendantes de celle-ci en sont informées et peuvent alors décider du comportement à suivre (s'adapter également, rompre la dépendance, négocier, etc). Lors d'une interaction de ce type, notre système n'offre toutefois que les mécanismes de notification ou d'application d'une adaptation. [Berger 2002, Blay-Fornarino et al. 2002] propose un service d'interaction qui pourrait être utilisé pour codifier les comportements des différentes interactions possibles. Le langage ISL utilisé permet une grande flexibilité comme la fusion d'interactions assurant la commutativité et la transitivité lors de l'instanciation d'interactions.

Respects des objectifs. La généricité de notre système est fournie par nos unités logicielles de conception : les entités, les entités adaptatives et les entités auto-adaptatives. Celles-ci abstraient la conception et propose une granularité variable d'adaptation (selon une échelle de niveaux d'abstractions).

Les entités adaptatives proposent différentes méthodes d'accès permettant l'ajout et le retrait d'adaptations. Les entités auto-adaptatives proposent différentes méthodes d'accès permettant l'ajout, le retrait et la modification d'une stratégie d'adaptation. Celle-ci maintient les décisions d'adaptations à prendre et interagit avec le service de détection et notification pour vérifier les variations et appliquer les adaptations le cas échéant. Ces méthodes autorisent les concepteurs, les applications ou le système d'adaptation lui-même à effectuer les spécialisations des entités.

Le découpage en entités, selon le niveau fonctionnel et niveau méta, avec l'utilisation de méthodes d'introspection et d'intercession, procure une grande dynamicité et une grande souplesse d'évolution à notre système. Toute adaptation peut, en effet, s'effectuer dynamiquement en modifiant ou redéfinissant uniquement la partie spécifique concernée.

Chapitre 6

Conception d'un système de gestion de ressources et de distribution d'applications en environnements mobiles

L'utilisation de terminaux portables couplés avec des moyens de communication sans-fil est en pleine expansion. Les utilisateurs de telles configurations attendent maintenant d'accéder aux mêmes fonctionnalités que celles présentes dans leur configuration classique (statique). Le chapitre 1 a mis en évidence les contraintes liées aux environnements mobiles, notamment la pauvreté des ressources d'un terminal portable (comparée aux besoins importants des applications actuelles) et leur extrême variabilité.

Pour pallier ces inconvénients, nous proposons une approche consistant à mettre à profit l'environnement d'un terminal portable pour l'exécution de ses applications. Notre approche prend donc deux hypothèses sur l'environnement : (i) la qualité et la quantité des ressources disponibles dans l'environnement d'un terminal portable sont supérieures (ou au moins égales) aux ressources dont dispose le terminal portable, (ii) la variabilité de l'environnement d'un terminal portable (en terme de conséquence de la mobilité : disponibilité des ressources, etc) est inférieure (ou au moins égale) à la variabilité du terminal portable. Ces hypothèses sont réalistes et s'appliquent dans la plupart des cas comme un problème de dimensionnement. Si l'on prend, comme exemple, un utilisateur à bord d'un train à grande vitesse, celui-ci est trop mobile vis à vis de cellules de communication de type réseaux locaux sans-fil, mais il ne l'est pas vis à vis de cellules de communication radio à courte portée avec les autres utilisateurs du train.

Ce chapitre présente les fonctionnalités de gestion de ressources et de distribution d'applications de notre système. Le paragraphe 6.1 reprend les propriétés que doit satisfaire notre système, tout particulièrement pour ces fonctionnalités. En premier lieu, le paragraphe 6.2 caractérise l'environnement mobile avec le modèle de *Ressources* / *Entités utilisatrices* (paragraphe 6.2.1), les particularités de l'environnement mobile (paragraphe 6.2.2) et la prise en compte de ce modèle dans les entités préalablement définies (paragraphe 6.2.3). Le pa-

ragraphe 6.3 décrit le modèle d'applications que nous allons considérer.

Ensuite, le paragraphe 6.4 définit le cœur de notre système de gestion des ressources et de distribution d'applications, avec son architecture et ses services (paragraphe 6.4.1), l'introduction de *politiques adaptatives* dans les services (paragraphe 6.4.2), la définition des services et politiques par rapport aux entités (paragraphe 6.4.3) et enfin la manière de contrôler de possibles adaptations en chaîne au sein de ces services (paragraphe 6.4.4).

Pour finir, nous concluons en revenant sur les travaux similaires et sur les objectifs fixés.

6.1 Propriétés du système de gestion de ressources et de distribution d'applications

Les fonctionnalités de gestion des ressources et de distribution des applications s'inscrivent dans notre système et doivent satisfaire, au même titre, aux propriétés générales énoncées :

Généricité : pour que les fonctionnalités de gestion des ressources et de distribution des applications puissent s'appliquer à n'importe quel environnement mobile, il est nécessaire de précisément caractériser : (i) l'environnement mobile, avec les ressources mises à disposition, avec leurs particularités comme la variabilité, etc, (ii) la manière dont les applications peuvent spécifier leurs besoins, (iii) et, enfin, la manière de faire correspondre les deux premiers points.

Modularité : le découpage en fonctionnalités ne doit pas occulter la granularité obtenue dans l'échelle d'abstractions du chapitre précédent. Les fonctionnalités de gestion des ressources et de distribution des applications doivent donc réutiliser le concept d'entité, avec éventuellement différentes modifications répondant aux exigences de ces fonctionnalités.

Adaptabilité :

- **Prise en compte de l'environnement :** le but des fonctionnalités de gestion des ressources et de distribution des applications est justement de tirer parti de l'environnement de la meilleure façon possible.
- **Prise en compte de l'application :** les fonctionnalités de gestion des ressources et de distribution des applications doivent s'acquitter de deux sortes d'exigences de la part d'une application : (i) les besoins (caractérisés à l'aide de la spécification citée dans le point "Généricité") doivent évidemment être satisfaits dans la mesure du possible (de l'environnement), (ii) tout comme dans le chapitre précédent une application peut spécialiser les adaptations possibles et les stratégies d'adaptations, ici, une application doit pouvoir spécialiser les différentes politiques correspondant à la manière dont elle souhaite que les ressources soient gérées et la manière dont différentes applications peuvent être distribuées.

Évolutivité : les spécifications doivent être suffisamment souples pour permettre d'ajouter (et de retirer) facilement de nouvelles ressources et de nouveaux besoins. Les fonctionnalités doivent aussi être suffisamment adaptables pour permettre d'ajouter (et de retirer) de nouvelles politiques de gestion des ressources ou de distribution des applications.

Dynamicité : il est indispensable que les évolutions précédemment citées puissent s'effectuer dynamiquement. En effet, en raison de la propriété de déplacement intrinsèque aux environnements mobiles, l'apparition et la disparition de ressources, l'arrivée et le départ dans différents réseaux de communication sans-fil sont considérés comme des évènements normaux qui doivent directement être impactés sur le fonctionnement de notre système et des applications.

Efficacité :

Performances : le paragraphe 3.2.3 a montré qu'il n'y a pas de politique optimale pour tous types d'environnements mobiles. Afin de spécialiser les politiques de nos fonctionnalités de gestion de ressources et de distribution d'applications, notre boîte à outils doit donc fournir diverses implantations optimisées pour différents environnements mobiles. Les mécanismes d'adaptation dynamique permettront ensuite d'appliquer judicieusement le changement d'implantations des politiques.

Stabilité : une variation dans l'environnement induit souvent plus d'un seul changement, se traduisant par l'application de plusieurs adaptations. L'application d'adaptations en chaîne peut, dans certains cas, se révéler désastreuse pour la stabilité du système. Par exemple, pour la fonctionnalité de distribution, si l'adaptation à appliquer est la migration d'une entité applicative, l'enchaînement de migrations conduit au résultat inverse d'optimisation d'exécution de l'application (de part le coût lié à chaque migration). Nous allons donc traiter dans ce chapitre des effets « boomerang » et « domino » (le système se trouve dans un état e qui déclenche la $i^{ème}$ adaptation sur une fonctionnalité et amène le système dans l'état e', cet état déclenche la $i+1^{ème}$ adaptation sur une autre fonctionnalité et amène le système dans l'état e'', etc).

6.2 Caractérisation de l'environnement mobile

Avant de définir les différents services de gestion de l'environnement mobile, il est nécessaire de caractériser les différents éléments se trouvant dans cet environnement mobile. Le paragraphe 6.2.1 définit ce que sont les ressources et les entités qui peuvent les utiliser. Le paragraphe 6.2.2 montre comment les aspects propres à l'environnement mobile sont pris en compte dans notre modèle. Et, enfin, le paragraphe 6.2.3 présente la manière dont cette utilisation est mise en place au niveau des entités.

6.2.1 Définition du modèle de Ressources / Entités utilisatrices

Dans le chapitre 2, nous avions donné une définition d'un environnement mobile. Afin de mettre en évidence plusieurs aspects intervenant dans la conception de notre système de gestion de ressources et de distribution d'applications, cette définition peut être précisée pour l'environnement d'un terminal portable :

Environnement d'un terminal portable : l'environnement d'un terminal portable est constitué de l'ensemble des *ressources* et *entités utilisatrices* présentes sur les terminaux (mo-

biles ou non)[1] accessibles dans la cellule de communication sans-fil.

Les ressources de l'environnement mobile correspondent ainsi aux éléments soit (i) fournissant une capacité d'effectuer des traitements, soit (ii) réalisant un traitement bien défini. Chaque ressource possède des caractéristiques propres et, pour les différencier, ces offres doivent donc être classifiées. Comme le montre la figure 6.2, nous avons opté pour une hiérarchie par spécialisation assez classique. Les ressources se divisent en deux catégories : les ressources matérielles et les ressources logicielles.

Les ressources de la première catégorie représentent les éléments matériels du terminal portable (le processeur, la mémoire, l'écran, la carte réseau, etc) mais représentent également les éléments matériels distants dans l'environnement mobile (une imprimante, un scanner, les éléments matériels de terminaux distants, etc). Les ressources de la seconde catégorie correspondent à différents éléments logiciels pouvant aller du niveau le plus proche de la machine (type et services des systèmes d'exploitation), au niveau des intergiciels (avec, par exemple, les services de notre propre système) jusqu'au niveau applicatif (librairies applicatives ou directement les applications : commerce électronique, journaux électronique, etc).

Les entités utilisatrices correspondent aux éléments logiciels ayant des besoins d'effectuer des traitements. Pour exprimer ces besoins en fonction des caractéristiques des ressources, nous avons opté pour un modèle Ressources / Offres et Entités utilisatrices / Demandes (voir figure 6.1).

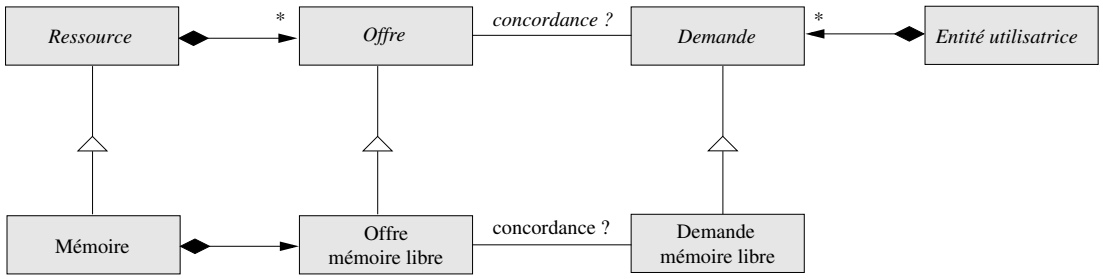


FIG. 6.1 – Modèle Ressources / Offres et Entités utilisatrices / Demandes

Une *Ressource* se voit donc adjointe un certain nombre d'*Offre*. Leur lien est une relation de composition et signifie donc qu'une instance d'*Offre* est liée à une seule instance de *Ressource* et que la destruction de l'instance *Ressource* entraîne la destruction de toutes les instances d'*Offre* liées (une offre n'a, en effet, aucun raison d'être s'il n'existe pas de ressource correspondante). L'*Entité utilisatrice* possède, de manière symétrique, un certain nombre de *Demande* (également liées par composition). Pour effectuer la correspondance entre ces *Offre* et *Demande*, plusieurs méthodes de concordance sont spécifiées.

Les attributs d'une ressource et les termes des offres et demandes ne sont concrètement implantés que lors de la spécialisation. Ainsi, dans la figure 6.1, *Ressource* est spécialisée en Mémoire, *Offre* est spécialisée en Offre mémoire libre. Si l'*Entité utilisatrice* nécessite de la mémoire libre, elle doit alors créer une demande coïncidente Demande mémoire libre, spécialisation de *Demande*.

[1] Y compris le terminal portable lui-même.

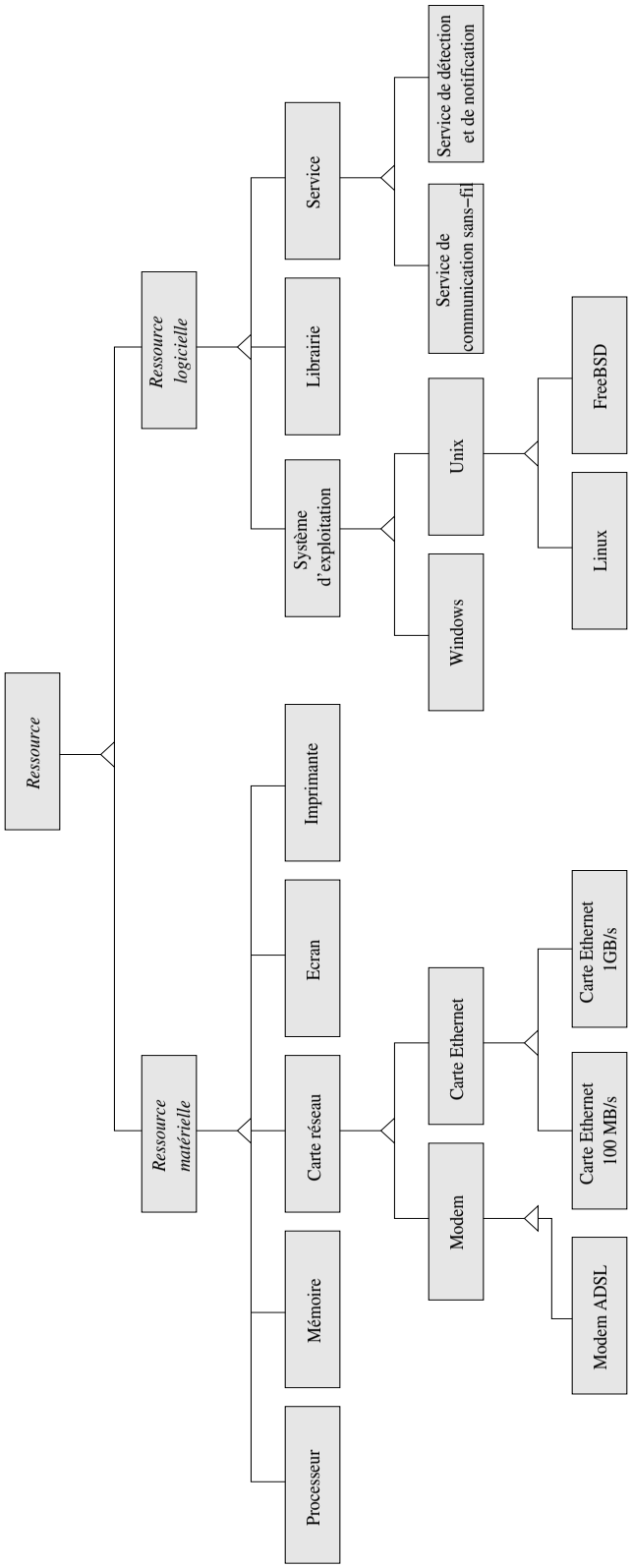


FIG. 6.2 – Exemples de classification des ressources par spécialisation

L'établissement d'une concordance est bijective, ce qui signifie qu'il faut que l'offre concorde avec la demande mais aussi que la demande concorde avec l'offre. Cela se traduit par les méthodes `boolean isConformed(Demand)` et `boolean isConformed(Offer)`, respectivement au sein d'Offre mémoire libre et de Demande mémoire libre, spécialisations d'*Offre* et de *Demande*.

La figure 6.3 donne l'exemple d'une implantation de ces méthodes :

```
class FreeMemoryOffer extends Offer { // Offre de mémoire disponible

   protected int freeMemory;
   ...
   public int getValue() {
      return(freeMemory);
   }
   ...
   public boolean isConformed(Demand demand) { // Méthode de vérification de la
                                               // conformité entre offre et demande
      // Vérification de la conformité du type
      if (!demand.getClass().getName()
          .equals("AeDEn.toolkit.environment.resources.FreeMemoryDemand"))
      {
         return(false);
      }
      else {
         return(decisionMetric(demand) == 1);
      }
   }
   ...
   public float decisionMetric(Demand freeMemoryDemand) { // Métrique de décision
      if (freeMemoryDemand.getValue() <= freeMemory) {
         return(1);
      }
      else {
         return(0);
      }
   }
   ...
}
```

FIG. 6.3 – Exemple d'implantation de la méthode de concordance et de la métrique de décision au sein d'une offre de mémoire disponible

En premier, les termes de l'offre sont caractérisés (ici, valeur `freeMemory` et méthode `int getValue()`). Ensuite, lors du test de concordance d'une demande, le type de la *Demande* est vérifié et doit correspondre à la demande symétrique de l'offre courante (ici, `FreeMemoryDemand`). Cette vérification est effectuée dans le but de pouvoir accéder aux attributs symétriques de ceux de l'offre (sinon, par exemple ici, l'application de la méthode `int getValue()` n'aurait aucun sens sur un autre type de demande comme Demande de service de détection). La méthode de concordance fait appel à une métrique de décision,

qui donne un coefficient de concordance entre l'offre et la demande concernés. Ici, dans notre exemple, la métrique est simple puisqu'elle est binaire (la demande en mémoire est-elle inférieure à l'offre ?). Mais, elle peut être plus complexe, comme par exemple, sous la forme d'un pourcentage reflétant la demande qui maximise le plus l'offre de mémoire.

Nous avons séparé ces deux méthodes par souci de réutilisabilité. En effet, la métrique de décision peut très bien être appelée par les entités utilisatrices pour qu'elles puissent reformuler leurs demandes de manière à atteindre la concordance. De la même manière, les mêmes méthodes de concordance et métrique de décision existent dans une *Demande* et peuvent être, par exemple, utilisées dans le cas où une ressource cherche à satisfaire exactement une entité utilisatrice définie.

6.2.2 Particularités liées à l'environnement mobile

Parmi les ressources de l'environnement, la plupart de celles-ci ne sont pas propres aux environnements mobiles et se retrouvent dans presque toutes les machines (processeur, mémoire, etc). Le fait d'être dans un environnement mobile nous oblige toutefois à devoir (i) prendre en compte des ressources qui n'existent pas ailleurs, et (ii) réexaminer les caractéristiques des ressources préexistantes en fonction de ces nouvelles ressources.

Ces nouveautés s'orientent selon deux axes :

(i) Axe du déplacement / mouvement :

(a) Plusieurs nouvelles ressources doivent être crées : carte réseau sans-fil de type WaveLan, carte réseau sans-fil de type Bluetooth, carte réseau sans-fil satellitaire, carte GPS (*Global Positioning System*), etc. Elles possèdent toutes, en plus des caractéristiques classiques (débit, latence, etc), de nouvelles caractéristiques comme la qualité du signal, la force du signal, le niveau du bruit ou la localisation. Ces caractéristiques ne sont pas facilement interprétables mais elles permettent de calculer des indices plus explicites, comme le taux de variabilité du débit de la bande passante, les probabilités de déconnexion / reconnexion ou une estimation de la durée de connexion / déconnexion.

(b) Après l'introduction de ces nouvelles ressources, les ressources préexistantes doivent être mises à jour avec la caractéristique de localisation. En effet, dans un environnement statique, peu importe la localisation d'une ressource du moment qu'elle est accessible. Par contre, dans un environnement mobile, la localisation permet de savoir si l'accessibilité est possible, durable et conditionne même les données échangées dans certaines application d'ubiquité [Spreitzer et Theimer 1993, Schmidt et al. 2002].

(ii) Axe de la consommation d'énergie :

(a) La principale ressource devant être créée est la batterie. Ces principales caractéristiques sont son niveau de charge et son taux de décharge. Ils permettent de calculer une estimation de la durée de vie de la batterie.

(b) Les ressources préexistantes doivent être mises à jour avec la puissance électrique qu'elles consomment. Cette puissance importe peu dans un environnement statique

où les machines sont reliées au secteur[2]. À l'inverse, dans un environnement mobile, la consommation des ressources se trouvant sur un terminal portable s'avère primordiale puisqu'elle conditionne la durée d'utilisation du terminal portable.

Les nouvelles ressources peuvent être insérées dans la hiérarchie précédente aussi facilement que les ressources préexistantes. Ces ajouts et modifications peuvent être réalisés dynamiquement, car la méthode de concordance effectue lors de l'exécution une vérification de type permettant de déterminer si les nouvelles caractéristiques sont bien accessibles aux demandes. Cela rend notre modèle de ressources parfaitement adapté aux environnements mobiles.

6.2.3 Gestion du modèle au niveau d'une entité

En examinant notre classification des ressources, il est aisé de constater que celle-ci parcourt plusieurs niveaux d'abstractions de conception (du niveau matériel jusqu'au niveau applicatif). De même, en examinant notre classification des abstractions de conception (voir paragraphe 5.3.1), il est aisé de voir que chaque niveau offre ses fonctionnalités aux niveaux supérieurs et nécessite des fonctionnalités des niveaux inférieurs.

Pour bénéficier des avantages de ces deux classifications, nous avons intégré le modèle Offres / Demandes à l'intérieur du modèle d'une entité. Cela nous amène à la définition d'une Ressource / Entité utilisatrice.

Une Ressource / Entité utilisatrice est l'unité logicielle pouvant gérer l'utilisation extérieure de sa propre fonctionnalité. Par définition, une ressource / entité utilisatrice est donc une entité (comme préalablement définie dans le chapitre précédent) à laquelle est rajoutée la gestion de ses offres / ses besoins de fonctionnalités.

La figure 6.4 présente la structure d'une ressource / entité utilisatrice :

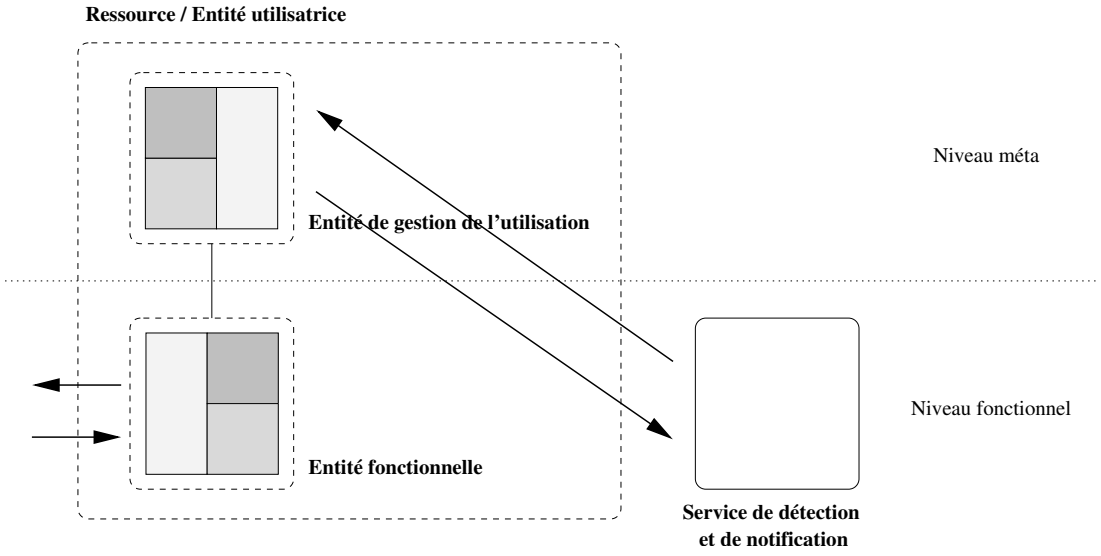


FIG. 6.4 – Structure d'une Ressource / Entité utilisatrice

[2]Sauf, peut-être, pour la facture d'électricité :-)

Au niveau fonctionnel, une ressource / entité utilisatrice comporte l'entité fonctionnelle qui effectue les traitements attendus. Dans le cas d'une ressource matérielle, l'entité fonctionnelle comprend l'ensemble des méthodes d'accès à la ressource physique. Au niveau méta, une entité de gestion de l'utilisation est liée à l'entité fonctionnelle et matérialise la description des offres et des demandes effectuées par l'entité. Les offres et les demandes étant liées à des ressources et entités utilisatrices pouvant fortement évoluer dans l'environnement mobile, l'entité de gestion de l'utilisation interagit avec le service de détection et notification afin de répercuter les variations sur les offres et les demandes. Comme pour les entités adaptatives et auto-adaptatives, nous avons choisi de modéliser l'entité de gestion de l'utilisation par une entité pour les mêmes raisons de dynamicité évoquées dans le paragraphe 5.2.2.

La figure 6.5 détaille la structure d'une ressource / entité utilisatrice, notamment la manière dont est implantée la gestion de l'utilisation :

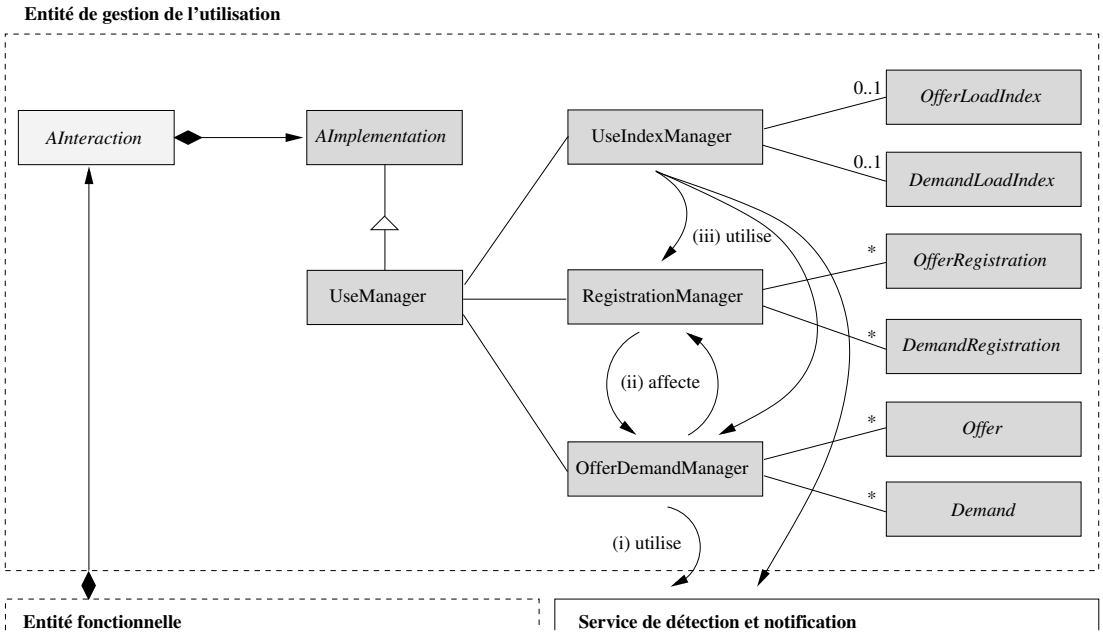


FIG. 6.5 – Relations entre éléments de la structure d'une Ressource / Entité utilisatrice

L'entité de gestion de l'utilisation est liée à l'entité fonctionnelle par une relation de composition. Comme pour les entités adaptatives, cela signifie que ces deux instances sont liées par l'exclusivité de leur relation et de leur cycle de vie.

Le cœur de l'entité de gestion de l'utilisation se trouve être le type UseManager, obtenu par héritage du type *AImplementation*. Il comporte les liens vers les trois parties essentielles suivantes :

(i) Le gestionnaire des offres et des demandes (type OfferDemandManager) **:** celui-ci est en charge de répertorier les offres (type *Offre*) et les demandes (type *Demand*) effectuées sur l'entité courante. En fonction des ressources dont il publie les offres, il utilise le service de détection et notification pour surveiller d'éventuelles variations (point i de la figure 6.5).

(ii) Le gestionnaire de réservations (type RegistrationManager) **:** lorsque la demande d'une entité utilisatrice concorde avec l'offre de la ressource, le gestionnaire de réservations est en charge d'enregistrer une réservation sur la ressource et sur l'entité utilisatrice. Il effectue également la libération de la ressource lorsque l'entité utilisatrice a terminé. Il interagit avec le gestionnaire des offres et demandes de deux manières possibles (point ii de la figure 6.5) :

(a) Lorsqu'une réservation ou une libération est effectuée, le gestionnaire de réservations informe le gestionnaire des offres et demandes pour que celui-ci modifie ses offres afin de s'accorder avec ces deux évènements.

(b) Si le gestionnaire des offres et des demandes retire une offre sur une ressource qui avait été réservée, le gestionnaire de réservations doit effectuer une libération et en informer l'entité utilisatrice concernée.

(iii) Le gestionnaire des indices d'utilisation (type UseIndexManager) **:** celui-ci comporte deux indices reflétant la charge supportée (type *OfferLoadIndex*) et induite (type *DemandLoadIndex*) par l'entité courante. Le rôle du premier indice est d'indiquer, dans le cas où il devient trop élevé, que l'entité doit être déchargée de la trop grande importance des traitements actuels. Le rôle du second indice est d'indiquer, dans le cas où il devient trop élevé, que l'entité utilise trop de ressources et qu'il faudrait peut-être diminuer ses demandes.

Les types de ces indices sont abstraits, laissant ainsi à chaque entité le soin de les spécialiser selon l'estimation de charge que chacune désire. Le calcul de ces indices peut prendre en compte différentes caractéristiques internes des offres et demandes (nombre, importance, etc) et des réservation (nombre, etc), mais aussi de différents paramètres extérieurs (comme plusieurs indices qui peuvent être directement mesurés sur les ressources à l'aide du service de détection et notification) (point iii de la figure 6.5). Laisser chaque entité spécialiser ses propres indices posent néanmoins le problème de l'homogénéité des indices. Nous avons opté pour la même échelle de spécification d'indice que l'on peut rencontrer dans les machines Unix : un indice de charge de 0 correspond à aucun traitement effectué par l'entité, un indice de charge de 1 correspond à une utilisation à plein temps de l'entité, une charge de 2 correspond à un partage équivalent en terme d'offres et de demandes à deux traitements, etc.

Les ressources particulières aux environnements mobiles peuvent être prises en compte par les nouveaux éléments de gestion introduits dans l'entité. Prenons l'exemple d'une entité se trouvant sur un terminal portable et ayant manifestée un intérêt pour les variations que celui-ci pourrait subir (enregistré auprès du service de détection et notification).

Si la probabilité de déconnexion laisse présumer d'une déconnexion prochaine, ou si le niveau de la batterie laisse présumer un arrêt prochain du terminal, le service de détection et notification avertit le gestionnaire des offres et demandes. Si l'entité se trouve être une ressource locale, le gestionnaire des offres et demandes peut décider de retirer ses offres (affectant aussi le gestionnaire des réservations). Si l'entité se trouve être une entité utilisatrice, le gestionnaire peut décider de retirer ses demandes, d'arrêter ses réservations locales ou de changer ses réservations locales pour des offres non locales (affectant également le gestionnaire des réservations).

Le service de détection et notification avertit également le gestionnaire d'utilisation qui modifie alors ses indices. Si l'entité se trouve être une ressource locale, l'indice de charge supportée peut être augmenté pour indiquer que la ressource va avoir de plus en plus de mal à réaliser les traitements en cours. Cet indice peut même être positionné à l'infini si la ressource estime ne pouvoir finir aucun traitement avant la déconnexion ou l'arrêt du terminal portable. Si l'entité est une entité utilisatrice, l'inde de charge induite peut lui aussi être augmenté pour indiquer que les traitements demandés sont trop contraignants pour le terminal portable et qu'ils devraient être redirigés vers une autre station. Cet indice peut même être positionné à l'infini si l'entité estime qu'elle ne pourra jamais obtenir le résultat de ses traitements.

Comme le montre l'exemple, la mise en œuvre du modèle Offres / Demandes s'accorde facilement avec les requis de l'environnement mobile. De plus, elle fournit de précieuses informations, les indices d'utilisation, pour notre système de distribution qui décidera, dans le paragraphe 6.4, du placement et de la migration des entités.

6.3 Caractérisation des applications

Avant de définir le service de distribution d'applications, il est nécessaire de définir le modèle d'applications que nous allons considérer. La figure 6.6 présente celui-ci :

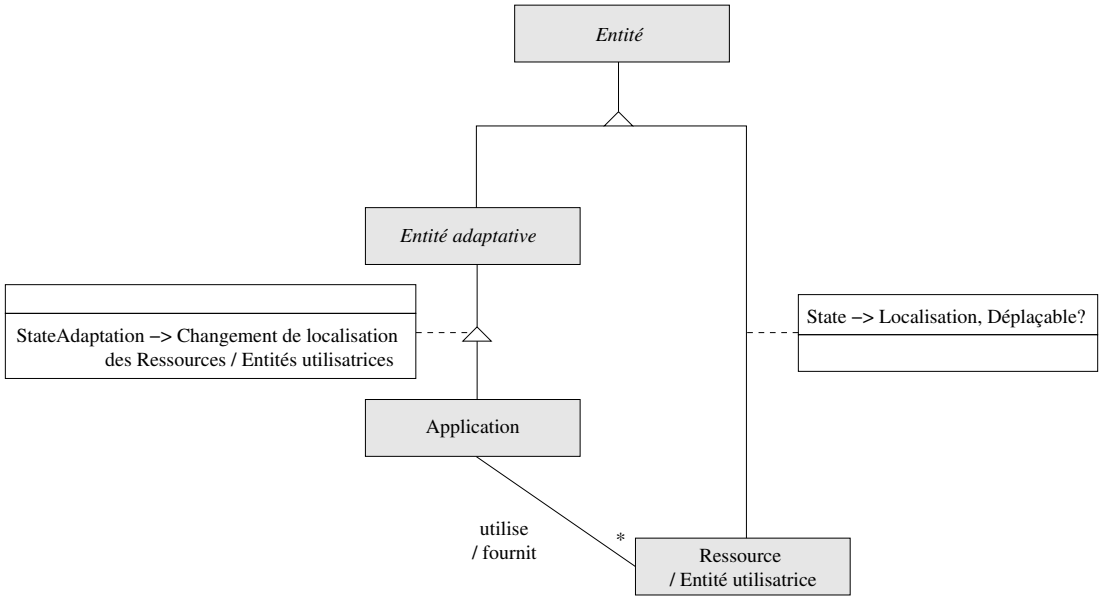


FIG. 6.6 – Modèle application par spécialisation d'entité adaptative

Nous avons précisé l'abstraction de conception, application, de l'échelle des abstractions. Une application est une entité adaptative qui utilise ou fournit un certain nombre de ressources ou entités utilisatrices. Étant une entité, une application maintient donc les mêmes dépendances de communication, de fonction et de niveau d'abstraction.

La spécialisation de l'entité adaptative en application est conditionnée par l'existence d'une adaptation de changement de localisation applicable aux ressources / entités utilisatrices liées.

Par souci d'indépendance des applications, nous avons choisi de mettre la procédure de migration au sein de l'application plutôt que dans le service de distribution. En effet, une application donnée peut très bien vouloir une méthode de migration forte alors qu'une autre souhaite une procédure de migration légère et personnalisée.

Une Ressource / Entité utilisatrice spécialise une entité[3] avec deux informations pour état : l'indication de localisation et l'indication que la ressource / entité utilisatrice est déplaçable ou non (dans le sens de possible à migrer). Cette dernière information permet d'autoriser ou non l'application d'adaptation de changement de localisation.

6.4 Caractérisation du système de gestion des ressources et de distribution des applications

Les caractérisations de l'environnement mobile et des applications nous permettent de mettre en place notre système de gestion des ressources et de distribution des applications. Le paragraphe 6.4.1 présente les services de notre système au sein d'une architecture conçue pour passer à l'échelle. Le paragraphe 6.4.2 introduit les politiques au sein des services. Le paragraphe 6.4.3 détaille comment les services et les politiques résultent de la spécialisation d'entités adaptatives. Enfin le paragraphe 6.4.4 expose la manière d'endiguer de possibles adaptations en chaîne au sein de ces services.

6.4.1 Architecture du système et passage à l'échelle des services

Dans un système de gestion des ressources et de distribution des applications, plusieurs services effectuant diverses tâches indispensables à l'ensemble des machines doivent être mis en place. Mais, étant dans un environnement mobile, le problème de la localisation de ces services se pose. Si l'on prend l'exemple du service E1 dans la figure 6.7, le déplacement du terminal portable où il se trouve entraîne la perte de connexion avec les autres machines et donc la fin des traitements et la perte de données de ce service pour les autres terminaux (point i de la figure 6.7). Une solution est de disposer du même service sur chaque machine. Mais, dans ce cas, des problèmes de compatibilité et de cohérence de comportements se posent. Si E1, E2, E3, E4 sont un seul et même service, interroger E1 doit rendre le même résultat que d'interroger E2, E3 ou E4.

Nous avons résolu ces problèmes en implantant nos services à l'aide d'entités adaptatives se trouvant à deux niveaux d'abstractions. Sur la figure 6.7, le service est implanté par l'entité E au niveau global qui représente les entités E1, E2, E3, E4 au niveau local. La cohérence de comportement est assurée, de la même manière que décrit dans le paragraphe 5.3.2, par l'application et la propagation des adaptations entre ces deux niveaux. Cette architecture permet également de facilement passer à l'échelle. Par exemple, dans notre figure, l'entité E peut se trouver au niveau global d'une cellule de communication, et aussi être au niveau local d'un ensemble de cellules.

[3] Une Ressource / Entité utilisatrice peut spécialiser une entité adaptative ou auto-adaptative si le concepteur développe une application adaptative ou auto-adaptative.

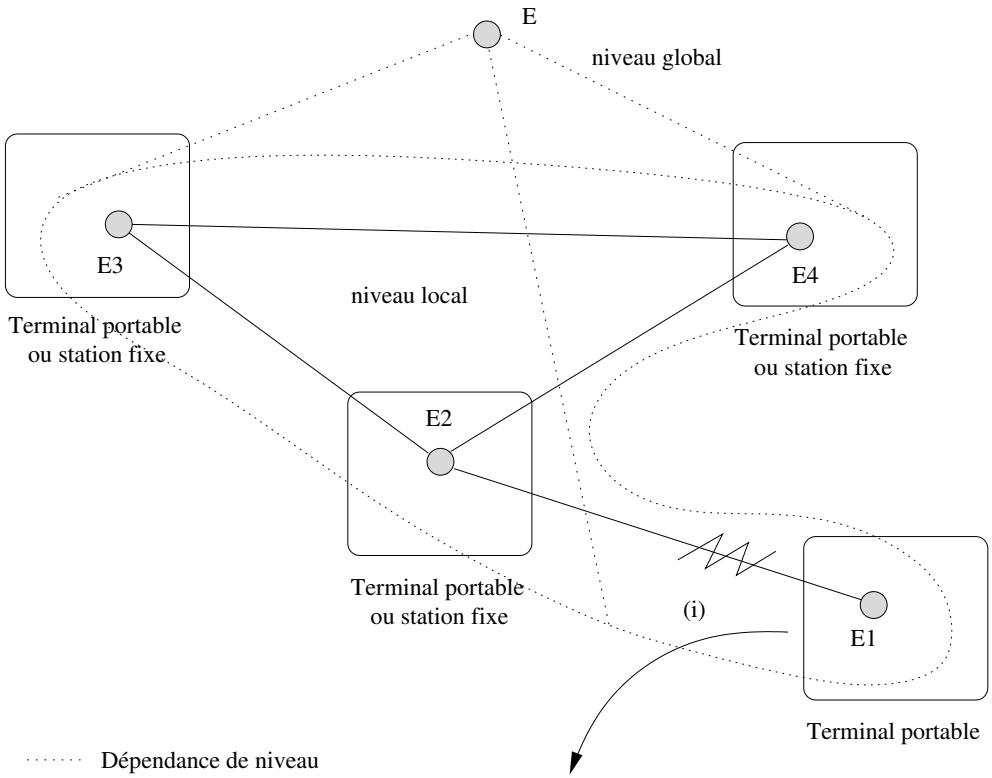


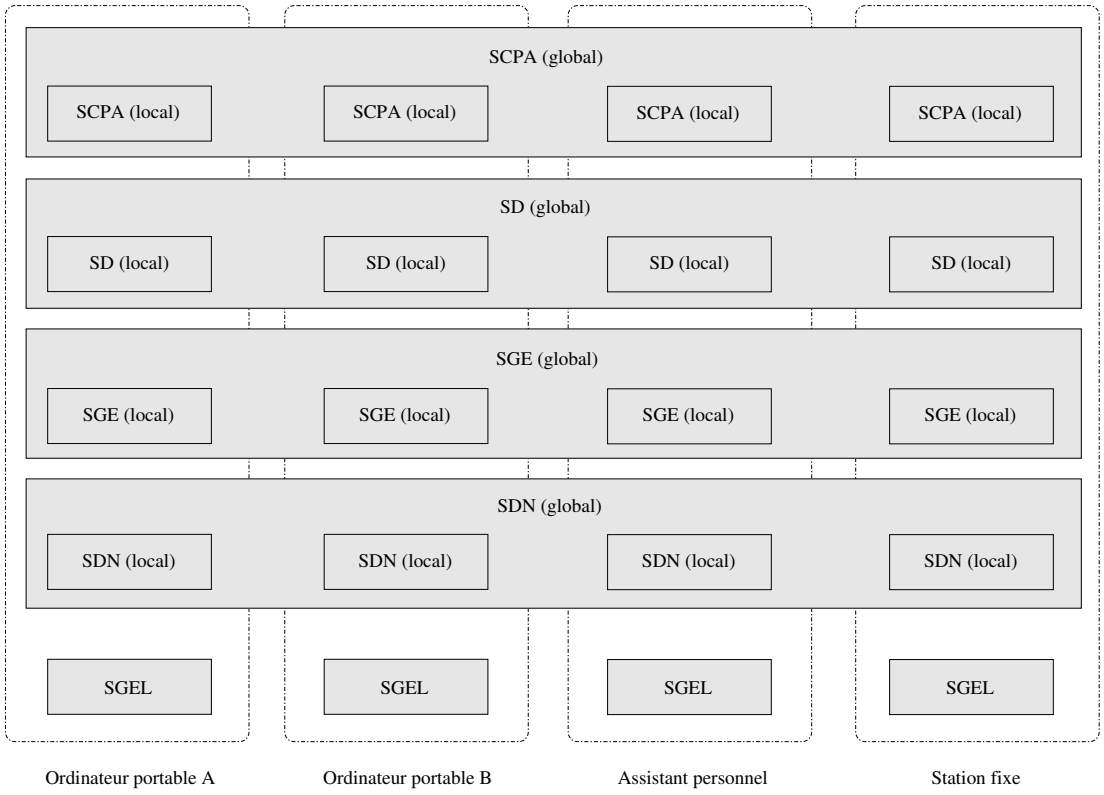
FIG. 6.7 – Implantation des services selon deux niveaux d'abstractions : local et global

Notre système se compose de cinq services construits selon ce modèle à deux niveaux d'abstractions (local, global) (voir figure 6.8) :

(i) Le service de gestion de l'environnement (SGE) : le rôle principal du SGE est de maintenir la description des ressources et entités utilisatrices de l'environnement. Cette description résulte des enregistrements de ressources et entités utilisatrices fournis par les SGEL de chaque terminal. Elle reprend (uniquement sous forme de lien), pour chaque ressource / entité utilisatrice, les principales informations les concernant : la liste des offres et demandes, les réservations, les indices d'utilisation, la station propriétaire, la possibilité de migration et la localisation actuelle.

Le SGE interagit avec le SDN, le SGEL et les applications. Le SDN notifie le SGE de tout changement intervenant sur les ressource ou les entités utilisatrices à propos desquelles le SGE à enregistré un intérêt de surveillance. Le SGEL notifie le SGE de tout nouvel enregistrement, ou de tout nouveau retrait, de ressources ou entités utilisatrices. Lorsqu'une application cherche une référence de ressource / entité utilisatrice, le SGE la lui retrouve et retourne.

(ii) Le service de gestion de l'environnement local (SGEL) : les rôles principaux du SGEL sont (a) de maintenir la liste des ressources et entités utilisatrices propres au terminal portable, et (b) de les enregistrer ou de les retirer du SGE pour les rendre accessibles ou



SGE : Service de Gestion de l'Environnement
SGEL : Service de Gestion de l'environnement Local
SD: Service de Distribution
SDN : Service de Détection et Notification
SCPA : Service de Contrôle de la Propagation des Adaptations

FIG. 6.8 – Architecture du système de gestion des ressources et de distribution des applications

non à l'ensemble des terminaux. La description des ressources / entités utilisatrices est conservée de la même manière que dans le SGE.

Le SGEL est un service uniquement local à un terminal, et c'est pour cela qu'il est le seul service à n'être pas construit selon les deux niveaux d'abstractions.

Comme le SGE, le SGEL interagit avec le SDN, le SGE et les applications. Le SDN notifie le SGEL de tout changement intervenant sur les ressources ou les entités utilisatrices locales à propos desquelles le SGEL à enregistré un intérêt de surveillance. Le SGEL notifie le SGE de tout nouvel enregistrement, ou de tout nouveau retrait, de ressources ou entités utilisatrices. Lorsqu'une application cherche une référence de ressource / entité utilisatrice locale, le SGEL la lui retrouve et retourne.

(iii) Le service de détection et notification (SDN) : les rôles principaux du service de détection et notification sont (a) de recevoir des demandes d'intérêt de surveillance sur des ressources ou entités utilisatrices, (b) de maintenir une surveillance de celles-ci, et (c) d'avertir de tout changement d'état les services ayant enregistrés l'intérêt. La manière

de décrire un intérêt pour tout changement dans une ressource / entité utilisatrice a été décrite dans les paragraphes 4.3.1 et 4.3.2.

Le SDN interagit avec tout service procédant à des enregistrements d'intérêts et, ici en particulier, le SGE et le SGEL.

(iv) Le service de distribution (SD) : le SD est en charge de la distribution des entités applicatives, qui peuvent être, comme l'a précisé le paragraphe 6.3, soit des ressources soit des entités utilisatrices. Selon les demandes et les offres des entités applicatives, le SD interroge le SGE sur les offres et ressources disponibles dans l'environnement et sur les indices d'utilisation. À partir de cela, il effectue trois opérations : (a) il calcule un placement pour les entités applicatives ressources / entités utilisatrices, (b) il applique à l'entité applicative les adaptations de changement de localisation respectant le placement calculé et (c) il met à jour le SGE avec les nouvelles entités et, particulièrement, leur nouvelle localisation.

Le SD interagit avec les applications et le SGE. En premier lieu, une application peut demander directement au SD d'être distribuée. Ensuite, lorsque le SGE reçoit des notification de changement du SDN, il avertit le SD si les ressources / entités utilisatrices concernées sont déplaçables. Selon l'importance du changement, le SD peut alors décider de réengager les trois opérations précédentes afin de trouver un placement plus approprié.

(v) Le service de contrôle de la propagation des adaptations (SCPA) : les problèmes d'application en chaîne d'adaptations, phénomènes « boomerang » et « domino », ont été décrit dans le paragraphe 5.2.4. Ces problèmes sont d'autant plus importants que nous avons choisi d'implanter la migration d'entité comme étant l'application d'une adaptation. Le rôle du SCPA est d'autoriser ou non l'application d'une adaptation. Pour cela, il (a) intercepte toute demande d'application d'adaptation, (b) compare avec l'historique des adaptations ayant déjà eu lieu, (c) décide de refuser ou d'autoriser l'adaptation et (d) dans le cas d'une acceptation, propage l'appel et met à jour son historique.

Le SCPA interagit avec toute entité adaptative, donc ici, avec les SGE, SGEL, SDN, SD mais aussi avec les entités applicatives.

6.4.2 Services et politiques

Le paragraphe 3.2.3 a montré qu'un système de répartition de charge s'appuie traditionnellement sur trois politiques : une politique d'information, une politique d'élection et une politique de placement. Il a également montré la diversité des algorithmes existants pour ces politiques, et qu'aucun ne convient à tous les environnements mobiles. Ci-dessous, le paragraphe 6.4.2.1 montre comment les politiques que nous introduisons dans les services permettent un placement dynamique. Le paragraphe 6.4.2.2 montre des exemples de changements d'algorithmes, rendus possibles grâce à nos *politiques adaptatives*.

6.4.2.1 Politiques de placement dynamique

La figure 6.9 présente les politiques que nous avons introduites au sein des services. La politique d'information a été divisée en plusieurs politiques réparties dans trois ser-

vices (SGEL, SGE, SDN) : *la politique d'enregistrement* (PEn) gère les informations locales, *la politique de gestion de l'état de l'environnement* (PGEE) gère les informations de l'environnement et *la politique de détection et notification des variations* (PDNV) effectue la mise à jour de ces informations. *La politique d'élection* (PEl) se trouve dans le SD. La politique de placement est aussi divisée en deux politiques dans deux services (SD, SCPA) : *la politique de décision de placement* (PDP) effectue les calculs de placement et demande à ce que les applications soient adaptées, et c'est *la politique de contrôle de la propagation des adaptations* (PCPA) qui décide si l'adaptation (migration) peut avoir lieu.

Notre système effectue un placement dynamique qui va être démontré par le déroulement du fonctionnement des politiques :

Une application peut déposer auprès du SD une demande de distribution (étape 1 de la figure 6.9). De manière à voir si l'environnement n'est pas trop chargé, la PEl demande l'état des ressources et entités utilisatrices au SGE (étape 2). Le SGE interrogé est local, mais il passe alors par le niveau global pour éventuellement demander des informations à ses homologues (étape 3). Selon les informations recueillies, la PEl détermine les entités qui peuvent être placées. Cette élection peut être établie en coopération avec les PEl des autres stations (étape 4). Les entités élues sont soumises à la PDP (étape 5) qui calcule alors un placement adapté. Cette décision de placement peut également être prise en coopération avec les PDP des autres stations (étape 6). Une fois la décision prise, une demande d'adaptation pour une nouvelle localisation est envoyée à l'application (étape 7). Cet appel est intercepté par le SCPA (étape 8) qui, conformément à sa PCPA ou à une décision collégiale (étape 9), peut décider de refuser cette adaptation ou de transmettre à l'application (étape 10). L'application pourra ensuite appliquer l'adaptation de placement ou de migration pour l'entité concernée (étape 11). Si l'adaptation est réalisée, la PDP enregistre alors la modification dans le SGE (étape 12), qui peut alors activer la surveillance de l'entité (étapes 13 et 14).

Quand le SDN détecte, selon sa PDNV, un changement dans une ressource (étape 15) ou une entité utilisatrice (étape 16), deux cas sont possibles. Si la ressource / entité utilisatrice est locale, le SDN notifie le SGEL (étape 17). Celui-ci, s'il l'avait enregistré dans l'environnement, peut décider de notifier le SGE (étape 18). Si la ressource / entité utilisatrice se trouve être dans l'environnement (étape 19), le SDN notifie alors directement le SGE (étape 20). Si la ressource / entité utilisatrice est déplaçable, le SGE notifie alors le SD (étape 21). Selon l'importance du changement constaté, la PEl détermine si cela doit affecter le placement actuel. Si l'élection est probante, l'établissement d'un nouveau placement est alors déclenché (étapes 2 à 14).

6.4.2.2 Politiques dynamiquement adaptatives

Les fonctionnalités rendues par nos services appliquent principalement les algorithmes se trouvant dans les différentes politiques. Pour que de la distribution résultent de bonnes performances applicatives, il est essentiel de choisir les « bons » algorithmes. Trouver un algorithme pour tous les environnements mobiles est une utopie tant la diversité de ceux-ci est grande. Catégoriser tous les algorithmes pour environnements statiques comme étant obsolètes est aussi un mauvais réflexe. De « bons » algorithmes pour un environnement mobile sont ceux qui sont le plus adéquats à un instant donné. Ce peuvent être des algorithmes optimisés pour un environ-

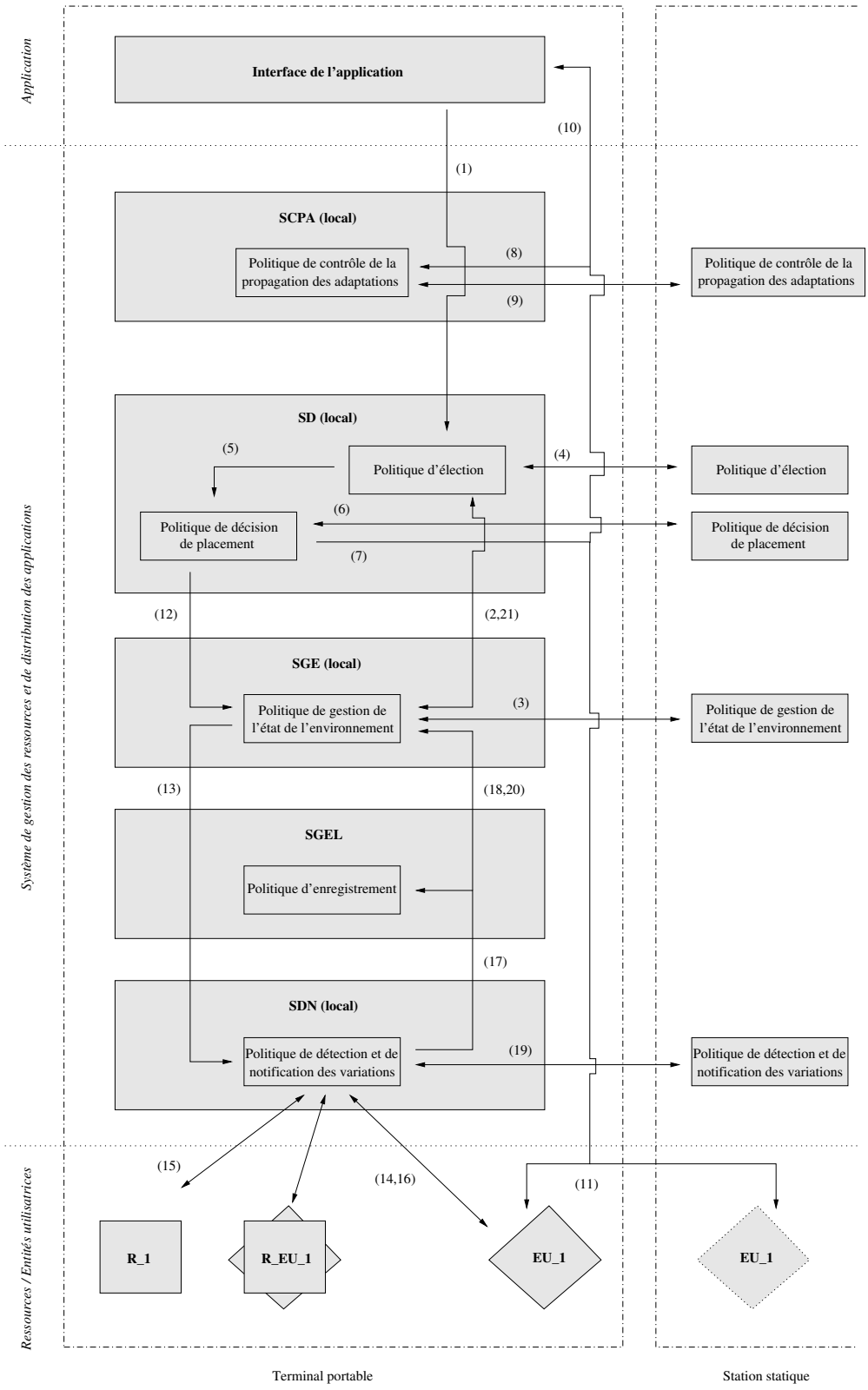


FIG. 6.9 – Introduction des politiques au sein des services

nement statique donné, des algorithmes optimisés pour un environnement mobile donné, des hybrides mêlant les deux optimisations ou des algorithmes personnalisés. L'important est donc d'avoir un mécanisme qui permette de changer d'algorithme selon les conditions d'exécution.

Nous avons donc défini des *politiques adaptatives*. Comme le montre la figure 6.10, le principe en est très simple :

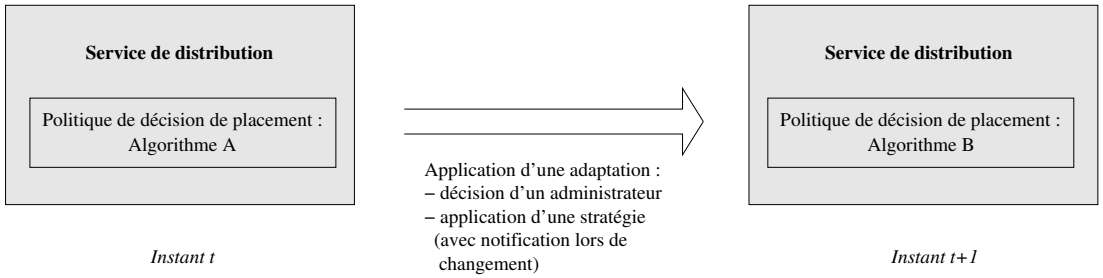


FIG. 6.10 – Exemple de politique adaptative de décision de placement

Chaque politique applique un seul algorithme à un instant t, mais peut dynamiquement changer pour un algorithme différent à un instant $t+1$. Le mécanisme de changement s'effectue par application d'une adaptation consécutive soit (i) à la décision d'un acteur extérieur, comme un administrateur, soit (ii) à la décision d'une stratégie d'adaptation.

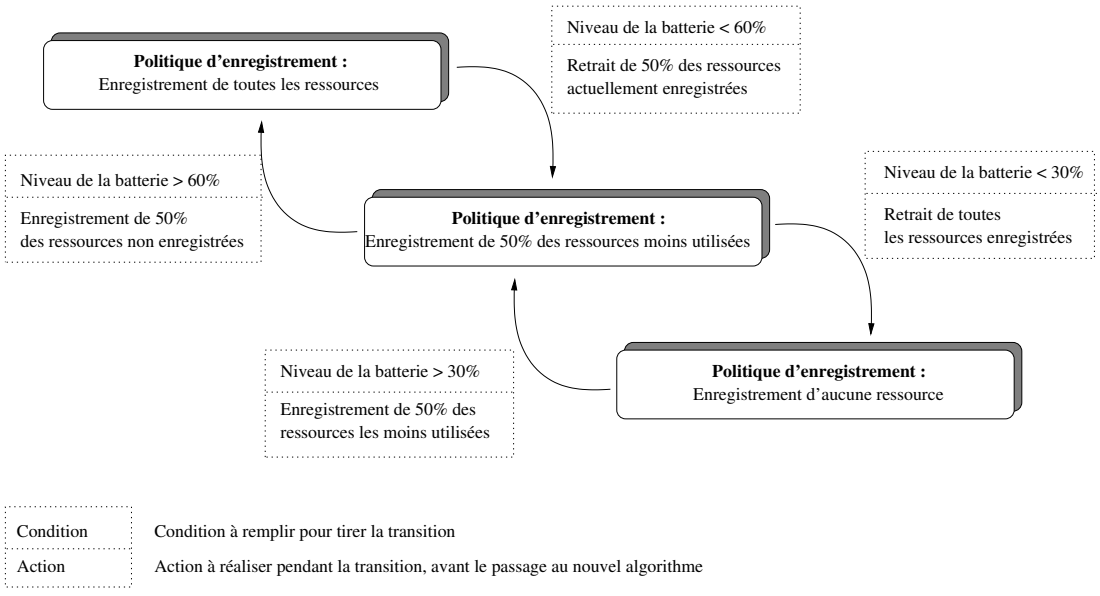


FIG. 6.11 – Exemple de stratégie pour la politique d'enregistrement

Ainsi, chaque politique peut avoir une stratégie pour adopter l'algorithme le plus approprié aux conditions de l'environnement mobile. Selon la politique considérée, le choix des algorithmes à adopter peut être très varié, comme parmi ceux présentés dans la figure 3.13. Selon la politique considérée, les conditions de changement d'algorithmes peuvent dépendre

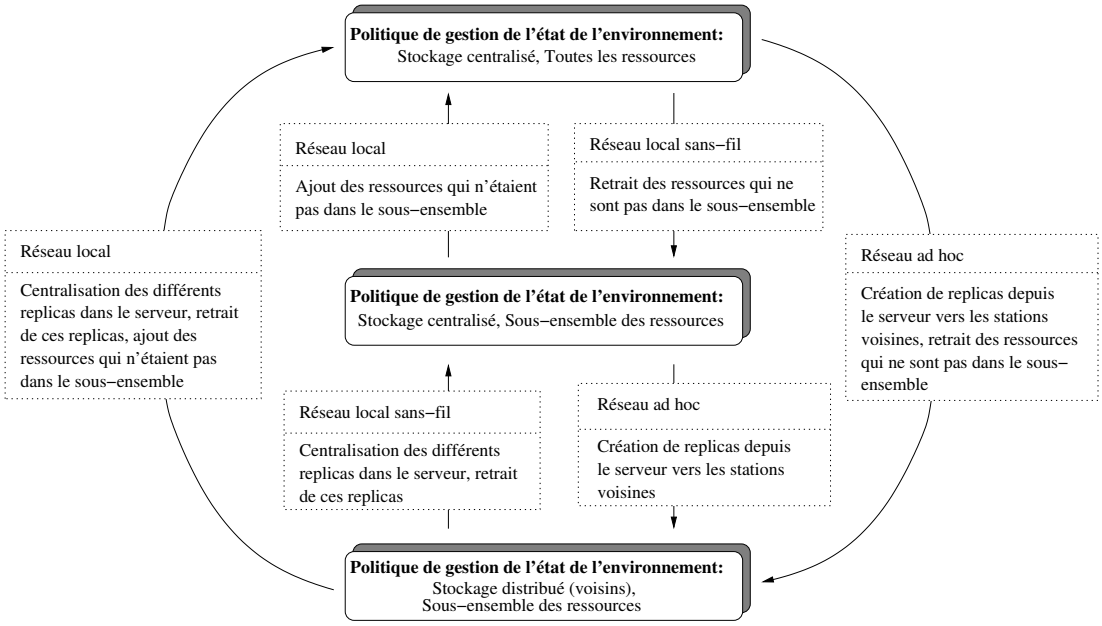


FIG. 6.12 – Exemple de stratégie pour la politique de gestion de l'état de l'environnement

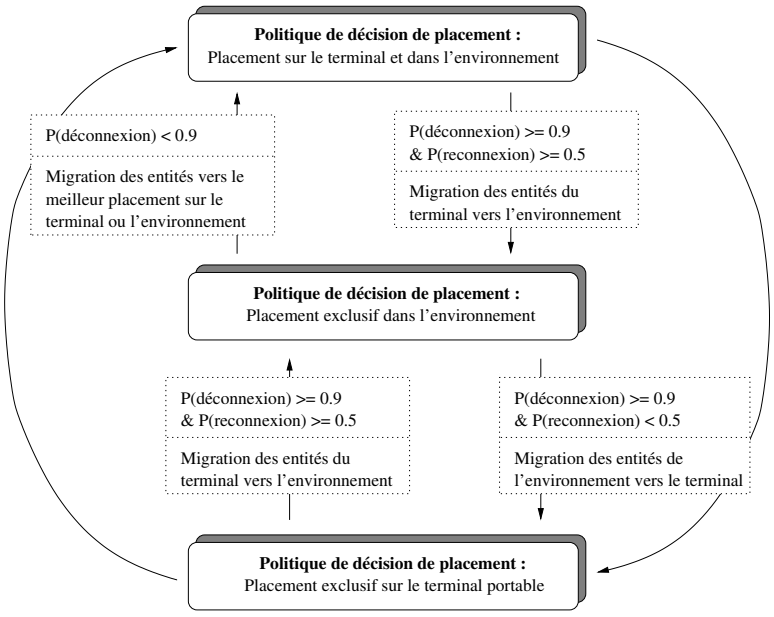


FIG. 6.13 – Exemple de stratégie pour la politique de décision de placement

de variations observées sur les ressources / entités utilisatrices.

Par exemple, les figures 6.11, 6.12 et 6.13 décrivent des stratégies possibles respectivement pour la politique d'enregistrement du SGEL, pour la politique de gestion de l'état de l'environnement du SGE et pour la politique de décision de placement du SD. La politique d'enregistrement dépend de la caractéristique d'une ressource locale, le niveau de batterie, tandis que la politique de gestion de l'état de l'environnement et la politique de décision de placement dépendent de caractéristiques de l'environnement, la topologie et les probabilités de déconnexion/reconnexion au réseau.

6.4.3 Définitions des services et politiques par spécialisations d'entités

Le modèle d'entité offre tout le matériel nécessaire pour définir les services et les politiques adaptatives de notre système. La figure 6.14 présente cette définition :

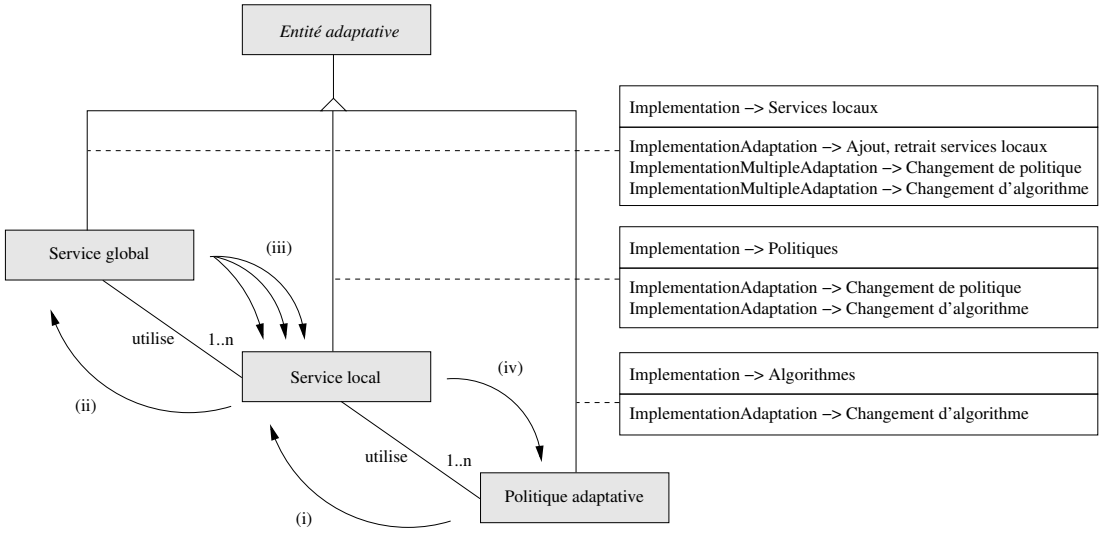


FIG. 6.14 – Modèle service (global et local) et politique adaptative par spécialisation d'entité adaptative

Au bas de cette mini-échelle d'abstractions, on peut trouver la politique adaptative. Celle-ci est définie par spécialisation d'une entité adaptative. Elle possède différents algorithmes pour implantations, et une adaptation de changement d'implantation permettant de changer d'algorithme.

À un niveau plus élevé, le service local spécialise aussi une entité adaptative. Il utilise une ou plusieurs politiques adaptatives. Il peut en changer ou changer leur algorithme grâce à deux adaptations.

Au plus haut niveau, se trouve le service global, spécialisation d'une entité adaptative. Il utilise un ou plusieurs services locaux. Les services locaux étant justement situés sur des terminaux mobiles, ils peuvent apparaître et disparaître de l'environnement mobile. Pour cela, le service global dispose d'une adaptation permettant l'ajout et le retrait de services locaux. Le service global est également en charge de maintenir une cohérence entre algorithmes et poli-

tiques. Il peut appliquer des changements d'algorithmes ou de politiques mais sur l'ensemble des services locaux. Il dispose pour cela d'adaptations multiples (conformes à celles présentées dans le paragraphe 5.2.4).

Pour illustrer ces adaptations multiples, prenons l'exemple de propagation d'une adaptation dans l'échelle. Un politique adaptative décide de changer d'implantation pour un algorithme nécessitant une parfaite cohérence avec tous ses homologues. Lors de l'application de l'adaptation, dans la procédure de transformation, la politique adaptative applique cette même adaptation sur le service local dont elle dépend (dépendance de niveau) (point i de la figure 6.14). Celui-ci, dans sa procédure de transformation, fait de même sur le service global (point ii). Le service global applique alors son adaptation multiple, qui a pour effet d'appliquer l'adaptation à l'ensemble des services locaux concernés (point iii). Enfin, ceux-ci répercutent l'adaptation sur leur politique adaptative (point iv). La politique adaptative initiatrice obtient donc une parfaite cohérence avec ses homologues.

6.4.4 Contrôle des adaptations

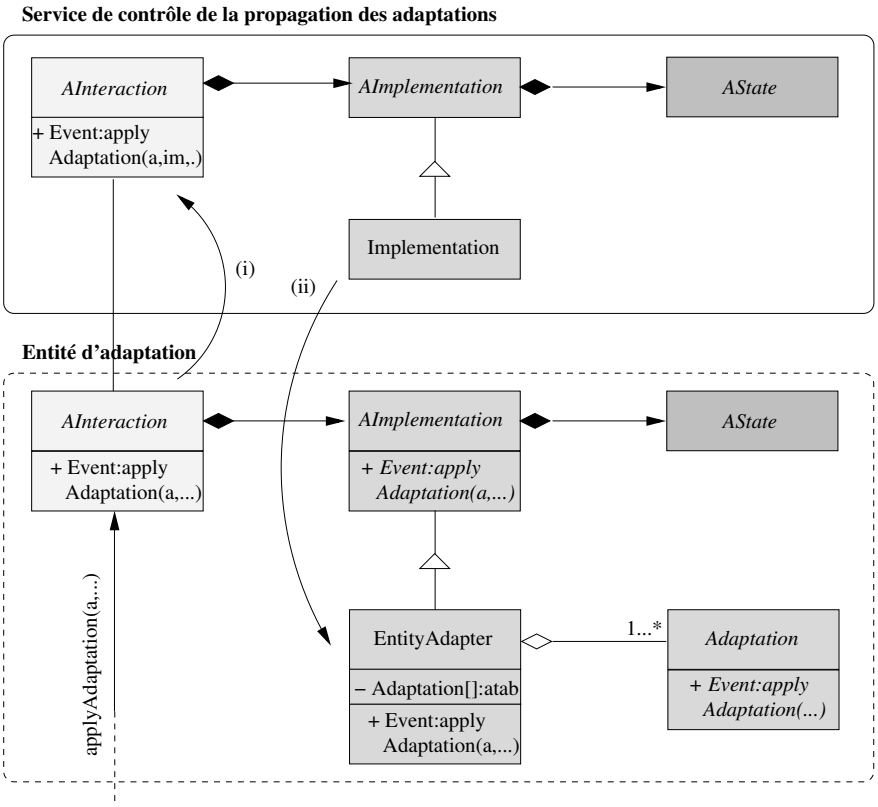


FIG. 6.15 – Relations entre éléments de la structure d'une entité adaptative et service de contrôle de la propagation des adaptations

Dans le cas de stratégies, l'application d'adaptations est la plupart du temps subordon-

née aux caprices des variations de l'environnement mobile. Ces variations peuvent notamment conduire aux effets « boomerang » et « domino ». Ces effets sont d'autant plus catastrophiques pour notre système que certaines adaptations ont un coût important (migration, propagation d'adaptation pour synchronisation, etc).

Pour endiguer ces problèmes nous avons introduit un service de contrôle de la propagation des adaptations (SCPA). Celui-ci est conçu comme un service global, avec des services locaux sur chaque machine, et une politique dans chaque service local. De nombreux algorithmes existent pour la détection de boucles (utilisation d'historiques, de marqueurs, etc) et peuvent être implanté dans la politique de notre SCPA. Nous n'allons pas revenir sur ceux-ci, nous laissons le soin aux administrateurs du système ou aux applications de choisir l'algorithme le plus approprié.

Par contre, plusieurs mécanismes indispensables pour ce contrôle sont détaillés ci-après.

La figure 6.15 présente les relations entre les éléments de la structure d'une entité adaptative et le SCPA. La demande d'application d'une adaptation est interceptée dans l'interaction de l'entité d'adaptation. Cette demande est redirigée vers l'interaction du SCPA (point i de la figure 6.15). Le SCPA décide d'autoriser ou non cette demande. Si la décision est positive, l'appel sera finalement redirigé vers le destinataire original de l'appel (le type EntityAdapter) (point ii de la figure 6.15)

```
abstract class AInteraction { // Interaction de l'entité d'adaptation

   protected AImplementation entityImplementation; // Lien vers l'implantation
   protected AInteraction controlEntityInteraction; // Lien vers le SCPA
   protected AInteraction adaptationEntityInteraction; // Lien vers l'entité d'adaptation
   ...
   public Event applyAdaptation(String adaptationName, String[] args) {
      if (this.getClass().getName()
          .equals("AeDEn.toolkit.service.InteractionSCPA"))
      {
        return(adaptationEntityInteraction.applyAdaptation(adaptationName, args));
      }
      else {
        return(controlEntityInteraction.applyAdaptation(adaptationName,
                                                        entityImplementation, args));
      }
   }
   ...
}
```

FIG. 6.16 – Interception de l'appel `applyAdaptation(a,args[])` de l'entité d'adaptation vers le service de contrôle de la propagation des adaptations

La figure 6.16 détaille la manière dont cette interception est faite. Le principe en est simple. Plutôt que d'être redirigé vers l'implantation de type EntityAdapter (`entityImplementation.applyAdaptation(adaptationName, args)`), l'appel est redirigé vers le SCPA (`controlEntityInteraction.applyAdaptation (adaptationName, entityImplementation, args)`). Cette nouvelle redirection comporte

un paramètre en plus, l'adresse de l'EntityAdapter (`entityImplementation`). Si l'adaptation est acceptée, cette adresse sera utilisée pour retransmettre l'appel (l'`Implementation` du SCPA effectuera un `entityImplementation.applyAdaptation(adaptationName, args)`)

Il existe un seul cas où cette redirection ne peut avoir lieu, si la demande d'adaptation concerne le SCPA lui-même. En effet, lors de la demande d'application d'une adaptation, les appels suivants sont bloqués en attente de la résolution de l'adaptation. Si le SCPA effectue une demande d'adaptation, les appels suivants seront en attente, notamment celui où il se demande s'il s'autorise à s'adapter. Dans la figure 6.16, notre interception prend ce cas en compte en testant si l'interaction courante est celle du SCPA. Si c'est le cas, la demande d'adaptation est directement transmise.

Les effets « boomerang » et « domino » peuvent donc se produire sur le SCPA. Il est donc fortement déconseillé que les stratégies du SCPA dépendent de paramètres hautement variables. Il est plutôt conseillé que ce service soit géré "à la main" par un administrateur.

6.5 Conclusion

Travaux similaires. [Courtrai et al. 2003] présente la plate-forme Concerto qui doit permettre le déploiement et le support de composants parallèles adaptatifs sur une grappe de station. La gestion des ressources présente une hiérarchie des ressources similaire à la notre et un mécanisme de sélection des ressources (`isMatchBy`) similaire à notre méthode de concordance. La notion de "rapport d'observation" présente une nouveauté. En effet, chaque ressource intègre son propre mécanisme de surveillance qui peut être directement appelé (`observe()`) et retourne un rapport d'observation. Par rapport à notre système, le contexte applicatif est le calcul sur grappe de station (*grid computing*) et, donc, aucun prise en compte d'un environnement mobiles n'est disponible. Par ailleurs, dans sa version actuelle, la distribution consiste juste à effectuer un placement statique selon des directives prédéfinies.

[Sourrouille et Contreras 2002] présente le projet ARTO (*Adaptable Real-Time Objects*) dont l'objectif est de réaliser des systèmes capables de modifier dynamiquement leur comportement en fonction des changements intervenant dans leur contexte d'exécution. Comme dans notre système, ARTO comporte des politiques de décision se trouvant à deux niveaux : au niveau local, chaque objet est autonome et a une entière liberté pour sa politique de décision, tandis qu'au niveau global l'efficacité et l'optimisation sont assurées par une gestion centralisée et collective du partage des ressources. Notre système est plus générique puisqu'il ne s'attache pas à des politiques précises et permet, au contraire, d'en changer facilement, alors qu'ARTO possède un algorithme centralisé optimisant la qualité de service à l'aide d'une heuristique favorisant le critère temporel.

[Balan et al. 2003] présente le système Chroma. Celui-ci partitionne automatiquement des applications gourmandes en ressources pour en effectuer une exécution distante depuis un terminal portable. Cette approche présente plusieurs avantages : le partitionnement est réalisé de manière déclarative à l'aide de *tactics* ; le choix de la tactique à utiliser est réalisé à l'exécution en fonction de (i) prévisions sur les ressources demandées, (ii) surveillances des ressources offertes et (iii) une *utility function* fournie par l'utilisateur et reflétant ces préférences. Par rapport

à cette approche, notre système adopte d'autres choix : (i) nous avons choisi une approche ouverte autorisant la spécialisation de tous les éléments de notre système, Chroma se place plutôt comme une approche transparente automatisant au maximum (l'utilisateur peut juste spécifier l'*utility function*), (ii) les hypothèses que nous avons faites sur l'environnement mobile dans le début de ce chapitre sont plus souples que celles imposées par Chroma : l'environnement doit être surapprovisionné en ressources et chaque station doit comporter le code des applications à exécuter (aucun placement ou migration n'est effectuée, mais juste une exécution distante de code déjà présent).

Respects des objectifs. Plusieurs efforts de spécifications ont permis la mise au point (i) d'une caractérisation de l'environnement mobile qui intègre les particularités des ressources et permet l'ajout de nouvelles ressources, (ii) d'une caractérisation de l'application et en particulier de ses besoins et (iii) de notre modèle Offres / Demandes qui assure la correspondance entre les deux premiers points.

À partir de cela, plusieurs services ont été réalisés pour offrir aux applications le meilleur de l'environnement : (i) le service de gestion de l'environnement, (ii) le service de gestion de l'environnement local, (iii) le service de détection et notification, (iv) le service de distribution et (v) le service de contrôle de la propagation des adaptations. Plusieurs abstractions de conception ont été introduites pour la mise en place de ces services : les Ressources et Entités utilisatrices pour le modèle Offres / Demandes ; une Application pour déterminer les entités à distribuer ; les Services globaux, locaux et les Politiques adaptatives pour la gestion de notre système. Celles-ci ont toutes spécialisées les entités adaptatives.

Afin d'appliquer les algorithmes les plus appropriés aux conditions de l'environnement, nous avons notamment mis en place les mécanismes qui permettent de changer dynamiquement de politique tout en conservant leur cohérence. Quelques exemples de stratégies faisant intervenir différents algorithmes ont été donnés. La stabilité de ces adaptations est garantie par le service de contrôle de la propagation des adaptations. Il est alors le seul élément pouvant être instable. Ce défaut peut être relativisé car ce service ne nécessite pas souvent d'être adapté et dans ce cas une utilisation éclairée permet de régler ce problème.

Troisième partie

Le système AeDEn

Chapitre 7

Mise en œuvre et expérimentations

Nous avons présenté, dans la seconde partie de ce document, la conception d'un système de gestion des environnements mobiles se basant sur un découpage en cadre de conception et boîte à outils. Notre système s'appuie principalement sur les fonctionnalités (i) d'adaptation et de réaction dynamique, (ii) de gestion des ressources de l'environnement et (iii) de distribution des applications. Nous avons réalisé un prototype de ce système, baptisé AeDEn (*Adaptive Distribution Environment*), et mené quelques expérimentations pour valider notre approche [Le Mouël et al. 2002].

Ce chapitre se divise en deux parties. Dans la première partie (paragraphe 7.1), nous détaillons la mise en œuvre d'AeDEn avec, plus particulièrement, l'utilisation du système MolèNE (paragraphe 7.1.1). La deuxième partie de ce chapitre (paragraphe 7.2) relate les expérimentations d'AeDEn à l'aide d'une application de type navigateur (paragraphe 7.2.1). Enfin, nous dressons le bilan de notre système.

7.1 Mise en œuvre d'AeDEn

Afin de construire notre système, nous avons répertorié les systèmes présentant des abstractions de conception intéressantes. Notre choix s'est porté sur le système MolèNE parce qu'il présente un modèle de composants et un modèle de communication adaptés aux environnements mobiles. Le paragraphe 7.1.1 présente la structure de MolèNE. Le paragraphe 7.1.2 présente le modèle de composants, et, plus particulièrement la manière dont nous avons fait concorder le modèle de composants adaptatifs avec notre modèle d'entités adaptatives. Le paragraphe 7.1.3 présente le modèle de communication par évènements. Enfin, Le paragraphe 7.1.4 donne l'exemple de mise en œuvre d'un service, le service de gestion de l'environnement local (SGEL).

7.1.1 Préliminaires MolèNE

MolèNE (*MObiLE Networking Environment*) [Segarra 2000] est mis en œuvre en utilisant le langage de programmation à objets Java [Sun 2003]. Il présente donc la simplicité de programmation du langage qui facilite la construction rapide de prototypes pour valider des

concepts et, d'autre part les nombreuses possibilités de l'environnement Java aussi bien au niveau du langage qu'au niveau des bibliothèques de programmation (réflexion, processus légers, synchronisations).

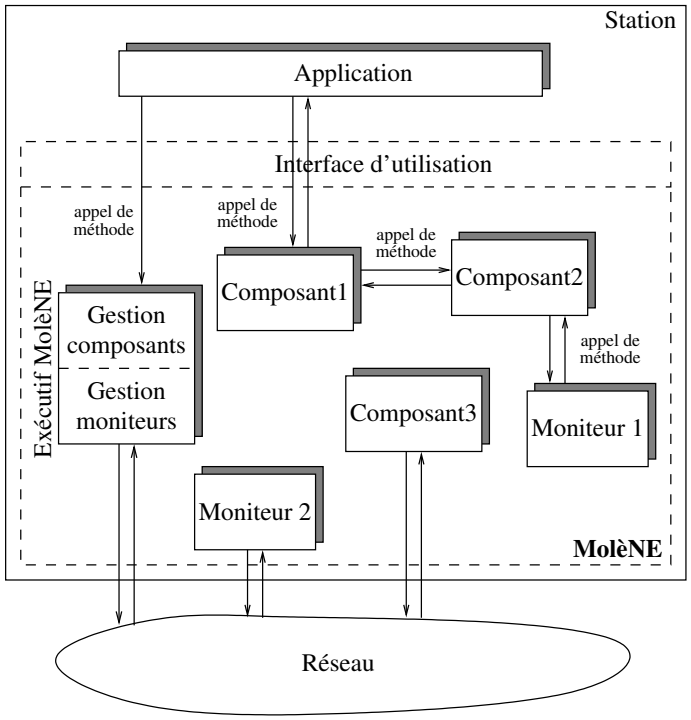


FIG. 7.1 – Structure de MolèNE

La gestion de l'environnement sans fil est effectuée dans MolèNE par des moniteurs de surveillance et des composants qui réagissent aux variations. Ces éléments sont mis en œuvre par des classes dont certaines sont laissées abstraites afin de permettre la spécialisation de MolèNE pour une application précise. Cette spécialisation peut être effectuée en utilisant des classes concrètes fournies par MolèNE ou bien en créant de nouvelles classes. Lorsque cette spécialisation est effectuée, les classes concernées peuvent être instanciées et leur exécution réalisée par la JVM (*Java Virtual Machine*) des stations impliquées.

La figure 7.1 représente un instant `t` de l'exécution de MolèNE sur une station. Chaque composant est mis en œuvre par un ensemble d'objets qui matérialisent sa structure fonctionnelle (voir paragraphe suivant pour les détails). Les moniteurs sont des instances d'une classe qui définit leur structure générale. Tous ces objets peuvent être instanciés par une application en utilisant l'interface offerte par l'exécutif MolèNE à cette fin. Celui-ci constitue le *"point d'entrée"* d'une application à MolèNE et offre des services de base nécessaires à l'exécution des composants et des moniteurs instanciés.

7.1.2 Définition du composant adaptatif MolèNE par spécialisation d'entité adaptative

Les fonctionnalités de MolèNE sont conçues de manière très modulaire en identifiant les diverses sous-fonctionnalités qu'elles fournissent et en implantant chacune d'elles par un *composant*.

L'architecture fonctionnelle d'un composant est guidée par une distinction entre les aspects interaction avec d'autres composants et les aspects fonction du composant elle même. Ces deux aspects ont été réifiés et ils sont représentés dans un composant par deux types d'objet différents, leur relation étant définie par le patron de conception *Strategy* décrit dans [Gamma et al. 1995]. Le type *MoleNEComponentInteraction* reçoit les demandes de service d'autres composants. Le type *MoleNEComponentImplementation* s'occupe de fournir le traitement correspondant au composant. La figure 7.2 illustre ces classes et leur relation. Le traitement réalisé par un composant peut ainsi être spécialisé par héritage de la classe *MoleNEComponentImplementation*.

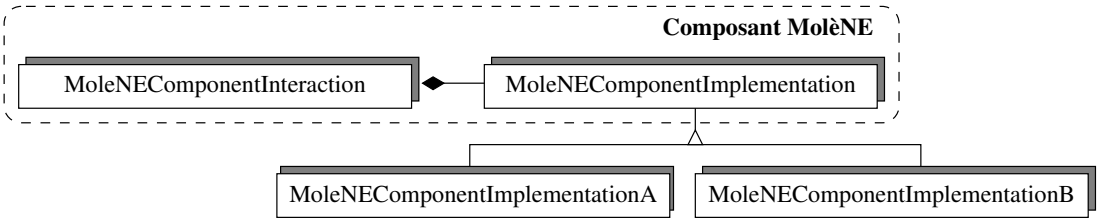


FIG. 7.2 – Structure fonctionnelle d'un composant MolèNE

Le composant représente l'unité logicielle minimale dont le comportement peut être rendu adaptatif. Le composant adaptatif représente l'unité logicielle minimale dont le comportement peut être adapté par l'application de réactions de changement de paramètres ou de réactions de remplacement de comportement.

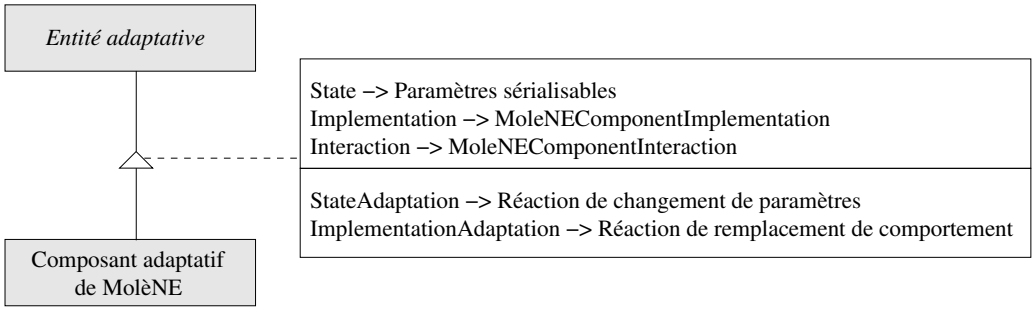


FIG. 7.3 – Modèle composant adaptatif MolèNE par spécialisation d'entité adaptative

De même que dans le paragraphe 5.3.3, pour pouvoir utiliser les composants adaptatifs de MolèNE comme abstraction de conception dans notre système, nous les avons définis comme

étant une spécialisation du modèle d'entités adaptatives. La figure 7.3 présente cette spécialisation.

L'état de l'entité fonctionnelle est composé des paramètres caractérisant la fonctionnalité attendue. En vue de la distribution, cet état est sérialisable. L'implantation de l'entité fonctionnelle correspond à la classe *MoleNEComponentImplementation*. L'interaction de l'entité fonctionnelle correspond à la classe *MoleNEComponentInteraction*. Les adaptations applicables sont de type *StateAdaptation* avec les réactions de changement de paramètres et de type *ImplementationAdaptation* avec les réactions de remplacement de comportement.

7.1.3 Communication par évènement

De nombreuses interactions ont lieu au sein de MolèNE. Ainsi, l'exécutif MolèNE interagit avec des composants et des moniteurs, les moniteurs interagissent entre eux et avec les composants et des interactions peuvent avoir lieu entre ces derniers. Dans tous les cas, le paradigme de communication utilisé est de type client/serveur. Des clients effectuent des demandes de service à des serveurs qui répondent à ces demandes. Demandes et réponses sont encapsulées dans MolèNE dans des objets de type *Event* et sont livrées à leur destinataire soit par un appel de méthode, soit en utilisant un composant de gestion de communications distantes. L'ensemble des classes disponibles pour la communication dans MolèNE est représenté dans la figure 7.4. Deux classes héritent de la classe mère *Event*. Un objet de type *RequestEvent* encapsule une demande de service tandis qu'un objet de type *ReplyEvent* correspond à une réponse.

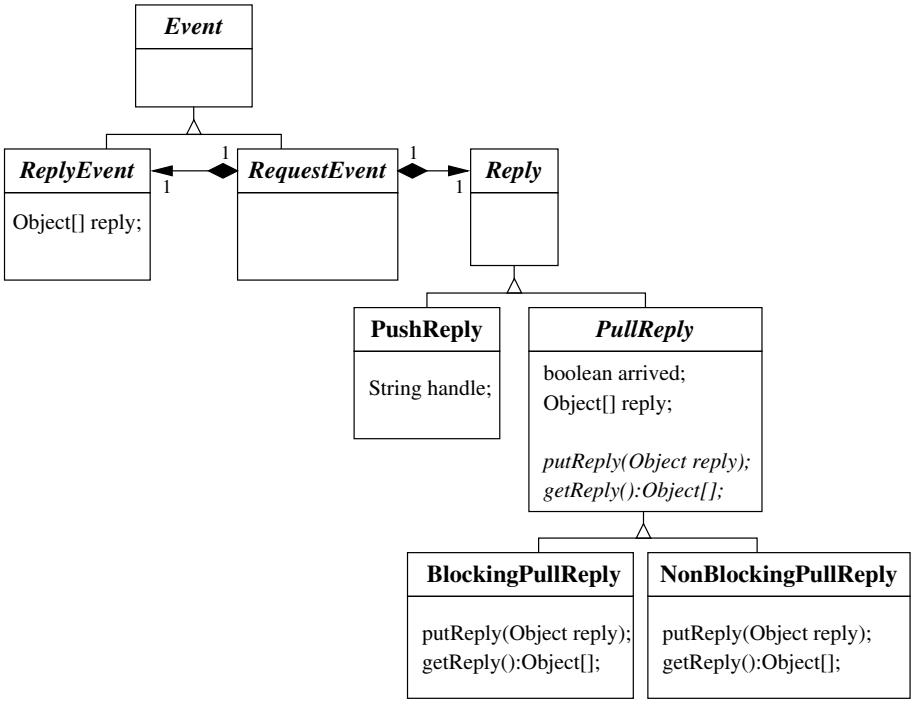


FIG. 7.4 – Hiérarchie de classes pour la communication dans MolèNE

Une entité (exécutif MolèNE, composant ou moniteur) peut obtenir la réponse à sa demande de service soit par notification asynchrone, soit par consultation. Une instance de la classe *Reply* doit être associée par l'entité à son objet de type *RequestEvent*. Lorsqu'il s'agit d'un *PushReply*, l'entité est notifiée de manière asynchrone de l'arrivée de la réponse. Lorsqu'il s'agit d'un *PullReply*, l'entité doit utiliser la méthode `getReply` qui la bloque (*BlockingPullReply*) ou non (*NonBlockingPullReply*) si la réponse n'est pas disponible.

7.1.4 Exemple : mise en œuvre du SGEL

Il serait fastidieux de décrire la mise en œuvre de tous les services et politiques de notre système. Aussi, nous détaillons ci-dessous la mise en œuvre d'un seul service, les autres étant construits pareillement. Nous avons choisi le service de gestion de l'environnement local (SGEL) car il présente une fonction initiatrice des ressources que ne possèdent pas les autres services. Le paragraphe 7.1.4.1 présente la structure du service et les messages échangés avec les politiques. Les paragraphes 7.1.4.2 et 7.1.4.3 décrivent, respectivement, la politique d'initialisation et la politique d'enregistrement.

7.1.4.1 Structure et messages échangés

Le service de gestion de l'environnement local, que nous présentons dans la figure 7.5, se compose de :

(i) l'environnement local, qui maintient une base de données de tous les éléments locaux (ressources et entités utilisatrices),

(ii) la politique d'initialisation, qui initie l'environnement local,

(iii) la politique d'enregistrement, qui décide de l'enregistrement des éléments locaux auprès de l'environnement global.

La communication au sein du SGEL et avec les autres services se déroule par envois et réceptions d'évènements :

(i) *InitParsingRequestEvent* : message envoyé à la politique d'initialisation de l'environnement local pour lancer l'initialisation de l'environnement local. Le message prend en paramètre un fichier contenant l'ensemble des éléments à ajouter dans l'environnement local.

(ii) *AddElementRequestEvent* : message envoyé à l'environnement local et à l'environnement global. Il permet l'ajout de nouveaux éléments, et prend en paramètres l'ensemble des éléments à ajouter.

(iii) *RemoveElementRequestEvent* : message envoyé à l'environnement local et à l'environnement global. Il permet le retrait des éléments, et prend en paramètre une liste des noms des éléments à retirer.

(iv) *GetElementRequestEvent* : ce message permet la demande d'un ensemble d'éléments auprès de l'environnement local ou global. Il prend en paramètre une liste des noms des éléments souhaités.

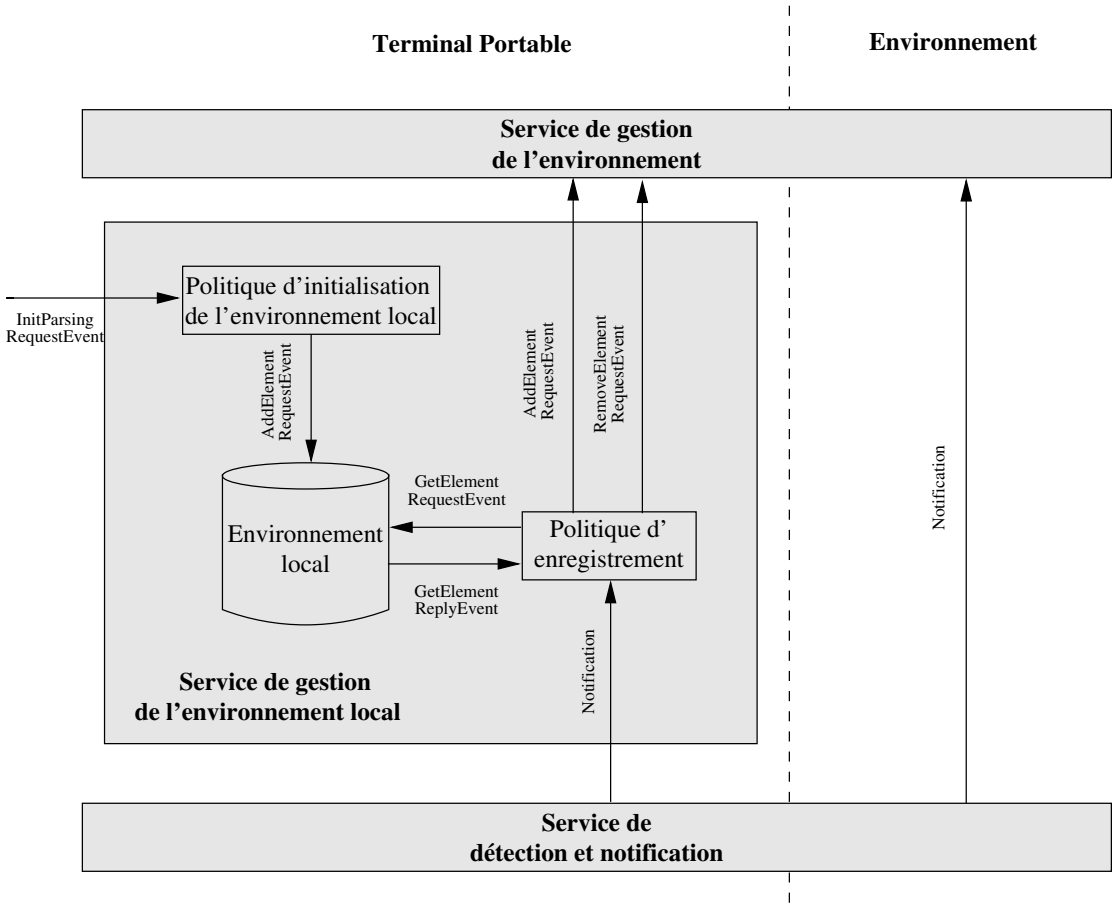


FIG. 7.5 – Communication au sein du service de gestion de l'environnement local

(v) *GetElementReplyEvent* : ce message est la réponse au message de demande des éléments *GetElementRequestEvent*. Il contient l'ensemble des éléments demandés.

(vi) *Notification* : les messages de notifications sont envoyés par le service de détection et notification auprès des services ou politiques ayant préalablement enregistrés un intérêt de surveillance sur un élément de l'environnement. Chaque message de notification contient un paramètre et sa valeur courante (exemple : probabilité de déconnexion = 0.5).

7.1.4.2 Politique d'initialisation

La politique d'initialisation de l'environnement local est conçue sous forme d'un composant adaptatif MolèNE. L'implantation du composant contient l'implantation d'un compilateur. Ce dernier prend en entrée la spécification d'un ensemble d'éléments ressources et entités utilisatrices, et génère des requêtes d'ajout des éléments dans l'environnement local. Les requêtes sont envoyées par l'intermédiaire de la partie *Interaction* du composant.

La figure 7.6 montre les interactions entre les différents objets contenus dans le composant d'initialisation de l'environnement local : les objets *Interaction* et *Implementation* représentent, respectivement, la partie interaction et implantation du composant. L'objet *Application* représente une application utilisatrice de la politique d'initialisation de l'environnement local.

Un message de type *AddElementRequestEvent*, contenant le fichier de description des éléments de l'environnement local, est envoyé par l'application *Application* à l'objet *Interaction*. Le fichier est passé à l'objet *Implementation*, celui-ci active le compilateur qui interprète le fichier. L'objet *Implementation* récupère les éléments locaux et génère un message de type *AddElementRequestEvent*, le message est ensuite envoyé par l'intermédiaire de l'objet *Interaction*.

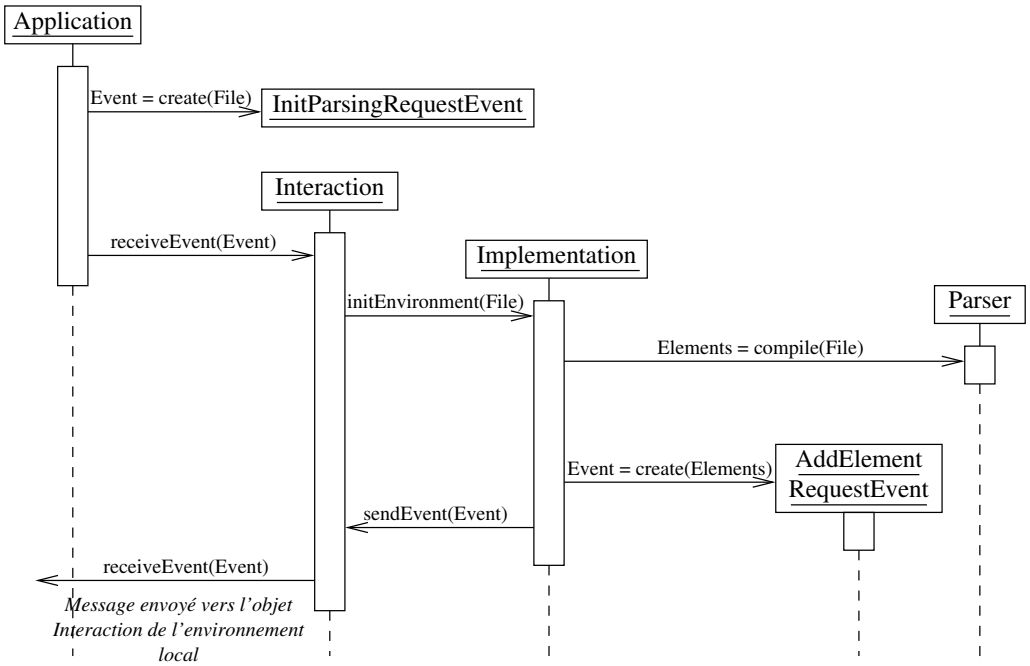


FIG. 7.6 – Interactions au sein de la politique d'initialisation de l'environnement local

Le compilateur que nous avons intégré au niveau de la politique d'initialisation de l'environnement local est JavaCC [BSD]. Il implémente une grammaire pour la description des éléments de l'environnement local (voir l'annexe A pour les détails). D'autres compilateurs, comme ceux pour RDF (*Resource Description Framework*) [W3C 1999b] ou WSDL (*Web Services Description Language*) [Chinnici et al. 2003], pourraient être intégrés. Le fait que la politique d'initialisation soit implantée par un composant adaptatif permettrait de passer d'un compilateur à un autre selon le type du fichier considéré en entrée.

7.1.4.3 Politique d'enregistrement

La politique d'enregistrement est implantée au niveau d'un composant adaptatif (figure 7.7). Le composant met en œuvre trois implantations : une implantation pour la mise à disposition totale des éléments, une implantation pour la mise à disposition partielle des éléments et une implantation où aucun des éléments n'est déclaré.

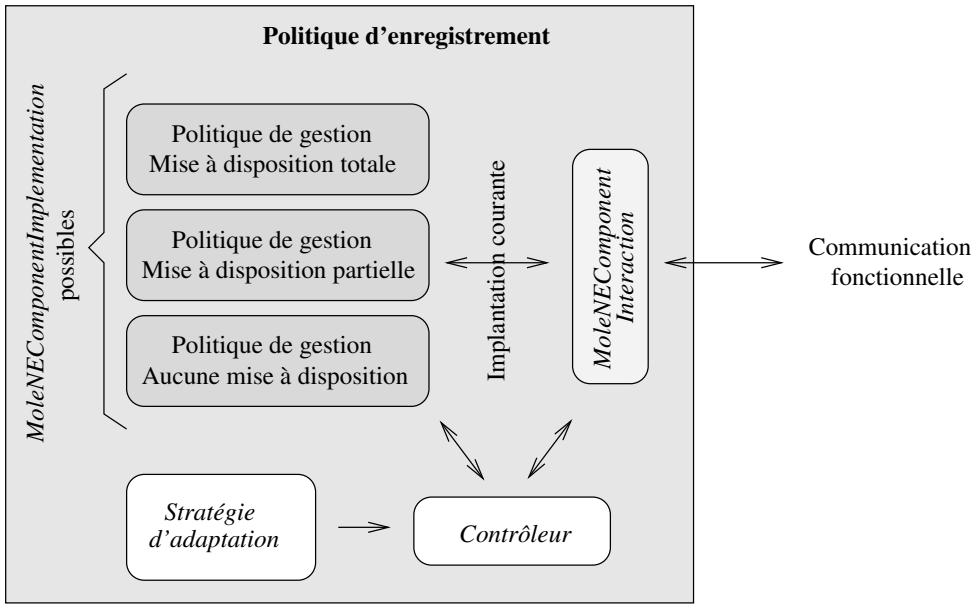


FIG. 7.7 – Politique d'enregistrement mis en œuvre par un composant adaptatif MolèNE

Le composant contient un objet *Interaction* qui se charge de la communication avec les autres composants du service de gestion de l'environnement local. La communication du composant de gestion de l'environnement local se fait avec l'environnement local et service de gestion de l'environnement. Le composant de gestion de l'environnement local contient, en outre, un contrôleur qui s'occupe de la mise en place des différentes implantations selon les variations de l'état de l'environnement. Pour ce faire, le contrôleur utilise une stratégie d'adaptation qui définit les transitions entre les implantations. La stratégie d'adaptation que nous utilisons dans le composant de gestion de l'environnement local est celle exposée dans la figure 6.11 du paragraphe 6.4.2.2.

La mise à disposition totale des éléments Cette implantation est exécutée lorsque la batterie est chargée. Pour effectuer l'enregistrement de tous les éléments locaux, deux tâches sont exécutées : (i) l'acquisition de tous les éléments non enregistrés dans l'environnement global, (ii) l'envoi de messages pour l'enregistrement de tous ces éléments. La figure 7.8 montre l'interaction entre les différents objets constituant le composant de politique d'enregistrement, lorsque l'implantation courante correspond à la mise à disposition de tous les éléments locaux.

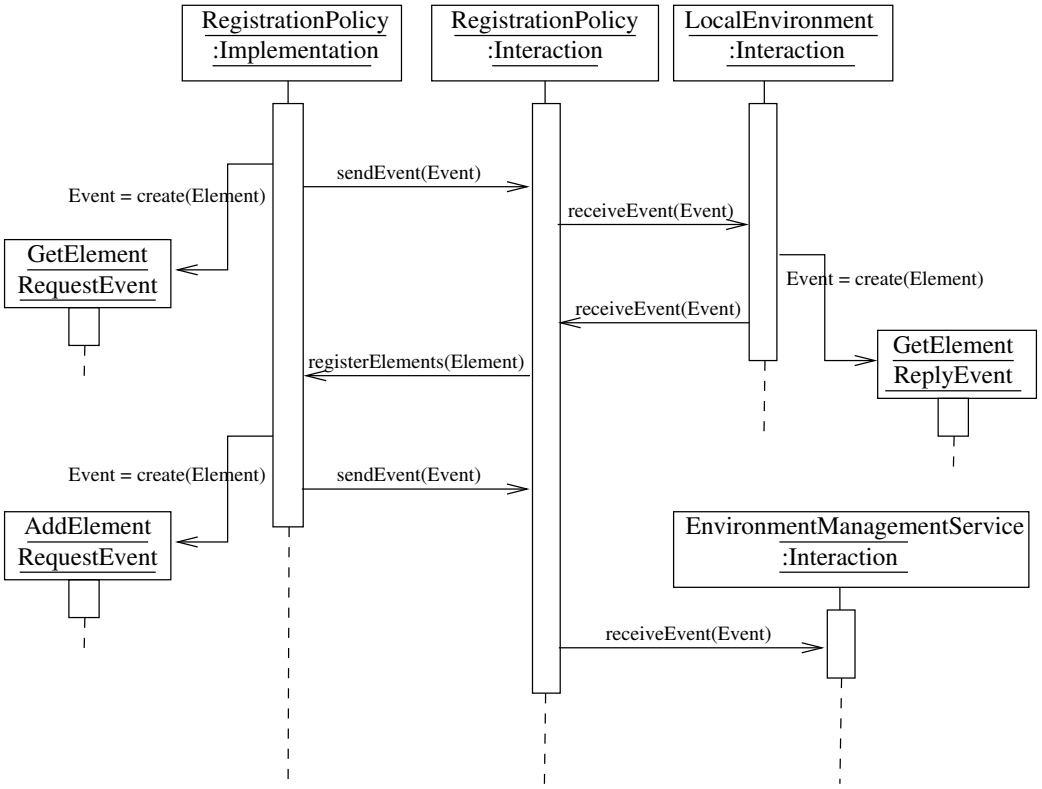


FIG. 7.8 – Interactions au sein de la politique d'enregistrement lors d'une mise à disposition totale des éléments

L'acquisition des éléments locaux se fait par envoi d'un message de type *GetElementRequestEvent*, le message est envoyé par l'objet *Interaction* du composant de gestion de l'environnement local vers l'objet *Interaction* de l'environnement local. La réponse à ce message est renvoyée par un message de type *GetElementReplyEvent*, ce message contient l'ensemble des éléments locaux. Ensuite, l'enregistrement des éléments locaux se fait par envoi d'un message de type *AddElementRequestEvent* vers l'objet *Interaction* du service de gestion de l'environnement global.

La mise à disposition partielle des éléments Le composant de gestion de l'environnement local procède à l'enregistrement partiel des éléments locaux, lorsque la charge de la batterie baisse mais n'est pas trop faible. La partie des éléments à déclarer ou retirer est prise en paramètre par l'implantation. Deux cas se présentent, si tous les éléments locaux sont enregistrés au niveau global (mise à disposition totale des éléments), le retrait d'une partie des éléments est effectué, sinon, si aucun élément n'est déclaré (aucune mise à disposition), alors une partie des éléments est mise à disposition. Les deux cas sont illustrés dans la figure 7.9.

Dans une première phase, le composant de gestion de l'environnement local envoie un message de type *GetElementRequestEvent*, pour l'acquisition des éléments enregistrés dans l'en-

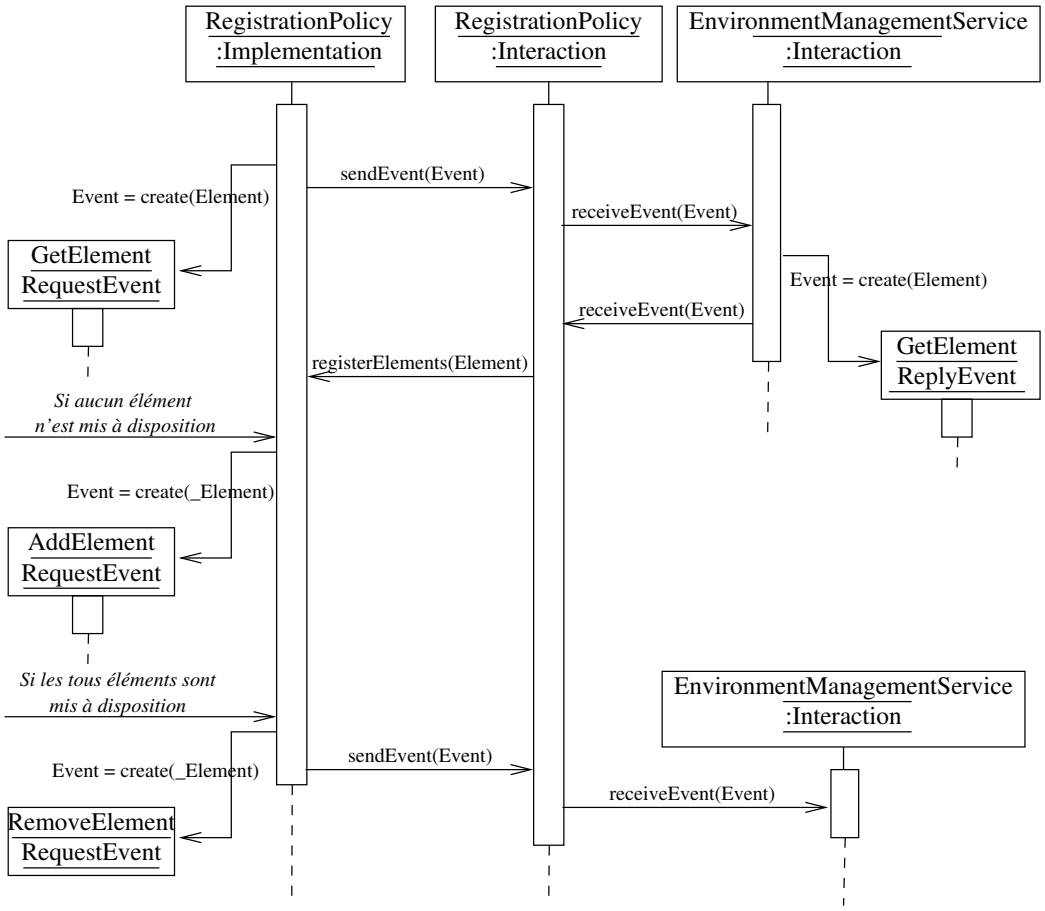


FIG. 7.9 – Interactions au sein de la politique d'enregistrement lors d'une mise à disposition partielle des éléments

vironnement global. La réponse est reçue sous forme d'un message *GetElementReplyEvent*. Si tous les éléments sont enregistrés, le composant envoi un message de retrait d'une partie des éléments *RemoveElementRequestEvent*, sinon, un message d'ajout d'une partie des éléments *AddElementRequestEvent* est envoyé.

Aucune mise à disposition Le retrait de tous les éléments se fait lorsque la batterie est faible. Le composant de gestion de l'environnement local envoie un message de type *GetElementRequestEvent* vers l'environnement local, pour l'acquisition de tous les éléments. Ensuite, un message de suppression de tous les éléments enregistrés dans l'environnement global *RemoveElementRequestEvent* est envoyé par l'intermédiaire de l'objet *Interaction* du composant de gestion de l'environnement local vers l'objet *Interaction* du service de gestion de l'environnement. La figure 7.10 présente cette fonctionnalité.

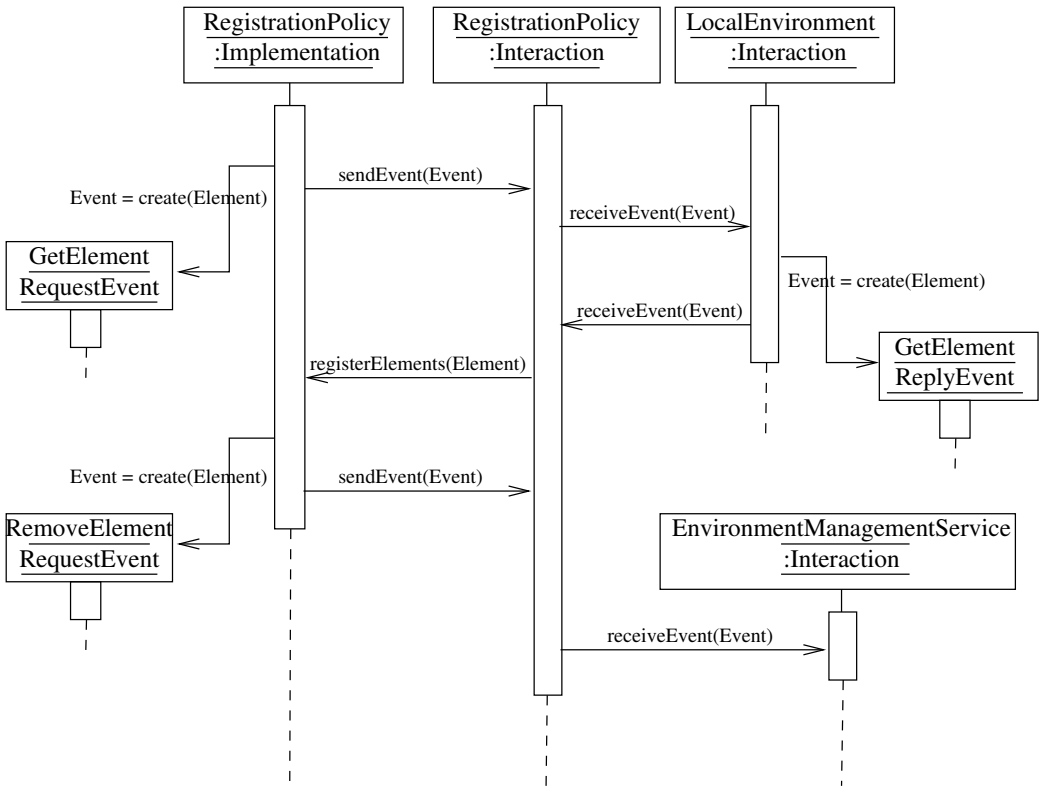


FIG. 7.10 – Interactions au sein de la politique d'enregistrement lors d'une mise à disposition nulle des éléments

7.2 Expérimentations d'utilisation d'AeDEn

La deuxième partie de ce chapitre relate les expérimentations d'AeDEn à l'aide d'une application de type navigateur (paragraphe 7.2.1). L'impact de la distribution est mesuré sur les performances de l'application (paragraphe 7.2.2) et sur les utilisations des ressources propres à l'environnement mobile (paragraphe 7.2.3).

7.2.1 Navigateur pour environnement mobile

Pour tester notre système, nous avons réalisé une application communément utilisée dans les environnements mobiles : un navigateur transformant les pages HTML pour qu'elles soient conformes aux capacités d'affichage du terminal portable considéré. Nous avons décomposé cette application en trois composants :

(i) Un composant de chargement (CC dans la figure 7.11) qui récupère les pages HTML et tous les fichiers associés.

(ii) Un composant de transformation (CT dans la figure 7.11) qui est en charge de modifier les pages HTML pour qu'elles puissent s'afficher sur le terminal portable.

(iii) Un composant d'affichage (CA dans la figure 7.11) qui est en charge d'afficher les pages transformées.

Dans notre test, le composant CC récupère les pages HTML avec toutes leurs images et le composant CT remplace ces dernières par le texte de l'attribut `ALT` des balises `IMG` contenues dans les pages[1]. Chacun de ces composants a des besoins particulier : le composant CC demande un accès Internet, le composant CT requière un minimum de puissance de calcul et le composant CA nécessite un écran. Pour cela, ces composants résultent d'une spécialisation d'entités utilisatrices permettant alors d'exprimer leurs besoins en terme de demandes de type *Demand*.

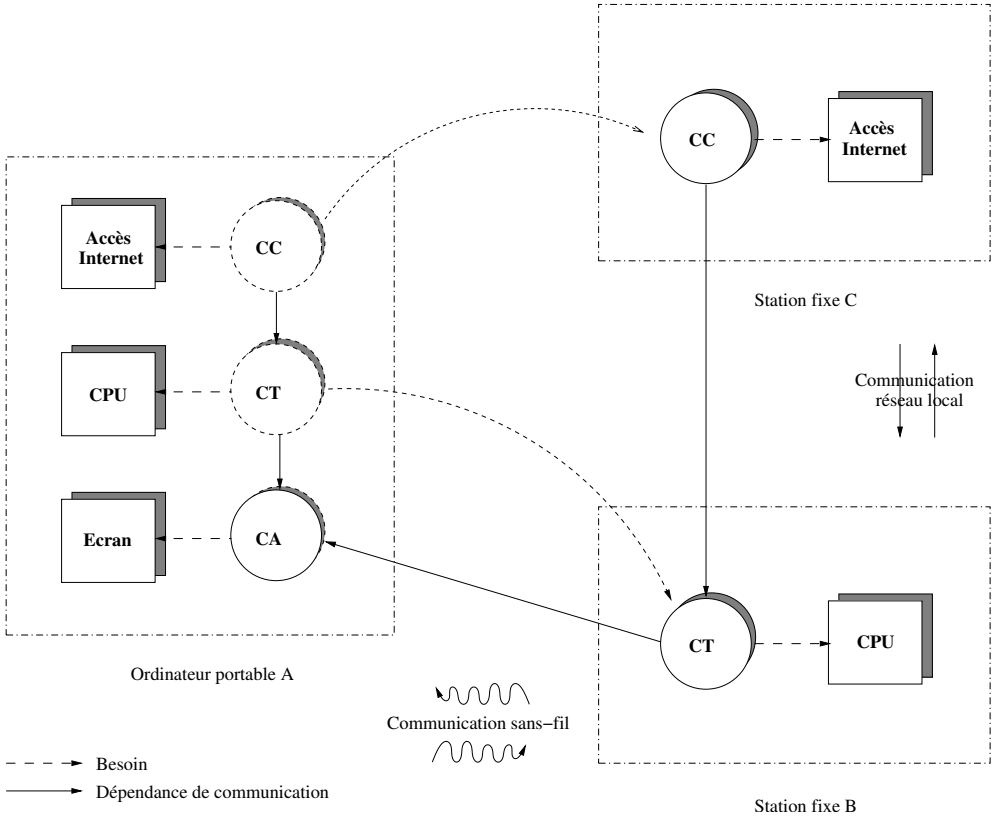


FIG. 7.11 – Placement des composants du navigateur

Trois stations constituent l'environnement d'exécution pour notre application : un ordinateur portable (A) offre un écran, de la puissance de calcul et un accès Internet ; une station fixe (B) offre de la puissance de calcul et une station fixe (C) offre un accès Internet. Dans notre test, l'ordinateur portable (A) est un IBM ThinkPad avec un processeur à 166 MHz, connecté au réseau local par une liaison Ethernet à 10 Mb/s quand l'ordinateur est statique (sur sa base) et au travers d'un réseau sans-fil WaveLan à 1.6 Mb/s quand l'ordinateur est mobile. Ces deux

[1] Il est, bien évidemment, facile d'imaginer des transformations plus complexes comme de la dégradation d'images, etc.

types de connexion autorisent l'accès à Internet. La station fixe (B) est une Ultra Sparc 60 avec deux processeurs à 360 MHz et la station fixe (B) fournit un accès Internet par le biais d'un serveur cache (*proxy*) se trouvant sur le même réseau local que les deux stations fixes.

Pour découvrir les bénéfices apportés par le distribution et plus particulièrement par les politiques adaptatives, nous avons testé simultanément trois politiques adaptatives :

(i) La politique de détection et notification des variations (PDNV) **:** la PDNV présente deux algorithmes : (a) un algorithme immédiat (PIDNV), qui notifie immédiatement lorsqu'une variation arrive, et (b) un algorithme périodique (PPDNV), qui notifie périodiquement des variations. La condition de changement d'algorithme est le passage d'un réseau de type Ethernet à un réseau de type WaveLan.

(ii) La politique d'enregistrement (PEn) **:** la PEn présente deux algorithmes : (a) un algorithme qui enregistre les ressources du terminal portable dans l'environnement (PEnRL : Ressources Locales), et (b) un algorithme qui n'enregistre aucune ressource dans l'environnement (PEnNR : No Ressource). La condition de changement d'algorithme est le niveau de charge de la batterie.

(iii) La politique de placement (PP) **:** la PP présente deux algorithmes : (a) un algorithme de placement qui utilise uniquement les ressources locales (PPRL), et (b) un algorithme de placement utilisant toutes les ressources de l'environnement (PPE). Le changement pour l'algorithme PPE s'effectue quand il n'y a plus de ressources locales disponibles.

En premier lieu, au lancement du navigateur, la PEn implante l'algorithme PEnRL et la PP implante l'algorithme PPRL. Ainsi, les trois composants de l'application sont placés sur l'ordinateur portable (A) (voir figure 7.11). Ensuite, quand le niveau de charge de la batterie atteint un certain seuil, la PEn change d'algorithme pour le PPL et retire donc de l'environnement les ressources appartenant à l'ordinateur portable (A). La PP change alors d'algorithme pour adopter l'algorithme PPE, et décide consécutivement de placer le composant CC sur la station fixe (C) et le composant CT sur la station fixe (B). Pendant le même temps, quand l'ordinateur portable est statique, la PDNV implante l'algorithme PIDNV, et, quand l'ordinateur portable est mobile, la PDNV implante l'algorithme PPDNV.

À partir de cette expérience, nous allons analyser l'impact de la distribution et des adaptations des politiques en terme de performances de l'application et en terme d'utilisation des ressources de l'ordinateur portable.

7.2.2 Résultats des performances de l'application

Les performances de l'application ont été réalisées pour le temps d'exécution (paragraphe 7.2.2.1), la quantité de données transférées sur le lien sans-fil (paragraphe 7.2.2.2) et l'impact de la migration (paragraphe 7.2.2.3).

7.2.2.1 Temps d'exécution

Le tableau 7.1 recense les temps d'exécution des composants CC et CT selon la station où ils s'exécutent. Ces temps ont été calculés comme la moyenne des temps d'exécution de

dix requêtes effectuées sur des sites Web populaires[2]. Le temps d'exécution du chargement est diminué de 50 à 60% lorsque le composant CC s'exécute sur la station (C). Ceci s'explique par le fait de ne plus subir le goulot d'étranglement du lien sans-fil et d'utiliser le cache réseau. Le temps d'exécution de la transformation est diminué de 40 à 50% lorsque le composant CT s'exécute sur la station (B). Ceci s'explique par l'utilisation d'un processeur plus performant.

	CC (A) (ms)	CC (C) (ms)	Speed Up %	CT (A) (ms)	CT (B) (ms)	Speed Up %
1 - Microsoft	3648	1722	52.8	966	564	41.7
2 - AOL	6567	2561	61.0	1841	1064	42.2
3 - Yahoo	2677	1283	52.1	720	414	42.5
4 - Lycos	4930	2863	41.9	1206	694	42.4
...	...	...	...	...	...	...
8 - NBC	5548	2613	52.9	1152	684	40.6
11 - Disney	3813	1853	51.4	927	541	41.7
15 - Real	5232	2378	54.5	1384	810	41.5
17 - Fortune City	6041	2417	60.0	1782	1018	42.9

TAB. 7.1 – Mesures du temps d'exécution des composants du navigateur selon leur station d'exécution

7.2.2.2 Quantité de données transférées sur le lien sans-fil

Comme le montre le tableau 7.2, la quantité de données transférées au travers du lien sans-fil est aussi diminuée. Ceci s'explique par le fait que seules les pages transformées sont transférées au composant CA. Ces résultats dépendent évidemment de la transformation appliquée et de la quantité de données à transformer (ici, la taille des images contenues dans les pages HTML). Pour la plupart des sites utilisés, la quantité de donnée est réduite de 40 à 60%, mais, pour les sites riches en images, cette réduction peut atteindre 70 à 80%.

7.2.2.3 Impact de la migration

Nous avons également mesuré l'impact de la migration sur les performances de l'application. Pour cette expérience, nous avons utilisé la sérialisation Java. Le temps de migration se décompose selon le temps de transfert et le temps de restauration d'un composant. Le tableau 7.3 montre ces deux temps pour les composants CC et CT suivant qu'ils doivent traiter une petite quantité de données ↘ (page HTML Yahoo : 28 Kb) ou une quantité importante de données ↗ (page HTML Fortune City : 132 Kb). Comme l'indique ces résultats,

[2]Extrait du "Global Top 50 Web" publié par Jupiter Media Metrix en avril 2001.

	CC (A) CT (A) Html + Img (Kbyte)	CC (C) CT (B) Html (Kbyte)	%
1 - Microsoft	22.9 + 17.3 = 40.2	20.1	50.0
2 - AOL	43.6 + 27.0 = 70.6	37.6	46.7
3 - Yahoo	16.0 + 12.0 = 28.0	15.6	44.3
4 - Lycos	26.7 + 13.5 = 40.2	25.6	36.3
…	…	…	…
8 - NBC	29.1 + 66.4 = 95.5	23.2	75.7
11 - Disney	22.6 + 78.6 = 101.2	18.3	81.9
15 - Real	32.6 + 71.8 = 104.4	29.5	71.7
17 - Fortune City	41.8 + 90.8 = 132.6	35.8	73.0

TAB. 7.2 – Mesures des données transférées sur le lien sans fil

	Temps de transfert (ms)	Temps de restauration (ms)	Équilibrage migrations (> nombre de requêtes)
CC (A → C) + Données ↘	5437	165	4
CC (A → C) + Données ↗	12920	178	3.6
CT (A → B) + Données ↘	2616	213	9.3
CT (A → B) + Données ↗	4690	236	6.4

TAB. 7.3 – Mesures de l'impact de la migration sur les performances du navigateur

le temps de restauration d'un composant est constant, tandis que le temps de transfert dépend grandement des données à sérialiser. Ces mesures nous permettent de calculer le nombre de requêtes nécessaires pour contrebalancer le coût de la migration ($CC(A) \times Nb_{requêtes} > CC(C) \times Nb_{requêtes} + Temps\ de\ transfert + Temps\ de\ restoration$). Ce calcul est indiqué dans la dernière colonne du tableau : ce nombre varie en fonction du composant considéré, 4 requêtes pour le composant CC et de 7 à 10 requêtes pour le composant CT. Ce nombre peut être utilisé dans les stratégies pour décider s'il y lieu d'appliquer les adaptations de migration.

Ces résultats montrent l'intérêt de distribuer les composants d'une application suivant de meilleurs ressources disponibles dans l'environnement mobile. Ce placement permet d'améliorer le temps d'exécution et la quantité de données transférées sur le lien sans-fil (sous réserve que l'application ait plus de données à traiter qu'elle n'en ait à afficher).

7.2.3 Résultats des utilisations des ressources propres à l'environnement mobile

Un autre but recherché par la distribution et les politiques adaptatives est d'économiser les ressources des terminaux portables. Pour cela, nous avons mesuré leur incidence sur le niveau de charge de la batterie (paragraphe 7.2.3.1) et sur le débit de la bande passante (paragraphe 7.2.3.2).

7.2.3.1 Niveau de charge de la batterie

La figure 7.12 présente la consommation de la batterie. Nous avons tout d'abord établi des taux de décharge de référence en fonction du mode d'utilisation (*Suspend, Stand by, Running*) et en fonction des ressources locales utilisées (écran, processeur, mémoire, carte réseau). Quand le navigateur s'exécute "à plein régime" sur l'ordinateur portable, la courbe représentant l'exécution de notre application à la même pente (-1,5) que celle représentant l'utilisation de plusieurs ressources simultanées. Quand le niveau de charge de la batterie atteint 40%, la PEn change l'algorithme PEnRL pour l'algorithme PEnNR, et décide alors de retirer ses ressources partagées. La PP change alors d'algorithme pour adopter l'algorithme PPE, et décide consécutivement de placer les composant CC et CT dans l'environnement. Ce changement se reflète dans la deuxième partie de la courbe. Sa pente (-0.84) est alors presque la même que celle de la courbe reflétant une utilisation uniquement du système et de l'écran (-0.94). Ce changement se traduit par une augmentation de la durée de vie de la batterie de 30% (88 mn au lieu de 64 mn).

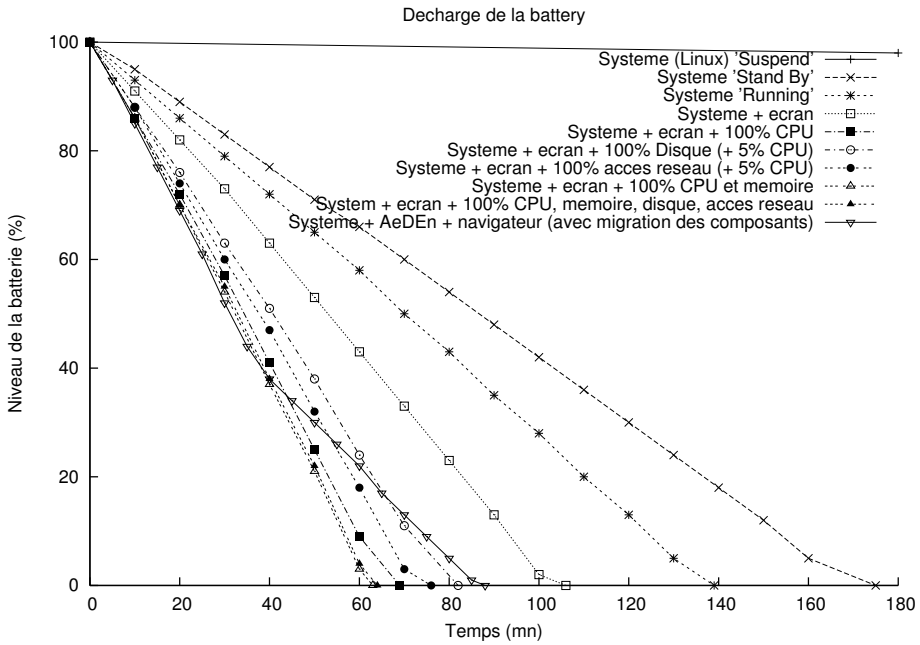


FIG. 7.12 – Décharge de la batterie selon son utilisation

7.2.3.2 Débit de la bande passante

Le débit de la bande passante a également été mesuré dans notre expérience. Pour cela, nous avons accru le trafic sur le lien de communication, en simulant une variation toutes les 100ms et en envoyant sa notification au service global de gestion de l'environnement (qui est distribué). Ce surcoût permet de démontrer l'intérêt d'avoir des algorithmes différents de notification selon la nature du lien considéré.

Quand l'ordinateur portable est statique et connecté avec le lien Ethernet, l'algorithme PIDNV est utilisé. Dû à notre simulation de variations, nous avons alors observé un surcoût de 400 Kb/s, ce qui correspond à 4% du débit total. Ce surcoût est suffisamment faible pour adopter cet algorithme dans ce cas.

Mais cet algorithme est-il toujours approprié quand l'ordinateur portable est mobile et connecté avec le lien WaveLan ? Dans ce cas précis, nous avons mesuré le débit de la bande passante pour chacun d'entre-eux. Les résultats sont présentés dans la figure[3] 7.13. De $t = 0$ à $t = 90000$ ms, la PDNV est implantée par l'algorithme PIDNV, et change ensuite pour l'algorithme PPDNV paramétré avec une période de 5000 ms.

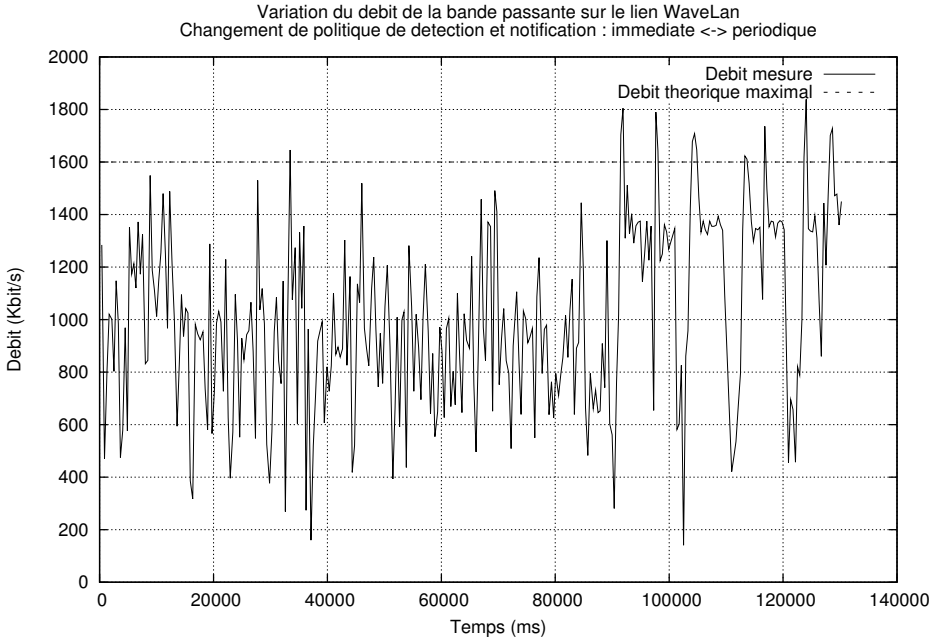


FIG. 7.13 – Débit mesuré de la bande passante selon l'algorithme de détection et notification utilisé

De manière à calculer le surcoût induit par chaque algorithme, nous utilisons la courbe de la figure 7.13 pour calculer les courbes de probabilité que la bande passante soit dispo-

[3]Dans la figure, le débit maximal théorique est dépassé à certains points. Ceci s'explique par le fait que la saturation de la bande passante produit un délai qui empêche les paquets de contrôle, utilisés pour mesurer le débit, d'être pris en compte dans la bonne période. Ainsi, quelques paquets envoyés pendant l'intervalle t sont pris en compte dans l'intervalle $t+1$.

nible à un instant t (voir figures 7.14 et 7.15). Nous avons ensuite cherché les intervalles tels que les probabilités, que la bande passante y soit disponible, soient les mêmes pour les algorithmes PIDNV et PPDNV. Les intervalles trouvés sont [800,1000] et [1200,1400] :

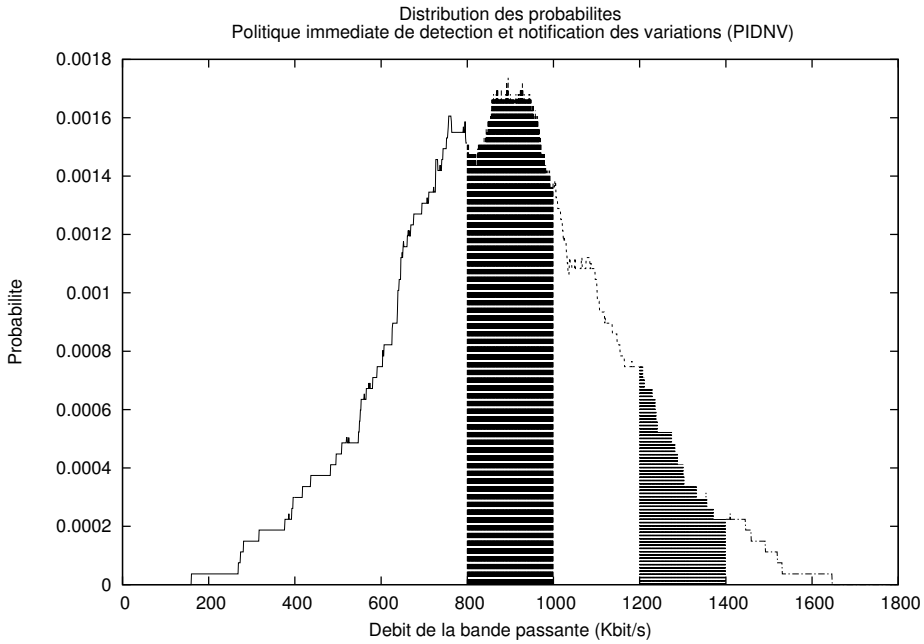


FIG. 7.14 – Courbe de distribution des probabilités dans le cas de l'algorithme immédiat

$$P_{PIDNV}(800 \leq \textit{débit} < 1000) = \sum_{X=800}^{X=1000} P_{PIDNV}(X) = 0.317 \tag{7.1}$$

$$P_{PIDNV}(1200 \leq \textit{débit} < 1400) = \sum_{X=1200}^{X=1400} P_{PIDNV}(X) = 0.085 \tag{7.2}$$

$$P_{PPDNV}(800 \leq \textit{débit} < 1000) = \sum_{X=800}^{X=1000} P_{PPDNV}(X) = 0.104 \tag{7.3}$$

$$P_{PPDNV}(1200 \leq \textit{débit} < 1400) = \sum_{X=1200}^{X=1400} P_{PPDNV}(X) = 0.279 \tag{7.4}$$

Ainsi, pour la même probabilité du débit de la bande passante (30%), l'algorithme PIDNV introduit un surcoût de 600 - 800 Kb qui correspond à 37.5 - 50% du débit maximal, tandis que l'algorithme PPDNV introduit un surcoût de 200 - 400 Kb qui correspond à 12.5 - 25% du débit maximal.

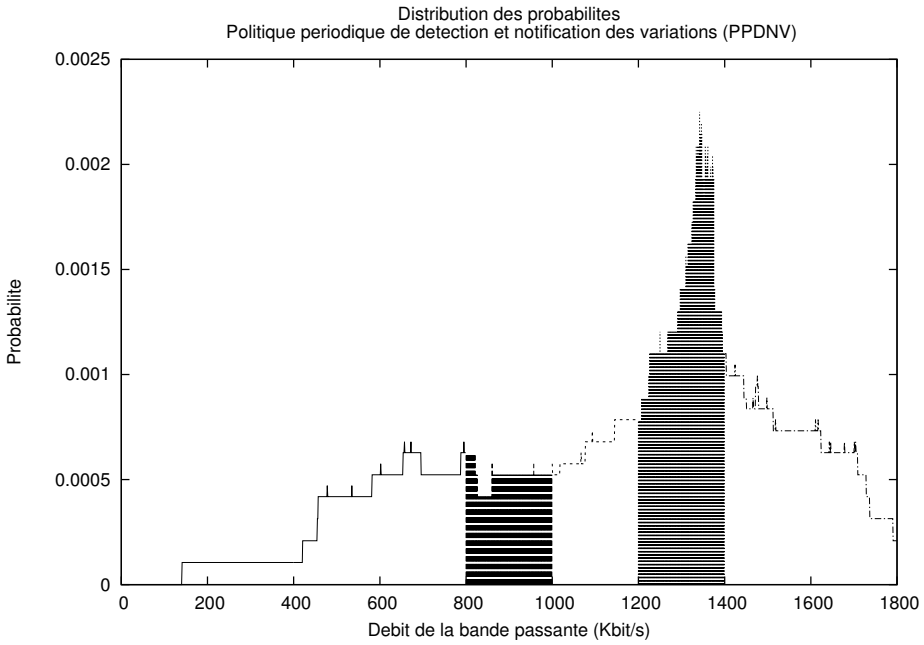


FIG. 7.15 – Courbe de distribution des probabilités dans le cas de l'algorithme périodique

Nous nous sommes aussi intéressé à la manière dont ces deux algorithmes réagissent aux variations. Pour cela, nous avons examiné la moyenne et l'écart-type de ces distributions de probabilités :

$$\mu_{PIDNV}(800 \leq d\acute{e}bit < 1000) = \frac{\sum_{X=800}^{X=1000} X \cdot P_{PIDNV}(X)}{\sum_{X=800}^{X=1000} P_{PIDNV}(X)} = 899 \tag{7.5}$$

$$\mu_{PPDNV}(1200 \leq d\acute{e}bit < 1400) = \frac{\sum_{X=1200}^{X=1400} X \cdot P_{PPDNV}(X)}{\sum_{X=1200}^{X=1400} P_{PPDNV}(X)} = 1311 \tag{7.6}$$

$$\sigma_{PIDNV}(800 \leq d\acute{e}bit < 1000) = \sqrt{\frac{\sum_{X=800}^{X=1000} P_{PIDNV}(X) \cdot (X - \mu_{PIDNV})^2}{\sum_{X=800}^{X=1000} P_{PIDNV}(X)}} = 59 \tag{7.7}$$

$$\sigma_{PPDNV}(1200 \leq \textit{débit} < 1400) = \sqrt{\frac{\sum_{X=1200}^{X=1400} P_{PPDNV}(X) \cdot (X - \mu_{PPDNV})^2}{\sum_{X=1200}^{X=1400} P_{PPDNV}(X)}} = 51 \qquad (7.8)$$

Ainsi, la distribution des probabilité de l'algorithme PPDNV est plus compacte autour de la moyenne que celle de l'algorithme PIDNV ($\sigma_{PPDNV} < \sigma_{PIDNV}$). Pour l'algorithme PPDNV, la probabilité que la bande passante disponible soit dans les intervalles [1000,1200] ou [1400,1600] décroît rapidement ($\sim$13-16%), tandis que, pour l'algorithme PIDNV, la probabilité pour les intervalles [600,800] ou [1000,1200] reste importante ($\sim$21-24%). Nous pouvons en déduire que l'algorithme PPDNV est moins sensible aux variations, ce qui le rend plus approprié pour un lien de type WaveLan.

Finalement, pour construire une politique de détection et notification qui n'introduise pas trop de surcoût et qui réagisse bien aux variations, il est heureux que notre système propose le changement d'algorithme. L'algorithme PIDNV convient parfaitement pour une connexion Ethernet. Pour une connexion de type WaveLan, l'algorithme PPDNV est un meilleur choix. Et pour d'autres types de connexion, comme les réseaux ad hoc, l'algorithme PPDNV ne serait plus tout à fait adapté [Vahdat et al. 2000] mais des algorithmes appropriés pourraient être dynamiquement incorporés à nos politiques adaptatives [Yu et Vahdat 2000].

7.3 Bilan

Notre système a montré qu'il peut facilement être spécialisé pour prendre en compte les abstractions de conception, comme les composants adaptatifs, de systèmes existants. L'expérimentation a permis de confirmer que la distribution en environnement mobile permet vraiment d'économiser les ressources des terminaux portables et d'améliorer les performances des applications en profitant de l'environnement.

Conclusion

Bilan

L'objectif de cette thèse était d'élaborer un système adaptatif de distribution en environnements mobiles. Dans un premier temps, nous nous sommes attachés à l'étude des systèmes existants. Notre approche chevauche le domaine des systèmes de gestion des environnements mobiles, le domaine des systèmes d'adaptation et le domaine des systèmes de distribution. L'étude parallèle de ces différents domaines nous a montré que (i) les systèmes de gestion des environnements mobiles offrent souvent des fonctionnalités trop spécifiques qui ne peuvent évoluer, (ii) les systèmes d'adaptation possèdent une grande généricité mais n'ont aucune fonctionnalité propre aux environnements mobiles et (iii) les systèmes de distribution ne possède aucune caractérisation des environnements mobiles et donc ne peuvent pas tirer parti de leurs particularités.

Après ce constat, nous nous sommes fixés des propriétés communes aux domaines précédents et que devait respecter notre approche : (i) *généricité*, (ii) *adaptabilité* en terme de *prise en compte de l'environnement* et *prise en compte de l'application*, (iii) *évolutivité*, (iv) *dynamicité*, (v) *efficacité* en terme de *performances* et *stabilité*. Dans cet esprit, nous avons proposé un système s'articulant autour de trois axes :

(i) En premier lieu, nous avons construit une architecture générique pour la gestion des environnements mobiles. Celle-ci se décompose en un *cadre de conception* et une *boîte à outils*. Le cadre de conception comporte des *fonctionnalités* couramment utilisées pour la gestion des environnements mobile. La boîte à outils comporte des *implantations* permettant de *spécialiser* les fonctionnalités avec un comportement défini. Quatre techniques de spécialisation sont disponibles : (i) la *composition*, (ii) l'*héritage/respect d'interfaces*, (iii) *la paramétrisation* et (iv) l'*ajout avec altération*[4].

(ii) Dans ce cadre, nous avons ensuite développé la *fonctionnalité d'adaptation et de réaction dynamique*. Celle-ci s'appuie sur un modèle d'abstraction de conception, l'*entité*, et sur une échelle des abstractions. À partir de ce modèle, nous avons défini par spécialisation dans l'échelle : (i) le modèle d'*entité adaptative* comprenant des *adaptations* et des *adaptations multiples* et (ii) le modèle d'*entité auto-adaptative* comprenant des *stratégies d'adaptations*.

[4]Notre système ne fournit pas d'outils pour l'ajout avec altération, mais, étant ouvert, il l'autorise et ne garantit pas alors la cohérence structurelle et fonctionnelle du système.

(iii) Dans ce cadre, nous avons enfin développé la *fonctionnalité de gestion des ressources et de distribution des applications*. Nous y avons caractérisé l'environnement mobile (avec toutes ses particularités) et les applications sur un modèle *ressources / offres* et *entités utilisatrices / demandes*. Ce modèle est obtenu par spécialisation du modèle d'entité. Nous y avons également défini et mis en place cinq services comprenant des *politiques adaptatives* : (i) *le service de gestion de l'environnement*, (ii) *le service de gestion de l'environnement local*, (iii) *le service de détection et notification*, (iv) *le service de distribution* et (v) *le service de contrôle de la propagation des adaptations*. Ces services sont également obtenus par spécialisation du modèle d'entité.

Nous avons validé notre approche par un prototype AeDEn et une application de type navigateur [Le Mouël 1999, Le Mouël et André 2000b, André et al. 2000, Le Mouël et André 2000a, Le Mouël et al. 2000, Le Mouël et André 2001, Le Mouël et al. 2002]. Ils ont confirmé que la distribution en environnement mobile permet d'économiser les ressources des terminaux portables et d'améliorer les performances des applications en profitant de l'environnement.

Perspectives

Ce travail nous a permis d'entrevoir plusieurs autres voix de recherche :

(i) Le système que nous proposons, grâce à ses différents niveaux d'abstraction, permet "normalement" de passer à l'échelle. Nous n'avons toutefois pas testé cette fonction. Un perspective intéressante serait donc de confronter les environnements mobiles, qui sont plutôt attachés à un endroit et de taille limitée, aux environnements à grande échelle, qui accroissent et abstraient les distances. Dans ce cadre, il serait possible (i) de développer des applications mobiles à grande échelle, (ii) de développer des environnements mobiles à grande échelle, ou (iii) d'insérer notre système dans un système à grande échelle préexistant comme Globus [Foster et Kesselman 1997] ou JXTA [Gong 2001].

(ii) Notre système ne fournit pas d'outils pour l'ajout avec altération. Pourtant, de nombreuses techniques d'adaptations dynamiques de code existent. Différentes techniques de spécialisation, comme l'évaluation partielle, la fragmentation de programme, la fusion et la fission de programme [Bobeff et Noyé 2003] pourraient être intégrées dans notre système.

(iii) Dans plusieurs cas, notre système pourrait être amélioré pour prendre des décisions plus "souples". Des capacités de négociation pourraient être examinées pour les cas suivants :

  (a) Lors de l'application d'une adaptation : lorsque deux entités sont liées par une dépendance, l'application d'une adaptation sur l'une peut affecter l'autre. Dans notre système, l'entité adaptée prévient ses entités dépendantes mais celles-ci ont un choix assez limité : (i) modifier leurs comportements pour être en conformité avec les adaptations réalisées, ou (ii) rompre les dépendances n'étant pas satisfaites ou ne pouvant s'adapter au nouveau comportement. Il pourrait être intéressant d'examiner une solution permettant d'entamer une procédure de négociation pour trouver un compromis d'adaptation satisfaisant les deux parties.

(b) Lors d'une concordance entre offre et demande : notre système utilise une procédure de concordance pour déterminer si une offre et une demande s'accordent. Comme précédemment, la décision prise est binaire, soit elles concordent, soit non. Il pourrait être intéressant d'avoir une solution qui permettent de négocier l'offre à la hausse ou la demande à la baisse si celles-ci ne sont pas trop disproportionnées (notre système propose déjà une métrique de décision qui pourrait être utilisée à cet effet).

Annexe A

Grammaire de description des éléments de l'environnement

Le compilateur JavaCC [BSD] intégré au niveau de la politique d'initialisation de l'environnement local implémente une grammaire pour la description des éléments de l'environnement local. Avant de présenter la grammaire, nous présentons quelque notions indispensables pour la bonne compréhension de celle-ci :

- (A)* : A peut exister zéro ou plusieurs fois.
- (A)+ : A peut exister une ou plusieurs fois.
- (A) ? : A est optionnel.
- $A \mid B$: A ou B.
- $\ll EOF \gg$: Caractère de fin de fichier.
- $\ll INTEGER \gg$: Nombre entier.
- $\ll FLOAT \gg$: Nombre réel.

La grammaire utilisée pour la description des éléments de l'environnement se présente comme suit :

$\ll INPUT \gg$: : ($\ll QUERY \gg$)* $\ll EOF \gg$

$\ll QUERY \gg$: : (
$\ll RESOURCE \gg$
|
$\ll USEENTITY \gg$
)
$\ll OWNERHOST \gg$
$\ll PLACEMENT \gg$
$\ll ISMOVABLE \gg$
($\ll ACCESSIBILITY \gg$) ?

≪ *OWNERHOST* ≫ : : ≪ *STATIONNAME* ≫

≪ *PLACEMENT* ≫ : : "PLACED ON" ≪ *STATIONTYPE* ≫
≪ *STATIONNAME* ≫

≪ *STATIONTYPE* ≫ : : "FIXED STATION" | "PDA" | ...

≪ *STATIONNAME* ≫ : : ≪ *IDENTIFIER* ≫

≪ *ISMOVABLE* ≫ : : "YES" | "NO"

≪ *ACCESSIBILITY* ≫ : : (
"ACCESSIBLE TO" (≪ *ACCESSLIST* ≫ | "ALL")
|
"NOT ACCESSIBLE")

≪ *ACCESSLIST* ≫ : : (≪ *USEENTITYNAME* ≫)+

≪ *RESOURCE* ≫ : : "RESOURCE" ≪ *RESOURCECLASS* ≫
≪ *RESOURCENAME* ≫ (≪ *OFFERS* ≫)* ?

≪ *RESOURCECLASS* ≫ : : "CPU" | "MEMORY" | ...

≪ *RESOURCENAME* ≫ : : ≪ *IDENTIFIER* ≫

≪ *USEENTITY* ≫ : : "USE ENTITY" ≪ *USEENTITYCLASS* ≫
≪ *USEENTITYNAME* ≫ (≪ *OFFERS* ≫)* ? (≪ *DEMANDS* ≫)* ?

≪ *USEENTITYCLASS* ≫ : : "COMPRESSION" | "DECOMPRESSION" | ...

≪ *USEENTITYNAME* ≫ : : ≪ *IDENTIFIER* ≫

≪ *IDENTIFIER* ≫ : : (≪ *CHAR* ≫ | " _ ")
(≪ *CHAR* ≫ | ≪ *INTEGER* ≫ | " − " | " _ ")*

≪ *DEMANDS* ≫ : : "DEMANDS" (≪ *RESSOURCE* ≫ | ≪ *USEENTITY* ≫)

≪ *OFFERS* ≫ : : "OFFERS" "[" ≪ *ATTRIBUTESLIST* ≫ "]"

≪ *ATTRIBUTESLIST* ≫ : : ≪ *ASSIGNEMENT* ≫ ("," ≪ *ASSIGNEMENT* ≫)*

≪ *ASSIGNEMENT* ≫ : : (≪ *ATTRIBUTENAME* ≫ "(" "VALUE"
≪ *OPERATOR* ≫ ≪ *VALUE* ≫ ≪ *UNIT* ≫
(")" | (≪ *UNIT* ≫ ","
(("MIN" ≪ *OPERATOR* ≫ ≪ *VALUE* ≫
")"
|
"," "MAX" ≪ *OPERATOR* ≫ ≪ *VALUE* ≫
)))
"MAX" ≪ *OPERATOR* ≫ ≪ *VALUE* ≫
)))

≪ *ATTRIBUTE* ≫ : : "CLOCKRATE" | "SIZE" | ...

≪ *OPERATOR* ≫ : : " = " | "≤" | "≥" | "<" | ">"

≪ *ASSIGNEMENT* ≫ : : ≪ *STRING* ≫ ≪ *FLOAT* ≫

≪ *STRING* ≫ : : (≪ *CHAR* ≫)*

≪ *CHAR* ≫ : : ["a" − "z"] | ["A" − "Z"]

≪ *UNIT* ≫ : : "Mhz" | "MO" | ...

Le fichier des éléments locaux interprété par le compilateur contient un ensemble de requêtes, chaque requête décrit un élément, ressource ou entité utilisatrice. La clause ≪ *HOST OWNER* ≫ est utilisée pour fixer à quelle station d'origine appartient un élément (comme

certains peuvent, en effet, migrer). La clause ≪ *PLACEMENT* ≫ spécifie la localisation actuelle de l'élément. La clause ≪ *ISMOVABLE* ≫ spécifie si l'élément peut être déplacé au cours de son exécution sur une autre station que sa station initiale. Ces clauses sont utilisées par le service de distribution du système AeDEn pour effectuer la distribution des éléments de l'environnement. La clause ≪ *ACCESSIBILITY* ≫ est optionnelle et exprime la notion de partage. En effet, chaque élément de l'environnement peut spécifier une liste des éléments qui peuvent l'utiliser, comme il peut interdire l'accès à tous les autres éléments.

Bibliographie

[10GEA] 10 Gigabit Ethernet Alliance, *Home Page*, http ://www.10gea.org. 1.2.3

[3GPP 2001] 3rd Generation Partnership Project, *Universal Mobile Telecommunications System (UMTS) ; General UMTS Architecture*, avril 2001, statut : « Publication (3GPP TS 23.101 version 4.0.0 Release 4) », http ://webapp.etsi.org/workprogram/Report_WorkItem.asp ?WKI_ID=13453. 1.2.2

[Adjie-Winoto et al. 1999] W. ADJIE-WINOTO, E. SCHWARTZ, H. BALAKRISHNAN et J. LILLEY, « The design and implementation of an intentional naming system », *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, p. 186–201, Kiawah Island Resort, near Charleston, South Carolina, USA, décembre 1999, http ://wind.lcs.mit.edu/papers/ins-sosp99.pdf. 3.2.1.3

[AgentBuilder] AgentBuilder, *Agent Construction Tools*, http ://www.agentbuilder.com/AgentTools/. 3.2.2

[Alanko et al. 1997] T. ALANKO, M. KOJO, M. LILJEBERG et K. RAATIKAINEN, « Mowgli : Improvements for Internet Applications Using Slow Wireless Links », *Proceedings of the 8th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Helsinki, Finland, septembre 1997. 3.1.1.1

[Ames et Gabor 2000] P. AMES et J. GABOR, « The Evolution of Third-Generation Cellular Standards », *Intel Technology Journal*, vol. Q2, mai 2000, http ://developer.intel.com/technology/itj/q22000/pdf/art_6.pdf. 1.2.2

[Andersen et al. 2000] A. ANDERSEN, G. S. BLAIR et F. ELIASSEN, « OOPP : A Reflective Component-Based Middleware », *Proceedings of NIK'2000*, Bodø, Norway, novembre 2000, http ://www.comp.lancs.ac.uk/computing/research/mpg/reflection/papers/nik.ps.gz. 2.4

[Anderson et al. 1995] T. E. ANDERSON, D. E. CULLER, D. A. PATTERSON et THE NOW TEAM, « A Case for NOW (Networks of Workstations) », *IEEE Micro*, vol. 15, n°1, janvier 1995, p. 54–64, http ://now.cs.berkeley.edu/Case/case.ps. 3.2.3

[Anderson et al. 1998] T. A. ANDERSON, Y. BREITBART, H. F. KORTH et A. WOOL, « Replication, Consistency, and Practicality : Are These Mutually Exclusive ? », *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, p. 484–495, Seattle, Washington, USA, juin 1998, http ://citeseer.nj.nec.com/anderson98replication.html. 3.1.3

[André et al. 2000] F. ANDRÉ, A.-M. KERMARREC et F. LE MOUËL, « Improvement of the QoS via an Adaptive and Dynamic Distribution of Applications in a Mobile Environment », *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'2000)*, p. 21–29, Nürnberg, Germany, octobre 2000, http ://www.irisa.fr/solidor/doc/ps00/SRDS2000.ps.gz. 4.3.1, 7.3

[André et Saint Pol 2000] F. ANDRÉ et E. SAINT POL, « A middleware for transactional hospital applications on local wireless networks », *Proceedings of 2000 International Conference on Parallel and Distributed Processing Techniques and Application (PDPTA'2000)*, Monte Carlo Resort Casino, Las Vegas, USA, juin 2000, http ://www.irisa.fr/solidor/doc/abstr00/pdpta00.html. 4.3.1

[André et Segarra 1999] F. ANDRÉ et M.-T. SEGARRA, « On Building a File System for Mobile Environments Using Generic Services », *Proceedings of the 12th International Conference on Parallel and Distributed Computing Systems (PDCS'99)*, Radisson Bahia Mar Beach Resort, Florida, USA, août 1999, http ://www.irisa.fr/solidor/doc/ps99/pdcs99.ps.gz. 2.2.3.2, 4.3.1

[Angin et al. 1998] O. ANGIN, A. T. CAMPBELL, M. E. KOUNAVIS et R. R.-F. LIAO, « The Mobiware Toolkit : Programmable Support for Adaptive Mobile Netwoking », *IEEE Personal Communications Magazine, Special Issue on Adapting to Network and Client Variability*, août 1998, http ://comet.ctr.columbia.edu/mobiware/papers/mobiware_pcs98.pdf. 2.3.2.1

[Aridor et Oshima 1998] Y. ARIDOR et M. OSHIMA, « Infrastructure for Mobile Agents : Requirements and Design », *Proceedings of Mobile Agents, 2nd International Workshop (MA'98)*, *Lecture Notes in Computer Science*, vol. 1477, p. 38–49, Springer Verlag, Stuttgart, Germany, septembre 1998. 3.2.2

[ATM] ATM Forum, *Home Page*, http ://www.atmforum.com. 1.2.3

[Bacon et al. 2000] J. BACON, K. MOODY, J. BATES, R. HAYTON, C. MA, A. MCNEIL, O. SEIDEL et M. SPITERI, « Generic Support for Distributed Applications », *IEEE Computer*, vol. 33, n°3, mars 2000, p. 68–76, http ://www.cl.cam.ac.uk/Research/SRG/opera/publications/Papers/computer-march2000.pdf. 3.2.2

[Baggio et al. 2001] A. BAGGIO, G. BALLINTIJN, M. VAN STEEN et A. S. TANENBAUM, « Efficient Tracking of Mobile Objects in Globe », *The Computer Journal*, vol. 44, n°5, 2001, p. 340–353, http ://www.cs.vu.nl/pub/papers/globe/compjournal.01.pdf. 3.2.1

[Baker et al. 1996] M. BAKER, X. ZHAO, S. CHESHIRE et J. STONE, « Supporting Mobility in MosquitoNet », *Proceedings of the 1996 USENIX Technical Conference*, p. 127–140, janvier 1996, http ://mosquitonet.stanford.edu/publications/usenix96.mobile.ps. 2.3.1.1

[Bakre et Badrinath 1995a] A. V. BAKRE et B. R. BADRINATH, « I-TCP : Indirect TCP for Mobile Hosts », *Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS'95)*, p. 136–143, Vancouver, British Columbia, Canada, mai 1995, ftp ://paul.rutgers.edu/pub/badri/itcp-tr314.ps.Z. 2.2.2.1

[Bakre et Badrinath 1995b] A. V. BAKRE et B. R. BADRINATH, « M-RPC : A Remote Procedure Call Service for Mobile Clients », *Proceedings of the 1st Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'95)*, p. 97–110, Berkeley, California, USA, novembre 1995, ftp ://paul.rutgers.edu/pub/badri/mrpc.ps.Z. 3.2.2

[Bakre et Badrinath 1997] A. V. BAKRE et B. R. BADRINATH, « Implementation and Performance Evaluation of Indirect TCP », *IEEE Transactions on Computers*, vol. 46, n°3, mars 1997, p. 260–278, http ://www.cs.kau.se/cs/prtp/papers/t0260.pdf.gz. 2.2.2.1

[Balachandran et al. 1997] A. BALACHANDRAN, A. T. CAMPBELL et M. E. KOUNAVIS, « Active Filters : Delivering Scaled Media to Mobile Devices », *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, St. Louis, Missouri, USA, mai 1997, http ://comet.ctr.columbia.edu/mobiware/papers/filters_nossdav97.pdf. 2.3.2.1

[Balakrishnan et al. 1995a] H. BALAKRISHNAN, S. SESHAN, E. AMIR et R. H. KATZ, « Improving TCP/IP Performance over Wireless Networks », *Proceedings of the 1st Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'95)*, p. 2–11, Berkeley, California, USA, novembre 1995, http ://www.cs.berkeley.edu/∼hari/papers/mcn.ps. 2.2.2.2

[Balakrishnan et al. 1995b] H. BALAKRISHNAN, S. SESHAN et R. H. KATZ, « Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks », *ACM Wireless Networks*, vol. 1, n°4, 1995, http ://daedalus.cs.Berkeley.edu/publications/winet.ps.gz. 2.2.2.2

[Balakrishnan et al. 1997] H. BALAKRISHNAN, V. N. PADMANABHAN, S. SESHAN et R. H. KATZ, « A comparison of mechanisms for improving TCP performance over wireless links », *IEEE/ACM Transactions on Networking*, vol. 5, n°6, 1997, p. 756–769, http ://www.cs.berkeley.edu/∼hari/papers/ton.ps. 2.2.2

[Balan et al. 2003] R. K. BALAN, M. SATYANARAYANAN, S. PARK et T. OKOSHI, « Tactics-Based Remote Execution for Mobile Computing », *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys'2003)*, San Francisco, CA, USA, mai 2003, http ://gs129.sp.cs.cmu.edu/papers/mobisys03.pdf. 6.5

[Ballintijn et al. 2000] G. BALLINTIJN, M. VAN STEEN et A.S. TANENBAUM, « Scalable Naming in Global Middleware », *Proceedings of the 13th International Conference on Parallel and Distributed Computing Systems (PDCS'2000)*, p. 624–631, Las Vegas, Nevada, USA, août 2000, http ://www.cs.vu.nl/pub/papers/globe/pdcs.00.pdf. 3.2.1

[Bandyopadhyay et Paul 1999] S. BANDYOPADHYAY et K. PAUL, « Evaluating the performance of mobile agent based message communication among mobile hosts in large ad hoc wireless network », *Proceedings of the 2nd ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, In conjunction with the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, p. 69–73, Seattle, Washington, USA, août 1999, http ://www.iimcal.ac.in/faculty/facpage.asp ?ID=somprakash&tab=4. 3.2.2.2

[Beauvois et Coupaye 2002] M. BEAUVOIS et T. COUPAYE, « La composition de services techniques vu comme une composition d'aspects non orthogonaux », *Actes de la Journées Composants (JC2002), ASF (ACM SIGOPS France)*, Grenoble, France, octobre 2002, http ://arcad.essi.fr/2002-10-composants/papiers/01-court-beauvois.pdf. 5.4

[Bellavista et al. 1999] P. BELLAVISTA, A. CORRADI et C. STEFANELLI, « A Secure and Open Mobile Agent Programming Environment », *Proceedings of the 4th International Symposium on Autonomous Decentralized Systems (ISADS'99)*, p. 238–245, Tokyo, Japan, mars 1999, http ://www.computer.org/proceedings/isads/0137/01370238abs.htm. 3.2.3.2

[Bellavista et al. 2000] P. BELLAVISTA, A. CORRADI et A. TOMASI, « The Mobile Agent Technology to Support and to Access Museum Information », *Proceedings of the 2000 ACM Symposium on Applied Computing (SAC'2000)*, p. 1006–1013, Villa Olmo, Como, Italy, mars 2000, http ://lia.deis.unibo.it/Staff/PaoloBellavista/papers/sac00a.pdf. 3.2.3.2

[Bellavista et al. 2001a] P. BELLAVISTA, A. CORRADI et C. STEFANELLI, « How to Monitor and Control Resource Usage in Mobile Agent Systems », *Proceedings of the 3rd International Symposium on Distributed Objects & Applications (DOA'01)*, p. 65–75, Rome, Italy, septembre 2001, http ://lia.deis.unibo.it/Staff/PaoloBellavista/papers/doa01.pdf. 3.2.3.2

[Bellavista et al. 2001b] P. BELLAVISTA, A. CORRADI et C. STEFANELLI, « Mobile Agent Middleware for Mobile Computing », *IEEE Computer*, vol. 34, n°3, mars 2001, p. 73–81, http ://www.computer.org/computer/co2001/r3073abs.htm. 3.2.3.2

[Bellavista et Corradi 2002] P. BELLAVISTA et A. CORRADI, « How to Support Internet-based Distribution of Video on Demand to Portable Devices », *Proceedings of the 7th IEEE International Symposium on Computers and Communications (ISCC'2002)*, p. 126–132, Taormina, Italy, juillet 2002, http ://www-lia.deis.unibo.it/Staff/PaoloBellavista/papers/iscc02a.pdf. 3.2.3.2

[Bellifemine et al. 1999] F. BELLIFEMINE, A. POGGI et G. RIMASSA, « JADE – A FIPA-compliant agent framework », *Proceedings of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'99)*, p. 97–108, London, UK, avril 1999, http ://sharon.cselt.it/projects/jade/papers/PAAM.pdf. 3.2.2.4

[Bergenti et Poggi 2001] F. BERGENTI et A. POGGI, « LEAP : a FIPA Platform for Handheld and Mobile Devices », *Proceedings of the 8th International Workshop on Agent Theories, Archi-*

tectures, and Languages (ATAL'2001), Seattle, Washington, USA, août 2001, http ://leap.crm-paris.com/public/docs/ATAL2001.pdf. 3.2.2.4

[Berger et al. 2001] M. BERGER, B. BAUER et M. WATZKE, « A Scalable Agent Infrastructure », *Proceedings of the 2nd International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, At the 5th International Conference on Autonomous Agents (Agents'2001)*, Montreal, Canada, juin 2001, http ://leap.crm-paris.com/public/docs/ScalableAgentsInfrastructrueFinal.pdf. 3.2.2.4

[Berger 2001] L. BERGER, *Mise en oeuvre des interactions en environnements distribués, compilés et fortement typés : le modèle MICADO*, Thèse de doctorat, Université de Nice, Nice, France, octobre 2001, http ://www.essi.fr/∼rainbow/Publi/TheseLaurent.ps.gz. 2.4

[Berger 2002] L. BERGER, « Intéraction et modèles de programmation, support des interactions dans les systèmes objets et componentiels », *Numéro spécial de la revue L'OBJET*, vol. 8, n°3, septembre 2002, p. 9–38. 5.4

[Bernard et al. 1991] G. BERNARD, D. STÈVE et M. SIMATIC, « Placement et migration de processus dans les systèmes répartis faiblement couplés », *Technique et Science Informatiques*, vol. 10, n°5, mai 1991, http ://etna.int-evry.fr/∼bernard/publis-gb/french.html. 3.2.3

[Bernard et Folliot 1996] G. BERNARD et B. FOLLIOT, « Caractéristiques Générales du Placement Dynamique : Synthèse et Problématique », *Tutoriel invité dans les actes de l'école d'été MASI-IMAG-INT-PRISM "Placement dynamique et répartition de charge : application aux systèmes parallèles et répartis"*, Presqu'île de Giens, France, juillet 1996, http ://www-inf.int-evry.fr/∼bernard/papers/PRC96.ps.gz. (document), 3.2.3

[Bharadvaj et al. 1998] H. BHARADVAJ, A. JOSHI et S. AUEPHANWIRIYAKUL, « An Active Transcoding Proxy to Support Mobile Web Access », *Proceedings of the 17th Symposium on Reliable Distributed Systems (SRDS'98)*, p. 118–123, West Lafayette, Indiana, USA, octobre 1998, http ://www.cs.umbc.edu/∼ajoshi/resch/mowserh.pdf. 3.1.1.3

[Bhattacharya et Das 1999] A. BHATTACHARYA et S. K. DAS, « LeZi-Update : An Information-Theoretic Approach to Track Mobile Users in PCS Networks », *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, p. 1–12, Seattle, Washington, USA, août 1999, http ://crewman.uta.edu/∼amiya/pubs/mcn99.pdf. 3.1.2

[Bianchi et al. 1998] G. BIANCHI, A. T. CAMPBELL et R. R.-F. LIAO, « On Utility-Fair Adaptive Services in Wireless Networks », *Proceedings of the 6th International Workshop on Quality of Service (IWQoS'98)*, Napa Valley, California, USA, mai 1998, http ://comet.columbia.edu/∼liao/publications/iwqos98.pdf. 2.3.2.1

[Biaz et Vaidya 1998] S. BIAZ et N. H. VAIDYA, « Distinguishing Congestion Losses from Wireless Transmission Losses : A Negative Result », *Proceedings of the 7th International Conference on Computer Communications and Networks (IC3N'98)*, p. 722–731, Lafayette, Louisiana, USA, octobre 1998, http ://www.cs.tamu.edu/faculty/vaidya/papers/mobile-computing/ic3n98.ps. 2.2.2.3

[Biaz et Vaidya 1999] S. BIAZ et N. H. VAIDYA, « Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver », *Proceedings of the IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET'99)*, p. 10–17, Richardson, Texas, USA, mars 1999, http ://www.cs.tamu.edu/faculty/vaidya/papers/mobile-computing/asset99.ps. 2.2.2.3

[Blair et al. 1998] G. BLAIR, G. COULSON, P. ROBIN et M. PAPATHOMAS, « An Architecture for Next Generation Middleware », *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*, p. 191–206, The Lake District, England, septembre 1998, ftp ://ftp.comp.lancs.ac.uk/pub/mpg/MPG-98-27.ps.Z. 2.4

[Blay-Fornarino et al. 2002] M. BLAY-FORNARINO, D. ENSELLEM, A. OCCELLO, A.-M. PINNA-DERY, M. RIVEILL, J. FIERSTONE, O. NANO et G. CHABERT, « Un service d'interactions :

principes et implémentation », *Actes de la Journées Composants (JC2002), ASF (ACM SIGOPS France)*, Grenoble, France, octobre 2002, http ://arcad.essi.fr/2002-10-composants/papiers/02-long-blay.pdf. 5.4

[Bluetooth] Bluetooth, *Home Page*, http ://www.bluetooth.com. 1.2.2

[Bobeff et Noyé 2003] G. BOBEFF et J. NOYÉ, « Molding Components using Program Specialization Techniques », *Actes de la Journée du groupe de travail OCM (Objets, Composants et Modèles) (en marge de LMO'2003)*, Vannes, France, février 2003, http ://www.univ-ubs.fr/valoria/Jacques.Malenfant/ALP.OCM/Journee2003/Bobeff.pdf. 5.4, 7.3

[Bobrow et al. 1988] D. BOBROW, L. DIMICHIEL, R. P. GABRIEL, S. KEENE, G. KICZALES et D. MOON, « Common LISP Object System Specification : X3J13 Document 88-002R », *ACM SIGPLAN Notices*, vol. 23, n°(Special Issue), septembre 1988, p. 1–143, http ://citeseer.nj.nec.com/context/81177/0. 2.4

[Bom et al. 1998] J. BOM, P. MARQUES, M. CORREIA et P. PINTO, « QoS Control : an Application Integrated Framework », *Proceedings of the 1st IEEE International Conference on ATM (ICATM'98)*, Colmar, France, juin 1998, http ://www.di.fc.ul.pt/∼mpc/icatm.ps.gz. 3.2.3

[Bonfigli et al. 2001] M. E. BONFIGLI, G. CABRI, L. LEONARDI et F. ZAMBONELLI, « Mobile Devices to Assist Cultural Visits », *Proceedings of the International Cultural Heritage Informatics Meeting (ICHIM'2001)*, p. 183–189, Politechnico di Milano, Milan, Italy, septembre 2001, http ://www.archimuse.com/ichim2001/abstracts/prg_115000651.html. 3.2.3.3

[Bouraqadi-Saâdani et al. 2001a] N. M. N. BOURAQADI-SAÂDANI, T. LEDOUX et M. SÜDHOLT, *A Reflective Infrastructure for Coarse-Grained Strong Mobility and its Tool-Based Implementation*, Invited presentation at the International Workshop on Experiences with reflective systems, In conjunction with Reflection 2001, the 3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns), septembre 2001, publié aussi dans [Bouraqadi-Saâdani et al. 2001b]. 2.4, 2.4.0.1

[Bouraqadi-Saâdani et al. 2001b] N. M. N. BOURAQADI-SAÂDANI, T. LEDOUX et M. SÜDHOLT, *A Reflective Infrastructure for Coarse-Grained Strong Mobility and its Tool-Based Implementation*, rapport technique n°TR 01/7/INFO, École des Mines de Nantes, Nantes, France, juillet 2001, http ://www.emn.fr/info/recherche/publications/RR01/01-7-INFO.pdf. A

[Box et al. 2000] D. BOX, D. EHNEBUSKE, G. KAKIVAYA, A. LAYMAN, N. MENDELSOHN, H. F. NIELSEN, S. THATTE et D. WINER, *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium, mai 2000, statut : « W3C Note, W3C Submission », http ://www.w3.org/TR/SOAP/. 3.2.2

[Bradshaw] J. BRADSHAW (éd.), *Handbook of Agent Technology*, AAAI/MIT-Press, à paraître, http ://mitpress.mit.edu. 3.2.2, A

[Brown et Singh 1997] K. BROWN et S. SINGH, « M-TCP : TCP for Mobile Cellular Networks », *ACM Computer Communication Review*, vol. 27, n°5, 1997, p. 19–43, ftp ://ftp.ece.orst.edu :/pub/users/singh/papers/mtcp.ps.gz. 2.2.2.1

[Bruneton et al. 2002] E. BRUNETON, R. LENGLET et T. COUPAYE, « ASM : un outil de manipulation de code pour la réalisation de systèmes adaptables (ASM : a code manipulation tool for the construction of adaptable systems) », *Actes de la Journées Composants (JC2002), ASF (ACM SIGOPS France)*, Grenoble, France, octobre 2002, http ://arcad.essi.fr/2002-10-composants/papiers/17-long-bruneton.pdf. 5.4

[BSD] Berkeley Software Distribution (BSD) License, *Java Compiler Compiler^{TM} (JavaCC^{TM}) - The Java Parser Generator - Home Page*, https ://javacc.dev.java.net. 7.1.4.2, A

[Budau et Bernard 2003] V. BUDAU et G. BERNARD, « Auto-Adaptation to Communication Environment through Dynamic Change of Communication Model », *Proceedings of 23rd International Conference on Distributed Computing Systems Workshops (ICDCS'2003 Workshops) – DARES - The International Workshop on Distributed Auto-Adaptive and Reconfigurable Systems*, p. 153–158, Providence, Rhode Island, USA, mai 2003, http ://csdl.computer.org/comp/proceedings/icdcsw/2003/1921/00/19210153abs.htm. 5.4

[Bézivin 2001] J. BÉZIVIN, « From Object Composition to Model Transformation with the MDA », *Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS'39)*, p. 350–354, Santa Barbara, California, USA, août 2001, http ://www.sciences.univ-nantes.fr/info/lrsg/Recherche/mda/TOOLS.USA.pdf. 5.3.1

[Cabillic et Puaut 1996] G. CABILLIC et I. PUAUT, « Dealing with Heterogeneity in Stardust : An Environment for Parallel Programming on Networks of Heterogeneous Workstations », *Proceedings of 2nd International Euro-Par Conference (Euro-Par'96)*, p. 114–119, Lyon, France, août 1996, http ://www.irisa.fr/solidor/doc/ps96/stardust-europar96.ps.gz. 3.2.3

[Cabri et al. 1998] G. CABRI, L. LEONARDI et F. ZAMBONELLI, « Reactive Tuple Spaces for Mobile Agent Coordination », *Proceedings of Mobile Agents, 2nd International Workshop (MA'98)*, *Lecture Notes in Computer Science*, vol. 1477, p. 237–248, Springer Verlag, Stuttgart, Germany, septembre 1998, http ://polaris.ing.unimo.it/MOON/papers/papers.html#Paper3. 3.2.2.4

[Cabri et al. 2000a] G. CABRI, L. LEONARDI et F. ZAMBONELLI, « Auction-Based Agent Negotiation via Programmable Tuple Spaces », *Proceedings of the 4th International Workshop on Cooperative Information Agents (CIA'2000)*, *Lecture Notes in Computer Science*, vol. 1860, p. 83–94, Springer Verlag, Boston, Massachusetts, USA, juillet 2000, http ://polaris.ing.unimo.it/MOON/papers/cia00.pdf. 3.2.3.3, 3.2.3.3

[Cabri et al. 2000b] G. CABRI, L. LEONARDI et F. ZAMBONELLI, « Context-Dependency in Internet-Agent Coordination », *Proceedings of the 1st International Workshop on Engineering Societies in the Agent World (ESAW'2000)*, *Lecture Notes in Computer Science*, vol. 1972, p. 51–63, Springer Verlag, Berlin, Germany, août 2000, http ://polaris.ing.unimo.it/MOON/papers/esaw00.pdf. 3.2.3.3

[Cabri et al. 2000c] G. CABRI, L. LEONARDI et F. ZAMBONELLI, « MARS : A Programmable Coordination Architecture for Mobile Agents », *IEEE Internet Computing*, vol. 4, n°4, juillet–août 2000, p. 26–35, http ://sirio.dsi.unimo.it/Zambonelli/PDF/MARS.pdf. 3.2.3.3

[Cáceres et Iftode 1994] R. CÁCERES et L. IFTODE, « The Effects of Mobility on Reliable Transport Protocols », *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems (ICDCS'94)*, p. 12–20, Poznan, Poland, juin 1994, http ://www.kiskeya.net/ramon/work/pubs/icdcs94.ps.gz. 2.2.2

[Cáceres et Iftode 1995] R. CÁCERES et L. IFTODE, « Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments », *IEEE Journal on Selected Areas in Communications*, vol. 13, n°5, juin 1995, p. 850–857, http ://www.kiskeya.net/ramon/work/pubs/jsac95.ps.gz. 2.2.2.3

[Campadello et al. 2000] S. CAMPADELLO, O. KOSKIMIES, K. RAATIKAINEN et H. HELIN, « Wireless Java RMI », *Proceedings of the 4th International Enterprise Distributed Objects Computing Conference (EDOC'2000)*, p. 114–123, Makuhari, Japan, septembre 2000, http ://www.cs.helsinki.fi/research/monads/papers/edoc2000/edoc2000.pdf. 3.2.2

[Campbell et al. 1999] A. T. CAMPBELL, M. E. KOUNAVIS et R. R.-F. LIAO, « Programmable Mobile Networks », *Computer Networks and ISDN Systems*, vol. 31, n°7, 1999, p. 49–73, http ://comet.ctr.columbia.edu/mobiware/papers/pmn_comnet.pdf. 2.3.2.1

[Campbell et Coulson 1997] A. T. CAMPBELL et G. COULSON, « A QOS Adaptive Multimedia Transport System : Design, Implementation and Experiences », *Distributed Systems En-*

gineering Journal, Special Issue on Quality of Service, vol. 4, n°1, mars 1997, p. 48–58, http ://comet.ctr.columbia.edu/∼campbell/andrew/publications/papers/dsej97.pdf. 3.2.3

[Campbell 1997] A. T. CAMPBELL, « Mobiware : QOS Aware Middleware for Mobile Multimedia Communications », *Proceedings of the 7th IFIP International Conference on High Performance Networking (HPN)*, p. 166–183, White Plains, New York, USA, avril 1997, http ://comet.ctr.columbia.edu/mobiware/papers/mobiware_hpn97.pdf. 2.3.2.1

[Caromel et al. 1998] D. CAROMEL, W. KLAUSER et J. VAYSSIERE, « Towards Seamless Computing and Metacomputing in Java », *Concurrency – Practice and Experience*, vol. 10, n°11–13, septembre–novembre 1998, p. 1043–1061, http ://www-sop.inria.fr/oasis/ProActive/doc/javallCPE.ps. 2.4

[Carriero et al. 1995] N. CARRIERO, E. FREEMAN, D. GELERNTER et D. KAMINSKY, « Adaptive Parallelism and Piranha », *IEEE Computer*, vol. 28, n°1, janvier 1995, p. 40–49, http ://www.cs.yale.edu/Linda/papers/shortp.ps. 3.2.2.4

[Carriero et Gelernter 1989] N. CARRIERO et D. GELERNTER, « How to Write Parallel Programs : A Guide to the Perplexed », *ACM Computing Surveys*, vol. 21, n°3, septembre 1989, p. 323–357, http ://www.cs.yale.edu/Linda/linda-lang.html. 3.2.2.4

[Casio] Casio, *Home Page*, http ://www.casio.com. 1.1.1

[Cazzola et Ancona 2000] W. CAZZOLA et M. ANCONA, *mChaRM : a Reflective Middleware for Communication-Based Reflection*, rapport technique n°DISI-TR-00-09, Università degli Studi di Genova, Genova, Italy, mai 2000, http ://www.disi.unige.it/person/CazzolaW/ps/DISI-TR-00-09.ps.gz. 2.4

[Cazzola 2000] W. CAZZOLA, *Communication-Oriented Reflection : a Way to Open Up the RMI Mechanism*, Thèse de doctorat, Università degli Studi di Genova, Milano, Italy, novembre 2000, http ://www.disi.unige.it/person/CazzolaW/ps/phd-thesis.ps.gz. 2.4

[Cen et al. 1995] S. CEN, C. PU, R. STAEHLI, C. COWAN et J. WALPOLE, « A Distributed Real-Time MPEG Video Audio Player », *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, *Lecture Notes in Computer Science*, vol. 1018, p. 142–153, Springer Verlag, Durham, New Hampshire, USA, avril 1995, ftp ://cse.ogi.edu/pub/dsrg/synthetix/nossdav.ps.gz. 3.2.3

[Cervantes et Ketfi 2002] H. CERVANTES et M. KETFI, « Composants adaptables au dessus d'OSGi », *Actes de la Journées Composants (JC2002), ASF (ACM SIGOPS France)*, Grenoble, France, octobre 2002, http ://www-adele.imag.fr/Les.Publications/intConferences/JOURNEES2002Cer.pdf. 5.4

[Çetintemel et al. 2001] U. ÇETINTEMEL, P. J. KELEHER et M. FRANKLIN, « Support for Speculative Update Propagation and Mobility in Deno », *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'2001)*, Mesa, Arizona, USA, avril 2001, http ://www.cs.umd.edu/∼keleher/papers/icdcs-deno01.pdf. 3.1.3.3

[Çetintemel et Keleher 2000a] U. ÇETINTEMEL et P. J. KELEHER, « Light-Weight Currency Management Mechanisms in Deno », *Proceedings of the 10th International Workshop on Research Issues on Data Engineering : Middleware for Mobile Business Applications and E-Commerce (RIDE'2000)*, p. 17–24, San Diego, California, USA, février 2000, http ://www.cs.umd.edu/∼keleher/papers/ride2k.pdf. 3.1.3.3

[Çetintemel et Keleher 2000b] U. ÇETINTEMEL et P. J. KELEHER, « Performance of Mobile, Single-Object, Replication Protocols », *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'2000)*, p. 218–227, Nürnberg, Germany, octobre 2000, http ://www.cs.umd.edu/∼keleher/papers/srds2000.pdf. 3.1.3.3

[Chalmers et al. 2001] D. CHALMERS, M. SLOMAN et N. DULAY, « Map Adaptation for Users of Mobile Systems », *Proceedings of the 10th International World Wide Web Conference (WWW'10)*, p. 735–744, Hong Kong, China, mai 2001, http ://www.doc.ic.ac.uk/∼dc/Papers/www10.pdf. 3.1.1.4

[Chalmers et Sloman 1999] D. CHALMERS et M. SLOMAN, « A Survey of Quality of Service in Mobile Computing Environments », *IEEE Communications Surveys*, vol. 2, n°2, Second Quarter 1999, http ://www.doc.ic.ac.uk/∼dc/Papers/QoSSurvey_published.pdf. 2.3.1

[Chan et al. 1997] A. C. F. CHAN, D. H. K. TSANG et S. GUPTA, « Impacts of Handoff on TCP Performance in Mobile Wireless Computing », *Proceedings of the IEEE International Conference on Personal Wireless Communications (ICPWC'97)*, p. 184–188, Mumbai (Bombay), India, décembre 1997, http ://www.ee.ust.hk/∼ustatm/papers/alder_icpwc97.ps. 2.2.2.2

[Chang et al. 1997] H. CHANG, C. D. TAIT, N. COHEN, M. SHAPIRO, S. MASTRIANNI, R. FLOYD, B. C. HOUSEL et D. B. LINDQUIST, « Web Browsing in a Wireless Environment : Disconnected and Asynchronous Operation in ARTour Web Express », *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'97)*, p. 260–269, Budapest, Hungary, septembre 1997, http ://www.acm.org/pubs/articles/proceedings/comm/262116/p260-chang/p260-chang.pdf. 2.2.4.1

[Chapin et al. 1999] S. J. CHAPIN, D. KATRAMATOS, J. F. KARPOVICH et A. S. GRIMSHAW, « The Legion Resource Management System », *Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'99), In conjunction with the 2nd Merged Symposium 13th International Parallel Processing Symposium & 10th Symposium on Parallel and Distributed Processing (IPPS/SPDP'99)*, *Lecture Notes in Computer Science*, vol. 1659, p. 162–178, Springer Verlag, San Juan, Puerto Rico, avril 1999, http ://www.cs.virginia.edu/∼legion/papers/legionrm.pdf. 3.2.3

[Chappell 1997] D. CHAPPELL, « The Microsoft Transaction Server (MTS) – Transactions Meet Components », *Patricia Seybold's Distributed Computing Monitor Newsletter*, juin 1997, http ://www.microsoft.com/com/wpaper/mtscomp.asp. 3.2.2

[Chatonnay et al. 2000] P. CHATONNAY, L. PHILIPPE et H. Y. CHAN, « Evaluation of a Multicriteria Method to Optimize Resource Access in Distributed Object Systems », *Parallel and Distributed Computing Practices*, vol. 3, n°1, décembre 2000, p. 21–31, http ://www.cs.okstate.edu/∼pdcp/vols/vol03/vol03no1abs.html. 3.2.3

[Cheverst et al. 2000] K. CHEVERST, N. DAVIES, K. MITCHELL, A. FRIDAY et C. EFSTRATIOU, « Developing a Context-aware Electronic Tourist Guide : Some Issues and Experiences », *Proceedings of the Annual Conference on Human Factors in Computing Systems (CHI'2000)*, p. 17–24, The Hague, The Netherlands, avril 2000, http ://www.guide.lancs.ac.uk/CHIpaper.pdf. 3.1.2

[Chiba et Tatsubori 1998] S. CHIBA et M. TATSUBORI, « Yet Another java.lang.Class », *Proceedings of the Workshop on Reflective Object-Oriented Programming and Systems at the 12th European Conference on Object-Oriented Programming (ECOOP'98)*, p. 372–373, Brussels, Belgium, juillet 1998, http ://www.csg.is.titech.ac.jp/∼mich/openjava/papers/chiba_ECOOP98ws.pdf. 2.4, 4.3

[Chiba 1995] S. CHIBA, « A Metaobject Protocol for C++ », *Proceedings of the 10th ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'95)*, Austin, Texas, USA, octobre 1995, http ://www.csg.is.titech.ac.jp/∼chiba/pub/chiba-oopsla95.ps.gz. 2.4

[Chiba 1998a] S. CHIBA, « Javassist – A Reflection-based Programming Wizard for Java », *Proceedings of the Workshop on Reflective Programming in C++ at the 13th ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'98)*, Vancouver, Canada, octobre 1998, http ://www.csg.is.titech.ac.jp/∼chiba/oopsla98/proc/chiba.pdf. 2.4, 4.3

[Chiba 1998b] S. CHIBA, « Macro Processing in Object-Oriented Languages », *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS Pacific'98)*, Melbourne, Australia, novembre 1998, http ://www.csg.is.titech.ac.jp/∼chiba/pub/chiba-tools98.ps.gz. 2.4

[Chiba 2000] S. CHIBA, « Load-Time Structural Reflection in Java », *Proceedings of the 14th European Conference on Object-Oriented Programming (ECOOP'2000)*, *Lecture Notes in Computer Science*, vol. 1850, p. 313–336, Springer Verlag, Cannes, France, juin 2000, http ://www.csg.is.titech.ac.jp/∼chiba/pub/chiba-ecoop00.pdf. 2.4, 4.3

[Chim et al. 1998] J. H. P. CHIM, M. GREEN, R. W. H. LAU, H. VA LEONG et A. SI, « On Caching and Prefetching of Virtual Objects in Distributed Virtual Environments », *Proceedings of the 6th ACM International Conference on Multimedia (Multimedia'98)*, p. 171–180, Bristol, England, septembre 1998, http ://www.acm.org/sigs/sigmm/MM98/electronic_proceedings/chim/. 3.1.2

[Chinnici et al. 2003] R. CHINNICI, M. GUDGIN, J.-J. MOREAU et S. WEERAWARANA, *Web Services Description Language (WSDL) 1.2*, World Wide Web Consortium, mars 2003, statut : « W3C Working Draft », http ://www.w3.org/TR/wsdl12/. 3.2.1, 7.1.4.2

[Compaq] Compaq, *Home Page*, http ://www.compaq.com. 1.1.1

[Corradi et al. 1999] A. CORRADI, L. LEONARDI et F. ZAMBONELLI, « Diffusive Load-Balancing Policies for Dynamic Applications », *IEEE Concurrency*, vol. 7, n°1, janvier–mars 1999, p. 22–31, http ://sirio.dsi.unimo.it/Zambonelli/PDF/Concurrency.pdf. 3.2.3

[Corson et Macker 1999] S. CORSON et J. MACKER, *Mobile Ad hoc Networking (MANET) : Routing Protocol Performance Issues and Evaluation Considerations*, Request For Comments (RFC) 2501, janvier 1999, statut : « Informational », http ://www.ietf.org/rfc/rfc2501.txt. 1.2.1.2

[Costa 2001] F. M. COSTA, *Combining Meta-Information Management and Reflection in an Architecture for Configurable and Reconfigurable Middleware*, Thèse de doctorat, Computing Department, Lancaster University, Lancaster, UK, septembre 2001, http ://www.comp.lancs.ac.uk/computing/users/fmc/papers/thesis.pdf. 2.4

[Courtrai et al. 2003] L. COURTRAI, F. GUIDEC, N. LE SOMMER et Y. MAHÉO, « Resource Management for Parallel Adaptive Components », *Proceedings of the Workshop on Java for Parallel and Distributed Computing (JPDC) at International Parallel and Distributed Processing Symposium (IPDPS'2003)*, Nice, France, avril 2003, http ://www.univ-ubs.fr/valoria/Composants/CASA/Concerto/papier_IPDPS2003.pdf. 6.5

[Czerwinski et al. 1999] S. E. CZERWINSKI, B. Y. ZHAO, T. D. HODES, A. D. JOSEPH et R. H. KATZ, « An Architecture for a Secure Service Discovery Service », *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, p. 24–35, Seattle, Washington, USA, août 1999, http ://ninja.cs.berkeley.edu/dist/papers/sds-mobicom.pdf. 3.2.1

[Dahm 1999] M. DAHM, « Byte Code Engineering », *Proceedings of Java-Informations-Tage (JIT'99)*, p. 267–277, Düsseldorf, Deutchland, septembre 1999, ftp ://ftp.inf.fu-berlin.de/pub/BCEL/paper.pdf. 5.4

[Davies et al. 1994] N. DAVIES, G. BLAIR, K. CHEVERST et A. FRIDAY, « Supporting Adaptive Services in a Heterogeneous Mobile Environment », *Proceedings of 1st Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, Santa Cruz, USA, décembre 1994, http ://www.comp.lancs.ac.uk/computing/research/mpg/most/reports/mcsa.ps. 2.3.1.3

[Davies et al. 1996] N. DAVIES, A. FRIDAY, G. BLAIR et K. CHEVERST, « Distributed Systems Support for Adaptive Mobile Applications », *ACM Mobile Networks and Applications, Special Issue on Mobile Computing - System Services*, vol. 1, n°4, 1996, p. 399–408, http ://www.comp.lancs.ac.uk/computing/research/mpg/most/reports/ACMNomadSI.ps. 2.3.1.3

[Davies et al. 1997] N. DAVIES, S. WADE, A. FRIDAY et G. BLAIR, « Limbo : A Tuple Space Based Platform for Adaptive Mobile Applications », *Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms (ICODP/ICDP '97)*, p. 291–302, Toronto, Canada, mai 1997, http ://www.comp.lancs.ac.uk/computing/research/mpg/most/reports/icodp97.ps.gz. 3.2.2.5

[Davies et al. 1998] N. DAVIES, A. FRIDAY, S. WADE et G. BLAIR, « L^{2}imbo : A Distributed Systems Platform for Mobile Computing », *Mobile Networks and Applications (MONET)*, vol. 3, n°2, 1998, p. 143–156, ftp ://ftp.comp.lancs.ac.uk/pub/mpg/MPG-97-03.ps.gz. 3.2.2.5

[Decker 2000] T. DECKER, « Virtual Data Space – Load Balancing for Irregular Applications », *Parallel Computing*, vol. 26, n°13–14, décembre 2000, p. 1825–1860, http ://www.uni-paderborn.de/cs/ag-monien/PUBLICATIONS/POSTSCRIPTS/De00_VDS.ps.Z. 3.2.3

[Deering et Hinden 1998] S. DEERING et R. HINDEN, *Internet Protocol, Version 6 (IPv6) Specification*, Request For Comments (RFC) 2460, décembre 1998, statut : « Draft Standard », http ://www.ietf.org/rfc/rfc2460.txt. 2.2, 2.2.1

[Dell] Dell, *Home Page*, http ://www.dell.com. 1.1.1

[Demers et al. 1994] A. J. DEMERS, K. PETERSEN, M. J. SPREITZER, D. B. TERRY, M. M. THEIMER et B. B. WELCH, « The Bayou Architecture : Support for Data Sharing among Mobile Users », *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, p. 2–7, Santa Cruz, California, USA, décembre 1994, http ://www2.parc.com/csl/projects/bayou/pubs/ba-mcw-94/MobileWorkshop.ps.gz. 3.1.3.1

[Dierks et Allen 1999] T. DIERKS et C. ALLEN, *The TLS Protocol Version 1.0*, Request For Comments (RFC) 2246, janvier 1999, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc2246.txt. 2.3.1.2

[d.net] distributed.net, *Home Page*, http ://www.distributed.net. 3.2.2

[dNitto Persone et al. 1998] V. DE NITTO PERSONE, V. GRASSI et A. MORLUPI, « Modeling and Evaluation of Prefetching Policies for Context-Aware Information Services », *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, p. 55–65, Dallas, Texas, USA, octobre 1998. 3.1.2

[Douglis et Ousterhout 1991] F. DOUGLIS et J. OUSTERHOUT, « Transparent Process Migration : Design Alternatives and the Sprite Implementation », *Software-Practice and Experience*, vol. 21, n°8, août 1991, p. 757–785, http ://www.douglis.org/fred/work/papers/mig.ps.gz. 3.2.3

[Droms et al. 2002] R. DROMS, J. BOUND, B. VOLZ, T. LEMON, C. PERKINS et M. CARNEY, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, Internet-Draft, novembre 2002, statut : « expire le 30 avril 2003 », http ://www.ietf.org/internet-drafts/draft-ietf-dhc-dhcpv6-28.txt. 2.2.1.1

[Dube et al. 1997] R. DUBE, C. D. RAIS et S. K. TRIPATHI, « Improving NFS Performance Over Wireless Links », *IEEE Transactions on Computers*, vol. 46, n°3, 1997, p. 290–298, http ://www.cs.umd.edu/projects/mcml/papers/toc97.ps. 2.2.3, 2.3.1.2

[Dumant et al. 1998] B. DUMANT, F. HORN, F. DANG TRAN et J.-B. STÉFANI, « Jonathan : an Open Distributed Processing Environment in Java », *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*, p. 173–190, The Lake District, England, septembre 1998, http ://www.objectweb.org/jonathan/current/doc/hrefs/framework.ps. 2.4

[Edwards et al. 1997] W. K. EDWARDS, E. D. MYNATT, K. PETERSEN, M. J. SPREITZER, D. B. TERRY et M. M. THEIMER, « Designing and Implementing Asynchronous Collaborative Applications with Bayou », *Proceedings of the 10th ACM Symposium on User Interface Software and Technology (UIST'97)*, p. 119–128, Banff, Alberta, Canada, octobre 1997, http ://www2.parc.com/csl/projects/bayou/pubs/uist-97/Bayou.pdf. 3.1.3.1

[ETSI] European Telecommunications Standards Institute, *BRAN Home Page*, http ://www.etsi.org/bran/. 1.2.2

[ETSI 1998] European Telecommunications Standards Institute, *Broadband Radio Access Networks (BRAN) ; HIgh PErformance Radio Local Area Network (HIPERLAN) Type 1 ; Functional specification*, juillet 1998, statut : « Publication, version 1.2.1 », http ://portal.etsi.org/bran/kta/Hiperlan/hiperlan1.asp. 1.2.2

[ETSI 2000] European Telecommunications Standards Institute, *Broadband Radio Access Networks (BRAN) ; HIPERLAN Type 2 ; System Overview*, février 2000, statut : « Publication, version 1.1.1 », http ://portal.etsi.org/bran/kta/Hiperlan/hiperlan2.asp. 1.2.2

[Fabre et Pérennou 1998] J.-C. FABRE et T. PÉRENNOU, « A Metaobject Architecture for Fault-Tolerant Distributed Systems : The FRIENDS Approach », *IEEE Transactions on Computers*, vol. 47, n°1, janvier 1998, p. 78–95, http ://dbserver.laas.fr/pls/LAAS/publis.rech_doc ?langage=FR&clef=19034. 2.4

[Ferguson et Senie 2000] P. FERGUSON et D. SENIE, *Network Ingress Filtering : Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, Request For Comments (RFC) 2827, Best Current Practice (BCP) 0038, mai 2000, statut : « Best Current Practice », ftp ://ftp.isi.edu/in-notes/bcp/bcp38.txt. 2.3.1.1

[Fielding et al. 1999] R. FIELDING, J. GETTYS, J. MOGUL, H. FRYSTYK, L. MASINTER, P. LEACH et T. BERNERS-LEE, *Hypertext Transfer Protocol – HTTP/1.1*, Request For Comments (RFC) 2616, juin 1999, statut : « Draft Standard », ftp ://ftp.isi.edu/in-notes/rfc2616.txt. 2.2, 2.2.4

[FIPA 2002] Foundation for Intelligent Physical Agents, *FIPA Abstract Architecture Specification*, décembre 2002, statut : « Standard, version L », http ://www.fipa.org/specs/fipa00001/. 3.2.2.4

[Floyd et al. 2000] S. FLOYD, J. MAHDAVI, M. MATHIS et M. PODOLSKY, *An Extension to the Selective Acknowledgement (SACK) Option for TCP*, Request For Comments (RFC) 2883, juillet 2000, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc2883.txt. 2.2.2.4

[Folliot et Sens 1994] B. FOLLIOT et P. SENS, « GATOSTAR : A Fault Tolerant Load Sharing Facility for Parallel Applications », *Proceedings of the 1st European Dependable Computing Conference (EDCC'94)*, *Lecture Notes in Computer Science*, vol. 852, p. 581–598, Springer-Verlag, octobre 1994, http ://www-src.lip6.fr/homepages/Pierre.Sens/publications/EDCC1.ps.gz. 3.2.3

[Folliot 1992] B. FOLLIOT, *Méthodes et Outils de Partage de Charge pour la Conception et la Mise en Œuvre d'Applications dans les Systèmes Réparties Hétérogènes*, Thèse de doctorat, MASI laboratory, Paris 6 University, Paris, France, décembre 1992. 3.2.3

[Forman et Zahorjan 1994] G. H. FORMAN et J. ZAHORJAN, « The Challenges of Mobile Computing », *IEEE Computer*, vol. 27, n°4, avril 1994, p. 38–47, http ://citeseer.nj.nec.com/38782.html. (document)

[Foster et Kesselman 1997] I. FOSTER et C. KESSELMAN, « Globus : A Metacomputing Infrastructure Toolkit », *International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, n°2, 1997, p. 115–128, ftp ://ftp.globus.org/pub/globus/papers/globus.pdf. 3.2.2, 7.3

[Fox et al. 1996] A. FOX, S. GRIBBLE, E. A. BREWER et E. AMIR, « Adapting to Network and Client Variability via On-Demand Dynamic Distillation », *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, *SIGPLAN Notices*, vol. 31, n°9, p. 160–170, ACM, Cambridge, Massachusetts, USA, octobre 1996, http ://daedalus.cs.berkeley.edu/publications/adaptive.ps.gz. 3.1.1.2

[Fox et al. 1997] A. FOX, S. D. GRIBBLE, Y. CHAWATHE, E. A. BREWER et P. GAUTHIER, « Cluster-Based Scalable Network Services », *Proceedings of the 16th ACM Symposium on Operating System*

Principles (SOSP'97), *Operating System Review*, vol. 31, n°5, p. 78–91, ACM Press, St. Malo, France, octobre 1997, http ://daedalus.cs.berkeley.edu/publications/sosp16.ps.gz. 3.2.3.1

[Fox et al. 1998a] A. FOX, I. GOLDBERG, S. D. GRIBBLE, D. C. LEE, A. POLITO et E. A. BREWER, « Experience With Top Gun Wingman : A Proxy-Based Graphical Web Browser for the 3Com PalmPilot », *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*, p. 407–424, Lake District, UK, septembre 1998, http ://daedalus.cs.berkeley.edu/publications/wingman.ps.gz. 3.2.3.1

[Fox et al. 1998b] A. FOX, S. D. GRIBBLE, Y. CHAWATHE et E. A. BREWER, « Adapting to Network and Client Variation Using Infrastructural Proxies : Lessons and Perspectives », *IEEE Personal Communications (invited submission)*, vol. 5, n°4, août 1998, p. 10–19, http ://daedalus.cs.berkeley.edu/publications/adapt.ps.gz. 3.2.3.1

[Fox et Brewer 1996] A. FOX et E. A. BREWER, « Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation », *Proceedings of the 5th International World Wide Web Conferences (WWW'5)*, *Computer Networks and ISDN Systems*, vol. 28, n°7–11, p. 1445–1456, Elsevier Science, Paris, France, mai 1996, http ://daedalus.cs.berkeley.edu/publications/www96.ps.gz. 3.1.1.2

[Franklin et Graesser 1996] S. FRANKLIN et A. GRAESSER, « Is it an Agent, or Just a Program ? : A Taxonomy for Autonomous Agents », *Proceedings of the 3rd International Workshop on Agent Theories, Architectures, and Language (ATAL'96)*, *Lecture Notes in Computer Science*, vol. 1193, p. 21–35, Springer Verlag, Budapest, Hungary, août 1996, ftp ://ftp.msci.memphis.edu/comp/caat/agentprog.ps.z. 3.2.2.2

[Friday et al. 1996] A. FRIDAY, G. BLAIR, K. CHEVERST et N. DAVIES, « Extensions to ANSAware for Advanced Mobile Applications », *Proceedings of the 1st International Conference on Distributed Platforms (ICDP'96)*, Dresden, Germany, février 1996, http ://www.comp.lancs.ac.uk/computing/research/mpg/most/reports/ icdp.adrian.ps. 2.3.1.3

[Friday et al. 1999] A. FRIDAY, N. DAVIES, G. BLAIR et K. CHEVERST, « Developing Adaptive Applications : The MOST Experience », *Journal of Integrated Computer-Aided Engineering*, vol. 6, n°2, 1999, p. 143–157, ftp ://ftp.comp.lancs.ac.uk/pub/mpg/MPG-99-10.ps.gz. 2.3.1.3

[Gamma et al. 1995] E. GAMMA, R. HELM, R. JOHNSON et J. VLISSIDES, *Design Patterns*, *Addison Wesley Professional Computing Series*, Addison Wesley, 1995, Addison Wesley Professional Computing Series, http ://www.aw.com. 2.4.0.2, 4.2, 5.2.1, 5.2.2.1, 5.2.3, 7.1.2

[GGF] Jini Activity Working Group – Global Grid Forum, *Home Page*, http ://www-unix.mcs.anl.gov/gridforum/jini/. 3.2.1.1

[Globalstar] Globalstar, *Home Page*, http ://www.globalstar.com. 1.2.2

[Gnutella] Gnutella, *Home Page*, http ://www.gnutella.com. 3.2.2

[Goff et al. 2000] T. GOFF, J. MORONSKI, D. S. PHATAK et V. GUPTA, « Freeze-TCP : A True End-to-End TCP Enhancement Mechanism for Mobile Environments », *Proceedings IEEE INFOCOM 2000*, p. 1537–1545, Tel Aviv, Israel, mars 2000, http ://www.ieee-infocom.org/2000/papers/501.pdf. 2.2.2.3

[Goldberg et Robson 1989] A. GOLDBERG et D. ROBSON, *Smalltalk-80 : The Language*, Addison Wesley, 1989, http ://www.aw.com/catalog/academic/product/0,4096,0201136880,00.html. 4.2

[Golm et Kleinöder 1997] M. GOLM et J. KLEINÖDER, « MetaJava – A Platform for Adaptable Operating-System Mechanisms », *Proceedings of the Workshop on Object-Orientation and Operating Systems at the 11th European Conference on Object-Oriented Programming (ECOOP'97)*, *Lecture Notes in Computer Science*, vol. 1357, p. 507–514, Springer Verlag, Jyväskylä, Finland, juin 1997, http ://www4.informatik.uni-erlangen.de/TR/pdf/TR-I4-97-10.pdf. 2.4

[Golm 1998] M. GOLM, « MetaXa and the Future of Reflection », *Proceedings of the Workshop on Reflective Programming in C++ and Java at the 13th ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'98)*, p. 1–5, Vancouver, Canada, octobre 1998, http ://www4.informatik.uni-erlangen.de/TR/pdf/TR-I4-98-09.pdf. 2.4

[Gomoluch et Schroeder 2001] J. GOMOLUCH et M. SCHROEDER, « Information Agents on the Move : A Survey on Load-Balancing with Mobile Agents », *Software Focus*, vol. 2, n°2, avril 2001, http ://www.soi.city.ac.uk/∼ck655/papers/swfocus.ps. 3.2.3

[Gong 2001] L. GONG, *Project JXTA : A Technical Overview*, Sun Microsystems, avril 2001, http ://www.jxta.org/project/www/docs/TechOverview.pdf. 3.2.2, 7.3

[Govea et Barbeau 2001] J. GOVEA et M. BARBEAU, « Results of Comparing Bandwidth Usage and Latency : Service Location Protocol and Jini », *Proceedings of the Workshop on Ad hoc Communications, In conjunction with the 7th European Conference on Computer Supported Cooperative Work (ECSCW 2001)*, Bonn, Germany, septembre 2001, http ://www.scs.carleton.ca/∼barbeau/Publications/2001/WAHC/govea.pdf. 3.2.1.2

[Gowing et Cahill 1996] B. GOWING et V. CAHILL, « Meta-Object Protocols for C++ : The Iguana Approach », *Proceedings of 1st International Conference on Meta-Level Architectures and Reflection (Reflection'96)*, p. 137–152, San Francisco, California, USA, avril 1996, ftp ://ftp.dsg.cs.tcd.ie/pub/doc/dsg-97.ps.gz. 2.4

[Gray et al. 1996] R. S. GRAY, D. KOTZ, S. NOG, D. RUS et G. CYBENKO, *Mobile agents for mobile computing*, rapport technique n°PCS-TR96-285, Department of Computer Science, Dartmouth College, Hanover, New Hampshire, USA, mai 1996, ftp ://ftp.cs.dartmouth.edu/TR/TR96-285.pdf. 3.2.2.3

[Gray et al. 2000] R. S. GRAY, G. CYBENKO, D. KOTZ et D. RUS, *Mobile agents : Motivations and State of the Art*, rapport technique n°TR2000-365, Department of Computer Science, Dartmouth College, Hanover, New Hampshire, USA, avril 2000, ftp ://ftp.cs.dartmouth.edu/TR/TR2000-365.pdf. A

[Gray et al. 2001a] R. S. GRAY, G. CYBENKO, D. KOTZ et D. RUS, *Mobile agents : Motivations and State of the Art*, In Bradshaw [Bradshaw], 2001, publié aussi dans [Gray et al. 2000]. 3.2.2

[Gray et al. 2001b] R. S. GRAY, D. KOTZ, R. A. PETERSON JR., J. BARTON, D. CHACÓN, P. GERKEN, M. HOFMANN, J. BRADSHAW, M. BREEDY, R. JEFFERS et N. SURI, « Mobile-Agent versus Client/Server Performance : Scalability in an Information-Retrieval Task », *Proceedings of Mobile Agents, 5th International Conference (MA'2001)*, *Lecture Notes in Computer Science*, vol. 2240, p. 229–243, Springer Verlag, Atlanta, Georgia, USA, décembre 2001, http ://agent.cs.dartmouth.edu/papers/gray :scalability.pdf. 3.2.2.2

[Gray et Cheriton 1989] C. G. GRAY et D. R. CHERITON, « Leases : An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency », *Proceedings of the 12th ACM Symposium on Operating System Principles (SOSP'89)*, p. 202–210, The Wigwam, Litchfield Park, Arizona, USA, décembre 1989, http ://www.acm.org/pubs/articles/proceedings/ops/74850/p202-gray/p202-gray.pdf. 2.2.3.1

[Gray 1997] R. S. GRAY, *Agent Tcl : A flexible and secure mobile-agent system*, Thèse de doctorat, Dartmouth College, Hanover, New Hampshire, USA, juin 1997, ftp ://ftp.cs.dartmouth.edu/TR/TR98-327.pdf. 5

[Gray 2000] R. S. GRAY, « Soldiers, Agents and Wireless Networks : A Report on a Military Application », *Proceedings of the 5th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'2000)*, Manchester, England, avril 2000, http ://actcomm.dartmouth.edu/papers/gray :overview.pdf. 3.2.2.3

[Grimshaw et al. 1997] A. S. GRIMSHAW, WM. A. WULF et THE WHOLE LEGION TEAM, « The Legion Vision of a Worldwide Virtual Computer », *Communication of the ACM*, vol. 40, n°1, janvier 1997, p. 39–45, http ://www.cs.virginia.edu/∼legion/papers/cacm.ps. 3.2.3

[Grinberg et al. 1995] D. GRINBERG, S. RAJAGOPALAN, R. VENKATESAN et V. K. WEI, « Splay Trees for Data Compression », *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, p. 522–530, San Francisco, California, USA, janvier 1995, http ://www.cs.cmu.edu/afs/cs.cmu.edu/user/dennis/www/papers/ps/soda95.ps. 3.1.1.1

[Guttman et al. 1999] E. GUTTMAN, C. PERKINS, J. VEIZADES et M. DAY, *Service Location Protocol, Version 2*, Request For Comments (RFC) 2608, juin 1999, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc2608.txt. 3.2.1.2

[Guttman 1999] E. GUTTMAN, « Service Location Protocol : Automatic Discovery of IP Network Services », *IEEE Internet Computing*, vol. 3, n°4, juillet–août 1999, p. 71–80, http ://www.srvloc.org/slp-article.pdf. 3.2.1.2

[Guy et al. 1998] R. G. GUY, P. L. REIHER, D. RATNER, M. GUNTER, W. MA et G. J. POPEK, « Rumor : Mobile Data Access Through Optimistic Peer-to-Peer Replication », *Proceedings of the ER'98 Workshops on Data Warehousing and Data Mining, Mobile Data Access, and Collaborative Work Support and Spatio-Temporal Data Management*, *Lecture Notes in Computer Science*, vol. 1552, p. 254–265, Springer Verlag, Singapore, novembre 1998, http ://fmg-www.cs.ucla.edu/rumor98/er98wmda.ps. 3.1.3.2

[Guyennet et al. 1997] H. GUYENNET, B. HERRMANN, F. SPIES et L. PHILIPPE, « A Comparison Study of Dynamic Load Balancing Algorithms », *International Journal of Mini and Microcomputers*, vol. 19, n°3, 1997, p. 70–77. 3.2.3

[Haas 1997] Z. J. HAAS, « Mobile-TCP : An Asymmetric Transport Protocol Design for Mobile Systems », *Proceedings of the International Conference on Communications (ICC'97)*, p. 1054–1058, Montreal, Canada, juin 1997, http ://www.ee.cornell.edu/∼haas/Publications/mmc.ps. 2.2.2.1

[Handspring] Handspring, *Home Page*, http ://www.handspring.com. 1.1.1

[Hansen et al. 1996] J. S. HANSEN, T. REICH et B. ANDERSEN, « Semi-Connected TCP/IP in a Mobile Computing Environment », *The International Workshop for Information Visualization and Mobile Computing (IMC'96)*, Rostock, Germany, février 1996, ftp ://ftp.diku.dk/pub/diku/users/cyller/taco/imc96paper.ps.gz. 2.2.2.1

[Hansen et Reich 1996] J. S. HANSEN et T. REICH, *Semi-Connected TCP/IP in a Mobile Computing Environment*, Thèse de Master, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, juin 1996, ftp ://ftp.diku.dk/pub/diku/users/cyller/taco/diku95-6-11.ps.gz. 2.2.2.1

[Harter et Hopper 1994] A. HARTER et A. HOPPER, « A Distributed Location System for the Active Office », *IEEE Network*, vol. 8, n°1, janvier 1994, p. 62–70, ftp.orl.co.uk :/pub/docs/ORL/tr.94.1.ps.Z. 1.2.2

[Helal et al. 1999] A. S. HELAL, B. HASKELL, J. L. CARTER, R. BRICE, D. WOELK et M. RUSINKIEWICZ, *Any Time, Anywhere Computing*, Kluwer Academic Publishers, 1999, http ://www.wkap.nl. 1.2.1.1

[Helin et al. 1999] H. HELIN, H. LAAMANEN et K. RAATIKAINEN, « Mobile Agent Communication in Wireless Networks », *Proceedings of Wireless'99/ITG'99*, p. 211–216, Munich, Germany, octobre 1999, http ://www.cs.helsinki.fi/research/monads/papers/wireless99/agent_comm.pdf. 3.2.2.2

[Hohl et al. 1999] F. HOHL, U. KUBACH, A. LEONHARDI, K. ROTHERMEL et M. SCHWEHM, « Next Century Challenges : Nexus – An Open Global Infrastructure for Spatial-Aware Applications », *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile*

Computing and Networking (MobiCom'99), p. 249–255, Seattle, Washington, USA, août 1999, http ://www.informatik.uni-stuttgart.de/ipvr/vs/Publications/1999-hohlEA-01.ps.gz. 3.1.2.3

[Hokimoto et al. 1996] A. HOKIMOTO, K. KURIHARA et T. NAKAJIMA, « An Approach for Constructing Mobile Applications using Service Proxies », *Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS'96)*, p. 726–733, Wanchai, Hong Kong, mai 1996, http ://mmmc.jaist.ac.jp :8000/publications/1996/PostScript/dcs96-hokimoto.ps.gz. 3.2.2

[Holland et Vaidya 1999] G. HOLLAND et N. H. VAIDYA, « Analysis of TCP Performance over Mobile Ad Hoc Networks », *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, p. 219–230, Seattle, Washington, USA, août 1999, http ://www.acm.org/pubs/articles/proceedings/comm/313451/p219-holland/p219-holland.pdf. 2.2.2

[Hollingsworth et Miller 1993] J. K. HOLLINGSWORTH et B. P. MILLER, « Dynamic Control of Performance Monitoring on Large Scale Parallel Systems », *Proceedings of the 7th ACM International Conference on Supercomputing (ICS'93)*, p. 185–194, Tokyo, Japan, juillet 1993, http ://www.cs.umd.edu/∼hollings/papers/ics93.pdf. 1

[Housel et al. 1998] B. C. HOUSEL, G. SAMARAS et D. B. LINDQUIST, « WebExpress : A Client/Intercept Based System for Optimizing Web Browsing in a Wireless Environment », *Mobile Networks and Applications (MONET)*, vol. 3, n°4, 1998, p. 419–431, http ://www.acm.org/pubs/articles/journals/monet/1999-3-4/p419-housel/p419-housel.pdf. 2.2.4.1

[Housel et Lindquist 1996] B. C. HOUSEL et D. B. LINDQUIST, « WebExpress : A System for Optimizing Web Browsing in a Wireless Environment », *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking (MobiCom'96)*, p. 108–116, Rye, New York, USA, novembre 1996, http ://www.acm.org/pubs/articles/proceedings/comm/236387/p108-housel/p108-housel.pdf. 2.2.4, 2.2.4.1

[HP] Hewlett-Packard, *Home Page*, http ://www.hp.com. 1.1.1

[Hupfer 1990] S. C. HUPFER, *Melinda : Linda with Multiple Tuple Space*, rapport technique n°YALE/DCS/RR-766, Department of Computer Science, Yale University, Connecticut, USA, février 1990, http ://www.cs.yale.edu/Linda/tech-reports.html. 3.2.2.5

[IBM] IBM, *Home Page*, http ://www.ibm.com. 1.1.1

[IEEE LAN] IEEE Working Group for CSMA/CD (Ethernet) based LANs Standards, *Home Page*, http ://grouper.ieee.org/groups/802/3/. 1.2.3

[IEEE WLAN] IEEE Working Group for WLAN Standards, *Home Page*, http ://grouper.ieee.org/groups/802/11/. 1.2.2

[IEEE 1995] IEEE, *IEEE 1394 Standard for a High Performance Serial Bus*, 1995, http ://standards.ieee.org/catalog/olis/busarch.html. 1.1.1

[IEEE 1999a] IEEE, *IEEE Std 802.11a-1999, Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications : High-speed Physical Layer in the 5 GHz Band*, 1999, statut : « Supplement to IEEE Std 802.11-1999, Adopted by ISO/IEC and redesignated as ISO/IEC 8802-11 :1999/Amd 1 :2000(E) », http ://a957.g.akamai.net/7/957/3680/v0001/standards.ieee.org/reading/ieee/std/lanman/802.11a-1999.pdf. 1.2.2

[IEEE 1999b] IEEE, *IEEE Std 802.11b-1999, Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications : Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, 1999, statut : « Supplement to IEEE Std 802.11-1999 », http ://a957.g.akamai.net/7/957/3680/v0001/standards.ieee.org/reading/ieee/std/lanman/802.11b-1999.pdf. 1.2.2

[IEEE 1999c] IEEE, *IEEE Std 802.11-1999, Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Édition 1999, statut : « Adopted by the ISO/IEC and redesignated as ISO/IEC 8802-11 :1999(E) », http ://a957.g.akamai.net/7/957/3680/v0001/standards.ieee.org/reading/ieee/std/lanman/802.11-1999.pdf. 1.2.2

[IEEE 2000] IEEE, *IEEE 1394a Standard for a High Performance Serial Bus – Amendment 1*, 2000, http ://standards.ieee.org/catalog/olis/busarch.html. 1.1.1

[IETF] IETF Routing Working Group : Mobile Ad-hoc Networks (MANET), *Home Page*, http ://www.ietf.org/html.charters/manet-charter.html. 1.2.1.2

[Imielinski et Korth 1996] T. IMIELINSKI et H. F. KORTH (éd.), *Mobile Computing*, *The Kluwer International Series in Engineering and Computer Science*, vol. 353, Kluwer Academic Publishers, février 1996, http ://www.wkap.nl. 1.1.2, A

[Ioannidis et al. 1991] J. IOANNIDIS, D. DUCHAMP et G. Q. MAGUIRE JR, « IP-Based Protocols for Mobile Internetworking », *Proceedings of the Conference on Communications Architecture & Protocols (SIGCOMM'91)*, p. 235–245, Zürich, Switzerland, septembre 1991, http ://www.acm.org/pubs/articles/proceedings/comm/115992/p235-ioannidis/p235-ioannidis.pdf. 2.2.1

[IrDA 1996a] Infrared Data Association, *Link Management Protocol*, janvier 1996, statut : « Version 1.1 », http ://www.irda.org/standards/pubs/IrData.zip. 1.2.2

[IrDA 1996b] Infrared Data Association, *Serial Infrared Link Access Protocol (IrLAP)*, juin 1996, statut : « Version 1.1, complément et errata du 5 janvier 1999 », http ://www.irda.org/standards/pubs/IrData.zip. 1.2.2

[IrDA 1998] Infrared Data Association, *IrDA Control Specification (Formerly IrBus) IrDA CIR (Control IR) Standard*, juin 1998, statut : « Version 1.0, errata du 26 octobre 1999 », http ://www.irda.org/standards/pubs/Irda_ControlV1p0E.zip. 1.2.2

[IrDA 2001] Infrared Data Association, *Serial Infrared Physical Layer Specification*, mai 2001, statut : « Version 1.4 », http ://www.irda.org/standards/pubs/IrData.zip. 1.2.2

[ISO 1986] International Organization for Standardization, Geneva, Switzerland, *ISO 8879 :1986 : Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, 1986, statut : « International Standard confirmed 13-08-2001 », http ://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail ?CSNUMBER=16387. 2.2.4

[ISO 1994] International Organization for Standardization, Geneva, Switzerland, *ISO/IEC 10918-1 :1994 : Information technology – Digital compression and coding of continuous-tone still images : Requirements and guidelines*, 1994, statut : « International Standard confirmed 21-01-2000 », http ://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail ?CSNUMBER=18902 &ICS1=35&ICS2=40&ICS3=. 3.1.1

[ISO 1996a] International Organization for Standardization, Geneva, Switzerland, *ISO/IEC 10746-2 :1996 : Information technology – Open Distributed Processing – Reference Model : Foundations*, 1996, statut : « International Standard confirmed 13-08-2001 », http ://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail ?CSNUMBER=18836 &ICS1=35&ICS2=80&ICS3=. 2.3.1.3

[ISO 1996b] International Organization for Standardization, Geneva, Switzerland, *ISO/IEC 10746-3 :1996 : Information technology – Open Distributed Processing – Reference Model : Architecture*, 1996, statut : « International Standard confirmed 13-08-2001 », http ://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail ?CSNUMBER=20697 &ICS1=35&ICS2=80&ICS3=. 2.3.1.3

[ISO 1998a] International Organization for Standardization, Geneva, Switzerland, *ISO/IEC 10746-1 :1998 : Information technology – Open Distributed Processing – Reference model : Overview*, 1998, statut : « International Standard under periodical review 27-02-2003 », http ://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail ?CSNUMBER=20696 &ICS1=35&ICS2=80&ICS3=. 2.3.1.3

[ISO 1998b] International Organization for Standardization, Geneva, Switzerland, *ISO/IEC 10746-4 :1998 : Information technology – Open Distributed Processing – Reference Model : Architectural semantics*, 1998, statut : « International Standard under periodical review 27-02-2003 », http ://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail ?CSNUMBER=20698 &ICS1=35&ICS2=80&ICS3=. 2.3.1.3

[Itoh et al. 1995] J.-I. ITOH, R. LEA et Y. YOKOTE, « Using meta-objects to support optimization in the Apertos operating system », *Proceedings of the USENIX Conference on Object-Oriented Technologies (COOTS'95)*, Monterey, California, USA, juin 1995, ftp ://ftp.cs.keio.ac.jp/pub/keio-cs-papers/mt/oops/1995/itojun-coots.ps.Z. 2.4

[ITU] ITU, *IMT-2000 Home Page*, http ://www.itu.int/imt/index.html. 1.2.2

[Jannink et al. 1997] J. JANNINK, D. LAM, N. SHIVAKUMAR, J. WIDOM et D. COX, « Efficient and Flexible Location Management Techniques for Wireless Communication Systems », *ACM/Balzer Journal of Wireless Networks (WINET)*, vol. 3, n°5, 1997, p. 361–374, http ://dbpubs.stanford.edu/pub/showDoc.Fulltext ?lang=en&doc=1997-60&format=ps&compression=gz. 2.2.1.1

[Jansen et al. 2001] M. JANSEN, E. KLAVER, P. VERKAIK, M. VAN STEEN et A.S. TANENBAUM, « Encapsulating Distribution in Remote Objects », *Information and Software Technology*, vol. 43, n°6, mai 2001, p. 353–363, http ://www.cs.vu.nl/pub/papers/globe/infsof.2001.pdf. 3.2.3

[Jennings et al. 1998] N. R. JENNINGS, K. P. SYCARA et M. WOOLDRIDGE, « A Roadmap of Agent Research and Development », *Autonomous Agents and Multi-Agent Systems*, vol. 1, n°1, 1998, p. 7–38, http ://www.ecs.soton.ac.uk/∼nrj/download-files/roadmap.pdf. 3.2.2.2

[Jeong 1996] K. JEONG, *Fault-tolerant Parallel Processing Combining Linda, Checkpointing, and Transactions*, Thèse de doctorat, Department of Computer Science, New York University, New York, USA, janvier 1996, http ://www.cs.nyu.edu/phd_students/binli/plinda/thesis-karp.ps. 3.2.2.4

[Jin et al. 1999] K. JIN, K. KIM et J. LEE, « SPACK : rapid recovery of the TCP performance using SPlit-ACK in mobile communication environments », *Proceedings of the IEEE TenCon'99*, p. 761–774, Cheju, Korea, septembre 1999. 2.2.2.3

[Jing et al. 1999] J. JING, A. HELAL et A. K. ELMAGARMID, « Client-Server Computing in Mobile Environments », *ACM Computing Surveys*, vol. 31, n°2, juin 1999, p. 117–157, http ://citeseer.nj.nec.com/412370.html. 3.2.2

[Johansen et al. 2002] D. JOHANSEN, K. J. LAUVSET, R. VAN RENESSE, F. B. SCHNEIDER, N. P. SUDMANN et K. JACOBSEN, « A TACOMA Retrospective », *Software – Practice and Experience*, vol. 32, n°6, mai 2002, p. 605–619, http ://www.cs.cornell.edu/fbs/publications/tacomaRetroSPE.ps. 3.2.2

[Joseph et al. 1997] A. D. JOSEPH, J. A. TAUBER et M. F. KAASHOEK, « Mobile Computing with the Rover Toolkit », *IEEE Transactions on Computers : Special issue on Mobile Computing*, vol. 46, n°3, mars 1997, p. 337–352, http ://www.pdos.lcs.mit.edu/papers/toc.ps.gz. 3.2.2.1

[Joseph et Kaashoek 1997] A. D. JOSEPH et M. F. KAASHOEK, « Building Reliable Mobile-Aware Applications using the Rover Toolkit », *Wireless Networks*, vol. 3, n°5, octobre 1997, p. 405–419, http ://www.pdos.lcs.mit.edu/papers/winet.ps.gz. 3.2.2.1

[Joshi et al. 1996] A. JOSHI, R. WEERASINGHE, S. MC.DERMOTT, B. TAN, G. BERNHARDT et S. WEERAWARANA, « Mowser : Mobile Platforms and Web Browsers », *Bulletin of the Technical Committee on Operating Systems and Application Environments (TCOS)*, vol. 8, n°1, 1996, http ://www.tcos.org/Bulletin/spring96/joshi.ps. 3.1.1.3

[Joshi et al. 1997] A. JOSHI, S. WEERAWARANA et E. HOUSTIS, « On Disconnected Browsing of Distributed Information », *Proceedings of the 7th IEEE International Workshop on Research Issues in Data Engineering (RIDE'97)*, p. 101–108, Birmingham, England, avril 1997, http ://www.cs.umbc.edu/∼joshi/resch/ride97.ps.gz. 3.1.1.3

[Jouppi 1990] N. P. JOUPPI, « Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers », *Proceedings of the 17th Annual International Symposium on Computer Architecture*, p. 364–373, Seattle, Massachusettes, USA, juin 1990, ftp ://ftp.digital.com/pub/Digital/WRL/research-reports/WRL-TN-14.ps.gz. 3.1.2

[Katz et al. 1996] R. H. KATZ, E. A. BREWER, E. AMIR, H. BALAKRISHNAN, A. FOX, S. GRIBBLE, T. D. HODES, D. JIANG, G. T. NGUYEN, V. N. PADMANABHAN et M. STEMM, « The Bay Area Research Wireless Access Network (BARWAN) », *Proceedings of the 41st IEEE Computer Society International Conference : Technologies for the Information Superhighway (COMPCON'96)*, p. 15–20, Santa Clara, California, USA, février 1996, http ://daedalus.cs.berkeley.edu/publications/Compcon9.ps.gz. 3.1.1.2

[Katz et Brewer 1996] R. H. KATZ et E. A. BREWER, « The Case for Wireless Overlay Networks », *Proceedings of the SPIE Multimedia and Networking Conference (MMNC'96)*, San Jose, California, USA, janvier 1996, http ://daedalus.cs.berkeley.edu/publications/SPIE96.ps.gz. 3.1.1.2

[Keleher et Çetintemel 2000] P. J. KELEHER et U. ÇETINTEMEL, « Consistency management in Deno », *Mobile Networks and Applications (MONET)*, vol. 5, n°4, 2000, p. 299–309, http ://www.cs.umd.edu/∼keleher/papers/monet99.pdf. 3.1.3.3

[Keleher 1999] P. J. KELEHER, « Decentalized Replicated-Object Protocols », *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC'99)*, p. 143–151, Atlanta, Georgia, USA, mai 1999, http ://www.cs.umd.edu/∼keleher/papers/podc99.pdf. 3.1.3.3

[Keshav 1991] S. KESHAV, « A Control-Theoretic Approach to Flow Control », *Proceedings of the Conference on Communications Architecture & Protocols (SIGCOMM'91)*, p. 3–15, Zürich, Switzerland, septembre 1991, http ://www.acm.org/pubs/articles/proceedings/comm/115992/p3-keshav/p3-keshav.pdf. 2.2.2.4

[Khare et Lawrence 2000] R. KHARE et S. LAWRENCE, *Upgrading to TLS Within HTTP/1.1*, Request For Comments (RFC) 2817, mai 2000, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc2817.txt. 2.2, 2.2.4

[Kiczales et al. 1991] G. KICZALES, J. DES RIVIERES et D. G. BOBROW, *The Art of the Metaobject Protocol*, MIT-Press, juillet 1991, http ://mitpress.mit.edu. 2.4

[Killijian et Fabre 2000] M.-O. KILLIJIAN et J.-C. FABRE, « Implementing a Reflective Fault-Tolerant CORBA System », *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'2000)*, p. 154–163, Nürnberg, Germany, octobre 2000, http ://dbserver.laas.fr/pls/LAAS/publis.rech_doc ?langage=FR&clef=39073. 2.4

[Kirtland 1997] M. KIRTLAND, « The COM+ Programming Model Makes it Easy to Write Components in Any Language », *Microsoft Systems Journal*, décembre 1997, http ://www.microsoft.com/msj/1297/complus2/complus2.htm. 3.2.2

[Kistler et Satyanarayanan 1992] J.J. KISTLER et M. SATYANARAYANAN, « Disconnected Operation in the Coda File System », *ACM Transactions On Computer Systems*, vol. 10, n°1, février 1992, p. 3–25, http ://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/s13.pdf. 1

[Kleinöder et Golm 1996a] J. KLEINÖDER et M. GOLM, « MetaJava : An Efficient Run-Time Meta Architecture for Java », *Proceedings of the International Workshop on Object Orientation in Operating Systems (IWOOOS'96)*, Seattle, Washington, USA, octobre 1996, http ://www4.informatik.uni-erlangen.de/TR/pdf/TR-I4-96-03.pdf. 2.4

[Kleinöder et Golm 1996b] J. KLEINÖDER et M. GOLM, *Transparent and Adaptable Object Replication Using a Reflective Java*, rapport technique n°TR-I4-96-07, Friedrich-Alexander-University, Erlangen-Nürnberg, Germany, septembre 1996, http ://www4.informatik.uni-erlangen.de/TR/pdf/TR-I4-96-07.pdf. 2.4

[Kojo et al. 1996] M. KOJO, K. RAATIKAINEN et T. ALANKO, *Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network*, p. 253–270, Volume 353 Imielinski et Korth [Imielinski et Korth 1996], février 1996, ftp ://ftp.cs.helsinki.fi/pub/Reports/by_Title/Connecting_Mobile_Workstations_to_the_Internet_over_a_Digital_Cellular_Telephone_Network.ps.gz. 3.1.1.1

[Kon et al. 1999] F. KON, R. H. CAMPBELL, B. SRINIVASAN, R. CHANDRA et A. VISWANATHAN, *Dynamic Reconfiguration of Scalable Internet Systems with Mobile Agents*, rapport technique n°UIUCDCS-R-99-2105, Department of Computer Science, University of Illinois at Urbana-Champaign, USA, mars 1999, http ://choices.cs.uiuc.edu/2k/papers/TR/config-agents.ps.gz. 2.4.0.2

[Kon et al. 2000a] F. KON, B. GILL, R. H. CAMPBELL et M. D. MICKUNAS, « Secure Dynamic Reconfiguration of Scalable CORBA Systems with Mobile Agents », *Proceedings of the IEEE Joint Symposium on Agent Systems and Applications / Mobile Agents (ASA/MA'2000)*, *Lecture Notes in Computer Science*, vol. 1882, p. 86–98, Springer Verlag, Zürch, Switzerland, septembre 2000, http ://choices.cs.uiuc.edu/2k/papers/asama2000.pdf. 2.4.0.2

[Kon et al. 2000b] F. KON, M. ROMÁN, P. LIU, J. MAO, T. YAMANE, L. C. MAGALHÃES et R. H. CAMPBELL, « Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB », *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'2000)*, p. 121–143, New York, USA, avril 2000, http ://choices.cs.uiuc.edu/2k/papers/middleware2000.pdf. 2.4, 2.4.0.2

[Kon et Campbell 2000] F. KON et R. CAMPBELL, « Dependence Management in Component-Based Distributed Systems », *IEEE Concurrency*, vol. 8, n°1, janvier–mars 2000, p. 26–36, http ://choices.cs.uiuc.edu/2k/papers/DependenceManagement-concurrency.pdf. 2.4.0.2

[Kon et Mandel 1995] F. KON et A. MANDEL, « SODA : A Lease-Based Consistent Distributed File System », *Proceedings of the 13th Brazilian Symposium on Computer Networks*, Belo Horizonte, Brazil, mai 1995, ftp ://ftp.ime.usp.br/pub/reports/comp/rt-mac-9303.ps.gz. 2.2.3.1

[Kotz et al. 2002] D. KOTZ, R. S. GRAY et D. RUS, *Future Directions for Mobile-Agent Research*, rapport technique n°TR2002-415, Department of Computer Science, Dartmouth College, Hanover, New Hampshire, USA, janvier 2002, ftp ://ftp.cs.dartmouth.edu/TR/TR2002-415.pdf. 3.2.2

[Kotz et Gray 1999] D. KOTZ et R. S. GRAY, « Mobile Agents and the Future of the Internet », *Operating Systems Review*, vol. 33, n°3, juillet 1999, p. 7–13, ftp ://ftp.cs.dartmouth.edu/pub/kotz/papers/kotz :future2.pdf. 3.2.2.2, 3.2.2.3

[Kubach et Rothermel 2001] U. KUBACH et K. ROTHERMEL, « Exploiting Location Information for Infostation-Based Hoarding », *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'2001)*, p. 15–27, Rome, Italy, juillet 2001, http ://www.informatik.uni-stuttgart.de/ipvr/vs/Publications/2001-kubach-02.pdf. 3.1.2.3

[Kuenning et al. 1997] G. H. KUENNING, P. REIHER et G. J. POPEK, « Experience with an Automated Hoarding System », *Personal Technologies*, vol. 1, n°3, septembre 1997, p. 145–155, ftp ://ftp.cs.ucla.edu/pub/ficus/geoff/perstech97.ps.gz. 3.1.3.2

[Kuenning et Popek 1997] G. H. KUENNING et G. J. POPEK, « Automated Hoarding for Mobile Computers », *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP'16)*, p. 264–275, Saint-Malo, France, octobre 1997, ftp ://ftp.cs.ucla.edu/pub/ficus/geoff/sosp97.ps.gz. 3.1.2, 3.1.3.2

[Kuenning 1997] G. H. KUENNING, *Seer : Predictive File Hoarding for Disconnected Mobile Operation*, Thèse de doctorat, University of California, Los Angeles, California, USA, mai 1997, ftp ://ftp.cs.ucla.edu/pub/ficus/geoff/kuenning_dissertation.ps.gz. 3.1.3.2

[Kumar et Satyanarayanan 1995] P. KUMAR et M. SATYANARAYANAN, « Flexible and Safe Resolution of File Conflicts », *Proceedings of the USENIX Winter 1995 Technical Conference*, p. 95–106, New Orleans, Louisiana, USA, janvier 1995, http ://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/usenix95.pdf. 2.2.3.3, 3.1.2.1

[Lampson 1986] B. LAMPSON, « Designing a Global Name Service », *Proceedings of the 5th Annual ACM Symposium on Principles of Distributed Computing (PODC'86)*, p. 1–10, Calgary, Alberta, Canada, août 1986, http ://research.microsoft.com/∼lampson/36-globalnames/Acrobat.pdf. 3.2.1

[Le et al. 1994] M. T. LE, S. SESHAN, F. BURGHARDT et J. RABAEY, « Software Architecture of the Infopad System », *Proceedings of the Mobidata Workshop on Mobile and Wireless Information Systems*, Rutgers, New Jersey, USA, novembre 1994, http ://bwrc.eecs.berkeley.edu/Publications/1994/Presentations/infopad_software_arch.mobidata/infopad_software_arch.mobidata.ps.gz. 3.2.2

[Le Mouël et al. 2000] F. LE MOUËL, M.T. SEGARRA et F. ANDRÉ, « Improving Mobile Computing Performance by Using an Adaptive Distribution Framework », *Proceedings of 7th International Conference on High Performance Computing (HiPC'2000)*, *Lecture Notes in Computer Science*, vol. 1970, p. 479–488, Springer Verlag, Bangalore, India, décembre 2000, http ://www.irisa.fr/solidor/doc/ps00/HiPC2000.ps.gz. 7.3

[Le Mouël et al. 2002] F. LE MOUËL, F. ANDRÉ et M.T. SEGARRA, « AeDEn : An Adaptive Framework for Dynamic Distribution over Mobile Environments », *Annales des Télécommunications*, vol. 57, n°11–12, novembre–décembre 2002, p. 1124–1148. 4.3.1, 7, 7.3

[Le Mouël et André 2000a] F. LE MOUËL et F. ANDRÉ, « AeDEn : un cadre général pour une distribution adaptative et dynamique des applications en environnements mobiles », *Actes du 3ème Colloque International sur les NOuvelles TEchnologies de la REpartition (NOTERE'2000)*, p. 171–182, Paris, France, novembre 2000, http ://www.irisa.fr/solidor/doc/ps00/NOTERE2000.ps.gz. 7.3, A

[Le Mouël et André 2000b] F. LE MOUËL et F. ANDRÉ, « Distribution over Mobile Environments », *Proceedings of 2000 ACM Symposium on Applied Computing (SAC'2000)*, p. 568–569, Villa Olmo, Como, Italy, mars 2000, http ://www.irisa.fr/solidor/doc/ps00/SAC2000.ps.gz. 7.3

[Le Mouël et André 2001] F. LE MOUËL et F. ANDRÉ, « AeDEn : un cadre général pour une distribution adaptative et dynamique des applications en environnements mobiles », *Revue Électronique sur les Réseaux et l'Informatique Répartie (RERIR)*, vol. 11, mars 2001, p. 169–181, tiré de [Le Mouël et André 2000a], http ://www.irisa.fr/solidor/doc/abstr01/RERIR2001.html. 7.3

[Le Mouël 1999] F. LE MOUËL, « Amélioration du niveau de service par la distribution adaptative d'applications dans un environnement mobile », *Journée Jeunes Chercheurs en Systèmes (JCS'99), dans les actes de la 1ère Conférence Française sur les Systèmes d'Exploitation (CFSE'1)*, p. 229–232, Rennes, France, juin 1999, http ://www.irisa.fr/solidor/doc/ps99/JCS99.ps.gz. 7.3

[Ledoux 1999] T. LEDOUX, « OpenCorba : a Reflective Open Broker », *Proceedings of the 2nd International Conference on Meta-level Architectures and Reflection (Reflection'99)*, *Lecture Notes in Computer Science*, vol. 1616, p. 197–214, Springer Verlag, Saint-Malo, France, juillet 1999, http ://www.emn.fr/cs/object/biblio/publications/reflection99.pdf.gz. 2.4

[Lehman et al. 1999] T. J. LEHMAN, S. W. MCLAUGHRY et P. WYCKOFF, « T Spaces : The Next Wave », *Proceedinsg of the 32nd Hawaii International Conference on System Sciences (HICSS'99)*, Island of Maui, Hawaii, USA, janvier 1999, http ://www.almaden.ibm.com/cs/Tspaces/papers/Cluster.ps.Z. 3.2.2

[Lei et Duchamp 1997] H. LEI et D. DUCHAMP, « An Analytical Approach to File Prefetching », *Proceedings of the USENIX 1997 Annual Technical Conference*, p. 275–288, Anaheim, California, USA, janvier 1997, http ://guinness.cs.stevens-tech.edu/∼djd/collected-papers/usenix97-prefetch.ps. 3.1.2

[Lenglet 2002] R. LENGLET, *Jabyce : A Java Framework for Bytecode Adaptation*, Présentation réunion ARCAD, mai 2002, http ://arcad.essi.fr/020524/Jabyce-2002-05-23.ppt. 5.4

[Liljeberg et al. 1996] M. LILJEBERG, H. HELIN, M. KOJO et K. RAATIKAINEN, « Enhanced Services for World-Wide Web in Mobile WAN Environment », *Proceedings of the IEEE Global Internet 1996 Conference*, London, England, novembre 1996, ftp ://ftp.cs.helsinki.fi/pub/Reports/by_Title/Enhanced_Services_for_World-Wide_Web_in_Mobile_WAN_Environment.ps.gz. 2.2.4, 3.1.1.1

[Litzkow et Livny 1990] M. J. LITZKOW et M. LIVNY, « Experience With The Condor Distributed Batch System », *Proceedings of the IEEE Workshop on Experimental Distributed Systems*, p. 97–101, Huntsville, Alabama, USA, octobre 1990, http ://www.cs.wisc.edu/condor/doc/experience.ps. 3.2.3

[Liu et al. 1995] G. LIU, A. MARLEVI et G. Q. MAGUIRE JR, « A Mobile Virtual-Distributed System Architecture for Supporting Wireless Mobile Computing and Communications », *Proceedings of the 1st Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'95)*, p. 111–118, Berkeley, California, USA, novembre 1995, http ://citeseer.nj.nec.com/liu95virtual.html. 3.1.2

[LoL] Linux on Laptops, *Running Linux on Palmtops*, http ://www.linux-laptop.net/palmtops.html. 1.1.1

[Long et al. 1996] S. LONG, R. KOOPER, G. D. ABOWD et C. G. ATKESON, « Rapid Prototyping of Mobile Context-Aware Applications : The Cyberguide Case Study », *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking (MobiCom'96)*, p. 97–107, Rye, New York, USA, novembre 1996, http ://www.cc.gatech.edu/fce/cyberguide/pubs/mobicom96-cyberguide.ps. 3.1.2

[Lui et al. 1998] J. C. S. LUI, O. K. Y. SO et T. S. TAM, « NFS/M : An Open Platform Mobile File System », *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, p. 488–495, Amsterdam, The Netherlands, mai 1998, http ://citeseer.nj.nec.com/197687.html. 2.2.3.1

[Maes et Nardi 1988] P. MAES et D. NARDI (éd.), *Meta-Level Architectures and Reflection*, North-Holland, 1988, http ://www.elsevier.nl. 2.4, A

[Maes 1988] P. MAES, *Issues in Computational Reflection*, p. 21–35, In Maes et Nardi [Maes et Nardi 1988], 1988, http ://www.elsevier.nl. 2.4

[Makpangou et al. 1994] M. MAKPANGOU, Y. GOURHANT, J.-P. LE NARZUL et M. SHAPIRO, « Fragmented Objects for Distributed Abstractions », *Readings in Distributed Computing Systems*, p. 170–186, IEEE Computer Society Press, juillet 1994, ftp ://ftp.inria.fr/INRIA/Projects/SOR/papers/1992/FO_ieeebook92.ps.gz. 3.2.3

[MAL] Mobile Agent Community, *The Mobile Agent List*, http ://mole.informatik.uni-stuttgart.de/mal/mal.html. 3.2.2

[Mao 1999] J. MAO, *Monitoring and Analyzing Method Invocations in the 2K Operating System*, Thèse de Master, Department of Computer Science, University of Illinois at Urbana-Champaign, mai 1999, http ://choices.cs.uiuc.edu/2k/papers/MS-monitoring.ps.gz. 2.4.0.2

[Mathis et al. 1996] M. MATHIS, J. MAHDAVI, S. FLOYD et A. ROMANOW, *TCP Selective Acknowledgement Options*, Request For Comments (RFC) 2018, octobre 1996, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc2018.txt. 2.2.2.4

[Mattern et Sturm 2003] F. MATTERN et P. STURM, « From Distributed Systems to Ubiquitous Computing - The State of the Art, Trends, and Prospects of Future Networked Systems », *Proceedings of KiVS 2003*, p. 3–25, Leipzig, Deutschland, février 2003, http ://www.inf.ethz.ch/vs/publ/papers/DisSysUbiComp.pdf. 3.2.2

[Mayers 1995] C. MAYERS, *Writing Distributed Applications using ANSA and ANSAware 4.1*, Training course, septembre 1995, http ://www.ansa.co.uk/ANSATech/95/Primary/12610002.pdf. 2.3.1.3

[Microsoft] Microsoft Corporation, *Windows Embedded Operating Systems Home Page*, http ://www.microsoft.com/windows/embedded/. 1.1.1

[Microsoft 2000] Microsoft Corporation, *Universal Plug and Play Device Architecture*, juin 2000, statut : « version 1.0 », http ://www.upnp.org/download/UPnPDA10_20000613.htm. 3.2.1

[Microsoft 2001] Microsoft Corporation, *Windows .NET Server Family Beta 3 – Technical Overview*, novembre 2001, http ://www.microsoft.com/windows.netserver/docs/TechOverview.doc. 3.2.2

[Microsoft 2002] Microsoft Corporation, *The .NET Compact Framework – Overview*, avril 2002, http ://msdn.microsoft.com/vstudio/device/compactfx.asp. 3.2.2

[Moats 1997] R. MOATS, *URN Syntax*, Request For Comments (RFC) 2141, mai 1997, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc2141.txt. 3.2.1

[Mockapetris 1987a] P. MOCKAPETRIS, *Domain Names – Concepts and Facilities*, Standard (STD) 0013, Request For Comments (RFC) 1034, novembre 1987, ftp ://ftp.isi.edu/in-notes/std/std13.txt. 3.2.1

[Mockapetris 1987b] P. MOCKAPETRIS, *Domain Names – Implementation and Specification*, Standard (STD) 0013, Request For Comments (RFC) 1035, novembre 1987, ftp ://ftp.isi.edu/in-notes/rfc1035.txt. 3.2.1

[Montenegro 2001] G. MONTENEGRO, *Reverse Tunneling for Mobile IP, revised*, Request For Comments (RFC) 3024, janvier 2001, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc3024.txt. 2.3.1.1

[Mummert et al. 1995] L. B. MUMMERT, M. EBLING et M. SATYANARAYANAN, « Exploiting Weak Connectivity for Mobile File Access », *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP'95)*, p. 143–155, Copper Mountain Resort, Colorado, USA, décembre 1995, http ://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/s15.pdf. 2.2.3.3, 3.1.2.1

[Murphy et al. 2000] A. L. MURPHY, G. P. PICCO et G.-C. ROMAN, *Lime : A Middleware for Physical and Logical Mobility*, rapport technique n°WUCS-00-05, Washington University, St. Louis, Missouri, USA, février 2000, http ://www.elet.polimi.it/Users/DEI/Sections/Compeng/GianPietro.Picco/papers/wucs0005.ps.gz. A

[Murphy et al. 2001] A. L. MURPHY, G. P. PICCO et G.-C. ROMAN, « Lime : A Middleware for Physical and Logical Mobility », *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'2001)*, p. 524–533, Mesa, Arizona, USA, avril 2001, publié aussi en version longue dans [Murphy et al. 2000], http ://www.cs.rochester.edu/u/murphy/papers/icdcs01.pdf. 3.2.2.6

[Myrinet] Myricom, *Home Page*, http ://www.myri.com. 1.2.3

[Namyst et Méhaut 1996] R. NAMYST et J.-F. MÉHAUT, « PM^2 : Parallel Multithreaded Machine. A Computing Environment for Distributed Architectures », *Parallel Computing : State-of-the-Art and Perspectives, Proceedings of the Conference ParCo'95, Septembre 1995, Ghent, Belgium*, *Advances in Parallel Computing*, vol. 11, p. 279–285, Elsevier, North-Holland, Amsterdam, février 1996, http ://www.ens-lyon.fr/∼rnamyst/ps/survey.ps. 3.2.3

[Nandagopal et al. 1999] T. NANDAGOPAL, T. KIM, P. SINHA et V. BHARGHAVAN, « Service Differentiation Through End-to-End Rate Control in Low Bandwidth Wireless Packet Networks », *Proceedings of the 6th IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99)*, San Diego, California, USA, novembre 1999, http ://timely.crhc.uiuc.edu/Papers/momuc99.wtcp.ps.gz. 2.2.2.4

[Napster] Napster, *Home Page*, http ://www.napster.com. 3.2.2

[Nee et al. 1997] P. NEE, K. JEFFAY et G. DANNEELS, « The Performance of Two-Dimensional Media Scaling for Internet Videoconferencing », *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, St. Louis, Missouri, USA, mai 1997, http ://www.cs.unc.edu/∼jeffay/papers/NOSSDAV-97.pdf. 3.2.3

[Nielsen et al. 1998] H. NIELSEN, M. SPREITZER, B. JANSSEN et J. GETTYS, *HTTP-NG Overview Problem Statement, Requirements, and Solution Outline*, Internet-Draft, novembre 1998, statut : « expire le 17 mai 1999 », http ://www.w3.org/Protocols/HTTP-NG/1998/11/draft-frystyk-httpng-overview-00.txt. 2.2.4

[Nielsen et al. 2000] H. NIELSEN, P. LEACH et S. LAWRENCE, *An HTTP Extension Framework*, Request For Comments (RFC) 2774, février 2000, statut : « Experimental », ftp ://ftp.isi.edu/in-notes/rfc2774.txt. 2.2.4

[Noble et al. 1995] B. NOBLE, M. PRICE et M. SATYANARAYANAN, « A Programming Interface for Application-Aware Adaptation in Mobile Computing », *Proceedings of the 2nd USENIX Symposium on Mobile & Location-Independent Computing*, p. 57–66, Ann Arbor, Michigan, USA, avril 1995, http ://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/mobile95.pdf. 2.3.2.2

[Noble et al. 1997] B. NOBLE, M. SATYANARAYANAN, J. E. TILTON, J. FLINN et K. R. WALKER, « Agile Application-Aware Adaptation for Mobility », *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP'16)*, Saint-Malo, France, octobre 1997, http ://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/s16.pdf. 2.3.2.2, 2.3.2.2

[Noble et Satyanarayanan 1994] B. NOBLE et M. SATYANARAYANAN, « An Empirical Study of a Highly Available File System », *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, p. 138–149, Vanderbilt University, Nashville, Tennessee, USA, mai 1994, http ://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/sigm94-coda.pdf. 3.1.2.1

[Noble et Satyanarayanan 1999] B. NOBLE et M. SATYANARAYANAN, « Experience with Adaptive Mobile Applications in Odyssey », *Mobile Networks and Applications (MONET)*, vol. 4, n°4, 1999, p. 245–254, http ://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/monet98-kluwer.pdf. 2.3.2.2

[Noble 2000] B. NOBLE, « System Support for Mobile, Adaptive Applications », *IEEE Personal Computing Systems*, vol. 7, n°1, février 2000, p. 44–49, http ://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/ieeepcs00.pdf. 2.3.2.2

[Obermeyer et Hawkins 2001] P. OBERMEYER et J. HAWKINS, *Microsoft .NET Remoting : A Technical Overview*, juillet 2001, http ://www.microsoft.com/serviceproviders/whitepapers/xml.asp. 3.2.2

[Okamura et Ishikawa 1994] H. OKAMURA et Y. ISHIKAWA, « Object Location Control Using Meta-level Programming », *Proceedings of the 8th European Conference on Object-Oriented Programming (ECOOP'94)*, *Lecture Notes in Computer Science*, vol. 821, p. 299–319, Springer Verlag, Bologna, Italy, juillet 1994, http ://www.csl.sony.co.jp/person/okamura/papers/okamura-ecoop94.pdf. 2.4

[Olivia et al. 1998] A. OLIVIA, I. C. GARCIA et L. E. BUZATO, *The Reflective Architecture of Guaraná*, rapport technique n°IC-98-14, Universidade Estadual de Campinas, Brazil, avril 1998, http ://www.dcc.unicamp.br/∼oliva/guarana/docs/design.ps.gz. 2.4

[Olivia et Buzato 1998] A. OLIVIA et L. E. BUZATO, « Composition of Meta-Objects in Guaraná », *Proceedings of the 13th ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'98)*, p. 82–86, Vancouver, Canada, octobre 1998, http ://www.dcc.unicamp.br/∼oliva/guarana/docs/composition.ps.gz. 2.4

[Olivia et Buzato 1999] A. OLIVIA et L. E. BUZATO, « The Design and Implementation of Guaraná », *Proceedings of the 5th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'99)*, p. 203–216, San Diego, California, USA, mai 1999, http ://www.dcc.unicamp.br/∼oliva/guarana/docs/desimpl.ps.gz. 2.4

[OMG 2000a] Object Management Group, *CORBA Trading Object Service Specification*, mai 2000, statut : « Version 1.0 », http ://cgi.omg.org/docs/formal/00-06-27.pdf. 3.2.1

[OMG 2000b] Object Management Group, *Mobile Agent Facility Specification*, janvier 2000, statut : « Version 1.0 », http ://www.omg.org/docs/formal/00-01-02.pdf. 3.2.3.2

[OMG 2002a] Object Management Group, *Common Object Request Broker Architecture (CORBA/IIOP)*, décembre 2002, statut : « Version 3.0.2 », http ://www.omg.org/docs/formal/02-12-02.pdf. 3.2.2

[OMG 2002b] Object Management Group, *CORBA Naming Service Specification*, septembre 2002, statut : « Version 1.2 », http ://www.omg.org/docs/formal/02-09-02.pdf. 3.2.1

[OMG 2003a] Object Management Group, *Unified Modeling Language*, mars 2003, statut : « Version 1.5 », http ://www.omg.org/docs/formal/03-03-01.pdf. 5.2.1

[OMG 2003b] Object Management Group, *Wireless Access & Terminal Mobility in CORBA*, mars 2003, statut : « Version 1.0 », http ://www.omg.org/docs/formal/03-03-64.pdf. 3.2.2

[Omicini et Zambonelli 1998] A. OMICINI et F. ZAMBONELLI, « Coordination of Mobile Information Agents in TuCSoN », *Internet Research*, vol. 8, n°5, 1998, p. 400–413, http ://sirio.dsi.unimo.it/Zambonelli/PDF/iiis98.pdf. 3.2.2.4

[Orinoco] Orinoco, *Home Page*, http ://www.orinocowireless.com. 1.2.2

[Padmanabhan et Mogul 1996] V. N. PADMANABHAN et J. C. MOGUL, « Using Predictive Prefetching to Improve World Wide Web Latency », *ACM SIGCOMM Computer Communication Review*, vol. 26, n°3, juillet 1996, http ://www.acm.org/sigcomm/ccr/archive/1996/jul96/ccr-9607-mogul-padmanabhan.pdf. 3.1.2

[Paepcke 1991] A. PAEPCKE, *PCLOS Reference Manual*, rapport technique n°HPL-91-182, Hewlett-Packard Laboratories, novembre 1991, http ://www-diglib.stanford.edu/∼paepcke/shared-documents/pclos-manual.ps. 2.4

[Page et al. 1998] T. W. PAGE, R. G. GUY, J. S. HEIDEMANN, D. RATNER, P. L. REIHER, A. GOEL, G. H. KUENNING et G. J. POPEK, « Perspectives on Optimistically Replicated, Peer-to-Peer Filing », *Software – Practice and Experience*, vol. 28, n°2, février 1998, p. 155–180, http ://fmg-www.cs.ucla.edu/ficus/publications/spe98.ps. 3

[Palm] Palm, *Home Page*, http ://www.palm.com. 1.1.1

[Patterson et al. 1995] R. H. PATTERSON, G. A. GIBSON, E. GINTING, D. STODOLSKY et J. ZELENKA, « Informed Prefetching and Caching », *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP'95)*, p. 79–95, Copper Mountain Resort, Colorado, USA, décembre 1995, http ://reports-archive.adm.cs.cmu.edu/anon/1995/CMU-CS-95-134R.ps. 3.1.2

[PCMCIA] Personal Computer Memory Card International Association PCMCIA, *Home Page*, http ://www.pcmcia.org. 1

[PCMCIA 2001] PERSONAL COMPUTER MEMORY CARD INTERNATIONAL ASSOCIATION PCMCIA (éd.), *PC Card Standard Release 8.0*, 2001, http ://www.pc-card.com/bookstore.htm#PC. 1.1.1

[Peine et Stolpmann 1997] H. PEINE et T. STOLPMANN, « The Architecture of the Ara Platform for Mobile Agents », *Proceedings of Mobile Agents, 1st International Workshop (MA'97)*, *Lecture Notes in Computer Science*, vol. 1219, p. 50–61, Springer Verlag, Berlin, Germany, avril 1997, http ://wwwagss.informatik.uni-kl.de/Projekte/Ara/Doc/architecture.ps.gz. 3.2.2

[Pennarun] A. PENNARUN, *The Linux APM Daemon – Home Page*, Net Integration Technologies (NITI), http ://www.worldvisions.ca/∼apenwarr/apmd/. 4.3.2

[Perkins et al. 1994] C. PERKINS, A. MYLES et D. B. JOHNSON, « IMHP : A Mobile Host Protocol for the Internet », *Computer Networks and ISDN Systems*, vol. 27, n°3, décembre 1994, p. 479–491, http ://www.cs.cmu.edu/afs/cs.cmu.edu/user/dbj/www/ftp/mobile/comnet94.ps. 2.2.1

[Perkins et al. 2003] C. PERKINS, D. B. JOHNSON et J. ARKKO, *Mobility Support in IPv6*, Internet-Draft, mai 2003, statut : « expire le 24 novembre 2003 », http ://www.ietf.org/internet-drafts/draft-ietf-mobileip-ipv6-22.txt. 2.2.1

[Perkins et Johnson 2001] C. PERKINS et D. B. JOHNSON, *Route Optimization in Mobile IP*, Internet-Draft, septembre 2001, statut : « expire le 6 mars 2002 », http ://www.ietf.org/internet-drafts/draft-ietf-mobileip-optim-11.txt. 2.3.1.1

[Petersen et al. 1996] K. PETERSEN, M. J. SPREITZER, D. B. TERRY et M. M. THEIMER, « Bayou : Replicated Database Services for World-wide Applications », *Proceedings of the 7th ACM SIGOPS European Workshop (EuroSIGOPS'96)*, p. 275–280, Connemara, Ireland, septembre 1996, http ://www2.parc.com/csl/projects/bayou/pubs/eurosigops-96/ScalableBayou.ps. 3.1.3.1

[Petersen et al. 1997] K. PETERSEN, M. J. SPREITZER, D. B. TERRY, M. M. THEIMER et A. J. DEMERS, « Flexible Update Propagation for Weakly Consistent Replication », *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP'16)*, p. 288–301, Saint-Malo, France, octobre 1997, http ://www2.parc.com/csl/projects/bayou/pubs/sosp-97/AE.ps.gz. 3.1.3.1

[Picco et al. 1999] G. P. PICCO, A. L. MURPHY et G.-C. ROMAN, « Lime : Linda Meets Mobility », *Proceedings of the 21st International Conference on Software Engineering (ICSE'99)*, p. 368–377, Los Angeles, California, USA, mai 1999, http ://www.cs.rochester.edu/u/murphy/papers/icse99.pdf. 3.2.2.6

[Picco et al. 2000] G. P. PICCO, A. L. MURPHY et G.-C. ROMAN, « Developing Mobile Computing Applications with Lime », *Proceedings of the 22nd International Conference on on Software Engineering (ICSE'2000)*, p. 766–769, Limerick, Ireland, juin 2000, http ://www.cs.rochester.edu/u/murphy/papers/icse00demo.pdf. 3.2.2.6

[Postel 1981] J. POSTEL, *Transmission Control Protocol*, Standard (STD) 0007, Request For Comments (RFC) 793, septembre 1981, statut : « Standard », ftp ://ftp.isi.edu/in-notes/std/std7.txt. 2.2, 2.2.2

[Pree 1997] W. PREE, « Essential Framework Design Patterns », *Object Magazine*, vol. 7, n°1, mars 1997, p. 34–37, http ://www.exciton.cs.rice.edu/comp410/frameworks/Pree/J008.pdf. 4.2

[Proxim] Proxim, *Home Page*, http ://www.proxim.com. 1.2.2

[Psion] Psion, *Home Page*, http ://www.psion.com. 1.1.1

[Quéma et al. 2002] V. QUÉMA, L. BELLISSARD et P. LAUMAY, « Application-Driven Customization of Message-Oriented Middleware for Consumer Devices », *Proceedings of the Workshop on Software Infrastructures for Component-Based Applications on Consumer Devices, in association with the 6th International Enterprise Distributed Object Computing Conference (EDOC'2002)*, Lausanne, Switzerland, septembre 2002, http ://sardes.inrialpes.fr/papers/files/02-Quema-WSICBACD.ps.gz. 5.4

[Ramakrishnan et al. 2001] K. RAMAKRISHNAN, S. FLOYD et D. BLACK, *The Addition of Explicit Congestion Notification (ECN) to IP*, Request For Comments (RFC) 3168, septembre 2001, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc3168.txt. 2.2.2.3

[Ramjee et al. 1999] R. RAMJEE, T. F. LA PORTA, S. THUEL, K. VARADHAN et S. Y. WANG, « HAWAII : A Domain-based Approach for Supporting Mobility in Wide-Area Wireless Networks », *Proceedings of the 7th Annual International Conference on Network Protocols (ICNP'99)*, p. 283–292, Toronto, Canada, novembre 1999, http ://www.bell-labs.com/user/ramjee/papers/icnp99.ps.gz. 2.2.1.1

[Ratner et al. 1996] D. RATNER, G. POPEK et P. REIHER, « The Ward Model : A Scalable Replication Architecture for Mobility », *Proceedings of the OOPSLA'96 Workshop on Object Replication and Mobile Computing (ORMC'96)*, San Jose, California, USA, octobre 1996, http ://fmg-www.cs.ucla.edu/ficus-members/ratner/papers/ormc96.ps.gz. 3.1.3.2

[Ratner et al. 1997] D. RATNER, G. POPEK et P. REIHER, *Dynamic Version Vector Maintenance*, rapport technique n°CSD-970022, Department of Computer Science, University of California, California, USA, juin 1997, http ://citeseer.nj.nec.com/ratner97dynamic.html. 3.1.3.2

[Ratner et al. 1999] D. RATNER, P. L. REIHER, G. J. POPEK et R. G. GUY, « Peer Replication with Selective Control », *Proceedings of the Mobile Data Access, 1st International Conference (MDA'99)*, *Lecture Notes in Computer Science*, vol. 1748, p. 169–181, Springer Verlag, Hong Kong, China, décembre 1999, http ://fmg-www.cs.ucla.edu/ratner/papers/mda99.ps.gz. 3.1.3.2

[Ratner et al. 2001] D. RATNER, P. L. REIHER, G. J. POPEK et G. H. KUENNING, « Replication Requirements in Mobile Environments », *Mobile Networks and Applications (MONET)*, vol. 6, n°6, novembre 2001, p. 525–533, http ://fmg-www.cs.ucla.edu/ratner/papers/dialm.ps.gz. 3.1.3.2

[Ratner 1995] D. RATNER, *Selective Replicaton : Fine-Grain Control of Replicated Files*, Thèse de Master, University of California, Los Angeles, California, USA, janvier 1995, ftp ://ftp.cs.ucla.edu/pub/ficus/ucla_csd_950007.ps.gz. 3.1.3.2

[Raverdy et al. 1998] P.-G. RAVERDY, H. VAN GONG et R. LEA, « DART : A Reflective Middleware for Adaptive Applications », *Proceedings of the Workshop on Reflective Programming in C++ and Java at the 13th ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'98)*, p. 37–40, Vancouver, Canada, octobre 1998, http ://www.csg.is.titech.ac.jp/∼chiba/oopsla98/proc/raverdy.pdf. 2.4

[Raverdy et Lea 1998] P.-G. RAVERDY et R. LEA, « DART : A Distributed Adaptive Run-Time », *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*, The Lake District, England, septembre 1998, http ://www.comp.lancs.ac.uk/computing/middleware98/conference/wips/raverdy98.ps.gz. 2.4

[Redmond et Cahill 2000] B. REDMOND et V. CAHILL, « Iguana/J : Towards a Dynamic and Efficient Reflective Architecture for Java », *Proceedings of the Workshop on Reflection and Meta-Level Architectures at the 14th European Conference on Object-Oriented Programming (ECOOP'2000)*, Cannes, France, juin 2000, http ://www.dsg.cs.tcd.ie/∼redmondb/iguanaj/Iguana-J-ECOOP2k.ps. 2.4, 4.3

[Reiher et al. 1994] P. L. REIHER, J. S. HEIDEMANN, D. RATNER, G. SKINNER et G. J. POPEK, « Resolving File Conflicts in the Ficus File System », *Proceedings of the USENIX Summer 1994 Technical Conference*, p. 183–195, Boston, Massachusetts, USA, juin 1994, ftp ://ftp.cs.ucla.edu/pub/ficus/usenix_summer_94_resolver.ps.gz. 3

[Reiher et al. 2000] P. REIHER, R. GUY, M. YARVIS et A. RUDENKO, « Automated Planning for Open Architectures », *Proceedings of the 3rd IEEE Conference on Open Architectures and Network Programming (OpenArch'2000)*, Tel-Aviv, Israel, mars 2000, http ://lasr.cs.ucla.edu/reiher/papers/planning.pdf. 3.2.3.1

[Rogue 2002] Rogue Wave Software, *Ruple : A Loosely Coupled Architecture Ideal for the Internet – White Paper*, janvier 2002, http ://www.roguewave.com/developer/tac/ruple/Ruple.pdf. 3.2.2

[Roose et al. 2002] P. ROOSE, M. DALMAU et F. LUTHON, « A Distributed Architecture for Cooperative and Adaptative Multimedia Applications », *Proceedings of the 26th International Computer Software and Applications Conference (COMPSAC'2002)*, p. 444–449, Oxford, England, août 2002, http ://ahuzki.iutbayonne.univ-pau.fr/∼roose/ pub/articles/compsac2002.pdf. 5.4

[Rosenberg et al. 1997] J. ROSENBERG, H. SCHULZRINNE et B. SUTER, *Wide Area Network Service Location*, Internet-Draft, novembre 1997, statut : « expire en mai 1997 », http ://www.globecom.net/ietf/draft/draft-ietf-svrloc-wasrv-01.html. 3.2.1.2

[Rowstron et Wood 1996] A. I. T. ROWSTRON et A. WOOD, « An Efficient Distributed Tuple Space Implementation for Networks of Workstations », *Proceedings of the 2nd International Euro-Par Conference (Euro-Par'96)*, *Lecture Notes in Computer Science*, vol. 1123, p. 510–513, Springer Verlag, Lyon, France, août 1996, http ://www.cs.york.ac.uk/linda/ps/YCS270.ps.gz. 3.2.2.4

[Samaraweera 1999] N.K.G. SAMARAWEERA, « Non-congestion packet loss detection for TCP error recovery using wireless links », *IEE Proceedings - Communications*, vol. 146, n°4, août 1999, p. 222–230. 2.2.2.3

[Sandhu et al. 1996] R. SANDHU, E. COYNE, H. FEINSTEIN et C. YOUMAN, « Role-Based Access Control Models », *IEEE Computer*, vol. 29, n°2, février 1996, p. 38–47, http ://ite.gmu.edu/list/journals/computer/pdf_ver/i94rbac.pdf. 2.4.0.2

[Satyanarayanan et al. 1990] M. SATYANARAYANAN, J. J. KISTLER, P. KUMAR, M. E. OKASAKI, E. H. SIEGEL et D. C. STEERE, « Coda : A Highly Available File System for a Distributed Workstation Environment », *IEEE Transactions on Computers*, vol. 39, n°4, avril 1990, p. 447–459, http ://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/tcc90.pdf. 1

[Satyanarayanan et al. 1994] M. SATYANARAYANAN, B. NOBLE, P. KUMAR et M. PRICE, « Application-Aware Adaptation for Mobile Computing », *Proceedings of the 6th ACM SIGOPS European Workshop : Matching Operating Systems to Application Needs*, p. 1–4, Dagstuhl Castle, Germany, septembre 1994, http ://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/dagstuhl94.pdf. 2.3.2.2

[Satyanarayanan 1990] M. SATYANARAYANAN, « Scalable, Secure, and Highly Available Distributed File Access », *IEEE Computer*, vol. 23, n°5, mai 1990, p. 9–21, http ://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/scalable90.pdf. 2.2.3.3

[Satyanarayanan 1996] M. SATYANARAYANAN, « Fundamental Challenges in Mobile Computing », *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, p. 1–7, Philadelphia, Pennsylvania, USA, mai 1996, http ://www.cs.cmu.edu/afs/cs/project/coda-www/ResearchWebPages/docdir/podc95.ps.gz. 1, 2

[Saygin et al. 2000] Y. SAYGIN, Ö. ULUSOY et A. K. ELMAGARMID, « Association Rules for Supporting Hoarding in Mobile Computing Environments », *Proceedings of the 10th International*

Workshop on Research Issues on Data Engineering : Middleware for Mobile Business Applications and E-Commerce (RIDE'2000), p. 71–78, San Diego, California, USA, février 2000, http ://www.cs.purdue.edu/homes/saygin/HOME/RIDE.ps. 3.1.2.2

[Schmidt et al. 2002] A. SCHMIDT, M. STROHBACH, K. VAN LAERHOVEN, A. FRIDAY et H.-W. GELLERSEN, « Context Acquisition Based on Load Sensing », *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp'2002)*, *Lecture Notes in Computer Science*, vol. 2498, p. 333–350, Springer Verlag, Göteborg, Sweden, octobre 2002, http ://www.comp.lancs.ac.uk/∼albrecht/pubs/pdf/schmidt_ubicomp_2002.pdf. 6.2.2

[Schueller et al. 2000] J. SCHUELLER, K. BEGAIN, M. ERMEL, T. MUELLER et M. SCHWEIGEL, « Performance Analysis of a Single UMTS Cell », *Proceedings of the European Wireless Communications Conference*, Dresden, Germany, septembre 2000, http ://www.torsten-mueller.net/publications/eww2000_UMTS.pdf. 1.2.2

[SCI] SCIzzL : the Local Area Memory Port Local Area MultiProcessor Scalable Coherent Interface and Serial Express Users, Developers, and Manufacturers Association, *Home Page*, http ://www.SCIzzL.com. 1.2.3

[Segarra et André 1999] M.-T. SEGARRA et F. ANDRÉ, « MFS : A Mobile File System Using Gereric System Services », *Proceedings of the 1999 ACM Symposium on Applied Computing (SAC'99)*, p. 419–420, San Antonio, Texas, USA, février 1999, http ://www.irisa.fr/solidor/doc/ps99/sac99.ps.gz. 2.2.3.2

[Segarra 2000] M.-T. SEGARRA, *Une plate-forme à composants adaptables pour la gestion des environnements sans fil*, Thèse de doctorat, Université de Rennes 1, Rennes, France, novembre 2000, ftp ://ftp.irisa.fr/techreports/theses/2000/segarra.ps.gz. 2.1, 4.3.1, 7.1.1

[Shapiro et al. 1989] M. SHAPIRO, Y. GOURHANT, S. HABERT, L. MOSSERI, M. RUFFIN et C. VALOT, « SOS : An Object-Oriented Operating System — Assessment and Perspectives », *Computing Systems*, vol. 2, n°4, décembre 1989, p. 287–338, ftp ://ftp.inria.fr/INRIA/Projects/SOR/papers/1989/SOS_computing-systems–fall89.ps.gz. 3.2.3

[Shepler et al. 2003] S. SHEPLER, B. CALLAGHAN, D. ROBINSON, R. THURLOW, C. BEAME, M. EISLER et D. NOVECK, *NFS version 4 Protocol*, Request For Comments (RFC) 3530, avril 2003, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc3530.txt. 2.2, 2.2.3

[Silaghi et Keleher 2001] B. D. SILAGHI et P. J. KELEHER, « Object Distribution with Local Information », *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS'2001)*, p. 381–388, Mesa, Arizona, USA, avril 2001, http ://www.cs.umd.edu/∼keleher/papers/icdcs-local01.pdf. 3.2.3

[Silva et al. 1994] J. G. SILVA, J. CARREIRA et L. SILVA, « ParLin : From a Centralized Tuple Space to Adaptive Hashing », *Proceedings of the World Transputer Congress'94*, Lake Como, Italy, septembre 1994, http ://www.uni-paderborn.de/pc2/services/software/parlin/WTC94.ps.Z. 3.2.2.4

[Silva et al. 2001] O. SILVA, A. GARCIA et C. J. LUCENA, « T-Rex : A Reflective Tuple Space Environment for Dependable Mobile Agent Systems », *Proceedings of 3rd Workshop on Wireless Communication and Mobile Computing (WCSF'2001), In conjunction with the 3rd International Conference on Mobile and Wireless Communication Networks (MWCN'2001)*, Recife, Brazil, août 2001, ftp ://ftp.teccomm.les.inf.puc-rio.br/pub/docs/TRex.zip. 3.2.2.4

[Sinha et al. 1999] P. SINHA, N. VENKITARAMAN, R. SIVAKUMAR et V. BHARGHAVAN, « WTCP : A Reliable Transport Protocol for Wireless Wide-Area Networks », *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, p. 231–241, Seattle, Washington, USA, août 1999, http ://www.acm.org/pubs/articles/proceedings/comm/313451/p231-sinha/p231-sinha.pdf. 2.2.2.4

[Skybridge] Skybridge, *Home Page*, http ://www.skybridgesatellite.com. 1.2.2

[Smith 1982] B. C. SMITH, *Procedural Reflection in Programming Languages*, Thèse de doctorat, Laboratory of Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusettes, USA, 1982, http ://citeseer.nj.nec.com/context/323370/0. 2.4

[Soliman et al. 2002] H. SOLIMAN, C. CASTELLUCCIA, K. EL-MALKI et L. BELLIER, *Hierarchical MIPv6 mobility management (HMIPv6)*, Internet-Draft, octobre 2002, http ://www.ietf.org/internet-drafts/draft-ietf-mobileip-hmipv6-07.txt. 2.2.1.1

[SONET] SONET, *Home Page*, http ://www.sonet.com. 1.2.3

[Sourrouille et Contreras 2002] J. L. SOURROUILLE et J. L. CONTRERAS, « Objets Autonomes Adaptables », *Actes de la Journées Composants (JC2002), ASF (ACM SIGOPS France)*, Grenoble, France, octobre 2002, http ://arcad.essi.fr/2002-10-composants/papiers/13-long-sourrouille.pdf. 6.5

[Spectrix] Spectrix Corporation, *Home Page*, http ://www.spectrixcorp.com. 1.2.2

[Spreitzer et al. 1997] M. J. SPREITZER, M. M. THEIMER, K. PETERSEN, A. J. DEMERS et D. B. TERRY, « Dealing with Server Corruption in Weakly Consistent, Replicated Data Systems », *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'97)*, p. 234–240, Budapest, Hungary, septembre 1997, http ://www2.parc.com/csl/projects/bayou/pubs/mobicom-97/ServerCorruption-Handout.fm.ps. 3.1.3.1

[Spreitzer et al. 1999] M. J. SPREITZER, M. M. THEIMER, K. PETERSEN, A. J. DEMERS et D. B. TERRY, « Dealing with Server Corruption in Weakly Consistent, Replicated Data Systems », *Wireless Networks*, vol. 5, n°5, 1999, http ://ip-sapp008.lwwonline.com/ips/frames/toc.asp ?J=5233&I=19. 3.1.3.1

[Spreitzer et Theimer 1993] M. J. SPREITZER et M. M. THEIMER, « Providing Location Information in a Ubiquitous Computing Environment », *Proceedings of the 14th Symposium on Operating System Principles (SOSP'93)*, p. 270–283, Asheville, North Carolina, USA, décembre 1993, http ://portal.acm.org. 6.2.2

[Srinivasan 1995] R. SRINIVASAN, *RPC : Remote Procedure Call Protocol Specification Version 2*, Request For Comments (RFC) 1831, août 1995, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc1831.txt. 3.2.2

[Stangel et Bharghavan 1998] M. STANGEL et V. BHARGHAVAN, « Improving TCP Performance in Mobile Computing Environments », *Proceedings of the International Conference on Communications (ICC'98)*, p. 584–589, Atlanta, Georgia, USA, juin 1998, http ://timely.crhc.uiuc.edu/Papers/icc98.2.ps.gz. 2.2.2.3

[Stroud et Wu 1994] R. J. STROUD et Z. WU, « Using Meta-Objects to Adapt a Persistent Object System to Meet Application Needs », *Proceedings of the 6th ACM SIGOPS European Workshop : Matching Operating Systems to Application Needs*, p. 35–38, Dagstuhl Castle, Germany, septembre 1994, http ://www.cs.ncl.ac.uk/research/trs/abstracts/513.html. 2.4

[Stroud et Wu 1995] R. J. STROUD et Z. WU, « Using Metaobject Protocols to Implement Atomic Data Types », *Proceedings of the 9th European Conference on Object-Oriented Programming (ECOOP'95)*, *Lecture Notes in Computer Science*, vol. 952, p. 168–189, Springer Verlag, Åarhus, Denmark, août 1995, http ://www.ifs.uni-linz.ac.at/∼ecoop/cd/papers/0952/09520168.pdf. 2.4

[Sullivan*III* et al. 1997] W. T. SULLIVAN*III*, D. WERTHIMER, S. BOWYER, J. COBB, D. GEDYE et D. ANDERSON, « A new major SETI project based on Project Serendip data and 100,000 personal computers », *« Astronomical and Biochemical Origins and the Search for Life in the Universe », Proceedings of the 5th International Conference on Bioastronomy*, *IAU Colloquium*, vol. 161,

Editrice Compositori, Bologna, Italy, 1997, http ://setiathome.ssl.berkeley.edu/woody_paper.html. 3.2.2

[Sun 1999] Sun Microsystems, *Jini Architectural Overview – Technical White Paper*, janvier 1999, http ://wwws.sun.com/software/jini/whitepapers/architecture.pdf. 3.2.1.1

[Sun 2000] Sun Microsystems, *Java^{TM} 2 Platform Micro Edition (J2ME^{TM}) Technology for Creating Mobile Devices – White Paper*, mai 2000, http ://java.sun.com/products/kvm/wp/KVMwp.pdf. 3.2.2.4

[Sun 2001a] Sun Microsystems, *The Application of Jini Technology to Enhance the Delivery of Mobile Services*, décembre 2001, http ://wwws.sun.com/software/jini/whitepapers/PsiNapticMIDs.pdf. 3.2.1.1

[Sun 2001b] Sun Microsystems, *Enterprise JavaBeans^{TM} Specification*, août 2001, statut : « Version 2.0, Final Release », ftp ://ftp.java.sun.com/pub/ejb/947q9tbb/ejb-2_0-fr2-spec.pdf. 3.2.2

[Sun 2001c] Sun Microsystems, *Java 2 Platform Enterprise Edition Specification*, juillet 2001, statut : « Version 1.3 », http ://java.sun.com/j2ee/j2ee-1_3-fr-spec.pdf. 3.2.2

[Sun 2001d] Sun Microsystems, *Java^{TM} 2 Platform, Micro Edition – Datasheet*, février 2001, http ://java.sun.com/j2me/j2me-ds-0201.pdf. 3.2.2, 3.2.2.4

[Sun 2002a] Sun Microsystems, *JavaSpaces^{TM} Service Specification*, avril 2002, statut : « version 1.2.1 », http ://wwws.sun.com/software/jini/specs/js1_2_1.pdf. 3.2.1.1, 3.2.2

[Sun 2002b] Sun Microsystems, *Java^{TM} Remote Method Invocation – Specification*, février 2002, statut : « Revision 1.8, Java 2 SDK Standard Edition v1.4 », ftp ://ftp.java.sun.com/docs/j2se1.4/rmi-spec-1.4.pdf. 3.2.2

[Sun 2003] Sun Microsystems, *Java^{TM} 2 SDK, Standard Edition Documentation*, juillet 2003, statut : « version 1.4.2 », http ://java.sun.com/j2se/1.4.2/docs/index.html. 7.1.1

[Sunshine et Postel 1980] C. SUNSHINE et J. POSTEL, *Addressing Mobile Hosts in ARPA Environment*, Internet Experiment Note (IEN) 135, mars 1980, ftp ://ftp.isi.edu/in-notes/ien/ien135.txt. 2.2.1

[Tait et al. 1995] C. D. TAIT, H. LEI, S. ACHARYA et H. CHANG, « Intelligent File Hoarding for Mobile Computers », *Proceedings of the 1st Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'95)*, p. 119–125, Berkeley, California, USA, novembre 1995, http ://www.cs.cmu.edu/afs/cs.cmu.edu/user/satya/Web/MCSALINK/PAPERS/tait95.pdf. 3.1.2

[Tanter et al. 2001] E. TANTER, N. M. N. BOURAQADI-SAÂDANI et J. NOYÉ, « Reflex – Towards an Open Reflective Extension of Java », *Proceedings of the 3rd International Conference on Meta-level Architectures and Separation of Crosscutting Concerns (Reflection'2001)*, *Lecture Notes in Computer Science*, vol. 2192, p. 25–43, Springer Verlag, Kyoto, Japan, septembre 2001, http ://www.dcc.uchile.cl/∼etanter/research/publi/reflex-Reflection2001.ps.gz. 2.4

[Tatsubori et al. 2000] M. TATSUBORI, S. CHIBA, M.-O. KILLIJIAN et K. ITANO, « OpenJava : A Class-Based Macro System for Java », *Reflection and Software Engineering, Papers from OORaSE 1999, 1st OOPSLA Workshop on Reflection and Software Engineering*, *Lecture Notes in Computer Science*, vol. 1826, p. 117–133, Springer Verlag, Denver, Colorado, USA, 2000, http ://www.csg.is.titech.ac.jp/∼mich/openjava/papers/mich_2000lncs1826.pdf. 2.4, 4.3

[Teledesic] Teledesic, *Home Page*, http ://www.teledesic.com. 1.2.2

[Teraoka et al. 1992] F. TERAOKA, K. C. CLAFFY et M. TOKORO, « Design, Implementation, and Evaluation of Virtual Internet Protocol », *Proceedings of the 12th International Conference on Distributed Computing Systems (ICDCS'92)*, p. 170–177, Yokohama, Japan, juin 1992, http ://citeseer.nj.nec.com/teraoka92design.html. 2.2.1

[Terry et al. 1995] D. B. TERRY, M. M. THEIMER, K. PETERSEN, A. J. DEMERS, M. J. SPREITZER et C. HAUSER, « Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System », *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP'95)*, p. 172–183, Copper Mountain Resort, Colorado, USA, décembre 1995, http ://www2.parc.com/csl/projects/bayou/pubs/sosp-95/BayouConflictsSOSPPreprint.ps.gz. 3.1.3.1

[Terry et al. 1998] D. B. TERRY, K. PETERSEN, M. J. SPREITZER et M. M. THEIMER, « The Case for Non-transparent Replication : Examples from Bayou », *IEEE Data Engineering Bulletin*, vol. 21, n°4, décembre 1998, p. 12–20, http ://www2.parc.com/groups/csl/projects/bayou/pubs/dataeng-98/DataEngineeringDec98.frame.pdf. 3.1.3.1

[Thomson et Narten 1998] S. THOMSON et T. NARTEN, *IPv6 Stateless Address Autoconfiguration*, Request For Comments (RFC) 2462, décembre 1998, statut : « Draft Standard », http ://www.ietf.org/rfc/rfc2462.txt. 2.2.1.1

[Tourrilhes] J. TOURRILHES, *Wireless Tools for Linux – Home Page*, Hewlett Packard, http ://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html. 4.3.2

[Touzet et al. 2001] D. TOUZET, J.-M. MENAUD, F. WEIS, P. COUDERC et M. BANÂTRE, « SIDE Surfer : Enriching Casual Meetings with Spontaneous Information Gathering », *ACM SigArch Computer Architecture Newsletter*, vol. 29, n°5, décembre 2001, p. 76–83, http ://www.emn.fr/x-info/jmenaud/specifique/Papiers/SIGARCHCAN/articleACMSIGARCH.ps. 4.3

[Travler] DARPA/CSTO, *Travler Home Page*, Contract n° : DABT63-94-C-0080, http ://fmg-www.cs.ucla.edu/travler98/welcome.html. 3.1.3.2

[Truyen et al. 2001] E. TRUYEN, B. VANHAUTE, W. JOOSEN, P. VERBAETEN et B. N. JØRGENSEN, « Dynamic and Selective Combination of Extensions in Component-Based Applications », *Proceedings of the 23rd International Conference on Software Engineering (ICSE'2001)*, p. 233–242, Toronto, Canada, mai 2001, http ://www.cs.kuleuven.ac.be/∼distrinet/projects/CORRELATE/PUBLICATIONS/ICSE2001.pdf. 2.4

[UDDI 2000] UDDI Consortium, *UDDI Technical White Paper*, septembre 2000, http ://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.PDF. 3.2.1

[USB 2000] USB 2.0 Technical Working Groups, *Universal Serial Bus Revision 2.0 specification*, avril 2000, http ://www.usb.org/developers/data/usb_20.zip. 1.1.1

[Vadet et Merle 2002] M. VADET et P. MERLE, « Adaptation des connecteurs dans le CCM », *Actes de la Journées Composants (JC2002), ASF (ACM SIGOPS France)*, Grenoble, France, octobre 2002, http ://arcad.essi.fr/2002-10-composants/papiers/11-long-vadet.pdf. 5.4

[Vahdat et al. 2000] A. VAHDAT, A. LEBECK et C. S. ELLIS, « Every Joule is Precious : The Case for Revisiting Operating System Design for Energy Efficiency », *Proceedings of the 9th ACM SIGOPS European Workshop*, septembre 2000, http ://www.cs.duke.edu/∼vahdat/ps/sigops00.pdf. 7.2.3.2

[Vaidya et al. 1999] N. VAIDYA, M. MEHTA, C. PERKINS et G. MONTENEGRO, *Delayed Duplicate Acknowledgements : A TCP-Unaware Approach to Improve Performance of TCP over Wireless*, rapport technique n°TR-99-003, Texas A&M University, USA, 1999, http ://playground.sun.com/∼gab/papers/delayed-dupacks.ps. 2.2.2.2

[Valkó 1999] A. G. VALKÓ, « Cellular IP – A New Approach to Internet Host Mobility », *ACM Computer Communication Review*, vol. 29, n°1, janvier 1999, p. 50–65, http ://comet.ctr.columbia.edu/cellularip/pub/ccr99.pdf. 2.2.1.1

[vSteen et al. 1999] M. VAN STEEN, P. HOMBURG et A. S. TANENBAUM, « Globe : A Wide-Area Distributed System », *IEEE Concurrency*, vol. 7, n°1, janvier 1999, p. 70–78, ftp ://ftp.cs.vu.nl/pub/papers/globe/ieeeconc.99.org.pdf. 3.2.3

[W3C 1999a] World Wide Web Consortium, *HTML 4.01 Specification*, décembre 1999, statut : « W3C Recommandation », http ://www.w3.org/TR/html4/. 2.2.4

[W3C 1999b] World Wide Web Consortium, *Resource Description Framework (RDF) Model and Syntax Specification*, février 1999, statut : « W3C Recommandation, errata REC-rdf-syntax-19990222 », http ://www.w3.org/TR/REC-rdf-syntax/. 3.2.1, 7.1.4.2

[W3C 2000] World Wide Web Consortium, *XHTML 1.0 : The Extensible HyperText Markup Language*, janvier 2000, statut : « W3C Recommandation », http ://www.w3.org/TR/html/. 2.2.4

[W3C 2002] World Wide Web Consortium, *Extensible Markup Language (XML) 1.1*, octobre 2002, statut : « W3C Candidate Recommandation », http ://www.w3.org/TR/xml11/. 2.2.4

[Wada et al. 1993] H. WADA, T. YOZAWA, T. OHNISHI et Y. TANAKA, « Mobile Computing Environment Based on Internet Packet Forwarding », *Proceedings of the Usenix Winter 1993 Technical Conference (USENIX Winter'93)*, p. 503–518, San Diego, California, USA, janvier 1993, http ://citeseer.nj.nec.com/context/287041/0. 2.2.1

[Wade 1999] S. WADE, *An Investigation into the use of the Tuple Space Paradigm in Mobile Computing Environments*, Thèse de doctorat, Computing Department, Lancaster University, Lancaster, UK, septembre 1999, ftp ://ftp.comp.lancs.ac.uk/pub/mpg/MPG-99-27.ps.gz. 3.2.2.5

[Wahl et al. 1997] M. WAHL, T. HOWES et S. KILLE, *Lightweight Directory Access Protocol (v3)*, Request For Comments (RFC) 2251, décembre 1997, statut : « Proposed Standard », ftp ://ftp.isi.edu/in-notes/rfc2251.txt. 3.2.1

[Walpole et al. 1999] J. WALPOLE, L. LIU, D. MAIER, C. PU et C. KRASIC, « Quality of Service Semantics for Multimedia Database Systems », *Proceedings of Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 8th Working Conference on Database Semantics (DS'8)*, p. 393–412, Rotorua, New Zealand, janvier 1999, http ://www.cse.ogi.edu/∼krasic/ds8.pdf. 3.1.1.4

[WAP 2000] WAP Forum, *Wireless Application Protocol - White Paper*, juin 2000, http ://www.wapforum.org/what/WAP_white_pages.pdf. 2.3.1.2

[WAP 2001a] WAP Forum, *Wireless Application Protocol - Architecture Specification*, juillet 2001, http ://www1.wapforum.org/tech/documents/WAP-210-WAPArch-20010712-a.pdf. 2.3.1.2

[WAP 2001b] WAP Forum, *Wireless Application Protocol - WAP 2.0 Technical White Paper*, août 2001, http ://www.wapforum.org/what/WAPWhite_Paper1.pdf. 2.3.1.2

[Weider et al. 1992] C. WEIDER, J. REYNOLDS et S. HEKER, *Technical Overview of Directory Services using the X.500 Protocol*, Request For Comments (RFC) 1309, For Your Information (FYI) 14, mars 1992, ftp ://ftp.isi.edu/in-notes/fyi/fyi14.txt. 3.2.1

[Welch et al. 2001] I. WELCH, R. STROUD et A. B. ROMANOVSKY, « Aspects of Exceptions at the Meta-level », *Proceedings of the 3rd International Conference on Meta-level Architectures and Separation of Crosscutting Concerns (Reflection'2001)*, *Lecture Notes in Computer Science*, vol. 2192, p. 280–281, Springer Verlag, Kyoto, Japan, septembre 2001, http ://www.comp.lancs.ac.uk/computing/users/marash/aopws2001/papers/welch.pdf. 2.4

[Welch et Stroud 1998] I. WELCH et R. STROUD, « Dynamic Adaptation of the Security Properties of Applications and Components », *Proceedings of the Workshop on Distributed Object Security at the 12th European Conference on Object-Oriented Programming (ECOOP'98)*, Brussels, Belgium, juillet 1998, http ://www.cs.ncl.ac.uk/research/dependability/reflection/down-loads/ewdos98.pdf. 2.4

[Welch et Stroud 1999] I. WELCH et R. STROUD, « From Dalang to Kava – the Evolution of a Reflective Java Extension », *Proceedings of the 2nd International Conference on Metalevel Architectures and Reflection (Reflection'99)*, *Lecture Notes in Com-*

puter Science, vol. 1616, p. 2–21, Springer Verlag, Saint-Malo, France, juillet 1999, http ://www.cs.ncl.ac.uk/research/dependability/reflection/downloads/reflection99.pdf. 2.4

[Welch et Stroud 2000] I. WELCH et R. STROUD, « Using Reflection as a Mechanism for Enforcing Security Policies in Mobile Code », *Proceedings of the 6th European Symposium on Research in Computer Security (ESORICS'2000)*, *Lecture Notes in Computer Science*, vol. 1895, p. 309–323, Springer Verlag, Toulouse, France, octobre 2000, http ://www.cs.ncl.ac.uk/research/dependability/reflection/downloads/esorics00.pdf. 2.4

[Welch et Stroud 2001] I. WELCH et R. STROUD, « Kava – Using Bytecode Rewriting to add Behavioural Reflection to Java », *Proceedings of the 6th USENIX Conference on Object-Oriented Technology (COOTS'2001)*, p. 119–130, San Antonio, Texas, USA, février 2001, http ://www.cs.ncl.ac.uk/research/dependability/reflection/downloads/coots01.pdf. 2.4

[Welch 1984] T. A. WELCH, « A Technique for High-Performance Data Compression », *IEEE Computer*, vol. 17, n°6, juin 1984, p. 8–19, http ://citeseer.nj.nec.com/context/4286/0. 3.1.1

[Wu et Schwiderski 1997] Z. WU et S. SCHWIDERSKI, *Reflective Java : Making Java Even More Flexible*, rapport technique n°APM.1936.02, ANSA, février 1997, http ://www.ansa.co.uk/ANSATech/97/Primary/193602.pdf. 2.4

[Yarvis et al. 2000] M. YARVIS, P. REIHER et G. J. POPEK, « A Reliability Model for Distributed Adaptation », *Proceedings of the 3rd IEEE Conference on Open Architectures and Network Programming (OpenArch'2000)*, Tel-Aviv, Israel, mars 2000, http ://ficus-www.cs.ucla.edu/yarvis/Conductor/papers/Reliability_OpenArch.pdf. 3.2.3.1

[Yavatkar et Bhagwat 1994] R. YAVATKAR et N. BHAGWAT, « Improving End-to-End Performance of TCP over Mobile Internetworks », *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, Santa Cruz, California, USA, décembre 1994, http ://citeseer.nj.nec.com/context/25177/0. 2.2.2.1

[Yokote 1992] Y. YOKOTE, « The Apertos Reflective Operating System : The Concept and Its Implementation », *Proceedings of 7th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'92)*, *SIGPLAN Notices*, vol. 27, n°10, p. 414–434, ACM, Vancouver, British Columbia, Canada, octobre 1992, ftp ://ftp.csl.sony.co.jp/CSL/CSL-Papers/92/SCSL-TR-92-014.ps.Z. 2.4

[Yokote 1999] Y. YOKOTE, « Past, Present, and Future of Aperios (abstract) », *Invited Talk at the 2nd International Conference on Meta-Level Architectures and Reflection (Reflection'99)*, *Lecture Notes in Computer Science*, vol. 1616, p. 153, Springer Verlag, Saint-Malo, France, juillet 1999, http ://link.springer.de/link/service/series/0558/bibs/1616/16160153.htm. 2.4

[Yu et Vahdat 2000] H. YU et A. VAHDAT, « Design and Evaluation of a Continuous Consistency Model for Replicated Services », *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI'2000)*, San Diego, California, USA, octobre 2000, http ://www.cs.duke.edu/∼vahdat/ps/tact_osdi.pdf. 7.2.3.2

[Zenel et Duchamp 1997a] B. ZENEL et D. DUCHAMP, « General Purpose Proxies : Solved and Unsolved Problems », *Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS-VI)*, p. 87–92, Cape Cod, Massachusetts, USA, mai 1997, http ://guinness.cs.stevens-tech.edu/∼djd/collected-papers/hot-os6.ps. 3.2.2

[Zenel et Duchamp 1997b] B. ZENEL et D. DUCHAMP, « A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment », *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'97)*, p. 248–259, Budapest, Hungary, septembre 1997, http ://guinness.cs.stevens-tech.edu/∼djd/collected-papers/mobicom97-filter.ps. 3.2.2.2

[Zenel 1999] B. ZENEL, « A general purpose proxy filtering mechanism applied to the mobile environment », *Wireless Networks*, vol. 5, n°5, 1999, p. 391–409, http ://www.cs.uno.edu/∼golden/6990MC/MobilePapers/zenel1.pdf. 3.2.2.2

[Zhao et al. 1998] X. ZHAO, C. CASTELLUCCIA et M. BAKER, « Flexible Network Support for Mobility », *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, p. 145–156, Dallas, Texas, USA, octobre 1998, http ://mosquitonet.stanford.edu/publications/flexible.ps. 2.3.1.1

[Zhao et al. 2001] X. ZHAO, C. CASTELLUCCIA et M. BAKER, « Flexible Network Support for Mobile Hosts », *Mobile Networks and Applications (MONET)*, vol. 6, n°2, mars 2001, p. 137–149, http ://mosquitonet.stanford.edu/publications/monet99.ps. 2.3.1.1

[Zhao et Baker 1997] X. ZHAO et M. BAKER, *Flexible Connectivity Management for Mobile Hosts*, rapport technique n°CSL-TR-97-735, Stanford University, California, USA, septembre 1997, ftp ://db.stanford.edu/pub/cstr/reports/csl/tr/97/735/CSL-TR-97-735.pdf. 2.3.1.1

[Zhou et al. 1992] S. ZHOU, J. WANG, X. ZHENG et P. DELISLE, *Utopia : A Load Sharing System for Large, Heterogeneous Distributed Computer Systems*, rapport technique n°CSRI-257, Computer Systems Research Institute, University of Toronto, Toronto, Canada, avril 1992, ftp ://ftp.cs.toronto.edu/pub/reports/csri/257/257.ps.Z. 3.2.3

[Zhou 1986] S. ZHOU, *A Trace-driven Simulation Study of Dynamic Load Balancing*, rapport technique n°UCB/CSD 87/305, Computer Science Division (EECS), University of California, Berkeley, California, USA, septembre 1986, http ://sunsite.berkeley.edu/TechRepPages/CSD-87-305. A

[Zhou 1988] S. ZHOU, « A Trace-driven Simulation Study of Dynamic Load Balancing », *IEEE Transactions on Software Engineering*, vol. 14, n°9, septembre 1988, p. 1327–1341, publié aussi dans [Zhou 1986]. 3.2.3

[Ziv et Lempel 1977] J. ZIV et A. LEMPEL, « A Universal Algorithm for Sequential Data Compression », *IEEE Transactions on Information Theory*, vol. 23, n°3, mai 1977, p. 337–343, http ://www.stanford.edu/class/ee398a/resources/ziv :77-SDC.pdf. 3.1.1

. . . Carpe Diem . . . □

Résumé

L'informatique mobile est un domaine en plein essor qui profite des percées technologiques dans le domaine des ordinateurs portables et dans le domaine des réseaux de communication sans-fil. Ces environnements mobiles présentent des particularités : (i) dû à des limitations de taille et de poids, un terminal portable offre peu de ressources et celles-ci sont susceptibles de varier, (ii) les réseaux sans fil offrent une bande passante plus faible, sujette à des variations importantes et de fréquentes déconnexions dues aux interférences avec l'environnement, et (iii) l'environnement d'un terminal portable change suite à ses déplacements, avec l'accès, ou la disparition de l'accès, à un certain nombre de stations (mobiles ou non) et de périphériques (imprimantes, scanner, etc).

L'objectif de cette thèse est de proposer une méthode qui généralise l'utilisation des ressources extérieures à un terminal portable au moyen de techniques de distribution prenant en compte ces critères de la mobilité. Nous proposons donc un système adaptatif de distribution des applications en environnements mobiles.

Pour cela, nous avons construit une architecture générique se décomposant en un cadre de conception et une boîte à outils. Le cadre de conception comporte des fonctionnalités couramment utilisées pour la gestion des environnements mobile. La boîte à outils comporte des implantations permettant aux concepteurs d'applications de spécialiser les fonctionnalités avec un comportement défini.

Dans ce cadre, nous avons plus précisément développé deux fonctionnalités. La fonctionnalité d'adaptation et de réaction dynamique définit, en particulier, un modèle d'entité auto-adaptative, dans laquelle les concepteurs peuvent dynamiquement spécialiser (i) les adaptations possibles de l'entité et (ii) la stratégie d'adaptation, correspondant aux changements de comportement à adopter en cas de variations dans les conditions d'exécution.

La fonctionnalité de gestion des ressources et de distribution des applications caractérise les particularités de l'environnement mobile et les besoins des applications au sein d'un modèle d'utilisation de type offres / demandes. Cinq services mettent en œuvre cette fonctionnalité : (i) le service de gestion de l'environnement, (ii) le service de gestion de l'environnement local, (iii) le service de détection et notification, (iv) le service de distribution et (v) le service de contrôle de la propagation des adaptations. Ceux-ci comportent des politiques adaptatives, basées sur le modèle de l'entité, qui peuvent être dynamiquement spécialisées par les concepteurs.

Un prototype, AeDEn, a été développé et plusieurs expériences ont confirmé que la distribution en environnement mobile permet d'économiser les ressources des terminaux portables et d'améliorer les performances des applications.

Mots clés : environnement mobile, terminaux portables, réseaux sans-fil, approche générique, cadre de conception, boîte à outils, système adaptatif dynamique, stratégies d'adaptation, informatique répartie, système de distribution dynamique, politiques adaptatives.

VU :

Le Directeur de Thèse
Françoise ANDRÉ

VU :

Le Responsable de l'École Doctorale
Jean-Pierre CONZE

VU pour autorisation de soutenance
Rennes, le
Le Président de l'Université de Rennes 1
Bertrand FORTIN

VU après soutenance pour autorisation de publication
Le Président de Jury