Déploiement et support à l'exécution de services communicants dans les environnements d'informatique ambiante

Frédéric Guidec

MÉMOIRE EN VUE DE L'OBTENTION DU DIPLÔME
D'HABILITATION À DIRIGER DES RECHERCHES EN INFORMATIQUE

présenté devant

L'UNIVERSITÉ DE BRETAGNE SUD
(SOUS LE SCEAU DE L'UNIVERSITÉ EUROPÉENNE DE BRETAGNE)

par

FRÉDÉRIC GUIDEC

Déploiement et support à l'exécution de services communicants dans les environnements d'informatique ambiante

Présenté le 27/06/2008 devant le jury composé de

M. Denis CAROMEL (rapporteur)
M. Patrice QUINTON (rapporteur)
M. Stéphane UBÉDA (rapporteur)
M. Pierre KUONEN (examinateur)
M. Jean-Louis PAZAT (examinateur)
M. Marcelo M. WANDERLEY (examinateur)
M. Pierre-François MARTEAU (examinateur)

RÉSUMÉ

Le concept d'informatique ambiante propose une vision dans laquelle les êtres humains sont continuellement entourés d'une myriade d'équipements numériques susceptibles d'interagir les uns avec les autres, et bien sûr avec les utilisateurs eux-mêmes. Déployer et supporter des services complexes dans un environnement d'informatique ambiante soulève cependant quelques défis majeurs. Ainsi, la grande hétérogénéité des équipements susceptibles d'héberger ces services rend difficile la conception d'applications logicielles pouvant fonctionner sur une grande variété de plates-formes, et capables de s'adapter le cas échéant aux caractéristiques changeantes de ces plates-formes. D'autre part, assurer la communication et la coordination entre des services s'exécutant sur des équipements distincts n'est pas chose aisée lorsque tout ou partie de ces équipements sont mobiles, et ne peuvent s'appuyer pour interagir sur une infrastructure de communication fixe, stable, et performante.

Ce document structuré en trois parties constitue une synthèse des activités de recherche auxquelles j'ai participé au cours des huit dernières années au sein de l'équipe CASA du laboratoire VALORIA[1]. Ces activités ont pour objectif commun de proposer des méthodes, modèles, et outils logiciels susceptibles d'aider à la conception, au déploiement, et à l'exécution de services applicatifs dans des environnements d'informatique ambiante.

La première partie du document fait état de travaux visant à doter des composants logiciels de la capacité de percevoir leur environnement d'exécution (vu comme un ensemble de ressources tant matérielles que logicielles), et d'adapter leur comportement en fonction des caractéristiques de cet environnement. Dans cette partie sont également présentés les résultats d'une étude axée sur le support de la qualité de service offerte aux composants logiciels s'exécutant dans un environnement aux ressources fluctuantes. L'approche proposée consiste à établir une relation contractuelle entre les composants logiciels et la plate-forme d'exécution sur laquelle on les déploie.

Dans la deuxième partie de ce document nous nous intéressons aux problèmes particuliers posés par la disponibilité fluctuante de la ressource « réseau » sur des terminaux mobiles assemblés en réseau mobile ad hoc (MANET) discontinu. Nous présentons notamment un protocole permettant d'assurer la dissémination d'information dans ce type d'environnement, ce protocole reposant sur les principes combinés de la communication « basée contenus » *(Content-Based Networking)*, de la communication opportuniste *(Opportunistic Networking)*, et de la communication tolérant les délais *(Delay-Tolerant Networking)*.

Dans la troisième partie du document nous menons une réflexion sur la nature et les caractéristiques des services applicatifs susceptibles d'être déployés dans des environnements d'informatique ambiante, en particulier lorsque ces environnements intègrent des équipements mobiles assemblés en réseaux ad hoc. Notre propos est illustré à travers deux exemples concrets, le premier visant à assurer le déploiement en mode pair à pair de composants logiciels sur des terminaux mobiles en l'absence de tout réseau d'infrastructure, le second permettant de déployer dans le même type d'environnement un service de type Usenet fonctionnant lui aussi en mode pair à pair.

[1] Laboratoire d'informatique de l'Université de Bretagne Sud.

Remerciements

Je remercie très sincèrement M. Denis Caromel (professeur à l'université de Nice Sophia Antipolis), M. Stéphane Ubéda (professeur à l'INSA de Lyon), et M. Patrice Quinton (professeur à l'ENS de Cachan, antenne de Bretagne), qui m'ont tous les trois fait l'honneur d'être rapporteurs de ce travail. Je remercie également M. Pierre Kuonen (professeur à l'École d'Ingénieurs de Fribourg), M. Jean-Louis Pazat (professeur à l'INSA de Rennes), M. Marcelo M. Wanderley (associate professor à l'université McGill de Montréal) et M. Pierre-François Marteau (professeur à l'Université de Bretagne Sud) d'avoir bien voulu participer au jury de soutenance en tant qu'examinateurs.

Ma gratitude va également à toutes les personnes qui, au cours des dix huit dernières années (eh oui, tout de même...) m'ont successivement permis de découvrir l'univers de la recherche en informatique, d'y prendre goût, d'y trouver une place, et de mener notamment à bien les travaux rapportés dans ce mémoire.

Dès mon stage de DEA, réalisé en 1990 à l'IRISA dans l'équipe ADP dirigée alors par Michel Raynal, Claude Jard m'a donné l'occasion de découvrir l'univers du calcul distribué sur super-calculateur T-Node.

À l'issue de ce stage de DEA, Henri Nussbaumer et Dominique Decotignie m'ont accueilli pendant deux ans en tant qu'assistant au sein du laboratoire d'informatique technique de l'EPFL[2]. Cette immersion dans un univers bouillonnant d'idées a sans doute fortement orienté les choix que j'ai faits par la suite en termes de formation et de carrière.

Françoise André et Jean-Marc Jézéquel m'ont ensuite fait confiance en me donnant l'occasion de préparer ma thèse de doctorat, sous leur direction conjointe, au sein de l'équipe PAMPA de l'IRISA. Le sujet de thèse portait sur une approche, alors considérée comme relativement exotique mais aujourd'hui communément admise, consistant à programmer des super-calculateurs MIMD à l'aide d'un langage à objets.

En m'offrant la possibilité de faire un post-doctorat de trois ans au sein du laboratoire d'informatique théorique de l'EPFL, Giovanni Coray et Pierre Kuonen m'ont permis de participer à plusieurs projets d'envergure impliquant calcul numérique intensif et optimisation combinatoire. Mon intérêt actuel pour les communications mobiles et pour l'informatique ambiante résulte certainement de l'expérience acquise dans certains des travaux menés à cette époque.

Les travaux de recherche rapportés dans ce document sont bien sûr avant tout des travaux collectifs, auxquels ont systématiquement contribué plusieurs membres de l'équipe CASA du laboratoire VALORIA. Je remercie donc tout particulièrement les doctorants qui ont eu ou ont encore à me supporter en tant qu'encadrant de leur travail de thèse : Nicolas Le Sommer (à présent membre permanent de l'équipe), Hervé Roussain, et Julien Haillot. Merci également à Jacques Malenfant, Patrice Frison, et Pierre-François Marteau pour avoir accepté de diriger ces travaux de thèse dont j'ai assuré l'encadrement. Je remercie par ailleurs tous les autres membres passés et présents de l'équipe CASA — à savoir Yves Mahéo, Luc Courtrai, Didier Hoareau, Romeo Said, et Salma Ben Sassi — pour l'ambiance amicale et néanmoins studieuse qui règne dans cette équipe.

Un remerciement tout particulier enfin à Yves Mahéo et Sylvie Gibet qui ont tous deux bien voulu assumer la tâche ingrate de relecteurs de ce mémoire.

[2]EPFL : École Polytechnique Fédérale de Lausanne.

Table des matières

Chapitre 1

Introduction

Ce document constitue, pour l'essentiel, une synthèse des travaux de recherche que j'ai menés depuis mon intégration au laboratoire VALORIA de l'Université de Bretagne Sud (UBS), en septembre 1998. Mon premier contact avec le monde de la recherche remontant cependant au tout début des années 1990, je commencerai par dresser ci-dessous un historique rapide de mes activités — et aussi de mes allers et venues de laboratoire en laboratoire — au cours de la période précédant ma nomination à l'UBS. Ces activités ne seront toutefois pas détaillées plus avant dans le reste de ce document, dans lequel je me focaliserai ensuite sur mes travaux les plus récents.

Étudiant à l'université de Rennes I au cours de l'année universitaire 1989-1990, c'est dans l'équipe ADP (Algorithmique Distribuée et Parallélisme) de l'IRISA que j'ai effectué mon stage de DEA, sous la direction de Claude Jard. Ce stage avait pour thème l'étude des possibilités de portage de l'environnement Echidna (environnement d'expérimentation de protocoles basé sur le langage Estelle) sur une machine à base de transputers. En guise d'expérimentation, j'ai porté à l'époque cet environnement sur le T-Node, machine parallèle dotée de 32 transputers T800 et qui venait tout juste d'arriver à l'IRISA. Les résultats des mesures de performances réalisées sur le T-Node afin de valider ce portage ont à l'époque fait l'objet d'un rapport de recherche INRIA[1], et d'une publication dans la Lettre du Transputer et des Calculateurs Distribués [54].

À l'issue de ce stage de DEA et avant de pouvoir commencer une thèse de doctorat, j'ai effectué mon service national au titre de la coopération au Laboratoire d'Informatique Technique (LIT) de l'École Polytechnique Fédérale de Lausanne (EPFL). Pendant cette période couvrant les années universitaires 1990-1991 et 1991-1992 j'ai travaillé dans le domaine des réseaux de terrain (réseaux locaux spécialisés destinés au monde industriel), et contribué notamment au développement du système Phoebus IIx développé par le laboratoire. J'ai également pu participer à cette époque au groupe de travail FICIM TC-T307 *(Fieldbus Integration into Computer Integrated Manufacturing — User Requirements)*, axe de recherche industrielle du projet européen ESPRIT 5206. Ces divers travaux ont donné lieu aux publications [55, 56, 60, 64, 79, 80].

De retour en France en octobre 1992, j'ai intégré l'équipe PAMPA (Programmation des Architectures Massivement PArallèles) de l'IRISA, dans laquelle j'ai commencé la préparation de ma thèse de doctorat sous la direction de Françoise André et l'encadrement de Jean-Marc Jézéquel. Mon tra-

[1] Les divers rapports de recherche et délivrables de projets auxquels j'ai contribué au cours de ma carrière ne sont pas mentionnés dans la bibliographie fournie en fin de document. Seuls sont listés les articles ayant fait l'objet d'une invitation ou d'une sélection par un comité de lecture, avec ensuite publication dans les actes d'une conférence ou dans un journal.

vail de thèse s'est inscrit dans le cadre du développement et de l'expérimentation d'EPEE (Environnement Parallèle d'Exécution de Eiffel), un *framework* de développement bâti à l'aide du langage à objets Eiffel, et destiné à faciliter le développement de code pour machines massivement parallèles à mémoire distribuée (*i.e.* super-calculateurs de type MIMD). J'ai ainsi enrichi l'environnement EPEE, dont le développement initial avait été réalisé par J.-M. Jézéquel, en le dotant de mécanismes génériques facilitant la distribution des données et la parallélisation des calculs. Ces outils ont ensuite été mis à profit pour développer une bibliothèque de démonstration, la bibliothèque PALADIN, dédiée au calcul d'algèbre linéaire sur machine parallèle. Ce travail s'est soldé par les publications [57, 58, 65, 66, 67, 68, 69, 70, 71, 75, 76, 77, 78, 98], et bien sûr par mon mémoire de thèse [59].

Après ma soutenance de thèse en 1995, j'ai rejoint le Laboratoire d'Informatique Théorique de l'École Polytechnique Fédérale de Lausanne (EPFL) afin d'y effectuer un séjour de post-doctorat de trois ans. Au cours de ce séjour, j'ai participé à plusieurs projets de recherche (dont deux projets européens) menés par le Groupe de Recherche en Informatique Parallèle de l'EPFL. C'est toutefois sur le projet européen STORMS (*Software Tools for the Optimization of Resources in Mobile Systems*) que s'est concentré l'essentiel de mon activité pendant toute cette période. Ce projet financé conjointement par la Communauté Européenne et par le Fond National Suisse s'inscrivait dans le cadre du 4^e programme cadre ACTS (*Advanced Communications, Technologies and Services*). Il impliquait la collaboration de douze partenaires européens académiques et industriels, et visait à la mise en œuvre d'outils logiciels (et notamment des programmes de calcul parallèles) capables d'aider à la conception et la planification des réseaux de télécommunications mobiles. Les travaux réalisés au cours de ce projet se sont soldés par les publications [17, 18, 19, 20, 21, 22, 23, 61, 62, 63, 72, 73, 74, 102, 140, 141, 142].

Au terme du projet STORMS, j'ai été nommé à la rentrée universitaire 1998 en tant que maître de conférences à l'Université de Bretagne Sud (UBS). J'ai alors rejoint le VALORIA, laboratoire d'informatique de l'UBS. Le VALORIA concentre ses activités autour du thème général de l'informatique diffuse — ou « ubiquitaire » — et de l'intelligence ambiante, privilégiant trois axes thématiques : les systèmes logiciels interactifs multimédia et intelligents, l'architecture des systèmes logiciels, et les intergiciels pour les systèmes distribués mobiles et communicants. Ce laboratoire a connu plusieurs phases de restructuration au cours des dix dernières années. Pour simplifier je présenterai les travaux rapportés dans ce document comme ayant tous été réalisés dans le cadre de l'équipe CASA, bien qu'en réalité plusieurs équipes ont été créées, puis dissoutes, avant que l'équipe CASA soit elle-même constituée.

L'équipe CASA est à ce jour constituée de quatre maîtres de conférences (Yves Mahéo, Luc Courtrai, Nicolas Le Sommer, et moi-même) et de trois doctorants (Julien Haillot, Romeo Said, Salma Ben Sassi). Deux thèses de doctorat ont déjà été soutenues dans cette équipe, la première par Nicolas Le Sommer en décembre 2003, la seconde par Didier Hoareau en décembre 2007 (Didier est à présent ATER à l'Université de La Réunion).

De manière générale les travaux de notre équipe couvrent le troisième des axes thématiques du laboratoire VALORIA, à savoir la conception d'intergiciels pour les systèmes distribués mobiles et communicants. Dans cette optique, notre objectif est de contribuer à promouvoir le développement de services applicatifs susceptibles d'être déployés et utilisés dans des environnements d'informatique ambiante, qui à ce jour constituent encore des environnements difficiles à maîtriser.

Les concepts « d'informatique ambiante » et « d'ubiquité numérique » nous proposent en effet une vision dans laquelle les êtres humains sont continuellement entourés d'une myriade d'équipements numériques susceptibles d'interagir les uns avec les autres, et bien sûr avec les utilisateurs eux-mêmes.

Depuis quelques années les applications informatiques ont effectivement commencé à s'échapper des unités centrales des « stations de travail » et autres « serveurs » dans lesquelles elles étaient confinées jusqu'alors, et s'infiltrent à présent dans chacun des objets qui peuplent notre environnement quotidien (*e.g.* assistants numériques personnels, *smartphones*, systèmes de navigation GPS, etc.).

Déployer et supporter des services élaborés dans un environnement d'informatique ambiante n'est pas chose aisée, et soulève même quelques défis majeurs. Ainsi, la grande hétérogénéité des équipements constituant un tel environnement rend difficile la conception d'applications logicielles pouvant fonctionner sur une grande variété de plates-formes. L'approche consistant à produire autant de versions d'une application qu'il existe de plates-formes cibles potentielles atteint ses limites, dans la mesure où l'ensemble de ces plates-formes ne cesse de croître. Cette observation justifie une approche qui consiste à essayer de concevoir des applications à base de composants logiciels capables de s'adapter en fonction du contexte dans lequel ils doivent s'exécuter. En outre, des plates-formes d'exécution capables d'héberger de tels composants sont nécessaires, ces plates-formes devant notamment fournir aux composants les informations qui vont leur permettre de prendre des décisions d'adaptation.

Un autre sujet épineux dans le domaine de l'informatique ambiante concerne la communication entre équipements mobiles. En effet, assurer la communication et la coordination entre des équipements n'est pas chose aisée, lorsqu'on ne peut s'appuyer pour ce faire sur les liens de transmission d'une infrastructure de communication fixe, stable, et performante. Dans de telles conditions le déploiement et l'exécution de services distribués sur des équipements mobiles nécessite que les paradigmes de communication et de coordination traditionnels soient revisités — et le cas échéant remplacés par d'autres — afin de tenir compte des contraintes spécifiques à des environnements dans lesquels tout ou partie des équipements peuvent être mobiles.

Les activités de recherche du groupe CASA visent à proposer des solutions aux divers problèmes évoqués ci-dessus. Notre approche consiste à développer des méthodes, modèles, et outils logiciels susceptibles d'aider à la conception, au déploiement, et à l'exécution de services applicatifs dans un environnement d'informatique ambiante impliquant des équipements mobiles. Nos activités se sont jusqu'à ce jour concentrées sur les trois points suivants :

- le déploiement et le support à l'exécution de services mis en œuvre à partir de composants logiciels ;
- la perception du contexte d'exécution des services et la gestion contractualisée des ressources utilisées par ces services ;
- le support de la communication dans les réseaux mobiles ad hoc discontinus.

Plan de ce document

Ce document est structuré en trois chapitres, correspondant aux trois grands axes thématiques sur lesquels il m'a été donné de travailler ces dernières années.

Le chapitre 2 fait état des activités menées dans le domaine général de la perception du contexte d'exécution *(context-awareness)*. Ces travaux ont pour l'essentiel été réalisés entre 2000 et 2003 dans le cadre du projet RASC *(Resource-Aware Software Components)*, dont la finalité était de doter des composants logiciels de la capacité de percevoir leur environnement d'exécution — vu comme un ensemble de ressources tant matérielles que logicielles — afin qu'ils puissent adapter leur comportement en fonction des fluctuations observées dans cet environnement (paragraphe 2.2). Ce projet a notamment servi de fil conducteur au travail de thèse de Nicolas Le Sommer, qui a approfondi ce

sujet en développant une approche originale de gestion contractualisée de l'accès aux ressources pour les composants logiciels (paragraphe 2.4). En 2002-2003, le projet ACI GRID Concerto nous a donné l'occasion d'étendre ces travaux afin d'en appliquer les résultats au domaine du *Grid Computing*, en concevant des méthodes et outils permettant à des composants logiciels parallèles déployés sur une grappe de machines de percevoir l'état de l'ensemble des ressources distribuées dans cette grappe (paragraphe 2.3).

Le chapitre 3 présente un bilan de mes activités passées et en cours dans le domaine du support de la communication dans les réseaux mobiles ad hoc (MANETs) discontinus. Ces réseaux sans fils, qui reposent sur des interactions directes entre équipements mobiles en l'absence de tout support d'infrastructure, présentent un potentiel extrêmement intéressant pour suppléer ou compléter les réseaux de communication traditionnels. Cependant l'utilisation du mode de communication ad hoc pose un certain nombre de problèmes difficiles, auxquels les protocoles de communication standards n'apportent pas de solutions. Les paragraphes 3.1 et 3.2 précisent ce qu'est exactement un réseau mobile ad hoc discontinu et mettent en évidence les problèmes spécifiques posés par ce type de réseau. Le paragraphe 3.3 présente de manière synthétique nos propres travaux dans ce domaine. Ces travaux ont commencé dès 2003 dans le cadre du projet MASC *(Mobile Adaptive Software Components)* financé par la Région Bretagne, et qui a fait l'objet du travail de thèse d'Hervé Roussain. Dans le cadre de ce projet, notre objectif était de supporter le déploiement et l'exécution de composants logiciels sur des équipements mobiles, en l'absence de toute infrastructure de communication fixe. Ce travail nous a permis de jeter les bases d'un modèle de communication adapté à de telles contraintes. Il a ensuite été repris et intégré à l'occasion du projet SARAH, labellisé par l'ANR en 2005 dans le cadre du programme ARA SSIA (Actions de Recherche Amont en Sécurité, Systèmes embarqués et Intelligence Ambiante). Ce projet, qui se poursuit encore à ce jour, est divisé en quatre thèmes d'activité dont l'un porte spécifiquement sur la problématique de la communication dans les MANETs discontinus. Le travail de thèse de Julien Haillot s'inscrit précisément dans cet axe thématique.

Le chapitre 4 présente des travaux que nous avons menés en vue d'intégrer les résultats des travaux évoqués plus haut, et d'en illustrer l'applicabilité à travers la mise en œuvre de services applicatifs opérationnels. Le paragraphe 4.1 motive notre démarche, en montrant qu'en dépit du grand nombre de travaux menés dans le domaine de la communication dans les réseaux mobiles ad hoc, rares sont les auteurs qui prennent la peine de mener une réflexion sur les applications susceptibles d'être mises en œuvre dans des environnements aussi atypiques. Le paragraphe 4.2 présente ensuite les travaux que nous avons réalisés en vue de supporter le déploiement d'applications conçues par assemblage de composants logiciels sur des terminaux mobiles assemblés en réseau ad hoc. Le paragraphe 4.3 présente quant à lui un service de discussion fonctionnant en mode pair-à-pair et pouvant être déployé et utilisé dans un réseau MANET discontinu.

Le chapitre 5 présente un bilan général des divers travaux que nous avons menés ces dernières années, et énumère quelques perspectives pour les années à venir. Des informations complémentaires concernant les contrats qui nous ont permis de financer notre activité sont fournies en annexe A (p. 107), et les stages de DEA et les travaux de thèse que j'ai encadrés sont listés en annexe B (p. 111).

Chapitre 2

Perception de l'environnement et gestion contractuelle de l'accès aux ressources dans les composants logiciels

2.1 Introduction

Les environnements d'informatique distribuée, et a fortiori les environnements d'informatique ambiante, se caractérisent en général par la grande hétérogénéité des équipements qui les constituent, et la dynamicité des ressources disponibles sur ces équipements. Pour palier ces contraintes, les travaux menés par l'équipe CASA au cours de la période 2002-2005 se sont inscrits dans le domaine général de la « perception du contexte d'exécution » *(context-awareness)*.

Dans [42], Anind K. Dey définit la notion de « contexte » comme englobant « toute information pouvant être utilisée pour caractériser la situation d'une entité ». Dans certains travaux la notion de contexte prend en compte des critères tels que la position géographique, les préférences de l'utilisateur (exprimées en général sous forme de « profil »), certains paramètres de configuration, etc. Dans les travaux de notre équipe, nous nous sommes focalisés sur la perception de l'environnement dans lequel les composants logiciels s'exécutent, cet environnement étant vu comme un ensemble de ressources dont la disponibilité et les caractéristiques peuvent varier au cours du temps.

Les paragraphes qui suivent dressent le panorama des travaux que nous avons menés ces dernières années dans cette optique. Nous nous sommes notamment efforcés de mettre à la disposition des concepteurs de composants Java des méthodes et outils permettant à ces composants de percevoir les ressources disponibles dans leur environnement d'exécution, et d'évaluer par introspection leur propre consommation vis-à-vis de ces ressources. Le paragraphe 2.2 présente notre contribution dans ce domaine, et décrit notamment les outils de type intergiciel que nous avons développés à cette occasion. Le paragraphe 2.3 montre comment ces outils ont ensuite pu être étendus et adaptés au domaine du *Grid Computing*, puis intégrés à une plate-forme destinée à supporter le déploiement de composants parallèles adaptables sur des grappes de machines. Enfin le paragraphe 2.4 montre comment ces mêmes outils ont également été mis à profit lors de la conception d'une plate-forme dédiée à l'hébergement sécurisé de programmes mobiles, cette plate-forme permettant aux programmes candidats à l'hébergement de négocier par contrat les conditions d'accès aux ressources qui leur sont nécessaires au cours de leur exécution.

2.2 L'environnement d'exécution vu comme un ensemble de ressources

2.2.1 Motivation

L'idée fondamentale selon laquelle des applications logicielles complexes devraient systématiquement être conçues par assemblage de composants logiciels disponibles « sur étagères » implique que chacun de ces composants puisse être développé par anticipation, en faisant si possible abstraction des caractéristiques de l'environnement dans lequel il pourra ensuite être installé et exécuté. Pourtant, les composants logiciels ne sont pas tous équivalents vis-à-vis des ressources qui leur sont nécessaires pour fonctionner correctement. Certains composants peuvent tolérer l'absence épisodique ou chronique de certaines ressources, alors que d'autres composants requièrent que l'accès aux ressources qui leurs sont nécessaires leur soit garanti pendant toute la durée de leur exécution. Ainsi, les composants englobant par exemple des codecs audio ou vidéo ne peuvent en général être déployés et utilisés que sur des systèmes offrant une puissance de calcul suffisante, et disposant donc notamment de micro-processeurs suffisamment puissants. D'autres composants dits « communicants » ne peuvent être déployés que sur des systèmes sur lesquels des interfaces de communication sont effectivement disponibles et accessibles. Dans le même ordre d'idée, des composants qui sont censés être persistants ne le seront effectivement que si le système sur lesquels on les déploie offre la possibilité d'assurer cette persistance, ce qui implique en général la disponibilité d'un quelconque support de stockage permanent dans ce système.

Le désir de concevoir et mettre en œuvre des composants logiciels aussi génériques que possible se heurte donc à l'hétérogénéité sans cesse grandissante des plates-formes susceptibles de supporter ensuite l'exécution de ces composants. Une solution possible consiste alors à se donner les moyens d'*adapter* les composants logiciels à leur environnement d'exécution. Une telle adaptation peut être réalisée statiquement, dynamiquement, ou en combinant ces deux méthodes. L'adaptation statique consiste, pour l'essentiel, à concevoir d'emblée des composants adaptés à un certain environnement d'exécution (par exemple en construisant ou en configurant ces composants en vue d'une installation sur l'environnement cible). Dans cette approche, l'adaptation s'effectue en amont de la phase d'exécution proprement dite, et ne peut guère être ensuite remise en question. Un exemple typique d'adaptation réalisée en amont de la phase d'exécution consiste à compiler le code source d'un composant en vue de produire un code exécutable dédié à un type de micro-processeur particulier.

Une approche alternative consiste à concevoir des composants et applications capables de s'exécuter sur une machine virtuelle, et des supports exécutifs permettant de disposer d'une telle machine virtuelle sur une grande variété de plates-formes cibles. L'engouement actuel pour le langage Java résulte ainsi en grande partie du fait que ce langage a été conçu d'emblée pour pouvoir s'exécuter sur une machine virtuelle, dont les spécifications ont d'ailleurs été définies en même temps que le langage lui-même. Cette caractéristique confère au code Java une très grande portabilité, puisqu'il suffit de porter la machine virtuelle Java sur un nouveau type de plate-forme pour que l'ensemble du code Java développé jusqu'à ce jour soit aussitôt exécutable sur cette plate-forme[1].

Une conséquence de l'approche consistant à développer des composants logiciels afin qu'ils puissent s'exécuter sur une machine virtuelle aux contours pré-définis est que les caractéristiques saillantes de la plate-forme hébergeant cette même machine virtuelle se trouvent alors masquées vis-

[1] On fait ici volontairement abstraction, pour simplifier le discours, du fait qu'il existe aujourd'hui plusieurs « éditions » de Java *(Java Standard Edition, Java Enterprise Edition, Java Micro-Edition)*, qui se différencient notamment les unes des autres par les possibilités fort distinctes qu'elles offrent aux développeurs.

à-vis des composants lors de leur exécution. Ainsi, un programme Java n'a au cours de son exécution qu'une visibilité extrêmement réduite des ressources disponibles au niveau du système sous-jacent (*i.e.* CPU, mémoire, système de fichiers, interfaces de communication, etc.), puisque la machine virtuelle lui masque la plupart de ces ressources. En conséquence, un programme Java peut difficilement *découvrir* au cours de son exécution quelles sont les ressources disponibles sur la plate-forme sur laquelle il s'exécute (*e.g.* existe-t-il au moins une interface de communication opérationnelle ? La plate-forme offre-t-elle un système de fichiers accessible ?). En outre un programme Java n'est pas en mesure d'estimer de quelles quantités de ressources il peut disposer au cours de son exécution (*e.g.* quel est le débit permis par l'interface de communication ? Quelle est la quantité de mémoire disponible ?), et n'est donc pas en mesure d'adapter son comportement en conséquence.

Les travaux que l'équipe CASA a menés ces dernières années dans le domaine du *context awareness* ont eu pour finalité de supporter l'adaptation *dynamique* entre composants et environnement d'accueil. Certains composants applicatifs peuvent en effet être conçus de manière à être adaptables vis-à-vis de leur environnement d'exécution, c'est-à-dire capables d'adapter leur comportement — et donc leurs besoins — en fonction de l'environnement dans lequel ils doivent s'exécuter. Cette approche présente l'avantage de permettre de prendre en compte le fait que les propriétés de l'environnement dans lequel un composant s'exécute peuvent évoluer *pendant* cette exécution. Ainsi, la connectivité d'une plate-forme d'exécution vis-à-vis d'un réseau quelconque peut fluctuer alors même que des composants communicants sont en cours d'exécution sur cette plate-forme. La puissance de calcul disponible sur une plate-forme, tout comme la quantité de mémoire disponible, peuvent de même varier selon la charge globale observée sur cette plate-forme. Inversement, les besoins d'un composant peuvent également varier au cours de son exécution. Il est donc souhaitable que ces besoins, tout comme les ressources disponibles sur la plate-forme qui l'accueille, puissent être réévalués en continu — et donc, dynamiquement —- tout au long de son exécution.

Dans le cadre du projet RASC *(Resource-Aware Software Components)* mené entre 2000 et 2003[2], nous nous sommes attachés à enrichir l'exécutif Java 2 standard afin d'offrir aux concepteurs de composants Java les moyens de modéliser sous forme d'objets les diverses ressources susceptibles d'être utilisées par ces composants, de découvrir les ressources disponibles sur la plate-forme supportant leur exécution, et enfin d'en superviser l'utilisation au cours de cette exécution.

Des environnements tels que JRes [9, 37], GVM [8], et KaffeOS [7] apportent des éléments de réponses à ces divers problèmes en fournissant des mécanismes permettant de comptabiliser l'utilisation de chaque type de ressource par une entité active (un *thread* dans le cas de JRes, un processus dans le cas de GVM et de KaffeOS). Cependant dans ces divers environnements les ressources dont la consommation peut être comptabilisée correspondent à un ensemble de types prédéfinis, et concernent exclusivement des ressources globales. On peut ainsi comptabiliser les accès au réseau réalisés par un *thread* (ou processus), mais on ne peut distinguer les transmissions réalisées vers une certaine machine distante, ni vers un numéro de port TCP ou UDP précis. On peut de même comptabiliser les accès au système de fichier dans son ensemble, mais on ne peut comptabiliser de manière différenciée les accès réalisés vers des répertoires ou fichiers spécifiques.

L'intergiciel que nous avons développé est baptisé RAJE *(Resource-Aware Java Environment)*. Contrairement à la plupart des produits évoqués plus haut, cet intergiciel est aisément extensible et peut donc être enrichi au fil du temps afin de prendre en compte de nouveaux types de ressources, ou être porté sur de nouvelles plates-formes matérielles. En outre il permet de superviser l'usage qui est fait des ressources à un niveau « global » (*e.g.* CPU, mémoire système, *swap*, etc.), mais aussi l'usage

[2]Des informations détaillées sur les divers projets menés par l'équipe CASA sont disponibles dans l'annexe A p. 107.

qui en est fait au niveau de chaque programme Java (*e.g.* sockets TCP et UDP, temps CPU et quantité de mémoire consommés par chaque *thread* Java, etc.).

2.2.2 Modélisation des ressources sous forme d'objets

Dans l'intergiciel RAJE, toutes les ressources sont modélisées sous forme d'objets Java. De ce point de vue, RAJE est semblable à d'autres intergiciels permettant la perception des ressources, tels que JRes [37] et KaffeOS [7]. RAJE définit des classes Java modélisant certaines ressources relatives à la plate-forme matérielle et au système d'exploitation tout entiers (*e.g.* CPU, mémoire système, swap, etc.), ainsi que des ressources exploitables au niveau applicatif (*e.g. sockets*, fichiers, *threads*, etc.). Certaines de ces classes sont propres à RAJE (*i.e.* il n'existe pas de classes équivalentes dans l'API Java standard), alors que d'autres ont été définies en étendant des classes de l'API standard. Par exemple, les classes standard *Socket* et *File* ont été étendues de telle sorte que tout accès aux ressources qu'elles représentent puisse être supervisé en cours d'exécution. De manière générale, des informations concernant l'état de n'importe quel type de ressource peuvent être obtenues en invoquant les méthodes appropriées sur les objets Java modélisant ces ressources au sein de l'intergiciel RAJE. Bien sûr la nature exacte des informations ainsi collectées dépend du type de ressource considéré. Une liste des principales ressources modélisées avec RAJE est reproduite ci-dessous. Pour chaque type de ressource considéré, on a fourni entre parenthèses quelques exemples d'informations pouvant être collectées :

- CPU du système (type de processeur, vitesse, taille du cache)
- Mémoire et *swap* du système (taille disponible, consommation actuelle)
- Processus et *threads* du système (consommation vis-à-vis du CPU et de la mémoire)
- Interfaces réseau[3] (type de l'interface, adresses, état)
- Alimentation (type d'alimentation, consommation instantanée et autonomie dans le cas d'alimentation sur batterie)
- *Threads* Java (consommation vis-à-vis du CPU et de la mémoire pour chaque *thread* ou groupe de *threads*, niveau de priorité, politique d'ordonnancement appliquée)
- *Sockets* TCP et UDP Java (adresses et numéros de ports locaux et distants, nombre d'octets émis et reçus, nombre de datagrammes UDP émis et reçus)
- Fichiers Java (nombre d'octets écrits dans — et lus depuis — chaque fichier)

RAJE a été conçue de manière à être aisément extensible. L'intergiciel peut donc être complété à tout instant par de nouvelles classes permettant de modéliser de nouveaux types de ressources.

2.2.3 Extension du JRE standard : un compromis nécessaire entre portabilité et contrôle

La majeure partie du code constituant l'intergiciel RAJE est du code Java, et en tant que tel est portable sur une grande variété de plates-formes cibles. Toutefois, la partie restante de ce code est constituée de code C natif permettant l'extraction d'information du système d'exploitation sous-jacent, et l'interaction avec les rouages internes de la JVM (machine virtuelle Java). L'intergiciel RAJE a dans un premier temps été implémenté de manière à s'interfacer avec un système d'exploitation de type Linux, et avec les rouages internes de la JVM Kaffe 1.0.6 distribuée par la société *Transvirtual Technology*. Les ressources système telles que le CPU, la mémoire, les interfaces réseau, etc. sont supervisées en

[3] La supervision des interfaces sans-fils de type Wi-Fi est possible, ce qui permet notamment à des programmes Java d'être réactifs vis-à-vis de la connectivité — souvent très fluctuante — dans ce type de réseau.

consultant le pseudo-système de fichiers */proc* du système Linux. D'autre part la JVM Kaffe 1.0.6 a été modifiée de telle sorte que les *threads* Java soient implémentés sous la forme de *threads* natifs dans le système d'exploitation. Cette approche permet un meilleur contrôle de la ressource CPU de la part des programmes Java. Elle permet par exemple d'évaluer précisément la quantité de CPU consommée par chaque *thread* Java au cours de son exécution, dans la mesure où cette information est directement accessible sous */proc* pour chaque *thread* Linux.

Les *threads* Linux étant en fait des *threads* POSIX, l'API des *threads* Java a été étendue afin que les *threads* Java puissent exploiter les trois types de politiques d'ordonnancement définies dans le standard POSIX (*i.e.* ordonnancement FIFO, *Round-Robin*, et une troisième politique baptisée `SCHED_OTHER`). Dans un système Linux, les politiques FIFO et *Round-Robin* sont réservées pour des processus et *threads* s'exécutant avec des privilèges de type *superuser* (ou *root*). Les processus et *threads* « normaux » sont censés se contenter de la politique `SCHED_OTHER`, qui dans le cas de Linux met en œuvre une politique de temps-partagé avec mécanisme de priorités dynamique. Les trois politiques d'ordonnancement sont donc disponibles dans l'intergiciel RAJE, mais ne peuvent être utilisées toutes les trois que si la JVM est lancée avec des privilèges de type *superuser*. Dans le cas contraire, les *threads* Java seront tous soumis à la politique d'ordonnancement par défaut `SCHED_OTHER`, et l'utilisateur aura toutefois la possibilité d'influer sur le niveau de priorité relative attribué à chaque *thread* dans un programme Java.

Dans la même optique, la consommation de mémoire de la part d'un programme Java peut être évaluée *thread* par *thread*. Ainsi, à chaque création d'un nouvel objet au cours de l'exécution de ce programme, la quantité de mémoire requise pour instancier cet objet est mise sur le compte du *thread* Java actif lors de cette instanciation. De même, lorsqu'un objet Java est récupéré par le ramasse-miettes de la JVM, la quantité de mémoire ainsi libérée est déduite du compte du *thread* ayant créé cet objet initialement.

Enfin, l'API standard des *threads* Java a été étendue afin que les quantités de mémoire et de temps CPU (exprimé en temps utilisateur et en temps système) consommées par chaque *thread* puissent être observées en invoquant simplement une méthode sur l'objet *Thread* correspondant. Les quantités de mémoire et de temps CPU consommées par un groupe de *threads* peuvent être observées de façon similaire.

Ces divers aménagements réalisés au niveau de certaines classes standard de l'API Java 2 et au niveau de la mise en œuvre de la JVM Kaffe 1.0.6 compromettent sérieusement la portabilité de l'intergiciel RAJE. C'est pourquoi une version expurgée de cet intergiciel, offrant moins de possibilités mais étant aussi nettement plus portable, a également été réalisée dans le cadre du projet RASC. Cette version « allégée » de l'intergiciel RAJE est évoquée plus en détails dans le paragraphe 2.2.7.

2.2.4 Découverte et suivi des ressources disponibles

Avec l'intergiciel RAJE toutes les ressources sont modélisées sous la forme d'objets Java. Puisque de tels objets peuvent être créés et détruits — ou, plus précisément, dé-référencés — dynamiquement par un programme en cours d'exécution, RAJE met en œuvre un registre de ressources, qui permet d'identifier et d'assurer le suivi des ressources au cours de l'exécution d'un programme. Lors de son instanciation, ce registre est automatiquement « peuplé » d'objets modélisant les ressources disponibles au niveau du système d'exploitation sous-jacent. Des ressources de niveau applicatif exploitées par le programme lui-même (*i.e. threads* Java, *sockets* TCP ou UDP, fichiers, etc.) viendront ensuite s'y ajouter au fur et à mesure qu'elles seront instanciées par un programme en cours d'exécution.

En consultant le registre des ressources, un programme peut identifier les objets modélisant des ressources dans son propre espace de nommage, et invoquer ensuite directement des méthodes sur ces objets pour en consulter l'état. Dans l'architecture Java 2 standard, plusieurs espaces d'exécution (communément appelés « domaines de protection » dans la littérature) peuvent être créés au sein d'une même machine virtuelle. Pour ce faire il suffit d'utiliser un objet *ClassLoader* distinct pour charger chaque programme d'application dans la machine virtuelle. Avec l'intergiciel RAJE, un registre de ressources distinct peut être associé si nécessaire à chaque domaine de protection, si bien que les objets ressources manipulés par différents programmes tournant dans la même JVM seront enregistrés et gérés séparément par ces multiples registres de ressources. Cette approche permet notamment de distinguer entre les ressources utilisées par différents programmes applicatifs tournant au sein d'une même JVM. Elle présente en outre l'avantage d'assurer une meilleur « étanchéité » entre ces programmes, et contribue ainsi à éviter l'accaparement ou la corruption de ressources lorsque des programmes concurrents sont hébergés par la même JVM. Cet aspect relatif à la sûreté d'exécution des programmes est abordé plus avant dans le paragraphe 2.4, qui rapporte les travaux que nous avons effectués dans le domaine de l'hébergement sécurisé de programmes non dignes de confiance.

2.2.5 Supervision de l'utilisation des ressources

RAJE fournit deux modèles distincts pour superviser l'utilisation des ressources au niveau d'un programme Java. Cette supervision peut être effectuée par consultation, ou par notification.

La supervision de ressources par notification repose sur un mécanisme de levée d'événements associés aux tentatives d'accès aux ressources. Avec ce modèle, toute tentative d'accès à une ressource de la part d'un programme Java peut être interceptée et traitée immédiatement par un processus de supervision. La supervision par notification est obtenue en implémentant un mécanisme de remontée d'événements *(event call-back)* vers des objets « auditeurs » *(listeners)* depuis les classes modélisant les ressources considérées. Tout objet ressource supportant ce type de supervision peut se voir associer un ou plusieurs auditeurs. À chaque fois qu'une méthode d'accès est invoquée sur l'objet ressource en question, celui-ci informe chacun de ses auditeurs qu'une tentative d'accès est en cours vers la ressource qu'il modélise. Cette approche permet de concevoir des moniteurs dédiés à la supervision des ressources, ces moniteurs étant capables de s'enregistrer en tant qu'auditeurs vis-à-vis d'un ou plusieurs objets ressources, et d'être tenus dès lors informés de l'usage qui est fait des ressources correspondantes.

La figure 2.1 fournit une vue partielle sur la hiérarchie des ressources pouvant être soumises à une supervision par notification, et des types d'auditeurs capables de superviser chacun un type de ressource particulier.

Certains types de ressources ne se prêtent pas bien à la supervision par notification. C'est par exemple le cas de ressources telles qu'un CPU, un *thread*, ou encore la mémoire système. Pour de telles ressources, l'intergiciel RAJE met en œuvre un mode de supervision par consultation *(polling)*, dans lequel un processus chargé de superviser l'utilisation qui est faite d'une certaine ressource doit en consulter l'état périodiquement, afin d'évaluer dans quelle mesure cette ressource a pu être utilisée depuis l'observation précédente.

Pour être observable par consultation, un objet modélisant une ressource doit être capable de produire sur demande un rapport d'observation. Par exemple, un objet *Thread* peut produire sur demande un objet de type *ThreadReport*, dont les attributs caractérisent l'état du *thread* correspondant. La figure 2.2 présente une partie de la hiérarchie des ressources observables par consultation dans la

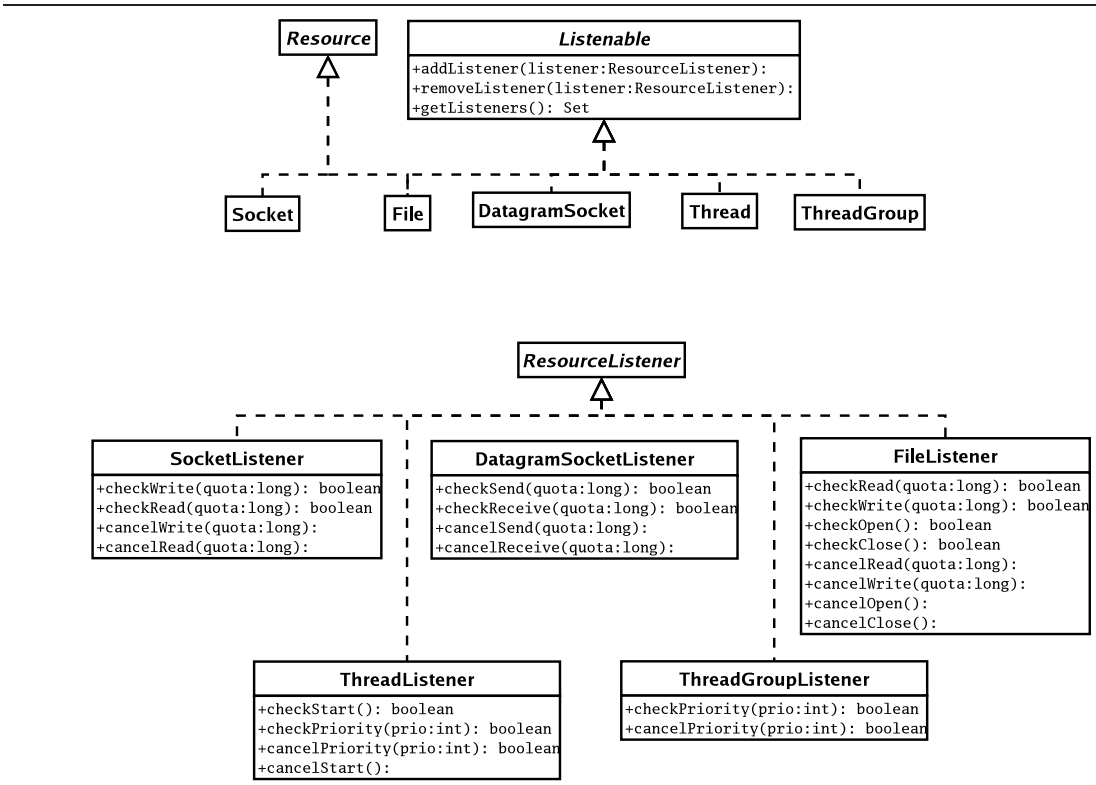


FIG. 2.1 – Modélisation des ressources pouvant faire l'objet d'une supervision par notification dans RAJE, et des types d'auditeurs associés à ces ressources

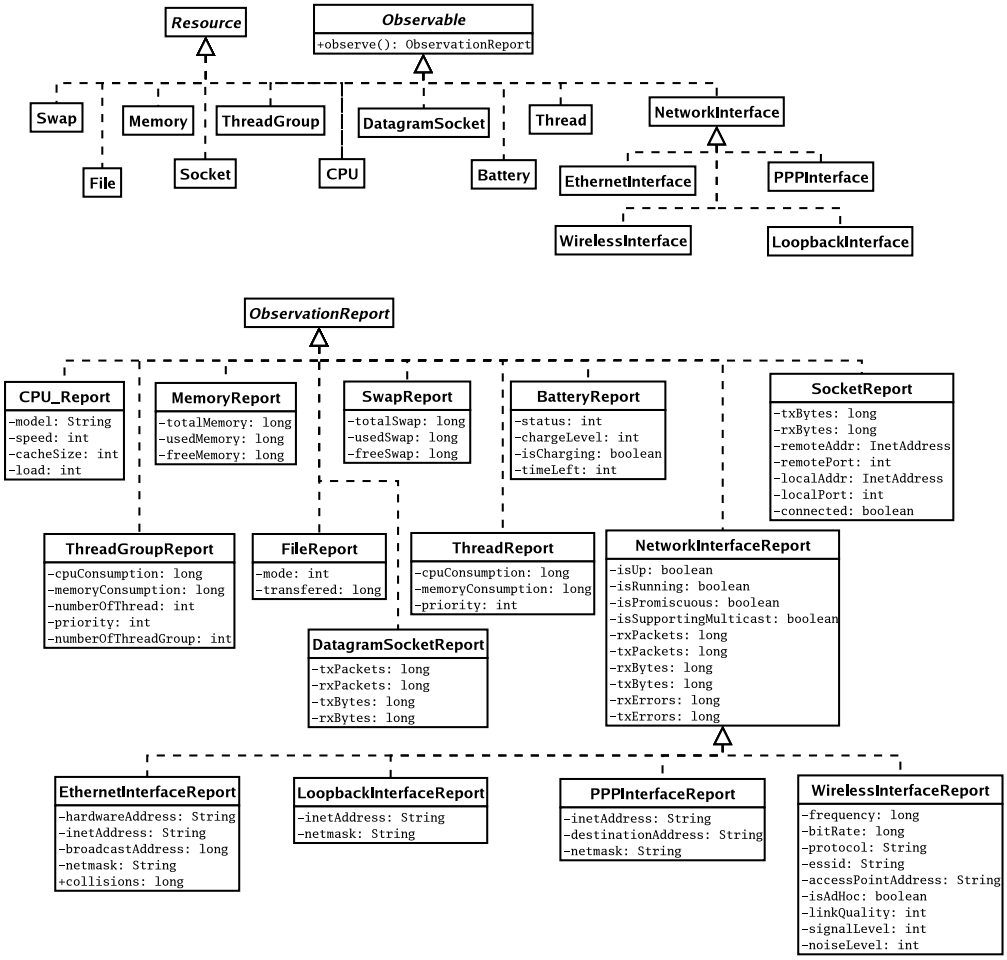


FIG. 2.2 – Modélisation des ressources pouvant faire l'objet de supervision par consultation dans RAJE, et des types de rapports d'observation qui leur sont associés

plate-forme RAJE, ainsi que des types des rapports d'observation associés.

Les deux modes de supervision supportés par l'intergiciel RAJE sont en fait complémentaires. Des moniteurs dédiés à la supervision des ressources peuvent être construits en s'appuyant sur l'un ou l'autre mode de supervision, voire en combinant ces deux modes si nécessaire (si le type de ressource considéré le permet).

2.2.6 Support pour le contrôle d'accès aux ressources : verrouillage d'objets ressources

RAJE fournit des mécanismes permettant de contrôler l'utilisation qui est faite de certaines ressources. Ainsi, lorsqu'un certain type de ressources le permet la classe correspondante implémente l'interface *Lockable*, grâce à laquelle un objet ressource peut être « verrouillé » et « déverrouillé » explicitement, ce qui permet d'en limiter l'utilisation. Lorsqu'un objet ressource est verrouillé, toute tentative d'accès à cette ressource va se solder par la levée d'une exception. Cette approche permet d'implémenter et de faire respecter des politiques d'ordonnancement ou de sécurisation de haut niveau, en donnant à des moniteurs chargés de la supervision des ressources la possibilité d'influer sur l'accessibilité de ces ressources depuis des programmes d'application Java. Cette fonctionnalité a notamment été utilisée dans la plate-forme JAMUS présentée dans le paragraphe 2.4.

2.2.7 Principaux résultats

Nos travaux portant sur la supervision de l'accès aux ressources de la part de composants logiciels ont pour l'essentiel été menés entre 2000 et 2003 dans le cadre du projet RASC, qui servait alors de fil conducteur au travail de thèse de Nicolas Le Sommer[4]. Ce travail s'est notamment soldé par le développement de l'intergiciel extensible RAJE, permettant la supervision des ressources de niveau système et de niveau applicatif depuis des programmes Java**.** Une version expurgée de cet intergiciel, baptisée SAJE *(System-Aware Java Environment)*, a également été produite et est depuis lors mise à disposition de la communauté scientifique sous licence LGPL[5]. L'intergiciel SAJE se distingue du précédent par le fait qu'il permet seulement d'exercer une supervision non intrusive sur les ressources de niveau système. La supervision des ressources de niveau applicatif (*e.g.* fichiers, *threads*, etc.) n'est pas possible, pas plus que le verrouillage de ressources. La mise en œuvre de l'intergiciel SAJE n'a donc nécessité — contrairement à celle de l'intergiciel RAJE — ni modification de l'API Java 2 standard, ni le recours à une JVM modifiée pour l'occasion. SAJE est donc totalement compatible avec le JRE standard, et peut en conséquence être distribué librement. En outre, le code natif permettant l'observation de l'état des ressources système dans l'intergiciel SAJE a été porté sur le SDK Windows, si bien que SAJE peut être utilisé indifféremment sur une plate-forme Linux ou sur une plate-forme Windows.

Le travail réalisé pour concevoir et mettre en œuvre les intergiciels RAJE et SAJE a fait l'objet des deux publications mentionnées ci-dessous. Par ailleurs il est aussi largement évoqué dans les diverses publications référencées dans les paragraphes 2.3.5 et 2.4.7.

– Frédéric Guidec and Nicolas Le Sommer. *Towards Resource Consumption Accounting and Control in Java : a Practical Experience.* In Workshop on Resource Management for Safe Language, ECOOP 2002, Málaga, Spain, June 2002. [81]

[4]Les divers travaux de thèse évoqués dans ce document sont listés dans l'annexe B p. 111.

[5] http://www-valoria.univ-ubs.fr/CASA/SAJE

– Nicolas Le Sommer and Frédéric Guidec. *Intégration des threads temps-réel POSIX sous Java.* In 13e Rencontres Francophones du Parallélisme (RenPar'13, ASTI'01), pages 91-95, Paris, France, April 2001. [111]

2.3 Gestion de ressources pour composants parallèles adaptables

2.3.1 Motivation

Dans le domaine du calcul numérique intensif, les super-calculateurs dédiés ont depuis quelques années tendance à céder la place à des architectures moins coûteuses reposant sur des grappes (ou *clusters*) de machines, ces machines étant elles-mêmes assemblées à partir de matériel courant disponible « sur étagère ». Un laboratoire ou une entreprise peut ainsi s'équiper à faible coût d'une plate-forme de calcul parallèle, en constituant une « grappe » de stations de travail interconnectées via un système de réseau local performant (de type Gibabit Ethernet par exemple). Des grappes de ce type peuvent ensuite être assemblées à plus grande échelle via Internet, et s'insérer ainsi dans un environnement de *Grid Computing*. Le déploiement d'applications parallèles sur des infrastructures de calcul couvrant une ou plusieurs grappes demeure à ce jour une entreprise périlleuse, même si des projets tels que Globus[6] ont fortement contribué à faciliter l'administration et l'exploitation de grilles de calculateurs. Parmi les problèmes qui font encore obstacle à l'utilisation systématique des grappes de stations de travail en tant que ressources de calcul pour le *Grid Computing*, le manque d'infrastructure logicielle constitue l'un des verrous majeurs qu'il convient de lever afin d'autoriser une véritable généralisation du calcul parallèle sur grilles.

Diverses approches sont envisageables pour concevoir et déployer une application capable d'exploiter une ou plusieurs grappes de stations de travail. Parmi celles-ci, l'approche par composants [163] est séduisante, dans la mesure où elle permet d'envisager le développement d'applications complexes par simple assemblage de composants pré-existants, chacun de ces composant étant conçu comme un « code parallèle » destiné à être déployé sur une grappe.

Peu de modèles supportent directement la notion de composant parallèle, c'est-à-dire de composant mettant en jeu des activités parallèles. Parmi les travaux prenant en considération cet aspect, on peut citer le *Common Component Architecture* (CCA) [4] qui définit un modèle de composants dédiés aux applications scientifiques parallèles. L'objectif principal de CCA est de permettre l'interopérabilité de codes scientifiques pré-existants. Dans cette architecture, l'accent est mis sur la définition d'un langage de définition d'interface scientifique (SIDL) indépendant de tout langage de programmation, et les spécifications d'un modèle de communication entre composants à base de ports. Cependant, CCA ne fournit aucun mécanisme pour l'adaptation des composants. Le projet Padico [41] (mené dans le cadre de l'ACI GRID-RMI) étant dédié au couplage de code scientifique, il présente des similarités avec CCA. Il étend le modèle de composant Corba (CCM) dans lequel un composant est associé à un unique espace d'adressage, la communication entre composants consistant alors à transférer les données d'un espace d'adressage à un autre via un mécanisme de communication performant. Un composant parallèle est défini comme étant une collection de composants CCM séquentiels qui exécutent en parallèle tout ou une partie de ses services (modèle d'exécution SPMD). Les travaux décrits dans [12] ont, tout comme ceux du projet Padico, été effectués dans le cadre de l'ACI GRID-RMI. Ils permettent de définir des composants parallèles aussi bien selon le modèle SPMD que MIMD : en

[6] http://www.globus.org

s'appuyant sur la bibliothèque Pro-Active [24], une implémentation du modèle de composants Fractal [15] a été réalisée. Cette implémentation permet la définition de composants hiérarchiques pour les grilles de calculs. Des mécanismes pour la communication asynchrone, la migration d'activités, le déploiement et la mise au point sont offerts. L'introspection sur les composants et le placement des activités est possible mais de façon hétérogène : l'introspection sur la structure des composants est implicite au modèle Fractal, Pro-Active fournit quant à lui des mécanismes permettant de gérer le placement des objets actifs.

Ainsi, les différents travaux présentés ci-dessus s'efforcent tous, par des approches différentes, de définir des modèles de composants pour les grilles de calculs. En règle générale cependant, ils ne permettent pas de récupérer de manière précise des informations relatives à l'environnement d'un composant (dont le système d'exploitation lui-même) et de ses sous-composants. Partant de l'hypothèse que de telles informations sont nécessaires à l'adaptation, nous nous sommes efforcés de fournir un cadre homogène en associant des fonctionnalités d'observation de ressources à la définition d'un modèle de composants approprié pour le développement d'applications distribuées.

Les travaux rapportés dans les paragraphes qui suivent ont été menés au cours de la période 2002-2003 dans le cadre du projet CONCERTO (cf. annexe A p. 107). Dans ce projet, notre objectif était de proposer un modèle de composant parallèle pouvant être déployé sur une grappe de machines, tout en développant les méthodes et outils permettant à ce type de composant de s'adapter aux spécificités matérielles et logicielles de la grappe cible. Ma participation au projet CONCERTO a principalement porté sur les aspects relatifs à la modélisation et la perception des ressources distribuées au niveau d'une grappe de machines. Ces aspects sont présentés en détails dans le paragraphe 2.3.3. En revanche dans le paragraphe qui suit je me contente de fournir une description assez succincte de la notion de composant parallèle telle qu'elle a été définie et mise en œuvre dans la plate-forme que nous avons développée.

2.3.2 Notion de « composant parallèle »

Les modèles de composants issus de l'industrie (*e.g.* COM de Microsoft [133], *Enterprise JavaBeans* de Sun [40], *Corba Component Model* [139] de l'OMG) ne sont pas conçus pour supporter des composants parallèles, c'est-à-dire des composants mettant en jeu des activités parallèles. Quelques travaux préliminaires ont cependant permis de proposer des ébauches de modèles ou de plates-formes destinés à supporter ce type de composants, mais ces travaux visent pour la plupart la réutilisation de codes de calcul intensif fondés sur le parallélisme de données. On peut notamment citer l'initiative du *Common Component Architecture Forum* dont l'objectif est de définir une API standard permettant la définition de « ports » garantissant l'interopérabilité de composants [4]. Par ailleurs, une approche étendant le CCM pour prendre en compte des collections de composants séquentiels identiques a été proposée dans [149]. Dans ces travaux, l'accent n'est pas mis sur l'adaptabilité des composants, mais sur la performance des communications devant être effectuées en parallèle lorsque deux composants parallèles interopèrent.

La plate-forme CONCERTO que nous avons conçue dans le cadre du projet du même nom est dédiée à l'accueil de composants parallèles *auto-adaptables*. Notre objectif n'était pas de proposer au cours de ce projet un nouveau modèle de composant — et de composant parallèle qui plus est —, mais plutôt de fournir une infrastructure *favorisant l'adaptation* des composants dans un environnement de type grille. Dans cette optique nous nous sommes donc contentés de proposer une définition minimale (développée ci-après) de ce que peut être un composant parallèle. En l'occurrence, le programmeur dé-

sirant développer un composant pour la plate-forme CONCERTO doit concevoir ce composant comme un ensemble de *threads* Java coopérants. Il doit en outre définir lui-même la partie métier du composant (nom, interface, mise en œuvre). La plate-forme lui offre cependant des mécanismes utiles pour la gestion des aspects non fonctionnels relatifs à la découverte des ressources disponibles sur la grappe hébergeant le composant, et à la supervision des ressources mobilisées par ce composant au cours de son exécution.

Interfaces du composant. Un composant parallèle accueilli par la plate-forme CONCERTO doit offrir trois interfaces :

- Interface « métier » : Aucune hypothèse n'est formulée sur le type d'interface métier du composant. Le programmeur de composant peut par exemple proposer une interface métier construite sur les services RMI Java. Le composant est alors un objet implantant l'interface *Remote*, dont les méthodes pourront être invoquées à distance par les clients. Le composant peut aussi être un serveur à l'écoute d'un port de la machine sur lequel les clients doivent ouvrir une socket. En outre, il peut proposer une interface métier distribuée (*i.e.* associés à plusieurs objets implantant chacun une partie de l'interface) afin de pouvoir être interconnecté en parallèle avec un autre composant parallèle.
- Interface « cycle de vie » : À travers cette interface, on peut contrôler les différentes étapes de la vie du composant. Ceci concerne essentiellement le déploiement du composant sur la grappe, et son arrêt, mais des services de sauvegarde et de restauration de l'état du composant pourraient également être offerts via cette interface.
- Interface « ressource » : Le composant comporte une interface ressource à travers laquelle on accède aux informations relatives aux ressources qu'il utilise. En outre, le composant est lui-même considéré comme une ressource dans la plate-forme CONCERTO et, à ce titre il définit une méthode *observe()* retournant un rapport d'observation le concernant. Ce rapport d'observation est, par défaut, constitué par agrégation de rapports concernant toutes les ressources élémentaires utilisées par le composant. Le programmeur d'un composant parallèle peut cependant, s'il l'estime nécessaire (pour des raisons de sécurité par exemple), modifier la portée des informations divulguées aux clients de son composant en définissant un type de rapport d'observation approprié.

Structure interne d'un composant. Pour construire un composant parallèle, le programmeur développe un ensemble de *threads* Java, ces *threads* coopérant entre eux pour réaliser les méthodes de l'interface métier du composant. Les *threads* sont regroupés en entités de placement (ou de distribution), appelées « fragments ». Un fragment est un sous-ensemble des *threads* d'un même composant, destinés à être placés au sein d'une même JVM sur un même nœud de la grappe. Ces *threads* pourront ainsi se partager un espace d'objets. La communication et la synchronisation des *threads* d'un même fragment s'effectuent donc comme dans n'importe quel programme Java multithreadé. En revanche, les *threads* appartenant à des fragments distincts doivent s'appuyer sur des mécanismes de communication et de synchronisation tels que *sockets*, RMI, etc.

Déploiement d'un composant. Pour déployer un composant sur une grappe, on doit fournir un fichier de description de déploiement de son composant. Ce fichier décrit :

- la structure du composant en termes de fragments et de *threads* à déployer ;

```
< ?xml version="1.0" encoding="ISO-8859-1" ?>
<component>
 <description>Genetic Algorithm Component</description>
 <name>
  <concerto-name>GenAlgoRes</concerto-name>
  <rmi-name>GeneticAlgorithmServer</rmi-name>
 </name>
 <requirements>
  <java kaffe="1.0.7" d-raje="1.0"/> <concerto-vers="1.0"/> <rmiregistry/>
 </requirements>
 <file>GenAlgorithm.jar</file>
 <fragment-list>
  <fragment name="GA-Master">
   <node>ANY</node> <thread-list> <runnable-class="ThMaster"/></thread-list>
  </fragment>
  <fragment name="GA-Slave">
   <node>ALL</node> <thread-list> <runnable-class="ThSlave"/> </thread-list>
  </fragment>
 </fragment-list>
</component>
```

FIG. 2.3 – Exemple de descripteur de déploiement de composant parallèle exploitable par la plate-forme CONCERTO

- des directives de placement des fragments. On pourra ainsi par exemple dupliquer certains fragments sur tous les nœuds, ou placer un fragment donné sur un nœud spécifique.
- les contraintes imposées par le composant pour que son déploiement soit possible (*e.g.*, présence d'une version précise de la JVM, d'un registre RMI...).

Nous avons développé un langage dédié (DSL : *Domain-Specific Language*), dérivé d'XML, permettant d'exprimer de telles directives. La plate-forme CONCERTO est capable d'interpréter ce langage afin d'assurer le déploiement et le lancement des composants sur une grappe. Un exemple de descripteur de déploiement exploitable par la plate-forme CONCERTO est reproduit dans la figure 2.3 (une analyse détaillée de ce descripteur est fournie dans [129]), et la figure 2.4 illustre le placement de composants parallèle sur une petite grappe constituée de trois machines.

2.3.3 Modélisation et contrôle des ressources distribuées

Principes généraux. L'objectif principal du projet CONCERTO était de fournir aux composants logiciels les moyens de percevoir leur environnement d'exécution, afin qu'ils puissent adapter leur mode de fonctionnement à l'état de cet environnement, voire aux variations observées dans cet environnement au cours de leur exécution. Dans cette optique nous avons pris pour base le cadre conceptuel offert par l'intergiciel RAJE (décrit au paragraphe 2.2), et l'avons étendu afin de répondre aux problèmes spécifiques posés par la perception des ressources distribuées dans une plate-forme de type grille. Le résultat de cette extension, baptisé D-RAJE *(Distributed Resource-Aware Java Environment)*, permet de modéliser l'ensemble d'une grappe comme un ensemble de ressources distribuées. En effet chaque nœud de la grappe, chaque composant parallèle, chaque fragment de composant parallèle doivent pouvoir être perçus comme des ressources à part entière au sein de la grappe. D'autre part, la dissémination des diverses ressources existant dans la grappe nous a amenés à définir un modèle dans lequel un composant désirant s'informer sur l'existence ou sur l'état de ces ressources

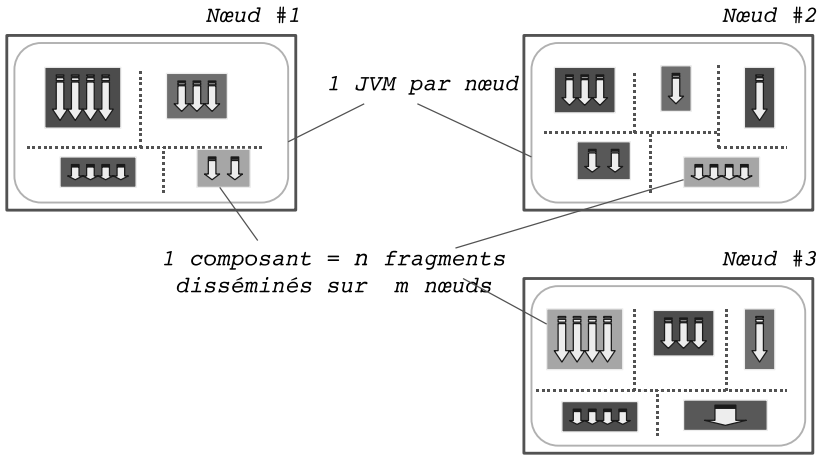


FIG. 2.4 – Exemple de déploiement de composants parallèles

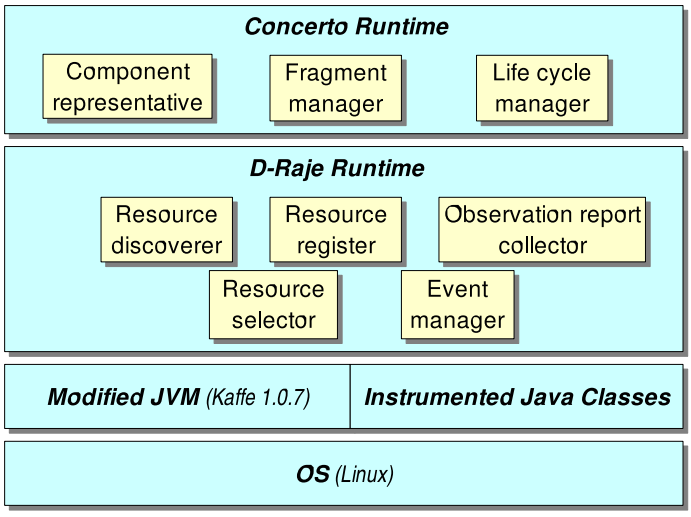


FIG. 2.5 – Structure d'une instance de la plate-forme CONCERTO (destinée à être déployée sur un nœud d'une grappe)

doit pouvoir s'abstraire des contraintes inhérentes à leur répartition physique. En d'autres termes, un composant doit pouvoir s'informer sur l'état des ressources disponibles sur un nœud particulier de la grappe, mais il doit aussi pouvoir collecter des informations concernant les ressources disséminées à travers l'ensemble de la grappe.

L'approche utilisée dans D-RAJE pour la modélisation des ressources est assez similaire à celle adoptée dans le *Common Information Model* [43]. Cependant D-RAJE va au-delà de la simple modélisation. En plus de permettre que les ressources soient modélisées et gérées à travers des objets Java, D-RAJE fournit des mécanismes permettant aux composants applicatifs :

- de découvrir l'existence de ressources spécifiques (ou de types de ressources spécifiques) dans leur environnement ;
- de rechercher des ressources spécifiques (ou de types de ressources spécifiques) dans leur environnement ;
- d'obtenir des informations sur l'état des ces ressources, suivant plusieurs modalités (observation directe ou notification sur événement)

L'intergiciel D-RAJE définit un cadre de conception pour la modélisation et l'observation des ressources. Contrairement aux approches visant la sécurité des applications (*e.g.* [37, 49, 45, 8, 7]), il définit des mécanismes génériques afin de faciliter l'intégration et l'observation de tout type de ressource.

L'aspect distribué des ressources est bien évidemment pris en compte. Les mécanismes cités ci-dessus sont implémentés de sorte que chaque ressource puisse être identifiée et observée de façon homogène indépendamment de sa localisation. En outre les objectifs de D-RAJE sont différents de ceux de la plupart des outils de modélisation et de monitoring de ressources proposés dans les projets de *Grid Computing* [101]. En effet, que ceux-ci reposent sur l'utilisation d'annuaires (*e.g.* Globus [38], Condor [148]) ou suivent une approche objet (*e.g.* Legion [29], Javalin [137]), les ressources considérées dans ces projets sont la plupart du temps des ressources de « gros grain » telles que des nœuds de calcul ou des unités de stockage. L'objectif de ces environnements est surtout d'être capable de collecter des informations sur des ressources disséminées afin d'ordonnancer l'exécution d'un certain nombre de calculs intensifs.

Modélisation des ressources distribuées. Toutes les ressources susceptibles d'être utilisées par des composants déployés sur la plate-forme CONCERTO devant être modélisées sous la forme d'objets Java, nous avons étendu la hiérarchie de classes modélisant les ressources dans l'intergiciel RAJE (décrit au paragraphe 2.2) en y adjoignant des classes modélisant des ressources spécifiques au domaine du calcul sur grappes. Une nouvelle classe *ClusterNode* a ainsi été définie afin de modéliser chacun des nœuds d'une grappe de machines, cette classe agrégeant en outre des instances des classes *CPU*, *Memory*, et *NetworkInterface* afin de donner une vision combinée des ressources élémentaires existant sur chaque nœud. Des classes *Fragment* et *Component* ont de même été introduites afin de donner corps à des notions propres au projet CONCERTO. Grâce à ces classes, un composant parallèle, tout comme un fragment de composant, peuvent être perçus comme des ressources à part entière au sein de la grappe. On pourra dès lors bénéficier de l'ensemble des services mis en œuvre dans CONCERTO pour gérer les composants déployés sur une grappe, et les fragments déployés sur chaque nœud de cette grappe.

Des classes modélisant les différents types de rapports d'observation pouvant être retournés pour chaque nouvelle catégorie de ressources ont bien sûr également été définies, et viennent ainsi complé-

ter l'ensemble des rapports déjà définis dans l'intergiciel RAJE.

Identification et localisation des ressources distribuées. L'intergiciel RAJE met en œuvre un mécanisme de registre de ressources permettant la découverte et le suivi de l'ensemble des ressources disponibles sur une machine. Dans l'intergiciel D-RAJE intégré à la plate-forme CONCERTO, ce mécanisme a été étendu afin de tenir compte de la distribution des diverses ressources sur l'ensemble de la plate-forme que constitue une grappe de calcul.

Chaque nœud de la grappe met en œuvre un registre local, mais l'ensemble des registres peuvent en outre interagir (via le mécanisme RMI) pour s'échanger des informations, et maintenir ensemble une vision globalisée et cohérente de l'ensemble des ressources de la grappe (voir figure 2.6).

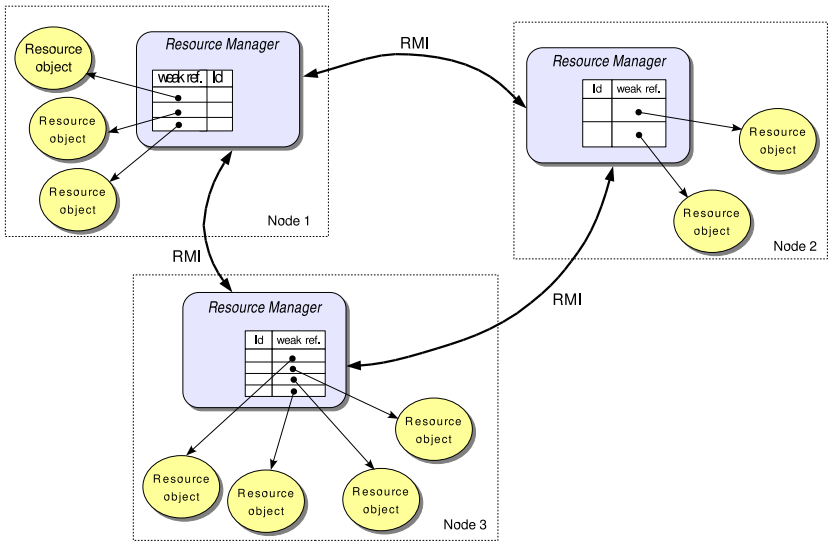


FIG. 2.6 – Architecture du système distribué de gestion de ressources, qui assure le suivi des resources disséminées sur l'ensemble de la grappe

Un gestionnaire de ressources est créé sur chaque nœud de la grappe à chaque fois que l'on déploie un nouveau composant. Sa fonction est de permettre l'identification, la localisation, et la collecte de rapports d'observation auprès :

- des ressources de niveau applicatif (*e.g. threads*, sockets, fichiers, etc.) utilisées par le composant auquel il est associé ;
- des ressources système de la grappe tout entière (considérées comme des ressources globales partagées entre tous les composants) ;
- des autres composants déployés sur la grappe (on rappelle que chaque composant est perçu comme une ressource, et peut donc produire sur demande un rapport d'observation le concernant).

Pour pouvoir cibler la recherche de ressources et la collecte de rapports d'observation, nous avons introduit la notion de « motif de recherche » (voir figure 2.7). L'objectif est ici de pouvoir décrire sous la forme d'objets Java diverses stratégies de recherche, telles que par exemple la recherche localisée (*i.e.* limitée à un nœud précis de la grappe), ou bien encore la recherche globalisée (*i.e.* réalisée sur l'ensemble des nœuds de la grappe).

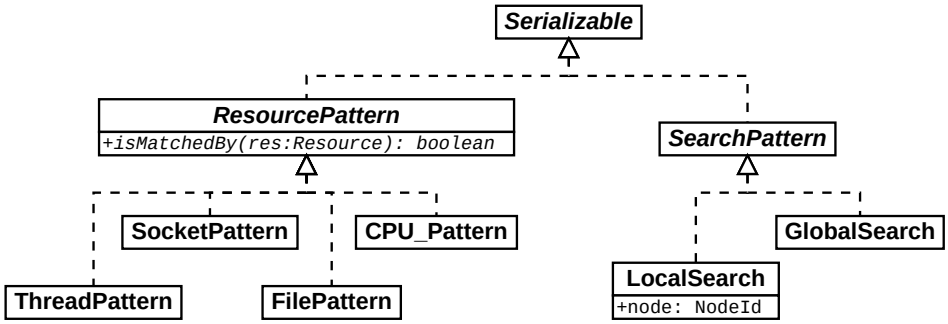


FIG. 2.7 – Modélisation des « motifs » servant à la sélection des ressources (partie gauche de l'arborescence) et à la description des stratégies de recherche (partie droite de l'arborescence) dans CONCERTO

```
ResourceManager manager = ResourceManager.getManager() ;
Set<ResourceId> localIds = manager.getResourceIds(new LocalSearch()) ;     // (1)
Set<ResourceId> remoteIds = manager.getResourceIds(new LocalSearch(remoteNodeId)) ;    // (2)
Set<ResourceId> allIds = manager.getResourceIds(new GlobalSearch()) ;     // (3)
[...]
Set<ObservationReport> localReports = manager.observe(localIds) ;     // (4)
Set<ObservationReport> remoteReports = manager.observe(remoteIds) ;     // (5)
Set<ObservationReport> allReports = manager.observe(allIds) ;     // (6)
```

FIG. 2.8 – Illustration du processus de recherche ciblée de ressources dans la plate-forme CONCERTO, et du processus de collecte de rapports relatifs à ces ressources

L'extrait de code reproduit dans la figure 2.8 illustre le schéma de consultation d'un gestionnaire de ressources. On utilise ici des motifs de recherche afin de préciser que la recherche doit porter (1) sur les ressources locales exclusivement ; (2) sur les ressources recensées sur un nœud distant dont l'identité est passée en paramètre ; (3) sur l'ensemble de la grappe.

Connaissant l'identifiant d'un objet ressource quelconque, on peut obtenir du gestionnaire de ressources qu'il collecte un rapport d'observation concernant cet objet précis (que celui-ci soit local ou distant) et nous retourne le rapport ainsi obtenu. Les étapes (4) à (6) dans la figure 2.8 montrent ainsi comment on peut demander au gestionnaire de ressources local de nous retourner des rapports d'observation concernant des ressources diverses, que celles-ce soient locales ou distantes.

Classification et sélection des ressources. Les ressources enregistrées auprès des gestionnaires de ressources pouvant être de nature très diverse (*CPU*, *Memory*, *Socket*, *Thread*, *File*, etc.), nous avons défini et mis en œuvre un mécanisme de classification et de sélection des ressources basé sur la notion de « motif de ressource ».

Une hiérarchie de classes définissant différents types de motifs a été définie (voir figure 2.7), et ces classes peuvent servir à effectuer des recherches de ressources plus ou moins sélectives dans l'ensemble d'une grappe.

L'exemple reproduit dans la figure 2.9 montre comment l'on peut par exemple définir un motif de recherche de type *SocketPattern* afin de localiser, parmi l'ensemble des ressources disséminées sur la grappe de machines, les ressources de type Socket répondant en outre à certains critères extrêmement

```
ResourceManager manager = ResourceManager.getManager() ;
ResourcePattern socketPattern =
      new SocketPattern(InetAddress.AnyAddress, "195.83.160/24",
                PortRange.AnyPort, new PortRange(0, 1023)) ;    // (1)
Set<ResourceId> socketIds = manager.getResourceIds(socketPattern) ;    // (2)
Set<ObservationReport> socketReports = manager.observe(socketIds) ;    // (3)
```

FIG. 2.9 – Exemple de localisation d'une catégorie particulière de ressources dans l'ensemble de la grappe, et de collecte de rapports d'observation concernant ces ressources spécifiques

précis. Dans le cas présent, le motif construit dans cet exemple va permettre de se focaliser sur les sockets ouverts vers une adresse distante située dans la gamme d'adresses 195.83.160/24, et vers un port compris dans l'intervalle [0, 1023]. Une fois défini ce motif de recherche (1), il suffit d'interroger le gestionnaire de ressources local pour obtenir la liste des objets ressources répondant à ces critères (2), et de demander ensuite au gestionnaire de lancer la collecte de rapports d'observation pour chacun de ces objets ressources (3).

2.3.4 Applications

La plate-forme CONCERTO a été validée en mettant en œuvre et en menant des tests avec plusieurs types de composants parallèles. Cette validation a notamment été réalisée dans le cadre d'une collaboration avec des membres de l'équipe RESO du LIP (ENS Lyon), qui mène des recherches dans le domaine des réseaux actifs orientés hautes performances. L'environnement Tamanoir développé par cette équipe doit répondre aux problèmes d'hétérogénéité et de déploiement dynamique de services posés par l'utilisation de réseaux actifs. Il fournit aux utilisateurs la possibilité de déployer et de maintenir des routeurs actifs appelés TAN (*Tamanoir Active Node*), distribués sur un réseau à grande échelle. Les routeurs actifs, qui permettent d'appliquer dynamiquement des services aux flux qui les traversent sont disposés à la périphérie du *core network* qui doit rester passif pour assurer de très hautes performances.

Le code Java constituant l'environnement Tamanoir a été encapsulé dans des composants parallèles, et ces composants ont ensuite été chargés dans la plate-forme CONCERTO. Ce travail a permis de comparer différentes stratégies de placement et d'adaptation des composants, de confirmer l'intérêt de notre plate-forme pour la définition et le déploiement d'un code parallèle sous forme de composant logiciel, et enfin d'évaluer les facilités offertes au programmeur pour rendre son composant adaptable. Des détails sur ce travail sont disponibles dans le rapport de fin de projet [127].

2.3.5 Principaux résultats

Nos travaux portant sur la supervision de l'accès aux ressources dans les grilles de calculateurs et sur l'hébergement de composants parallèles auto-adaptables sur ce type de plate-forme ont été menés en 2002-2003 dans le cadre du projet CONCERTO (cf. annexe A p. 107).

Ces travaux se sont soldés par la mise en œuvre d'une plate-forme expérimentale opérationnelle, permettant de déployer, d'exécuter, et de superviser l'exécution de composants logiciels parallèles sur une grappe de stations. Cette plate-forme peut être exploitée et contrôlée soit via une simple interface textuelle, soit via une interface graphique (voir figure 2.10).

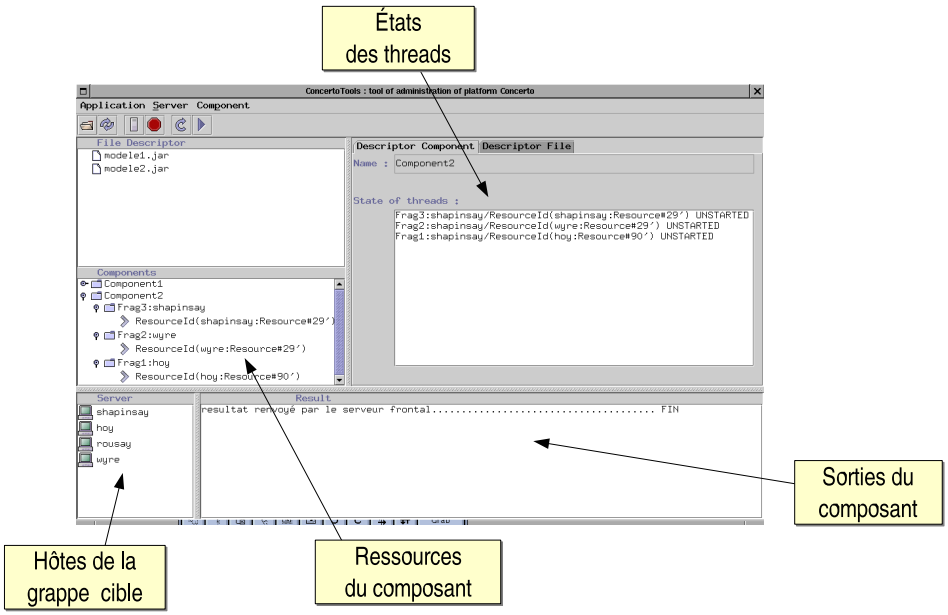


FIG. 2.10 – Aperçu de l'interface graphique permettant l'administration de la plate-forme CONCERTO

Le travail réalisé au cours du projet CONCERTO a fait l'objet des publications suivantes :

– Luc Courtrai, Frédéric Guidec, Nicolas Le Sommer, and Yves Mahéo. *Resource Management for Parallel Adaptive Components.* In Workshop on Java for Parallel and Distributed Computing (à IPDPS'03), pages 134-141, Nice, France, April 2003. IEEE CS. [33]

– Yves Mahéo, Frédéric Guidec, and Luc Courtrai. *A Java Middleware Platform for Resource-Aware Distributed Applications.* In 2nd Int. Symposium on Parallel and Distributed Computing (ISPDC'03), pages 96-103, Ljubljana, Slovenia, October 2003. IEEE CS. [128]

– Yves Mahéo, Frédéric Guidec, and Luc Courtrai. *Middleware Support for the Deployment of Resource-Aware Parallel Java Components on Heterogeneous Distributed Platforms.* In 30th Euromicro Conference - Component-Based Software Engineering Track, pages 144-151, Rennes, France, September 2004. IEEE CS. [129]

– Yves Mahéo, Frédéric Guidec, and Luc Courtrai. *Towards Resource-Aware Parallel Components.* In International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04), pages 1006-1012, Las Vegas, Nevada, USA, June 2004. CSREA Press. [130]

Outre ces publications, ce travail a été présenté à l'occasion de journées thématiques [34, 36] et à l'occasion de l'école d'été GRID'2002 [35]. Il a enfin été détaillé dans le rapport de fin de projet [127].

2.4 Gestion contractualisée de l'accès aux ressources pour les composants logiciels

2.4.1 Motivation

Dans les paragraphes 2.2 et 2.3 on a évoqué le fait qu'il peut être intéressant de concevoir des composants logiciels capables de percevoir les ressources disponibles dans leur environnement d'exécution, et de s'adapter en continu en fonction de leur disponibilité. Dans ce paragraphe nous abordons une approche qui peut être vue comme alternative ou complémentaire de la précédente, et qui consiste à faire en sorte d'adapter ou de dimensionner l'environnement d'exécution lui-même en fonction des besoins des composants. Une telle approche peut s'inscrire dans une démarche générale visant à garantir un certain niveau de qualité de service aux composants, ce qui implique notamment que le service attendu par ces composants soit connu et parfaitement spécifié. Cette même approche peut également permettre d'éviter que des composants en cours d'exécution puissent accaparer des ressources dont ils n'ont a priori pas besoin, ce qui aurait notamment pour effet de compromettre les conditions d'exécution d'autres composants s'exécutant sur la même plate-forme (voire l'intégrité de la plate-forme tout entière). Dans les deux cas il est nécessaire que l'on soit en mesure :

1. d'évaluer et d'exprimer les besoins des composants en termes de ressources nécessaires à leur exécution ;
2. d'évaluer et d'exprimer de même quelles sont les ressources disponibles sur telle ou telle plate-forme cible ;
3. d'utiliser ces informations pour alimenter des processus de décision (ou d'adaptation) visant à déterminer :
   (a) dans quelle mesure un composant peut ou non être chargé et exécuté sur une certaine plate-forme ;
   (b) dans quelle mesure un composant en cours d'exécution peut ou non être autorisé à accéder à une certaine ressource.

Au cours des projets RASC et MASC[7], qui se sont étalés sur la période 2000-2005, nous nous sommes intéressés au problème de gestion contractualisée de l'accès aux ressources de la part des composants logiciels. L'idée générale est que tout composant logiciel susceptible d'être chargé et exécuté sur une certaine plate-forme devrait être en mesure d'exprimer ses attentes vis-à-vis des ressources offertes par cette plate-forme, et de négocier en conséquence les conditions d'accès à ces diverses ressources avec le support exécutif de la plate-forme.

Des travaux similaires ont déjà été menés, la plupart du temps dans une optique de sécurisation des plates-formes d'exécution, sans toutefois que la notion de contrat passé entre composants logiciels et plate-forme d'exécution soit clairement définie et supportée. Ainsi, l'environnement d'exécution de Java (JRE *: Java Runtime Environment*) met en œuvre un modèle de sécurité connu sous le nom de *SandBox*. Dans les premières versions de la JRE, ce modèle donnait au code local — considéré comme sûr — la possibilité d'accéder à toutes les ressources du système. En revanche un code distant (téléchargé sous la forme d'une *applet)* était considéré comme suspect, et se voyait interdire l'accès à la plupart des ressources sensibles du système [52]. Avec la plate-forme Java 2 ce modèle de type « tout ou rien » fut abandonné au profit d'un nouveau modèle reposant sur la notion de « domaine de

[7] Ces projets sont décrits en détails dans l'annexe A p. 107.

protection » *(protection domain)* [52, 53, 166]. Un domaine de protection constitue un environnement d'exécution dont la politique de sécurité peut être spécifiée sous la forme de permissions. Un contrôleur d'accès attaché à chaque domaine de protection supervise les accès aux ressources et applique la politique de sécurité définie par les permissions accordées au domaine. J-Kernel étend cette notion de domaine de protection en offrant un mécanisme de communication et de partage de données entre domaines [93]. La communication est cependant limitée aux appels de méthodes sur des objets nommés *capabilities*.

Le modèle de sécurité mis en œuvre dans J-Kernel et dans le JRE repose sur des mécanismes « sans état ». On ne peut conditionner l'accès à une certaine ressource en fonction des accès réalisés précédemment à cette même ressource. On ne peut donc poser des contraintes d'ordre quantitatif (quantité de CPU, quotas d'entrée-sortie, etc.) sur les ressources manipulées au sein d'un domaine de protection. Les mécanismes de J-Kernel et du JRE ne permettent donc pas de prévenir les dysfonctionnements résultant de l'utilisation abusive d'une certaine ressource (attaques de type déni de service, etc.).

Comme expliqué dans le paragraphe 2.2.1, des environnements tels que JRes [9, 37], GVM [8], et KaffeOS [7] offrent des mécanismes permettant de comptabiliser — voire limiter — l'utilisation de chaque type de ressource par une entité active (un *thread* dans le cas de JRes, un processus dans le cas de GVM et de KaffeOS), mais les ressources sur lesquelles portent la comptabilisation et le contrôle sont des ressources globales.

Les projets Naccio [44, 45] et Ariel proposent chacun un langage et des mécanismes permettant de définir de manière très précise la politique de sécurité qui doit être appliquée à un programme d'application lors de son exécution. L'application d'une politique de sécurité est réalisée statiquement, par réécriture du *bytecode* du programme d'application, mais aussi des classes de l'API de la plate-forme Java. Les auteurs de Naccio précisent fort justement que cette approche permet de réduire le surcoût engendré par la supervision d'un programme au cours de son exécution. En revanche la génération d'une nouvelle API vérifiant une politique de sécurité donnée est une opération extrêmement coûteuse. Cette approche se prête donc bien à la génération anticipée d'un ensemble d'API pré-définies garantissant chacune le respect d'une politique de sécurité générique. Par contre elle ne permet pas d'assurer la supervision d'un programme d'application en fonction d'une politique de sécurité définie à partir des besoins exprimés par ce même programme lors de son démarrage.

La gestion sous forme contractualisée de la qualité de service dans les systèmes d'objets distribués a fait l'objet de nombreux travaux, e.g. [50, 13, 145]. En général les approches proposées impliquent la définition d'un formalisme permettant d'exprimer les caractéristiques de qualité de service attendues ou offertes par les diverses entités du système considéré (*e.g.* langage QML dans le cas de [50]). Elles impliquent en outre la mise en œuvre de mécanismes capables de vérifier que les contrats passés entre ces entités sont respectés, voire d'agir sur ces entités afin de garantir le respect de ces contrats.

Les paragraphes qui suivent fournissent un aperçu d'une plate-forme que nous avons développée dans la même optique que les travaux qui viennent d'être évoqués. Cette plate-forme, baptisée JAMUS (*Java Accommodation of Mobile Untrusted Software*), a été conçue dans le cadre du projet RASC en vue d'assurer l'hébergement sécurisé de programmes mobiles Java « non dignes de confiance »[8]. Elle utilise les services de supervision de l'accès aux ressources fournis par l'intergiciel RAJE (tel que décrit dans le paragraphe 2.2) afin de prévenir et de sanctionner tout comportement non prévu de la part des programmes qu'elle héberge. Chaque programme candidat à l'hébergement par la plate-

[8]Une description beaucoup plus détaillée de cette plate-forme est fournie dans le mémoire de thèse de Nicolas Le Sommer [105], ainsi que dans certaines des publications listées dans le paragraphe 2.4.7.

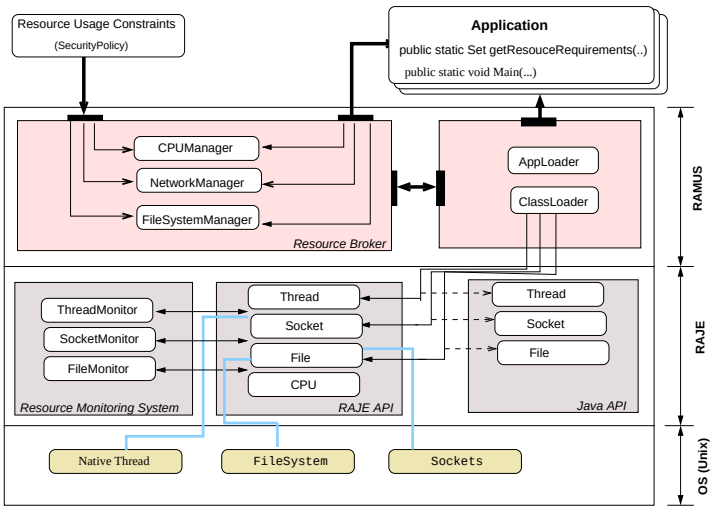


FIG. 2.11 – Architecture générale de la plate-forme JAMUS

forme doit en effet décrire ses besoins en termes de ressources sous forme qualitative (*e.g.*, droits d'accès à tout ou partie du système de fichiers) et quantitative (*e.g.*, quotas d'accès requis en lecture et en écriture). La plate-forme est capable d'analyser ces besoins en les confrontant à un ensemble de contraintes posées sur l'utilisation des ressources dont elle dispose. Elle met en œuvre un mécanisme de contrôle d'admission reposant sur le principe de la réservation de ressources afin de décider si elle peut ou non accepter d'héberger un programme. Dans l'affirmative, le programme accueilli est soumis à une supervision constante au cours de son exécution, la plate-forme empêchant toute utilisation de ressources non conforme aux besoins exprimés initialement par ce programme.

La plate-forme JAMUS permet donc aux programmes candidats à l'hébergement de négocier par contrat l'accès aux ressources qui leurs sont nécessaires, et elle assure ensuite la disponibilité de ces ressources, en prévenant notamment toute tentative d'accaparement de ressources par des composants ne respectant pas leurs engagements contractuels. Contrairement à des modèles de gestion de contrats tels que ceux proposés dans [50, 13, 145], le modèle mis en œuvre dans la plate-forme JAMUS permet une relation totalement symétrique entre les deux parties contractantes. Un programme hébergé par JAMUS peut à tout instant initier une phase de renégociation du contrat passé avec la plate-forme, mais cette dernière peut également imposer au programme une renégociation de ses conditions d'exécution si celles-ci sont amenées à évoluer.

2.4.2 Vue d'ensemble de la plate-forme JAMUS

L'architecture générale de la plate-forme JAMUS est présentée dans la figure 2.11. Comme indiqué plus haut tout programme candidat à l'hébergement par la plate-forme doit être capable d'exprimer ses besoins en décrivant les ressources qui vont lui être utiles au cours de son exécution, et les conditions dans lesquelles il doit pouvoir accéder à ces ressources. En exprimant ses besoins, un programme demande à bénéficier d'un certain service de la part de la plate-forme, et s'engage dans le même temps à ne pas utiliser d'autres ressources que celles dont il fait explicitement la demande. De son côté, la

plate-forme acceptant d'accueillir un programme s'engage à lui fournir les ressources demandées, et se réserve en même temps le droit de sanctionner le programme hébergé si le comportement de celui-ci n'est pas conforme aux engagements pris lors du contrôle d'admission.

L'originalité de la plate-forme JAMUS réside principalement dans la réciprocité des engagements pris entre la plate-forme d'accueil et les programmes qu'elle héberge. Les besoins exprimés par un programme sont à la fois interprétés comme tels, et comme un engagement à ne pas chercher à accéder à d'autres ressources que celles dont il est fait explicitement mention. Grâce à cet engagement, JAMUS peut offrir une certaine qualité de service aux programmes hébergés, tout en leur assurant une relative sécurité au cours de leur exécution (sécurité résultant essentiellement de la prévention du déni de service). Ces caractéristiques résultent de l'application des deux principes suivants au sein de la plate-forme JAMUS.

Principe du contrôle d'admission. Tout programme candidat à l'hébergement doit passer avec succès une épreuve de contrôle d'admission. Ce contrôle est assuré par un courtier de ressources, capable d'interroger le programme afin de connaître ses besoins propres, et d'utiliser cette information pour décider du sort de ce programme. Les besoins exprimés par le programme le sont sous la forme de profils d'utilisation de ressources, dont la structure est détaillée dans la section 2.4.3. Le fonctionnement du courtier de ressources est quant à lui décrit dans la section 2.4.4.

Principe de la supervision des programmes hébergés. Le fait qu'un programme ait passé avec succès l'épreuve du contrôle d'admission ne signifie pas nécessairement qu'en cours d'exécution il se contentera d'utiliser les ressources qui lui ont été accordées à l'issue de ce contrôle. En effet un programme conçu de façon maladroite ou dans une optique malveillante pourrait tenter d'accéder à des ressources qui ne lui sont *a priori* pas destinées, ou bien encore chercher à consommer plus de ressources qu'il ne lui en a été accordé. Un programme accueilli par la plate-forme JAMUS est donc considéré comme étant *a priori* non digne de confiance. En conséquence son exécution doit être supervisée afin de vérifier que l'utilisation qu'il fait des ressources demeure conforme aux modalités convenues lors du contrôle d'admission. Les mécanismes assurant la supervision d'un programme en cours d'exécution sont présentés dans la section 2.4.6.

2.4.3 Profils d'utilisation de ressources

Lors de son lancement la plate-forme d'accueil JAMUS reçoit en paramètres un ensemble de contraintes précisant les ressources qu'elle va pouvoir mettre à disposition des programmes hébergés, ainsi que les modalités d'utilisation de ces ressources (permissions et quotas). Ces informations vont permettre au courtier de ressources d'initialiser les gestionnaires de ressources grâce auxquels il va pouvoir assurer le suivi des ressources utilisées et disponibles sur la plate-forme d'accueil.

Les informations décrivant les ressources et leurs modalités d'accès sont fournies sous la forme de profils d'utilisation de ressources, instanciées à partir de la classe *ResourceUsageProfile* (voir figure 2.12). Cette classe permet de décrire à la fois les contraintes posées sur les ressources offertes par la plate-forme JAMUS, et les exigences des programmes d'application vis-à-vis de ces mêmes ressources. Un profil d'utilisation de ressources est décrit sous la forme d'un objet possédant trois attributs *pattern*, *permission*, et *quota*, référençant respectivement des objets implémentant les interfaces *ResourcePattern*, *ResourcePermission*, et *ResourceQuota* (voir fig. 2.12). Il existe des implé-

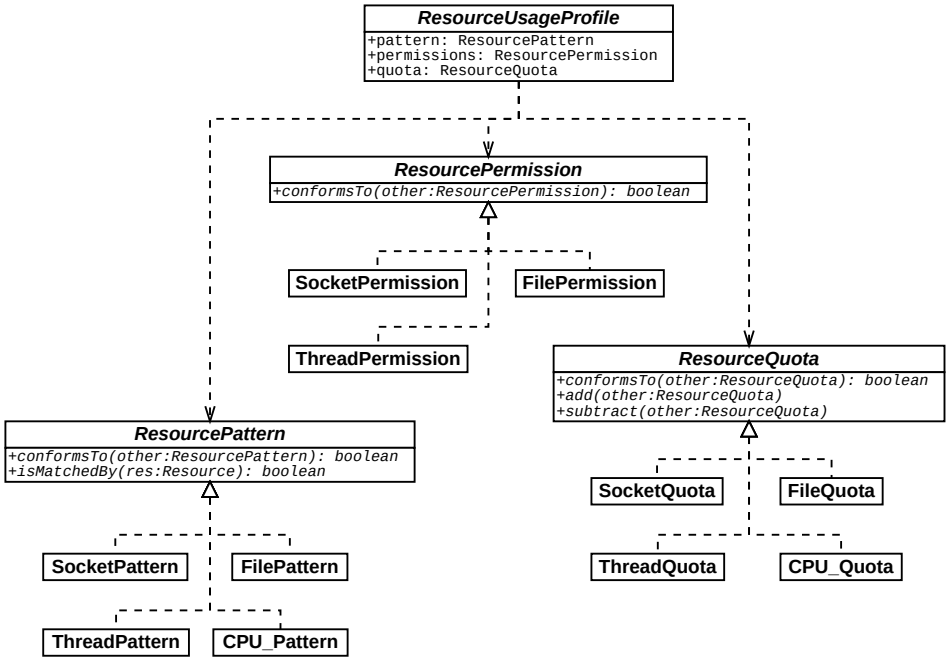


FIG. 2.12 – Modélisation des profils d'utilisation de ressources sous forme d'objets.

mentations spécialisées de ces interfaces pour chaque type de ressource considéré à l'heure actuelle dans l'environnement RAJE. En outre de nouvelles classes peuvent aisément être développées afin de décrire de nouvelles modalités d'utilisation pour des ressources déjà prises en compte, ou bien afin d'intégrer de nouveaux types de ressources dans le système.

L'attribut *pattern* d'un profil d'utilisation sert à identifier précisément sur quelle catégorie de ressources portent les besoins ou contraintes exprimées dans ce profil. Il permettra de sélectionner en cours d'exécution parmi les divers objets modélisant des ressources dans le système ceux qui seront concernés par ce profil d'utilisation. L'attribut *permission* précise quelles sont les opérations permises (par le système) ou requises (par un composant) sur le type de ressource considéré. Il définit donc les modalités d'utilisation des ressources sous une forme qualitative. L'attribut *quota* complète cette information en précisant pour sa part les modalités d'utilisation des ressources sous une forme quantitative.

La figure 2.13 illustre la manière selon laquelle un programme d'application susceptible d'être accueilli par la plate-forme JAMUS peut exprimer ses besoins vis à vis des ressources offertes par cette plate-forme. Dans cet exemple le programme définit la méthode statique *getResourceRequirements()*, à laquelle le lanceur de la plate-forme JAMUS fera systématiquement appel pour se renseigner sur les besoins des programmes candidats à l'hébergement. Dans le cas présent, les besoins du programme de la figure 2.13 se déclinent en quatre profils d'utilisation distincts, dont les deux premiers concernent l'utilisation de ressources de type *Socket*, et les deux derniers portent sur l'utilisation du système de fichiers. Une analyse détaillée des besoins exprimés par le programme considéré dans la figure 2.13 est disponible dans [113].

```
public class MyProgram {
  public static Set getResourceRequirements(String[] args) {
    Set requirements = new HashSet() ;
    int ko = 1024 ; int Mo = 1024*1024 ;

    // Profil d'utilisation global s'appliquant à toutes les communications par sockets :
    // 12 sockets ouverts max., quota de 2 Mo en émission, 8 Mo en réception.
    requirements.add(new ResourceUsageProfile(new SocketPattern(),
                                              new SocketPermission(SocketPermission.all),
                                              new SocketQuota(12, 2*Mo, 8*Mo))) ;

    // Profil d'utilisation sélectif ne concernant que les connexions établies
    // vers des hôtes du domaine 195.83.160/24 :
    // 4 sockets ouverts max., quota de 512 koctets en émission, 128 koctets en réception.
    requirements.add(new ResourceUsageProfile(new SocketPattern("195.83.160/24"),
                                              new SocketPermission(SocketPermission.all),
                                              new SocketQuota(4, 512*ko, 128*ko))) ;

    // Profil d'utilisation global s'appliquant à tous les accès au système de fichiers :
    // 8 fichiers ouverts max., quota de 1 Mo en écriture, 2 Mo en lecture.
    requirements.add(new ResourceUsageProfile(new FilePattern("/tmp"),
                                              new FilePermission(FilePermission.all),
                                              new FileQuota(8, 1*Mo, 2*Mo))) ;

    // Profil d'utilisation sélectif ne concernant que le sous répertoire /tmp/myStuff :
    // 4 fichiers ouverts max., quota de 5 ko en lecture, écriture interdite.
    requirements.add(new ResourceUsageProfile(new FilePattern("/tmp/myStuff"),
                                              new FilePermission(FilePermission.readOnly),
                                              new FileQuota(4, 0, 5*ko))) ;

    return requirements;
  }

  public static void main(String[] args) { . . . } :
}
```

FIG. 2.13 – Exemple de programme d'application capable d'exprimer ses besoins en termes de ressources via la méthode *getResourceRequirements()*

```
<?xml version="1.0"?>
<!DOCTYPE resource-setup SYSTEM "resource-setup.dtd">
<resource-setup>
 <filesystem>
  <directory directory-path="/tmp">
   <directory-permissions read="yes" write="yes"/>
   <directory-quotas read="8MB" write="8MB"/>
   <allocation-mode="reservation" />
  </directory>
</filesystem>
<network>
 <domain domain-name="*" >
  <domain-remote-port value="80"/>
  <domain-protocol name="TCP"/>
  <domain-permission receive="yes" send="yes"/>
  <domain-quota receive="18GB" send="80MB"/>
  <allocation-mode="best effort" />
 </domain>
</network>
</resource-setup>
```

FIG. 2.14 – Exemple de spécification en XML des ressources disponibles dans la plate-forme JAMUS

2.4.4 Courtier de ressources

Le courtier de ressources intégré à la plate-forme JAMUS met en œuvre un mécanisme de réservation de ressources (inspiré du mécanisme décrit dans [99]) afin de garantir aux programmes hébergés la disponibilité des ressources dont ils ont fait la demande. Il s'appuie sur des gestionnaires de ressources dédiés chacun à un type de ressource particulier. La plate-forme intègre en fait trois gestionnaires de ressources distincts, qui sont responsables respectivement de la gestion des ressources relatives au réseau, au CPU, et au système de fichiers (voir figure 2.11).

Contrôle d'admission. Lors du lancement de la plate-forme le courtier de ressources distribue à chaque gestionnaire de ressources un ensemble de profils d'utilisation décrivant les contraintes posées sur les ressources disponibles sur la plate-forme et dont ce gestionnaire va devoir se préoccuper. Ces contraintes sont exprimées dans un fichier XML (voir l'exemple reproduit dans la figure 2.14), qui est fourni à la plate-forme JAMUS dès son lancement.

Lorsqu'un programme candidat à l'hébergement est soumis au contrôle d'admission, le courtier de ressources analyse les différentes demandes formulées par ce programme, demandes qui sont également exprimées sous la forme de profils d'utilisation de ressources. Pour chaque demande, le courtier identifie le gestionnaire de ressources concerné, et le charge de décider de l'admissibilité de cette demande. Un programme ne peut être admis que si les gestionnaires ont admis toutes les demandes formulées par ce programme.

Réservation et libération des ressources. La politique d'allocation des ressources mise en œuvre dans la plate-forme JAMUS est basée sur le modèle de la réservation. Lors de l'analyse d'une demande formulée par un programme d'application, le gestionnaire réalisant cette analyse identifie les

contraintes dont le profil coïncide avec celui de la demande, et s'assure que les permissions et quotas d'accès demandés par le programme candidat sont conformes aux permissions accordées dans ces contraintes.

Si ces tests de conformité se terminent avec succès, cela signifie que la demande considérée peut *a priori* être satisfaite sans violer les contraintes posées sur les ressources disponibles. Cependant le programme ayant formulé cette demande ne pourra être déclaré admissible que si toutes les autres demandes formulées par le programme peuvent également être satisfaites.

Dans l'affirmative, une deuxième passe est réalisée sur les demandes du programme afin que les gestionnaires de ressources lui réservent effectivement les ressources demandées. Pour chaque demande ainsi examinée une seconde fois, la réservation s'effectue en prélevant sur les quotas associés aux contraintes concernées les valeurs de quotas associés à cette demande.

Lorsqu'un programme hébergé par la plate-forme JAMUS arrive au terme de son exécution, le courtier de ressources déclenche une série d'opérations inverses à celles décrites ci-dessus afin de recréditer au niveau des gestionnaires de ressources les ressources qui avaient été réservées pour ce programme.

2.4.5 Gestionnaire de contrats

Lorsqu'un programme applicatif franchit avec succès la phase de contrôle d'admission, une relation contractuelle est établie de fait entre ce programme et la plate-forme JAMUS qui accepte de l'accueillir. En effet, la spécification des besoins en ressources exprimés par ce composant peut être interprétée comme un engagement à ne pas chercher à utiliser d'autres ressources que celles pour lesquelles il a fait explicitement une demande. Le programme applicatif s'engage donc vis-à-vis de la plate-forme JAMUS à adopter un certain comportement, et à ne pas s'en éloigner sous peine d'être considéré comme étant responsable d'une violation du contrat passé avec la plate-forme. De son côté, la plate-forme JAMUS acceptant d'accueillir un composant sous certaines conditions s'engage à lui garantir le maintien de ces conditions d'exécution. Le mécanisme de réservation de ressources est d'ailleurs censé contribuer au respect de cet engagement pris par la plate-forme, en réduisant le risque que des ressources critiques pour un composant deviennent indisponibles au cours de son exécution.

La figure 2.15 illustre un scénario au cours duquel un programme candidat à l'hébergement par la plate-forme JAMUS soumet au gestionnaire de contrats de la plate-forme deux contrats successifs, correspondant chacun à une demande différente en termes de modalités d'accès aux ressources requises par ce programme. Pour chaque contrat proposé, le gestionnaire de ressource consulte à son tour le courtier de ressources afin de s'enquérir de la disponibilité des ressources demandées. Dans le cas du premier contrat proposé, la demande formulée par le programme candidat ne peut être satisfaite. Le gestionnaire de contrats obtient du courtier de ressources un descriptif des clauses du contrat qui ne peuvent être satisfaites. Ce descriptif est retourné au programme afin que celui-ci puisse éventuellement ajuster sa demande en conséquence. Dans le scénario considéré ici, le programme soumet effectivement une deuxième demande au gestionnaire de contrats, et cette demande peut cette fois être satisfaite par le courtier de ressources. Celui-ci réserve alors les ressources demandées par le programme (avec l'accord du gestionnaire de contrats), et le programme candidat reçoit finalement confirmation qu'il a été admis pour exécution sur la plate-forme.

FIG. 2.15 – Exemple de négociation des modalités d'accès aux ressources entre un programme candidat à l'hébergement et la plate-forme JAMUS

Support dynamique des avenants aux contrats. Les besoins d'un programme en termes de ressources peuvent évoluer alors même que ce programme s'exécute, et cette évolution n'est pas nécessairement prédictible avant le début de l'exécution. Les ressources disponibles au niveau de la plate-forme JAMUS peuvent également fluctuer dynamiquement (une interface de communication peut par exemple être tantôt disponible, tantôt indisponible, sans que le support exécutif JAMUS soit directement responsable de telles fluctuations). Il peut donc être nécessaire, dans certaines circonstances, de réviser périodiquement le contrat passé entre un programme et la plate-forme qui l'accueille. Un mécanisme de négociation d'avenants a donc été défini et mis en œuvre au niveau du gestionnaire de contrats. Grâce à ce mécanisme, un programme s'exécutant sur la plate-forme peut à tout instant proposer des avenants au contrat négocié lors de son admission initiale, ces avenants lui permettant par exemple de négocier l'accès à de nouvelles ressources, de modifier les conditions d'accès à certaines ressources auxquelles il avait déjà accès, voire de renoncer à des ressources dont il n'a plus l'usage.

2.4.6 Supervision des programmes en cours d'exécution

Chaque programme d'application hébergé par la plate-forme JAMUS s'exécute sous la supervision d'un moniteur de composant, instance de la classe *ComponentMonitor* définie dans la hiérarchie de classes de l'intergiciel RAJE. Ce moniteur utilise les profils d'utilisation fournis par le programme pour instancier des moniteurs de ressources spécialisés chargés de vérifier le respect de ces profils. Un moniteur de ressource admet en paramètre lors de sa création un profil d'utilisation de ressources. Son rôle est de superviser l'utilisation qui est faite d'une certaine catégorie de ressources (celles qui satisfont le critère de sélection exprimé par l'attribut *pattern* du profil d'utilisation associé), et de

s'assurer que cette utilisation est conforme aux contraintes exprimées par les attributs *permission* et *quota* de ce même profil.

L'environnement RAJE fournit une catégorie de moniteurs de ressources spécifique pour chaque type de ressource considéré. Ainsi la classe *SocketMonitor* définit un type de moniteur capable de superviser l'utilisation des ressources de type *Socket*. Un *SocketMonitor* admet en paramètre lors de sa création un profil d'utilisation dont les attributs référencent des objets de type *SocketPattern*, *SocketPermission*, et *SocketQuota*. Il a donc pour fonction de vérifier pendant l'exécution d'un programme d'application que les *sockets* créés par celui-ci et qui satisfont le critère de sélection du *pattern* spécifié dans le profil sont utilisés conformément aux modalités précisées par les attributs *permission* et *quota* de ce profil.

Par exemple, lors du lancement du programme d'application reproduit dans la figure 2.13, un moniteur de ressources spécifique va être lancé pour chacun des quatre profils d'utilisation définis par ce programme pour exprimer ses besoins. Les divers moniteurs ainsi créés vont superviser les accès au réseau et au système de fichiers de la part du programme, et s'opposer si nécessaire à toute tentative d'accès non conforme aux besoins exprimés initialement par le programme. En cas de violation des modalités d'accès aux ressources définies contractuellement entre un programme d'application et la plate-forme JAMUS qui l'héberge, JAMUS dispose de plusieurs méthodes pour sanctionner le programme fautif, et peut être configurée afin d'adopter une politique plus ou moins draconienne vis-à-vis de ce type de programme. La méthode la moins « agressive » consiste simplement à lever une exception afin de l'informer qu'il sort du cadre d'utilisation des ressources défini initialement. À l'extrême inverse, la plate-forme peut mettre un terme à l'exécution d'un programme fautif, libérant ainsi toutes les ressources qu'il consommait jusqu'alors. Des méthodes intermédiaires peuvent également être appliquées, en verrouillant par exemple les ressources qu'un programme utilise à tort (ou de façon disproportionnée), et en obligeant ce programme à négocier un avenant au contrat qui le lie à la plate-forme s'il entend continuer à utiliser ces ressources.

2.4.7 Principaux résultats

Nos travaux portant sur la contractualisation de l'accès aux ressources pour les composants logiciels ont dans un premier temps été menés entre 2000 et 2003 dans le cadre du projet RASC, qui servait alors de fil conducteur au travail de thèse de Nicolas Le Sommer. Ce travail s'est notamment soldé par le développement de la plate-forme d'hébergement sécurisé JAMUS, par les publications suivantes, ainsi bien sûr que par le mémoire de thèse de Nicolas Le Sommer [105].

- Nicolas Le Sommer and Frédéric Guidec. *A Contract-Based Approach of Resource-Constrained Software Deployment.* In Judith Bishop, editor, 1st International IFIP/ACM Working Conference on Component Deployment (CD 2002), volume 2370 of LNCS, pages 15-30, Berlin, Germany, June 2002. Springer Verlag. [112]

- Nicolas Le Sommer and Frédéric Guidec. *JAMUS : une plate-forme d'accueil sécurisée pour le code mobile.* In M. Dao and M. Huchard, editors, Langages et Modèles à Objets (LMO 2002), L'Objet, pages 203-215, Vannes, France, February 2002. Hermes Science. [113]

- Nicolas Le Sommer and Frédéric Guidec. *JAMUS : Java Accommodation of Mobile Untrusted Software.* In 4th Nord EurOpen/Usenix Conference (NordU 2002), pages 38- 48, Helsinki, Finland, February 2002. Multiprint, Helsinki. Best Paper. [114]

Par la suite, les résultats de ces travaux ont été repris et enrichis dans le cadre du projet MASC *(Mobile Adaptive Software Components)*, financé par la Région Bretagne entre 2002 et 2005, et dans lequel s'inscrivait notamment le travail de thèse d'Hervé Roussain (cf. annexes A et B). Ce travail s'est soldé par le développement de la plate-forme expérimentale JASON, permettant le déploiement de composants logiciels en mode pair-à-pair sur des terminaux mobiles assemblés en réseau ad hoc discontinu, et l'hébergement sécurisé de ces composants sur chaque terminal grâce au support exécutif fourni par JAMUS. Le développement de la plate-forme JASON est notamment relaté dans les publications listées ci-dessous. Les aspects relatifs au déploiement de code en mode pair-à-pair à l'aide de cette plate-forme sont quant à eux présentés en détails dans le paragraphe 4.2.2.

- Nicolas Le Sommer, Frédéric Guidec, and Hervé Roussain. *A Context-Aware Middleware Platform for Autonomous Application Services in Dynamic Wireless Networks.* In First International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06), number 9, Nice, France, May 2006. ACM Digital Library. [116]
- Hervé Roussain, Nicolas Le Sommer, and Frédéric Guidec. *Towards an Asynchronous Dissemination and a Safe Deployment of Lightweight Programs in Mobile Networks.* In IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2005), pages 481-483, Taormina, Italy, June 2005. IEEE CS. [154]

Chapitre 3

Communication dans les réseaux mobiles ad hoc discontinus

3.1 Introduction

De nos jours la plupart des ordinateurs portables, assistants numériques personnels (PDA: *Personal Digital Assistant*), et autres *tablet PCs* sont pourvus d'interfaces de communication sans fils. Des standards de communication tels que Wi-Fi [1] (IEEE 802.11 [1]), WiMAX[2] (IEEE 802.16 [3]), et plus généralement les normes de téléphonie cellulaire (*i.e.* GSM/GPRS, etc.) permettent de connecter de tels équipements à un réseau d'infrastructure, via des stations de base ou des points d'accès jouant le rôle de passerelles entre l'univers sans fils et l'univers filaire (y compris Internet).

Le fait de permettre à des terminaux mobiles d'accéder, via des passerelles adéquates, aux vastes ressources offertes par les réseaux d'infrastructure présente des avantages indéniables. On téléphone ainsi aussi facilement à son voisin qu'à une personne située à des milliers de kilomètres de là. On expédie et reçoit du courrier électronique depuis la terrasse d'un café, depuis un hall de gare, voire depuis la cabine d'un avion de ligne. Cependant, avec cette approche, les terminaux mobiles sont étroitement dépendants de l'accessibilité et de la disponibilité d'une infrastructure de communication fixe, sans laquelle aucun service ne saurait être fourni aux utilisateurs nomades. Ainsi, deux utilisateurs dotés de terminaux GSM et situés à quelques centaines de mètres l'un de l'autre ne sont pas en mesure d'établir entre eux une communication téléphonique s'ils ne se trouvent pas tous deux dans la zone de couverture d'un réseau GSM. De la même manière, un réseau reposant sur des points d'accès Wi-Fi ne présente d'intérêt que pour les utilisateurs qui se situent dans les zones couvertes par ces points d'accès.

Des technologies de transmission sans fils telles que Wi-Fi et Bluetooth (IEEE 802.15 [2]) per-

[1] Dans ce document nous nous conformons à l'usage commun, et utilisons le terme Wi-Fi pour désigner des équipements conformes à la norme IEEE 802.11. En fait , Wi-Fi *(Wireless Fidelity)* est une marque déposée par la *Wi-Fi Alliance*, qui a défini des jeux de tests censés attester de la conformité d'un équipement à la norme 802.11, et qui n'accorde l'estampille Wi-Fi qu'aux équipements ayant subi ces tests avec succès. Les procédures de test étant extrêmement coûteuses, un fabricant de matériel peut fort bien refuser de s'y soumettre, et mettre sur le marché un équipement qui sera conforme en tous points à la norme 802.11 sans pour autant avoir reçu l'estampille commerciale « Wi-Fi ».

[2] À l'instar du terme Wi-Fi, le terme WiMAX *(Worldwide Interoperability for Microwave Access)* est une marque commerciale déposée par le *WiMAX Forum*. On l'utilise donc, à tort, pour désigner les équipements conformes à la famille — sans cesse grandissante — de normes IEEE 802.16.

mettent pourtant à des terminaux mobiles de communiquer directement les uns avec les autres, via des liaisons radio à courte portée, et sans avoir recours pour ce faire à des équipements d'infrastructure. On utilise communément l'expression « communication ad hoc » pour désigner ce mode de communication, et l'on parle par extension de « réseau ad hoc » pour désigner un ensemble d'au moins deux équipements interagissant sans passer par un réseau d'infrastructure [143].

Du point de vue d'un programme applicatif s'exécutant sur un équipement mobile, l'aptitude à pouvoir communiquer avec d'autres programmes s'exécutant sur d'autres équipements est bien sûr primordiale dès lors qu'on cherche à développer des applications réparties. Pourtant, un équipement mobile est susceptible d'observer une connectivité intermittente et des débits variables, ce qui peut avoir des effets non négligeables sur les applications qu'il supporte. Le réseau constitue donc bien une ressource à part entière sur un équipement mobile, la disponibilité tout comme les propriétés de cette ressource pouvant fluctuer considérablement alors même que des applications qui en dépendent sont en cours d'exécution.

L'équipe CASA a commencé à s'intéresser dès le début des années 2000 à cette « ressource » particulière qu'est le réseau, en concentrant son attention sur les problèmes spécifiques posés par la communication dans les réseaux mobiles ad hoc. Le paragraphe 3.2 présente un panorama des travaux de la communauté scientifique dans ce domaine, tout en mettant l'accent sur les difficultés présentées par certains réseaux dits « discontinus », pour lesquels il n'existe pas encore de solutions véritablement satisfaisantes. Nos propres travaux sont ensuite présentés dans le paragraphe 3.3.

3.2 Problématique des réseaux mobiles ad hoc discontinus

> Ad hoc : locution latine signifiant littéralement « qui va vers ce vers quoi il doit aller », ou encore « formé dans un but précis. » On utilise généralement l'expression « ad hoc » pour qualifier une solution qui a été conçue spécifiquement afin de résoudre un problème précis, et qui de ce fait est difficilement généralisable, et difficilement applicable à d'autres types de problèmes.

Notion de réseau ad hoc. Dans le domaine des réseaux de télécommunication, le terme « ad hoc » est en général utilisé pour désigner un type particulier de réseau sans fils, dans lequel l'architecture même du réseau — ainsi que ses conditions de fonctionnement — posent des problèmes qui ne se posent pas dans les réseaux filaires traditionnels, et qui appellent donc des solutions adaptées à ce type précis de réseau.

Les réseaux ad hoc ont fait l'objet de nombreux travaux de recherche au cours de la dernière décennie. Ces travaux ont dans un premier temps été menés essentiellement à des fins d'applications militaires. Dans ce domaine, l'objectif est notamment de pouvoir disposer de réseaux capables de s'auto-configurer *(self-configuration)*, voire de « s'auto-réparer » *(self-healing)* en cas de perte de certains éléments du réseau (nœuds ou liens). En outre, dans les réseaux tactiques certains éléments du réseau doivent pouvoir être mobiles (*e.g.* équipements embarqués dans des véhicules ou transportés par des fantassins) et capables de fonctionner entre ou pendant les déplacements. Depuis quelques années des applications civiles sont également envisagées. La communication en mode ad hoc peut en effet se justifier dès lors que le recours à un réseau d'infrastructure s'avère soit techniquement difficile,

soit économiquement peu rentable (*e.g.* secouristes intervenant à la suite d'une catastrophe naturelle, équipes de scientifiques travaillant en terrain désertique, systèmes de communication inter-véhicules, réseaux de capteurs, etc.).

À ce jour il n'existe pas à proprement parler d'architecture générique de réseau ad hoc, mais plutôt une constellation d'architectures particulières que l'on peut s'efforcer de classer en grandes « familles ». Parmi ces familles on peut par exemple distinguer celle des réseaux centrés sur la personne (PANs : *Private Area Networks)*, chaque réseau étant constitué des divers équipements électroniques portés par un individu (téléphone portable, agenda numérique personnel, oreillette, etc.). La technologie Bluetooth (IEEE 802.15), par exemple, permet à de tels équipements d'interagir et de se coordonner afin d'offrir certains services avancés à l'utilisateur. Dans ce type de réseau, on attend des divers équipements impliqués qu'ils se découvrent les uns les autres, s'associent, et fonctionnent en bonne intelligence. Les équipements sont donc amenés à constituer un réseau ad hoc, dont les contours demeurent toutefois fluctuants dans la mesure où de nouveaux équipements peuvent se joindre au réseau — ou d'autres le quitter — à tout instant.

Dans les réseaux véhiculaires (VANET : *Vehicular Ad hoc NETworks)*, des équipements embarqués à bord de véhicules coopèrent afin de s'échanger des informations relatives à la circulation routière (*e.g.* notification de ralentissements, présence de zones de travaux ou de bouchons, etc.), voire de coordonner l'évolution des véhicules (*e.g.* circulation en convoi, gestion de priorités, etc.).

Dans les réseaux de capteurs *(sensor networks)*, le réseau est constitué d'un ensemble d'éléments dotés à la fois de d'interfaces de communication sans fils, et de capteurs leur permettant de percevoir certains paramètres de leur environnement (*e.g.* luminosité, température, hygrométrie, etc.). Ce type de réseau est également né dans le domaine militaire, mais des applications civiles commencent à apparaître. En règle générale, les capteurs constituent ensemble un réseau ad hoc afin d'assurer la remontée des informations vers un équipement capable de les exploiter ou de les relayer vers un autre réseau (*e.g.* liaison satellitaire, etc.). Dans un réseau de capteurs l'accent est souvent mis sur la minimisation de la consommation d'énergie (les capteurs étant irrémédiablement « perdus » lorsque leur batterie est épuisée) et sur la résistance aux pannes (*e.g.* contournement d'un capteur défaillant ou compromis). Dans certains cas les capteurs peuvent être mobiles. Ce sera par exemple le cas lorsque ces capteurs sont portés par des individus, que ceux-ci soient humains (*e.g.* collecte de données médicales) ou non (*e.g.* collecte d'informations sur les déplacements des zèbres [97] ou des cétacés [159]).

Problématique inhérente aux réseaux mobiles ad hoc. Le fait que, dans certains types de réseaux, les équipements communicants puissent être mobiles induit bien sûr un niveau de complexité supplémentaire. En effet la topologie même du réseau se trouve continuellement altérée, ce qui oblige à réévaluer continuellement par quel chemin doivent transiter les données. Dans les paragraphes qui suivent on utilisera indifféremment les expressions « réseau mobile ad hoc » ou « MANET » *(Mobile Ad hoc NETwork)* pour désigner un réseau ad hoc constitué d'équipements mobiles.

Les premiers travaux de recherche relatifs aux MANETs ont eu pour objectif principal de définir des méthodes capables d'assurer l'acheminement de messages — ou, plus classiquement, de paquets IP — de bout en bout dans ce type de réseau. Dans cette optique, l'approche communément adoptée consiste à considérer chaque terminal mobile comme un routeur potentiel. Considérons l'exemple illustré dans la figure 3.1, et supposons que dans ce réseau l'hôte A ait besoin de faire parvenir un paquet IP à l'hôte B. La problématique générale du routage ad hoc consiste à identifier au moins un trajet possible permettant de faire cheminer ce paquet IP de proche en proche, depuis l'émetteur A, jusqu'au destinataire B. On notera que, la topologie d'un réseau mobile ad hoc pouvant varier en

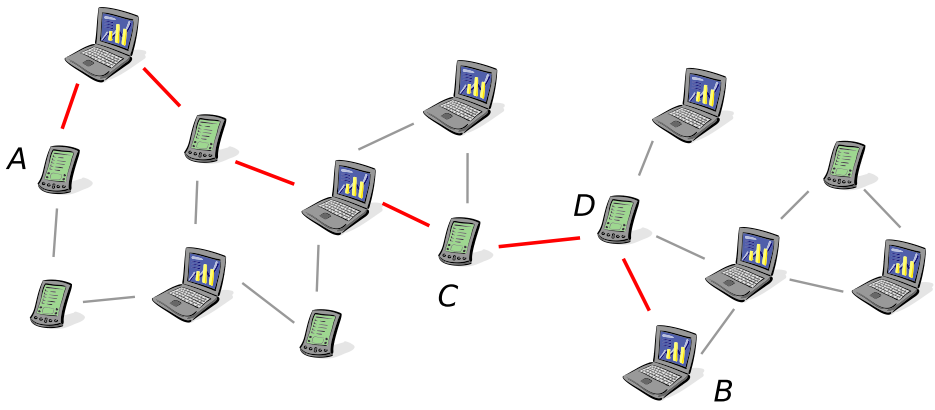


FIG. 3.1 – Exemple de routage entre A et B dans un réseau mobile ad hoc

continu, un chemin exploitable à un moment précis entre deux hôtes A et B ne sera plus nécessairement disponible quelques instants plus tard. Le routage dans un MANET implique donc de remettre sans cesse en question les chemins de routage qui ont pu être identifiés jusqu'alors, ces chemins ayant pu devenir inutilisables ou inefficaces.

On peut également noter qu'un mécanisme de routage similaire à celui qui est mis en œuvre pour acheminer les paquets IP dans Internet n'est envisageable que dans un MANET dense et connexe. La figure 3.2 représente un tel réseau dense, constitué de 600 hôtes mobiles évoluant dans une zone de $1\,km \times 1\,km$. Dans cette figure, le réseau est représenté sous la forme traditionnelle d'un graphe dont les nœuds symbolisent les hôtes (capables de communiquer par liaisons radio directes), et les arêtes symbolisent le fait que deux nœuds sont à portée radio l'un de l'autre, et sont donc susceptibles de communiquer ensemble[3]. Dans le cas présent, on considère un environnement « ouvert » (*i.e.* sans obstacles à la propagation des ondes radio), et l'interface radio équipant chaque hôte est supposée avoir une portée omni-directionnelle de 100 mètres. Deux nœuds du graphe sont donc reliés par une arête si ces nœuds se trouvent à au plus 100 mètres l'un de l'autre.

Dans la figure 3.2 on peut constater que la distribution des nœuds dans l'espace est telle qu'aucun d'entre eux ne se trouve totalement isolé, et donc incapable de communiquer avec quelque voisin que ce soit. En outre le graphe représentant le réseau est connexe : il est donc possible de trouver, pour chaque paire de nœuds du réseau, au moins un chemin permettant de relier ces deux nœuds. Les nœuds du réseau modélisant en fait des équipements mobiles, la topologie même du réseau évolue au fil du temps, des arêtes apparaissant et disparaissant entre les nœuds alors que ceux-ci se déplacent dans la zone considérée.

Les travaux relatifs au routage dans les réseaux mobiles ad hoc ont initialement visé à proposer des solutions pour assurer le routage dans des réseaux présentant les propriétés qui viennent d'être énoncées. Le paragraphe suivant dresse un panorama des travaux réalisés dans ce domaine. Par la suite, l'activité de recherche a été étendue de manière à intégrer des réseaux dans lesquels ces propriétés de densité et de connexité favorables ne sont pas satisfaites. Le paragraphe 3.2.2 décrit la problématique propre à ce type de réseaux, et présente les principales approches proposées pour y supporter les

[3]En toute rigueur, on pourrait utiliser des arêtes orientées pour signifier le fait que les transmissions radio entre deux nœuds ne sont pas toujours possibles dans les deux sens.

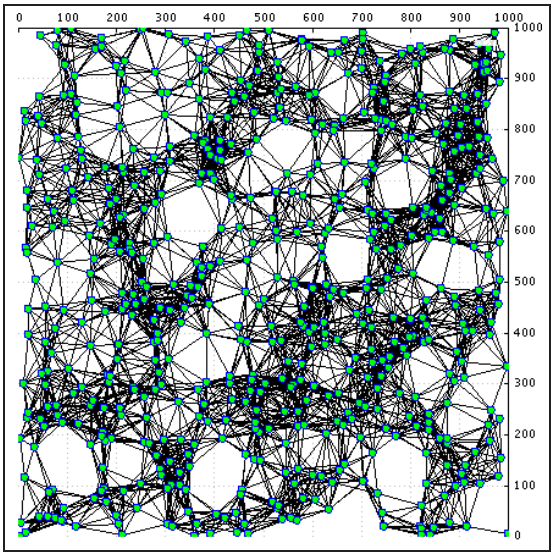


FIG. 3.2 – Exemple de réseau ad hoc dense

communications.

3.2.1 Protocoles de routage pour réseaux mobiles ad hoc

De très nombreux protocoles de routage dynamique pour réseaux mobiles ad hoc ont vu le jour ces dernières années, ces protocoles ayant tous pour vocation de tenir compte de la dynamicité des réseaux dans lesquels ils sont censés être mis en œuvre. Le groupe MANET *(Mobile Ad hoc NETworks)* de l'IETF[4] a entrepris de recenser et évaluer ces divers protocoles, et certains d'entre eux (*e.g.* OLSRv2 et DYMO) ont récemment été sélectionnés en vue d'une standardisation par l'IETF. Ces protocoles peuvent être classés en quelques grandes « familles », énumérées et discutées brièvement ci-dessous. Des études comparatives de ces divers protocoles sont par ailleurs disponibles, notamment dans [155] et dans [28].

Protocoles de routage pro-actifs. Dans le routage pro-actif *(proactive routing* ou *table-driven routing)*, chaque hôte s'efforce de maintenir constamment à jour sa propre table de routage, de sorte que lorsqu'un paquet IP doit être émis (ou routé) par cet hôte la route que ce paquet doit suivre soit d'ores et déjà connue, et donc immédiatement exploitable. L'approche communément adoptée pour réaliser ce type de routage consiste à faire en sorte que chaque hôte diffuse périodiquement sa propre table de routage, et la mette par ailleurs à jour en fonction d'informations similaires reçues de tous ses voisins. Les avantages majeurs présentés par les protocoles de routage pro-actifs sont les bonnes performances qu'ils permettent d'obtenir lorsqu'un paquet IP doit suivre une route dans laquelle tous les hôtes-relais sont effectivement disponibles, et disposent chacun d'une table de routage à jour. Les inconvénients majeurs sont le surcoût important occasionné par le trafic de contrôle nécessaire au maintien à jour des informations de routage, et des temps de réaction souvent assez longs lorsque des changements

[4] *IETF: Internet Engineering Task Force.*

dans la topologie du réseau obligent l'ensemble des hôtes à corriger leurs tables de routage en conséquence. Le groupe MANET de l'IETF a retenu le protocole OLSRv2 *(Optimized Link-State Routing protocol* [30, 147]) comme protocole pro-actif de référence en vue d'une standardisation.

Protocoles de routage réactifs. Dans le routage réactif *(reactive routing* ou *on-demand routing)*, les tables de routage ne sont mises à jour que lorsque le besoin s'en fait sentir. L'approche générale est la suivante : lorsqu'un hôte devant router un paquet IP vers une certaine direction constate que sa table de routage ne contient aucune indication pour atteindre cette destination, il diffuse dans l'ensemble du réseau un message de contrôle *(Route Request)* invitant tous les hôtes du réseau à mettre à jour leurs tables de routage vis-à-vis de la destination visée. En règle générale, cette mise à jour s'effectue grâce à un second message de contrôle initié par l'hôte destinataire lui-même, ce message remontant (en *source-routing*) vers l'hôte qui cherche à l'atteindre. Alternativement, une diffusion peut être utilisée pour inciter tous les hôtes à mettre à jour dans leur table de routage les routes menant vers ce destinataire précis. L'avantage majeur des protocoles reposant sur une approche réactive est que le surcoût occasionné par la mise à jour des informations de routage est directement proportionné, au cas par cas, aux flux de données circulant dans le réseau. Les principaux inconvénients sont que ces protocoles nécessitent un temps de latence conséquent avant qu'une route vers une certaine destination puisse être exploitée pour la première fois (ou encore en cas de changements de topologie fréquents, obligeant à une réactualisation fréquente des routes dans le réseau). Le protocole DYMO *(DYnamic Manet On-demand Routing* [27]) est en passe d'être standardisé par l'IETF en tant que protocole réactif de référence.

Protocoles de routage hybrides. Les protocoles de routage dits « hybrides » (*e.g.* HSLS, TORA, ZRP) sont des protocoles qui combinent les approches pro-actives et réactives afin de bénéficier du meilleur de ces deux approches. Dans certains cas, les protocoles hybrides s'appuient sur une approche pro-active afin de créer des routes principales couvrant à peu près toute la superficie du réseau. Ces routes passent par un certain nombre d'hôtes dits « actifs », à partir desquels le routage vers des hôtes passifs avoisinants peut être effectué de manière réactive. On réalise ainsi un découpage du réseau en un certain nombre de « grappes » *(clusters)* entre lesquelles le routage est assuré via l'approche proactive, et au sein desquelles le routage réactif permet d'atteindre les hôtes en fonction des besoins. On notera que la démarche inverse est également possible, le routage proactif étant réalisé au sein de chaque grappe, et le routage réactif entre les différentes grappes du réseau. Le découpage du réseau en « grappes » peut être réalisé, selon les cas, selon des critères de localisation géographique, ou sur la base d'un découpage purement logique du réseau. Dans certains cas les « grappes » sont elles-mêmes définies de manière à constituer une structure hiérarchisée, la position d'un hôte spécifique dans cette hiérarchie déterminant alors la méthode de routage utilisée pour atteindre cet hôte *(hierarchical routing*, e.g. HSR, ATR).

Protocoles de routage géographique. Dans ce type de protocoles (*e.g.* SiFT, DREAM, ZHLS, LAR), la position des hôtes doit être connue, de manière relative ou absolue. On peut alors tenir compte des distances physiques séparant les hôtes, ainsi que de leur distribution relative dans l'espace, afin d'adapter les méthodes de routage en conséquence. Par exemple, lorsqu'un message doit être acheminé sur une grande distance à travers le réseau, il n'est pas nécessaire que tous les hôtes servant de relais pour ce message disposent dans leur table de routage d'une entrée spécifique pour le destinataire du message. Pour les relais situés à une bonne distance du destinataire, il suffit en effet

de router le message dans la direction générale dans laquelle le destinataire est censé se trouver. Des relais plus proches du destinataire, en revanche, devront être plus précis dans leur activité de routage afin que le message soit peu à peu guidé vers sa destination effective.

Autres catégories de protocoles. Au delà des grandes familles de protocoles qui viennent d'être évoquées (et qui ne sont d'ailleurs pas totalement exclusives les unes vis-à-vis des autres), on peut également évoquer celle des protocoles dans lesquels le processus de routage est réalisé en s'efforçant de minimiser la consommation énergétique sur les hôtes impliqués, ou bien encore de distribuer au mieux cette consommation sur l'ensemble des hôtes du réseau *(power-aware routing protocols,* e.g. EADSR, DSRPA).

On peut également noter que, bien que la plupart des protocoles évoqués ci-dessus ont pour finalité d'assurer le routage d'un message ou paquet IP vers un destinataire précis *(unicast routing)*, il existe également des protocoles spécifiques destinés à assurer soit la diffusion globale *(broadcast,* e.g. DSM-RB), soit la diffusion sélective *(multicast,* e.g. MZR, ODMRP) de messages dans le réseau. Dans chacun de ces cas, on peut encore distinguer entre des protocoles reposant sur des approches pro-actives, réactives, hybrides, etc. pour gérer l'acheminement des messages au sein du réseau.

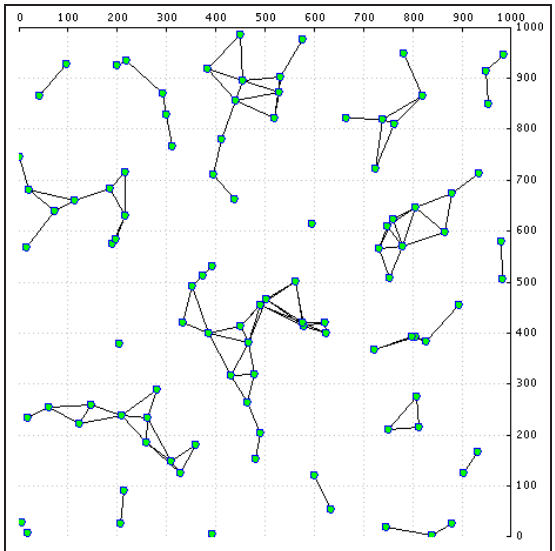
3.2.2 Problématique spécifique aux réseaux mobiles ad hoc discontinus

Les propositions visant à assurer le routage dynamique de paquets IP au sein d'un réseau mobile ad hoc reposent toutes sur l'hypothèse selon laquelle la densité et la distribution spatiale des terminaux sont favorables, et permettent effectivement aux protocoles de routage dynamique de jouer leur rôle. En effet, lorsqu'un paquet IP est émis dans le réseau, un chemin de routage doit impérativement être disponible entre l'émetteur et le destinataire du paquet pour que cette transmission puisse avoir lieu. Considérons de nouveau l'exemple de la figure 3.1, et supposons qu'à l'instant où l'hôte A décide d'adresser un paquet IP à l'hôte B, l'un des hôtes C ou D ne soit plus disponible pour assurer la connectivité entre la partie gauche et la partie droite du réseau. Aucun cheminement n'est alors possible entre A et B, et en l'absence de toute possibilité d'acheminement du paquet émis par A à l'intention de B, ce paquet IP est considéré comme étant irrémédiablement perdu.

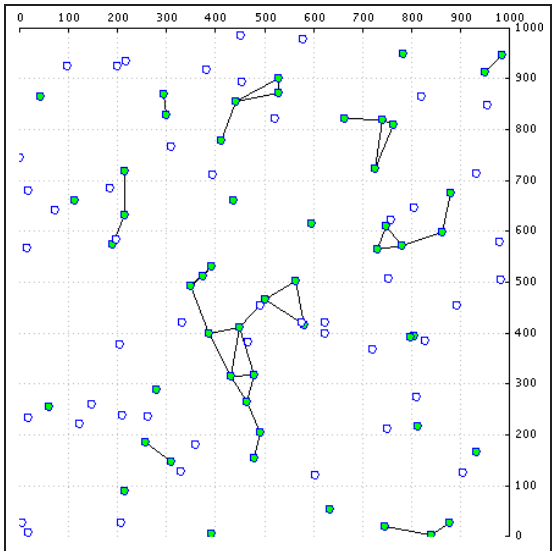
Les divers protocoles de routage dynamique évoqués au paragraphe 3.2.1 ne sont donc utilisables que dans des réseaux ad hoc suffisamment denses et connexes. En revanche ils n'apportent pas de solution satisfaisante pour assurer l'acheminement de données dans des réseaux ad hoc fortement clairsemés et partitionnés, c'est-à-dire des réseaux dans lesquels les terminaux mobiles peuvent se trouver isolés les uns des autres de manière épisodique, voire de manière chronique [11].

La figure 3.3-a présente un réseau ad hoc constitué de 50 hôtes mobiles, répartis sur une superficie de $1\,km \times 1\,km$. Dans cet exemple on peut constater que le graphe représentant le réseau n'est pas connexe. La densité des nœuds est insuffisante pour qu'on puisse trouver un chemin reliant toute paire de nœuds du réseau. On voit donc apparaître des sous-graphes connexes — on parlera « d'îlots » — au sein desquels les communications sont possibles (en utilisant éventuellement du routage dynamique), mais entre lesquels aucune communication n'est a priori possible.

La figure 3.3-b présente un autre réseau ad hoc, très similaire au précédent, mais dans lequel la connexité du graphe représentant le réseau est encore réduite du fait que certains nœuds sont supposés être éteints ou mis en veille, et donc inaptes à communiquer avec leur entourage. La *volatilité* des nœuds (*i.e.* le fait qu'ils puissent être tantôt actifs, tantôt inactifs du point de vue du réseau) est donc



(a) Exemple de MANET clairsemé, tous les noeuds étant supposés actifs et aptes à communiquer



(b) Exemple de MANET clairsemé, les noeuds blancs étant supposés éteints, et donc inaptes à communiquer

FIG. 3.3 – Exemples de MANETs discontinus

un paramètre qui peut influer de manière non négligeable sur les possibilités de communication au sein d'un réseau mobile ad hoc.

Principe de la communication « opportuniste tolérant les délais ». De nombreux travaux ont été lancés ces dernières années en vue de s'accommoder de l'absence de connectivité de bout-en-bout dans les réseaux mobiles ad hoc discontinus. L'idée générale est de remplacer ou de compléter les mécanismes de routage dynamique conçus pour les réseaux denses par des mécanismes permettant de tolérer les ruptures de connectivité occasionnelles ou chroniques. Pour ce faire, l'approche communément adoptée consiste à combiner les principes de la communication dite « tolérant les délais » *(DTN : Delay-Tolerant Networking)* et de la communication dite « opportuniste », et à appliquer ces principes combinés dans les réseaux mobiles ad hoc.

Le concept de réseau tolérant les délais a initialement été introduit dans le cadre d'études menées par le groupe IPNRG *(Inter-Planetary Networking Research Group)* de l'IRTF[5], ces études visant à concevoir un « Internet inter-planétaire ». Dans ce type de réseau, les principaux défis qu'il faut relever résultent des délais de transmission très élevés et de la connectivité fluctuante qu'on est susceptible d'observer. En effet, lors de transmissions réalisées par exemple entre deux planètes du système solaire, les délais de propagation des signaux radio peuvent être de l'ordre de plusieurs minutes, voire plusieurs heures. La plupart des protocoles de communication utilisés actuellement dans Internet ne sont pas adaptés à de tels temps de transmission. D'autre part, compte tenu des positions relatives des planètes, de leur rotation, des phénomènes d'occultation, de l'utilisation éventuelle de satellites (naturels ou artificiels) en guise de relais, la connectivité entre un émetteur et un récepteur peut être

[5]L'IRTF *(Internet Research Task Force)* est un organe dépendant de l'IAB *(Internet Architecture Board)* dont le rôle est de promouvoir les études portant sur l'évolution à long terme d'Internet. L'IETF *(Internet Engineering Task Force)* se focalise par contraste sur la gestion d'Internet dans sa forme actuelle, et son évolution à plus court terme.

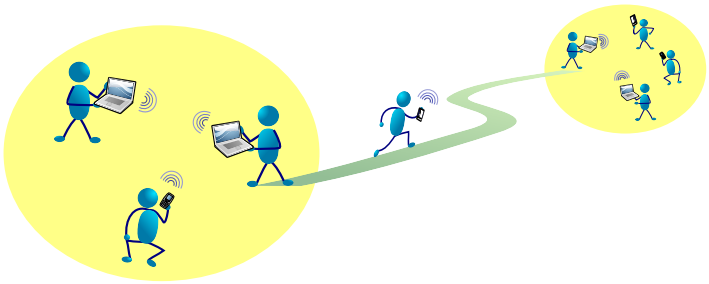


FIG. 3.4 – Détail illustrant le transport de messages par un terminal mobile se déplaçant entre deux îlots constitué chacun de quelques terminaux

fluctuante, l'émetteur étant alors contraint d'exploiter de manière opportuniste des « fenêtres temporelles » favorables pour adresser des messages au récepteur. Les travaux du groupe IPNRG ont permis de poser les bases d'une nouvelle architecture de réseau capable de tolérer de telles contraintes [46]. Ces travaux ont ensuite été repris, et se poursuivent depuis lors, sous les auspices d'un nouveau groupe de travail mandaté par l'IRTF. Ce groupe, baptisé DTNRG[6] *(pour Delay-Tolerant Networking Research Group)*, s'est donné pour mission de promouvoir les activités de recherche menées dans le domaine général des réseaux tolérant les délais. Il est en effet aujourd'hui admis que les problèmes identifiés par l'IPNRG dans le cadre de son étude sur les réseaux inter-planétaires[7] peuvent également être rencontrés dans des réseaux de moindre envergure — et notamment les réseaux mobiles ad hoc discontinus — dans lesquels une connectivité permanente de bout-en-bout à travers le réseau ne peut être garantie.

De manière générale, l'approche adoptée consiste à doter tout ou partie des terminaux de la capacité de stocker des messages en transit dans un cache avant de les réémettre au moment opportun. Lorsque les terminaux considérés sont mobiles, cette aptitude à stocker temporairement les messages peut être mise à profit afin de faire transporter physiquement les messages hébergés par un terminal qui se déplace. L'expression *Store, Carry, and Forward* est utilisée pour désigner ce type particulier de traitement des messages impliquant leur stockage, leur transport éventuel par des terminaux en mouvement, et leur renvoi final vers d'autres terminaux relais ou vers les terminaux destinataires.

La figure 3.4 illustre un cas typique dans lequel un terminal mobile transporté par un individu se déplaçant entre deux groupes de personnes (ces groupes n'étant pas à portée radio l'un de l'autre) peut contribuer à transporter des messages d'un groupe à l'autre, assurant ainsi une certaine forme de « connectivité » — non instantanée, et justifiant donc une approche « tolérant les délais » — entre ces deux groupes. Dans les deux exemples présentés dans la figure 3.3, on conçoit aisément que des hôtes mobiles se déplaçant dans la zone considérée peuvent fort bien contribuer à transporter des messages, permettant ainsi à ces messages de passer d'un îlot de connectivité à l'autre.

En appliquant le modèle de la communication tolérant les délais dans les réseaux mobiles ad hoc, on ouvre ainsi d'intéressantes perspectives dans la mesure où cette approche permet de s'affranchir de l'exigence de connectivité de bout-en-bout imposée jusqu'alors par les algorithmes de routage dynamique évoqués dans le paragraphe 3.2.1. Ce constat a motivé depuis quelques années l'ouverture de

[6] `http://www.dtnrg.org`

[7] Les travaux portant spécifiquement sur les réseaux inter-planétaires se poursuivent aujourd'hui dans le cadre du groupe d'intérêt IPNSIG *(Inter-Planetary Networking Special Interest Group*, `http://www.ipnsig.org`).

nouveaux champs d'activité de recherche, dans lesquelles les diverses approches proposées diffèrent cependant assez nettement selon :

- les caractéristiques du type de réseau considéré (*e.g.* envergure du réseau, schémas de mobilité et de contacts entre hôtes mobiles au sein de ce réseau, etc.) ;
- le type de communications réalisées dans ce réseau (*e.g.* communications point-à-point ou multi-points) ;
- la finalité de ces communications (*e.g.* collecte de données dans un réseau de capteurs mobiles, interactions pair-à-pair entre hôtes participant à un réseau ad hoc « pur », communication avec un réseau d'infrastructure par le biais de passerelles faisant le lien entre l'univers sans fils et discontinu et l'univers filaire continu, etc.).

3.2.3 Routage dynamique dans les MANETs discontinus

Une étude comparative de travaux récents réalisés dans le domaine du routage dans les réseaux ad hoc discontinus est disponible dans [171]. Il est intéressant de noter que bon nombre des travaux recensés dans cet article se donnent pour objectif de supporter l'acheminement de messages en mode *unicast*, c'est-à-dire vers une destination unique. Pour ce faire les approches diffèrent cependant selon les hypothèses formulées vis-à-vis des caractéristiques du réseau considéré (*e.g.* topologie, densité et schéma de mobilité des hôtes, etc.).

Idéalement, l'acheminement d'un message entre un hôte source et un hôte destinataire à travers un MANET discontinu devrait pouvoir s'apparenter à l'acheminement des paquets IP dans Internet. Chaque hôte mobile peut être amené à jouer le rôle de relais (ou routeur), et peut donc recevoir un message d'un hôte voisin (on qualifie ici d'hôtes « voisins » des hôtes qui se trouvent à portée radio l'un de l'autre) afin de le relayer vers un autre voisin. L'absence de connectivité de bout en bout à travers le réseau rend toutefois le problème nettement plus compliqué qu'il l'est dans Internet. En effet, un hôte peut recevoir un message de l'un de ses voisins, déposer ce message dans un cache local, se déplacer pendant un certain temps dans le réseau, et rencontrer un autre voisin auquel il peut à son tour confier le message. Le modèle de « transfert par délégation » *(custody transfer)* défini dans [48] repose sur ce principe général : un hôte ayant accepté de stocker (et éventuellement de transporter) un message doit impérativement transférer ce message à un autre hôte avant de pouvoir lui-même abandonner toute responsabilité vis-à-vis de ce même message. Ce modèle présente l'avantage d'être particulièrement économique en termes de ressources, puisqu'un message est sous la responsabilité d'un hôte — et d'un seul — à tout instant au cours de son acheminement. En revanche, en confiant la responsabilité d'un message à un seul hôte à tout instant, on court le risque de perdre ce message si son porteur est lui-même détruit ou connaît une défaillance. En outre, déterminer une route exploitable pour acheminer un message entre un hôte source et un autre destinataire nécessite de pouvoir identifier une succession d'hôtes intermédiaires susceptibles d'accueillir temporairement ce message. Dans le cas d'un réseau mobile ad hoc discontinu, la détermination d'une route exploitable se ramène donc à un problème d'identification d'un cheminement possible dans l'espace *et* dans le temps (on parlera de « chemin spatio-temporel ») pour que le message considéré puisse effectivement être relayé de proche en proche jusqu'à sa destination.

Dans certains types de réseau, l'application du modèle de transfert par délégation est effectivement envisageable. De manière générale il s'agit de réseaux dans lesquels certains hôtes mobiles se déplacent de façon planifiée (ou tout au moins prévisible), voire de manière contrôlable. Des communications peuvent ainsi être réalisées sur de grandes distances, en utilisant des bus comme des

transporteurs capables d'acheminer des messages, soit vers un autre quartier de la ville, soit vers la ville ou le village voisin [170]. Des projets effectifs reposant sur ce principe ont d'ores et déjà été lancés dans certains pays émergents afin de supporter, par exemple, les communications entre plusieurs villages [86]. Lorsqu'un bus passe dans un village, des messages que les villageois ont pu enregistrer sur un ordinateur mis à leur disposition sont automatiquement transférés (en général via une liaison Wi-Fi ou Bluetooth) vers un autre appareil placé à bord du bus. Ces messages seront de même relayés vers les ordinateurs destinataires lorsque le bus passera dans les villages correspondants.

Le modèle du transfert par délégation peut également être utilisé dans des environnements dans lesquels les déplacements des hôtes mobiles peuvent être contrôlés explicitement. Des travaux tels que [172, 125] étudient ainsi les stratégies à mettre en œuvre lorsque certains au moins des hôtes mobiles sont des engins robotisés capables de se déplacer afin d'aider à l'acheminement des messages.

Dans un réseau dont les schémas de mobilité des hôtes ne sont pas connus a priori, il est malaisé de déterminer un cheminement spatio-temporel précis pour chaque message devant être transporté à travers le réseau. Certains travaux reposent donc sur un modèle dit « de dissémination épidémique » afin de maximiser les chances qu'un message atteigne son destinataire[8]. Dans ce modèle, chaque message est assimilé à un « virus », ou une « maladie contagieuse », qui va peu à peu « infecter » tous les hôtes du réseau [165, 136]. Initialement, seul l'hôte émetteur est infecté, ce qui revient à dire que lui seul est dépositaire du message. Lorsque cet hôte va entrer en contact avec d'autres hôtes non infectés, chacun d'entre eux va à son tour être infecté, c'est-à-dire recevoir une copie du message et déposer cette copie dans un cache local. Tout hôte infecté devient donc un vecteur de l'épidémie (*i.e.* un transporteur d'une copie du message initial), et va contribuer à la propagation de cette épidémie au gré de ses déplacements, et de ses rencontres avec d'autres hôtes mobiles. À terme, le destinaire du message va lui-même se trouver infecté par l'épidémie, c'est-à-dire recevoir le message qui lui était destiné.

On peut noter que cette approche est évidemment fort coûteuse dans la mesure où elle mobilise énormément de ressources dans le réseau. Chaque hôte du réseau est en effet susceptible d'être atteint par l'épidémie, c'est-à-dire d'avoir à recevoir, stocker, puis relayer à son tour une copie de chaque message circulant dans le réseau. De nombreuses variantes à ce modèle ont donc été proposées en vue de réduire son coût, ces variantes impliquant notamment tout ou partie des points suivants :

- **« Guérison » du réseau *(network healing)* :** Lorsqu'un message atteint son destinataire, il se peut que d'autres hôtes du réseau n'aient pas encore été atteints par l'épidémie servant à propager ce message. En outre, des hôtes ayant été atteints par l'épidémie ne savent pas que le message a atteint son destinataire, et vont donc continuer à essayer de « contaminer » d'autres hôtes autour d'eux. Pour éviter que l'épidémie continue à se propager inutilement, le destinataire peut à son tour injecter un « antidote » dans le réseau. Cet antidote prend la forme d'un message de contrôle, qui va également se propager dans le réseau selon le modèle épidémique, mais qui va avoir pour effet de signifier aux hôtes qui vont le recevoir qu'il ne leur est plus nécessaire de contribuer à propager le message initial. On peut noter que lorsque l'antidote atteint l'émetteur du message initial, ceci a pour effet d'indiquer à cet émetteur que ce message a bien atteint le destinataire visé. L'antidote joue donc aussi le rôle d'accusé de réception pour l'émetteur.
- **Limitation du nombre de porteurs d'un message :** De la même manière qu'une épidémie ne touche pas tous les individus de la même façon, on peut faire en sorte que certains hôtes

[8] Certains travaux (*e.g.* [39]) parlent aussi de « rumeur » ou de « commérage » *(gossiping)* pour désigner des modèles de dissémination de l'information finalement très semblables au modèle épidémique.

mobiles soient plus enclins que d'autres à jouer le rôle de transporteurs pour un certain message. Certaines des approches proposées reposent sur des méthodes probabilistes [126], d'autres sur des heuristiques destinées à contrôler la création des copies [160, 92], afin de limiter le nombre d'hôtes porteurs d'un message dans le réseau.

– **Maîtrise de la portée spatio-temporelle de la propagation épidémique :** Dans Internet il est possible de limiter la « durée de vie » d'un paquet IP en bornant explicitement le nombre de sauts que ce paquet est autorisé à réaliser en passant de routeur en routeur. Le même mécanisme peut être utilisé afin de limiter grossièrement la portée d'une propagation épidémique dans un réseau mobile ad hoc discontinu. Cependant, les hôtes étant a priori capables de se déplacer dans le réseau, et donc de parcourir éventuellement de longues distances avant de relayer un message, cette approche ne doit pas être comprise comme permettant de limiter la portée géographique d'un message. Dans un réseau dans lequel les hôtes mobiles sont capables d'estimer leur position géographique, on peut en revanche limiter la portée spatiale de la propagation d'un message en spécifiant dans l'en-tête de ce message qu'il ne doit se propager que dans une zone bien identifiée. On peut de même borner la durée pendant laquelle un message est susceptible de se propager dans le réseau en spécifiant une durée de vie explicite pour ce message. Passé la date de péremption de ce message, tous les hôtes transportant des copies de ce message les feront disparaître de leur cache, ce qui aura pour effet de mettre un terme à la propagation de ce message dans le réseau.

Le modèle de transfert par délégation et le modèle de dissémination épidémique peuvent être vus comme deux approches radicalement opposées. Avec le modèle de transfert par délégation on cherche à faire cheminer chaque message le long d'un chemin spatio-temporel — et un seul — au sein du réseau. Pour ce faire on doit être capable de déterminer quel chemin le message doit suivre, et l'on est particulièrement vulnérable à tout événement imprévu survenant dans le réseau (*e.g.* disparition d'un hôte censé participer à l'acheminement du message). Avec l'approche épidémique on tente au contraire d'exploiter plusieurs chemins spatio-temporels simultanément, voire *tous* les chemins possibles si l'on ne limite en rien la propagation de l'épidémie. En procédant ainsi on augmente les chances que le message arrive rapidement à destination, mais l'acheminement d'un seul message mobilise une très grande quantité de ressources dans le réseau.

Certains travaux récents se situent entre ces deux approches extrêmes. En règle générale ces travaux visent à minimiser le nombre de transporteurs pour chaque message en *sélectionnant* les terminaux mobiles les plus aptes à contribuer de manière efficace à l'acheminement de ce message vers sa destination. Pour ce faire des heuristiques doivent être définies afin d'évaluer dans quelle mesure un terminal peut être « utile » à l'acheminement d'un message.

Ces approches dites *utility-based* peuvent tenir compte de paramètres divers, tels que l'état de la batterie sur un hôte mobile, la quantité d'espace encore disponible dans le cache de cet hôte, etc. Cependant, la plupart des travaux portant sur des approches *utility-based* supposent, soit une connaissance a priori des schémas de mobilité des terminaux dans le réseau, soit une aptitude à prédire la mobilité de ces terminaux en fonction d'un historique des schémas de mobilité observés dans le passé [126]. Pour cette raison, les algorithmes proposés sont en général spécifiquement conçus afin d'exploiter des caractéristiques de mobilité bien précises (*e.g.* schémas de mobilité sociale ou professionnelle dans le cas de terminaux transportés par des êtres humains, schémas de mobilité routière dans le cas de terminaux embarqués dans des véhicules, etc.).

Tous les travaux qui viennent d'être évoqués visent à supporter le trafic *unicast* à travers le réseau, c'est-à-dire l'acheminement de chaque message vers un destinataire précis. Certains travaux portent

toutefois sur d'autres types de trafic, et notamment la diffusion totale *(broadcast)* ou la diffusion sélective *(multicast)* dans le réseau [146, 92, 156]. En fait le modèle épidémique pur tel que défini dans [165] se prête parfaitement à un exercice de diffusion totale, puisqu'il consiste à faire en sorte que tout hôte du réseau soit « infecté » par « l'épidémie » que constitue le message à transmettre.

3.2.4 Communication « basée contenus » dans les MANETs discontinus

Les diverses méthodes permettant d'acheminer des messages en mode *unicast*, *broadcast*, ou *multicast* ont toutes pour caractéristique commune que la communication s'effectue en fonction de la destination visée *(destination-driven communication)*, cette destination étant en général spécifiée sous la forme d'une adresse (*e.g.* adresse IP, nom DNS, adresse Email, etc.). Ceci implique notamment que l'émetteur d'un message soit en mesure de déterminer quel est — ou quels sont — le(s) destinataire(s) de ce message avant de pouvoir l'injecter dans le réseau.

Les applications dédiées au partage d'information, à la distribution d'informations thématiques ou événementielles, à l'annonce et à la découverte de services, nécessitent pourtant un modèle de communication dans lequel l'information devrait pouvoir parvenir à tout récepteur intéressé, plutôt qu'à des destinataires identifiés. La communication dite « basée contenus » *(content-based communication*[9]*)* est un style de communication qui répond parfaitement aux besoins de telles applications. Avec ce type de communication, l'information circule dans le réseau en fonction de l'intérêt que les nœuds du réseau y portent, plutôt que vers une destination bien établie [26]. Les nœuds du réseau affichent leur intérêt pour un certain type d'information, en *souscrivant* afin de recevoir spécifiquement les messages transportant ce type d'information. Les producteurs d'information se contentent d'injecter (ou *publier*) des messages dans le réseau, sans spécifier une destination spécifique pour ces messages. Le système de communication sous-jacent se charge de délivrer chaque message aux nœuds qui ont souscrit afin de recevoir le type d'information qu'il contient.

La communication « basée contenus » permet un découplage clair entre producteurs et récepteurs d'information. Elle est donc tout particulièrement adaptée à une utilisation dans un réseau mobile ad hoc, dans lequel elle peut faire office de paradigme de communication pour toutes les applications ayant pour finalité le partage d'informations, la distribution d'informations thématiques ou événementielles, l'annonce et la découverte de services, etc.

De très nombreux articles concernant la communication basée contenus ont déjà été publiés, mais ces articles décrivent pour la plupart des méthodes applicables, soit dans des réseaux câblés et stables, soit dans des MANETs connexes [32, 132, 144]. Ils proposent en général de construire et de maintenir une structure de routage dirigé par le contenu, cette structure permettant alors de router les messages depuis leurs producteurs vers tous les récepteurs intéressés.

Une exception notable vis-à-vis de cette approche est le protocole décrit dans [10]. Ce protocole ne cherche pas à bâtir une quelconque structure afin de supporter les décisions de routage. Il s'appuie au contraire sur une utilisation systématique des transmissions en diffusion, laissant aux hôtes qui reçoivent un message diffusé par l'un de leurs voisins la décision d'accepter ou non ce message en vue de le rediffuser à leur tour. Cette décision doit être prise en fonction d'une estimation de la distance séparant l'hôte considéré des récepteurs intéressés. Ce protocole ne peut donc fonctionner que dans un environnement connexe, dans lequel un chemin existe toujours entre émetteur et récepteurs. En conséquence il ne pourrait fonctionner de façon satisfaisante dans un MANET discontinu.

[9] L'expression *interest-based communication* est aussi utilisée dans la littérature.

Le support de la communication basée contenus dans les MANETs discontinus est l'objectif spécifique affiché dans [31]. Cet article décrit une approche dans laquelle une structure de routage multi-sauts en fonction du contenu (limitée à un certain « horizon ») est bâtie autour de chaque hôte mobile. Une fonction dite « d'utilité » *(utility function)* est utilisée pour sélectionner les meilleurs relais pour chaque type de message, et des relais mobiles peuvent aider à disséminer des messages entre des fragments non connectés du réseau.

Les paragraphes suivants présentent un protocole que nous avons développé, et qui reprend certaines des caractéristiques des deux protocoles évoqués ci-dessus. Plutôt que d'essayer de construire une structure de routage, il utilise des diffusions périodiques (également limitées à un certain « horizon » autour de chaque émetteur) permettant à chaque hôte mobile de se faire connaître de ses voisins, de leur faire savoir à quel type de documents ils s'intéresse (via la description de son « profil d'intérêt »), et enfin de leur proposer des documents dont il dispose et qui correspondent à leurs propres profils d'intérêt.

3.3 Développement d'un protocole de communication « basée contenus » pour MANETs discontinus

3.3.1 Vue d'ensemble

Le protocole que nous avons défini s'inscrit dans la lignée des protocoles reposant sur le principe général du « papotage » (*gossiping*[10]), dans lequel des hôtes mobiles profitent de contacts occasionnels pour échanger des informations diverses, en fonction de leurs centres d'intérêts respectifs. Ce type de protocole a d'ailleurs été formalisé de façon assez abstraite avec l'algorithme *Autonomous Gossipping* (A/G) [39]. Dans la définition de cet algorithme, le schéma de dissémination de l'information est assimilé à un processus d'épidémie, chaque hôte étant plus ou moins susceptible d'être « infecté » par tel ou tel type d'information. Une différence fondamentale entre notre protocole et l'algorithme A/G est que ce dernier repose exclusivement sur des transmissions à un saut (et donc des échanges s'effectuant exclusivement entre voisins directs), alors que notre protocole peut exploiter des transmissions multi-sauts au sein d'un fragment connexe (îlot) du réseau mobile ad hoc.

Pour ce faire, notre protocole se décompose en deux couches. La couche supérieure supporte la dissémination basée contenus, et dans un mode tolérant les délais, d'éléments d'information structurés que nous désignons par le terme générique de « documents ». Cette couche assure le stockage de documents dans le cache local d'un hôte mobile, de telle sorte que cet hôte fasse office de transporteur pour les documents maintenus dans son cache lorsqu'il se déplace dans le réseau. Elle définit également les modalités d'interaction entre des hôtes voisins dans le réseau, afin que ces hôtes puissent échanger des documents en fonction de leur profils d'intérêt respectifs. Des hôtes voisins sont des hôtes qui résident — a priori temporairement — dans le même fragment connecté du réseau (ou « îlot »). L'interaction entre ces hôtes nécessite qu'ils soient en mesure de communiquer, en utilisant pour ce faire des transmissions à un saut ou à plusieurs sauts. La couche inférieure de notre protocole fournit les mécanismes permettant d'assurer le relais multi-sauts au sein d'un îlot.

[10] *Gossiping* peut également se traduire par « bavardage » ou « commérage ».

3.3.2 Dissémination de documents basée contenu, et tolérant les délais (couche supérieure)

Utilisation frugale du médium radio. Notre protocole a été conçu de manière à être aussi frugal que possible vis-à-vis des ressources qu'il consomme, et en particulier vis-à-vis de l'utilisation du médium radio. Dans cette optique, il a été conçu de manière à minimiser systématiquement la quantité de données transmises par chaque terminal sur le canal radio, tout en évitant les retransmissions inutiles de ces données. Il s'appuie notamment, chaque fois que c'est possible, sur des tranmissions en diffusion *(broadcast)* plutôt que sur des transmissions en mode *unicast*. Ainsi, lorsqu'un hôte doit faire parvenir un message à tous ses voisins directs, cette opération est réalisée en une seule diffusion de ce message, plutôt qu'en adressant une copie de ce message successivement à chacun de ses voisins. De ce point de vue notre approche s'apparente à celle prônée dans [10] et dans [167], par exemple. Cette approche peut d'ailleurs paraître assez naturelle dans un environnement dans lequel les transmissions se font via un médium radio, puisqu'avec ce type de médium toute transmission se traduit bien par une diffusion dans l'éther. Pourtant, dans de nombreux travaux récents on ressent de la part des auteurs une certaine réticence à exploiter des transmissions en diffusion. Cette réticence est motivée par le fait qu'avec une technologie de transmission sans fils telle que Wi-Fi, par exemple, l'émission d'une trame en mode *broadcast* (*i.e.* vers tous les hôtes voisins) est effectivement nettement moins fiable que son émission en mode *unicast* (*i.e.* vers un hôte voisin explicitement désigné). Cette différence est due au fait que, dans le protocole MAC 802.11, l'émission d'une trame en mode *unicast* implique l'utilisation d'un mécanisme d'acquittement, avec réémission en cas de défaut d'acquittement. L'émission d'une trame en mode *broadcast*, en revanche, n'implique aucun mécanisme de ce type. Avec la technologie Wi-Fi, l'émission de trames en mode *broadcast* est donc nettement plus hasardeuse que l'émission de trames en mode *unicast*. En cas d'interférences pendant l'émission d'une trame, celle-ci peut ne pas être reçue sans que ni l'émetteur, ni le(s) destinataire(s) en soient informés. En contrepartie, l'émission d'une trame 802.11 en mode *broadcast* est nettement moins coûteuse qu'une émission en mode *unicast*, dans la mesure où elle n'implique ni trames d'acquittement, ni réémissions éventuelles. Il en résulte une consommation moindre des ressources mises en jeu lors de la transmission (i.e. occupation moindre du médium radio, et consommation moindre d'énergie au niveau des hôtes impliqués).

En concevant notre protocole nous avons choisi de privilégier une consommation réduite des ressources, au détriment de la fiabilité des transmissions. Nous avons donc choisi d'utiliser des transmissions en mode *broadcast* chaque fois que c'est possible, tout en assurant que le protocole permette aux hôtes de résister aux échecs de transmissions. De manière générale les interactions entre hôtes voisins reposent sur un schéma d'échanges opportunistes plutôt que sur un schéma transactionnel strict. Ainsi, lorsqu'un hôte diffuse par exemple une annonce à l'intention de ses voisins (cet aspect du protocole sera détaillé plus loin), certains d'entre eux peuvent ne pas recevoir cette annonce, sans que ce phénomène compromette en aucune façon le bon fonctionnement de l'hôte émetteur ou des récepteurs potentiels.

Entités manipulées par le protocole. Les principales entités manipulées par la couche supérieure de notre procotole sont décrites ci-dessous.

Documents, descripteurs, et identifiants — Un document est une unité d'information structurée, qu'un hôte peut injecter dans le réseau afin qu'il s'y propage, et soit à terme réceptionné par tout hôte affichant un intérêt pour le type particulier d'information contenue dans ce document. Un document est en fait constitué de deux parties : son descripteur, et son contenu. Le descripteur peut être

```
<descriptor
  id="254d3g64z36cd"
  service="filesharing"
  type="application/pdf"
  date="Fri Oct 12 09:52:11 CEST 2007"
  deadline="Sat Oct 13 14:00:00 CEST 2007"
  from="Fred"
  keywords="mobile,ad hoc,delay-tolerant,opportunistic,gossip-based"
/>
```

FIG. 3.5 – Exemple de descripteur de document

vu comme une collection d'attributs fournissant n'importe quel type d'information à propos du document, comme par exemple son identifiant, son origine, sa date de production, le type et la nature de son contenu, une liste de mots-clés le caractérisant, etc. Le seul attribut dont la présence est absolument requise dans le descripteur d'un document est son identifiant. Cet identifiant doit en outre être unique, car il va permettre d'assurer les échanges entre hôtes dans le réseau, tout en leur évitant de stocker ou d'échanger des doublons. De manière générale, nous faisons l'hypothèse que le poids d'un document dépasse très nettement celui de son descripteur, qui est lui-même supérieur à celui de son identifiant. Des ordres de grandeurs typiques sont : *O(10 ko)* pour un document, *O(100 o)* pour un descripteur, et *O(10 o)* pour un identifiant. Ce contraste marqué entre les poids relatifs des entités manipulées dans notre protocole est systématiquement mis à profit afin de limiter les quantités de données échangées entre les hôtes mobiles.

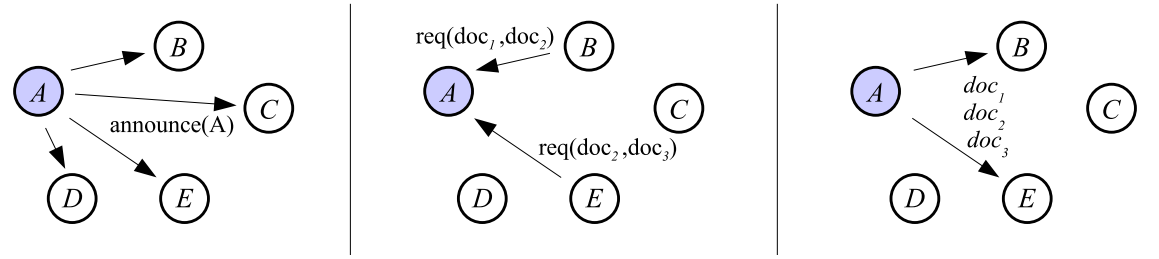
Un exemple de descripteur de documents est présenté dans la figure 3.5. Ce descripteur concerne un document diffusé dans le cadre d'un service de partage de fichiers. Le descripteur spécifie notamment quel est le type du contenu de ce document (un document PDF en l'occurrence), et il contient des mots-clés caractérisant ce document. On peut noter que l'identité de l'émetteur est indiquée (bien que ce ne soit pas une obligation), ainsi que les dates de production et de péremption de ce document. Dans le cas présent, le document n'est pas censé se propager dans le réseau au delà des date et heure indiquées.

Cache — Chaque hôte maintient un cache dans lequel des documents peuvent être stockés. Notre protocole ne fait aucune hypothèse quant à la capacité de stockage de chaque hôte mobile. En revanche il est admis que cette capacité est a priori limitée, et qu'elle peut être différente d'un hôte à l'autre. Plusieurs travaux comparant les mérites respectifs de diverses stratégies de gestion de cache dans les réseaux tolérant les délais ont fait l'objet de publications ces dernières années (*e.g.* [92]). Dans nos propres travaux, nous ne cherchons pas à développer de nouvelles méthodes de gestion de cache. Nous supposons simplement que chaque hôte applique une politique quelconque afin de gérer son propre cache. Par ailleurs nous ne faisons pas l'hypothèse que tous les hôtes d'un réseau appliquent nécessairement la même politique de gestion de cache.

Profils d'intérêt — Le profil d'intérêt d'un hôte caractérise le type de documents qui l'intéressent et donc, implicitement, le type de documents qu'il souhaite recevoir, et pour lesquels il est également prêt à jouer le rôle de transporteur mobile. Pour des raisons pratiques, nous définissons le profil d'un hôte comme un prédicat applicable à des descripteurs de documents. En appliquant ce prédicat au descripteur d'un document, un hôte peut ainsi décider s'il doit réceptionner ce document et le placer

```
<profile
  service="filesharing"
  keywords="sensor|vehicular|opportunistic"
/>
```

FIG. 3.6 – Exemple de spécification du profil d'intérêt d'un hôte mobile



(a) A diffuse une annonce (b) B et E demandent des documents (c) A diffuse les documents demandés

FIG. 3.7 – Illustration d'un cycle de transmission élémentaire autour de l'hôte A

dans son cache. Un exemple de profil d'intérêt basique est reproduit dans la figure 3.6. Ce profil spécifie que l'hôte considéré est intéressé par tout document diffusé dans le cadre du service de partage de fichiers, dès lors que ce document est caractérisé par au moins l'un des trois mots-clés indiqués. Un hôte affichant ce profil d'intérêt se considèrerait donc comme étant intéressé par le document dont le descripteur est présenté dans la figure 3.5.

On peut noter que le profil d'intérêt d'un hôte peut être défini, dans certains cas, directement par le propriétaire ou administrateur de cet équipement. Dans d'autres cas ce sont les applications s'exécutant sur cet équipement qui pourraient être amenées à ajuster d'elles-mêmes le profil de l'hôte, en fonction de leurs propres besoins. En fait de nombreuses stratégies de définition et de gestion du profil d'intérêt d'un hôte mobile sont envisageables. Pour l'heure nous nous contentons de décrire un protocole qui a été développé tout spécialement afin de pouvoir supporter la dissémination de documents dans le réseau, tout en tenant compte des profils des hôtes constituant ce réseau.

Comme indiqué précédemment, notre protocole a été conçu de manière à être particulièrement frugal vis-à-vis des ressources consommées afin d'assurer la dissémination de documents dans le réseau. Nous nous sommes notamment efforcés de minimiser le volume des données transmises sur le médium radio, tout en évitant les retransmissions inutiles de ces données. L'interaction entre hôtes mobiles repose sur un schéma extrêmement simple, illustré dans la figure 3.7. Dans ce schéma, chaque hôte informe périodiquement son voisinage de son propre profil intérêt, tout en proposant à ses voisins de leur fournir les documents qui sont déjà disponibles dans son propre cache. Lorsqu'un hôte découvre ainsi que l'un de ses voisins est en mesure de lui fournir un document qui l'intéresse (c'est-à-dire un document qui correspond à son propre profil d'intérêt, mais qui ne se trouve pas déjà dans son cache local), il peut adresser une requête au voisin concerné afin de se procurer le document en question. Des contacts transitoires entre hôtes mobiles sont ainsi exploités de manière opportuniste afin d'échanger des documents entre ces hôtes, en tenant compte de leurs profils d'intérêt respectifs, et des documents dont ils disposent déjà.

Annonce périodique du catalogue et du profil d'intérêt personnel d'un hôte. Chaque hôte n_i diffuse périodiquement une annonce qui combine :

- une description de son propre profil d'intérêt $prof(n_i)$
- un catalogue $cat(n_i)$, contenant les descripteurs de documents qui sont disponibles dans son cache local, et qu'il estime être susceptible d'intéresser ses voisins

Cette annonce est diffusée sous la forme d'un unique message de contrôle (fig. 3.7-a), dont la portée de diffusion peut être fixée explicitement par l'hôte émetteur (cet aspect est expliqué plus en détails dans la section 3.3.3).

En diffusant périodiquement une description de son propre profil d'intérêt, un hôte permet à ses voisins de découvrir quel type de documents l'intéresse. Inversement, en recevant le même genre d'information de chacun de ses voisins, l'hôte en question est en mesure de maintenir une vision précise de ce qui les intéresse. Le contenu du catalogue qu'il diffuse périodiquement peut ainsi être ajusté en continu, de manière à ne contenir que les descripteurs de documents qui sont effectivement susceptibles d'intéresser ses voisins. Le coût résultant de la diffusion de ce catalogue peut ainsi être réduit au minimum, la taille du catalogue lui-même étant ajustée au plus bas afin de ne jamais proposer aux hôtes voisins des documents qui, de toute façon, ne correspondent aucunement à leurs profils d'intérêt respectifs. Un hôte disposant dans son cache d'un très grand nombre de documents s'abstiendra de diffuser un catalogue portant sur tous ces documents si seulement un très petit nombre d'entre eux présente effectivement de l'intérêt pour ses voisins.

Réception du catalogue émis par un voisin. Lorsqu'un hôte réceptionne une annonce contenant le catalogue d'un voisin, il examine les descripteurs contenus dans ce catalogue afin d'identifier des documents dont les caractéristiques correspondent à son profil d'intérêt, et qui ne se trouvent pas déjà dans son cache local. Si de tels documents sont effectivement mentionnés dans le catalogue, alors l'hôte récepteur de l'annonce construit une requête à l'intention de l'annonceur, cette requête contenant simplement les identifiants des documents qu'il souhaite obtenir de ce dernier. La requête est ensuite émise à destination de l'annonceur, encapsulée dans un message de contrôle émis en mode *unicast* (fig. 3.7-b).

Traitement des requêtes reçues des voisins. Après avoir diffusé une annonce contenant son catalogue, un hôte est susceptible de recevoir une ou plusieurs requêtes provenant de ses voisins. Ces requêtes sont traitées séquentiellement : pour chaque document demandé par un voisin, l'hôte récupère ce document du cache local, et le diffuse dans le réseau (fig. 3.7-c). On peut noter que le document demandé est diffusé, plutôt que d'être adressé spécifiquement au demandeur en mode *unicast*. Ce choix de conception s'explique par le fait qu'après avoir proposé à ses voisins de leur fournir un certain document, un hôte peut fort bien recevoir plusieurs requêtes pour ce même document de la part de plusieurs voisins. Dans un tel scénario, tous les voisins demandant à obtenir une copie du même document peuvent être satisfaits par une unique diffusion de ce document par l'hôte qui le leur a proposé. Ainsi, pour éviter que des requêtes successives concernant un même document entraînent des réémissions inutiles de ce document, chaque hôte maintient à jour un historique des documents qu'il a récemment diffusés à la demande de ses voisins. Cet historique est réinitialisé à chaque fois que l'hôte diffuse une nouvelle annonce contenant son profil et son catalogue. Ainsi, lorsque plusieurs voisins d'un hôte cherchent à obtenir de lui le même document, ce document est diffusé une fois et une seule sur le médium de transmission.

Réception de nouveaux documents. Tout hôte réceptionnant un document diffusé par l'un de ses voisins doit vérifier s'il est lui-même intéressé par ce document. Si c'est le cas, alors le document peut être déposé dans le cache local (s'il ne s'y trouvait pas déjà). Dès lors, tout hôte ayant décidé d'accepter un nouveau document va pouvoir jouer le rôle de transporteur mobile pour ce document, et contribuer ainsi à en assurer la dissémination dans le réseau.

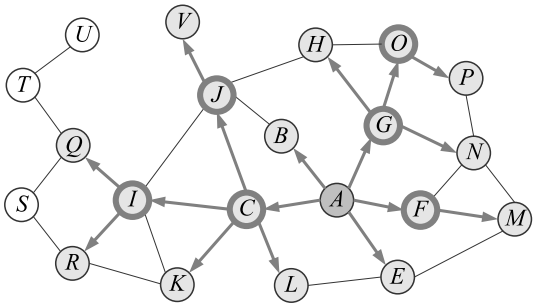
On peut noter que ce modèle de communication permet à des hôtes mobiles de collecter des documents qui les intéressent sans même avoir à en faire la demande, en les interceptant simplement lors de leur diffusion sur le médium radio. Ainsi, un hôte n_i peut obtenir un document en en faisant la demande explicite à l'un de ses voisins n_j, mais il peut aussi obtenir ce document parce qu'un autre hôte n_k (qui n'est d'ailleurs pas forcément l'un de ses propres voisins) en a fait la demande à n_j. Les expérimentations que nous avons menées (tant en conditions réelles qu'avec un simulateur) ont montré que cette possibilité qu'ont les hôtes mobiles d'obtenir certains documents « par hasard », sans même en avoir fait la demande, constitue une conséquence logique mais néanmoins intéressante de notre décision d'utiliser des transmissions en diffusion plutôt qu'en mode *unicast* chaque fois que c'est possible. En effet, un hôte qui obtient un document intéressant sans même en avoir fait la demande au préalable va pouvoir s'abstenir de demander ce document plus tard à un quelconque voisin. On économise ainsi un certain nombre de transmissions, ce qui dans un environnement aux ressources limitées est toujours appréciable.

3.3.3 Relais immédiat des messages entre hôtes voisins (couche inférieure)

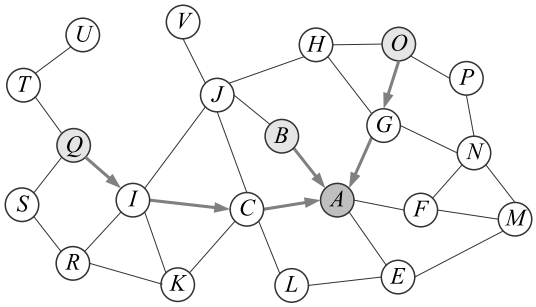
Comme expliqué dans la section précédente, la couche supérieure de notre protocole nécessite qu'un hôte soit en mesure d'envoyer des messages (contenant soit ses profil et catalogue, soit une requête, soit un document) à ses voisins du moment. Le relais immédiat de messages (par opposition au relais différé) est donc souhaitable afin d'exploiter au mieux la connectivité transitoire existant entre des hôtes qui se trouvent appartenir — peut-être de façon extrêmement fugitive — à un même fragment connexe (îlot) du réseau. La couche inférieure de notre protocole a pour vocation de supporter ce type de relais immédiat, qu'il s'agisse de faire parvenir un message à un hôte spécifique situé dans le voisinage de l'émetteur (trafic *unicast*), ou bien à l'ensemble des voisins de cet émetteur (trafic *broadcast*).

Relais immédiat de messages à diffuser. La diffusion multi-sauts dans un MANET est réputée être une opération extrêmement coûteuse en termes de bande passante, et qui peut même à l'occasion mener au phénomène dit « d'orage de diffusion » *(broadcast storm problem)* [138]. Pour limiter le coût occasionné par la diffusion d'un message, la couche inférieure de notre protocole met en œuvre un mécanisme qui est inspiré de celui utilisé dans le protocole OLSR *(Optimized Link State Routing)* pour assurer la diffusion dans l'ensemble du réseau d'informations portant sur l'état des liens dans le réseau [30, 147].

Concrètement, chaque hôte sélectionne périodiquement un sous-ensemble de ses voisins directs (situés à portée radio) et les enregistre en tant que relais multi-points *(MPR : Multi-Point Relays)*. Il s'appuiera ensuite sur ces MPRs, et sur eux-seuls, pour diffuser des messages au delà de sa propre portée radio. Dans OLSR, qui est un protocole de routage proactif, les MPRs sont utilisés exclusivement pour diffuser dans l'ensemble du réseau (supposé connexe) des informations de contrôle relatives à l'état des tables de routage de chacun des hôtes. Ces informations sont alors utilisées localement par chaque hôte pour mettre à jour sa propre table de routage. Dans notre protocole, les MPRs servent à



(a) A diffuse un message dans son environnement à 3 sauts, ce message étant relayé exclusivement par les relais multi-points (i.e. {C, G, F}, puis {I, J, O}).



(b) B, O, et Q répondent au message précédent, en utilisant le *source-routing* pour spécifier dans chaque message le chemin à suivre pour "remonter" vers A.

FIG. 3.8 – Illustration des deux modes de relais immédiat multi-sauts supportés par le protocole (trafic *broadcast* à gauche, trafic *unicast* avec routage par la source à droite)

diffuser n'importe quel type de message (qu'il s'agisse d'un message encapsulant des informations de contrôle ou des données utiles), au sein de l'îlot auquel appartient l'hôte émetteur, et ce seulement jusqu'à une certaine distance de cet émetteur. En effet, avec notre protocole l'émetteur d'un message à diffuser a la possibilité de spécifier combien de sauts consécutif (entre hôtes voisins) ce message va pouvoir effectuer avant de cesser de se propager. La figure 3.8-a montre ainsi un exemple dans lequel l'hôte A diffuse un message, qui pourrait par exemple être une annonce contenant son profil d'intérêt et le catalogue des documents qu'il met à disposition de ses voisins. Dans cet exemple, chaque copie du message n'est autorisée à se propager que sur trois sauts consécutifs, ce qui explique pourquoi les hôtes S, T et U, qui sont situés trop loin de l'hôte A bien qu'appartenant au même îlot que lui, ne reçoivent pas son annonce. En outre, on peut observer dans la figure que seuls certains voisins de l'hôte A réémettent effectivement le message lorsqu'il le reçoivent. Dans ce cas précis, il s'agit des hôtes C, F, et G, qui font office de relais multi-points après le premier saut réalisé par le message, et des hôtes I, J, et O, servant de relais multi-points après le second saut du message.

L'algorithme utilisé par chaque hôte pour sélectionner parmi ses voisins ceux qui vont servir de relais multi-points n'est décrit ci-dessous que très sommairement, dans la mesure où il s'agit d'un algorithme « emprunté » au protocole OLSR, et qu'à ce titre il a déjà été abondamment décrit et validé dans la littérature [147]. Il a notamment été démontré que, dans un réseau MANET suffisamment dense, l'approche consistant à faire réaliser la diffusion par des MPRs est nettement moins coûteuse que celle qui consiste à réaliser une simple inondation, chaque hôte réémettant simplement chaque message sur le médium radio lorsqu'il le reçoit pour la première fois.

Dans l'algorithme mis en œuvre dans le protocole OLSR, chaque hôte doit diffuser périodiquement un message de contrôle visant à informer ses voisins directs (*i.e.* voisins à un saut) de sa présence dans le réseau, tout en leur décrivant la perception qu'il a lui-même de son propre voisinage. En recevant de tels messages de ses divers voisins, un hôte est en mesure d'identifier quels sont ses voisins à un saut et ses voisins à deux sauts, et utiliser cette information pour calculer l'ensemble des MPRs sur lesquels il va pouvoir s'appuyer pour diffuser des messages sur l'ensemble de son voisinage à deux sauts. Dans le protocole OLSR (tel que décrit par exemple dans [147]), des messages de contrôle spécifiques sont donc diffusés périodiquement par chaque hôte, ces messages contenant les informations requises pour le calcul des ensemble de MPRs. Dans notre propre protocole, l'information permettant

le calcul de ces MPRs est transportée dans les messages de contrôle utilisés par ailleurs pour diffuser les annonces réalisées par la couche supérieure du protocole (cf. paragraphe 3.3.2). Ainsi le calcul des ensembles de MPRs ne nécessite la diffusion d'aucun message supplémentaire dans le réseau : les deux types d'information de contrôle (requises par les deux couches du protocole) sont transmises simultanément dans le réseau.

Relais immédiat de messages *unicast*. La couche supérieure de notre protocole nécessite que les hôtes mobiles soient en mesure d'émettre une requête en réponse à une annonce qu'ils viennent de recevoir, cette requête étant bien sûr adressée à l'émetteur de l'annonce (cf. paragraphe 3.3.2). Des messages *unicast* doivent donc pouvoir être routés vers l'émetteur d'un message qui vient tout juste d'être diffusé. Le routage par la source (dans lequel l'émetteur d'un message spécifie explicitement quel trajet ce message doit suivre dans le réseau) est approprié dans de telles circonstances. Chaque message diffusé par la couche basse de notre protocole contient un historique des hôtes par lesquels il a transité jusqu'alors. Cet historique est bien sûr mis à jour à chaque fois que le message transite par un nouvel hôte. Ainsi, lorsqu'un hôte ayant réceptionné un message diffusé estime qu'il doit y répondre, le trajet que doit suivre cette réponse peut être déduit du chemin que le message diffusé a lui-même suivi pour atteindre cet hôte. On peut noter que, pour que cette approche fonctionne dans un réseau dont tous les hôtes sont fortement mobiles, il est nécessaire qu'un message *unicast* émis en réponse à un message diffusé soit émis sans attendre. Dans de telles conditions, on peut estimer que le cheminement que le message diffusé vient de suivre est encore praticable — mais en sens inverse — pour la réponse à ce message.

Considérons de nouveau l'exemple illustré dans la figure 3.8-a, et supposons que les hôtes B, Q, et O décident de répondre au message diffusé par A. La figure 3.8-b montre comment ces réponses peuvent se propager en « remontant » le chemin que le message diffusé vient juste de « descendre », chaque réponse contenant la description explicite du cheminement qu'elle doit suivre afin d'atteindre l'hôte A.

3.3.4 Évaluation

Notre protocole a été évalué au cours de plusieurs campagnes de tests, qui nous ont permis d'observer comment il se comporte dans différentes conditions. Ces campagnes ont été réalisées à la fois dans des conditions d'expérimentation réelles impliquant une douzaine d'ordinateurs portables, et à l'aide du simulateur MADHOC dont une description sommaire est fournie dans le paragraphe 3.3.5. Quelques uns des résultats obtenus lors des simulations sont présentés ci-dessous.

Paramètres de simulation. Nous considérons un scénario de simulation dans lequel une population de 120 individus évolue dans un environnement de type « campus » (ou dans l'enceinte d'une entreprise). Cet environnement est constitué de quatre bâtiments (voir fig. 3.9), répartis sur une superficie de $300\,m \times 800\,m$. Chacun des individus évoluant dans cet environnement est supposé être équipé d'un PC portable doté d'une interface Wi-Fi (IEEE 802.11).

La mobilité des individus — et donc, indirectement, celle des équipements qu'ils transportent — est simulée en utilisant une variante du modèle *Random Waypoint* traditionnel : un indidivu peut rester immobile dans un bâtiment pendant un certain temps, avant de se déplacer vers une destination choisie au hasard dans l'un quelconque des cinq bâtiments (y compris le bâtiment dans lequel il se trouve

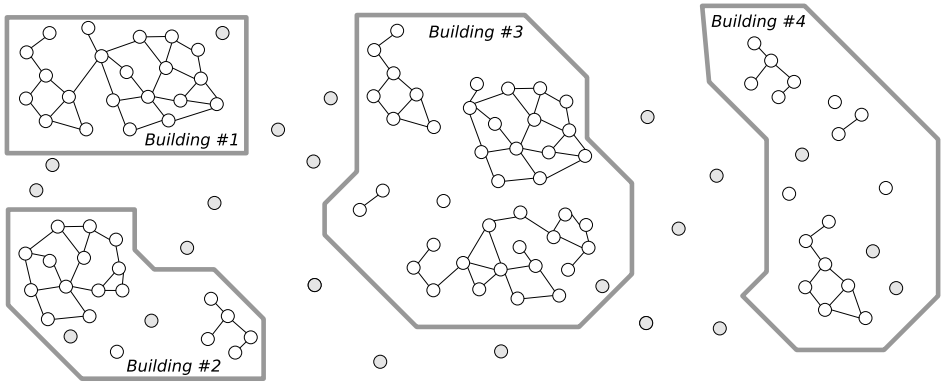


FIG. 3.9 – Illustration du type d'environnement considéré au cours de la campagne de simulation

déjà). Ce modèle permet donc de simuler des déplacements d'individus de bâtiment en bâtiment, mais aussi au sein d'un même bâtiment.

Pour obtenir les résultats décrits plus loin, les paramètres de mobilité suivants ont été utilisés : la vitesse de déplacement des individus peut varier entre 0,5 m/s et 2 m/s (ce qui correspond à des vitesses de déplacements typiques pour des piétons). Un individu peut rester immobile (entre deux déplacements consécutifs) pour une durée se situant entre 30 secondes et 3 minutes. La proportion de déplacements au sein d'un même bâtiment est fixée à 40 % (contre 60 % pour les déplacements entre bâtiments). Les interfaces équipant les PC portables sont supposées avoir une portée de transmission omni-directionnelle de 40 mètres en milieu fermé (*i.e.* à l'intérieur d'un bâtiment), et de 100 mètres en milieu ouvert. Enfin, puisque dans la « vraie vie » un PC portable est en général éteint (ou mis en veille) lorsque son propriétaire se déplace, le modèle de mobilité que nous avons défini tient compte de cette observation. Avec ce modèle, un hôte mobile (*i.e.* un PC portable dans le cas présent) est considéré comme étant éteint (et donc incapable de communiquer avec tout autre équipement) lorsque son propriétaire est en mouvement, et comme étant allumé (et donc apte à interagir avec les équipements voisins) lorsque son propriétaire est à l'arrêt.

Scénario de communication. Nous considérons un scénario dans lequel les hôtes mobiles produisent de nouveaux documents (chaque document pesant 50 ko), et les injectent dans le réseau à un rythme moyen de 1 document (par hôte) toutes les 5 minutes. Compte tenu du nombre d'hôtes constituant le réseau, ceci correspond à un rythme global de 1 nouveau document injecté dans le réseau toutes les 2,5 secondes. Chaque document porte (dans son descripteur) une étiquette qui le place dans une certaine catégorie thématique. Il y a 16 catégories distinctes, mais chaque document s'inscrit dans seulement l'une de ces catégories.

Chaque hôte mobile affiche de l'intérêt pour 2 des 16 catégories (soit $1/8$ du trafic global), et souhaite donc recevoir les documents relatifs à ces deux catégories exclusivement. Le nombre total d'hôtes dans le réseau (*i.e.* 120 hôtes) a été choisi de telle sorte qu'on puisse attribuer un profil d'intérêt distinct à chacun de ces hôtes. En d'autres termes, on ne peut trouver deux hôtes dans le réseau qui affichent exactement le même profil d'intérêt, et soient donc intéressés exactement par les mêmes types de documents.

Paramètres protocolaires. Le comportement général de notre protocole est conditionné par deux paramètres essentiels. Le premier de ces paramètres est la période avec laquelle un hôte mobile va diffuser des annonces (contenant son profil d'intérêt et le catalogue des documents qu'il propose à ses voisins). Une valeur convenable pour cette période peut être déduite des conditions dans lesquelles le protocole doit être utilisé, et notamment des caractéristiques de mobilité des hôtes dans cet environnement. Ainsi, avec les paramètres de simulation décrits plus haut, les hôtes mobiles sont supposés être transportés par des piétons, et n'être actifs que lorsque leurs propriétaires sont immobiles. Dans de telles conditions il n'est pas nécessaire d'adopter une période d'annonce très courte. En l'occurrence, une période d'annonce de 15 secondes suffit pour laisser aux hôtes mobiles, lors d'une réactivation consécutive à un déplacement, le temps de découvrir les équipements voisins, et d'échanger des documents avec ces voisins. Une période d'annonce plus courte serait toutefois nécessaire si les hôtes mobiles étaient susceptibles de communiquer pendant leurs déplacements (comme ce pourrait être le cas, par exemple, pour des PDAs), cette période devant en outre être d'autant plus courte que les hôtes en question se déplacent rapidement (et sont donc susceptibles d'observer de très brefs contacts avec des équipements voisins). Inversement, la période d'annonce pourrait être rallongée, si les hôtes considérés étaient susceptibles de passer beaucoup plus de temps au même endroit, et donc d'observer autour d'eux un voisinage relativement stable.

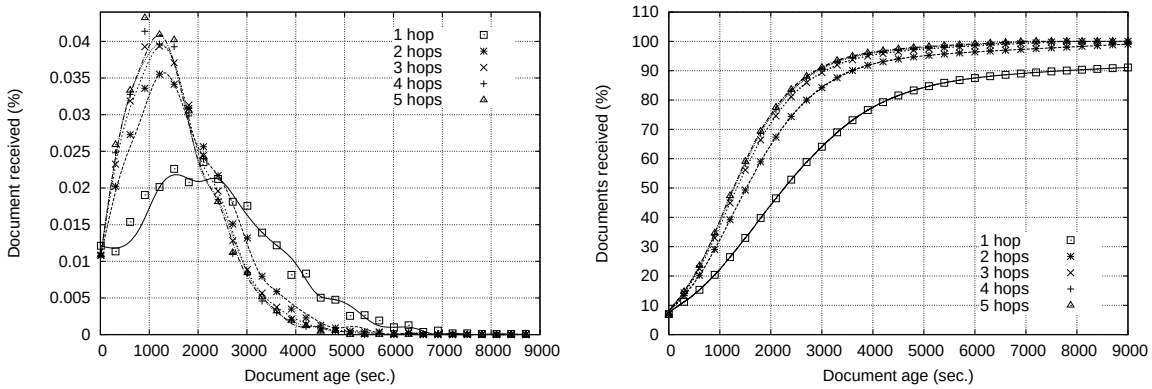
Le second paramètre qui peut être ajusté dans notre protocole est le nombre de sauts admissible dans le cas du relais immédiat d'un message par la couche inférieure du protocole, et en particulier lorsqu'un hôte diffuse une annonce à l'intention de ses voisins. En ajustant ce paramètre, on peut en effet déterminer l'étendue de la « sphère de communication » au centre de laquelle se trouve chaque hôte mobile, c'est-à-dire — indirectement — l'envergure du voisinage avec lequel il est disposé à échanger des documents avant de se déplacer de nouveau dans le réseau.

Vitesse de propagation des documents. Notre objectif premier est d'observer dans quelle mesure la portée du relais immédiat de messages peut influencer les performances générales de notre protocole. Le résultat attendu est que, lorsque chaque hôte mobile est autorisé à utiliser le relais multi-sauts afin d'accroître la population des voisins avec lesquels il peut interagir, la dissémination des documents dans le réseau s'effectue plus rapidement que lorsque chaque hôte ne peut interagir qu'avec ses voisins directs (*i.e.* voisins « à un saut »).

Pour vérifier que cette hypothèse est effectivement vérifiée, nous considérons dans un premier temps un scénario — fort peu réaliste il est vrai — dans lequel les documents peuvent se propager éternellement dans le réseau. Pour ce faire nous supposons que la capacité du cache est illimitée sur chacun des hôtes considérés dans la simulation, et que les documents injectés dans le réseau n'ont pas de durée de vie précise.

Le modèle de mobilité utilisé dans cette campagne de simulation (*i.e. Constrained Random Waypoint)* garantit que chaque hôte mobile entrera à un moment ou un autre en contact avec chacun des autres hôtes du réseau. Dans ces conditions, un document pouvant demeurer éternellement dans le réseau est assuré d'atteindre à terme tous les hôtes intéressés. Cependant, le temps nécessaire pour qu'un document atteigne un récepteur précis peut être nettement influencé par les paramètres de notre protocole, et notamment par l'étendue du voisinage avec lequel chaque hôte est autorisé à interagir.

Dans la figure 3.10, on observe le temps nécessaire (en moyenne) à un document pour atteindre les hôtes pour lesquels il présente de l'intérêt. Plus précisément, la fig. 3.10-a montre la distribution normalisée de l'âge des documents lors de leur réception, et la fig. 3.10-b montre la distribution cumulée correspondante.



(a) Distribution de l'âge des documents lors de leur réception

(b) Distribution cumulée de l'âge des documents lors de leur réception

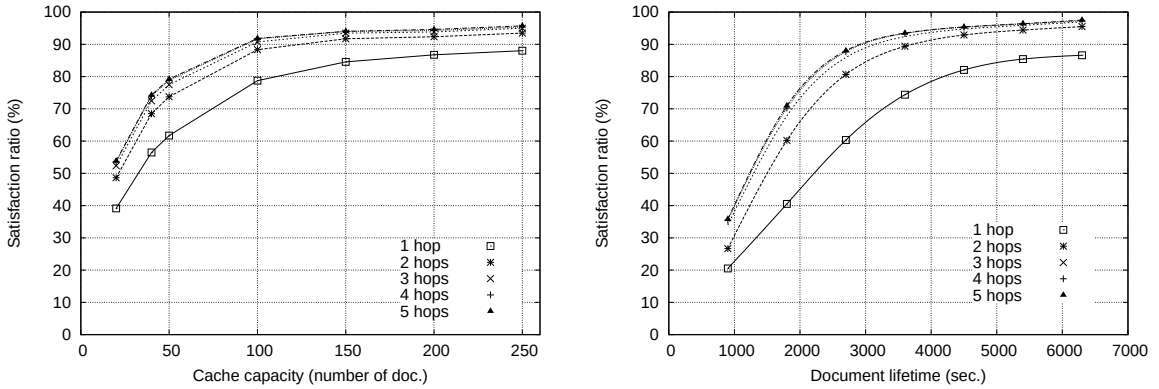
FIG. 3.10 – Distributions simple et cumulée de l'âge des documents lors de leur réception par un hôte intéressé

Considérons tout d'abord le cas où les hôtes ne peuvent utiliser que des transmissions à 1-saut pour interagir avec leur voisinage. En d'autres termes, il s'agit d'un cas dans lequel la couche inférieure de notre protocole ne joue aucun rôle, seules les communications entre voisins directs étant possibles. On peut constater que, dans de telles circonstances, environ 40 % des documents atteignent néanmoins les hôtes intéressés en moins de 30 minutes. Au bout d'une heure, environ 75 % des documents ont atteint les récepteurs intéressés, et après environ deux heures environ 90 % des documents ont atteint les récepteurs intéressés. Ces valeurs vont nous servir de référence pour évaluer l'impact du relais immédiat multi-sauts, tel qu'il est assuré par la couche inférieure de notre protocole.

La courbe de la figure 3.10-a montre que, lorsque du relais à deux sauts est utilisé (c'est-à-dire lorsque un hôte peut interagir directement avec ses voisins à 1 saut et à 2 sauts), la plupart des documents parviennent aux hôtes intéressés en moins de 20 minutes (contre 30 minutes lorsque seules sont utilisées des communications à 1 saut). Dans ces conditions, environ 98 % des documents parviennent en fait aux hôtes intéressés en moins de 2 heures, environ 90 % y parviennent en moins d'une heure, et environ 60 % en moins de 30 minutes.

Une amélioration similaire — quoique moins importante quantitativement — peut être observée lorsque l'on étend encore la sphère de communication de chaque hôte en l'autorisant à interagir avec ses voisins à 3 sauts, à 4 sauts, et à 5 sauts respectivement.

En fait, avec les paramètres de simulation utilisés dans cette campagne d'évaluation, les îlots (ou fragments connexes du réseau) pouvant se former au sein des bâtiments ont une élongation qui varie entre 0 (lorsqu'un hôte isolé n'a aucun voisin) et 7 sauts, avec une valeur moyenne de 4.2 sauts. Ceci explique pourquoi le fait d'étendre la portée (théorique) du relais immédiat multi-sauts dans le paramétrage de notre protocole n'apporte pas nécessairement d'amélioration significative dans la simulation. Lorsqu'un hôte est autorisé à diffuser ses annonces jusqu'à, par exemple, 8 sauts de là, encore faut-il qu'il ait effectivement des voisins situés aussi loin de lui-même. Une raison complémentaire est que la vitesse à laquelle les documents se propagent entre les bâtiments (ou entre des zones non connectées d'un même bâtiment) dépend directement de la vitesse à laquelle les transporteurs mobiles de ces documents — c'est-à-dire, ici, des piétons — se déplacent dans la zone de simulation.



(a) Taux de satisfaction vs. capacité des caches (b) Taux de satisfaction vs. durée de vie des documents

FIG. 3.11 – Variation du taux de délivrance (taux de remise des documents aux hôtes intéressés) en fonction des contraintes posées sur la dissémination

Cette première expérience confirme qu'en étendant la sphère de communication de chaque hôte mobile, notre protocole permet bien une dissémination plus rapide des documents dans chaque îlot, ce qui contribue aussi à faire croître plus rapidement le nombre d'hôtes porteurs d'un même document, et permet au final une dissémination plus rapide des documents sur l'ensemble des fragments non connectés du réseau.

Influence de la capacité du cache. Dans les simulations dont les résultats ont été discutés plus haut, nous avons supposé que les documents pouvaient se propager indéfiniment dans le réseau. Cependant une telle hypothèse n'est évidemment guère réaliste, dans la mesure où les ressources disponibles dans un réseau de type MANET sont en général limitées, et souvent inférieures à celles dont on peut disposer dans un environnement filaire. Par exemple, le cache dans lequel un hôte mobile peut stocker des documents est forcément de capacité limitée. Une politique de gestion de cache appropriée doit donc être définie et appliquée sur chaque hôte mobile afin d'éviter la saturation de cache et, si cette saturation survient malgré tout, de résoudre ce problème.

Les courbes de la figure 3.11-a montrent l'influence de la capacité du cache sur les performances observées lors de la dissémination de documents. Pour produire ces résultats nous avons mené une nouvelle campagne de simulations, en faisant varier sur chaque hôte la capacité du cache entre 50 et 200 documents. Pendant ces simulations la politique de gestion de cache appliquée était telle que, lorsqu'un cache atteignait le niveau de saturation, le document le plus ancien contenu dans ce cache était supprimé afin de libérer de la place pour un nouveau document.

La figure 3.11-a montre l'évolution du taux de délivrance (défini comme le pourcentage de documents atteignant effectivement les récepteurs intéressés), variant en fonction de la capacité du cache sur chaque hôte. On peut tout d'abord constater que les courbes présentées confirment l'intuition naturelle selon laquelle un hôte mobile doté d'un cache de grande capacité est en mesure de transporter les documents plus longtemps (et donc plus loin) dans le réseau.

On peut également observer l'influence de l'utilisation du relais immédiat multi-sauts sur la dissémination des documents. Dans la figure 3.11-a on voit notamment que le taux de délivrance augmente

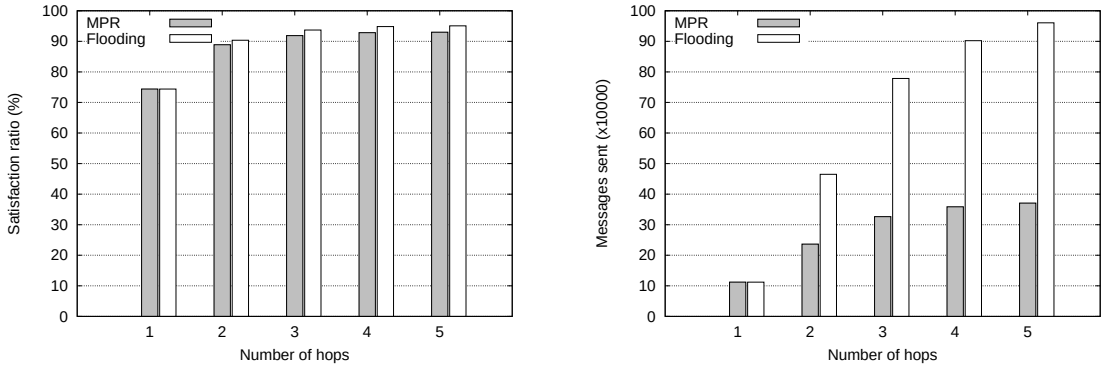
de manière significative lorsque la portée du relais immédiat est étendue à quelques sauts autour de chaque hôte. Considérons par exemple le cas où chaque hôte ne dispose que d'un cache capable de stocker 100 documents. Dans ces conditions, les documents injectés dans le réseau ne parviennent (en moyenne) qu'à 78 % des récepteurs intéressés si chaque hôte ne peut interagir qu'avec ses voisins directs. En revanche ce taux est accru de 10 % lorsque chaque hôte peut interagir avec ses voisins à 2 sauts, et il est encore accru de 2 % supplémentaires lorsque chaque hôte peut atteindre ses voisins à 3 sauts.

Influence de la durée de vie des documents. Une autre manière d'éviter que les documents demeurent éternellement dans le réseau consiste à donner à chaque document une durée de vie précise, lors de son injection dans le réseau. Ainsi, dès qu'un document devient obsolète, toutes les copies de ce document sont automatiquement supprimées des caches dans lesquelles elles pouvaient être stockées. Cette méthode peut d'ailleurs être utilisée, soit en remplacement, soit en complément de celle qui consiste à limiter la capacité des caches sur les hôtes mobiles.

Les courbes de la figure 3.11-b montrent l'influence de la durée de vie des documents sur l'efficacité de leur dissémination. Ces résultats ont été obtenus avec des caches de capacité illimitée, afin que les deux types de contraintes (durée de vie des documents et capacité des caches) n'interfèrent pas pendant les simulations. Dans la figure 3.11-b nous présentons l'évolution du taux de documents parvenant aux récepteurs intéressés en fonction de la durée de vie attribuée à ces documents. Comme on pouvait s'y attendre, le taux de délivrance s'accroît avec la durée de vie des documents. Cependant on peut constater que, cette fois encore, l'utilisation du mécanisme de relais immédiat multi-sauts permet une amélioration significative des performances. Ainsi, lorsque les documents ont une durée de vie de seulement 30 minutes, ils ne sont reçus (en moyenne) que par 30 % des récepteurs intéressés si seuls les échanges entre voisins directs (à 1 saut) sont possibles. Ce taux est accru de 20 % lorsque chaque hôte peut atteindre ses voisins à 2 sauts, et il l'est encore de 7 % lorsque chaque hôte peut atteindre ses voisins à 3 sauts.

Il est à noter que le fait de limiter la durée de vie des documents n'est pas tout à fait équivalent au fait de limiter la capacité du cache sur chaque hôte, bien que ces deux approches puissent être utilisées pour réguler la quantité de documents en cours de dissémination dans le réseau. En fait, la capacité du cache sur un hôte donné résulte en général directement de la quantité de ressources de stockage disponibles sur cet hôte. Il s'agit donc pour l'essentiel d'une contrainte liée à la gestion même de cet hôte, en fonction des ressources dont il dispose. En revanche, la durée de vie octroyée à un document doit être fixée par l'émetteur de ce document. Elle peut être fixée soit sur la base d'une estimation du temps nécessaire à ce document pour atteindre tout ou partie des hôtes qu'il va intéresser, soit sur la base d'une estimation de la durée pendant laquelle les informations contenues dans ce document doivent être considérées comme étant pertinentes pour les hôtes récepteurs. Dans le premier cas, la décision est d'ordre stratégique et nécessite une certaine connaissance de l'état du réseau (*i.e.* envergure, schémas de mobilité des hôtes, fréquences des contacts, etc.). Dans le second cas, la décision est plutôt d'ordre sémantique, puisqu'elle conditionne la persistance des documents dans le réseau en fonction de leur contenu.

Évaluation du coût du relais immédiat multi-sauts. Les résultats précédents confirment qu'en utilisant le mécanisme de relais immédiat multi-sauts, la dissémination de documents dans le réseau peut être réalisée de façon plus rapide, et donc plus efficacement. Ils montrent également qu'une faible extension de la sphère de communication de chaque hôte mobile (lui permettant par exemple



(a) Taux de satisfaction (b) Coût de la transmission

FIG. 3.12 – Comparaison entre des protocoles utilisant des relais multi-points et un simple mécanisme d'inondation

d'atteindre ses voisins à 2 ou 3 sauts) peut déjà apporter un progrès significatif par rapport au cas où seuls sont possibles les échanges entre voisins directs.

L'inconvénient des transmissions multi-sauts est bien évidemment qu'elles mobilisent énormément de ressources. En effet, à chaque fois qu'un hôte relaie un message, cette opération draine la batterie de cet hôte ainsi que celles des récepteurs voisins, tout en monopolisant le médium radio tout autour de l'hôte relais.

En concevant notre protocole nous avons décidé de nous appuyer sur le principe des relais multi-points (principe emprunté au protocole OLSR) afin d'assurer la diffusion des messages. De toute évidence, il aurait été plus simple pour nous de mettre en œuvre un simple algorithme d'inondation pour diffuser ces messages. Puisque les résultats présentés plus haut montrent qu'une amélioration significative des performances de la dissémination des documents peut être obtenue (dans le scénario considéré) en relayant les messages sur iun très faible nombre de sauts (typiquement deux ou trois sauts), on peut légitimement se demander si l'utilisation de relais multi-points dans un tel contexte apporte un quelconque bénéfice par rapport à un simple mécanisme d'inondation.

Pour comparer les coûts de fonctionnement respectifs de deux approches reposant, l'une sur des relais multi-points, l'autre sur un mécanisme d'inondation, nous avons mis en œuvre une version dégradée de notre protocole, dans laquelle la couche inférieure du protocole assure la diffusion de messages par inondation plutôt que via des relais multi-points. Nous avons ensuite comparé le comportement de ces deux versions du protocole, en appliquant bien sûr dans les deux cas le même scénario de mobilité et de communication. Les résultats sont présentés dans la figure 3.12. Ils ont été produits en réalisant des simulations de 4 heures (en temps simulé), avec des caches de capacité illimitée, et des documents ayant chacun une durée de vie d'un heure.

On peut tout d'abord constater (fig. 3.12-a) que les deux versions du protocole ne donnent pas tout à fait les mêmes taux de délivrance. Ceci est dû au fait qu'avec la version standard de notre protocole (utilisant les MPRs) la dissémination des documents s'effectue légèrement plus lentement qu'avec la version reposant sur le mécanisme d'inondation. En effet, un hôte dont le voisinage est modifié doit attendre quelques temps (précisément, deux cycles d'annonce consécutifs) avant de découvrir l'ensemble de ses nouveaux voisins, et donc de pouvoir adapter ses annonces à leurs profils

d'intérêts respectifs. Dans la version reposant sur de l'inondation, en revanche, ce temps de latence initial n'existe pas : un hôte dont le voisinage change peut immédiatement proposer un catalogue de documents susceptibles d'intéresser ses nouveaux voisins.

Le taux de délivrance est donc très légèrement inférieur avec la version standard de notre protocole qu'avec la version pratiquant l'inondation. Cette différence demeure cependant inférieure à 3 %, alors que le coût relatif lié à l'utilisation de l'une ou l'autre méthode de diffusion est fort différent. On peut en effet observer dans la figure 3.12-b que le nombre de transmissions élémentaires (correspondant au nombre de fois où un message est émis ou réémis dans le réseau) est très nettement supérieur avec le mécanisme d'inondation qu'avec le mécanisme reposant sur les MPRs. Ce constat conforte notre décision de nous appuyer sur des relais multi-points pour assurer la diffusion de messages tout autour d'un hôte émetteur. Enfin on peut noter que le coût de la diffusion via des MPRs ne croît pas énormément lorsque la sphère de communication de chaque hôte est étendue au delà de ses voisins à 4 ou 5 sauts. Ceci résulte du fait qu'avec les paramètres utilisés lors de cette campagne de simulation, l'élongation de chaque îlot dans le réseau se situe à 4.2 sauts en moyenne.

3.3.5 Mise en œuvre

Intégration à la plate-forme intergicielle DoDWAN. Le protocole décrit dans les paragraphes précédents a été pleinement implémenté en Java, et embarqué dans la plate-forme intergicielle DoDWAN *(Document Dissemination in Wireless Ad hoc Networks)*. DoDWAN est une plate-forme que nous avons conçue afin d'aider à la mise en œuvre de services applicatifs capables de fonctionner dans un réseau MANET discontinu.

Cette plate-forme peut être perçue comme un MOM *(Message-Oriented Middleware)*. Elle se distingue toutefois de produits tels que JMS *(Java Message Service)* ou Emma [136] par le fait que les messages qu'elle manipule sont en fait des documents à part entière, le traitement accordé à ces documents par DoDWAN pouvant dépendre de leurs caractéristiques propres, et non simplement de leur destination assignée. À chaque document est associé un descripteur exprimé en XML (comme illustré dans la figure 3.5). Le profil caractérisant l'intérêt qu'un hôte — sur lequel s'exécute DoDWAN — porte à tel ou tel type de documents est également exprimé en XML (voir fig. 3.6). DoDWAN supporte par ailleurs la fragmentation et le réassemblage des documents lourds, qui ne peuvent être transmis en une seule fois sur le médium de transmission. Lorsqu'un document de ce type doit être diffusé, il est découpé en segments distincts, qui sont ensuite encapsulés dans des « documents fragments », dont les descripteurs reprennent les mêmes attributs qui caractérisaient le document initial. Ces fragments peuvent alors être diffusés, et se propager dans le réseau indépendamment les uns des autres. Ils seront à terme collectés par les seuls hôtes intéressés, au niveau desquels le réassemblage des divers fragments permettra de reconstituer le document initial. On peut noter qu'avec cette approche un hôte peut faire office de transporteur pour certains fragments d'un document, même s'il ne dispose pas de l'ensemble des fragments qui permettraient de reconstituer ce document dans son intégralité. Cette caractéristique contribue à assurer la robustesse du système, dans la mesure où il n'est pas nécessaire qu'un hôte dispose d'un document complet pour pouvoir malgré tout contribuer à la dissémination de ce document dans le réseau.

Les documents (complets ou fragmentés) hébergés par DoDWAN sont maintenus dans un cache, qui peut être mis en œuvre à la fois en mémoire et dans un système de fichiers. Le maintien de tout ou partie du cache en mémoire permet d'obtenir des temps de réponse courts lorsqu'il faut, par exemple, rechercher dans le cache les documents répondant au profil d'un hôte voisin. Dans le même temps,

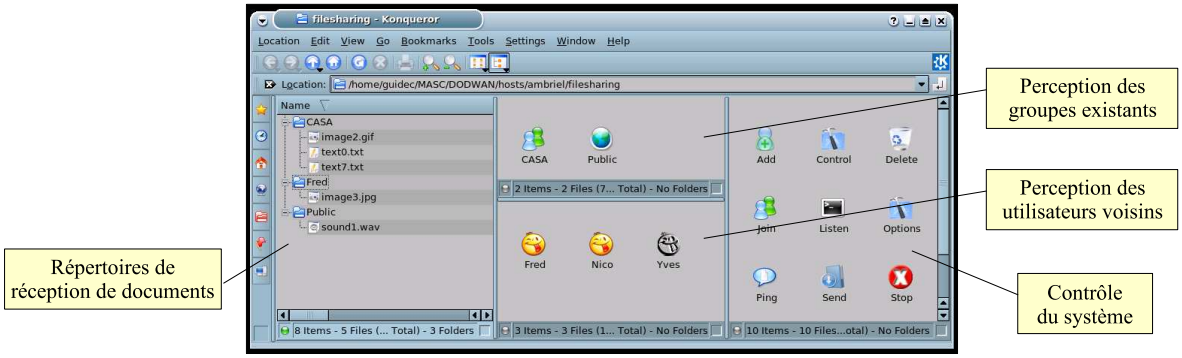


FIG. 3.13 – Aperçu de l'interface graphique de la maquette de démonstration de la plate-forme DOD-WAN

le maintien du cache dans un système de fichiers permet d'assurer sa persistance en cas d'arrêt de l'intergiciel.

Les communications s'effectuent à travers une pile IP standard, les documents étant embarqués dans des datagrammes UDP. Jusqu'à ce jour les expérimentations ont pour l'essentiel été menées avec des interfaces de type Wi-Fi (IEEE 801.11), mais des essais réalisés avec des interfaces Bluetooth (IEEE 802.15) se sont également avérés concluants. Par ailleurs, un interfaçage de DODWAN avec des équipements de type PR4G (postes VHF/FM utilisés par la plupart des forces armées européennes) devrait être effectué dans les mois à venir[11].

Maquette de démonstration. DODWAN offre une API de type *publish/subscribe*, qui permet de développer aisément des services applicatifs exploitant le modèle de communication opportuniste tolérant les délais.

Une maquette de démonstration reposant sur DODWAN a par ailleurs été bâtie afin d'illustrer à la fois le potentiel de la plate-forme et du modèle de communication qu'elle supporte. Cette maquette utilise une interface graphique développée en détournant le navigateur Konqueror de sa fonction première (voir figure 3.13). Avec cette interface, un utilisateur peut découvrir quels sont les autres utilisateurs présents dans son voisinage. Il peut en outre s'inscrire dans un groupe d'utilisateurs (ou créer un nouveau groupe au besoin) afin de partager des documents avec les autres membres du groupe. Il peut adresser un document à un utilisateur précis (que celui-ci soit présent dans le voisinage ou non), ou à un groupe d'utilisateurs. Il peut de même recevoir les documents que d'autres utilisateurs lui ont adressés, et ouvrir ces documents localement à l'aide des applications disponibles sur son terminal. Les « documents » en question peuvent être des fichiers de n'importe quel type (*e.g.* fichiers texte, PDF, images, fichiers audio ou vidéo, etc.), Konqueror se chargeant d'invoquer l'application adéquate pour ouvrir chaque fichier. L'acheminement des documents entre les différents utilisateurs repose bien sûr sur les mécanismes de communication offerts par la plate-forme DODWAN, et notamment sur le protocole de dissémination « basée contenus » présenté précédemment.

Il est intéressant de noter que notre maquette de démonstration offre des services similaires à ceux évoqués dans [123]. Cet article propose en effet d'adopter une approche semblable à la nôtre pour

[11] Ce projet est évoqué plus en détails dans le chapitre 5.

permettre à des équipements mobiles d'échanger des contenus typés (tels que de petits fichiers audio ou vidéo), en utilisant pour ce faire des « canaux » thématiques permettant aux utilisateurs de publier et de souscrire à certains types de contenus de façon sélective.

Interfaçage avec le simulateur MADHOC. À ce jour la plate-forme DoDWAN a été utilisée sur des ensembles intégrant au plus une douzaine d'équipements mobiles. Dans de telles conditions, il est déjà fort difficile d'observer finement le comportement du protocole qui permet à ces équipements de communiquer. Pour aller au delà, et évaluer comment notre protocole peut se comporter dans des environnements intégrant plusieurs dizaines, voire plusieurs centaines d'équipements mobiles, DoDWAN a été conçu afin de pouvoir aisément s'interfacer avec le simulateur de réseau mobile MADHOC [94]. Ce simulateur a été développé par des membres du laboratoire d'informatique du Havre (LITIS), avec lesquels nous collaborons par ailleurs dans le cadre du projet ANR SARAH (cf. annexe A). Il présente l'avantage de pouvoir simuler des réseaux de grande taille, incluant potentiellement plusieurs milliers d'hôtes mobiles. MADHOC définit en outre un certains nombre de modèles de mobilité pré-définis, tels que le très classique modèle de *Random Waypoint* (dans lequel les mobiles évoluent en ligne droite et à vitesse constante vers une cible choisie au hasard, puis se fixent une autre cible à atteindre de la même façon), mais aussi des modèles plus « réalistes » de mobilité urbaine (*e.g.* circulation dans un centre commercial, dans les rues d'une agglomération, etc.). Dans le cadre du projet SARAH, nous avons interfacé les plates-formes DoDWAN et MADHOC, et avons en outre apporté de nouvelles fonctionnalités au simulateur MADHOC, telles que la possibilité de simuler la volatilité des hôtes mobiles (*i.e.* le fait que certains équipements, et notamment les PC portables et PDAs, ne sont pas nécessairement allumés à tout instant).

3.3.6 Principaux résultats et perspectives

Résultats. Nos travaux portant sur le support de la communication dans les réseaux mobiles ad hoc discontinus ont débuté au cours du projet MASC *(Mobile Adaptive Software Components)*, dans lequel s'inscrivait notamment le travail de thèse d'Hervé Roussain (cf. annexes A et B). Les travaux réalisés à cette période nous ont permis de jeter les bases d'un modèle de communication reposant sur des échanges opportunistes entre terminaux mobiles, et de bâtir à l'aide de ce modèle un système pair-à-pair dédié au déploiement de composants logiciels (ce système est décrit en détails dans le paragraphe 4.2). Les publications listées ci-dessous portent sur le modèle de communication défini à cette époque.

- Frédéric Guidec and Hervé Roussain. *Asynchronous Document Dissemination in Dynamic Ad Hoc Networks.* In J. Cao et al., editor, Second International Symposium on Parallel and Distributed Processing and Applications (ISPA'04), volume 3358 of LNCS, pages 44-48, Hong-Kong, China, December 2004. Springer Verlag. [82]
- Frédéric Guidec and Hervé Roussain. *Dissémination asynchrone d'information en mode peer-to-peer dans les réseaux ad hoc.* In Premières journées francophones : mobilité et ubiquité (UbiMob 2004), pages 207-210, Nice, France, June 2004. Cépaduès Editions. [83]
- Hervé Roussain and Frédéric Guidec. *A Peer-to-Peer Approach to Asynchronous Data Dissemination in Ad Hoc Networks.* In International Conference on Pervasive Computing and Communications (PCC'04), pages 799-805, Las Vegas, Nevada, USA, June 2004. CSREA Press. [151]

Les travaux entrepris au cours du projet MASC dans le domaine de la communication dans les MANETs discontinus nous ont ensuite permis de définir les lignes directrices du projet ANR SARAH, qui se poursuit encore à ce jour, et dont l'un des thèmes d'activité porte spécifiquement sur la problématique de la communication dans ce type de réseau (cf. annexe A). Le travail de thèse de Julien Haillot s'inscrit dans cet axe thématique (cf. annexe B). En poursuivant les travaux initiés au cours du projet MASC, nous avons notamment pu raffiner le protocole de dissémination basée contenus décrit dans le paragraphe 3.3 en y intégrant le support du routage immédiat multi-sauts (*i.e.* diffusion via des relais multi-points, et routage en mode *source routing*) qui n'était pas présent dans les versions initiales de notre protocole. Les travaux menés dans ce domaine dans le cadre du projet SARAH ont fait l'objet des publications suivantes.

- Frédéric Guidec. *Communication dans les réseaux mobiles ad hoc discontinus.* La Lettre Techniques de l'Ingénieur, (8) :3-4, July 2007. Article invité. [84]
- Frédéric Guidec and Yves Mahéo. *Opportunistic Content-Based Dissemination in Disconnected Mobile Ad Hoc Networks.* In International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007), pages 49-54, Papeete, French Polynesia (Tahiti), November 2007. IEEE Press. [85]
- Julien Haillot and Frédéric Guidec. *A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks.* In IEEE 22nd International Conference on Advanced Information Networking and Applications (AINA'08), pages 188-195, Okinawa, Japon, March 2008. [88]
- Julien Haillot and Frédéric Guidec. *Communication « basée contenus » dans les réseaux mobiles ad hoc discontinus*. In 8e Conférence Internationale sur les NOuvelles TEchnologies de la REpartition (NOTERE'08), Lyon, juin 2008. À paraître. [87]

Perspectives. Les perspectives de notre travail dans le domaine de la communication dans les réseaux mobiles ad hoc discontinus impliquent l'élargissement de notre activité afin de couvrir à terme :

- une gamme d'environnements diversifiée (*e.g.* réseaux de capteurs mobiles, réseaux véhiculaires, etc.) ;
- une gamme d'interfaces de communication variée (*e.g.* Wi-Fi, Bluetooth, Zigbee, WiMAX, etc.) ;
- des scénarios de déploiement variés (*e.g.* réseaux mobiles ad hoc « pur », réseaux impliquant des équipements fixes ou « infostations », réseaux hybrides combinant terminaux mobiles et infrastructure fixe, etc.) ;
- des modes d'acheminement variés (*e.g.* routage par délégation, routage par la source, routage *geocast*, communication de « groupe » ou « tribale[12] », etc.) ;

Pour faciliter cet élargissement de notre champ d'activité, l'intergiciel DODWAN dont nous avons entrepris le développement dans le cadre du projet SARAH devrait notamment évoluer afin d'adopter une forme modulaire favorisant l'intégration de « composants » assurant chacun l'une des fonctions élémentaires de la communication (*e.g.* gestion du cache, perception du voisinage de la part des hôtes

[12] L'expression imagée « communication tribale » désigne ici aussi bien les communications pouvant être réalisées entre collégiens ou lycéens dans une cour de récréation, entre collègues dans une entreprise, entre membres d'une même famille, etc. Plus formellement, il s'agit de communications réalisées au sein d'une population relativement « fermée » et dans laquelle les contacts sont assez fréquents. Dans une telle population des communications peuvent être réalisées en exploitant de manière opportuniste les contacts occasionnels entre membres de la population, et en utilisant chaque membre comme « porteur » des messages à transmettre à tout ou partie de la population.

mobiles, gestion du relais multi-sauts immédiat ou différé entre hôtes voisins, support de schémas d'interaction événementiels ou transactionnels entre hôtes mobiles, etc.). L'idée générale est en effet de faire évoluer cet intergiciel afin qu'il puisse à terme être utilisé comme une sorte de « boîte à outils » polyvalente, facilitant la mise en œuvre, l'intégration, et l'exploitation effective de nouveaux moyens de communication dédiés aux équipements mobiles.

Dans le cadre du travail de thèse de Julien Haillot (cf. annexe B), il est notamment prévu d'interfacer à court terme l'intergiciel *DoDWAN* avec des équipements de type PR4G (Postes Radio de 4^e Génération). Ces équipements développés par THALES/Communications sont des postes émetteurs-récepteurs VHF/FM fortement sécurisés, utilisés notamment par les forces armées françaises. Ils supportent à la fois la communication en phonie et la transmission de données via une pile TCP/IP standard. Contrairement aux interfaces de type Wi-Fi ou Bluetooth qui permettent de disposer de liaisons à hauts débits sur de très courtes distances (*i.e.* au plus quelques centaines de mètres), les équipements PR4G permettent quant à eux d'établir des liaisons à bas débits sur des distances de plusieurs dizaines de kilomètres. En portant l'intergiciel DoDWAN sur une plate-forme d'expérimentation PR4G disponible au CELAR[13], nous souhaitons notamment évaluer le comportement de notre protocole de dissémination « basée contenus » sur un réseau ad hoc de grande envergure impliquant toutefois des débits faibles à modérés.

[13]CELAR : Centre d'Électronique de l'Armement situé à Bruz, à proximité de Rennes.

Chapitre 4

Applications et services applicatifs pour terminaux mobiles communicants

4.1 Introduction

Dans le chapitre 3 j'ai présenté divers travaux réalisés par la communauté scientifique, et notamment par l'équipe CASA, dans le domaine de la communication dans les réseaux mobiles ad hoc (MANETs) discontinus. Bien que beaucoup de ces travaux récents visent à développer des protocoles permettant d'assurer les communications dans ce type de réseaux, rares sont ceux qui intègrent une réflexion sur la conception et la mise en œuvre de services applicatifs capables d'exploiter effectivement de tels protocoles. Pourtant, les caractéristiques mêmes de la communication opportuniste tolérant les délais impliquent que les applications exploitant ce type de communication soient capables de tolérer des transmissions asynchrones. En effet, ainsi que l'a relevé Kevin Fall dans [47] : *"Asynchronous communication can be supported in either connected networks, or occasionally-connected (intermittent) networks. The converse is not true: synchronous communication can only be supported in networks that can reasonably be considered to be always connected."*

Des articles récents recensent un certain nombre de services applicatifs susceptibles de pouvoir être mis en œuvre dans des réseaux discontinus [157, 96, 123, 168]. Des services tels que le courrier électronique (Email) et la distribution d'informations ou d'événements y sont présentés comme étant des candidats évidents, dans la mesure où ils peuvent a priori tolérer des conditions dans lesquelles la connectivité de bout en bout entre utilisateurs — ou entre systèmes communicants — n'est pas assurée. Il est pourtant intéressant de noter que, bien qu'un service tel que le courrier électronique soit effectivement un service intrinsèquement asynchrone, sa mise en œuvre repose traditionnellement sur une architecture et des protocoles qui ne se satisfont guère d'une connectivité partielle. En effet, cette mise en œuvre s'effectue habituellement selon le modèle client-serveur, à l'aide de protocoles tels que SMTP, POP, et IMAP. Or le modèle client-serveur suppose que les serveurs soient aisément accessibles depuis les clients. En outre les trois protocoles applicatifs précédemment cités sont tous les trois bâtis au dessus du protocole de transport TCP, qui nécessite lui-même une connectivité stable et de bout en bout entre clients et serveurs. Ainsi, même un service tel que celui du courrier électronique doit être sérieusement reconsidéré si l'on souhaite pouvoir le mettre en œuvre dans un réseau à connectivité partielle. Plusieurs articles récents proposent donc d'utiliser des protocoles de routage *unicast* tolérant les délais pour mettre en œuvre des services de messagerie capables de fonc-

tionner dans un MANET discontinu. [96] et [168] s'intéressent notamment à la mise en œuvre d'un service de courrier électronique dans des environnements hybrides combinant réseau mobile ad hoc discontinu et réseau d'infrastructure traditionnel (*i.e.* Internet). Les approches proposées impliquent, d'une part, de déployer des passerelles dédiées capables de faire le lien entre l'univers déconnecté du réseau MANET et l'univers connecté d'Internet. D'autre part, des protocoles de routage *unicast* capables de fonctionner dans un MANET discontinu doivent être utilisés pour supporter les liaisons entre les terminaux mobiles et les passerelles en question. D'autres articles considèrent le déploiement d'un service de courrier électronique dans un environnement dépourvu de tout réseau d'infrastructure, c'est-à-dire lorsque le réseau est exclusivement constitué de terminaux mobiles. L'approche consiste alors à développer un système de messagerie purement pair-à-pair, fonctionnant sur la base d'une collaboration entre terminaux mobiles, sans jamais avoir recours à des serveurs de messagerie dédiés.

[122] propose une architecture de diffusion d'informations (de type « journaux électroniques »), dans laquelle des équipements fixes — a priori reliés à Internet — font office de sources d'information auprès desquelles des terminaux mobiles peuvent se procurer les informations en question. Ces informations sont ensuite relayées de proche en proche lorsque les terminaux mobiles entrent en contact les uns avec les autres.

[123] décrit un système dit de « *podcasting* sans fils » permettant à des utilisateurs équipés de terminaux communicants de s'échanger des contenus typés (tels que de petits fichiers audio ou vidéo). Pour ce faire chaque utilisateur peut inscrire son propre terminal à un certains nombre de « canaux » thématiques, dans lesquels il lui sera ensuite possible de « publier » des contenus (c'est-à-dire les mettre à disposition de la communauté), tout en recevant de même les contenus publiés dans les mêmes canaux par d'autres utilisateurs. Les échanges entre terminaux mobiles s'effectuent de manière opportuniste, au gré des déplacements des utilisateurs. Il est intéressant de noter que la maquette de démonstration de la plate-forme intergicielle DoDWAN (présentée brièvement dans le paragraphe 3.3.5) offre aux utilisateurs un service tout à fait similaire à celui présenté dans [123], la notion de « canal thématique » étant simplement remplacée, dans notre maquette, par la notion de « groupe d'utilisateurs ».

À l'instar des divers travaux qui viennent d'être évoqués, les travaux menés par l'équipe CASA intègrent une réflexion sur les services pouvant être mis en œuvre dans un environnement de type MANET discontinu. Les paragraphes qui suivent présentent deux des axes que nous avons suivis ces dernières années (le premier dans le cadre de la thèse d'Hervé Roussain, le second dans le cadre de celle de Julien Haillot) afin d'apporter des éléments de réponse à la problématique générale du support de services applicatifs dans les MANETs discontinus. Le paragraphe 4.2 présente un système que nous avons conçu afin de supporter le déploiement sur des terminaux mobiles d'applications conçues par assemblage de composants logiciels. Le paragraphe 4.3 présente quant à lui un service de discussion similaire dans son principe au système Usenet, mais fonctionnant en mode pair-à-pair. Ces deux services se caractérisent bien sûr par leur aptitude à fonctionner sur des terminaux mobiles interagissant exclusivement en mode ad hoc, en l'absence de tout réseau d'infrastructure. Ils utilisent tous deux les fonctionnalités offertes par l'intergiciel de communication DoDWAN pour supporter les interactions opportunistes entre terminaux évoluant de manière non planifiée.

4.2 Déploiement de composants logiciels sur des terminaux mobiles communicants : une approche coopérative

4.2.1 Motivation

Les systèmes logiciels devenant de plus en plus complexes, une approche encouragée pour maîtriser cette complexité consiste à concevoir ces systèmes en termes d'assemblages de composants logiciels. Les composants logiciels peuvent être vus comme des « briques » élémentaires réutilisables et remplaçables destinées à remplir une fonction clairement définie au sein d'une application logicielle [6, 163]. Les travaux menés dans le domaine général du génie logiciel ont contribué à identifier un « cycle de vie » du logiciel regroupant les phases d'analyse, de conception, de production, de test, et enfin de déploiement du logiciel. Cette dernière phase, qui englobe toutes les activités menées *après* le développement du logiciel, couvre en elle-même un processus suffisamment complexe et important pour constituer un domaine de recherche à part entière. Des travaux tels que [25] et [124] ont permis d'identifier les phases élémentaires sur lesquelles repose ce processus.

Dans le cadre du projet MASC (cf. annexe A), nous nous sommes intéressés à la problématique générale du déploiement d'applications bâties par assemblage de composants logiciels sur des terminaux mobiles capables de communication ad hoc.

Vision traditionnelle du schéma de déploiement de composants logiciels. Dans la vision traditionnelle du « cycle de vie » des composants logiciels, on considère en général que le fait de mettre des composants à disposition d'utilisateurs potentiels consiste simplement à entreposer ces composants dans un espace de stockage, que l'on désigne communément par le terme évocateur de « dépôt de composants[1] » (*component repository*). On suppose en outre que les dépôts de composants sont censés être connus de tous, et être aisément accessibles pour quiconque souhaitera y prélever des composants afin de bâtir une application logicielle (excepté bien sûr lorsque des contraintes de sécurité obligent par exemple à restreindre l'accès à certains composants à des utilisateurs autorisés).

Les études portant sur la mise en œuvre et l'exploitation effectives des dépôts de composants se concentrent pour la plupart sur la résolution des problèmes posés par la gestion même de ces dépôts, en couvrant des aspects tels que la recherche dans un dépôt de composants présentant certaines caractéristiques, l'équilibrage de charge entre plusieurs dépôts « miroirs » capables de fournir a priori les mêmes composants, la prise en compte des dépendances entre composants, la gestion des versions multiples d'un même composant *(versioning)*, etc. En revanche, l'acheminement des composants depuis ces dépôts vers la plate-forme de déploiement visée est en général considéré comme ne posant pas de problème particulier. Ainsi la plupart des travaux menés dans ce domaine reposent sur l'hypothèse — en général non formulée — selon laquelle l'obtention d'un composant se ramène, une fois identifié le dépôt capable de fournir ce composant, à une simple opération de téléchargement du composant considéré, ou plus exactement d'un « paquet » contenant ce composant. On suppose donc qu'un programme serveur est mis en œuvre au niveau de chaque dépôt, et qu'il suffit à la plate-forme sur laquelle une application doit être déployée de se comporter en tant que cliente vis-à-vis d'un ou de plusieurs serveurs pour se procurer des paquets contenant les composants nécessaires à ce déploiement.

Ce schéma dans lequel le déploiement de composants implique le téléchargement de ces composants depuis un ou plusieurs dépôts distants se prête bien aux environnements dans lesquels la

[1] L'expression « étagères à composants » est également utilisée dans la littérature.

connectivité entre plates-formes cibles et dépôts de composants ne pose pas de problème particulier. Internet constitue bien sûr l'environnement idéal de ce point de vue, et un modèle d'interaction de type client-serveur est parfaitement adapté à ce type d'environnement. Il existe d'ailleurs des dépôts accessibles via des protocoles standards tels que HTTP et FTP. Ces dépôts se présentent souvent comme de simples sites Web, mais offrent parfois des fonctionnalités de recherche avancée. Certains d'entre eux sont dédiés à un type de composants particulier (l'Alliance OSGi gère par exemple un dépôt de *bundles* OSGi[2]), alors que d'autres hébergent n'importe quel type de composants (le dépôt *ComponentSource*[3] héberge indifféremment des composants .NET, ActiveX, Java, C++, etc.).

Des plates-formes supportant l'assemblage et l'exécution d'applications à base de composants existent par ailleurs pour différents modèles de composants. Ces diverses plates-formes sont en général capables de se comporter en tant que clientes vis-à-vis de serveurs hébergeant des dépôts de composants. Par exemple, les plates-formes *Java Web Start* [162] et *Apache Maven* [164] supportent toutes deux le déploiement, la mise à jour, et l'exécution d'applications Java. Les plates-formes *JBoss*[4] et *JOnAS*[5] sont quant à elles dédiées à l'exécution de composants *JavaBeans*. La plate-forme *Felix*[6] supporte l'exécution de services OSGi, et la plate-forme *Julia* l'exécution de composants *Fractal* [14].

On peut noter que le principe consistant à mettre à disposition des composants logiciels sur des dépôts aisément accessibles a depuis longtemps été adopté dans la communauté des développeurs et utilisateurs des systèmes d'exploitation Linux. Bien qu'on puisse parfois émettre certaines réserves — avec quelque raison —- quant au fait d'utiliser le terme de « composants logiciels » pour qualifier le contenu des paquets disponibles sur les dépôts Linux, l'existence même de ces dépôts et la quantité de paquets qu'ils proposent démontrent l'intérêt qu'il y a à découper un système logiciel complexe en entités plus simples et composables. Ainsi, la distribution Debian GNU/Linux[7] est à ce jour constituée d'environ 20.000 paquets logiciels, représentant un volume compressé de l'ordre de 18 Go (pour environ 50 Go de code et données une fois ces paquets installés sur une plate-forme cible). Ces paquets contiennent à la fois les éléments constitutifs du système d'exploitation lui-même, et des éléments complémentaires (bibliothèques, code source, code exécutable, données, etc.) permettant d'installer une multitude de programmes applicatifs sur une machine cible. L'installation d'une distribution Linux tout comme sa mise à jour peuvent s'effectuer en ligne, en téléchargeant les paquets nécessaires depuis un ou plusieurs serveurs hébergeant des dépôts « miroirs », répartis sur la planète.

Dans certains travaux récents il a été proposé de s'éloigner du modèle client-serveur sur lequel repose le schéma de déploiement traditionnel. Ainsi, SoftwareDock [91] et Tacoma [161] utilisent tous deux des agents mobiles comme support au transport des composants, ces agents circulant librement entre producteurs et consommateurs de composants. Cette approche est censée permettre d'équilibrer la charge entre fournisseurs de composants, et d'aider à supporter d'éventuelles pannes de ces fournisseurs. Dans la même optique, [51] propose d'organiser les équipements participant au déploiement de services OSGi en réseau pair-à-pair structuré sur la base de clés de hachage. Bien que ces divers travaux s'éloignent quelque peu du modèle client-serveur traditionnel, ils nécessitent cependant un environnement dans lequel les échanges entre fournisseurs et consommateurs de composants peuvent s'appuyer sur un réseau connexe et relativement stable.

[2] `http://www2.osgi.org/Repository`
[3] `http://www.componentsource.com`
[4] `http://www.jboss.org`
[5] `https://wiki.objectweb.org/jonas`
[6] `http://felix.apache.org`
[7] Il existe à ce jour une centaine de distributions distinctes de Linux, dont certaines sont dédiées aux systèmes temps-réels, d'autres aux systèmes embarqués, d'autres encore au multimédia, etc.

CORBA-*LC* définit la notion de « composant léger CORBA », et l'article [158] énumère un certain nombre d'exigences en termes de conception et de mise en œuvre pour assurer le déploiement de tels composants légers. Cet article suggère notamment que les composants devraient être déployés en adoptant un modèle de « réseau de pairs », dans lequel l'ensemble du réseau pourrait jouer le rôle de dépôt distribué dans lequel des ressources (et notamment des composants légers) peuvent être entreposées. Cependant, bien qu'il soit précisé dans l'article [158] en question que les défaillances de certains noeuds du réseau et des liens entre noeuds du réseau devraient être supportées, il semble que seul le déploiement dans un environnement quasi-stable reposant sur un réseau d'infrastructure soit réellement considéré dans cet article.

Déploiement de composants sur des terminaux mobiles communicants. Lorsque des applications à base de composants logiciels doivent être déployées sur des terminaux mobiles, le problème ne se pose pas dans les mêmes termes selon que ces terminaux sont capables d'accéder à un réseau d'infrastructure ou non. Lorsqu'un terminal peut accéder à un réseau d'infrastructure (*e.g.* Internet), le déploiement de composants sur ce terminal peut se faire à peu près dans les mêmes conditions que pour un équipement non mobile. Si la connectivité vers le réseau d'infrastructure — et donc l'accès à des dépôts de composants fixes — est sporadique, le propriétaire d'un terminal mobile doit alors exploiter les plages de connectivité pour installer sur ce terminal de nouvelles applications, ou mettre à jour des applications existantes. Dans ce scénario le modèle de déploiement basé sur une relation de type client-serveur demeure suffisant.

Le déploiement de composants sur des terminaux mobiles interagissant en mode ad hoc a fait l'objet des travaux présentés dans [5]. Cet article présente la plate-forme JDRUMS, qui s'appuie sur des équipements dédiés appelés « magasins » (ou JSTORES) sur lesquels tournent des programmes serveurs de composants. Ces programmes doivent s'enregistrer auprès d'un système de registre JINI afin d'être localisables par les équipements sur lesquels des applications peuvent être déployées. Bien que le déploiement de logiciel sur des équipements mobiles soit explicitement ciblé dans ces travaux, les magasins JSTORES sont supposés être stables et accessibles à tout instant depuis les terminaux mobiles. Cette approche se prête donc à un déploiement dans des réseaux ad hoc connexes, et dans lesquels un mécanisme de routage dynamique peut être utilisé afin d'assurer une connectivité de bout en bout entre systèmes clients et systèmes serveurs.

De nombreux travaux portant sur la tolérance aux déconnexions [100, 134] peuvent contribuer à aider au déploiement de composants dans des environnements temporairement déconnectés. Cependant ces travaux visent essentiellement à proposer des façons d'émuler des opérations connectées dans des environnement temporairement déconnectés. Elles se prêtent donc mieux au déploiement dans des réseaux sans fils à couverture discontinue (de type *hot spot*) qu'au déploiement dans des réseaux ad hoc discontinus proprement dits.

Le modèle SATIN [169] associe un modèle de composant et un modèle de déploiement, et vise spécifiquement le déploiement de composants logiciels dans les réseaux ad hoc discontinus. Il se base sur l'utilisation de primitives d'auto-organisation de la mobilité du code, à partir desquelles il est possible de construire des systèmes mobiles auto-organisés. Le support à l'acheminement des composants est présenté comme permettant la reconfiguration d'architectures de composants par le biais de la migration de code. Ce support repose sur des mécanismes offerts par EMMA [135], un intergiciel permettant la dissémination « épidémique » de messages, très semblable dans son principe à l'intergiciel DODWAN que nous avons nous-même développé (cf. paragraphe 3.3.5).

Dans le cadre du projet MASC (cf. annexe A) nous nous sommes intéressés aux problèmes spé-

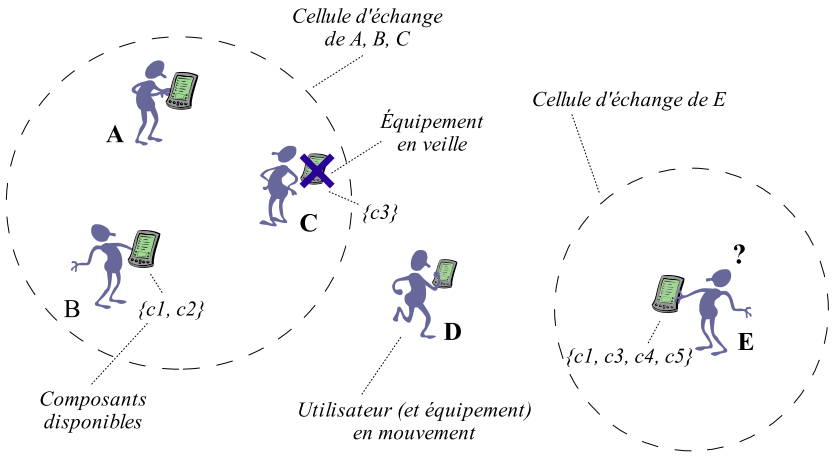


FIG. 4.1 – Illustration du déploiement de composants dans un réseau mobile ad hoc discontinu

cifiques posés par le déploiement d'applications à base de composants logiciels sur des terminaux mobiles interagissant exclusivement en mode ad hoc, en l'absence de toute possibilité d'accès à un réseau d'infrastructure. Dans notre étude nous nous sommes principalement focalisés sur un scénario dans lequel des individus équipés de terminaux mobiles (*e.g.* ordinateurs portables, assistants numériques personnels, etc.) peuvent trouver intérêt à coopérer afin de partager des composants logiciels. Cependant d'autres scénarios sont tout aussi envisageables, comme par exemple un scénario dans lequel des terminaux embarqués sur une flotte de véhicules doivent procéder à la mise à jour de tout ou partie de leur base logicielle, et sont contraints de procéder à cette mise à jour sur la base d'échanges opportunistes entre terminaux voisins.

La figure 4.1 illustre le type de scénario que nous avons considéré dans ce projet. Dans ce scénario des utilisateurs équipés de terminaux capables de communication en mode ad hoc évoluent librement. L'ensemble des terminaux constitue ainsi un réseau mobile ad hoc discontinu (MANET discontinu), dont les caractéristiques principales ont été mises en relief dans le paragraphe 3.2.2.

Dans un scénario semblable à celui présenté dans la figure 4.1, le propriétaire d'un terminal souhaitant déployer sur cet équipement une nouvelle application ne peut se comporter en simple client, se procurant les composants nécessaires auprès de serveurs dédiés à l'hébergement de dépôts de composants. En effet, en l'absence de tout réseau d'infrastructure le recours à un serveur fixe n'est pas envisageable. En outre aucun terminal n'est a priori suffisamment stable, suffisamment accessible depuis tous les autres terminaux, et enfin suffisamment doté en ressources pour qu'on puisse lui confier le rôle de serveur de composants pour l'ensemble de la communauté.

En revanche, chaque terminal étant censé pouvoir proposer à son propriétaire de charger et lancer un certain nombre d'applications produites par assemblage de composants, on peut considérer qu'un terminal aura à maintenir un dépôt local dans lequel seront entreposés ces composants. Dès lors il devient envisageable de substituer au modèle client-serveur évoqué précédemment un modèle d'interaction pair-à-pair, dans lequel chaque terminal mobile est capable de coopérer avec les terminaux du voisinage, fournissant à ses voisins des copies des composants entreposés dans son dépôt local, et se procurant auprès d'eux les composants qui lui font défaut.

Considérons l'ensemble de terminaux reproduit dans la figure 4.1, et supposons que le propriétaire du terminal *A* a pris la décision d'y déployer une application nécessitant l'assemblage de composants *c1*, *c2*, et *c3*. Dans l'exemple représenté le terminal *A* va pouvoir se procurer les composants *c1* et *c2* auprès du terminal *B* voisin. En revanche le composant *c3* n'est a priori pas accessible immédiatement : ce composant est en effet bien entreposé sur le dépôt local du terminal *C*, mais ce terminal a temporairement été mis en veille par son propriétaire.

Il est intéressant de noter dans cet exemple que le fait que le déploiement de l'application demandée par le propriétaire du terminal *A* ne puisse être réalisée immédiatement n'implique aucunement que ce déploiement ne pourra être mené à plus ou moins long terme. Ainsi, il suffirait par exemple que le terminal *B* soit réactivé dans le scénario précédent, ou encore que le propriétaire de *A* se déplace jusqu'à entrer dans la cellule d'échange du terminal *E*, pour que le composant *c3* manquant puisse enfin être collecté par le terminal *A*, et que l'installation de l'application sur ce terminal puisse donc être menée à terme.

Ainsi, alors que le déploiement de composants peut, dans un réseau d'infrastructure, être perçu comme une opération quasi instantanée du point de vue des utilisateurs (si l'on fait abstraction des délais de transmission et des temps de réponse des équipements impliqués), la même opération doit être réalisée de façon progressive dans un réseau ad hoc discontinu, chaque terminal mobile exploitant de manière opportuniste les contacts fugitifs avec d'autres terminaux pour obtenir d'eux les composants qui lui manquent, tout en leur fournissant le même service en échange.

La section suivante présente une vue d'ensemble de la plate-forme CODEWAN *(COmponent DEployment in Wireless Ad hoc Networks)*, que nous avons développée dans le cadre du projet MASC afin de supporter le déploiement d'applications à base de composants logiciels sur des terminaux mobiles capables de communiquer en mode ad hoc. Cette plate-forme permet notamment de gérer sur chaque terminal un dépôt dans lequel des composants peuvent être entreposés, et elle offre des mécanismes grâce auxquels des terminaux voisins vont pouvoir s'échanger des copies de composants, mais aussi rechercher des composants spécifiques dans leur voisinage, ou encore annoncer à leur voisinage quels sont les composants dont ils disposent localement.

Une différence notable entre la plate-forme CODEWAN et des produits évoqués précédemment tels que les plates-formes SATIN et CORBA-$\mathcal{LC}$ est que CODEWAN est a priori plus polyvalente que ces dernières. En effet la plate-forme SATIN ne peut déployer que des composants SATIN, de la même manière que la plate-forme CORBA-$\mathcal{LC}$ ne peut déployer que des composants CORBA. Par contraste CODEWAN peut assurer le déploiement de paquets logiciels dont le contenu (*i.e.* le type de composant contenu dans chaque paquet) n'est pas imposé, CODEWAN laissant à des exécutifs distincts le soin d'assembler et de faire tourner les applications bâties en assemblant ces composants.

4.2.2 Vue d'ensemble de la plate-forme CODEWAN

La figure 4.2 offre une vue d'ensemble de la plate-forme CODEWAN. Cette plate-forme est construite suivant un modèle en couches. La couche supérieure offre un environnement d'exécution pour les applications considérées. Comme expliqué plus haut la plate-forme CODEWAN n'est pas spécifiquement liée à un type d'exécutif particulier, ni même à un modèle de composants particulier. En fait, elle assure le déploiement en mode coopératif de *paquets logiciels*, quel que soit le type de *composants* contenus dans ces paquets. Plusieurs exécutifs peuvent donc co-exister au niveau supérieur de la plate-forme CODEWAN. À ce jour la plate-forme peut ainsi s'interfacer avec les exécutifs suivants :

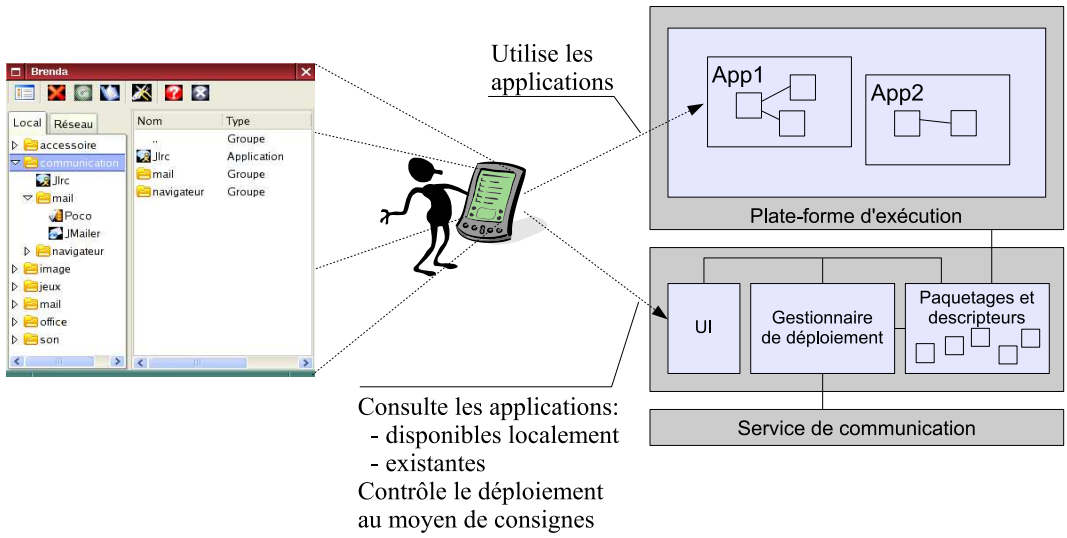


FIG. 4.2 – Vue d'ensemble de la plate-forme de déploiement coopératif CODEWAN

– un JRE *(Java Runtime Environment)* permettant l'exécution d'applications Java standard.
– la plate-forme JAMUS *(Java Accommodation for Mobile Untrusted Software)* que nous avons également développée et qui est plus particulièrement dédiée à l'hébergement sécurisé d'applications Java non sûres (cf. paragraphe 2.4).
– la plate-forme JULIA [16], qui implémente le modèle de composants Fractal, et supporte donc l'exécution d'applications bâties selon ce modèle.
– la plate-forme OSCAR [90], qui supporte l'exécution de services bâtis selon le modèle OSGi.

D'autres types d'exécutifs pourraient bien sûr être intégrés à l'avenir à cette plate-forme, comme par exemple un exécutif capable de supporter des applications réalisées par assemblage de composants CCM [131], un exécutif supportant l'exécution de scripts (de type *Perl* ou autre), etc.

La couche inférieure du modèle présenté dans la figure 4.2 doit permettre les interactions entre terminaux mobiles évoluant au sein d'un réseau MANET discontinu. Dans les toutes premières versions de la plate-forme CODEWAN, cette couche mettait en œuvre des services spécifiques destinés à assurer les communications dans un réseau mobile ad hoc discontinu [82, 83, 151]. Ces services ont par la suite été remaniés, et intégrés dans la plate-forme de communication DODWAN, telle qu'elle a été présentée dans le paragraphe 3.3.5 de ce document. À ce jour la couche inférieure de l'architecture présentée dans la figure 4.2 repose donc sur l'intergiciel DODWAN, vis-à-vis duquel la plate-forme CODEWAN est perçue comme étant un service applicatif comme un autre.

La couche intermédiaire de notre modèle est décrite en détails dans le reste de ce chapitre. Cette couche a pour fonction principale d'assurer, de manière coopérative, la dissémination de composants logiciels au sein d'une communauté de terminaux mobiles interagissant en mode ad hoc. Elle met en œuvre, sur chaque terminal mobile, un dépôt local dans lequel des composants vont pouvoir être entreposés et mis à disposition des exécutifs qui constituent la couche supérieure du modèle. Elle comprend en outre un gestionnaire de déploiement, dont la fonction première est d'interagir avec les gestionnaires pairs situés sur des terminaux voisins, en utilisant pour ce faire les moyens de communication fournis par la couche inférieure du modèle. Ce gestionnaire de déploiement va donc être capable de

répondre à des requêtes provenant des terminaux voisins, et d'adresser lui-même des requêtes à ces terminaux voisins, ces requêtes se soldant de manière générale par le dépôt ou le prélèvement de composants au niveau du dépôt local. Le gestionnaire de déploiement a également pour rôle d'interagir avec le propriétaire de l'équipement mobile afin de lui présenter une vision cohérente de l'état du dépôt local. Le propriétaire de l'équipement va ainsi disposer à tout instant d'une vision des applications déjà installées localement (une application est considérée comme « installée » lorsque tous les composants requis pour son installation sont présents dans le dépôt local, et que cette application peut donc être chargée et lancée à l'aide de l'un des exécutifs peuplant la couche supérieure de la plate-forme), des applications installables (c'est-à-dire des applications dont tous les composants ne sont pas disponibles localement, mais pourraient éventuellement le devenir si l'utilisateur donnait pour consigne au gestionnaire de déploiement de se procurer les éléments manquants), et enfin des applications en cours d'installation (c'est-à-dire celles pour lesquelles l'utilisateur a formulé une consigne d'installation auprès du gestionnaire de déploiement).

4.2.3 Composants, applications, et paquets

Comme évoqué dans le paragraphe 4.2.1, les phases initiales du déploiement de composants nécessitent leur stockage préalable dans des dépôts, et leur transport à travers le réseau. Pour ce faire les composants sont encapsulés dans des paquets (alias *packages* ou *bundles*), qui constituent donc des unités de stockage et de transport pour les composants. Les paquets peuvent servir à encapsuler, outre du code exécutable, des données requises par une application, la description de l'architecture d'une application (telle que la description d'un assemblage de composants CCM [131], un descripteur d'architecture Fractal [16], etc.), ou encore des informations relatives aux conditions d'installation et de lancement d'une application (description des ressources requises par une application, manifeste indiquant la classe principale d'une application Java, etc.).

Descripteurs de paquets. Dans le système CODEWAN, chaque paquet se voit associer un descripteur, qui apporte des informations sur l'identité du paquet, sur son contenu, etc. Le descripteur de paquet est destiné à être embarqué dans le paquet lui-même. Il peut cependant être aussi extrait de ce paquet, et manipulé indépendamment de celui-ci. Un terminal mobile va ainsi pouvoir transmettre à ses voisins les descripteurs des paquets dont il dispose localement (dans son dépôt), sans pour autant leur transmettre l'intégralité des paquets en question. Un terminal va de même pouvoir s'appuyer sur les informations contenues dans des descripteurs de paquets afin de décider s'il est pertinent pour lui de chercher à se procurer ces paquets.

Les descripteurs de paquets manipulés par la plate-forme CODEWAN sont exprimés à l'aide d'un DSL *(Domain-Specific Language)* que nous avons défini. La figure 4.3 présente ainsi le descripteur d'un paquet encapsulant le composant principal d'une application Java permettant de disposer d'un service de messagerie fonctionnant en mode pair-à-pair.

On peut noter que ce descripteur fait état de dépendances vis-à-vis d'autres paquets. De telles dépendances entre paquets peuvent traduire des dépendances entre les éléments de code (source ou exécutable) embarqués dans les paquets considérés. Elles peuvent également traduire des dépendances entre la description de l'architecture d'une application et les composants constitutifs de cette application (cette description d'architecture et ces composants pouvant être encapsulés dans des paquets distincts). Lorsque le propriétaire d'un équipement mobile donne au gestionnaire de déploiement la consigne de déployer localement une certaine application, le gestionnaire de déploiement peut ainsi

```
<package-descriptor>
  <general-information
    name="JMessager"
    version="1.3"
    provider="Valoria laboratory"
    category="communication/messaging"
    summary="JMessager is a P2P messager"
    type="application/java"  />
  <java-application
    main="masc.jmessager.JMessagerImpl"  />
  <dependencies>
    <required-package
      name="JMessengerUI"  version="1.2"  />
    <required-package
      name="P2PAsyncDissemination"  />
    <optional-package name="AddressBook" version="2.0"  />
  </dependencies>
</package-descriptor>
```

FIG. 4.3 – Exemple de descripteur de paquet caractérisant une application Java

```
<package-descriptor>
  <general-information
  ....
    type="application/fractal"  />
  <fractal-application
    main="masc.jmessager.JMessagerImpl"
    interface="java.lang.Runnable"  />
  ....
</package-descriptor>
```

FIG. 4.4 – Exemple de descripteur de paquet caractérisant une application Fractal

s'appuyer sur les dépendances exprimées dans le descripteur de cette application afin de déterminer quels sont les paquets dont il dispose déjà localement, et quels sont ceux qu'il va lui falloir essayer de se procurer auprès des terminaux voisins.

De manière générale, il y a lieu de distinguer entre un paquet contenant le composant principal (on parle aussi de « composant maître ») d'une application, et un paquet contenant un composant quelconque, susceptible d'être utilisé pour déployer plusieurs applications distinctes. Le descripteur d'un paquet contenant le composant principal d'une application se caractérise par la présence d'un attribut *type*, qui indique le type de l'application considérée, et précise quel est le point d'entrée permettant de lancer cette application. Ainsi, le descripteur de la figure 4.3 caractérise une application Java, qui peut être lancée en invoquant la méthode *main()* de la classe *masc.jmessager.JMessagerImpl*).

La présence de l'attribut *type* permet donc à la plate-forme CODEWAN d'identifier les paquets contenant les composants principaux des applications, et indique en même temps quel type d'exécutif doit permettre de charger et lancer l'application considérée. La figure 4.4 montre, à titre d'exemple, en quoi le descripteur de l'application *JMessager* serait différent du descripteur présenté dans la figure 4.3 s'il s'agissait en fait d'une application Fractal, et non d'une application Java. Un composant Fractal pouvant offrir plusieurs interfaces, on notera que le descripteur précise dans ce cas quelle est

l'interface qu'il convient d'utiliser pour lancer l'application.

4.2.4 Rôle et fonctionnement du gestionnaire de déploiement

Le gestionnaire de déploiement a pour mission d'assurer la gestion d'un dépôt de paquets local, d'interagir avec l'utilisateur afin de recueillir ses consignes vis-à-vis des applications dont il souhaite pouvoir disposer sur son terminal, et enfin de collaborer avec les gestionnaires de déploiement pairs tournant sur d'autres terminaux pour se procurer les paquets qui lui manquent, tout en s'efforçant de les aider à faire de même.

Gestion du dépôt de paquets local. Le dépôt intégré à la plate-forme CODEWAN peut en fait héberger indifféremment des paquets complets, ou bien simplement des descripteurs de paquets. Le gestionnaire de déploiement assure la gestion de ce dépôt, s'efforçant d'y entreposer les paquets nécessaires à l'installation des applications requises par l'utilisateur, et coopérant pour ce faire avec les gestionnaires de déploiement tournant sur d'autres terminaux mobiles.

Un terminal mobile étant en général limité en ressources, le dépôt de paquets mis en œuvre sur ce terminal est lui-même forcément limité en capacité. Le gestionnaire de déploiement peut donc être amené à supprimer certains paquets du dépôt local afin de libérer de l'espace pour y déposer de nouveaux paquets. En règle générale, le gestionnaire de déploiement s'abstiendra cependant de supprimer un paquet, dès lors que celui-ci contient un composant dont la disponibilité est nécessaire pour lancer une application demandée par l'utilisateur. Si par contre le dépôt contient un paquet qui ne sert apparemment à rien (par exemple parce que l'application nécessitant ce paquet a été précédemment « désinstallée » à la demande de l'utilisateur), le gestionnaire de déploiement va s'efforcer de maintenir ce paquet dans le dépôt aussi longtemps que possible, mais finira par le supprimer s'il se trouve obligé de faire de la place pour de nouveaux paquets. Ainsi, en maintenant dans la mesure du possible dans le dépôt local des paquets qui ne présentent pourtant pas d'intérêt immédiat pour l'utilisateur, le gestionnaire de déploiement se trouve en mesure de satisfaire des gestionnaires pairs tournant sur d'autres terminaux mobiles, s'il s'avère que ces gestionnaires sont à la recherche des paquets en question.

Interactions avec l'utilisateur. L'interaction entre l'utilisateur et le gestionnaire de déploiement s'effectue via une interface de contrôle qui permet à l'utilisateur de consulter l'état du dépôt de composants local, et de passer au gestionnaire des consignes relatives à l'installation de nouvelles applications (voire à la désinstallation d'applications existantes).

De manière générale l'utilisateur ne s'intéresse qu'aux *applications* susceptibles d'être déployées sur son terminal, et non aux composants permettant de bâtir ces applications. Comme expliqué dans la section 4.2.3, une application se distingue d'un simple composant par la présence de l'attribut *type* dans son descripteur. Le gestionnaire de déploiement est donc en mesure de présenter à l'utilisateur, à travers une interface graphique ou textuelle, une liste détaillée des applications dont il a connaissance, en précisant pour chacune de ces applications son état actuel. En effet, à tout instant une application dont le descripteur est présent dans le dépôt local de la plate-forme CODEWAN peut être :

- **installée :** une application est dite « installée » localement lorsque cette application a déjà été chargée — ou peut être chargée immédiatement — dans l'un des exécutifs associés à la plate-forme

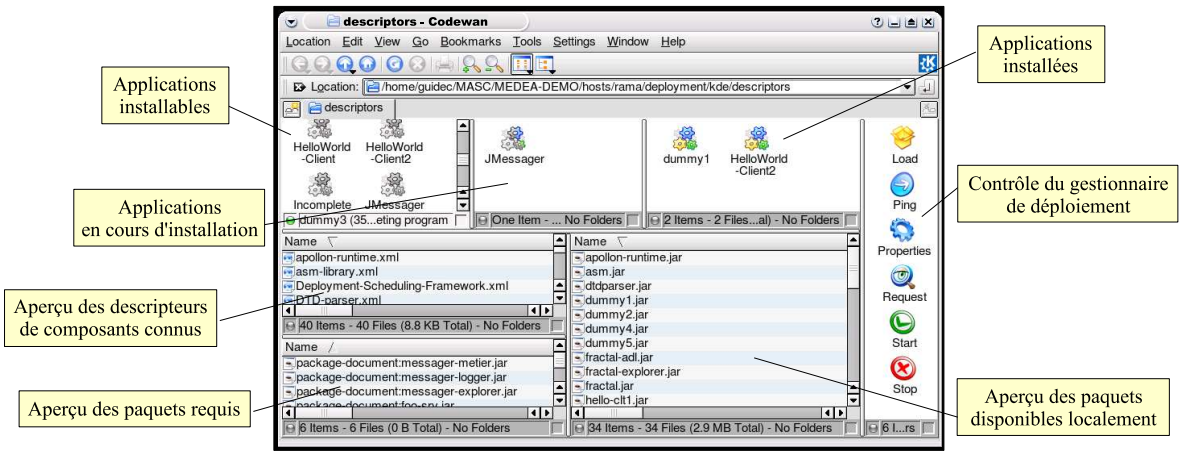


FIG. 4.5 – Interface graphique de la plate-forme CODEWAN

CODEWAN. Cet état implique que tous les paquets nécessaires à ce chargement sont disponibles dans le dépôt local.

- **installable :** une application est dite « installable » lorsque certains des paquets contenant les composants nécessaires à son installation ne sont pas disponibles dans le dépôt local. L'utilisateur peut demander l'installation d'une application de ce type, ce qui a pour effet de donner pour consigne au gestionnaire de déploiement de se procurer tous les paquets manquants auprès de gestionnaires pairs tournant sur d'autres terminaux mobiles.
- **en cours d'installation :** une application est « en cours d'installation » lorsque l'utilisateur a demandé cette installation, mais que tous les paquets nécessaires n'ont pas encore été récupérés par le gestionnaire de déploiement.

L'interface permettant à l'utilisateur de contrôler le gestionnaire de déploiement peut exister sous différentes formes. Par défaut la plate-forme CODEWAN offre une interface textuelle, accessible en ligne de commande. Des variantes graphiques ont également été développées. L'une de ces interfaces a été développée en SWT *(Standard Widget Toolkit)*, et dimensionnée tout spécialement pour pouvoir être utilisée sur l'écran d'un assistant numérique personnel (voir fig. 4.2). Une autre interface a été bâtie en détournant certains des outils graphiques de l'environnement KDE *(K Desktop Environment)* de leur fonction première, ce qui constitue une façon simple de développer une interface rudimentaire à moindres frais (voir fig. 4.5).

En utilisant l'une ou l'autre de ces interfaces de contrôle, un utilisateur de la plate-forme CODEWAN peut :

- voir l'ensemble des applications *installées* sur la plate-forme, et charger et lancer n'importe laquelle de ces applications grâce à l'exécutif approprié, ou en demander la désinstallation ;
- voir l'ensemble des applications *installables* sur la plate-forme, et demander au gestionnaire de déploiement d'initier le processus d'installation de n'importe laquelle de ces applications ;
- consulter l'état des applications *en cours d'installation*, et éventuellement annuler un processus d'installation en cours ;
- charger des paquets logiciels depuis une source quelconque (*e.g.* clé USB, voire serveur distant

dans le cas d'un terminal disposant d'une connectivité temporaire à Internet) et les entreposer dans le dépôt local, ce qui a pour effet de mettre ces paquets à disposition du gestionnaire de déploiement.

Les interfaces de contrôle permettent en outre d'offrir à l'utilisateur une vision détaillée de l'état du dépôt de composants local, et du gestionnaire de déploiement associé. On voit ainsi sur la figure 4.5 que l'utilisateur peut par exemple voir quels sont les descripteurs de paquets disponibles dans le dépôt local (et visualiser le contenu de ces descripteurs s'il le souhaite), quels sont les paquets disponibles localement, quels sont les paquets que le gestionnaire de déploiement s'efforce d'obtenir auprès d'autres terminaux, etc. Ces diverses informations ne sont toutefois pas nécessaires pour une simple utilisation de la plate-forme CODEWAN.

4.2.4.1 Interactions pair à pair entre gestionnaires de déploiement

Un gestionnaire de déploiement ayant reçu pour consigne de se procurer les paquets nécessaires à l'installation d'une certaine application doit, pour ce faire, obtenir les paquets qui lui font défaut auprès d'autres gestionnaires de déploiement tournant sur d'autres terminaux mobiles. Plusieurs modes d'interaction entre gestionnaires de déploiement voisins peuvent être mis en œuvre dans la plate-forme CODEWAN, en s'appuyant sur les services de communication fournis par la couche inférieure de la plate-forme. Chaque gestionnaire de déploiement peut ainsi :

1. *annoncer à son voisinage quels sont les paquets disponibles sur le dépôt local (paquets qu'il est donc en mesure de fournir sur demande)* ;
   L'annonce de la disponibilité de paquets s'effectue en diffusant à l'intention des terminaux voisins un message contenant les descripteurs des paquets proposés. Une telle annonce peut être diffusée périodiquement, ou par exemple lorsqu'un nouveau terminal mobile est détecté dans le voisinage de l'émetteur. Elle peut également être diffusée en réponse à une sollicitation émise par un terminal voisin, comme évoqué dans les items suivants.
2. *recevoir les annonces diffusées par les terminaux voisins, et entreposer les descripteurs ainsi obtenus dans le dépôt local ;*
   Un terminal recevant une annonce diffusée par l'un de ses voisins peut ainsi découvrir l'existence de paquets dont il n'avait pas connaissance jusqu'alors. Les descripteurs de paquets contenus dans cette annonce peuvent être entreposés dans le dépôt local. Le gestionnaire de déploiement est alors en mesure d'enrichir la liste des applications dont l'installation est proposée à l'utilisateur.
3. *chercher dans le voisinage des paquets déjà identifiés, ou des paquets dont les descripteurs répondent à certaines caractéristiques* ;
   La recherche de paquets s'effectue en diffusant une requête invitant les terminaux voisins à annoncer la disponibilité dans leur propre dépôt local de paquets répondant à certains critères, comme par exemple des paquets contenant des composants d'un certain type, des paquets provenant d'un certain producteur, etc. Un gestionnaire de déploiement recevant une requête de ce type va chercher dans son propre dépôt des paquets satisfaisant la demande, et si de tels paquets existent diffuser une annonce contenant les descripteurs de ces paquets (selon la procédure décrite dans l'item 1).
4. *demander à un terminal voisin la transmission d'un paquet précis ;*
   Ayant découvert que l'un de ses voisins peut lui fournir un paquet qui lui fait défaut, un terminal peut adresser à ce voisin une requête l'enjoignant d'émettre le paquet en question. Cette

émission peut être réalisée en mode *unicast* ou *broadcast* selon les modalités de transmission prévues dans les paramètres de configuration de la plate-forme.

5. *recevoir des paquets, et les entreposer dans le dépôt local.*
   Un gestionnaire de déploiement ayant demandé explicitement à l'un de ses voisins d'émettre un paquet est bien sûr censé réceptionner ce paquet, et l'entreposer dans le cache local. Cependant, lorsque les paquets sont diffusés plutôt que d'être émis en mode *unicast*, le gestionnaire de déploiement d'un terminal peut recevoir un paquet même s'il n'en a pas fait lui-même la demande au préalable. Dans ce cas, il lui est possible d'entreposer malgré tout ce paquet dans le dépôt local (si toutefois il reste une place suffisante dans ce dépôt), anticipant ainsi sur une éventuelle demande future de la part de l'utilisateur, et se donnant en outre la possibilité d'aider d'autres terminaux à se procurer ce paquet à l'avenir.

Les opérations élémentaires énumérées ci-dessus permettent de définir une grande variété de schémas de fonctionnement pour les gestionnaires de déploiement mis en œuvre dans la plate-forme CODEWAN.

4.2.4.2 Principales étapes dans le scénario de déploiement d'une application

Découverte de l'existence de nouvelles applications. À tout instant, le gestionnaire de déploiement tournant sur un terminal mobile maintient dans le dépôt local un ensemble de descripteurs d'applications. Pour enrichir cet ensemble, et donc pouvoir proposer un choix d'applications installables aussi vaste que possible à l'utilisateur, le gestionnaire de déploiement peut se contenter de réceptionner les annonces diffusées par ses voisins, et extraire de ces annonces les descripteurs d'applications dont il n'avait pas encore connaissance afin de les entreposer dans le dépôt local. Les gestionnaires de déploiement s'informent donc mutuellement, en continu, des applications existantes.

Une approche alternative — ou complémentaire — pour parvenir au même résultat consiste pour le gestionnaire de déploiement à diffuser périodiquement une requête invitant les terminaux voisins à diffuser dans une annonce les descripteurs des applications dont ils ont connaissance.

D'autre part, on peut noter qu'un gestionnaire de déploiement peut être configuré de manière à ne s'intéresser qu'à certaines catégories d'applications, et filtrer en conséquence les descripteurs qu'il accepte d'entreposer dans le dépôt local.

Lancement d'un processus d'installation d'application. Pour initier le déploiement d'une application sur son terminal mobile, un utilisateur doit a priori utiliser l'interface offerte par le gestionnaire de déploiement, et sélectionner avec cette interface une application marquée comme étant « installable » (*i.e.* certains paquets requis par cette application étant encore absents du dépôt local). Ce scénario implique cependant que l'application en question soit connue du gestionnaire de déploiement, c'est-à-dire que le descripteur de cette application soit déjà présent dans le dépôt local. Il peut aussi arriver que l'utilisateur lui-même ait connaissance de l'existence d'une certaine application, et soit donc en mesure d'indiquer au gestionnaire de déploiement le nom de cette application. Le gestionnaire de déploiement peut alors se mettre en quête du descripteur de cette application et, une fois obtenu ce descripteur, en déduire quels sont les paquets qu'il lui faut encore se procurer pour pouvoir installer l'application localement.

Identification des paquets manquants (exploitation des informations de dépendance). Lorsque le gestionnaire de déploiement reçoit pour consigne d'installer localement une application, il lui faut examiner le descripteur de cette application (après l'avoir obtenu si nécessaire) afin d'identifier les paquets dont cette application dépend.

Selon le principe même de l'assemblage d'applications par composants, plusieurs applications peuvent être bâties en utilisant des composants communs (et donc les mêmes paquets). En identifiant les paquets requis pour installer une nouvelle application, le gestionnaire de déploiement peut constater que certains de ces paquets sont déjà présents dans le dépôt local, ayant déjà servi à installer d'autres applications localement. Il peut également arriver qu'un gestionnaire de déploiement, ayant réceptionné de manière opportuniste des paquets diffusés par des terminaux voisins, ait décidé d'entreposer ces paquets « au cas où » dans le dépôt local. Ce gestionnaire de déploiement recevant de l'utilisateur l'ordre d'installer une nouvelle application peut alors, dans le meilleur des cas, constater que tous les paquets nécessaires à cette installation sont d'ores et déjà disponibles dans le dépôt local. Dans de telles circonstances l'installation de l'application demandée est considérée comme étant immédiatement achevée, et cette application peut immédiatement être chargée dans l'exécutif adéquat.

Dans la plupart des cas, cependant, lorsque l'utilisateur demande l'installation d'une nouvelle application il est probable que le gestionnaire de déploiement va constater que certains paquets au moins lui font défaut. Dès lors l'objectif du gestionnaire de déploiement va être de se procurer les paquets en question auprès d'autres terminaux mobiles. Le gestionnaire de déploiement doit donc maintenir, pour chaque application dont l'installation a été demandée par l'utilisateur, une liste des identifiants des paquets manquants nécessaires à cette installation. Pour des raisons d'efficacité de la mise en œuvre, le gestionnaire de déploiement doit aussi maintenir une structure de données inversée permettant de maintenir, pour chaque paquet manquant, la liste des applications auxquelles ce paquet est nécessaire. Ces diverses structures d'information sont bien bien sûr mises à jour en continu, afin de tenir compte des nouvelles demandes exprimées par l'utilisateur, mais aussi à chaque fois que le gestionnaire de déploiement parvient à obtenir l'un des paquets manquants. Dans ce cas précis, l'identifiant du paquet reçu peut être supprimé de la liste des paquets manquants, mais le gestionnaire de déploiement doit aussi examiner les dépendances exprimées dans le descripteur de ce paquet afin d'identifier d'autres paquets qui lui font éventuellement encore défaut. Si cet examen révèle que des paquets supplémentaires sont effectivement nécessaires à l'installation des applications demandées par l'utilisateur, alors la liste des paquets manquant pour l'application considérée doit être complétée en conséquence.

Recherche des paquets manquants. La recherche des paquets manquants est une opération réalisée de manière proactive, qui implique la diffusion par le gestionnaire de déploiement de requêtes informant les terminaux voisins des paquets qu'il souhaite se procurer. Cette opération peut être réalisée périodiquement, ou être déclenchée par un événement tel que la détection d'un nouveau terminal dans le voisinage.

Une requête contient simplement la liste des identifiants des paquets recherchés. Les terminaux voisins sont censés y répondre en diffusant en retour une annonce indiquant la disponibilité de tout ou partie des paquets recherchés.

On peut noter que, dans la mesure où une annonce de disponibilité de paquets est diffusée sur le médium radio, le gestionnaire de déploiement d'un terminal peut dans certains cas intercepter une annonce l'informant que des paquets qu'il recherche sont disponibles sur un terminal voisin, sans même avoir auparavant diffusé une requête dans ce sens. Certains paquets peuvent donc être localisés sim-

plement en « écoutant » le canal radio, et en interceptant les annonces de disponibilité diffusées sur ce canal. La plate-forme CODEWAN permet de combiner les deux formes de découverte de paquets (*i.e.* découverte proactive et découverte réactive) afin de définir des stratégies de déploiement combinant ces deux approches, ou afin de supporter l'interaction entre des terminaux qui n'adoptent pas tous exactement le même mode de fonctionnement au sein du réseau.

Obtention de paquets manquants. Lorsque le gestionnaire de déploiement découvre qu'un ou plusieurs paquets qu'il recherche sont disponibles sur un terminal voisin, il peut réagir en conséquence en adressant à ce terminal une requête l'invitant à émettre ces paquets sur le médium radio. Comme évoqué plus haut, cette émission peut être effectuée en mode *unicast* (et donc ne bénéficier qu'au terminal demandeur), ou être effectuée en mode *broadcast* (et donc bénéficier potentiellement à d'autres terminaux voisins). L'adoption de l'une ou l'autre approche peut notamment être influencée par des contraintes d'ordre technique, certaines technologies de transmission sans fils (*e.g.* Bluetooth) privilégiant les transmissions point-à-point par rapport aux transmissions en diffusion, alors que d'autres technologies (*e.g.* Wi-Fi) permettent d'exploiter les deux modes de transmission (avec toutefois les contraintes évoquées au paragraphe 3.3).

Terminaison du processus d'installation d'une application. Du point de vue du gestionnaire de déploiement, l'installation d'une application est considérée comme étant achevée lorsque tous les paquets requis pour faire fonctionner cette application sont disponibles dans le dépôt local. Dès lors l'application en question peut être présentée à l'utilisateur comme étant « installée. » L'utilisateur peut donc lancer cette application en demandant qu'elle soit chargée dans l'exécutif approprié.

L'utilisateur peut par ailleurs décider d'annuler une installation en cours. Il lui suffit d'informer le gestionnaire de déploiement qu'il renonce à poursuivre l'installation de l'application considérée. Dans ce cas le gestionnaire de déploiement parcourt l'ensemble des paquets déjà présents dans le dépôt local et, pour chaque paquet requis par l'application considérée le marque comme étant désormais inutilisé (excepté bien sûr si ce paquet est également nécessaire à d'autres applications déjà installées ou en cours d'installation). Un paquet marqué comme inutilisé dans le dépôt local peut immédiatement être supprimé, ou au contraire y être maintenu aussi longtemps que le dépôt n'est pas saturé. En cas de saturation du dépôt, le gestionnaire de déploiement est alors autorisé à supprimer les paquets inutilisés afin de libérer de l'espace pour y entreposer de nouveaux paquets.

4.2.5 Problèmes de sécurité soulevés par l'approche coopérative

Avec la plate-forme CODEWAN nous prônons une approche dans laquelle le déploiement d'applications à base de composants logiciels repose sur un principe de partage spontané entre utilisateurs de terminaux mobiles, considérant que ces utilisateurs pourraient trouver bénéfice à s'échanger des paquets logiciels de gré à gré plutôt que de s'astreindre à récupérer systématiquement ces paquets depuis des serveurs distants (et difficilement accessibles, voire tout bonnement inaccessibles). Cette vision s'accommode assez bien avec la dynamique du développement logiciel *Open Source*, qui autorise en général que du code (source ou exécutable) soit dupliqué à l'envi, et qui n'impose pas que ce code soit systématiquement obtenu auprès d'une quelconque source privilégiée. Le modèle d'échange coopératif de paquets logiciels peut cependant soulever un certain nombre d'inquiétudes légitimes relatives aux problèmes de sécurité posés par cette approche. Ainsi, un utilisateur pourrait être réticent à l'idée de devoir charger et exécuter sur son terminal une application bâtie par assemblage de composants à

l'origine douteuse. Ce problème devrait cependant pouvoir être résolu — au moins partiellement — en adoptant des mécanismes permettant à la fois de certifier l'origine des paquets logiciels (via un procédé de signature électronique) et d'empêcher toute corruption de ces paquets au cours de leur dissémination de proche en proche dans le réseau (via par exemple l'insertion de clés de hachage chiffrées dans ces paquets).

Une autre inquiétude légitime de la part des utilisateurs concerne l'effet que pourrait avoir une application maligne sur les ressources disponibles sur leur terminal mobile, qu'il s'agisse de ressources matérielles susceptibles d'être détournées ou accaparées, ou de ressources logiques (*e.g.* fichiers de données) pouvant être corrompues ou détruites par une application maligne. Une approche permettant de répondre à cette inquiétude consiste à associer à la plate-forme de déploiement coopératif CODEWAN des exécutifs capables d'offrir aux applications des environnements d'exécution sécurisés. Ainsi, la plate-forme JAMUS que nous avons précisément développée dans cette optique (cf. paragraphe 2.4) permet l'hébergement de programmes applicatifs Java considérés comme non sûrs. La combinaison du modèle de déploiement coopératif supporté par la plate-forme CODEWAN avec l'exécutif sécurisé JAMUS a d'ailleurs effectivement été réalisée au cours du projet MASC, et ce travail a fait l'objet de trois publications au cours des années 2004-2005 [119, 120, 121]. Selon le même principe, des exécutifs capables d'héberger d'autres types de composants (*e.g.* composants CCM, Fractal, etc.) de manière sécurisée pourraient être développés et associés à la plate-forme CODEWAN.

4.2.6 Principaux résultats et perspectives

Résultats. Les travaux précédemment évoqués ont principalement été menés entre 2003 et 2005 par Hervé Roussain dans le cadre du projet MASC. Ce projet portait à la fois sur la problématique de la communication dans les réseaux mobiles ad hoc discontinus (nos résultats dans ce domaine ont été évoqués dans le paragraphe 3.3.6), et sur celle du déploiement de composants logiciels dans ce type d'environnements. Le modèle de déploiement coopératif tel qu'il est mis en œuvre dans la plate-forme CODEWAN a notamment fait l'objet des publications suivantes :

– Hervé Roussain and Frédéric Guidec. *Cooperative Component-Based Software Deployment in Wireless Ad Hoc Networks.* In 3rd International Working Conference on Component Deployment (CD'05), volume 3798 of LNCS, pages 1-16, Grenoble, France, November 2005. Springer Verlag. [152]

– Hervé Roussain and Frédéric Guidec. *Déploiement de composants logiciels sur des équipements mobiles communicants : une approche coopérative.* In Journées Composants 2005, pages 83-88, Le Croisic, France, April 2005. [153]

La publication suivante porte plus spécifiquement sur l'intégration d'un support de déploiement coopératif (fourni par la plate-forme CODEWAN) et d'un support d'exécution sécurisé (fourni par la plate-forme JAMUS) afin d'offrir aux utilisateurs de terminaux mobiles la possibilité de se procurer des applications en mode pair à pair, et d'exécuter ensuite ces applications sans compromettre l'intégrité des autres applications et données hébergées par leur terminal.

– Hervé Roussain, Nicolas Le Sommer, and Frédéric Guidec. Towards an Asynchronous Dissemination and a Safe Deployment of Lightweight Programs in Mobile Networks. In IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'05), pages 481-483, Taormina, Italy, June 2005. IEEE CS. [154]

Enfin la publication mentionnée ci-dessous, plus récente, replace ce travail dans le contexte plus large de l'ubiquité numérique, en considérant la problématique du déploiement et de l'hébergement sécurisé d'applications dans des environnements hybrides combinant réseaux ad hoc et zones de type *hot spot*, dans lesquelles l'accès à un réseau d'infrastructure est possible.

– Nicolas Le Sommer, Frédéric Guidec, and Hervé Roussain. *A Context-Aware Middleware Platform for Autonomous Application Services in Dynamic Wireless Networks.* In First International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06), number 9, Nice, France, May 2006. ACM Digital Library. [116]

Perspectives. Les perspectives de ce travail concernent notamment l'application du modèle de déploiement coopératif mis en œuvre dans la plate-forme CODEWAN afin d'assurer l'installation et la mise à jour proactives de composants logiciels sur des flottes de terminaux mobiles. Dans les pages précédentes la plate-forme CODEWAN a été présentée sous un point de vue centré sur l'utilisateur final, celui-ci ayant la possibilité de « tirer » *(pull)* du réseau auquel son terminal participe les composants logiciels dont il souhaite pouvoir disposer sur ce terminal. Le modèle de déploiement coopératif qui sous-tend le fonctionnement de la plate-forme pourrait cependant être tout aussi bien utilisé dans un schéma inverse, dans lequel des composants pourraient être « poussés » *(push)* vers certains terminaux. L'administrateur d'un ensemble de terminaux mobiles (comme par exemple des terminaux embarqués sur une flotte de véhicules) pourrait ainsi déclencher sur l'ensemble de ces terminaux une vague d'installations ou de mises à jour du logiciel, sans avoir pour autant à intervenir successivement et explicitement sur chacun de ces terminaux.

Un autre axe de réflexion méritant d'être exploré plus avant concerne la notion de fonctionnement en mode « dégradé » d'une application. En effet, le déploiement coopératif basé sur des échanges opportunistes entre terminaux mobiles implique que, lorsque l'utilisateur demande l'installation d'une nouvelle application sur son terminal, il lui faille ensuite attendre jusqu'à ce que tous les paquets nécessaires à cette installation aient pu être regroupés dans le dépôt local de ce terminal. C'est là une conséquence directe du schéma de déploiement asynchrone propre à ce modèle de déploiement. Dans certains cas il serait cependant envisageable qu'une application puisse être installée sur un terminal, et donc mise à disposition de l'utilisateur, avant même que tous les composants nécessaires à cette application soient disponibles sur le dépôt local. Une telle perspective présenterait un intérêt tout particulier dans les environnements que nous considérons, dans la mesure où elle permettrait de mettre au plus tôt à disposition d'un utilisateur une application offrant un service dégradé ou incomplet, quitte à ce que cette application soit par la suite complétée (et le service rendu amélioré) au fur et à mesure que les composants nécessaires seront collectés par le terminal.

4.3 Un système de discussion pair-à-pair pour MANETs discontinus

Dans ce paragraphe je présente le fruit d'un travail que nous avons réalisé en vue d'illustrer, à travers le développement d'une application réaliste, l'intérêt du protocole de dissémination « basée contenus » présenté dans le chapitre 3.3. Dans ce cas précis nous nous sommes attachés à développer un système de discussion reposant sur la plate-forme intergicielle DODWAN, et fournissant aux utilisateurs un service inspiré du système Usenet. Comme le système Usenet, ce système doit permettre à des utilisateurs équipés de matériel portable de poster et de recevoir des articles thématiques. Contrairement au système Usenet, cependant, il ne doit pas dépendre d'un ensemble de serveurs dédiés pour

stocker les articles et les fournir à la demande. Il doit s'appuyer exclusivement sur une approche pair à pair, impliquant la collaboration des terminaux mobiles portés par les utilisateurs, de telle sorte que les articles se disséminent dans le réseau en étant stockés, transportés, et échangés de façon opportuniste par ces terminaux.

Description succincte du système Usenet. Usenet (contraction de *User Network*) est un système de discussion non-centralisé, en service depuis les années 1980. Il permet aux utilisateurs de poster librement des articles dans des rubriques thématiques appelées *newsgroups*. Ce système repose sur un vaste ensemble de serveurs inter-connectés, assemblés deux à deux de manière à constituer un réseau de type *overlay* au dessus d'Internet. Chaque serveur héberge un certain nombre de *newsgroups*, ce qui signifie qu'il peut stocker des articles relatifs à ces *newsgroups*, et fournir ces articles sur demande à des serveurs pairs ou aux programmes clients des utilisateurs. Les utilisateurs doivent en effet utiliser un programme client spécifique (ou *newsreader*) afin d'accéder aux articles disponibles sur un serveur. Avec ce programme client, un utilisateur peut souscrire à certains des *newsgroups* proposés par un serveur, poster des articles dans ces *newsgroups*, et enfin récupérer les articles récemment publiés dans chacun de ces *newsgroups*. Les serveurs pairs se synchronisent régulièrement deux par deux pour s'échanger les articles nouveaux. Les interactions entre serveurs pairs, tout comme les interactions entre clients et serveurs, sont réalisées à l'aide du protocole NNTP *(Network News Transfer Protocol)*.

Vers un système de discussion « à la Usenet » pour MANETs discontinus. Le système Usenet a été conçu de manière à s'appuyer sur les liens de communication fiables et stables d'Internet. En outre, il repose sur un découplage clair entre clients et serveurs : les utilisateurs sont censés utiliser des programmes clients spécifiques *(newsreaders)* pour accéder à des serveurs distants afin de poster ou de relever de nouveaux articles via ces serveurs.

Pour illustrer les potentialités du modèle de communication opportuniste tolérant les délais dans les MANETs discontinus, nous nous sommes donné pour objectif de concevoir et mettre en œuvre un système de discussion que nous avons baptisé OPPDNA *(Opportunistic Dissemination of Newsgroup Articles)*. Ce système doit être fonctionnellement assez semblable au système Usenet, mais il doit pouvoir fonctionner sur un ensemble de terminaux portables (tels que des ordinateurs portables ou des PDAs) interagissant exclusivement via des transmissions ad hoc. Un utilisateur devrait donc pouvoir poster et recevoir des articles thématiques d'une manière similaire à celle qui est possible avec le système Usenet. En fait, nous souhaitons permettre à l'utilisateur d'utiliser n'importe quel *newsreader* existant (c'est-à-dire, concrètement, un client NNTP standard) pour accéder aux *newsgroups* diffusés dans un réseau MANET, même si le système assurant la diffusion des articles dans ce contexte repose sur un modèle fort différent de celui sur lequel repose le système Usenet.

Dans un MANET discontinu, aucun hôte ne peut être considéré comme étant suffisamment stable, suffisamment accessible depuis tous les autres hôtes du réseau, et suffisamment doté en ressources pour jouer le rôle de serveur de *newsgroups* pour l'ensemble de la communauté. En conséquence, un *newsreader* tournant sur le terminal portable d'un utilisateur ne peut être associé à un serveur unique distant. La solution que nous avons adoptée consiste à faire en sorte que chaque hôte mobile fasse tourner localement un pseudo-serveur NNTP, auquel le *newsreader* de l'utilisateur peut se connecter à chaque fois que l'utilisateur souhaite poster ou récupérer de nouveaux articles.

Dans l'architecture Usenet standard, les serveurs sont administrés par du personnel spécifique. L'administrateur d'un serveur peut créer sur ce serveur des *newsgroups* locaux, décider quels *news-*

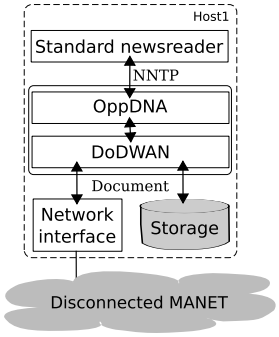


FIG. 4.6 – Architecture d'une instance du système OPPDNA

groups il doit héberger localement, avec quels serveurs pairs (ou *feeds*) il doit se synchroniser, et avec quelle fréquence ces synchronisations doivent être effectuées. Dans le système OPPDNA, le propriétaire — ou l'utilisateur attitré — d'un terminal mobile doit en quelque sorte jouer le rôle d'administrateur pour cet terminal. Un terminal mobile étant en général très limité en ressources, il n'est pas raisonnable d'imaginer qu'un utilisateur puisse autoriser son terminal à héberger *tous* les newsgroups existants. Au contraire, la liste des *newsgroups* hébergés localement doit être définie soigneusement par l'utilisateur, en sélectionnant évidemment en priorité les *newsgroups* dont il souhaite pouvoir consulter les articles et, dans un second temps, des *newsgroups* additionnels qui ne l'intéressent pas forcément, mais qu'il souhaite cependant contribuer à disséminer dans le réseau.

Les administrateurs de serveurs Usenet peuvent s'appuyer, pour configurer ces serveurs, sur des listes de *newsgroups* accessibles par exemple sur le Web. Cette approche n'est bien sûr pas applicable dans un MANET discontinu, puisque aucun terminal ne peut servir de source de référence capable de fournir la liste de tous les *newsgroups* existants. Un mécanisme de *découverte* doit donc être mis en place afin de permettre aux terminaux mobiles d'apprendre l'existence de tel ou tel *newsgroup*.

4.3.1 Architecture du système OPPDNA

L'architecture générale d'un programme OPPDNA est présentée dans la figure 4.6. Une instance de ce programme doit être exécutée en tâche de fond sur tout terminal mobile participant au système de discussion OPPDNA. Il est organisé en deux couches. La couche supérieure assure la gestion des *newsgroups* hébergés localement, tout en interagissant avec le *newsreader* de l'utilisateur via une session NNTP standard. La couche inférieure gère les interactions avec d'autres instances du programme OPPDNA, ces instances tournant sur d'autres hôtes mobiles. En pratique, le comportement attendu est que lorsque deux terminaux mobiles — et donc deux instances du programme OPPDNA — sont à portée radio l'un de l'autre, ce contact soit utilisé de manière opportuniste pour échanger des articles ayant trait aux *newsgroups* intéressant chacun des hôtes pairs. Le protocole de dissémination basée contenus embarqué dans la plate-forme intergicielle DODWAN (cf. paragraphe 3.3) est bien sûr parfaitement adapté à la réalisation de tels échanges entre des hôtes mobiles entrant fugitivement en contact dans un MANET discontinu.

Ainsi, à chaque fois qu'un utilisateur va poster un article dans un *newsgroup* donné, cet article va simplement être passé du *newsreader* au programme OPPDNA local. Il va alors être converti dans un format que l'intergiciel DODWAN est capable d'exploiter, et le document résultant va simplement être déposé dans le cache maintenu localement par DODWAN.

```
<descriptor
  id="4737511714$426a74cc"
  service="OppDNA"
  newsgroup="comp.networking"
/>
```

FIG. 4.7 – Exemple de descripteur caractérisant un article publié dans le *newsgroup comp.networking*

```
<profile
  service="OppDNA"
  newsgroup="oppdna.admin|comp.networking|sensor.networking"
/>
```

FIG. 4.8 – Exemple de profil d'intérêt caractérisant les *newsgroups* auxquels un hôte mobile est abonné

Convertir un article de *newsgroup* en un document pouvant être manipulé par DODWAN est assez simple. Concrètement, il suffit d'extraire de l'en-tête de cet article (dont le format est conforme aux directives du RFC 850 [95]) les informations qui vont permettre de produire un descripteur exploitable par DODWAN, à savoir l'identifiant unique attribué à cet article, et le nom du *newsgroup* dans lequel il a été posté. Un exemple de descripteur ainsi produit est représenté dans la figure 4.7.

Une fois déposé dans le cache local géré par DODWAN, cet article va se disséminer dans le réseau en passant de manière opportuniste sur tout terminal dont le profil d'intérêt indique qu'il s'intéresse au *newsgroup* dans lequel l'article a initialement été posté.

Inversement, à chaque fois que l'intergiciel DODWAN va parvenir à obtenir d'un terminal voisin un article correspondant à son propre profil d'intérêt — c'est-à-dire un article publié dans un *newsgroup* qu'il a reçu l'ordre d'héberger localement — cet article va simplement être déposé dans le cache local, et la couche supérieure du programme OPPDNA va être notifiée de cette réception afin que l'article en question puisse être mis à disposition du *newsreader* de l'utilisateur.

« Administration » du programme OPPDNA. Le profil d'intérêt qui détermine quels types de documents l'intergiciel DODWAN doit s'efforcer d'obtenir de ses voisins doit être défini en fonction des préférences de l'utilisateur. Un exemple de profil est reproduit dans la figure 4.8. Ce profil caractérise un hôte s'intéressant aux articles publiés dans les *newsgroups oppdna.admin* (ce *newsgroup* ayant un rôle spécifique, discuté plus loin), *comp.networking*, et *sensor.networking*.

Dans le système Usenet standard, un *newsreader* peut enregistrer les souscriptions de l'utilisateur (c'est-à-dire la liste des *newsgroups* qu'il souhaite pouvoir consulter), et obtenir les articles correspondants du serveur NNTP auquel il est associé. Cependant le *newsreader* n'est pas en mesure d'indiquer au serveur que celui-ci devrait lui-même s'abonner à tel ou tel *newsgroup*, les abonnements du serveur vis-à-vis de serveurs pairs étant géré directement par l'administrateur. Dans le système OPPDNA il faut donc permettre à un utilisateur de configurer lui-même le programme OPPDNA dont il dépend. Plusieurs approches alternatives sont envisageables pour ce faire. On peut par exemple doter le programme OPPDNA d'une interface utilisateur conviviale (sous forme textuelle ou graphique), avec laquelle l'utilisateur pourra contrôler le comportement de ce programme. Une autre approche consiste à utiliser un *newsgroup* spécifiquement dédié à l'administration du système OPPDNA, l'utilisateur pouvant alors passer des consignes au programme OPPDNA en postant simplement des articles res-

pectant un format précis dans ce *newsgroup* d'administration.

Supposons par exemple qu'un utilisateur souhaite créer un nouveau newsgroup baptisé *opportunistic.networking.* Dans le système Usenet traditionnel, seul l'administrateur d'un serveur est en mesure de créer un nouveau *newsgroup*. Dans le système OPPDNA, il suffit à un utilisateur quelconque de poster dans le groupe d'administration *oppdna.admin* un article dont le champ *Subject* respecte un format précis :

```
Subject:  ANNOUNCE opportunistic.networking
```

Le corps de l'article en question peut contenir une description en format libre du *newsgroup* ainsi créé, de sa finalité, des sujets abordés dans ce *newsgroup*, etc.

Par défaut le programme OPPDNA est configuré de manière à réceptionner tous les articles postés dans le *newsgroup oppdna.admin*. Un utilisateur souhaitant se tenir informé des newsgroups existants n'a ainsi qu'à consulter les articles postés dans ce *newsgroup.*

Supposons à présent qu'un utilisateur, découvrant l'existence du nouveau *newsgroup opportunistic.networking*, décide que ce *newsgroup* l'intéresse, et que son terminal doit dorénavant collecter et héberger les articles correspondants. Il suffit à cet utilisateur de poster dans le *newsgroup oppdna.admin* un article donnant pour ordre au programme OPPDNA local de s'abonner au *newsgroup* visé :

```
Subject:  SUBSCRIBE opportunistic.networking
```

Lorsqu'un article de ce type est reçu par la couche supérieure du programme OPPDNA, cet article n'est pas passé à l'intergiciel DODWAN comme le serait un article normal. Au lieu de cela, le profil d'intérêt qui détermine le comportement de l'intergiciel est modifié de telle sorte qu'à partir de cet instant, il va s'efforcer de récupérer et stocker tout article posté dans le *newsgroup opportunistic.networking*.

L'utilisateur peut bien sûr passer une consigne au programme OPPDNA l'enjoignant de se désabonner d'un certain *newsgroup*. Pour ce faire il suffit de poster dans le groupe *oppdna.admin* un article contenant une commande de type `UNSUBSCRIBE` spécifiant le nom du *newsgroup* visé. Le profil d'intérêt qui détermine le comportement de l'intergiciel DODWAN est alors modifié en conséquence, et le cache dans lequel l'intergiciel maintient les articles récents est purgé de tout article relatif au *newsgroup* considéré.

4.3.2 Principaux résultats et perspectives

Résultats. Le développement du système OPPDNA a été réalisé en 2007 dans le cadre du projet SARAH (cf. annexe A). L'objectif de ce travail était de proposer, concevoir, et mettre en œuvre à titre expérimental un service opérationnel de niveau applicatif capable d'exploiter le protocole de dissémination basée contenus présenté dans le paragraphe 3.3, et s'interfaçant notamment avec la plate-forme DODWAN (paragraphe 3.3.5). Ce travail a fait l'objet de la publication suivante.

– Julien Haillot and Frédéric Guidec. *Towards a Usenet-like Discussion System for Users of Disconnected MANETs.* In First IEEE International Workshop on Opportunistic Networking (WON'08), Okinawa, Japon, March 2008. [89]

Perspectives. Les perspectives de prolongation de ce travail sont assez nombreuses dans la mesure où notre objectif était, en concevant le système OppDNA, d'*illustrer* le potentiel du modèle de dissémination « basée contenus » dans les MANETs discontinus, plutôt que d'aboutir à un produit fini. Quelques uns des éléments qui mériteraient ainsi d'être améliorés sont énumérés ci-dessous.

- *Gestion de cache et gestion de priorités sur les newsgroups.*
  Dans le système Usenet, l'administrateur d'un serveur est censé s'assurer que la base dans laquelle le serveur héberge les *newsgroups* est régulièrement purgée des articles les plus anciens, de telle sorte que de l'espace soit toujours disponible pour stocker les articles plus récents. Le même problème doit être traité dans le système OppDNA, mais ce problème concerne alors chaque utilisateur, plutôt qu'un seul administrateur. La plate-forme intergicielle DoDWAN implémente un certain nombre de méthodes pour gérer le cache sur un hôte mobile. Par exemple ce cache peut être purgé automatiquement des documents les plus anciens, les plus volumineux, ou bien encore les plus rarement demandés. En outre il est possible de définir des niveaux de priorité sur les documents maintenus dans le cache. L'utilisateur d'un terminal faisant tourner le programme OppDNA peut donc souscrire à certains *newsgroups*, en dédiant une fraction plus ou moins importante du cache à l'hébergement de chaque *newsgroup*.

- *Contrôle de la durée de dissémination des articles dans les* newsgroups.
  Selon le RFC 850 (qui définit la structure des articles dans le système Usenet), l'en-tête d'un article peut inclure un champ *Expire* spécifiant la date à partir de laquelle cet article doit être considéré comme étant obsolète. Ce champ est en fait rarement utilisé dans le système Usenet, dans la mesure où les serveurs NNTP sont en mesure de maintenir des articles dans leur bases de données pendant très longtemps. En revanche ce type d'information peut être extrêmement utile dans le système OppDNA, dans la mesure où les articles doivent être stockés sur des terminaux mobiles aux ressources extrêmement limitées. Ainsi, en postant un article contenant une indication explicite de sa durée de vie, un utilisateur peut empêcher que cet article soit maintenu trop longtemps dans le réseau.

- *Contrôle de la portée géographique de la dissémination des articles.*
  Il serait intéressant que le système OppDNA donne aux utilisateurs la possibilité de restreindre la portée de dissémination de certains articles. En fait, dans certains cas un utilisateur peut vouloir poster un article à l'intention d'autres utilisateurs situés dans une zone relativement bornée (comme par exemple des personnes situées dans le même bâtiment, le même quartier, la même ville, etc.). Dans la mesure où un article peut être transporté sur de grandes distances par un terminal dont le propriétaire se déplace, la méthode traditionnelle consistant à borner le nombre de sauts qu'un article peut franchir en se propageant de proche en proche dans le réseau n'est pas une façon très efficace de contrôler la portée spatiale de cette propagation. Dans un futur proche il est assez probable que de très nombreux terminaux seront capables de se positionner dans l'espace grâce à un service de géo-localisation (reposant lui-même sur des technologies de type GPS, etc.). Le système OppDNA pourrait exploiter un tel service en donnant aux utilisateurs la possibilité de spécifier, lors de la publication d'un article, la portée de la dissémination attendue pour cet article. Pour limiter la portée de dissémination spatiale d'un article il suffirait par exemple d'inclure dans son en-tête une indication de la position à partir de laquelle cet article a été publié initialement, et de l'étendue de la dissémination attendue autour de cette position. Le protocole de dissémination basée contenus implémenté dans l'intergiciel DoDWAN pourrait aisément être étendu pour tenir compte de telles contraintes de propagation.

- *Hybridation des systèmes OppDNA et Usenet.* Le système OppDNA a été conçu de manière

à pouvoir fonctionner sur un ensemble de terminaux assemblés en réseau mobile ad hoc discontinu. Le système Usenet quant à lui a été conçu pour fonctionner dans un environnement connexe et stable de type Internet. OPPDNA pourrait être étendu de façon à permettre à ses utilisateurs de bénéficier du meilleur des deux mondes, en supposant toutefois que les terminaux des utilisateurs (ou tout au moins certains d'entre eux) puissent être connectés à Internet, ne serait-ce que de manière sporadique. Dans la mesure où le programme OPPDNA implémente le protocole NNTP (pour interagir avec le *newsreader* de l'utilisateur), il pourrait être modifié de façon à ce que, à chaque fois qu'un utilisateur connecte son terminal à Internet, cette connexion soit exploitée par le programme OPPDNA pour échanger des articles avec un serveur NNTP distant. Ainsi, une connectivité — même sporadique — avec des passerelles donnant accès à Internet (comme par exemple des points d'accès Wi-Fi) permettrait aux terminaux de se synchroniser vis-à-vis des *newsgroups* hébergés par des serveurs NNTP standards, et les articles ainsi obtenus pourraient ensuite se propager de proche en proche parmi les terminaux grâce au système OPPDNA. Inversement, un article posté au niveau d'un terminal pourrait se propager au sein du MANET discontinu, jusqu'à ce qu'il atteigne un terminal qui soit en mesure de le relayer vers un serveur NNTP distant.

Chapitre 5

Conclusion

Les travaux présentés dans ce document ont été menés au cours des huit dernières années (entre 2000 et 2008), et se sont presque tous inscrits dans le cadre de projets[1] labellisés et financés successivement par la Région Bretagne (projet MASC), le Ministère de la Recherche (projet CONCERTO), et l'Agence Nationale de la Recherche (projet SARAH).

La chronologie de ces travaux révèle une évolution lente, mais certaine, de mes préoccupations et de mes centres d'intérêt au cours de ces années. En fait une évolution semblable peut être observée à l'échelle de l'équipe CASA tout entière. Ceci résulte tout simplement du fait que la stratégie appliquée dans cette équipe de petite taille a jusqu'à présent consisté à focaliser l'ensemble des énergies disponibles sur quelques thèmes fédérateurs, plutôt que de les disperser sur une multitude de thèmes divergents.

Nos travaux ont ainsi dans un premier temps été concentrés de manière quasi-exclusive sur la problématique de la perception de l'environnement *(context-awareness)* de la part de composants logiciels, ces composants étant déployés et s'exécutant dans un environnement d'informatique ambiante caractérisé par des ressources hétérogènes et fluctuantes. Cet axe de recherche a notamment motivé le stage de DEA, puis la thèse de doctorat de Nicolas Le Sommer (cf. annexe B). L'ensemble du travail réalisé dans cette voie nous a permis de proposer des outils d'ordre méthodologique et de type intergiciel permettant de doter les composants logiciels de la possibilité de percevoir l'environnement dans lequel ils s'exécutent et, le cas échéant, d'adapter leur comportement aux caractéristiques de cet environnement. Cette approche s'est notamment soldée par le développement des intergiciels RAJE et SAJE (cf. paragraphe 2.2), qui permettent à des composants Java d'observer finement les ressources offertes par leur environnement d'exécution et, dans le cas de RAJE, d'évaluer leur propre consommation vis-à-vis de ces ressources.

Ces résultats nous ont permis de conduire une étude axée sur le support de la qualité de service offerte aux composants logiciels s'exécutant dans un environnement aux ressources fluctuantes. L'approche que nous avons proposée consiste à établir une relation contractuelle entre les composants logiciels et la plate-forme d'exécution sur laquelle on les déploie. Dans cette approche, un composant logiciel doit être capable d'exprimer ses propres besoins vis-à-vis des ressources offertes par la plate-forme visée afin de négocier avec celle-ci les modalités d'accès à ces ressources. La plate-forme expérimentale JAMUS, développée par Nicolas Le Sommer dans le cadre de son travail de doctorat (cf. paragraphe 2.4 et annexe B), supporte ce type de relation contractuelle vis-à-vis de programmes Java.

[1] Rappel : ces divers projets sont détaillés dans l'annexe A p. 107.

Dans ce cas précis, les programmes applicatifs hébergés par la plate-forme sont considérés comme étant non sûrs, et donc a priori non dignes de confiance. La plate-forme assure donc l'hébergement sécurisé de ces programmes en les supervisant au cours de leur exécution afin de prévenir notamment toute tentative d'utilisation illicite des ressources mises à leur disposition (*e.g.* détournement de ressources, déni de service, etc.).

Notre activité dans le domaine de la perception du contexte d'exécution a également été étendue aux environnements distribués. Dans le cadre du projet CONCERTO (cf. paragraphe 2.3 et annexe A) nous nous sommes en effet intéressés au déploiement de composants logiciels parallèles sur des grappes de machines, et à la perception de la part de ces composants de l'ensemble des ressources disséminées à l'échelle d'une grappe.

Avec le projet MASC *(Mobile Adaptive Software Components*, cf. annexe A) dans lequel s'est inscrit le travail de doctorat de Hervé Roussain (cf. annexe B), nous avons commencé à nous intéresser aux environnements d'informatique ambiante atypiques, voire « difficiles » *(challenged environments)*, que constituent les environnements constitués d'équipements mobiles et communicants. Les environnements de ce type diffèrent assez fortement les uns des autres selon que l'on considère un ensemble de capteurs mobiles *(sensor network)*, un ensemble de terminaux embarqués sur des véhicules *(VANET : Vehicular Ad hoc NETwork)*, ou tout simplement des terminaux (PDAs, *smartphones*, etc.) transportés par des individus. En règle générale cependant les équipements présents se caractérisent par des ressources extrêmement limitées, mais se distinguent en même temps les uns des autres par l'hétérogénéité de ces ressources. L'utilisation de mécanismes de perception de l'environnement (vu comme un ensemble de ressources) est donc inévitable dans de tels environnements, dès lors que l'on veut permettre aux composants logiciels déployés sur les équipements considérés d'adapter dynamiquement leur comportement aux ressources disponibles sur ces équipements.

En fait l'une des ressources dont la disponibilité n'est absolument pas garantie dans un environnement d'informatique mobile est le réseau lui-même, ou plus exactement la connectivité réseau (vue comme l'aptitude pour un équipement mobile à communiquer avec un ou plusieurs autres équipements). Il nous est donc rapidement apparu comme nécessaire d'intégrer dans notre réflexion portant sur les ressources disponibles dans un environnement d'informatique ambiante la « ressource réseau ».

La majorité des travaux réalisés jusqu'à ce jour dans le domaine des réseaux mobiles portent en fait sur des réseaux dans lesquels des terminaux mobiles dépendent d'une infrastructure de communication fixe (*e.g.* réseaux GSM/GPRS ou Wi-Fi). Pour nous distinguer de ces travaux nous avons choisi de nous intéresser, dans le cadre du projet MASC, aux problèmes posés dans le contexte spécifique des réseaux mobiles ad hoc, c'est-à-dire des environnements dans lesquels tous les équipements sont mobiles et peuvent, potentiellement, interagir les uns avec les autres sur la base de transmissions directes à faible portée. Dans ce type d'environnement la « ressource réseau » est une ressource éminemment fluctuante et incertaine, dans la mesure où elle dépend à tout instant de la position relative des divers équipements communicants les uns par rapport aux autres. Nous nous sommes donc intéressés aux protocoles conçus afin d'assurer les communications dans les réseaux mobiles ad hoc, et constatant que la plupart des protocoles développés au cours de la dernière décennie ne peuvent fonctionner que dans des réseaux ad hoc denses et connexes (ce que les réseaux réels ne sont pas forcément), nous avons entrepris de développer de nouveaux protocoles capables de fonctionner dans des réseaux à connexité partielle ou intermittente. Cette approche nous a permis d'inscrire dès 2004 notre activité dans la dynamique alors toute récente de la communication opportuniste et tolérant les délais *(opportunistic networking, delay-tolerant networking)*, qui constitue aujourd'hui un champ d'activité en plein essor dans le domaine des réseaux mobiles ad hoc. Dans le cadre du projet MASC nous avons

ainsi conçu un protocole permettant d'assurer la dissémination « basée contenus » de données entre des terminaux mobiles assemblés en réseau ad hoc discontinu (*i.e.* non connexe). Ce protocole, qui repose sur le principe général de la diffusion épidémique contrôlée, a depuis lors été raffiné dans le cadre du travail de doctorat de Julien Haillot (cf. annexe B). Il a notamment été intégré à la plate-forme intergicielle DoDWAN *(Document Dissemination in Wireless Ad hoc Networks)* dont le développement constitue l'un des objectifs du projet ANR SARAH (cf. annexe A). La plate-forme DoDWAN doit ainsi offrir des moyens de communication grâce auxquels des services applicatifs élaborés pourront être déployés et utilisés au sein d'un réseau ad hoc discontinu.

En parallèle avec nos travaux portant sur le support de la communication dans les réseaux mobiles ad hoc, nous menons en effet depuis plusieurs années une réflexion sur les services pouvant être déployés dans ce type de réseaux. L'une des approches pertinentes dans cette voie consiste à privilégier le déploiement d'applications distribuées fonctionnant en mode pair à pair plutôt qu'en mode client-serveur. Le modèle pair à pair présente des propriétés d'auto-organisation et d'auto-maintenance qui sont extrêmement avantageuses dans des environnements dans lesquels la connectivité de bout en bout entre clients et serveurs ne peut être assurée. En outre des applications reposant sur ce modèle posent a priori moins de problèmes de « passage à l'échelle » que des applications nécessitant des serveurs dédiés. Il est à noter que la fourniture de services au sens traditionnel du terme, *i.e.* entre terminaux « prestataires » de services et terminaux « consommateurs » de ces services, fait également l'objet d'études menées par l'équipe CASA dans le cadre du projet SARAH. Ces travaux n'ont cependant pas été détaillés dans ce document. En effet je n'y ai pas participé moi-même jusqu'à présent, ayant concentré mon attention sur les aspects relatifs à la communication dans les réseaux ad hoc discontinus, et la mise en œuvre de services pair à pair dans ce type de réseaux.

Dans le cadre du projet MASC nous avons ainsi développé à titre expérimental une plate-forme destinée à assurer le déploiement sur des terminaux mobiles d'applications conçues par assemblage de composants logiciels (cf. paragraphe 4.2.2). Cette plate-forme repose sur un schéma de fonctionnement purement pair à pair, dans lequel les terminaux impliqués coopèrent pour s'échanger de manière opportuniste les composants qui leur font défaut. Elle exploite pour ce faire le protocole de dissémination opportuniste « basée contenus » évoqué plus haut.

La conception de services applicatifs capables de fonctionner sur des terminaux mobiles fait également l'objet de l'un des axes d'activité du projet SARAH. Dans cette optique nous avons notamment conçu un système permettant à des utilisateurs dotés de terminaux capables de communication ad hoc de bénéficier d'un service de discussion similaire à celui fourni par le système Usenet. Notre système diffère cependant de Usenet par le fait que, contrairement à ce dernier, il ne repose pas sur un modèle client-serveur, mais sur un modèle purement pair à pair (cf. paragraphe 4.3). Cette fois encore, les terminaux mobiles utilisent notre protocole de dissémination « basée contenus » pour s'échanger de manière sélective des articles de *newsgroups*.

Il ressort de ce bilan que, tout en maintenant nos activités dans le domaine général de l'informatique ambiante, nous avons peu à peu fait évoluer ces activités vers la prise en compte de l'hétérogénéité, de la mobilité, et des ruptures de continuité qui en résultent dans les systèmes impliquant des équipements mobiles. Je suis pour ma part intimement convaincu que des systèmes constitués d'équipements capables de communiquer de gré à gré, en l'absence de toute infrastructure de communication fixe, sont appelés à occuper dans l'avenir un certain nombre de « niches écologiques » que les systèmes traditionnels, plus lourds et plus coûteux, ne peuvent guère combler. La communication en situation de crise (*e.g.* communications tactiques, coordination d'équipes de secours, etc.), dans des milieux dépourvus d'infrastructures de communication (*e.g.* individus ou robots travaillant en

zone désertique), mais aussi la communication au sein d'une population limitée (*e.g.* communication « tribale »), sont autant d'exemples illustrant le potentiel de la communication ad hoc. Les travaux de recherche réalisés jusqu'à ce jour n'ont fait qu'effleurer ces divers domaines d'application possibles.

Bibliographie

[1] Information Technology, Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks, Specific Requirements Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. ANSI/IEEE Std 802.11, 1999.

[2] Information Technology, Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks, Specific Requirements Part 15 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs). ANSI/IEEE Std 802.15, 2002.

[3] Information Technology, Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks, Specific Requirements Part 16 : Air Interface for Fixed Broadband Wireless Access Systems. ANSI/IEEE Std 802.16, 2004.

[4] R. Amstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski. Towards a Common Component Architecture for High-Performance Scientific Computing. In *Proc. of the 8th International Symposium on High-Performance Computing*, Redondo Beach, Californie, August 1999.

[5] Jesper Andersson. A Deployment System for Pervasive Computing. In *Proceedings of the International Conference on Software Maintenance (ICSM'2000)*, pages 262–270, San Jose, October 2000.

[6] Felix Bachmann, Len Bass, Charles Buhman, Santiago Comella-Dorda, Fred Long, John Robert, Robert Seacord, and Kurt Wallnau. Technical concepts of component-based software engineering. Technical report, Carnegie Mellon University – Software Engineering Institute, 2000.

[7] Godmar Back, Wilson C. Hsieh, and Jay Lepreau. Processes in KaffeOS : Isolation, Resource Management, and Sharing in Java. In *4th Symposium on Operating Systems Design and Implementation*, October 2000.

[8] Godmar Back, Patrick Tullmann, Legh Stoller, Wilson C. Hsieh, and Jay Lepreau. Techniques for the Design of Java Operating Systems. In *USENIX Annual Technical Conference*, June 2000.

[9] Nataraj Bagaratnan and Steven B. Byrne. Resource Access Control for an Internet UserAgent. In *Third USENIX Conference on Object-Oriented Technologie and Systems*, 1997.

[10] R. Baldoni, R. Beraldi, M. Migliavacca, L. Querzoni, G. Cugola, and L. Migliavacca. Content-Based Routing in Highly Dynamic Mobile Ad Hoc Networks. *Journal of Pervasive Computing and Communication*, 1(4) :277–288, dec 2005.

[11] Claudio Basile, Marc-Olivier Killijian, and David Powell. A Survey of Dependability Issues in Mobile Wireless Networks. Technical report 02637, LAAS CNRS Toulouse, France, oct 2002.

[12] F. Baude, D. Caromel, and M. Morel. From Distributed Objects to Hierarchical Grid Components. In *Proc. of International Symposium on Distributed Objects and Applications (DOA'2003)*, LNCS, Catania, Sicile, November 2003.

[13] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau, and Damien Watkins. Making components contract aware. *Computer*, 32(7) :38–45, 1999.

[14] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani. The Fractal Component Model and Its Support in Java. *Software Practice and Experience, special issue on Experiences with Auto-adaptive and Reconfigurable Systems*, 36(11–12), 2006.

[15] E. Bruneton, T. Coupaye, and J.-B. Stefani. Recursive and dynamic software composition with sharing. In *Proc. of the 7th ECOOP International Workshop on Component-Oriented Programming (WCOP'02)*, Malaga, Espagne, June 2002.

[16] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, and Jean-Bernard Stefani. An Open Component Model and Its Support in Java. In *Component-Based Software Engineering, 7th International Symposium*, pages 7 – 22. Springer-Verlag Heidelberg, 2004.

[17] P. Calégari, F. Guidec, and P. Kuonen. A Parallel Genetic Approach to Transceiver Placement Optimisation. In C.-A. Héritier and B. Chopard, editors, *Proceedings of the SIPAR Workshop'96 : Parallel and Distributed Systems*, pages 21–24, October 1996.

[18] P. Calégari, F. Guidec, and P. Kuonen. Urban Radio Network Planning for Mobile Phones. *EPFL Supercomputing Review*, (9) :4–10, November 1997. *(Invited paper)*.

[19] P. Calégari, F. Guidec, P. Kuonen, B. Chamaret, S. Josselin, D. Wagner, and M. Pizarosso. Radio Network Planning with Combinatorial Optimization Algorithms. In Chr. Christensen, editor, *Proceedings of the ACTS Mobile Telecommunications Summit 96*, volume 2, pages 707–713, November 1996.

[20] P. Calégari, F. Guidec, P. Kuonen, and D. Kobler. Parallel Island-Based Genetic Algorithm for Radio Network Design. *Journal of Parallel and Distributed Computing (JPDC) : Special Issue on Parallel Evolutionary Computing, Academic Press*, 47(1) :86–90, November 1997.

[21] P. Calégari, F. Guidec, P. Kuonen, and F. Nielsen. Combinatorial Optimization Algorithms for Radio Network Planning. In *Proceedings of the 10th Franco-Japanese, 5th Franco-Chinese Conference on Combinatorics and Computer Science*, 1997.

[22] P. Calégari, F. Guidec, P. Kuonen, and F. Nielsen. Combinatorial Optimization Algorithms for Radio Network Planning. *Theoretical Computer Science (TCS), Special Issue on Combinatorics and Computer Science*, 265(1-2) :235–245, 2001.

[23] P. Calégari, P. Kuonen, F. Guidec, and D. Wagner. A Genetic Approach to Radio Network Optimization for Mobile Systems. In *Proceedings of the IEEE 47th Vehicular Technology Conference (VTC)*, volume 2 of *Technology in Motion*, pages 755–759. IEEE, May 1997.

[24] D. Caromel, W. Klauser, and J. Vayssiere. Towards Seamless Computing and Metacomputing in Java. *Concurrency Practice and Experience*, 10(11–13), November 1998.

[25] Antonio Carzaniga, Alfonso Fuggetta, Richard S. Hall, Dennis Heimbigner, André van der Hoek, and Alexander L. Wolf. A characterization framework for software deployment technologies. Technical Report CU-CS-857-98, Dept. of Computer Science, University of Colorado, April 1998.

[26] Antonio Carzaniga and Alexander L. Wolf. Content-based Networking : A New Communication Infrastructure. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, number 2538 in LNCS, pages 59–68, Scottsdale, Arizona, October 2001. Springer-Verlag.

[27] I. Chakeres and C. Perkins. Dynamic MANET On-demand (DYMO) Routing. Technical Report draft-ietf-manet-dymo-11, IETF, November 2007.

[28] Jörg Kaiser Changling Liu. A Survey of Mobile Ad Hoc network Routing Protocols. Technical report, University of Magdeburg, 2005.

[29] S. J. Chapin, D. Katramatos, J. Karpovich, and A. Grimshaw. The Legion Resource Management System. In *Proc. of the 5th Workshop on Job Scheduling Strategies for Parallel Processing, in conjunction with the International Parallel and Distributed Processing Symposium (IPDPS '99)*, April 1999.

[30] T. Clausen and P. Jacquet. Optimized Link-State Routing Protocol (OLSR). Technical Report RFC 3626, Project Hipercom, INRIA, October 2003.

[31] Paola Costa, Mirco Musolesi, Cecilia Mascolo, and Gian Pietro Picco. Adaptive Content-based Routing for Delay-tolerant Mobile Ad Hoc Networks. Technical report, UCL, aug 2006.

[32] Paolo Costa and Gian Pietro Picco. Semi-Probabilistic Content-Based Publish-Subscribe. In *25th International Conference on Distributed Computing Systems (ICDCS 2005)*, pages 575–585, Columbus, Ohio, USA, jun 2005. IEEE Computer Society.

[33] Luc Courtrai, Frédéric Guidec, Nicolas Le Sommer, and Yves Mahéo. Resource Management for Parallel Adaptive Components. In *Workshop on Java for Parallel and Distributed Computing, IPDPS'03*, pages 134–141, Nice, France, April 2003. IEEE CS.

[34] Luc Courtrai, Frédéric Guidec, and Yves Mahéo. Concerto : composants parallèles adaptables, October 2001.

[35] Luc Courtrai, Frédéric Guidec, and Yves Mahéo. Concerto : gestion de ressources pour composants parallèles adaptables. In *Chapitre 4 des actes de l'École GRID'2002*, pages 41–53, Aussois, France, December 2002.

[36] Luc Courtrai, Frédéric Guidec, and Yves Mahéo. Gestion de ressources pour composants parallèles adaptables. In *Journées Composants, Systèmes à composants adaptables et extensibles*, Grenoble, France, October 2002.

[37] Grzegorz Czajkowski and Thorsten von Eicken. JRes : a Resource Accounting Interface for Java. In *ACM OOPSLA Conference*, 1998.

[38] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid Information Services for Distributed Resource Sharing. In *10th IEEE Int. Symposium on High-Performance Distributed Compu ting*. IEEE Press, August 2001.

[39] Anwitaman Datta, Silvia Quarteroni, and Karl Aberer. Autonomous Gossiping : a Self-Organizing Epidemic Algorithm for Selective Information Dissemination in Mobile Ad-Hoc Networks. In *IC-SNW'04 (International Conference on Semantics of a Networked World)*, number 3226 in LNCS, pages 126–143, Paris, jun 2004.

[40] L. DeMichiel. Enterprise javabeans specification, version 2.1. Rapport technique, Sun Microsystems, June 2002.

[41] A. Denis, C. Pérez, and T. Priol. Towards High Performance CORBA and MPI Middleware for Grid Computing. In *2nd Int. Workshop on Grid Computing*, Denver, Colorado, USA, November 2001.

[42] Anind K. Dey. Understanding and Using Context. *Journal on Personal and Ubiquitous Computing*, 5(1) :4–7, February 2001.

[43] DMTF. CIM specification v2.2. Technical Report DSP0004, Data Management Task Force, http ://www.dmtf.org 1999.

[44] David Evans. *Policy-Directed Code Safety*. PhD thesis, Massachussets Institute of Technology, February 2000.

[45] David Evans and Andrew Twyman. Flexible Policy-Directed Code Safety. In *IEEE Security and Privacy*, May 1999.

[46] Kevin Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of ACM SIGCOMM03*, August 2003.

[47] Kevin Fall. Messaging in Difficult Environments. Technical report, Intel Research Berkeley, 2004.

[48] Kevin Fall, Wei Hong, and Samuel Madden. Custody Transfer for Reliable Delivery in Delay Tolerant Networks. Technical report, Intel Research Berkeley, 2003.

[49] Fangzhe Chang and Ayal Itzkovitz and Vijay Karamcheti. User-level Resource-constrained Sandboxing. In -. The 4th USENIX Windows Systems Symposium, August 2000.

[50] S. Frolund and J. Koistinen. Quality-of-service specification in distributed object systems. *IOP/BCS Distributed Systems Engineering Journal*, December 1998. to appear.

[51] Stéphane Frénot. Gestion du déploiement de composants sur réseau p2p. In *DECOR'04*, pages 113–124, 2004.

[52] Li Gong. Java Security : Present and Near Future. *IEEE Micro*, - :14–19, May 1997.

[53] Li Gong and Roland Schemers. Implementing Protection Domains in the Java Development Kit 1.2. In *Internet Society Symposium on Network and Distributed System Scurity*, March 1998.

[54] F. Guidec. Performances des communications sur le T-Node. *Lettre du Transputer et des Calculateurs Distribués*, décembre 1990. (Également disponible en publication interne de l'IRISA, Campus de Beaulieu, F-35042 Rennes, septembre 1990.).

[55] F. Guidec. La simulation des réseaux de terrain. *Marché Suisse des Machines*, (22) :20–23, octobre 1992. *(Invited paper)*.

[56] F. Guidec. Réseaux de terrain : un cas pratique. *Marché Suisse des Machines*, (6) :42–47, mars 1993. *(Invited paper)*.

[57] F. Guidec. Programmation par objets et parallélisme de données. In Luc Bougé), editor, *Actes des 6èmes Rencontres Francophones du Parallélisme (RenPar'6)*, pages 187–191, juin 1994.

[58] F. Guidec. Object-Oriented Parallel Software Components for Supercomputing. In Peters D'Hollander, Joubert and Trystram, editors, *Parallel Computing : State of the Art and Perspectives. Proceedings of PARCO'95 (Parallel Computing), Gent, Belgium*, Advances in Parallel Computing. Universiteit Gent, North-Holland, 1995.

[59] F. Guidec. *Un cadre conceptuel pour la programmation par objets des architectures parallèles distribuées : application à l'algèbre linéaire*. Thèse de doctorat, IFSIC / Université de Rennes 1, juin 1995.

[60] F. Guidec, J.-P. Baguet, J.-D. Decotignie, L. Ruiz, and G. Ulloa. Fieldbus Requirements and Architecture for CNC. In *Proceedings of the European Control Conference (ECC'91), Grenoble*, July 1991.

[61] F. Guidec, P. Calégari, and P. Kuonen. Object-Oriented Parallel Software for Radio Wave Propagation Simulation in Urban Environment. In S. Lengauer, M. Griebl, and S. Gorlatch, editors, *EuroPar'97 Parallel Processing (Third International EuroPar Conference, Passau, Germany, August 1997, Proceedings)*, volume 1300 of *Lecture Notes in Computer Science*, pages 832–839. Springer, September 1997.

[62] F. Guidec, P. Calégari, and P. Kuonen. Parallel Irregular Software for Wave Propagation Simulation. In Hertzberger and Sloot, editors, *High-Performance Computing and Networking (HPCN Europe'97, Vienna)*, number 1225 in Lecture Notes in Computer Science, pages 84–94. Springer Verlag, April 1997.

[63] F. Guidec, P. Calégari, P. Kuonen, and M. Pahud. Object-Oriented Parallel Software for Parallel Radio Wave Propagation Simulation in Urban Environment. *Computers and Artificial Intelligence*, 18(6) :525–539, November 1999.

[64] F. Guidec, J.-D. Decotignie, C. Fuhrman, J. Hernandez, P. Raja, and L. Ruiz. DRUGH : a Simulator for Distributed Real-Time Applications. In A. Verbraeck and E.J.H. Kerchoffs, editors, *European Simulation Symposium*, pages 423–429, Society for Computer Simulation International, Delft University of Technology, October 1993.

[65] F. Guidec and J.-M. Jézéquel. Embedding Data Parallelism in Sequential Object Oriented Languages. In *Proceedings of TOOLS Europe'93, workshop on Distribution and Concurrency, Paris*. Prentice-Hall, March 1993. Position paper.

[66] F. Guidec and J.-M. Jézéquel. EPEE : an Environment for Parallel Execution of Eiffel. International Eiffel User Conference (IEUC'93), March 1993. *Invited paper*.

[67] F. Guidec and J.-M. Jézéquel. Numeric Parallel Programming with Sequential Object Oriented Languages. In *Proceedings of the First Annual Object-Oriented Numerics Conference (OONSKI'93), Sunriver, Oregon*, pages 55–69. Rogue Wave Software/SIAM, April 1993.

[68] F. Guidec and J.-M. Jézéquel. Redistribution dynamique dans EPEE. In *Actes des journées du GDR Programmation, Paris, Projet SOPAD*, octobre 1993.

[69] F. Guidec and J.-M. Jézéquel. Design of a Parallel Object-Oriented Linear Algebra Library. In Proceedings of IFIP WG10.3. K. M. Decker and R. M. Rehmann, editors, *Programming Environments for Massively Parallel Distributed Systems*, pages 359–364. Birkhäuser Verlag, Basel, July 1994. ISBN 3-7643-5090-3.

[70] F. Guidec and J.-M. Jézéquel. Polymorphic Matrices in Paladin. In *Workshop on Object-Based Parallel and Distributed Computation (OBPDC'95)*, Tokyo, June 1995. LNCS. *(Invited paper)*.

[71] F. Guidec, J.-M. Jézéquel, and J.-L. Pacherie. An Object-Oriented Framework for Supercomputing. *Journal of Systems and Software, Special Issue on Software Engineering for Distributed Computing*, June 1996.

[72] F. Guidec, P. Kuonen, and P. Calégari. ParFlow++ : a C++ Parallel Application for Wave Propagation Simulation. volume 10, pages 68–73. SPEEDUP Society, December 1996.

[73] F. Guidec, P. Kuonen, and P. Calégari. Radio Wave Propagation Simulation on the Cray T3D. *Future Generation Computer Systems (FGCS), N.H. Elsevier*, 13(4-5) :279–289, March 1998. ISSN 0167-739X.

[74] F. Guidec, P. Kuonen, and P. Calégari. Radio Wave Propagation Simulation on the Cray T3D. In E.H. D'Hollander, G.R. Joubert, F.J. Peters, and U. Trottenberg, editors, *Proceedings of ParCo'97, Bonn*, Parallel Computing : Fundamentals, Applications and New Directions, pages 155–162. Elsevier Science B.V., 1998. ISBN 0-444-82882-6.

[75] F. Guidec and Y. Mahéo. Un support d'exécution pour machines parallèles. In Luc Bougé), editor, *Actes des 6èmes Rencontres Francophones du Parallélisme (RenPar'6)*, pages 207–212, juin 1994.

[76] F. Guidec and Y. Mahéo. POM : une machine virtuelle parallèle incorporant des mécanismes d'observation. *Calculateurs Parallèles*, 7(2), 1995. Numéro spécial consacré aux environnements d'exécution de programmes parallèles.

[77] F. Guidec and Y. Mahéo. POM : a Parallel Observable Machine. In Peters D'Hollander, Joubert and Trystram, editors, *Parallel Computing : State of the Art and Perspectives. Proceedings of PARCO'95 (Parallel Computing), Gent, Belgium*, Advances in Parallel Computing. Universiteit Gent, North-Holland, 1995.

[78] F. Guidec and Y. Mahéo. POM : a Virtual Parallel Machine Featuring Observation Mechanisms. In *Proc. of the International Conference on High Performance Computing, New Delhi, India*, December 1995.

[79] F. Guidec, P. Raja, J. Hernandez, L. Ruiz, and J.-D. Decotignie. Simulating Fieldbus Applications with DRUGH Simulator. *Computers in Industry, Elsevier Press BV, Holland*, 1995.

[80] F. Guidec, L. Ruiz, P. Raja, and J. Hernandez. Characterizing Temporal Consistency in a Distributed Real-Time Environment. In *Proceedings of the Singapore International Conference on Intelligent Control and Instrumentation(SICICI'92), Singapore*, pages 766–771, February 1992.

[81] Frédéric Guidec and Nicolas Le Sommer. Towards Resource Consumption Accounting and Control in Java : a Practical Experience. In *Workshop on Resource Management for Safe Language, ECOOP 2002*, Málaga, Spain, June 2002.

[82] Frédéric Guidec and Hervé Roussain. Asynchronous Document Dissemination in Dynamic Ad Hoc Networks. In J. Cao et al., editor, *Second International Symposium on Parallel and Distributed Processing and Applications (ISPA'04)*, volume 3358 of *LNCS*, pages 44–48, Hong-Kong, China, December 2004. Springer Verlag.

[83] Frédéric Guidec and Hervé Roussain. Dissémination asynchrone d'information en mode peer-to-peer dans les réseaux ad hoc. In *Premières journées francophones : mobilité et ubiquité (UbiMob 2004)*, pages 207–210, Nice, France, June 2004. Cépaduès Editions.

[84] Frédéric Guidec. Communication dans les réseaux mobiles ad hoc discontinus. *La Lettre Techniques de l'Ingénieur*, (8) :3–4, July 2007. *Invited paper*.

[85] Frédéric Guidec and Yves Mahéo. Opportunistic Content-Based Dissemination in Disconnected Mobile Ad Hoc Networks. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007)*, pages 49–54, Papeete, French Polynesia (Tahiti), November 2007. IEEE Press.

[86] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, U. Ismail, and S. Keshav. Design and Implementation of the KioskNet System. In *International Conference on Information Technologies and Development*, December 2007.

[87] Julien Haillot and Frédéric Guidec. Communication "basée contenus" dans les réseaux mobiles ad hoc discontinus. In *8e Conférence Internationale sur les NOuvelles TEchnologies de la REpartition (NOTERE 2008)*, Lyon, France, 2008. ACM. À paraître.

[88] Julien Haillot and Frédéric Guidec. A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks. In *IEEE 22nd International Conference on Advanced Information Networking and Applications (AINA'08)*, pages 188–195. IEEE CS, March 2008.

[89] Julien Haillot and Frédéric Guidec. Towards a Usenet-like Discussion System for Users of Disconnected MANETs. In *First IEEE International Workshop on Opportunistic Networking (WON'08)*, pages 1678–1683, March 2008.

[90] Richard S. Hall and Humberto Cervantes. An OSGi Implementation and Experience Report. In *IEEE Consumer Communications and Networking Conference*, January 2004.

[91] Richard S. Hall, Dennis Heimbigner, and Alexander L. Wolf. A Cooperative Approach to Support Software Deployment Using the Software Dock. In *International Conference on Software Engineering*, pages 174–183, 1999.

[92] Khaled A. Harras, Kevin C. Almeroth, and Elisabeth M. Belding-Royer. Delay Tolerant Mobile Networks (DTMNs) : Controlled Flooding in Sparse Mobile Networks. In *IFIP Networking Conference, Waterloo, Ontario, CANADA*, May 2005.

[93] Chris Hawblitzel, Chi-Chao Chang, Grzegorz Czajkowski, Deyu Hu, and Thorsten von Eicken. Implementing Multiple Protection Domains in Java. In *USENIX Annual Technical Conference*, June 1998.

[94] L. Hogie, P. Bouvry, and F. Guinand. The MADHOC simulator. http ://www-lih.univ-lehavre.fr/ hogie/madhoc.

[95] Mark R. Horton. Standard for Interchange of USENET Messages. Technical Report RFC 850, Project Hipercom, INRIA, June 1983.

[96] Tuomo Hyyryläinen, Teemu Kärkkäinen, Cheng Luo, Valdas Jaspertas, Jouni Karvo, and Jörg Ott. Opportunistic Email Distribution and Access in Challenged Heterogeneous Environments. In *Proceedings of the 2007 SIGCOMM Workshop on Challenged Networks (CHANTS'07)*, pages 97–100. ACM Press, August 2007.

[97] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. I. Rubenstein. Energy-efficient Computing for Wildlife Tracking : Design Tradeoffs and Early Experiences with ZebraNet. *ACM SIGPLAN Notices*, 37 :96–107, 2002.

[98] J.-M. Jézéquel, F. Guidec, and F. Hamelin. Parallelizing Object-Oriented Software Through the Reuse of Parallel Components. *Object-Oriented Systems*, 1 :149–170, 1994.

[99] Kihun Kim and Klara Nahrstedt. A Resource Broker Model with Integrated Reservation Scheme. In *IEEE International Conference on Multimedia and Expo (II)*, pages 859–862, 2000.

[100] Nabil Kouici, Denis Conan, and Guy Bernard. Caching Components for Disconnection Management in Mobile Environments. In *International Symposium on Distributed Objects and Applications*, October 2004.

[101] K. Krauter, R. Buyya, and M. Maheswaran. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. *Sofware – Practice and Experience*, 32(2) :135–164, February 2002.

[102] P. Kuonen, F. Guidec, and P. Calégari. Multilevel Parallelism Applied to the Optimization of Mobile Networks. In A. Tentner, editor, *Proceedings of the conference on High-Performance Computing (HPC'98)*, Society for Computer Simulation International, pages 277–282, April 1998. ISBN 1-56555-145-1.

[103] Nicolas Le Sommer. Contractualisation des ressources pour les composants logiciels. Rapport de stage de DEA, DEA, IFSIC/UBS, 2000.

[104] Nicolas Le Sommer. A Contract-Based Approach of Resource Management in Informations Systems. In *9th International Conference on Object-Oriented Information Systems (OOIS 2003)*, volume 2817 of *LNCS*, Genève, Switzerland, September 2003. Springer Verlag.

[105] Nicolas Le Sommer. *Contractualisation des ressources pour les composants logiciels : une approche réflexive*. Mémoire de thèse, Université de Bretagne-Sud, December 2003.

[106] Nicolas Le Sommer. Contractualisation des ressources pour le déploiement de composants logiciels. In *1ère conférence Francophone sur le Déploiement et la (Re)-Configuration de Logiciels (Decor 2004)*, pages 211–222, Grenoble, France, June 2004. Net Print, Eybens.

[107] Nicolas Le Sommer. Towards Dynamic Resource Contractualisation for Software Components. In *2nd International Working Conference on Component Deployment (CD 2004)*, volume 3083 of *LNCS*, pages 129–143, Edinburg, Scotland, UK, May 2004. Springer Verlag.

[108] Nicolas Le Sommer. A Framework for Service Provision in Intermittently Connected Mobile Ad hoc Networks. In *8th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM 2007)*, Helsinki, Finland, June 2007. IEEE Computer Society. Short Paper.

[109] Nicolas Le Sommer. Service Provision in Disonnected Mobile Ad Hoc Networks. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007)*, pages 125–130, Papeete, French Polynesia (Tahiti), November 2007. IEEE Press.

[110] Nicolas Le Sommer. Vers une ubiquité d'accès aux services dans les réseaux mobiles ad hoc. In *7ème Conférence Internationale sur les NOuvelles TEchnologies de la REpartition (Notere 2007)*, pages 207–218. Hermes Sciences, June 2007. To appear.

[111] Nicolas Le Sommer and Frédéric Guidec. Intégration des threads temps-réel POSIX sous Java. In *13e Rencontres Francophones du Parallélisme (RenPar'13, ASTI'2001)*, pages 91–95, Paris, France, April 2001.

[112] Nicolas Le Sommer and Frédéric Guidec. A Contract-Based Approach of Resource-Constrained Software Deployment. In Judith Bishop, editor, *1st International IFIP/ACM Working Conference on Component Deployment (CD 2002)*, volume 2370 of *LNCS*, pages 15–30, Berlin, Germany, June 2002. Springer Verlag.

[113] Nicolas Le Sommer and Frédéric Guidec. JAMUS : une plate-forme d'accueil sécurisée pour le code mobile. In M. Dao and M. Huchard, editors, *Langages et Modèles à Objets (LMO'2002)*, L'Objet, pages 203–215, Vannes, France, February 2002. Hermes Science.

[114] Nicolas Le Sommer and Frédéric Guidec. JAMUS : Java Accommodation of Mobile Untrusted Software. In *4th Nord EurOpen/Usenix Conference (NordU 2002)*, pages 38–48, Helsinki, Finland, February 2002. Multiprint, Helsinki. Best Paper.

[115] Nicolas Le Sommer and Frédéric Guidec. Middleware Support for Resource-Constrained Software Deployment. In *4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2003)*, volume 2893 of *LNCS*, pages 49–60, Paris, France, November 2003. Springer Verlag.

[116] Nicolas Le Sommer, Frédéric Guidec, and Hervé Roussain. A Context-Aware Middleware Platform for Autonomous Application Services in Dynamic Wireless Networks. In *First International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'2006)*, number 9, Nice, France, May 2006. ACM Digital Library.

[117] Nicolas Le Sommer and Frédéric Guidec. A Contract-Based Approach of Resource-Constrained Software Deployment. In J. van Leeuwen G. Goos, J. Hartmanis, editor, *Proceedings of the First International IFIP/ACM Working Conference on Component Deployment (CD'2002, Berlin, Germany)*, number 2370 in Lecture Notes in Computer Science, pages 15–30. Springer, June 2002.

[118] Nicolas Le Sommer and Frédéric Guidec. JAMUS : Java Accommodation of Mobile Untrusted Software. In *4th EurOpen/USENIX Conference (NordU'2002, Helsinki, Finland)*, February 2002. http ://www.univ-ubs.fr/valoria/Orcade/RASC/Publications/NordU2002.pdf.

[119] Nicolas Le Sommer and Hervé Roussain. JASON : an Open Platform for Discovering, Delivering and Hosting Applications in Mobile Ad Hoc Networks. In *International Conference on Pervasive Computing and Communications (PCC'04)*, pages 714–720, Las Vegas, Nevada, USA, June 2004. CSREA Press.

[120] Nicolas Le Sommer and Hervé Roussain. JASON : une plate-forme ouverte pour la découverte et l'hébergement de services applicatifs dans les réseaux ad hoc. In *Premières journées francophones : mobilité et ubiquité (UbiMob 2004)*, pages 187–194, Nice, France, June 2004. Cépaduès Editions.

[121] Nicolas Le Sommer and Hervé Roussain. Une approche pour une continuité de service pour les utilisateurs de terminaux mobiles. In *Deuxièmes journées francophones : mobilité et ubiquité (UbiMob 2005)*, Grenoble, France, June 2005.

[122] Jérémie Leguay, Anders Lindgren, James Scott, Timur Friedman, and Jon Crowcroft. Opportunistic Content Distribution in an Urban Setting. In *CHANTS '06 : Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 205–212, Pisa, Italy, 2006. ACM Press.

[123] Vincent Lenders, Gunnar Karlsson, and Martin May. Wireless Ad Hoc Podcasting. In *SECON '07. 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 273 – 283. IEEE CNF, June 2007.

[124] Vincent Lestideau. *Modèles et environnement pour configurer et déployer des systèmes logiciels*. PhD thesis, Université de Savoie, December 2003.

[125] Qun Li and Daniela Rus. Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 44–55, Boston, August 2000. ACM Press.

[126] A. Lindgren, A. Doria, and O. Schelen. Probabilistic Routing in Intermittently Connected Networks. In *Proceedings of the The First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004)*, Fortaleza, Brazil, August 2004.

[127] Yves Mahéo, Luc Courtrai, and Frédéric Guidec. Concerto : composants parallèles adaptables. Rapport de fin de projet aci-grid 2001 :je2-concerto, Valoria, Université de Bretagne-Sud, December 2003.

[128] Yves Mahéo, Frédéric Guidec, and Luc Courtrai. A Java Middleware Platform for Resource-Aware Distributed Applications. In *2nd Int. Symposium on Parallel and Distributed Computing (ISPDC'2003)*, pages 96–103, Ljubljana, Slovenia, October 2003. IEEE CS.

[129] Yves Mahéo, Frédéric Guidec, and Luc Courtrai. Middleware Support for the Deployment of Resource-Aware Parallel Java Components on Heterogeneous Distributed Platforms. In *30th Euromicro Conference - Component-Based Software Engineering Track*, pages 144–151, Rennes, France, September 2004. IEEE CS.

[130] Yves Mahéo, Frédéric Guidec, and Luc Courtrai. Towards Resource-Aware Parallel Components. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04)*, pages 1006–1012, Las Vegas, Nevada, USA, June 2004. CSREA Press.

[131] Raphaël Marvie, Philippe Merle, Jean-Marc Geib, and Mathieu Vadet. OpenCCM : une plateforme ouverte pour composants CORBA. In *Actes de la seconde Conférence Française sur les Systèmes d'Exploitation (CFSE'2)*, Paris, France, April 2001.

[132] René Meier and Vinny Cahill. STEAM : Event-Based Middleware for Wireless Ad Hoc Network. In *International Conference on Distributed Computing Systems, Workshops (ICDCSW '02)*, pages 639–644, jul 2002.

[133] Microsoft. The component object model specification. Rapport technique, Microsoft Corporation, October 1995.

[134] Marija Mikic-Rakic and Nenad Medvidovic. Toward a Framework for Classifying Disconnected Operation Techniques. In *ICSE 2003 Workshop on Software Architectures for Dependable Systems*, 2003.

[135] Mirco Musolesi, Cecilia Mascolo, and Stephen Hailes. Adapting Asynchronous Messaging Middleware to Ad Hoc Networking. In *2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC2004)*, 2004.

[136] Mirco Musolesi, Cecilia Mascolo, and Stephen Hailes. EMMA : Epidemic Messaging Middleware for Ad hoc networks. *Personal and Ubiquitous Computing Journal*, 2005. To Appear.

[137] M. Neary, A. Phipps, S. Richman, and P. Capello. Javalin 2.0 : Java-based Parallel Computing on the Internet. In *European Parallel Computing Conference (Euro-Par'2000)*, Munich, Allemagne, August 2000.

[138] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162. ACM/IEEE, 1999.

[139] OMG. Corba components. Rapport technique OMG-orbos-99-07-01, OMG TC Documents, July 1999.

[140] M. Pahud, F. Guidec, and T. Cornu. Analyses de performances d'un algorithme de propagation d'ondes. In *9es Rencontres Francophones du Parallélisme (RenPar'9), Lausanne*, 1997.

[141] M. Pahud, F. Guidec, and T. Cornu. Performance Analysis of a Parallel Program for Wave Propagation Simulation. In S. Lengauer, M. Griebl, and S. Gorlatch, editors, *EuroPar'97 Parallel Processing (Third International EuroPar Conference, Passau, Germany, August 1997, Proceedings)*, volume 1300 of *Lecture Notes in Computer Science*, pages 1030–1033. Springer, September 1997.

[142] M. Pahud, F. Guidec, and T. Cornu. Performance Evaluation of a Radio Wave Propagation Parallel Simulator. In *Proceedings of the Third International Conference on Massively Parallel Computing System (MPCS'98), Colorado Springs, USA*, April 1998.

[143] Charles E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2001.

[144] Milenko Petrovic, Vinod Muthusamy, and Hans-Arno Jacobsen. Content-Based Routing in Mobile Ad Hoc Networks. In *Proc. of the 2nd Annual International Conference on Mobile and Ubiquitous Systems : Networking and Services (MobiQuitous'05)*, San Diego, California, USA, July 2005. IEEE Press.

[145] Noël Plouzeau, Stéphane Lorcy, and Jean-Marc Jézéquel. A framework managing quality of service contracts in distributed applications. In *TOOLS (26)*, pages 125–, 1998.

[146] P. Poupyrev, M. Kosuga, and P. Davis. Analysis of Wireless Message Broadcast in Large Ad Hoc Networks of PDAs. In *Proceedings of the Fourth IEEE conference on Mobile and Wireless Communications Networks*, pages 299–303, 2002.

[147] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks. In *35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, page 298. IEEE CS, 2002.

[148] R. Raman, M. Livny, and M. Solomon. Matchmaking : Distributed Resource Management for High Throughput Computing. In *Proc. of the 7th IEEE International Symposium on High Performance Distributed Computing*, Chicago, USA, July 1998.

[149] A. Ribes. Vers l'utilisation de composants parall<e8>le pour le couplage de codes de calcul scientifique. In *Actes des 14e Rencontres francophones sur le parall<E9>lisme, Renpar'14*, Hammamet, Tunisie, April 2002.

[150] Davy Robert. Échanges de fichiers condidentiels via un médium partagé. Rapport de stage de DEA, DEA IHM, UBS, 2003.

[151] Hervé Roussain and Frédéric Guidec. A Peer-to-Peer Approach to Asynchronous Data Dissemination in Ad Hoc Networks. In *International Conference on Pervasive Computing and Communications (PCC'04)*, pages 799–805, Las Vegas, Nevada, USA, June 2004. CSREA Press.

[152] Hervé Roussain and Frédéric Guidec. Cooperative Component-Based Software Deployment in Wireless Ad Hoc Networks. In *3rd International Working Conference on Component Deployment (CD 2005)*, volume 3798 of *LNCS*, pages 1–16, Grenoble, France, November 2005. Springer Verlag.

[153] Hervé Roussain and Frédéric Guidec. Déploiement de composants logiciels sur des équipements mobiles communicants : une approche coopérative. In *Journées Composants 2005*, pages 83–88, Le Croisic, France, April 2005.

[154] Hervé Roussain, Nicolas Le Sommer, and Frédéric Guidec. Towards an Asynchronous Dissemination and a Safe Deployment of Lightweight Programs in Mobile Networks. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2005)*, pages 481–483, Taormina, Italy, June 2005. IEEE CS.

[155] Elisabeth M. Royer and Chai-Keong Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[156] Yoav Sasson, David Cavin, and André Schiper. Probabilistic Broadcast for Flooding in Mobile Ad Hoc Networks. Technical Report IC/2002/54, Swiss Federal Institute of Technology (EPFL), 2002.

[157] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot. Haggle : a Networking Architecture Designed Around Mobile Users. In *Proceedings of the 2006 IFIP Conference on Wireless on Demand Network Systems and Services (IFIP WONS 2006)*, January 2006.

[158] Diego Sevilla, José M. García, and Antonio Gómez. Design and Implementation Requirements for CORBA Lightweight Components. In *Proceedings of International Conference on Parallel Processing. Workshop on Metacomputing Systems and Applications.*, pages 213–218, September 2001.

[159] T. Small and Z. J. Haas. The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way). In *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, Annapolis, MD, USA, June 2003.

[160] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and Wait : an Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *WDTN '05 : Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, Philadelphia, Pennsylvania, USA, 2005. ACM.

[161] Nils P. Sudmann and Dag Johansen. Software Deployment Using Mobile Agents. In Judith Bishop, editor, *Proceedings of the IFIP/ACM Working Conference on Component Deployment (CD 2002)*, volume 2370 of *LNCS*, pages 97–107, Berlin, Germany, June 2002. Springer.

[162] Sun Microsystems. Java Web Start 1.5.0 Documentation, 2004.

[163] Clemens Szyperski. *Component Software : Beyond Object-Oriented Programming*. ACM Press, Addison-Wesley, 1998.

[164] The Apache Software Foundation. Apache Maven. http ://maven.apache.org/.

[165] Amin Vahdat and David Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical report, Duke University, April 2000.

[166] Bill Veners. *Inside the Java 2 Virtual Machine*. Mac Graw-Hill, 1998.

[167] Einar Vollset, Ken Birman, and Robbert van Renesse. Chickweed : Group Communication for Embedded Devices in Opportunistic Networking Environments. In *3rd International Workshop on Dependable Embedded Systems, in conjunction with 25th Symposium on Reliable Distributed Systems (WDES 2006)*, Leeds, UK, October 2006.

[168] Eiko Yoneki and Jon Crowcroft. Haggle Delivrable 6.1 : User-Centred Design of Haggle Applications, February 2007. Haggle project, http ://www.haggleproject.org/images/b/b3/Haggle_D6.1.pdf.

[169] Stefanos Zachariadis, Cecilia Mascolo, and Wolfgang Emmerich. SATIN : A Component Model for Mobile Self Organisation. In *CoopIS/DOA/ODBASE (2)*, pages 1303–1321, 2004.

[170] Xiaolan Zhang, Jim Kurose, Brian Neil Levine, Don Towsley, and Honggang Zhang. Study of a Bus-Based Disruption Tolerant Network : Mobility Modeling and Impact on Routing. In *Proc. ACM Intl. Conf. on Mobile Computing and Networking (Mobicom'07)*, pages 195–206, September 2007.

[171] Zhensheng Zhang. Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks : Overview and Challenges. *IEEE Communications Surveys and Tutorials*, 8(1) :24–37, jan 2006.

[172] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. In *Proceedings of ACM Mobihoc 2004*, Tokyo Japan, May 2004.

Annexe A

Projets

Les divers travaux mentionnés dans ce document ont pour la plupart été réalisés dans le cadre de projets impliquant tout ou partie des membres de l'équipe CASA, et dans certains cas des partenaires extérieurs au laboratoire VALORIA. Ces divers projets sont présentés ci-dessous.

A.1 Projet RASC (2000-2003)

Le projet RASC *(Resource-Aware Software Components)* a servi de fil conducteur aux activités menées au sein de l'équipe CASA entre 2000 et 2003, et notamment au travail de thèse de Nicolas Le Sommer. Ce projet interne, sans label ni financement spécifiques, a cependant eu le mérite de fédérer les activités de notre équipe autour d'une thématique commune : la mise à disposition, pour les composants logiciels, de moyens de percevoir leur environnement vu comme un ensemble de ressources.

Le travail réalisé dans le cadre de ce projet s'est notamment soldé par le développement des intergiciels d'introspection RAJE *(Resource-Aware Java Environment)* et SAJE *(System-Aware Java Environment)*, décrits dans le paragraphe 2.2 de ce document. Une version distribuée de l'intergiciel RAJE, baptisée D-RAJE, a par ailleurs été conçue et intégrée à la plate-forme CONCERTO (cf. paragraphe 2.3).

Une méthode originale permettant d'assurer sous forme contractuelle la gestion des ressources mises à disposition des composants logiciels a par ailleurs été définie dans le cadre du projet RASC. Cette méthode, décrite dans le paragraphe 2.4, a notamment permis la mise en œuvre de la plate-forme JAMUS, dédiée à l'hébergement sécurisé d'applications non dignes de confiance. Les résultats du projet RASC ont été présentés dans les publications [81, 104, 106, 107, 111, 112, 113, 114, 115].

A.2 Projet ACI GRID CONCERTO (2002-2004)

Le projet CONCERTO, dirigé par Yves Mahéo, a été labellisé par le Ministère de la Recherche fin 2001, dans le cadre du programme ACI GRID 2001. Ce projet financé pour deux ans a à l'époque valu à l'équipe CASA le statut de « Jeune équipe » dans la classification des projets ACI.

L'objectif du projet Concerto était de contribuer à l'étude de l'application des technologies de

développement à base de composants au domaine du *Grid Computing.* Notre approche a consisté en la définition d'un modèle simple de construction des applications parallèles à base de composants, chacun de ces composants étant conçu d'emblée comme un « code parallèle » destiné à être déployé sur une plate-forme distribuée. L'originalité du modèle de composant *Concerto* tel que nous l'avons défini réside dans le fait qu'il inclut la notion d'activité comme élément de base d'un composant. En outre ce modèle intègre des mécanismes permettant aux composants de percevoir l'état de leur environnement d'exécution, et de s'adapter aux fluctuations observées dans cet environnement.

Ce travail a fait l'objet de 6 présentations dans des conférences (dont 4 d'audience internationale) entre 2002 et 2004 [33, 34, 35, 36, 129, 130], et d'un rapport de recherche présenté au Ministère [127].

A.3 Projet Région MASC (2002-2005)

Le projet MASC *(Mobile Adaptive Software Components)*, dont j'ai été le porteur, a été financé de 2002 à 2005 par la Région Bretagne dans le cadre du programme 1042 « Renouvellement des compétences dans les laboratoires » (contrat B/1042/2002/012/MASC).

Ce projet s'inscrivait dans la thématique générale de l'ubiquité numérique. L'objectif était de proposer des modèles et un support permettant le déploiement de services applicatifs à base de composants logiciels sur des équipements mobiles et hétérogènes, tels que des ordinateurs portables ou des PDA équipés de moyens de communication sans fils (de type Bluetooth, Wi-Fi, etc.). Cet objectif impliquait la conception de composants logiciels capables de percevoir leur environnement de déploiement et d'exécution, mais aussi de services de type intergiciel *(middleware)* aptes à supporter de tels composants. Il nécessitait également la prise en compte des contraintes spécifiques présentées par les plates-formes de déploiement considérées (*i.e.* des équipements mobiles aux ressources limitées) et des moyens de communication disponibles pour contrôler et pour faire interagir de tels équipements (*i.e.* faible portée des transmissions, non-connexité du réseau, déconnexions fréquentes et non prédictibles, etc.).

Le travail de thèse d'Hervé Roussain s'est inscrit dans le cadre de ce projet, auquel a également participé Nicolas Le Sommer. Ce travail s'est notamment soldé par la définition d'un modèle de communication basé sur la dissémination sélective d'information dans les réseaux mobiles ad hoc [82, 83, 151], et d'un modèle de déploiement de composants logiciels reposant sur la coopération de terminaux mobiles [152, 153]. En combinant ces modèles avec certains résultats du projet RASC, nous avons ensuite pu définir une architecture permettant de sécuriser l'hébergement des applications ainsi déployées sur les terminaux mobiles [119, 120, 121, 154]. Enfin, en exploitant les mécanismes de *context-awareness* développés dans le projet RASC, nous avons également pu définir un support intergiciel permettant de guider le déploiement d'applications en fonction du contexte propre à chaque terminal mobile [116].

A.4 Projet ANR SARAH (2005-2008)

Le projet SARAH (Services Asynchrones pour Réseaux mobiles Ad Hoc), dont je suis le coordinateur, a été labellisé fin 2005 par l'ANR dans le cadre du programme ARA SSIA (Actions de Recherche Amont en Sécurité, Systèmes embarqués et Intelligence Ambiante). Il implique, outre l'équipe CASA du VALORIA, trois autres équipes académiques, membres des laboratoires LaBRI

(Univ. de Bordeaux), LITIS (Univ. du Havre), et XLIM (Univ. de Limoges).

L'objectif principal du projet SARAH est d'étudier les possibilités offertes par la communication opportuniste pour supporter le déploiement et l'utilisation de services distribués sur des terminaux mobiles assemblés en réseau ad hoc discontinu. Le scénario-type servant de fil conducteur dans le cadre du projet est celui d'une population d'individus équipés d'ordinateurs portables ou d'assistants numériques personnels, mais d'autres scénarios sont également à l'étude (*e.g.* terminaux communicants embarqués dans des véhicules). Le projet vise donc à explorer des voies susceptibles de permettre d'assurer la communication entre des terminaux qui peuvent n'être qu'épisodiquement, voire presque jamais à portée directe les uns des autres. D'autre part, le projet SARAH vise également à définir des modèles de services applicatifs adaptés aux environnements considérés, à proposer des moyens de sécuriser ces services, et enfin à en valider le comportement via des approches formelles, et via la simulation. Ces différents objectifs définissent les quatre thèmes d'activité complémentaires couverts dans le projet :

1. Support de la communication dans un MANET discontinu.
2. Support au déploiement et à l'exécution de services dans ce type d'environnement.
3. Sécurisation des communications et des services.
4. Simulation et validation des mécanismes de communication et des services.

L'équipe CASA est principalement impliquée dans les travaux relatifs aux deux premiers des thèmes listés ci-dessus. Les travaux rapportés dans le paragraphe 3.3 de ce mémoire s'inscrivent dans le premier de ces thèmes, et les travaux rapportés dans les paragraphes 4.2.2 et 4.3 s'inscrivent dans le deuxième thème.

Trois thèses sont par ailleurs en cours dans l'équipe CASA sur les thèmes abordés dans ce projet. La thèse de Julien Haillot, dont j'assure l'encadrement, a débuté en octobre 2006. Elle porte sur la définition et la validation d'un modèle de communication asynchrone pour réseaux ad hoc non filaires. La thèse de Romeo Said, encadrée par Yves Mahéo, a également débuté en octobre 2006. Elle concerne le support de services logiciels dans les réseaux tolérant les délais. Enfin la thèse de Salma Ben Sassi, encadrée par Nicolas Le Sommer, a débuté en octobre 2007. Elle porte sur la mise en œuvre de services contextuels dans les réseaux mobiles ad hoc, avec application aux systèmes d'information routière. Ces trois thèses sont réalisées sous la direction de Pierre-François Marteau.

La participation de l'équipe CASA au projet SARAH s'est jusqu'à ce jour soldée par les publications [84, 85, 87, 88, 89, 108, 109, 110].

Annexe B

Encadrement doctoral

Dans cette annexe sont présentés les divers travaux de DEA et de thèse que j'ai encadrés depuis ma prise de fonction à l'Université de Bretagne Sud en septembre 1998.

B.1 Stage de DEA de Nicolas Le Sommer

Au cours de l'année universitaire 1999-2000 j'ai proposé, conjointement avec le Prof. Jacques Malenfant (qui dirigeait à l'époque l'équipe ORCADE du laboratoire VALORIA), un sujet de stage de DEA dont l'objectif était d'étudier l'extension de l'approche contractuelle « à la Eiffel » aux contrats dits non-fonctionnels portant sur les ressources. Ce sujet a fait l'objet du stage de DEA de Nicolas Le Sommer (alors étudiant en DEA à l'IFSIC, Rennes), stage qui s'est déroulé de février à juin 2000, et dont j'ai assuré l'encadrement [103]. Nicolas Le Sommer a par la suite poursuivi ce travail dans le cadre de sa thèse préparée au sein du VALORIA.

B.2 Stage de DEA de Davy Robert

Davy Robert, étudiant dans le DEA IHM de l'UBS, a effectué son stage au sein de l'équipe CASA de février à juin 2003. Ce stage avait pour objectifs l'étude des mécanismes de chiffrement et de signature électronique disponibles à l'heure actuelle, et la définition d'un modèle permettant d'appliquer ces mécanismes en vue de sécuriser l'échange de documents électroniques sur un médium partagé (tels qu'un canal radio utilisé conjointement par des équipements mobiles assemblés en réseau ad hoc). La plate-forme de déploiement cible considérée au cours de ce travail était un ensemble d'équipements mobiles (*e.g.* ordinateurs portables et PDA) assemblés en réseau ad hoc [150].

B.3 Thèse de doctorat de Nicolas Le Sommer

Le travail de thèse de Nicolas Le Sommer s'est déroulé d'octobre 2000 à septembre 2003 dans le cadre du projet RASC *(Resource Aware Software Components)*, avec un financement MENRT. J'en ai assuré l'encadrement, sous la direction de Jacques Malenfant (PR au VALORIA).

L'objectif général de ce travail était d'étudier l'extension de l'approche contractuelle « à la Eiffel » aux contrats dits non-fonctionnels portant sur les ressources, et d'appliquer la notion de contractualisation de l'accès aux ressources au domaine des composants logiciels.

Le travail de Nicolas Le Sommer s'est notamment concrétisé par la mise en œuvre d'un environnement permettant l'observation et le contrôle fin des ressources utilisées par des composants logiciels Java *(RAJE : Resource-Aware Java Environment)*, et par la construction d'une plate-forme expérimentale dédiée à l'hébergement de codes mobiles non sûrs *(JAMUS : Java Accommodation of Mobile Untrusted Software)*. Il est à noter qu'une version simplifiée de l'environnement RAJE est à présent diffusée sous licence GPL et disponible sur les pages Web de notre laboratoire. Cette version baptisée *SAJE (System-Aware Java Environment)* offre un sous-ensemble des services d'introspection définis dans l'environnement *RAJE*, ces services étant ici limités à l'introspection vis-à-vis des ressources système utilisées par les composants logiciels.

Nicolas Le Sommer a soutenu sa thèse le 18 décembre 2003. Le jury était composé de : P. Frison (PR, président du jury), M. Riveill (PR, rapporteur), J.-L. Pazat (MC HDR, rapporteur), J. Malenfant (PR, directeur de thèse), F. Guidec (examinateur, co-directeur de thèse) [105].

Les résultats obtenus pendant la durée de cette thèse ont fait l'objet de 9 présentations dans des conférences et workshops (dont 6 d'audience internationale, et un *Best paper*) avec comité de sélection et publication dans les actes [33, 81, 104, 111, 112, 113, 114, 115, 117, 118]. D'autre part, Nicolas Le Sommer ayant été recruté par l'UBS en tant que maître de conférences en septembre 2004, il poursuit à présent ce travail en l'inscrivant dans les activités présentes du groupe CASA [106, 107, 108, 109, 110, 116, 119, 120, 121, 154].

B.4 Thèse de doctorat d'Hervé Roussain

Le travail de thèse d'Hervé Roussain s'est inscrit dans le cadre du projet MASC *(Mobile Adaptive Software Components)*, avec un financement de la Région Bretagne. J'en ai assuré l'encadrement, d'octobre 2002 à novembre 2005, sous la direction de Patrice Frison (PR au VALORIA).

Dans le cadre du projet MASC, Hervé Roussain s'est intéressé plus spécifiquement au déploiement d'applications à base de composants logiciels sur des équipements mobiles interagissant en mode ad hoc, c'est-à-dire des équipements capables de communiquer directement avec leurs voisins en l'absence de tout réseau d'infrastructure. Ce travail s'est notamment concrétisé par la définition d'un modèle de déploiement reposant sur le principe du *peer-to-peer*, et par la mise en œuvre d'un prototype permettant d'illustrer ce modèle et d'en valider le comportement dans un contexte réaliste.

Ayant connu de sérieux problèmes de santé au cours de l'année 2006, Hervé Roussain n'a malheureusement pas été en mesure de mener son travail de thèse jusqu'à la soutenance. Compte tenu du retard pris dans ce travail et de l'absence de financement qui lui aurait permis de le mener à terme, il a choisi d'y renoncer pour s'investir dans une recherche d'emploi (Hervé Roussain est aujourd'hui ingénieur R&D dans une SSII établie dans la région parisienne). On peut cependant noter que les résultats qu'il a obtenus en travaillant sur cette thèse ont fait l'objet de 10 présentations dans des conférences et workshops (dont 6 d'audience internationale) avec comité de sélection et publication dans les actes [82, 83, 116, 119, 120, 121, 151, 152, 153, 154].

B.5 Thèse de doctorat de Julien Haillot

Julien Haillot bénéficie depuis octobre 2006 d'une allocation de thèse accordée par la DGA. Son travail de thèse s'inscrit dans le cadre du projet SARAH. J'en assure l'encadrement, sous la direction de Pierre-François Marteau (PR, directeur du VALORIA).

L'objectif de ce travail est de définir, formaliser, et valider un modèle de communication capable de résister aux contraintes inhérentes aux réseaux mobiles ad hoc à forte dynamicité et faible connexité. Ce modèle devra notamment tenir compte de la volatilité des terminaux (*i.e.* terminaux tantôt allumés – donc opérationnels du point de vue du réseau –, et tantôt éteints ou mis en veille), et permettre des échanges asynchrones entre ces terminaux lorsque ceux-ci ne sont pas simultanément actifs dans le réseau. Il devra également permettre l'acheminement de messages au sein d'un réseau fractionné en îlots (*i.e.* sous-composantes connexes du réseau global), en mettant à profit la mobilité des terminaux eux-mêmes afin de transporter l'information d'îlot en îlot. Dans cette optique, l'idée générale est de s'inspirer du modèle de réseau « tolérant les délais » (ou *DTN*, pour *Delay-Tolerant Network*) proposé récemment par le groupe DTNRG *(DTN Research Group)* de l'IRTF *(Internet Research Task Force)*, et d'appliquer ce principe au cas particulier des réseaux mobiles ad hoc. Ce travail a déjà fait l'objet de deux publications internationales [88, 89] et une publication nationale [87], avec comité de sélection et publication dans les actes.