Mémoire d'Habilitation à Diriger les Recherches

présentée à

L'UPMC Sorbonne Universités

Spécialité

Informatique

présenté par

Denis Conan
Maître de Conférences
Institut Mines-Télécom, Télécom SudParis
CNRS SAMOVAR UMR 5157, Équipe ACMES

Le jeudi 9 juillet 2015

Contributions aux systèmes répartis en environnements ubiquitaires : adaptation, sensibilité au contexte et tolérance aux fautes

Jury

Rapporteurs		
Didier	Donsez	Professeur, Université Joseph Fourier Grenoble 1
Pascal	Felber	Professeur, Université de Neuchâtel
Rachid	Guerraoui	Professeur, École Polytechnique Fédérale de Lausanne
Examinateurs		
Marc-Olivier	Killijian	Directeur de Recherche CNRS, LAAS
Michel	Riveill	Professeur, Université de Nice-Sophia Antipolis
Pierre	Sens	Professeur, UPMC Sorbonne Universités
Guy	Bernard	Professeur Émérite, Institut Mines-Télécom / Télécom SudParis
Bruno	Defude	Professeur, Institut Mines-Télécom / Télécom SudParis

Résumé

D'années en années, nous observons l'arrivée sur le marché d'ordinateurs personnels de plus en plus petits pour des utilisateurs de plus en plus nombreux, ainsi des assistants personnels numériques et des objets dits connectés, en passant par les téléphones mobiles. Tous ces dispositifs tendent à être interchangeables du point de vue des ressources en mémoire, en calcul et en connectivité : par exemple, les téléphones mobiles sont devenus des équipements informatiques de moins en moins spécialisés ou de plus en plus universels et font dorénavant office en la matière de portails d'accès aux capteurs présents dans l'environnement immédiat de l'utilisateur.

L'enjeu abordé dans nos travaux est la construction de systèmes répartis incluant ces nouveaux dispositifs matériels. L'objectif de mes recherches est la conception des paradigmes d'intermédiation génériques sous-jacents aux applications réparties de plus en plus ubiquitaires. Plus particulièrement, la problématique générale de mes travaux est la définition du rôle des intergiciels dans l'intégration des dispositifs mobiles et des objets connectés dans les architectures logicielles réparties. Ces architectures logicielles reposaient très majoritairement sur des infrastructures logicielles fixes au début des travaux présentés dans ce manuscrit.

Dans ce manuscrit, je décris mes travaux sur trois sujets : *1)* l'adaptation des applications réparties pour la continuité de service pendant les déconnexions, *2)* la gestion des informations du contexte d'exécution des applications réparties pour leur sensibilité au contexte, et *3)* les mécanismes de détection des entraves dans les environnements fortement dynamiques tels que ceux construits avec des réseaux mobiles spontanés. Sur le premier sujet, nous fournissons une couche intergicielle générique pour la gestion des aspects répartis de la gestion des déconnexions en utlisant une stratégie d'adaptation collaborative dans les architectures à base d'objets et de composants. Sur le deuxième sujet, nous étudions les paradigmes architecturaux pour la construction d'un service de gestion de contexte générique, afin d'adresser la diversité des traitements (fusion et agrégation, corrélation, détection de situation par apprentissage, etc.), puis nous adressons le problème de la distribution des informations de contexte aux différentes échelles de l'Internet des objets. Enfin, sur le troisième sujet, nous commençons par la détection des modes de fonctionnement pour l'adaptation aux déconnexions afin de faire la différence, lorsque cela est possible, entre une déconnexion et une défaillance, et ensuite nous spécifions et construisons un service de gestion de groupe partitionnable. Ce service est assez fort pour interdire la construction de partitions ne correspondant pas à la réalité de l'environnement à un instant donné et est assez faible pour être mis en œuvre algorithmiquement.

Remerciements

Je remercie très vivement les Professeurs Didier Donsez, Pascal Felber et Rachid Guerraoui qui ont répondu positivement à ma demande d'être rapporteur. Je joins à ces remerciements les Professeurs Marc-Olivier Killijian, Michel Riveill, Pierre Sens, Guy Bernard, et Bruno Defude. Je les remercie sincèrement du grand honneur qu'ils me font de constituer le jury.

Le travail présenté dans ce mémoire est un travail collectif. Il n'aurait pas été possible sans la qualité de l'ambiance, de la réflexion et de la confiance présentes dans le projet MARGE, et plus largement dans le département Informatique de Télécom SudParis ainsi que l'équipe ACMES. J'adresse un merci particulier à mes collègues du projet MARGE, Chantal, Djamel, Michel, Pascal, Sophie, et aux différents directeurs du département Informatique pendant ces années, Guy Bernard, Bruno Defude, et Djamel Belaïd. Je ne présenterais pas ce mémoire s'il n'y avait pas eu l'initiation et l'encouragement permanent de Guy Bernard, qui m'a transmis, dès les premiers jours, le goût de la recherche et d'engager des projets de recherches collectifs, et qui me fait le plaisir d'être présent pour cette étape. Merci aussi à Bruno, Chantal, Christian, Claire, Gaël, Léon, Michel, Samir, Sophie pour leurs relectures.

J'associe enfin les étudiants qui ont effectué un stage ou une thèse avec moi pendant quelques mois ou pendant une plus longue durée aux résultats présentés dans ce manuscrit : Usman Bhatti, Anic Priye Singh Birrgi, Nabil Kouici, Xuan Li, Léon Lim, Cong Kinh Nguyen, Tuan Dung Nguyen, Lynda Temal, Jing Zhu. J'adresse un remerciement particulier à Nabil et Léon qui ont effectué leur thèse sous mon encadrement.

Table des matières

1 Introduction **1**
1.1 Contexte général . . . 1
1.2 Problématique générale . . . 3
1.3 Contributions . . . 4
1.4 Plan du document . . . 7

2 Adaptation pour la continuité de service pendant les déconnexions **8**
2.1 Motivations . . . 9
2.2 Structuration des applications sur les appareils mobiles . . . 10
2.3 Gestion de cache de composants déconnectés . . . 13
2.4 Bilan . . . 16

3 Sensibilité au contexte des applications réparties autonomiques **18**
3.1 Motivations . . . 19
3.2 Structuration de la gestion de contexte . . . 20
3.3 Distribution des informations de contexte . . . 24
3.4 Bilan . . . 30

4 Tolérance aux fautes des applications réparties en environnement mobile **31**
4.1 Motivations . . . 32
4.2 Détections de déconnexion, de défaillance et de partition . . . 34
4.3 Détection de participants et gestion de groupe partitionnable . . . 36
4.4 Bilan . . . 41

5 Conclusion et perspectives **43**
5.1 Bilan . . . 43
5.2 Perspectives . . . 45

Références **a**

Liste des publications **l**

1 Introduction

Les travaux décrits dans ce manuscrit correspondent à mes activités de recherche réalisées à Télécom SudParis depuis septembre 2000. Ces activités ont pour objectif d'améliorer la conception et la mise en œuvre des applications réparties à l'aide d'intergiciel (en anglais, *middleware*). Elles se situent donc dans la communauté intergiciel ainsi qu'à l'interface des domaines « système réparti », « informatique mobile et ubiquitaire », et plus récemment « Internet des objets ». L'objectif général est l'intégration des dispositifs mobiles et des objets connectés dans les architectures réparties. Pour ce faire, j'étudie plus particulièrement l'adaptation, la sensibilité au contexte et la tolérance aux fautes. Mes travaux visent à aider les concepteurs à maîtriser la complexité sans cesse croissante des applications réparties qui s'exécutent dans des environnements dynamiques et instables, de l'informatique en réseau local à l'informatique mobile et ubiquitaire, et à l'Internet des objets. Ils ont été menés en collaboration avec de nombreux collègues et j'emploierai la première personne du pluriel pour les présenter dans la mesure où ils sont le résultat de nombreuses interactions.

Dans la suite de cette section, je présente le contexte général des travaux dans la section 1.1, la problématique générale dans la section 1.2, les contributions dans la section 1.3, et le plan du document dans la section 1.4.

1.1 Contexte général

Classes de dispositifs informatiques. G. Bell relate l'histoire de l'industrie du matériel informatique en définissant des classes de dispositifs informatiques caractérisées par la même gamme de prix ou un environnement de programmation similaire ou encore une même interface de communication [23]. Ces classes évoluent, au sens où de nouvelles classes apparaissent, selon trois chemins : prix constants avec performances accrues, super-calculateurs (les plus puissants à une date donnée), et plus petits et moins chers. Ensuite, l'auteur formule une loi indiquant que, sur chacun des chemins d'évolution, une nouvelle classe apparaît chaque décennie qui permet la floraison de nouveaux cas d'utilisation et de nouveaux services informatiques. Par exemple, sur le chemin d'évolution des classes de dispositifs matériels plus petits et moins chers, nous notons l'arrivée sur le marché des ordinateurs personnels dans les années 1980, des assistants personnels numériques dans les années 1990, des téléphones mobiles (en anglais, *smartphones*) dans les années 2000, et des objets connectés dans les années 2010. Tous ces dispositifs tendent aujourd'hui à être interchangeables du point de vue des ressources en mémoire, en calcul et en connectivité : par exemple, les téléphones mobiles sont devenus des équipements informatiques de moins en moins spécialisés ou de plus en plus universels et font dorénavant office en la matière de véritables ordinateurs personnels. L'un des enjeux abordé dans nos travaux est la construction de systèmes répartis incluant ces nouveaux dispositifs matériels, principalement les téléphones mobiles et de plus en plus les objets connectés.

Intergiciels pour systèmes répartis. Pour suivre l'apparition de ces nouvelles classes d'ordinateurs, la couche intermédiaire entre les systèmes d'exploitation et les applications qui s'appelle l'intergiciel, évolue. Les travaux présentés dans ce manuscrit ciblent les intergiciels pour systèmes répartis dont le rôle est de faciliter la conception des applications réparties en masquant l'hétérogénéité des matériels et les aspects bas niveaux de la répartition dans des abstractions comme les patrons de conception et les styles architecturaux [128]. Ainsi, dans le domaine des intergiciels pour systèmes répartis, les

évolutions citées plus haut correspondent aux noms évocateurs d'informatique mobile dans les années 1990 [188], d'informatique ubiquitaire aussi appelée informatique diffuse (en anglais, *pervasive*) dans les années 2000 [208, 186] et d'Internet des objets dans les années 2010 [13]. Nous introduisons en quelques mots chacune de ces évolutions dans les paragraphes qui suivent.

Informatique mobile. Les recherches en informatique mobile prennent comme origine la mobilité des dispositifs matériels ou des utilisateurs. Quatre contraintes système sont alors incontournables [188] : *a)* comparés aux dispositifs matériels fixes, les dispositifs mobiles sont limités en termes de ressources en mémoire (volatile et stable) et en calcul ; *b)* leur perte ou leur vol sont potentiellement plus fréquents et donc leur niveau de confiance est plus faible lorsqu'ils sont insérés dans un système plus vaste ; *c)* les communications par réseau sans fil sont moins performantes, moins fiables, et soumises à des variations de qualité plus fortes et plus fréquentes ; et *d)* la dépendance sur la batterie interne oblige à gérer la consommation d'énergie. En contexte de mobilité, les solutions classiques des systèmes répartis doivent être réétudiées. Citons deux exemples. Le sujet de l'accès distant à l'information est revisité pour recharger en données un appareil mobile de manière analogue à la recharge en énergie avec une batterie pour un accès ultérieur hors connexion [55]. Les solutions de tolérance aux fautes considèrent autrement le paradigme client–serveur pour tolérer les déconnexions [112]. En outre, de nouveaux problèmes apparaissent comme le support de l'adaptation dynamique des applications au mode de fonctionnement déconnecté ou comme la gestion de l'énergie. De nouveaux concepts et de nouveaux services intergiciels sont alors mis en œuvre : par exemple, le concept d'opération déconnectée pour la continuité de service pendant les déconnexions [123] et le service de prédiction de la consommation d'énergie pour décider de l'exécution de code mobile à distance sur une machine fixe [94].

Informatique ubiquitaire. L'expression « informatique ubiquitaire » est forgée par M. Weiser en 1991 [208]. Dans son article, il développe le principe selon lequel les technologies les plus profondément acceptées sont celles qui disparaissent, c'est-à-dire qui prennent en compte l'environnement humain et sont suffisamment connues, et donc utilisables, pour que les utilisateurs finaux les manipulent inconsciemment. Quelques mois avant le lancement du magazine *IEEE Pervasive Computing* en 2002, M. Satyanarayanan fait le point sur les avancées en système réparti depuis la vision de M. Weiser et donne une liste de défis spécifiques à l'informatique ubiquitaire [186]. Tout d'abord, il remarque que la mobilité est intrinsèque à la vie de tous les jours. L'agenda de la recherche en informatique ubiquitaire inclut donc celui de l'informatique mobile, avec une relecture des solutions existantes dans un environnement « saturé » de dispositifs informatiques communicants [18]. Ensuite, un premier défi particulièrement mis en avant par l'informatique ubiquitaire est l'organisation d'espaces intelligents dans lesquels l'utilisateur est immergé [182]. Un exemple typique est une salle de réunion équipée de capteurs et d'actionneurs. L'utilisateur se voit offert l'accès à de nouveaux services pour agir sur l'ambiance de la salle (air conditionné, éclairage, etc.) et les applications existantes peuvent s'adapter au contexte de l'utilisateur (présence, niveau de bruit, etc.). Un second défi cité par M. Satyanarayanan et lié au premier est l'invisibilité, dont un objectif intermédiaire important est la minimisation de la « distraction » de l'utilisateur, c'est-à-dire la minimisation des interactions générées à cause de la présence de dispositifs matériels enfouis dans l'espace physique de l'utilisateur [85]. Enfin, notons l'importance prise par la propriété d'utilisabilité qui explique le développement de nombreux scénarios applicatifs dans les publications de la thématique.

Internet des objets. L'Internet des objets met en avant de nouvelles variétés de matériels, qui sont si petits qu'ils sont appelés choses ou objets [1], tels que des étiquettes RFID, des capteurs et des actionneurs, et qui, grâce à un schéma d'adressage unique, sont capables d'interagir entre eux et de coopérer avec leurs voisins pour atteindre un but commun [13]. Maintenant que ces objets sont accessibles depuis Internet, ce ne sont pas seulement les données en provenance des capteurs physiques à proximité de l'utilisateur, c'est-à-dire dans l'espace ambiant de l'utilisateur, mais aussi les objets des espaces ambiants distants qui acquièrent une visibilité globale : par exemple, les téléphones mobiles deviennent des portails d'accès aux capteurs présents dans l'environnement immédiat de l'utilisateur [95, 117, 109, 217]. En outre, maintenant que ces objets connectés communiquent entre eux, ils doivent être vus non seulement comme des producteurs d'informations mais aussi comme des consommateurs et des processeurs d'informations impliqués dans des communications de machine à machine (en anglais, *machine to machine, M2M*). Une conséquence importante est que les systèmes répartis tendent à devenir des systèmes « socio-techniques » avec l'utilisateur qui fait partie du système [203]. Parmi les problématiques à revisiter dans le contexte de l'Internet des objets, citons la définition de la notion de contexte avec la transparence de la distribution des informations de contexte [164]. Enfin, parmi les nouveaux défis, la visibilité globale des objets connectés pose des questions concernant le respect de la vie privée des personnes immergées dans des environnements ubiquitaires intégrés à Internet [134].

1.2 Problématique générale

L'un des traits singuliers de nos sociétés étant l'ubiquité de l'informatique et l'impact sur la vie quotidienne de nos contemporains qui en découle, la communauté des chercheurs en informatique s'efforcent d'aider les concepteurs d'applications grand public à maîtriser la complexité sans cesse croissante de la répartition. Par conséquent, l'objectif des recherches dans le domaine des intergiciels pour la répartition est de concevoir les paradigmes d'intermédiation génériques sous-jacents aux applications réparties de plus en plus ubiquitaires. Plus particulièrement, la problématique générale de mes travaux est la définition du rôle des intergiciels dans l'intégration des dispositifs mobiles et des objets connectés dans les architectures logicielles réparties. Ces architectures logicielles réparties reposaient très majoritairement sur des infrastructures fixes, c'est-à-dire sans mobilité, au début des travaux présentés dans ce manuscrit. Ainsi, je m'intéresse aux intergiciels dans l'informatique mobile, dans l'informatique ubiquitaire et dans l'Internet des objets.

Cette problématique recouvre des scénarios applicatifs divers qui étaient hier considérés comme non réalisables. Certaines parties des scénarios qui suivent sont aujourd'hui en cours de réalisation, même si c'est encore souvent de manière laborieuse. Les contributions présentées dans la suite ont pour objectifs d'aider à leur mise en œuvre future. À la suite de [208, 186, 18, 182], citons donc quelques domaines applicatifs à titre d'exemple : les applications réparties collaboratives dont l'objectif est de permettre le travail en mode déconnecté tout en convergeant vers un objectif partagé ; les applications e-commerce personnalisées (par profil utilisateur par exemple) pour faciliter les achats dans une galerie marchande avec annonce des ventes *flash*, gestion des bons de réduction et des cartes de fidélité, et informations complémentaires sur les produits (par exemple les vêtements) augmentés de capteurs ou de *QR Code* ; les applications e-santé pour le suivi des soins à domicile par des intervenants multiples, pour la détection de

1. Dans la suite du document, nous utilisons le terme générique d'objet connecté sans distinguer les objets autonomes des objets alimentés et sans faire la différence entre les objets dits pleinement fonctionnels qui sont en charge de sous-réseaux et ceux disposant de fonctionnalités plus réduites qui ne peuvent communiquer qu'avec les premiers [199]. La raison est que nous n'adressons pas la problématique de la gestion des réseaux de capteurs.

situations médicales à risque pour des personnes fragilisées, ou encore pour la gestion de crise lors d'un plan rouge ; le e-tourisme avec des guides touristiques pro-actifs sur des mobiles appareillés d'un dispositif de réalité augmentée et intégrant le partage d'informations via les réseaux sociaux ; les applications pour salles de réunions intelligentes équipées de dispositifs reliés entre eux tels que des tableaux, des murs et des tables interactifs, et des détecteurs de présence sur les chaises ; la ville intelligente pour des liaisons multimodales douces, pour la connaissance de la qualité de vie par des mesures de la pollution, ou encore pour la surveillance et l'optimisation de la consommation d'énergie ; les applications pour maison intelligente qui contrôlent des équipements domotiques ; ou encore les jeux pervasifs multijoueurs sur appareils mobiles équipés de capteurs et de dispositifs de réalité augmentée, et grâce auxquels les joueurs sont immergés dans des environnements physiques parsemés d'objets connectés utilisés dans le jeu.

Ces exemples d'applications réparties sont la source de nombreuses questions pour notre problématique générale, comme par exemple : en suivant [188, 112], qui, de l'application ou de l'intergiciel, fait quoi, quand, et comment lorsque la continuité de service doit être assurée alors que les conditions d'utilisation font alterner des périodes de bonne connectivité réseau et des périodes de déconnexion ? ; en suivant [118, 61], à quoi de telles applications sont-elles sensibles ou quels éléments constituent leur contexte d'exécution ? comment le contexte est-il modélisé ? comment est organisée l'autonomie de ces applications réparties ? comment sont organisés leur structure et leur comportement ? ; en suivant [87, 14], quelles sont les caractéristiques des fautes dans ces environnements d'exécution ? quelles propriétés de ces applications réparties dynamiques peuvent être garanties ?

La section qui suit présente succinctement mes contributions effectuées sur la problématique générale de l'intégration des dispositifs mobiles et des objets connectés dans les architectures réparties.

1.3 Contributions

Dans ce manuscrit, je présente nos travaux sur les intergiciels pour les applications réparties en environnements mobiles et ubiquitaires, et dans l'Internet des objets, selon trois sujets : l'adaptation pour la continuité de service pendant les déconnexions, la sensibilité au contexte des applications réparties et la tolérance aux fautes des applications réparties en environnement mobile. J'introduis nos contributions sur ces trois sujets dans cette section.

Adaptation pour la continuité de service pendant les déconnexions. La nomadicité des utilisateurs et des dispositifs informatiques implique deux contraintes importantes lors de la conception des systèmes répartis : la limitation des ressources matérielles imposée par la taille et le poids des appareils mobiles, et la variabilité de la qualité des communications réseau. La contrainte sur les ressources (mémoire, calcul et énergie) implique une répartition des traitements « lourds » sur des machines fixes accédées par les dispositifs mobiles. La seconde contrainte implique la gestion de la continuité de service en présence de déconnexions. Ces deux contraintes qui expriment un compromis entre autonomie et interdépendance sont prises en compte dans la littérature par le maintien d'une connexion logique en utilisant le concept d'opération déconnectée [124, 123].

À la fin des années 1990, les travaux répertoriés par J. Jing et coauteurs [112] mettent en avant les aspects comme par exemple l'extension du modèle client–serveur, la transparence en modifiant le système d'exploitation (interception, appels retours [en anglais, *callbacks*], etc.), et la conception de règles de réconciliation conformes à la sémantique des opérations (sur un système de fichiers ou sur des objets). Dans les solutions de la littérature que sont Coda [124, 123], Rover [114, 113], et Bayou [165], l'adaptation

est principalement de la responsabilité du système d'exploitation et les applications n'interviennent que très peu dans les choix des mécanismes mis en jeu : choix des entités mandataires déployées avant et utilisées pendant les déconnexions, journalisation des opérations pendant les déconnexions, et réconciliation des opérations lors des reconnexions. Dans nos travaux, nous étudions l'approche consistant à considérer la continuité de service comme une extra-fonctionnalité gérée au niveau de l'intergiciel et en mode collaboratif.

Historiquement, et au moment où nos recherches sur le sujet commencent au début des années 2000, le principal apport des intergiciels est la généricité, vue entre autres à travers les propriétés de portabilité (en termes d'interfaces de protocoles réseau, d'interfaces de systèmes d'exploitation et de langages de programmation) et d'interopérabilité (entre les canevas logiciels des différents éditeurs logiciels). Un autre apport important des recherches en intergiciel est la mise en exergue de patrons de conception aidant la structuration des fonctionnalités de la couche d'intermédiation entre l'application et le système d'exploitation, celle-là même qui orchestre l'adaptation de l'application aux conditions d'exécution. Nos travaux s'inscrivent dans cette mouvance et notre objectif est de fournir une couche intergicielle générique pour la gestion des aspects répartis de la gestion des déconnexions. Ainsi, nous prenons l'exemple de l'adaptation aux déconnexions pour évaluer l'expression de la stratégie d'adaptation collaborative dans les intergiciels orientés objet et composant.

Nous commençons par une étude utilisant les paradigmes des intergiciels orientés objet de l'écosystème CORBA de l'OMG [156] pour apporter la gestion de déconnexion aux applications patrimoniales, c'est-à-dire en rendant les services de façon aussi transparente que possible par séparation des préoccupations. Ensuite, nous étudions la stratégie d'adaptation collaborative avec l'approche ingénierie dirigée par les modèles MDA [159] et le modèle de composant CCM [158], avec une attention plus particulière sur les points suivants : le processus de développement que nous désirons centré architecture et basé composants, l'extension du modèle de conteneur de composants pour intégrer les objets de contrôle pour la gestion de déconnexion, et la prise en compte des dépendances entre les composants pour la gestion d'un cache de composants.

Sensibilité au contexte des applications réparties. À la suite des travaux sur l'adaptation pour la continuité de service, nous nous sommes posés la question de la systématisation de la surveillance de l'environnement d'exécution des applications. Cela nous amène à la notion de contexte d'exécution et à la thématique de la sensibilité au contexte des applications en environnement ubiquitaire, puis des applications au-desssus de l'Internet des objets.

L'informatique ubiquitaire et l'Internet des objets ciblent des environnements ouverts dans lesquels les utilisateurs et les dispositifs apparaissent et disparaissent à une fréquence nouvelle. Si l'utilisateur final doit gérer ces événements, la charge cognitive induite est en contradiction avec la vision de M. Weiser des technologies informatiques qui disparaissent car ne requérant pas d'intervention du tout ou uniquement des manipulations inconscientes de la part de l'utilisateur. Dans un autre domaine, celui des systèmes d'information, IBM a proposé au début des années 2000 l'informatique autonomique (en anglais, *autonomic computing*) pour rassembler les travaux autour de l'autogestion des systèmes informatiques et proposer dans une architecture générique le principe de la boucle d'autonomie. La boucle d'autonomie est organisée autour du sigle éponyme *MAPE-K* pour *Monitor*, *Analyse*, *Plan*, *Execute*, *and Knowledge* [118] : le système est supposé réflexif (le K) ; cette connaissance comprend les informations obtenues en surveillant le système dans son environnement d'exécution (le M) ; ces informations sont analysées pour en déduire

une situation d'adaptation (le A) ; lorsqu'une adaptation est planifiée (le P), un plan de déploiement ou de configuration de certaines parties du système est exécuté (le E).

Dans la vision donnée par IBM dans [118], les systèmes ciblés sont des systèmes d'information et l'autonomie concerne l'administration de ces systèmes : le M de *MAPE-K* correspond principalement à la supervision des entités gérées. En informatique ubiquitaire, la capacité d'adaptation aux changements de situations de l'environnement est appelée la sensibilité au contexte [190, 67, 214]. Les applications sensibles au contexte sont consommatrices d'informations dites de contexte, car obtenues après filtrage et traitement de nombreuses informations brutes issues directement de l'environnement d'exécution : ressources système, préférences données par l'utilisateur, données provenant des capteurs, etc. L'entité logicielle responsable de la collecte, du traitement (filtrage, agrégation, etc.) et de la présentation des informations de contexte aux applications s'appelle communément un gestionnaire de contexte [61].

Sans ce service intergiciel de gestion de contexte, l'application doit elle-même gérer les accès aux sources de contexte et les traitements des informations de contexte. Il en résulte un entremêlement des parties de l'application dédiées à la gestion des informations de contexte avec les parties dites fonctionnelles. Cette non-séparation des préoccupations complique la constitution de bibliothèques de patrons de conception et d'idiomes pour la gestion de contexte, et rend ainsi difficile la réutilisation de la gestion de contexte d'une application à une autre. L'objectif des travaux sur ce sujet est de fournir un service intergiciel qui gère de multiples clients et qui organise la distribution et le traitement des informations de contexte depuis les sources de données jusqu'aux applications sensibles au contexte.

Nous commençons par une étude sur les paradigmes architecturaux pour la construction d'un service de gestion de contexte générique, afin d'adresser la diversité des traitements des informations de contexte (fusion et agrégation d'événements, corrélation d'événements, détection de situation par apprentissage, etc.). En outre, le service est flexible pour permettre l'ajout de fonctionnalités comme le traitement de la qualité de contexte. Plus récemment, nous adressons le problème de la distribution des informations de contexte aux différentes échelles de l'Internet des objets avec une attention particulière aux spécificités de la gestion de contexte : la fourniture des modes observation et notification, la gestion du respect de la vie privée, et surtout le passage à l'échelle avec le nouveau paradigme du multi-échelle.

Tolérance aux fautes des applications réparties en environnement mobile. En parallèle des travaux sur l'adaptation des applications réparties aux déconnexions et de leur sensibilité au contexte, nous nous posons la question de la tolérance aux fautes des applications réparties utilisées en situation de mobilité.

L'immersion des utilisateurs dans les environnements ubiquitaires entraîne une dépendance forte vis-à-vis de systèmes informatiques complexes. La sûreté de fonctionnement (en anglais, *dependability*) [135, 14] d'un système informatique est « la qualité du service qu'il délivre, qualité telle que les utilisateurs puissent lui accorder une confiance justifiée » [136]. Le but est de concevoir, réaliser et utiliser des systèmes informatiques où la faute est naturelle, prévue et tolérable. Les deux attributs que nous étudions dans nos travaux sont la fiabilité et la disponibilité. Le moyen que nous utilisons est la tolérance aux fautes à l'aide de services intergiciels. Dans nos travaux, nous considérons les fautes qui sont appelées des détériorations physiques [14]. Sans traitement particulier, ces fautes ne sont pas tolérées et font dévier les applications réparties du fonctionnement attendu ; ce sont alors des défaillances. Nous nous limitons à la tolérance aux fautes par masquage : les propriétés de correction et de vivacité du système sont toujours respectées en présence des fautes considérées [87].

Les défaillances suite à des détériorations physiques qui sont les plus couramment rencontrées en situation de mobilité sont les arrêts francs de machines ou de liens de communication, les déconnexions de machines (suite à des arrêts francs, à la faible qualité du réseau sans fil, ou encore à la décision de l'utilisateur de s'isoler pour préserver la batterie de son dispositif mobile [2]) et les partitionnements réseau (suite à des arrêts francs ou des déconnexions). Lors d'une déconnexion, un appareil mobile est isolé du reste du système, et lors d'un partitionnement, c'est un ensemble de machines communiquant entre elles qui est isolé. La tolérance aux fautes est obtenue en mettant en œuvre deux types de mécanismes : la détection et le recouvrement (aussi appelé correction).

Reprenons à titre d'exemple les deux sujets de l'adaptation aux déconnexions et de la sensibilité au contexte. Dans le premier, par exemple dans les applications de travail collaboratif telles que des éditeurs de documents, les entités déconnectées utilisent des données répliquées pendant les phases de déconnexion ou de faible connectivité, et une réconciliation des données s'opère lors des reconnexions ; ce sont les techniques de réplication optimiste de données [184] qui permettent le recouvrement par poursuite [14] de l'application répartie. Dans le second sujet, par exemple dans les jeux pervasifs multijoueurs, les communications s'effectuent entre des entités appartenant à un groupe ; l'une des approches les plus courantes pour tolérer les fautes utilise un service de gestion de groupe [29, 171, 57], qui permet d'obtenir la redondance spatiale avec des communications fiables, voire avec des propriétés d'ordre comme la causalité ou l'atomicité. Dans ces deux exemples, les applications réparties sensibles au contexte peuvent être intéressées par faire la différence entre déconnexions et arrêts francs : la cause phénoménologique de la déconnexion (un détecteur local surveillant la qualité du réseau sans fil ou l'utilisateur se déconnectant) peut dans certains cas être identifiée et fournie aux autres membres du groupe qui changent alors leur mode d'utilisation de l'application. Enfin, en situation de forte mobilité dans des environnements avec réseaux sans fil spontanés, les participants forment des partitions autonomes fortement dynamiques et les applications souhaitent connaître à un instant donné la composition du groupe.

Nos contributions concernent les mécanismes de détection. Nous commençons par une étude de la détection des modes de fonctionnement pour l'adaptation aux déconnexions afin de faire la différence, lorsque cela est possible, entre une déconnexion et une défaillance. Ensuite, nous adressons le problème de la gestion de groupe partitionnable pour les applications réparties utilisant des réseaux mobiles spontanés pour leurs communications.

1.4 Plan du document

Les trois sujets précédemment présentés, nommément l'adaptation pour la continuité de service pendant les déconnexions, la sensibilité au contexte des applications réparties autonomiques, et la tolérance aux fautes des applications réparties en environnement mobile, sont développés respectivement dans les sections 2, 3 et 4. Dans la section 5, je conclus et trace des perspectives générales à ces travaux.

2. La faiblesse du niveau de la batterie est dans ce cas la détérioration physique à l'origine de la déconnexion. Mais, nous passerons sous silence dans la suite ce type de faute et ne considérerons que les défaillances qu'elles provoquent : les déconnexions.

2 Adaptation pour la continuité de service pendant les déconnexions

Une application s'exécutant sur un dispositif en situation de mobilité possède quatre modes de fonctionnement : connecté, partiellement ou faiblement connecté, déconnecté et veille [167]. Dans le mode connecté, le terminal mobile dispose d'une bonne connexion au réseau, à la manière d'une station fixe. Dans le mode faiblement connecté [149], le terminal mobile ne dispose plus pour communiquer avec le réseau que d'un lien à faible débit ou à latence élevée par exemple à la suite de perturbations du réseau hertzien. Dans le mode déconnecté, le terminal mobile est isolé soit parce qu'il n'est plus physiquement relié au réseau soit parce qu'il est impossible de maintenir une connexion sans fil de qualité suffisante. Si le dispositif est un téléphone mobile, un quatrième mode de fonctionnement existe : le mode veille utilisé notamment pour préserver les ressources énergétiques. L'objectif de notre étude est la gestion des déconnexions pour la continuité de service. Nous considérons les modes connecté, faiblement connecté et déconnecté.

Nous distinguons deux types de déconnexion : volontaires et involontaires. Les déconnexions volontaires sont déclenchées par l'utilisateur afin de préserver les ressources énergétiques de son appareil ou le forfait des communications sans fil, afin d'éviter d'être perturbé, ou lorsque les transmissions par réseau sans fil sont interdites dans des situations particulières comme pendant les phases de décollage et d'atterrissage d'un avion. Les déconnexions involontaires sont dues à des déplacements dans des zones d'ombre radio ou faiblement couvertes par des réseaux sans fil, et lors de brusques transferts intercellulaires (en anglais, *handover*).

Au début des années 2000, les principales solutions sur le sujet (Coda [124, 123], Rover [114, 113], et Bayou [165]) mettent en œuvre le concept d'opération déconnectée et appliquent une stratégie d'adaptation dite transparente [188], c'est-à-dire sans modification de l'application mais par une prise en charge complète par le système d'exploitation. L'état de l'art de J. Jing et coauteurs montre que la stratégie d'adaptation transparente est limitée [112]. À la même époque, quelques essais existent sur l'utilisation des intergciels pour la tolérance aux déconnexions. Π^2 [181] et ALICE [27] utilisent une version minimum de CORBA à destination des petits matériels [154] et la spécification CORBA « Wireless Access and Terminal Mobility in CORBA » [157] pour gérer les déconnexions de courte durée. Mais, le problème des déconnexions de longue durée avec mandataires n'est pas abordé. Nous étudions la mise en place de mandataires pour les déconnexions de longue durée par l'utilisation du concept d'opération déconnectée dans les intergiciels orientés objet et composant. Notre approche se base sur les capacités de réflexivité des intergiciels [121, 80, 44] et met en œuvre une stratégie d'adaptation par collaboration entre l'application et l'intergiciel [188].

Dans la section 2.1, je motive l'approche suivie en introduisant les stratégies, les types et les moyens d'adaptation choisis. Ensuite, j'organise la présentation de nos contributions en deux temps : dans la section 2.2, la structuration des applications sur les appareils mobiles selon qu'elles sont orientées objet ou composant, et dans la section 2.3, la gestion du cache des composants déconnectées sur le client. Enfin, je conclus dans la section 2.4 par un bilan sur ces résultats.

2.1 Motivations

L'adaptation aux déconnexions est la capacité d'un système à s'ajuster pour continuer à fonctionner en présence de déconnexions. Une adaptation se définit par une stratégie (qui définit qui de l'application ou de l'intergiciel intervient dans l'adaptation), un type (qui indique à quel moment du développement ou de l'exécution l'adaptation est effectuée) et des moyens (quels mécanismes sont mis en œuvre pour l'adaptation, et plus particulièrement, quels mécanismes de séparation des préoccupations sont utilisés).

Stratégies d'adaptation. L'intervalle des stratégies d'adaptation est délimité par deux extrémités [188]. Dans la stratégie « laisser-faire », l'adaptation est entièrement de la responsabilité des applications. Cela évite de recourir à un support système (système d'exploitation ou intergiciel). À l'autre extrémité, la stratégie « transparence » fait en sorte que l'adaptation soit entièrement de la responsabilité d'un intergiciel ou d'un système d'exploitation. Entre ces deux extrêmes, dans la stratégie d'adaptation « collaboration », l'application et l'intergiciel se concertent. L'intergiciel surveille les ressources, signale aux applications tout changement significatif, et fournit les mécanismes d'adaptation. L'application, quant à elle, fournit les politiques d'adaptation. Les nombreux travaux synthétisés dans [112] montrent que les stratégies laisser-faire et transparence ne sont pas adéquates pour supporter les déconnexions. En effet, dans la stratégie laisser-faire, il manque un contrôleur central arbitrant les demandes de ressources concurrentes en conflit et les développeurs doivent également gérer les aspects extrafonctionnels. Dans la stratégie transparence, certaines adaptations peuvent être insatisfaisantes ou même contre-productives pour certaines applications car elles ne prennent pas en compte la sémantique des applications. C'est donc la stratégie collaboration que nous étudions pour la gestion de déconnexion. Par ailleurs, comme dans [187], nous introduisons l'utilisateur comme troisième acteur de l'adaptation, en plus de l'application et de l'intergiciel. L'utilisateur peut intervenir de trois façons différentes : en guidant l'application pour changer son comportement, en demandant à l'intergiciel de garantir un certain niveau de ressources ou de qualité de service, et enfin en répondant à des suggestions de l'intergiciel ou de l'application.

Type d'adaptation. Trois types d'adaptation existent [40] : adaptation statique, adaptation dynamique et auto-adaptation. Premièrement, l'adaptation statique intervient pendant le développement ou avant l'exécution, soit en modifiant le code, soit en modifiant les options de compilation ou de déploiement. Elle nécessite de connaître *a priori* l'environnement d'exécution de l'application. Deuxièmement, l'adaptation dynamique intervient pendant l'exécution de l'application. Elle est réalisée par une intervention extérieure, la plupart du temps humaine. Troisièmement, l'auto-adaptation intervient également pendant l'exécution, mais peut être initiée par l'application elle-même ou par l'intergiciel. Les choix sont par exemple effectués de manière automatique par l'intergiciel à partir de politiques fixées par l'application. Dans nos travaux, pour exprimer nos solutions de collaboration, nous utilisons tous les types d'adaptation et exprimons classiquement nos solutions sous la forme de mécanismes (qui opèrent des actions d'adaptation) et de politiques (qui agencent les mécanismes mis en œuvre).

Moyens de l'adaptation et séparation des préoccupations. La séparation des préoccupations est une approche visant à simplifier la conception des applications. Selon cette approche, une application contient deux parties distinctes : les préoccupations fonctionnelles et les préoccupations extrafonctionnelles. Les préoccupations fonctionnelles sont celles qui réalisent le service de cœur rendu par l'application.

Les préoccupations extrafonctionnelles représentent la partie qui adapte les préoccupations fonctionnelles à un environnement et/ou à une utilisation particulière, ici la gestion de déconnexion en environnement mobile. Cinq techniques sont communément utilisées comme moyens d'adaptation par séparation des préoccupations : les mécanismes d'interception, la réflexivité, le paradigme composant/conteneur, l'ingénierie dirigée par les modèles (par exemple MDA dans le monde CORBA), et la programmation orientée aspect.

Les mécanismes d'interception mettent en œuvre le patron de conception intercepteur [191, 155] qui permet aux applications d'étendre un environnement d'exécution de façon transparente en intégrant de nouvelles préoccupations extrafonctionnelles grâce à des interfaces prédéfinies. Nous utilisons les intercepteurs dits de références IOR (*Interoperable Object Reference*) qui offrent la possibilité de modifier les politiques de gestion d'objets ainsi que les intercepteurs de requêtes qui interceptent le flux de requêtes–réponses à travers l'ORB (*Object Request Broker*).

Selon le concept de réflexivité [121, 80, 44], un programme peut posséder une représentation de lui-même et s'en servir pour raisonner (introspection) et se modifier (adaptation). L'adaptation peut être structurelle (modification de la structure du système) ou comportementale (changer le comportement du système). Dans nos travaux, nous utilisons les deux types de réflexivité pour ajouter des objets mandataires et opérer des redirections de messages.

Dans les architectures à base de composants [196], le composant qui correspond à la partie fonctionnelle est encapsulé dans un conteneur qui contient les objets de contrôle de la partie extrafonctionnelle. Dans nos travaux, nous proposons une architecture de conteneur pour composant dit déconnecté pour intégrer le concept d'objet déconnecté [123] dans les plateformes CCM [158].

L'approche MDA de conception architecturale basée modèles [159] propose de construire un modèle du système indépendant des plateformes (en anglais, *Platform Independant Model*, PIM) avant d'obtenir par transformation et spécialisation des modèles dépendants d'une plateforme (en anglais, *Platform Specific Models*, PSM). Le PIM spécifie la structure et le comportement du système en faisant abstraction des détails spécifiques comme le modèle de composants. Un PSM sert essentiellement de base à la génération du code vers la plateforme visée. C'est dans le PSM que s'effectue généralement le lien entre les parties fonctionnelles et les parties extrafonctionnelles. Dans nos travaux, nous définissons un méta-modèle pour la gestion du cache des objets déconnectés. Ce méta-modèle est indépendant des modèles de composants.

Une dernière solution pour la séparation des préoccupations est la programmation orientée aspect [122]. Un programme orienté aspect est constitué de deux parties : un programme de base qui définit les préoccupations fonctionnelles de l'application, et un ou plusieurs programmes séparés, écrits dans des langages éventuellement spécialisés, qui définissent les préoccupations extrafonctionnelles à associer. La coordination entre les deux parties se fait par un procédé d'insertion appelé tissage (en anglais, *weaving*), qui peut être statique ou dynamique. Le principe est de spécifier les endroits où le tissage doit être effectué dans le code fonctionnel. Dans nos travaux, nous n'avons pas utilisé la programmation orientée aspect.

2.2 Structuration des applications sur les appareils mobiles

Les recherches effectuées dans les années 1990 montrent que la stratégie « transparence », intensément étudiée à la fin de la décennie 1980 (par exemple [69, 72, 201]), n'est pas adéquate pour la continuité de service en environnement mobile. Plus précisément, les résultats des projets Coda [124, 123] et Odyssey [149, 152], Rover [114, 113], et Bayou [165] emploient le concept d'opération déconnectée pour

fournir de bonnes performances en situation de variation importante de la bande passante. Mais, ces travaux démontrent aussi le besoin d'une intervention de l'application pour obtenir une meilleure agilité en termes de vitesse de réaction et de précision (expression par l'application des conditions spécifiques pour déclencher une adaptation) et une meilleure fidélité en termes de cohérence de données (expression par l'application des règles de réconciliation à appliquer lors de la reconnexion).

Après les travaux de la décennie 1980 en système d'exploitation réparti sur les micro-noyaux (par exemple [31, 1, 38, 179, 180]), les années 1990 ont vu l'installation de la thématique des intergiciels avec la percée de standards comme CORBA [218] de l'OMG. Les apports des intergiciels sont la généricité avec une architecture unifiée, une description architecturale des applications indépendante des langages de programmation, des systèmes d'exploitation et des protocoles de communication, et la séparation des préoccupations avec les services intergiciels qui sont séparés des entités fonctionnelles applicatives et qui proposent des services extrafonctionnels à ces entités applicatives.

Ainsi, au début des années 2000, nous avons considéré qu'il serait intéressant d'étudier les apports des intergiciels en termes de généricité et de séparation des préoccupations pour la continuité de service des applications mobiles, et en étudiant le concept d'opération déconnectée sous deux acceptions : objets et composants. Nous commençons notre étude par l'organisation de la micro-architecture des applications sur les appareils mobiles. Dans un premier temps, nous mettons l'accent sur la stratégie « transparence » pour la tolérance aux déconnexions des applications patrimoniales CORBA en utilisant les mécanismes d'interception portables de CORBA. Dans un second temps, nous étudions la stratégie « collaboration » avec le paradigme composant/conteneur et proposons une structuration des conteneurs.

Objets déconnectés pour applications patrimoniales CORBA. En 1998, l'OMG propose une version minimum de CORBA à destination des petits matériels comme les assistants personnels numériques (en anglais, *personal digital assistants*) [154]. Il s'ensuit l'apparition de canevas logiciels fonctionnant sur des systèmes d'exploitation comme GNU/Linux Familiar ou Microsoft WindowsCE. En 2001, l'OMG adopte la spécification « Wireless Access and Terminal Mobility in CORBA » [157]. Elle cible les déconnexions transitoires en définissant un cadre pour gérer les transferts intercellulaires avec le concept de référence d'objet mobile (*mobile IOR*) et la mise en place de tunnels avec GIOP (*General Inter-ORB Protocol*). Ces deux spécifications ouvrent la possibilité de concevoir des expériences de continuité de service pour des applications s'exécutant à l'époque sur des PDAs. Π^2 [181] et ALICE [27] utilisent ces deux spécifications pour gérer deux mandataires (un sur le client et un sur le serveur) afin de tolérer les déconnexions lors de changements de cellules dans les réseaux sans fil. Dans les deux cas, seules des déconnexions de courte durée sont tolérées et gérées de manière transparente car les requêtes en erreur pendant les déconnexions longues lèvent une exception que le client doit traiter. Dans nos travaux, nous nous appuyons en plus sur le concept d'intercepteur portable, dont la spécification est publiée en 2001 [155], pour une gestion transparente des déconnexions au niveau des intercepteurs.

Les idées principales de notre proposition sont les suivantes. Un objet CORBA mandataire de l'objet distant que nous appelons objet déconnecté, est déployé automatiquement sur l'appareil mobile. Afin de gérer les potentiels conflits entre les différents objets clients sur l'appareil, la gestion des ressources du dispositif mobile est centralisée au niveau de l'intergiciel. Cela concerne la gestion des ressources réseau et la gestion des journaux des opérations effectuées pendant les phases de déconnexion. Par ailleurs, ces services sont assurés par des objets CORBA qui sont donc accessibles par les applications ; cela facilite la mise en place de la collaboration. Aussi, c'est l'objet CORBA du service de journalisation qui envoie

directement le contenu des journaux lors des reconnexions et ce même objet reçoit sous la forme de *CORBA Objects By Value* les instructions de l'application pour la gestion des journaux : par exemple, les opérations de compression d'un journal et les opérations de réconciliation qui dépendent de la sémantique applicative. La fidélité s'en trouve ainsi améliorée [16]. Par ailleurs, étant un objet CORBA, l'objet gérant les journaux dispose d'un accès simple aux autres services de l'intergiciel comme le nommage. Enfin, un objet déconnecté est similaire, en termes d'interface et de conception, à l'objet distant qu'il représente. Mais, cet objet déconnecté est spécifiquement construit pour gérer les phases de faible connectivité et de déconnexion. Par conséquent, c'est le concepteur de l'application qui choisit entre une conception simple ou complexe pour obtenir un service fortement ou faiblement dégradé lors des phases de déconnexion.

La solution est mise en œuvre avec le canevas logiciel ORBacus 4.1.0. Une campagne de mesures des performances a été effectuée sur iPAQ cadencé à 206 Mhz, disposant de 16 MB de ROM et 32 MB de RAM, et exécutant le système d'exploitation GNU/Linux Familiar ou Microsoft WindowsCE et la machine virtuelle J2ME IBM J9. L'ordre de grandeur de la durée du traitement d'une requête de petite taille envoyée par un iPAQ à un objet sur une machine fixe est de $50ms$ et $20ms$ pour Familiar et WindowsCE, respectivement. Le surcoût dû aux intercepteurs portables est évalué entre 7% à 13% sur Familiar et entre 1% à 33% sur WindowsCE, le maximum étant observé pour les messages de petite taille correspondant à une requête contenant des arguments de moins de 128 octets sans valeur de retour. Bien que non négligeable, le coût des intercepteurs apparaît comme raisonnable et prometteur, compte tenu de l'évolution des performances des appareils mobiles.

Conteneurs de composants CORBA pour opérations déconnectées. Pour une meilleure séparation des préoccupations et afin de capitaliser les bonnes pratiques mises en exergue dans les intergiciels orientés objet, le concept de composant est aujourd'hui fortement préconisé dans la conception d'intergiciel. Un autre intérêt des composants est leur meilleure intégration, par rapport aux objets, dans une conception centrée architecture. En effet, le composant est un concept particulièrement pertinent lorsque le modèle de composant est hiérarchique et autorise le partage : un composant composite peut être « ouvert » et ainsi montré un détail de l'architecture ; le composant primitif constitué de quelques classes est celui qui sera effectivement mis en œuvre dans un langage de programmation orienté objet ; le partage de composants permet d'exprimer la mise à disposition d'un service à plusieurs applications d'un hôte.

Un composant est une entité logicielle qui fournit et requiert des services, regroupés au sein d'interfaces dites offertes et requises. Les composants sont assemblés à l'aide de liaisons représentant des chemins de communication entre une interface requise par un composant et une interface compatible fournie par un autre composant. L'orientation composant est souvent associée à un langage de description d'architecture. Dans le domaine des intergiciels, le concept associé au composant et qui donne lieu à beaucoup d'études, est le conteneur. Le conteneur est l'hôte d'accueil du composant qui définit l'environnement intergiciel du composant ; historiquement, le conteneur est apparu comme le lieu de mise en œuvre des patrons de conception établis comme les meilleures pratiques des intergiciels orientés objet : patron de nommage, patrons d'interception, etc. C'est donc tout naturellement que nous nous sommes intéressés à la conception de conteneurs pour la continuité de service.

Dans notre canevas logiciel DOMINT, l'architecture des conteneurs que nous employons est inspirée de l'architecture des conteneurs ouverts [202] de OpenCCM [79], qui est une mise en œuvre du modèle de composant CCM [158]. Comme représenté dans la figure 1, le conteneur contrôle les interactions des

composants avec le monde extérieur à travers des entités appelées objets [3] de contrôle. Pour la gestion des déconnexions, nous proposons une architecture des conteneurs pour des applications mobiles où les hôtes peuvent être des clients et des serveurs pour d'autres hôtes. Nous définissons cinq objets de contrôle dans le conteneur : un détecteur local de connectivité, un contrôleur d'accès au service de gestion du cache des composants déconnectés, un gestionnaire de connecteurs, et les intercepteurs des requêtes entrantes et sortantes. Les services de journalisation des requêtes pendant les déconnexions et de réconciliation lors des reconnexions font partie des travaux d'autres membres de l'équipe (cf. [48]). Le principe du détecteur de connectivité est présenté dans la section 4. Nos contributions sur la gestion de cache des composants mandataires sont développées dans la section qui suit.

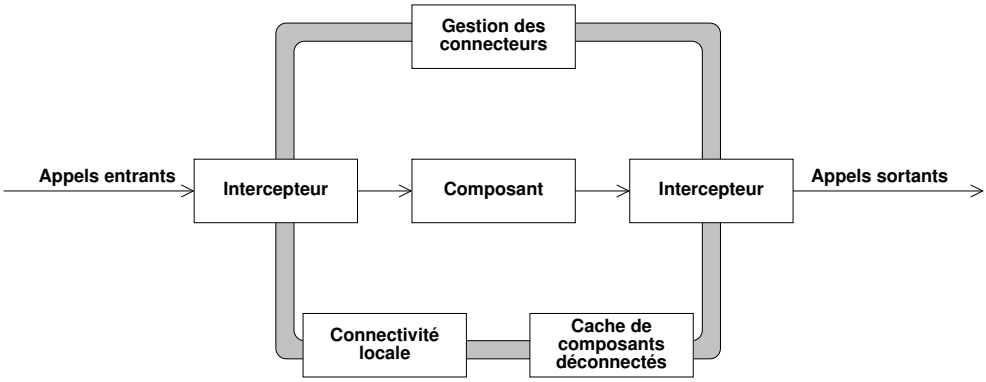


FIGURE 1 – Architecture du conteneur de DOMINT

2.3 Gestion de cache de composants déconnectés

Comme dessinée dans la section précédente, l'architecture de réalisation du paradigme « opération déconnectée » requiert l'instanciation et la gestion d'une entité dite déconnectée, qui est le mandataire de l'entité distante non accessible lors des déconnexions. Les travaux que je décris maintenant concernent la gestion du cache de ces entités déconnectées, qui sont des composants déconnectés dans notre étude.

Nous avons étudié la littérature dans l'état de l'art publié dans l'article [245]. Les premières solutions de gestion de cache pour tolérer les déconnexions apparaissent dans les systèmes d'exploitation pour modifier le système de gestion répartie de fichiers (par exemple Coda [123, 295] et Odyssey [152], Amigos [10], Seer [130] et Roam [170], PFS [71]). Même s'ils apparaissent beaucoup plus tard dans les travaux de la littérature, viennent ensuite conceptuellement les systèmes orientés page Web, qui introduisent les liens hypertextuels, qui sont une manière de rendre explicites des relations de dépendances entre fichiers (par exemple [20], WebExpress [105], SliCache [116], SIZE [210], GDSF [54], Ariadne [183]). Puis, c'est la structure interne des fichiers qui est décomposée logiquement en tables, enregistrements et champs dans les bases de données. Les solutions profitent alors des concepts de relation entre tables et de transaction pour délimiter des portions de code exécutées de façon isolée etatomique. La gestion du cache et de la réconciliation devient ainsi plus fine (par exemple [26], Bayou [165, 200], Oasis [172], Deno [46]). Ensuite, l'orientation objet apparaît de façon concomitante à l'essor des méthodes de conception logicielle, donnant naissance aux premières méthodes de conception d'applications réparties tolérant les déconnexions.

3. Le terme « objet » est ici pris dans son acception générale. Dans la mise en œuvre, nous utilisons des composants Fractal [41, 42].

En outre, les propriétés des objets, principalement l'encapsulation et le polymorphisme, permettent de commencer à concevoir des solutions séparant en partie la gestion des déconnexions du code métier de l'application répartie (par exemple Rover [114, 113] et CASCADE [56]). Enfin, le modèle de conception orienté composant est plus puissant pour exprimer l'adaptation pour la gestion du cache (par exemple DisconnectP [142] et Achilles [126]). En effet, les modèles de composants définissent un composant comme une unité de déploiement et rendent explicites les dépendances entre composants via les interfaces offertes et requises. Les travaux sur les politiques de déploiement et de remplacement du cache largement étudiées dans les autres orientations sont alors applicables. Ces politiques bénéficient en plus des mécanismes de réflexivité abondamment utilisés dans les architectures à base de composants pour introduire et manipuler des méta-données. Ces méta-données permettent une connaissance du contexte : environnement d'exécution, profil architectural de l'application, profil de l'utilisateur final, etc. Cette connaissance du contexte autorise par conséquent l'exploration de nouvelles politiques de déploiement et de remplacement.

Le gestionnaire du cache doit offrir trois politiques de déploiement, de remplacement et de cohérence [83]. La politique de déploiement détermine les entités déconnectées à créer dans le cache du terminal mobile, à quel moment et pour quelle durée. La politique de remplacement décide quelles entités déconnectées doivent être supprimées lorsqu'il n'existe plus assez d'espace mémoire. La politique de cohérence maintient la cohérence entre l'état des entités dans le cache et l'état des entités distantes. Dans nos travaux, je n'adresse pas la question de la cohérence, ce sujet étant abordé par d'autres membres de l'équipe [48]. Avec l'avènement de l'orientation composant, nous considérons qu'il est intéressant *1)* d'exploiter les dépendances entre les entités architecturales pour concevoir de nouvelles politiques de déploiement et de remplacement, et *2)* d'exprimer les éléments configurables de ces politiques dans la modélisation des éléments de l'architecture afin d'améliorer l'utilisabilité et la fidélité. Nous commençons par la modélisation de l'architecture avant de présenter les politiques de déploiement et de remplacement.

Modélisation. Nous défendons l'idée selon laquelle les points de variation de la structure ou du comportement d'une architecture doivent être identifiés et modélisés, avant d'être manipulés pendant l'exécution. Ce principe est tout particulièrement approprié pour la gestion d'extrafonctionnalités qui mettent en œuvre la stratégie d'adaptation « collaboration ». Avec une approche dirigée par les modèles, nous rendons explicites les rôles des différents intervenants, de l'utilisateur qui émet des préférences dans un profil au développeur qui orchestre les mécanismes d'adaptation, en passant par l'architecte qui définit la politique d'adaptation. Les choix de l'architecte complétés des préférences de l'utilisateur sont des informations gérées par l'intergiciel lors de l'exécution. Notre approche collaborative de gestion du cache est mise en œuvre dans le processus de développement appelé MADA (*Mobile Application Development Approach*) qui est dirigé par les modèles, centré architecture, et basé composants.

Nous introduisons un méta-modèle pour définir les méta-données de gestion du cache de composants déconnectés. Premièrement, la méta-donnée « déconnectabilité » indique si le composant résidant sur un hôte distant peut avoir un mandataire sur le terminal mobile que nous appelons un composant déconnecté. Si c'est le cas, le composant distant est dit déconnectable. C'est l'architecte qui spécifie cette propriété car c'est lui qui possède la meilleure connaissance des fonctionnalités et des contraintes de l'architecture de l'application. Par exemple, pour des raisons de sécurité, l'architecte peut interdire la déconnectabilité de certains composants. Ensuite, la méta-donnée « nécessité » appliquée à un composant déconnectable est utilisée par l'architecte puis l'utilisateur pour indiquer la nécessité de la présence d'un mandataire sur l'appareil mobile. L'utilisateur peut ainsi rendre nécessaire un composant déconnectable mis non

nécessaire par l'architecte. Par ailleurs, les liens de dépendance entre composants sont aussi estampillés pour indiquer qu'un composant déconnecté est nécessaire lorsque le composant qui l'appelle (qui en dépend) devient nécessaire. Enfin, la méta-donnée « priorité » permet d'arbitrer les choix restants lorsque des composants déconnectés doivent être évincés du cache.

Nous mettons en œuvre le patron de conception Façade [84] et la modélisation de l'architecture selon "4+1" vues [129]. Le patron de conception Façade permet de simplifier l'accès au système par un unique point d'entrée pour tous les cas d'utilisation. Dans la vue « cas d'utilisation », l'architecte montre explicitement à l'utilisateur quels sont les cas d'utilisation déconnectables, rendant ainsi déconnectables, voire nécessaires, certains des composants actifs dans les cas d'utilisation déconnectables. Pour chaque cas d'utilisation déconnectable, l'architecte indique, par un nouveau cas d'utilisation adjoint au premier, le comportement du système pendant les déconnexions pour la fonctionnalité concernée. En d'autres termes, l'architecte spécifie le mode dégradé du système pendant les déconnexions. L'architecture indique aussi des priorités sur les cas d'utilisation. L'utilisateur peut surcharger ces méta-données pendant l'exécution en émettant des préférences sur les cas d'utilisation déconnectables non nécessaires. Le méta-modèle de MADA est représenté dans la figure 2. Notons que le profil de l'application est modélisé avec les méta-modèles EDOC [160] et UML4CCM [161], et que les cas d'utilisation sont (par abus de langage) nommés des services dans la figure.

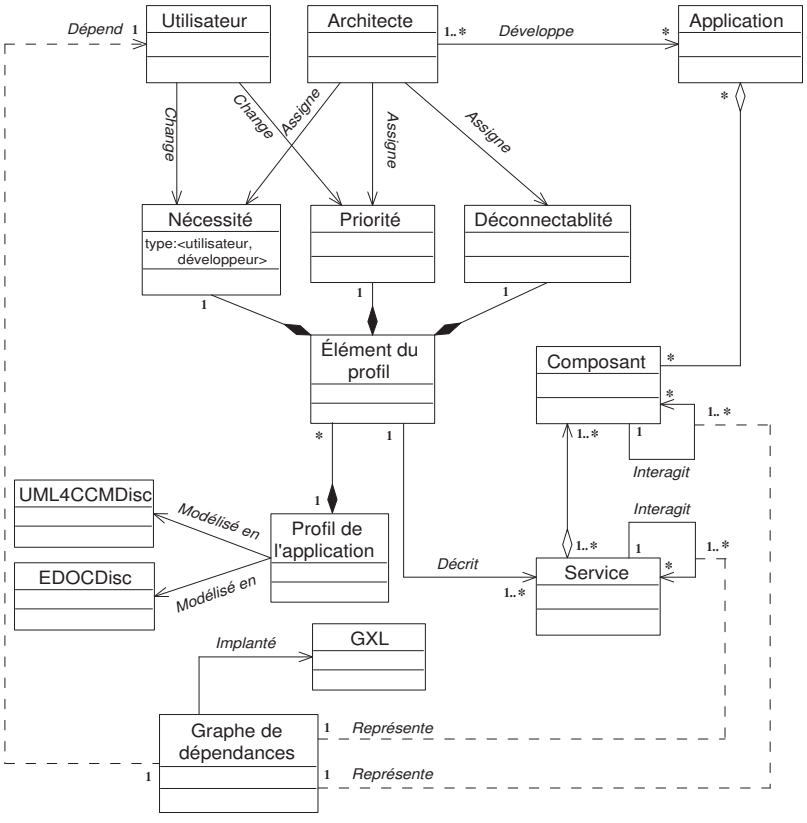


FIGURE 2 – Méta-modèle de MADA

Politiques de déploiement et de remplacement. Nous définissons trois instants d'utilisation de la politique de déploiement. Premièrement, lors du démarrage de l'application sur l'appareil mobile, les

composants déconnectés estampillés comme étant nécessaires par l'architecte et correspondant aux cas d'utilisation déconnectables sont déployés dans le cache de composants déconnectés. Nous supposons que la taille du cache est strictement supérieure à la taille requise par ce premier déploiement. Sinon, l'application ne démarre pas. Deuxièmement, pendant l'exécution, l'utilisateur peut surcharger les méta-données de nécessité et le gestionnaire du cache essaie alors de déployer les composants déconnectés estampillés nécessaires par l'utilisateur. Troisièmement, lors d'une invocation, le gestionnaire du cache essaie de déployer les composants déconnectés correspondant aux composants déconnectables mais non nécessaires. Le déploiement d'un composant déconnecté non nécessaire peut impliquer le déploiement d'autres composants déconnectés dans le cas où des liens de dépendances sont estampillés nécessaires.

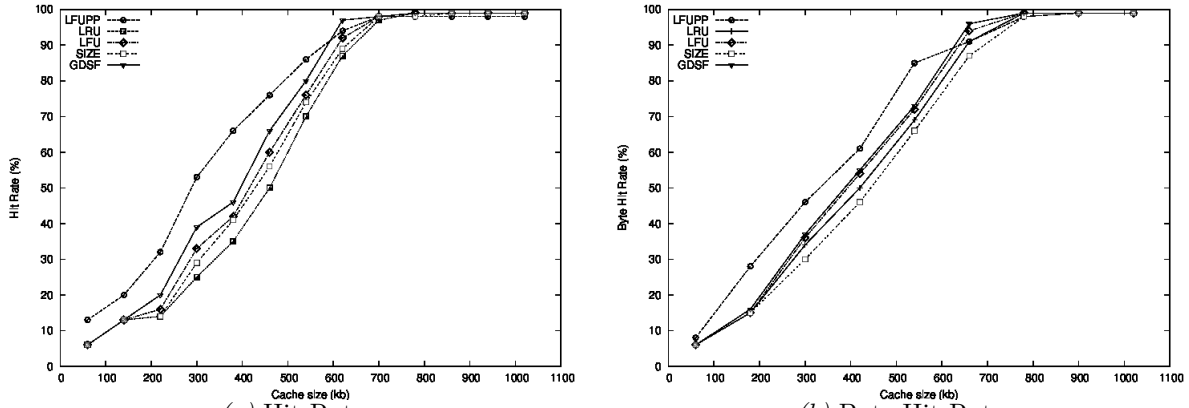
Nous définissons deux cas d'utilisation de la politique de remplacement. Dans le premier cas appelé remplacement à la demande de l'utilisateur, ce dernier retire la nécessité sur des cas d'utilisation pour éviter de provoquer l'éviction de composants déconnectés. Un composant déconnecté est retiré du cache s'il ne reste pas d'opération le concernant dans le journal, c'est-à-dire d'opération non encore exécutée par le composant distant, et si le composant n'est pas nécessaire pour des composants déconnectés qui restent dans le cache. Toujours à la demande de l'utilisateur, un cas d'utilisation qui devient nécessaire peut provoquer l'éviction de composants déconnectés non nécessaires et ayant une priorité moindre. Dans le second cas, périodiquement, le gestionnaire du cache analyse l'espace libre et libère de l'espace pour garder une taille minimale afin d'anticiper les futurs besoins et d'accélérer les futurs déploiements.

Nous proposons une nouvelle politique de remplacement appelée LFUPP pour *Least Frequently Used with Periodicity and Priority*, qui améliore la politique LFU (*Least Frequently Used*). Le gestionnaire du cache calcule la fréquence d'utilisation d'un composant déconnecté en remettant périodiquement la valeur à 0 afin d'éviter la pénalisation due aux scénarios d'utilisation en rafale de façon sporadique. Le composant déconnecté possédant la fréquence la plus basse est retiré, la priorité servant à départager au besoin. LFUPP utilise bien évidemment les méta-données de nécessité avant d'utiliser la fréquence et la priorité.

Le gestionnaire de cache de DOMINT est réalisé avec le modèle de composant Fractal [226] en utilisant le composant Cache manager de Perseus [235] et la librairie NanoXML [233] pour Windows Mobile 2003. Un outil de gestion des méta-données de déconnectabilité, de nécessité et de priorité a aussi été prototypé. Les mesures montrent que le temps nécessaire pour la manipulation du graphe de dépendances avant le lancement du déploiement au pré-chargement est de l'ordre de la milliseconde. Concernant la politique de remplacement LFUPP, nous la comparons à deux politiques traditionnelles, LFU et LRU, et deux autres utilisées dans le domaine des caches Web, GDSF [56] et SIZE [209]. Comme montré dans la figure 3, LFUPP offre une meilleure disponibilité pour les caches de petite taille car elle bénéficie de la connaissance du profil de l'application : dépendances, priorités, granularité fine des composants. Pour les caches de grande taille, toutes les politiques offrent des résultats comparables.

2.4 Bilan

Dans cette section, nous avons montré que le passage de l'orientation objet à l'orientation composant permet une meilleure séparation des préoccupations, et facilite l'identification des points de variation et la gestion des dépendances entre entités pour l'adaptation aux déconnexions. Ces éléments sont importants pour la gestion des extrafonctionnalités qui requièrent la collaboration entre l'application et l'intergiciel. Par ailleurs, la modélisation dans l'architecture des points de variation permet une clarification des rôles des différents intervenants. Dans notre cas, l'architecte définit les points de variation, identifie les



(a) Hit Rate *(b)* Byte Hit Rate

FIGURE 3 – Évaluation de la politique de remplacement de DOMINT

composants déconnectables, et spécifie la politique d'adaptation ; les développeurs mettent en œuvre la politique d'adaptation avec des mécanismes d'adaptation ; et les utilisateurs finaux peuvent indiquer des préférences pendant l'exécution. Ainsi, les différents intervenants manipulent un profil de l'application qui est pris en compte par l'intergiciel pour gérer un cache de composants déconnectés.

La généricité est obtenue en proposant un méta-modèle de la gestion des déconnexions (MADA) et des services génériques (DOMINT). L'utilisabilité a été évaluée dans le cadre des projets ITEA Eureka VIVIAN (2000-2002) et OSMOSE (2003-2005), et du projet RNTL franco-finlandais AMPROS (2003–2005). Plus particulièrement, avec les partenaires du projet AMPROS, nous avons élaboré un scénario de gestion de crise « plan rouge » en Finlande qui utilise des équipements de première urgence possédant des capteurs, par exemple des « sarcophages » pour protéger du froid. La démonstration décrivait la gestion des fiches médicales en situations de déconnexion avec continuité de service et de reconnexion avec réconciliation.

Ce travail a été effectué en collaboration avec Guy Bernard, Sophie Chabridon et Chantal Taconet. Il comprend la contribution de la thèse de Nabil Kouici [127]. Les principales publications sur ce sujet sont les suivantes : ANTe [245], DOA'2003 [259], et IPDPS Workshops'2002 [289] et EDOC Workshop'2002 [288]. Les travaux sur ce sujet ainsi que le développement de la plateforme DOMINT ont été arrêtés en 2006.

Dans cette section, nous avons montré l'apport des intergiciels réflexifs dans l'adaptation des applications pour la continuité de service en situations de déconnexion ou de faible connectivité. Dans la section qui suit, nous présentons comment les applications sensibles à leur contexte d'exécution peuvent détecter ces situations d'adaptation.

3 Sensibilité au contexte des applications réparties autonomiques

L'adaptation pour la continuité de service en situation de déconnexion est une problématique importante de l'informatique mobile. Le passage à l'informatique ubiquitaire renforce le besoin de structurer, par exemple sous la forme de boucles d'autonomie [118], les systèmes répartis pour leur adaptation à des conditions d'exécution variées. Par exemple, la prolifération des dispositifs matériels dispersés dans le monde physique requiert une meilleure structuration du service intergiciel de détection des situations d'adaptation. Nous sommes ainsi passés du problème spécifique de la surveillance de la connectivité réseau pour détecter les déconnexions au sujet plus général de la gestion de contexte. Fonctionnellement, l'architecture d'un gestionnaire de contexte est classiquement organisée en trois couches : collecte des données, perception du contexte et présentation des informations de contexte [61, 68, 214].

Au moment où nous commençons nos recherches sur ce sujet, la gestion de contexte n'est pas identifiée comme une fonctionnalité fournie par un service intergiciel à part entière. En outre, les solutions *ad hoc* existantes sont limitées aux réseaux locaux, c'est-à-dire aux environnements ambiants qui entourent quelques utilisateurs (cf. l'état de l'art dans [17]).

Historiquement, deux approches distinctes existent dans la littérature pour la gestion de contexte : centrée utilisateur et supervision système. Dans l'approche centrée utilisateur, comme identifié dans les états de l'art [148, 101] du milieu des années 2000, les limites des solutions existantes sont le manque de composition aisée des informations de contexte, le passage à l'échelle, tant en termes de quantité d'informations de contexte que de nombre d'applications clientes, et le manque de réutilisabilité. À la même époque, l'approche par supervision qui est ancienne [192], voit un regain d'intérêt avec l'émergence des grappes et des grilles de machines (par exemple, Phoenix [34], LeWYS [47]). La faiblesse de cette approche est le niveau d'abstraction des données collectées, qui restent souvent numériques et sont difficiles à interpréter symboliquement. Nos premiers travaux visent à concilier les approches centrées utilisateurs et supervision pour allier l'utilisabilité et l'expressivité des détections de situations de haut niveau d'abstraction, avec contrôle des dissémination dans les systèmes à large échelle, avec la structuration de l'intergiciel à base de composants pour améliorer la composition des informations de contexte.

La gestion de contexte est un problème traité généralement dans le cadre d'environnements ambiants, c'est-à-dire autour de petits groupes d'utilisateurs co-localisés. Dans le cadre de systèmes impliquant des réseaux ambiants, des nuages, et des objets de l'Internet des objets, la gestion de contexte devient autrement plus complexe [24, 164]. Elle doit prendre en compte l'hétérogénéité des données, répartir les traitements et les flux d'informations, et autoriser la mobilité des éléments de l'architecture. En outre, de nouveaux problèmes sont mis en exergue comme par exemple la mise à disposition d'informations sur le contexte de l'utilisateur qui exposent la vie privée des personnes à proximité des objets connectés à l'Internet. Dans nos seconds travaux présentés dans cette section, nous concevons un service réparti de gestion de contexte basé sur le mode notification (événementiel) mais autorisant aussi le mode observation (par requête), les deux étant nécessaires pour les applications. En outre, nous abordons l'ajout du respect de la vie privée [134] dans les systèmes répartis à base d'événements [74, 151]. Enfin, nous explorons le nouveau paradigme du support de l'aspect multi-échelle [177, 176] pour aborder la distribution des informations de contexte en provenance de l'Internet des objets dans un système à large échelle.

Dans la section 3.1, je présente les exigences du service de gestion de contexte sur lesquelles nos travaux se concentrent. Ensuite, dans les sections 3.2 et 3.3, je développe nos contributions en termes de structu-

ration du service de gestion de contexte et de distribution des informations de contexte, respectivement. Enfin, je conclus dans la section 3.4 par un bilan sur ces travaux.

3.1 Motivations

Par rapport au problème général de la gestion de flux d'informations [63], les besoins particuliers à la gestion des informations de contexte sur lesquels portent nos contributions, sont les suivants [17, 24, 164] : les modes observation et notification, la gestion de la qualité de contexte et du respect de la vie privée, le contrôle de la dissémination dans les systèmes à large échelle et le support de l'aspect multi-échelle, et enfin l'utilisabilité et l'extensibilité.

Modes observation et notification. Le mode d'échange des informations de contexte est de deux types : observation *versus* notification. Dans le mode observation, c'est le client qui initie par une requête l'échange des informations de contexte. Puisque le client contrôle les échanges, il n'est pas dérangé de manière abusive. Mais, en plus de la connaissance de ses besoins en termes d'informations de contexte, le client doit aussi connaître les moments pertinents pour envoyer ses requêtes, cette pertinence étant évaluée par rapport à la probabilité de changements de valeurs des informations de contexte utilisées par l'application.

Dans le mode notification, le client souscrit un abonnement auprès du gestionnaire de contexte pour recevoir ultérieurement les informations de contexte. La notification peut être périodique ou suite à des changements de valeurs. Le client doit donc fournir dans la souscription les informations de contexte utilisées par l'application ainsi que les formules permettant de détecter les changements de valeur. Il en résulte donc une meilleure réactivité. En conclusion, nous estimons que les deux modes sont nécessaires.

Qualité de contexte et respect de la vie privée. L'Internet des objets permet la collecte d'une quantité très importante d'informations de contexte. Mais, ces informations sont connues pour être imparfaites et incertaines [100]. La gestion de la qualité de contexte est la capacité à qualifier les informations de contexte avec des méta-données (de qualité). Les critères de qualité sont nombreux et la liste ne doit pas être figée. À titre d'exemple, les critères les plus courants sont la précision, la probabilité de justesse, la résolution et la fraîcheur. La gestion du respect de la vie privée est vue comme la capacité à contrôler « quoi », « comment », « quand », « où », « avec qui » partager des informations et dans quelles circonstances [134]. Ces deux sujets sont traités par d'autres membres de l'équipe [48, 108]. Notre objectif est donc que ces résultats puissent s'insérer dans nos solutions.

Contrôle de la dissémination et support de l'aspect multi-échelle. Classiquement, le passage à l'échelle est la capacité du gestionnaire de contexte à ne pas avoir des performances qui s'écroulent lorsque le nombre de clients ou la quantité d'informations de contexte augmentent. Quant au paradigme du support de l'aspect multi-échelle [177, 176], il est emprunté aux travaux de membres de l'équipe et de participants au projet INCOME, et plus particulièrement à Sam Rottenberg dans le cadre de ses travaux de thèse en collaboration avec Chantal Taconet, Sébastien Leriche, Claire Lecocq et Thierry Desprats. La définition, que nous développons ci-dessous, du support de l'aspect multi-échelle (en anglais, *multis-calability*) est la suivante : c'est la capacité de couvrir plusieurs échelles dans au moins une dimension. La distribution multi-échelle est différente conceptuellement de la distribution à large échelle. L'aspect

large échelle possède une signification quantitative alors que l'aspect multi-échelle est d'abord attaché à une signification qualitative due à l'hétérogénéité.

Dans des études récentes [32, 82, 120], l'expression « système réparti multi-échelle » est utilisée pour mettre en avant la complexité consécutive à différents types d'hétérogénéité : réseaux parcourus, matériels utilisés, répartition géographique des matériels, organisation des groupes d'utilisateurs, etc. Nous défendons l'idée que certaines sources d'hétérogénéité possèdent un fort impact sur la distribution des informations de contexte en provenance de l'Internet des objets et peuvent aussi être utilisées pour attaquer le problème du passage à l'échelle.

Voici la terminologie utilisée pour la caractérisation de l'aspect multi-échelle des systèmes répartis. Lors de l'étude de l'architecture logicielle d'un système réparti, tout point de vue architectural [111] est potentiellement étudié selon plusieurs dimensions. Une dimension est le mesurage d'une caractéristique et est associée à une mesure numérique ou sémantique. En utilisant la mesure donnée, une dimension peut être divisée en échelles ordonnées. Les auteurs proposent un vocabulaire de points de vue, dimensions et échelles. Par exemple, dans le point de vue « utilisateur », il est intéressant, par exemple pour le respect de la vie privée, de considérer les dimensions « appartenance à un groupe » avec comme échelles les niveaux dans la hiérarchie des groupes d'utilisateur [24]. Dans le point de vue « réseau », il est intéressant d'étudier plusieurs dimensions comme par exemple la « portée de transmission », la « latence » ou le « débit » et de classer les réseaux parcourus dans des échelles différentes (PAN, LAN, MAN, WAN, etc.) [120]. Enfin, dans le point de vue « géographie », la distinction des échelles « adresse », « rue », « quartier », « ville »... dans la dimension « zone administrative » est intéressante pour gérer la pertinence de la dissémination d'informations [82].

Utilisabilité et extensibilité L'utilisabilité est caractérisée par un ensemble d'attributs contribuant à l'effort (englobant la compréhension, l'apprentissage, l'applicabilité, l'application et le contrôle opérationnel des concepts logiques) requis pour l'utilisation d'un logiciel par un ensemble d'utilisateurs avéré ou implicite [110]. Nous incluons ainsi dans l'utilisabilité la possibilité d'exprimer des besoins en informations de contexte, soit en exprimant les informations demandées (l'utilisateur est le client qui demande les informations) soit sous la forme de traitements à effectuer (l'utilisateur est le concepteur de ces traitements). Les méthodologies et l'outillage font aussi partie de la mesure de l'utilisabilité.

L'extensibilité est caractérisée par un ensemble d'attributs contribuant à l'effort demandé pour prendre en compte de nouveaux besoins en termes de conception et d'exécution. Cela inclut la modifiabilité et l'interopérabilité avec d'autres systèmes [110]. En gestion de contexte, cette qualité est recherchée pour l'intégration des technologies de l'Internet des objets et pour la combinaison des différents types de traitement de l'information de contexte (fusion et agrégation d'événements, corrélation d'événements, détection de situation par apprentissage, etc.).

Dans la section qui suit, je commence par la partie locale de l'architecture de gestion de contexte qui prend en compte les exigences que je viens d'introduire. Les aspects liés à la distribution seront étudiés ensuite.

3.2 Structuration de la gestion de contexte

Concevoir la gestion de contexte comme un service de l'intergiciel nous amène à étudier la structuration de ce service, toujours avec les objectifs de généricité et de séparation des préoccupations. Au début

des années 2000, les travaux de la littérature se classent en deux grandes familles de styles architecturaux [211] : l'orientation donnée et l'orientation processus. Dans l'orientation donnée, citons CASS [75], Gaia [173], CoBrA [53], CMF [125], SOCAM [91], et PACE [101]. Les patrons de conception mis en exergue s'appellent « Tableau noir » et « Espace de n-uplets ». La gestion des informations de contexte est centralisée dans une base de données accessible comme un tableau noir ou dans une mémoire virtuelle partagée gérant l'espace comme un ensemble d'enregistrements ou n-uplets. Dans cette orientation, de nombreux travaux s'intéressent à l'expressivité du nommage et des langages de requêtes. Les forces de cette orientation sont la simplicité du paradigme, la facilité d'ajout et de suppression de nouvelles sources et de nouveaux clients, et l'asynchronisme entre producteurs et consommateurs. La faiblesse principale est la gestion centralisée.

Face à l'orientation donnée, l'orientation processus encapsule toutes les acquisitions et transformations d'informations de contexte dans des entités de traitement séparées appelées : « objet » pour Context Toolkit [68], RCSM [214], SAJE [60], Hydrogen [103], MoCA [64], et WildCAT [66] ; ou « composant » pour MoCoA [194], Draco [169], Le Contexteur [62], et DigiHome [175]. L'expression des dépendances et des flots d'informations entre ces entités dépend du paradigme de conception choisi. Le concepteur dessine un graphe (une forêt) de ces entités pour exprimer ses besoins en informations de contexte. Il peut aussi réutiliser des parties de graphes conçus par d'autres. Dans cette orientation, les travaux de la littérature s'intéressent beaucoup à l'encapsulation des entités de traitement et à leur réutilisation. Les forces principales de l'orientation processus sont l'efficience permise par le contrôle des ressources consommées par le graphe et par les échanges directs sans intermédiation. La difficulté principale est la complexité globale de la gestion du graphe.

Ces deux orientations sont de notre point de vue complémentaires. Les solutions de l'orientation donnée supposent que les données sont directement accessibles, et cherchent à détecter des situations les plus complexes possibles. Les solutions de l'orientation processus structurent les traitements en introduisant des possibilités de partage et de répartition. Plus précisément, nous défendons l'idée selon laquelle un gestionnaire de contexte doit être *1)* centré utilisateur pour fournir des informations de contexte de haut niveau aisément interprétables, *2)* construit à partir d'éléments composés plutôt que programmés pour faciliter la conception, et *3)* performant en contrôlant et minimisant l'utilisation des ressources.

Nous proposons un canevas logiciel pour la composition d'informations de contexte dont l'originalité est l'expression de la composition de contexte dans un langage de définition d'architecture logicielle et la projection de cette architecture sur un graphe de composants. Ainsi, nous facilitons la conception, la composition, l'adaptation et la réutilisation des politiques de gestion de contexte.

Architecture à base de composants. L'architecture globale du gestionnaire de contexte est classiquement organisée en trois couches [61, 68, 214]. La première couche de l'architecture est constituée de la collecte des données brutes et a pour rôle d'intégrer les nombreux canevas logiciels utilisés pour les différentes sources d'informations de contexte. La couche de collecte fournit les données brutes à la base de traitements (fusion et agrégation d'événements, corrélation d'événements, détection de situation par apprentissage, etc.) pour obtenir des informations de contexte symboliques de plus haut niveau d'abstraction. Ces traitements constituent la couche de perception du contexte. La troisième couche présente les informations de contexte aux clients, qui sont des applications ou d'autres services intergiciels.

Nous supposons que l'application est construite avec des composants, l'accès aux informations de contexte s'effectuant via le conteneur du composant applicatif. Les couches logicielles collecte, perception,

et présentation correspondent à des métiers divers : des fournisseurs d'informations brutes aux concepteurs d'applications sensibles au contexte. Les cycles de vie des éléments physiques ou logiques observables sont très différents, avec des besoins en ressources système différents. Contrairement aux solutions de la littérature, notre architecture de gestion de contexte est construite à base de cycles indépendants « collecte / perception / présentation ». De tels cycles complets sont présents dans les collecteurs de contexte, plusieurs autres dans les processeurs de contexte, et de même pour le conteneur sensible au contexte.

Pour la conception de notre solution, nous appliquons les principes de base de la construction d'intergiciels : le canevas logiciel est construit à partir d'éléments génériques, des composants, spécialisables et modulaires afin de composer plutôt que de programmer. Nous obtenons ainsi une vision unifiée dans laquelle les mêmes concepts (composant, liaison, interface) sont utilisés pour développer les applications et le service intergiciel. Par ailleurs, la notion d'architecture logicielle associée à celle de composant permet d'exprimer la composition des entités logicielles indépendamment de leur mise en œuvre, rendant ainsi plus aisée la compréhension de l'ensemble. La notion d'architecture logicielle favorise aussi la dynamicité en autorisant la redéfinition des liaisons, facilitant alors la reconfiguration. Notre proposition, COSMOS, intègre donc les objectifs suivants : *a)* composer des informations de contexte de manière déclarative, à opposer aux approches « par programmation » utilisées dans les canevas logiciels existants, *b)* isoler chaque couche de l'architecture de gestion de contexte des autres couches afin de promouvoir la séparation des préoccupations et des cycles de vie, et *c)* fournir les concepts « système » pour gérer finement les ressources consommées par les différents traitements.

Le concept de base de COSMOS est le nœud de contexte, qui manipule une information de contexte modélisée par un composant. Les nœuds de contexte sont organisés hiérarchiquement, avec possibilité de partage. Tous les composants d'une hiérarchie sont potentiellement accessibles par les clients. La figure 4 montre un exemple de graphe de nœuds de contexte. Les propriétés des nœuds sont les suivantes :

- passif ou actif : chaque nœud peut être passif ou actif avec exécution périodique de tâches dans des activités ;
- observation ou notification : les informations de contexte circulant du bas vers le haut de la hiérarchie dans des messages, lorsque la circulation s'effectue à la demande d'un nœud parent ou d'un client, c'est une observation. Dans le cas contraire, c'est une notification ;
- passant ou bloquant : lors d'une observation, un nœud passant demande d'abord un nouveau rapport d'observation à ses enfants, puis calcule un rapport d'observation qu'il transmet. Lors d'une notification, un nœud passant calcule un nouveau rapport d'observation avec la nouvelle notification, puis transmet vers le haut l'information calculée. Dans le cas bloquant, le nœud observé fournit l'information qu'il détient sans observer les enfants et le nœud notifié modifie son état interne sans notifier les parents ;

Pour la conception de graphes de nœuds, nous transposons dans le monde composant certains patrons de conception du monde objet [84]. L'organisation d'informations de contexte sous forme arborescente nécessite d'isoler avec le patron de conception « Composite » les différents sous-arbres afin de faciliter leur composition. Chaque nœud de l'arborescence est construit à partir d'un squelette qui décrit l'assemblage de l'opérateur avec les composants de gestion des ressources (mémoire et activité) et les composants des nœuds fils directs dans la hiérarchie ; c'est le patron de conception « Patron de méthodes ». Le patron de conception « Poids-mouche » est utilisé pour partager de manière performante de nombreux composants. Enfin, grâce au patron de conception « Singleton », la gestion de la mémoire et des activités est effectuée

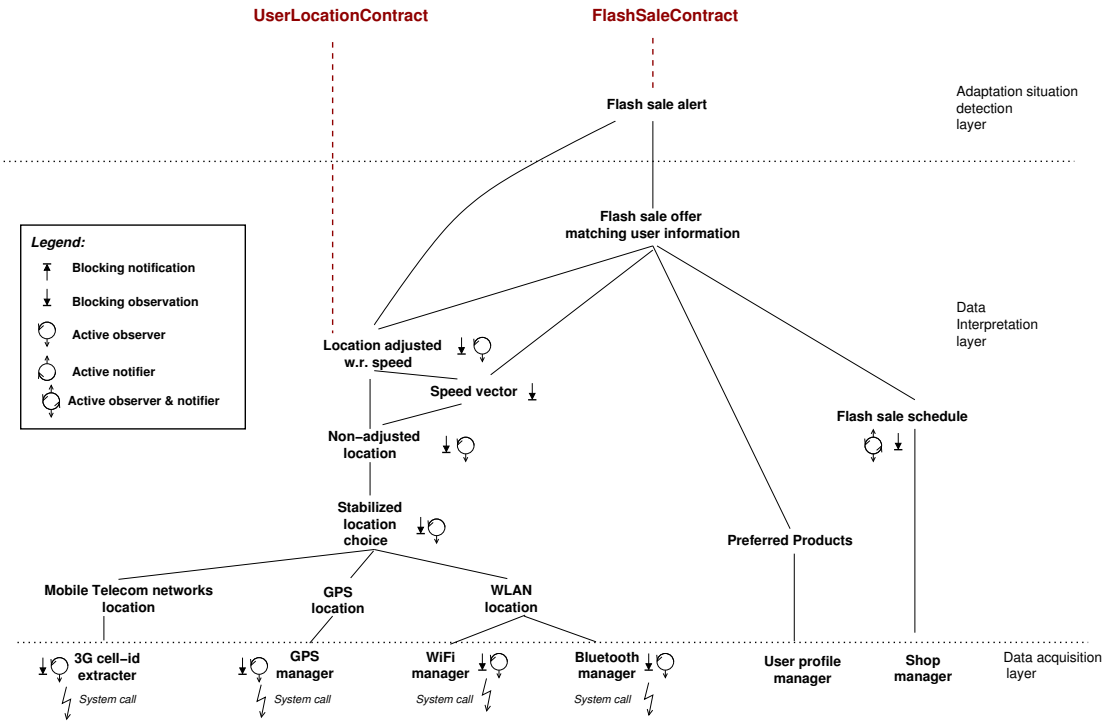


FIGURE 4 – Graphe de nœuds de contexte COSMOS pour le cas d'utilisation « vente flash »

par un nombre très restreint de gestionnaires de messages et d'activités partagés.

Le canevas logiciel COSMOS [219] est mis en œuvre avec le modèle de composant FRACTAL [41, 42] du consortium OW2 [226, 229, 230], auquel est associé le langage de description d'architecture FRACTAL ADL [138, 227]. Nous utilisons également la bibliothèque de composants DREAM [137, 222] pour construire les gestionnaires de messages et d'activités. Pour faciliter la reconfiguration d'une application à base de composants FRACTAL, nous utilisons FPath [65, 224]. L'extensibilité de la solution est démontrée en insérant la préoccupation de gestion de qualité de contexte dans les traitements des nœuds de contexte [48, 242] et en démontrant la complémentarité avec l'approche basée ontologie dans une architecture hybride [253]. Par ailleurs, nous avons utilisé COSMOS dans divers projets pour collecter et interpréter des informations à partir de sondes système sur GNU/Linux, J2ME CLDC et Android, et du simulateur de contexte Siafu [237]. Comme l'une des nouveautés est l'utilisation de composants à granularité très fine, nous évaluons l'utilisation de l'approche à base de composants de petite taille sur des téléphones mobiles. Prenons l'exemple d'un téléphone Samsung Galaxy S2 (Android 2.3.5, ARM Cortex-A9 Dual-core cadencé à $1.2GHz$, $16GB$ de ROM et $1GB$ de RAM, dont $660MB$ libres). La traversée d'un nœud sans traitement est effectuée en $1,76ms \pm 0,06ms$ pour une nœud feuille et en $3,62ms \pm 0,03ms$ pour un nœud non feuille. Enfin, environ 5500 nœuds peuvent être instanciés et la hauteur de l'arbre entre deux nœuds bloquants peut atteindre 45 nœuds.

Méta-modélisation, langage dédié et ingénierie dirigée par les modèles. L'architecture de base de COSMOS est complétée par la modélisation conceptuelle de la notion de contexte [198, 197] et par la définition d'un langage dédié à la conception de politiques de gestion de contexte, le tout mis en action dans une approche d'ingénierie dirigée par les modèles.

Les nœuds de contexte sont conçus pour être hautement configurables via de nombreux attributs (8 par nœud). Par ailleurs, chaque nœud peut être partagé par plusieurs autres nœuds, et ce quelle que soit leur place « au-dessus » dans le graphe. L'inconvénient est la complexité de la configuration : le langage d'architecture Fractal ADL, s'il est très expressif, présente un aspect très technique, verbeux et donc peu sûr, même en utilisant les annotations Fraclet [178, 225]. Nous avons donc conçu le langage dédié COSMOS DSL [220], qui permet de déclarer des nœuds de contexte, de les configurer et de les composer simplement. Ces garanties sont assurées par des constructions de haut niveau qui abstraient la complexité de la définition de l'architecture sous-jacente. La figure 5 présente la déclaration d'une partie du graphe de l'exemple de la figure 4. D'un point de vue pratique, le méta-modèle de COSMOS est écrit en EMF (*Eclipse Modeling Framework*) [223]. Le langage COSMOS DSL et son outillage sont construits à partir du méta-modèle avec Xtext [240]. Enfin, nous avons aussi spécifié formellement en Alloy [212] un système de type de message avec une propriété de substitution de messages [221], mais sans inclure cette vérification de type dans l'outillage.

```
chunk position = {
    classname : evry2mallqoc.PositionChunk,
    typeparam : evry2mallqoc.PositionInfo }
message location = {
    position, accuracy, freshness, trustworthiness }
operator computeGprsLocation = {
    output : location,
    classname : evry2mallqoc.ComputeGprsPosition,
    input : gprs }
node gprsLocation = {
    operator : computeGprsLocation,
    children : gprsExtracter }
node stabilizedLocation = {
    operator : computeStabilization,
    children : gprsLocation gpsLocation wlanLocation }
operator computeNonAdjustedLocation = {
    output : location,
    classname : evry2mallqoc.ComputeNonAdjustedLocation,
    input : location }
node nonAdjustedLocation = {
    operator : computeNonAdjustedLocation,
    children : stabilizedLocation }
operator computeAdjustedLocation = {
    output : location,
    classname : evry2mallqoc.ComputeAdjustedLocation,
    input : location speed }
node adjustedLocation = {
    operator : computeAdjustedLocation,
    children : nonAdjustedLocation speedVector }
configuration : gprsExtracter
    [observethrough=false] [periodobserve=10000]
configuration : stabilizedLocation
    [observethrough=false] [periodobserve=2000]
configuration : nonAdjustedLocation
    [observethrough=false] [periodobserve=3000]
configuration : adjustedLocation
    [observethrough=false] [periodobserve=5000]
```

Figure 5 – Exemple de la figure 4 écrit avec COSMOS DSL

Dans cette section, l'architecture à base de composants considère par défaut qu'un graphe de nœuds est une entité de déploiement, même si des travaux de l'écosystème Fractal permettent de répartir le graphe sur un ensemble de machines. Dans la section qui suit, nous présentons les travaux en cours pour architecturer la distribution des informations de contexte en un système réparti à base d'événements.

3.3 Distribution des informations de contexte

Les travaux décrits dans cette section sont en cours et les premiers résultats sont en train d'être finalisés.

Nous ciblons les applications sensibles aux informations de contexte en provenance de l'Internet des objets. Bien sûr, il est possible d'utiliser tous les paradigmes de distribution existants, en considérant que les informations de contexte ne sont qu'un type particulier d'informations à disséminer. L'approche client–serveur, c'est-à-dire avec une phase de découverte avant l'appel synchrone à l'entité fournissant l'information de contexte est mise en œuvre dans MoCoA [194] (avec le concept d'objet sensible), CoBrA [53] (construit sur un bus CORBA d'agents intelligents du modèle FIPA), HiCon [131] et SPACES [273]

(construit au-dessus de COSMOS avec le paradigme REST). EgoSpaces [115] est construit au-dessus de Lime [150], qui met en œuvre le paradigme des espaces des n-uplets de Linda [88]. MUSIC [106] organise des disséminateurs d'informations de contexte réunis dans un réseau pair-à-pair construit avec JXTA [231] pour l'intermédiation entre les clients et les fournisseurs. Enfin, un dernier ensemble de solutions utilisent le paradigme événement : par exemple, Gaia [174, 173] et Mobile Gaia [195] avec un service de gestion d'événements CORBA, et Pervaho [73] avec JMS.

Parmi les approches intergicielles pour les systèmes émergents de l'Internet des objets, le modèle de communication publier-souscrire [74] qui est mis en œuvre dans les systèmes répartis à base d'événements fait partie des solutions les plus utilisées pour les applications construites selon le paradigme *sense and respond*. Le système réparti à base d'événements s'occupe du routage des notifications entre les producteurs et les consommateurs. Les producteurs sont des clients qui publient des notifications tandis que les consommateurs sont des clients qui réagissent aux notifications transmises par le système. Ce choix des systèmes répartis à base d'événements est judicieux pour les trois propriétés suivantes : *1)* découplage dans le temps entre les fournisseurs et les consommateurs, *2)* découplage dans l'espace, et *3)* découplage de la synchronisation.

Les systèmes répartis à base d'événements sont classés en deux catégories principales selon le type de filtrage utilisé dans les annonces et les souscriptions. Dans le cas du filtrage dits « basés contenu », tout le contenu de la notification est analysable ; il n'y a donc pas lieu d'ajouter des méta-données pour le routage, comme cela est nécessaire pour le filtrage « basé sujet ». Le principe du routage avec le filtrage basé sujet est de compléter les notifications à router par une séquence de mots pris dans un vocabulaire de mots organisé en forêt, et d'effectuer le routage à l'aide d'expressions régulières. Ce dernier type de filtrage est communément mis en œuvre dans les architectures centralisées pour mieux passer à l'échelle en termes de nombre de clients et de notifications : par exemple, le standard AMQP [8], avec des canevas logiciels libres très populaires tels que RabbitMQ [236], propose des facilités dites de *clustering* dans les nuages. Les solutions basées sujet ne sont pas suffisantes et doivent être complétées par des solutions basées contenu pour les raisons suivantes : *a)* les solutions basées sujet requièrent un consensus sur la taxonomie et la hiérarchisation des sujets, ce qui pose problème dans les applications de l'Internet des objets avec des clients très variés et non connus lors de la conception ; *b)* l'expressivité du filtrage basé sujet est faible comparée à celle du filtrage basé contenu. Par conséquent, dans la suite, nous nous intéressons aux solutions avec du filtrage basé contenu.

Deux des classes de solutions parmi celles citées ci-avant réalisent le modèle de communication publier-souscrire : les réseaux logiques de courtiers (en anglais, *brokers*) [102, 151] et les réseaux pair-à-pair [119]. Les solutions répertoriées dans la littérature [119] pour réaliser des systèmes répartis à base d'événements au-dessus de réseaux pair-à-pair et qui utilisent le filtrage basé contenu supposent un modèle de données structurées : les filtres de souscription et d'annonce sont des conjonctions de filtres d'attributs. Comme étudié dans la section 3.2, nous souhaitons autoriser tout type de traitement sur les informations de contexte, y compris l'apprentissage ou d'autres méthodes utilisant des ontologies, qui requièrent un modèle de données semi-structurées à la XML (RDF, OWL, etc.). Par ailleurs, les utilisateurs finaux, via leurs appareils mobiles, sont de plus en plus des fournisseurs d'informations de contexte [109]. Ces utilisateurs risquent d'être très réticents à l'idée que des informations sur leur contexte ambiant soient publiées via des pairs installés à l'autre bout de la planète. Nous devons leur proposer des solutions utilisant le principe de localité, avec des liens entre courtiers déployés au plus près des producteurs et des consommateurs lorsqu'ils sont proches.

Ainsi, nous considérons qu'il est intéressant d'étudier la distribution des informations de contexte en provenance de l'Internet des objets à l'aide de systèmes répartis à base d'événements construits comme des réseaux logiques de courtiers, qui mettent en œuvre du filtrage basé contenu avec un modèle de données semi-structurées. Nous commençons par définir les fonctionnalités de notre solution avant d'aborder le problème spécifique de la distribution à large échelle.

Distribution avec un système réparti à base d'événements. Nous commençons par répondre aux besoins exprimés dans la section 3.1. Premièrement, l'extensibilité se traduit entre autres par la prise en compte des gestionnaires de contexte basés ontologie et l'intégration des multiples canevas logiciels de collecte des informations en provenance de capteurs divers. Une conséquence est le choix du modèle de données semi-structurées XML. Dans ce modèle de données, même en contraignant le développeur à concevoir ses filtres d'annonce et de souscription comme des conjonctions de filtres construits sur des expressions XPATH, ce qui serait en contradiction avec le besoin d'utilisabilité, il nous paraît très difficile d'optimiser le routage en utilisant des propriétés de couverture, d'intersection, ou de fusion de filtres XPATH [151]. Nous utilisons donc dans nos premiers travaux un routage simple qui, pour chaque souscription, organise un graphe orienté vers le courtier d'accès du consommateur, et dans lequel les notifications suivent les arcs vers le consommateur. Cela exacerbe le problème du contrôle de la dissémination dans les systèmes à large échelle, que nous abordons dans un prochain paragraphe.

Deuxièmement, le mode de fonctionnement natif des systèmes répartis à base d'événements étant la notification, nous devons étudier plus particulièrement la mise en œuvre du mode observation nécessaire à la gestion de contexte. Le cas d'usage typique est celui dans lequel la fréquence de publication de l'information de contexte est faible et le consommateur se connecte au système réparti de gestion de contexte pour obtenir une information de contexte et se déconnecter aussitôt. C'est pourquoi, pour des raisons d'utilisabilité et en indiquant que cette fonctionnalité est à utiliser avec beaucoup de parcimonie, nous ajoutons un mode d'observation par requête anonyme avec réponses multiples. L'observation est anonyme car le consommateur ne connaît pas le producteur pour lui adresser sa requête. Le consommateur émet une demande avec un filtre similaire à un filtre de souscription ; la demande est diffusée dans le réseau de courtiers ; chaque courtier garde en mémoire la dernière notification de chaque filtre d'annonce ; un algorithme de vague Écho [193] « ramène » vers le consommateur les notifications qui passent le filtre de la demande. Les réponses sont donc multiples.

Troisièmement, comme expliqué ci-avant, dans un système réparti à base d'événements classique, les filtres d'annonces sont déposés sur le courtier d'accès du producteur et les filtres de souscription sont diffusés dans le réseau logique de courtiers. Ce mode de fonctionnement est le plus intéressant lorsque les producteurs sont plus « stables » que les consommateurs. Afin d'autoriser des consommateurs plus volatiles, nous ajoutons le mode de fonctionnement dual : les souscriptions restent locales aux courtiers d'accès des consommateurs et les annonces sont installées sur tous les courtiers ; les notifications de ce type d'annonces sont diffusées dans tout le réseau de courtiers. Encore une fois, cette caractéristique exacerbe le problème du contrôle de la dissémination dans les systèmes à large échelle. En conséquence, lors de l'annonce ou de la souscription, les clients indiquent si l'opération est locale ou globale. Nous autorisons le cas extrême de souscriptions et annonces locales. Cette situation peut survenir par exemple lorsque le producteur et le consommateur sont dans le même espace physique géré par un courtier unique.

La solution est spécifiée formellement en logique temporelle et mise en œuvre par le canevas logiciel libre muDEBS [232]. Nous avons tout d'abord donné les propriétés du système réparti avec annonce :

(sûreté) *1)* la notification doit passer le filtre d'annonce via lequel elle est publiée ; *2)* un client ne reçoit que les notifications auxquelles il a souscrit ; *3)* il ne reçoit que des notifications qui ont été produites ; *3)* il ne reçoit la notification qu'une seule fois ; et (vivacité) *4)* il reçoit toutes les notifications qui passent son filtre de souscription. Ensuite, nous avons spécifié les propriétés du routage distribué et montré qu'elles respectent les propriétés précédentes. Les propriétés du routage réparti sont les suivantes : (validité locale) *a)* seules les notifications qui passent un filtre d'annonce sont routées ; *b)* seules les notifications qui passent un filtre de souscription sont livrées et le sont immédiatement ; (validité distante monotone ultime) *c)* lorsqu'un consommateur souscrit à un filtre et que ce filtre n'est pas résilié, il existe un instant après lequel toutes les notifications passant ce filtre sont notifiées au consommateur. Nous avons démontré que les algorithmes de muDEBS respectent les propriétés du routage réparti. Enfin, muDEBS utilisent Java nio pour les communications et google-gson [228] pour la sérialisation, et fonctionne donc sur les téléphones mobiles Android.

Qualité de contexte et respect de la vie privée. Comme expliqué dans la section 3.1, des membres de l'équipe abordent dans leurs recherches les problématiques de la qualité de contexte et du respect de la vie privée. Les contributions qui suivent concernent l'élaboration de ces propositions dans un système réparti à base d'événements construit au-dessus d'un réseau logique de courtiers.

L'originalité de l'approche est le traitement conjoint des deux aspects : qualité de contexte et respect de la vie privée. S. Chabridon et coauteurs dans [49] montrent la nécessité d'une solution intégrée reposant sur les notions d'exigence et de garantie. Les producteurs d'informations de contexte expriment des garanties sur la qualité des données qu'ils produisent et des exigences de respect de leur vie privée quant à l'utilisation qui en sera faite. Inversement, les consommateurs expriment des exigences en matière de qualité de contexte et des garanties de respect de la vie privée. Concernant la gestion de la qualité de contexte, notre contribution est d'exprimer les garanties de qualité de contexte dans les filtres d'annonce des producteurs et les exigences dans les filtres de souscription des consommateurs.

Concernant le respect de la vie privée, nous considérons que les courtiers sont sûrs et que les notifications produites sont authentiques. Dans le cas contraire, des mécanismes de confidentialité doivent être ajoutés : par exemple, par encryptage des filtres et des notifications [19]. Dans nos travaux, nous nous focalisons sur le mécanisme du contrôle d'accès. Une architecture intégrant le modèle de contrôle d'accès à base de rôles (RBAC) a été proposée dans [25]. Cependant, comme expliqué dans [49], le modèle RBAC est trop limité pour exprimer des contraintes de vie privée. Des concepts plus complexes tels que l'intention ou le consentement doivent être exprimés. C'est pour cela que les partenaires du projet ANR INCOME ont choisi l'approche du contrôle d'accès à base d'attributs (ABAC pour *Attribute-Based Access Control*) : une politique attachée à une ressource décrit les opérations autorisées ; les demandeurs fournissent des attributs appelés les informations ABAC ; ces attributs et les conditions d'exécution sont analysés avant l'accès à la ressource pour une opération donnée. Dans notre solution, chaque courtier inclut une entité appelée *Policy Enforcement Point* du standard OASIS XACML [213]. Les annonces sont complétées par des politiques d'accès et les souscriptions par des informations ABAC.

Distribution multi-échelle pour le contrôle de la dissémination à large échelle. Dans les systèmes basés contenu réalisés au-dessus de réseaux de courtiers, l'installation des filtres d'annonce et de souscription ainsi que le routage peuvent générer beaucoup de messages et impliquer de nombreux courtiers. Le coût en termes de nombre de messages et de nombre d'opérations de filtrage est une problématique

importante pour les systèmes déployés à une grande échelle et tout spécialement les systèmes hétérogènes impliquant des nuages, des nuages de proximité (en anglais, *cloudlets*), des ordinateurs fixes, des ordinateurs portables, des tablettes, des téléphones, des objets connectés, etc. Par conséquent, une attention particulière doit donc être portée à la capacité d'adresser les systèmes à large échelle.

La modélisation et la caractérisation de l'hétérogénéité des systèmes répartis sont étudiées par des membres de l'équipe qui proposent le canevas logiciel de caractérisation multi-échelle MuSCa [177, 176]. Notre contribution au problème du routage des informations de contexte dans les systèmes à large échelle est construite avec MuSCa. Nous associons le concept système d'échelle (des systèmes répartis multi-échelles [cf. section 3.1]) au concept de portée de distribution (en anglais, le concept de *scope* des systèmes répartis à base d'événements [77, 76]), et nous introduisons l'expression « distribution multi-échelle » (en anglais, *multiscoping distribution*) pour exprimer que la portée de distribution d'une notification est contrainte selon plusieurs critères, c'est-à-dire dans plusieurs zones logiques correspondant chacune à une échelle dans une dimension d'un point de vue de l'architecture répartie. Nous défendons l'idée de limiter le routage dans des portées de distribution : « une portée de distribution est une abstraction qui rassemble un ensemble de clients (producteurs et consommateurs) de telle manière que les notifications des producteurs sont confinées aux consommateurs appartenant à la même portée ; une portée peut être récursivement membre d'autres portées » [76]. Les clients estampillent leurs filtres d'annonce et de souscription avec un ensemble de portées, une par dimension ; une notification publiée par un producteur est visible pour un consommateur si elle est visible dans pour ce consommateur toutes les dimensions. Nous proposons une solution pour spécifier la demande des clients et réaliser le routage correspondant à cette demande.

La figure 6 montre deux graphes de portées de distribution, avec les pseudo-portées $\bot$ (*bottom*) et $\top$ (*top*), dont les rôles sont expliqués plus loin. Les arêtes des graphes modélisent la relation « est une portée enfant de ». Dans chaque dimension, le réseau logique de courtiers est divisé en zones ; chaque zone correspond à une portée de distribution ; chaque courtier connaît quelques portées ; deux zones voisines reliées par une relation « est une portée enfant de » possèdent au moins un courtier en commun ; les courtiers en commun sont responsables des passages entre zones. Pour passer d'une portée à une autre, une notification doit passer un filtre dit de visibilité, qui est aussi basé contenu. Les notifications circulent logiquement dans le graphe d'une dimension en définissant ce que nous appelons un chemin de visibilité avec les contraintes suivantes : lorsque publiée, une notification est visible dans la portée du producteur et transitivement dans les portées parentes ; une notification visible dans une portée est visible dans les portées enfants. Les chemins de visibilité sont donc composés de deux parties, possiblement vides : une partie montante suivie d'une partie descendante. Dans un contexte multi-échelle, c'est-à-dire avec plusieurs dimensions ayant chacunes leur graphe de portées de distribution, une notification est visible pour un consommateur s'il existe un chemin de visibilité entre le producteur et le consommateur dans chacune des dimensions.

Puisqu'il n'est pas raisonnable d'obliger les clients à indiquer dans leurs annonces et leurs souscriptions une portée pour chaque dimension du système, afin de permettre de faire évoluer le nombre de dimensions, et pour des raisons de compatibilité avec les systèmes sans concept de portée, nous utilisons les pseudo-portées $\bot$ et $\top$ comme suit. Un client qui indique $\bot$ dans une annonce déclare que, pour les dimensions pour lesquelles il n'a pas spécifié de portée, il ne contraint pas la distribution des informations de contexte. Un client qui indique $\top$ dans une souscription souhaite être notifié sans contrainte de visibilité dans les dimensions pour lesquelles il n'a pas spécifié de portée.

Avec l'aspect multi-échelle, le routage réparti fait intervenir deux types de routage : le routage classique

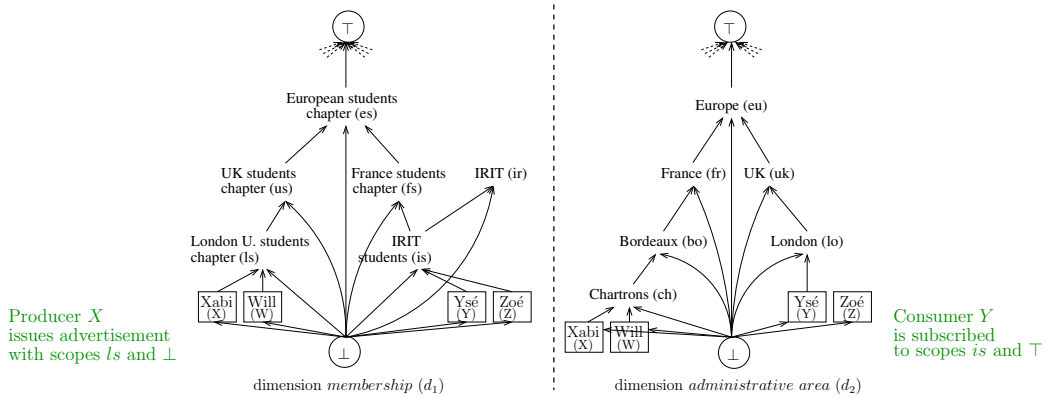


FIGURE 6 – Exemple de graphe de portées de distribution de deux dimensions

de courtier en courtier qui construit des chemins dits de routage entre les producteurs et les consommateurs, et le routage dans les graphes de portées de distribution qui définit des chemins dits de visibilité entre les producteurs et les consommateurs. Le routage « complet » construit des chemins dits de transition entre producteurs et consommateurs qui sont des concaténations de morceaux de chemins de routage et de visibilité.

Les propriétés d'un système réparti multi-échelle à base d'événements sont les suivantes : (sûreté) *1)* la notification doit passer le filtre de l'annonce via laquelle elle est publiée ; *2)* un client ne reçoit que les notifications auxquelles il a souscrit ; *3)* il ne reçoit que des notifications qui ont été produites et qui sont visibles ; *3)* il ne reçoit la notification qu'une seule fois ; et (vivacité) *4)* il reçoit toutes les notifications qui passent son filtre de souscription et qui sont visibles. Les propriétés du routage distribué multi-échelle sont les suivantes : (validité locale) *a)* seules les notifications qui passent un filtre d'annonce sont routées ; *b)* seules les notifications qui passent un filtre de souscription sont notifiées et le sont immédiatement ; (validité distante monotone ultime) *c)* les tables de routage construisent des chemins de routage corrects des producteurs vers les consommateurs ; et *d)* les tables de routage construisent des chemins de visibilité corrects des producteurs vers les consommateurs. Nos algorithmes sont écrits et prouvés, et mis en œuvre dans muDEBS. Nous réalisons actuellement une première campagne d'évaluation des performances.

Le concept de portée de distribution a été introduit dans les réseaux logiques de courtiers par L. Fiege et coauteurs [76] et par H. Parzyjegla dans [162], mais les exigences du routage réparti n'étaient ni explicites ni formalisées. En outre, notre solution n'impose pas de spécifier les portées de toutes les dimensions sur toutes les annonces et toutes les souscriptions. Dans [76, 162], seuls les principes des algorithmes sont donnés, et sans idée de preuve. Par ailleurs, les graphes de portées de distribution ressemblent beaucoup à des hiérarchies de sujets des filtres basés sujet et les auteurs ne donnent pas d'indication précise sur la manière de projeter les portées sur le graphe des courtiers. Plus généralement, l'idée de limiter la portée de distribution apparaît dans quelques gestionnaires de contexte de la littérature, mais de façon plus limitée et moins générique. Dans HiCon [131], l'architecture possède seulement trois niveaux : personnel, régional et global. Mobile Gaia [195] regroupe des nœuds dans des groupes spontanés et organise la distribution des informations de contexte dans ces groupes, mais sans possibilité de passage d'un groupe à l'autre.

3.4 Bilan

Nous avons présenté nos contributions sur un service intergiciel de gestion de contexte pour les applications sensibles aux informations de contexte en provenance de l'environnement ambiant et de l'Internet des objets. Le gestionnaire de contexte est construit en deux temps : les entités déployées sont des graphes de nœuds de contexte, qui sont des composants, et sont construites avec le canevas logiciel COSMOS ; et la distribution des informations de contexte entre graphes est prise en charge par muDEBS, qui est basé sur un réseau logique de courtiers réalisant un système réparti à base d'événements pour modèle de données semi-structurées avec filtrage basé contenu et routage simple.

Concernant les besoins initialement exprimés en section 3.1, les modes observation et notification sont nativement fournis par les interfaces des nœuds de contexte de COSMOS. Pour obtenir le mode observation dans muDEBS, nous ajoutons la notion de requête anonyme synchrone avec réponses multiples. La qualité de contexte est obtenue en complétant le modèle de contexte avec des méta-données de qualité de contexte. La solution mise en œuvre pour la gestion du respect de la vie privée repose sur un contrôle d'accès basé attributs mis en forme avec XACML. Nous proposons le concept de multiples portées de distribution (en anglais, *multiscoping*) afin d'adresser les systèmes à large échelle avec de nombreux clients et courtiers. Pour une meilleure utilisabilité, dans COSMOS, le développeur bénéficie du langage dédié COSMOS DSL. Dans muDEBS, nous fournissons la possibilité d'écrire des filtres complexes (en JavaScript) sur des schémas XML. Remarquons que le lien entre COSMOS DSL et les contrats de muDEBS serait intéressant à effectuer. Enfin, pour l'extensibilité, les deux canevas logiciels sont dotés d'interfaces et sont utilisables sur les systèmes d'exploitation des téléphones mobiles.

Ce travail a été effectué en collaboration avec de nombreuses personnes. Les travaux sur COSMOS ont démarré avec Romain Rouvoy et Lionel Seinturier dans le cadre d'un séjour d'étude dans l'équipe INRIA ADAM (devenue SPIRALS) de janvier à juin 2006. Les travaux ont ensuite été menés avec Sophie Chabridon (qualité de contexte), Chantal Taconet (modèle de contexte et ingénierie dirigée par les modèles), Léon Lim (stage M2 au premier semestre 2008 sur COSMOS DSL), et Cong Kinh Nguyen (CDD ingénieur de novembre 2008 à décembre 2010 sur Android et COSMOS DSL). COSMOS a été développé et utilisé dans le cadre du projet FUI du Pôle de Compétitivité Industries du Commerce CAPPUCINO (2007–2010), de l'ARC INRIA BROCCOLI (2008–2009), du projet Incitatif Institut Télécom CASAC (2008), et du projet TOTEM du Programme Inter Carnot-Fraunhofer (2009–2012). Les principales publications sur ce sujet sont les suivantes : SCICO (2013) [242], TSI (2008) [244], IEEE DSOnline (2008) [243], DAIS'2007 [257], DOA'2009 [255], ICDIM'2010 [253], et DAIS'2011 [251].

Le travail sur muDEBS s'effectue en collaboration avec Sophie Chabridon (TSP), Thierry Desprats (IRIT), Thierry Laborde (IRIT), Samer Machara Marquez (TSP) et Pierrick Marie (IRIT) sur la qualité de contexte et le respect de la vie privée, Chantal Taconet, Claire Lecocq et Sam Rottenberg pour la caractérisation de systèmes répartis multi-échelles, et Léon Lim dans le cadre de son post-doctorat (octobre 2013–août 2015) sur la distribution multi-échelle à base de multiples portées. muDEBS est développé et utilisé dans le cadre du projet ANR INCOME (2012–2015). Les premières publications sur ce travail sont les suivantes : M4IoT'2014 [276], et MW4NG'2014 [277].

4 Tolérance aux fautes des applications réparties en environnement mobile

Dans nos travaux sur le thème de la tolérance aux fautes, nous nous focalisons sur la détection des détériorations physiques [135], c'est-à-dire des fautes matérielles (pas de fautes dans le logiciel), opérationnelles (pas de fautes dans le développement), non malicieuses (dans l'objectif), non délibérées (dans l'intention), et permanentes comme transitoires. Nous considérons les réseaux sans fil les plus dynamiques, c'est-à-dire sans infrastructure : les réseaux mobiles spontanés (en anglais, *Mobile Ad hoc NETWorks* ou MANETs). Les MANETs sont constitués de nœuds mobiles auto-organisés. Les systèmes conçus au-dessus des MANETs sont dynamiques : les nœuds peuvent rejoindre et quitter le système aussi arbitrairement que rapidement. De plus, les défaillances, les déconnexions et les mouvements des nœuds peuvent isoler un nœud ou un groupe de nœuds du reste des participants de l'application répartie. Les études expérimentales confirment la fréquence importante de partitionnements du réseau MANET [96]. En plus du partitionnement du réseau provoqué par des événements imprévus, les applications doivent également supporter les opérations de déconnexions volontaires. Par exemple, un participant peut volontairement décider de se déconnecter afin d'exécuter les applications en local.

Dans nos premiers travaux présentés dans cette section, nous considérons comme dans [184] que les déconnexions et les partitionnements sont transitoires, mais peuvent durer suffisamment longtemps pour gêner les utilisateurs qui souhaitent continuer « à travailler », au risque d'avoir à gérer ultérieurement de potentiels conflits lors des réconciliations. Dans cette problématique de la réplication optimiste, il est intéressant de faire si possible la différence entre les déconnexions et les défaillances. En effet, dans une approche collaborative de l'adaptation aux déconnexions, l'utilisateur participe à l'adaptation en changeant son comportement pendant ses phases de déconnexions, voire lorsque d'autres participants sont déconnectés. Nous proposons une architecture constituée d'un détecteur local du mode de fonctionnement de l'appareil mobile et de détecteurs répartis non fiables pour les défaillances, les déconnexions et les partitions. Ces détecteurs permettent la mise en œuvre d'une des primitives de communication fondamentales pour la tolérance aux fautes dans les systèmes répartis partiellement asynchrones : le consensus. Le consensus permet à un ensemble de processus, chacun possédant sa propre valeur initiale, de décider ultimement une valeur unique [78]. Le détecteur de partition peut aussi servir de détecteur de participants [45] pour un service de gestion de groupe dit partitionnable [7, 57] dans laquelle chaque participant appartient à une partition et peut continuer « à travailler » avec les participants de sa partition. Ainsi, plusieurs partitions disjointes évoluent concurremment et indépendamment les unes des autres.

Avec le détecteur de partition des premiers travaux présentés dans cette section, le système reste instable tant que les partitions ne sont pas « étanches », c'est-à-dire tant que des participants de partitions différentes arrivent par intermittence à échanger des messages. C'est pourquoi cette gestion de groupe partitionnable construit des composantes dites complètement stables [57] dans lesquelles les participants mutuellement atteignables sont isolés des participants des autres composantes stables. Mais, S. Pleish et coauteurs montrent dans [168] qu'une spécification de gestion de groupe partitionnable pour composantes complètement stables permet des mises en œuvre autorisant l'installation de vues capricieuses (en anglais, *spurious views*) : l'installation de vues capricieuses correspond au fait qu'un ensemble de participants est autorisé à installer une nouvelle vue à n'importe quel instant et sans aucune raison en supprimant de manière arbitraire des participants corrects et mutuellement connectés. En d'autres termes, ces installations de vues ne sont pas justifiées car elles ne reflètent pas des événements qui se sont produits dans

l'environnement. La conséquence est que le service de gestion de groupe partitionnable n'aide pas à obtenir des informations correspondant à la réalité de l'environnement. Ainsi, la spécification de gestion de groupe partitionnable doit faire face aux deux exigences antagonistes suivantes [9] : *1)* être forte pour faciliter la conception des applications réparties dans les systèmes partitionnables et *2)* être faible pour être résoluble. Dans [168], les auteurs montrent que les spécifications existantes (par exemple, celles de G. Chockler et coauteurs [57], dont se rapprochent nos premiers travaux, et de Ö. Babaoğlu et coauteurs [15]) ne satisfont pas ces deux exigences. Dans nos travaux, nous proposons un modèle de système réparti construit au-dessus des MANETs, puis une spécification, à notre connaissance la première spécification, qui satisfait les deux exigences pré-citées. Ensuite, nous construisons un service de gestion de groupe partitionnable comme une séquence de consensus abandonnables qui met en œuvre cette spécification.

Dans la section 4.1, je présente les éléments structurants et motivants de nos travaux. Ensuite, dans les sections 4.2 et 4.3, je développe nos contributions en termes de détection de déconnexion, de défaillance et de partition, puis de détection de participants et de gestion de groupe partitionnable.

4.1 Motivations

Les travaux sur les mécanismes de détection que nous développons sont contraints par un certain nombre de résultats d'impossibilité et s'intéressent à la propriété de dynamicité, qui est intrinsèque aux environnements mobiles.

Impossibilité du consensus et de la gestion de groupe, et théorème CAP. Dans un système réparti asynchrone, c'est-à-dire dans lequel il n'existe pas de borne sur la vitesse de calcul des processeurs ou sur le temps de transmission des messages, il est impossible de résoudre le problème du consensus entre processus répartis dès qu'un seul d'entre eux peut défaillir par arrêt franc [78]. La raison est que nous ne pouvons pas dire si un processus est défaillant ou simplement très lent à s'exécuter et à répondre. Dans les systèmes de communication par passage de messages, les travaux de la littérature s'attaquent au problème du consensus de trois manières différentes : par des algorithmes non déterministes qui cherchent à obtenir un consensus avec une probabilité égale à 1 [35, 12], par le renforcement des hypothèses sur l'asynchronie du système jusqu'à permettre la résolution du consensus [70, 4, 5], et ou abstraction des conditions de synchronie dans des détecteurs de défaillance non fiables qui permettent aux processus non défaillants de s'organiser entre eux pour atteindre le consensus [52]. Nous nous limitons aux algorithmes déterministes et nous supposons que le modèle de système réparti est partiellement synchrone [70] tel que les délais de calcul et de transmission ne sont pas connus, mais il existe un instant et des bornes tels que, après cet instant, les délais sont inférieurs à ces bornes. Dans les premiers travaux que je présente dans cette section, nous complétons les détecteurs de défaillance non fiables avec des détecteurs de déconnexion et des détecteurs de partition. Concernant les détecteurs de défaillance non fiables, deux propriétés les caractérisent : la complétude et la précision. De façon générale, la complétude définit les exigences en termes de défaillances détectées, tandis que la précision définit les exigences en termes de fausses suspicions.

Contrairement aux systèmes de partition primaire, dans les systèmes dits partitionnables, plusieurs partitions évoluent concurremment et indépendamment les unes des autres [7, 57]. En autorisant les opérations de fusion et de séparation ainsi que l'exécution concurrente dans des groupes distincts, c'est-à-dire puisqu'elle ne requiert aucun accord « global », la gestion de groupe partitionnable échappe au résultat d'impossibilité en cas de défaillance par arrêt franc d'un seul processus [51]. Par ailleurs, selon le théorème

CAP (pour *Consistency*, *Availability* et *Partition-tolerance*) [81, 36, 90], un système réparti asynchrone sujet à des défaillances par arrêt franc ne peut pas fournir en même temps les trois propriétés suivantes : *1)* la cohérence (le « C ») stipulant que les données sont cohérentes et à jour, *2)* la haute disponibilité (le « A ») imposant que le service est toujours disponible, et *3)* la tolérance au partitionnement du réseau (le « P ») impliquant que l'application répartie progresse malgré le fait que certains messages peuvent être perdus à cause d'un partitionnement du réseau. Si le système est partiellement synchrone alors il est possible d'assurer deux des trois propriétés. Dans notre étude, puisque le partitionnement du réseau est une caractéristique intrinsèque des MANETs, le « P » doit être assuré tout en fournissant un compromis [4] entre le « C » et le « A » : les groupes de participants évoluent de façon autonomes et continuent à offrir autant de services que possible. Pour cela, il est nécessaire d'avoir un mécanisme (ou service intergiciel) qui organise de manière explicite les participants dans des groupes afin de réduire les effets des partitionnements du réseau sur le « C » et le « A ». C'est le service de gestion de groupe partitionnable.

Dynamicité Les mouvements des nœuds et la variabilité de la qualité des communications sans fil modifient la topologie et rendent le graphe dynamique. Sur une période de temps donnée, nous distinguons trois types de liens de communication : *a)* ultimement actif, *b)* ultimement inactif et *c)* actif par intermittence (pour toujours). Un lien ultimement actif transporte ultimement les messages sans perdre aucun d'entre eux. Un lien ultimement inactif arrête ultimement de transporter des messages. Enfin, les messages transportés par un lien actif par intermittence peuvent être perdus. Les liens actifs par intermittence sont la source de l'instabilité du système. Cela se traduit par le fait que deux participants sont mutuellement atteignables seulement par intermittence. Dans le pire des cas, aucune exécution utile ne peut être terminée si deux participants ne sont pas mutuellement atteignables aux moments où ils tentent d'échanger des messages.

Par ailleurs, dans notre seconde contribution présentée dans la section 4.3, nous adressons les applications réparties avec un ensemble de participants non connu au début de l'exécution et qui évolue tout au long de l'exécution. Ainsi, contrairement aux systèmes statiques, les participants ne se connaissent pas nécessairement les uns les autres avant de se rencontrer. Chaque participant est cependant identifié de manière unique. Nous considérons le modèle d'arrivée infinie avec accès simultané borné (en anglais, *infinite arrival model with bounded concurrency*) [145, 146] : sur une période de temps finie, seul un nombre fini de participants (ou nœuds) exécutent l'application répartie, mais le nombre total de participants dans une exécution peut cependant croître à l'infini au fil du temps. Autrement dit, chaque exécution possède un niveau d'accès simultané fini mais inconnu.

Enfin, dans les MANETs, les dispositifs mobiles communiquent entre eux en diffusant des messages qui sont reçus par les voisins directs qui se trouvent dans leur portée de transmission. Les liens de communication sont unidirectionnels. La communication entre deux nœuds non voisins est possible via des chemins qui sont modélisés par des séquences de liens sans fil établis dynamiquement entre les différents nœuds du réseau.

Des travaux de la littérature tels que [147, 11, **?**] considèrent ce type de dynamicité avec l'ensemble des participants inconnus, des communications sans fil avec une primitive diffusion aux voisins, et une mobilité des participants. Nous y ajoutons la tolérance aux partitions sans présence d'une partition primaire.

4. Les travaux sur la réplication optimiste [184] existe précisément à cause de l'impossibilité d'avoir le « C » et le « A » en cas de partitionnement.

4.2 Détections de déconnexion, de défaillance et de partition

Le mode de fonctionnement d'un appareil mobile est établi par ce que nous appelons un détecteur de connectivité. Il donne des informations sur une ressource telle que la bande passante ou la batterie pour déterminer localement si l'appareil mobile peut ou non utiliser telle ou telle interface réseau. La prédiction du niveau de ressource de l'appareil mobile joue un rôle clé dans l'établissement du mode de fonctionnement. Dans la problématique de gestion de la carence des ressources de communication, la prédiction permet d'anticiper les besoins de chargement du cache en prévision de déconnexions (cf. section 2). Par contre, dans une situation de très bonne connectivité, elle pourrait être utilisée pour décider le lancement d'exécutions distantes (en anglais, *off-loading*) afin de préserver les ressources locales (processeur, batterie, mémoire, etc.). Nous proposons une solution de surveillance locale du réseau sans fil et du niveau de la batterie, qui sont les deux causes physiques des déconnexions, pour établir le mode de fonctionnement d'un dispositif mobile en évitant les phénomènes d'instabilité.

Les informations de déconnexion et de reconnexion du détecteur de connectivité restent locales. Une information de déconnexion globale est souhaitable afin d'en tenir compte dans les solutions de recouvrement suite aux déconnexions. Dans le mode faiblement connecté, le dispositif mobile n'est peut-être pas capable d'envoyer des messages applicatifs, mais nous pouvons essayer d'envoyer des messages de contrôle. Ainsi, nous introduisons, dans l'architecture de détection, un détecteur de déconnexion qui échange les informations sur les événements de déconnexion et de reconnexion avec les autres détecteurs de déconnexion.

Avec l'utilisation des réseaux sans fil, les partitionnements du réseau deviennent fréquents à cause des déconnexions. Afin de limiter l'impact d'un partitionnement, les participants dans une partition peuvent mettre en place un mode dégradé des applications réparties ; la distinction des causes de non-présence (défaillance ou déconnexion) d'un participant dans une partition est une information potentiellement intéressante pour organiser ce mode dégradé. Pour faciliter le travail autonome des participants d'une partition autonome complètement stable, nous proposons un détecteur de partition non fiable qui assure les propriétés suivantes : un processus correct détecte ultimement tous les processus qui n'appartiennent pas à sa partition et suspecte ultimement tous les processus corrects hors de sa partition.

Je présente maintenant les détecteurs les uns après les autres en commençant par le plus élémentaire, le détecteur de connectivité.

Détection de connectivité. Le détecteur de connectivité est l'entité donnant les informations sur une ressource telle que, si le niveau de disponibilité de cette ressource est insuffisant, le processus doit être déconnecté. Des exemples de ressources à contrôler pour la connectivité sont le niveau de la batterie de l'appareil mobile et le pourcentage de la bande passante du réseau sans fil.

L'objectif du détecteur de connectivité est de stabiliser l'application répartie afin de minimiser le nombre de transferts d'états. L'effet « ping-pong » intervient si le niveau de disponibilité de la ressource oscillant autour d'une valeur frontière (seuil) entre deux modes de connectivité, provoque des changements répétés. Dans [97], R. Harbus note un effet similaire qu'il appelle « effet boomerang » dans le contexte de la migration de processus. Nous cherchons donc à obtenir une propriété de *prévention contre l'effet « ping-pong »* : un niveau de disponibilité de ressource qui oscille autour d'un seuil provoque un seul changement de mode.

La solution classique au problème de l'effet ping-pong est d'introduire un double seuil. Dans notre cas, puisque l'effet intervient dans deux situations (déconnexion et reconnexion), un mécanisme d'hystérésis

comprenant deux doubles seuils est nécessaire. Le détecteur de connectivité utilise un mécanisme d'hystérésis pour lisser les variations du niveau de disponibilité d'une ressource. En conséquence, est introduit le mode partiellement connecté, ajouté aux modes connecté et déconnecté. Pratiquement, dans le mode partiellement connecté, les messages sont transmis aux mandataires présents localement, puis aux processus distants (cf. section 2).

Détections de défaillance, de déconnexion et de partition. Nous considérons un système réparti partiellement synchrone dans lequel il existe des bornes sur les délais d'exécution des processeurs et de transmission de messages, mais ces bornes ne sont pas connues et n'existent qu'à terme, c'est-à-dire après une durée globale de stabilisation (en anglais, *Global Stabilisation Time*). La topologie du graphe change suite aux mouvements des nœuds, aux défaillances et aux déconnexions, mais ce sont toujours les mêmes processus. Puisque les dispositifs mobiles consomment plus d'énergie pour émettre que pour recevoir un message, nous considérons que les liens entre processus sont unidirectionnels.

Les processus et les liens peuvent défaillir par arrêt franc. Par définition, les processus non défaillants sont dits corrects. Les liens corrects sont supposés équitables : un lien équitable peut perdre des messages mais lorsque l'émetteur émet le même message un nombre non borné de fois à un récepteur correct, alors le récepteur reçoit le message un nombre non borné de fois. En outre, puisque dans ces premiers travaux nous ciblons les applications collaboratives mettant en œuvre la réplication optimiste pour tolérer les déconnexions et les partitionnements, nous supposons comme dans [163] que les processus corrects commencent et terminent en mode connecté et qu'ils ne défaillent pas pendant les déconnexions.

Nous équipons chaque nœud de trois détecteurs : le détecteur de défaillance non fiable à base de battements de cœur $\mathcal{HBFD}$, le détecteur de déconnexion à base de vecteurs $\mathcal{VBDD}$ et le détecteur de partition ultimement parfait $\mathcal{EPPD}$. Le détecteur de défaillance non fiable $\mathcal{HBFD}$ est inspiré de la classe des détecteurs de défaillance non fiables proposée par M.K. Aguilera et coauteurs dans [2, 3]. L'argument principal pour ce choix est la nécessité de construire des algorithmes dits silencieux, c'est-à-dire qui arrêtent d'émettre des messages à terme même en cas de partitionnement du réseau. Nous avons besoin de cette propriété pour le détecteur de déconnexion. Une autre raison justifiant ce choix est que ces détecteurs de défaillance n'utilisent pas de temporisation. Dans notre mise en œuvre algorithmique, $\mathcal{HBFD}$ fournit en plus des compteurs de battements de cœur des informations sur la topologie en gardant la liste des processus accessibles via les voisins. Pour chaque voisin, nous avons ainsi la liste des processus atteignables en passant par ce voisin. La propriété d'atteignabilité est exprimée comme suit : pour chaque processus p, pour chaque processus voisin r, l'ensemble des processus atteignables par r contient ultimement tous les processus corrects, par exemple q, tel qu'il existe un chemin équitable de p vers q via r et un chemin équitable de q vers p via r. Par ailleurs, $\mathcal{HBFD}$ accepte des requêtes de ré-initialisation à l'ensemble vide des ensembles de processus atteignables par les voisins, ceci afin de redémarrer une phase de découverte de la topologie. Cette fonctionnalité est utilisée par notre détecteur de partition $\mathcal{EPPD}$. Les propriétés du détecteur de défaillance non fiable $\mathcal{HBFD}$ sont les suivantes :

— complétude : sur chaque processus correct p, la valeur du compteur de battements de cœur d'un processus hors partition est bornée ;

— précision : sur chaque processus correct p, les compteurs de battements de cœur ne décroissent pas et les valeurs des compteurs des processus corrects dans la même partition sont non bornés.

Contrairement aux défaillances qui sont complètement imprévisibles, les déconnexions qui sont la conséquence d'un faible niveau de connectivité, autorisent l'essai d'émission de messages de contrôle

même si la qualité du lien réseau sans fil est mauvaise. Ainsi, durant la courte période entre le mode connecté et le mode déconnecté, nous pouvons essayer d'alerter les autres processus. En d'autres termes, nous bénéficions du détecteur de connectivité pour « voir venir » une déconnexion et pour stabiliser le système (cf. la propriété de prévention contre l'effet « ping-pong »). En outre, lorsque l'utilisateur se déconnecte volontairement, nous prenons le temps d'informer les autres processus. Cependant, lors d'une déconnexion soudaine, le processus est suspecté de défaillance ; cette fausse suspicion est corrigée lors de la reconnexion. $\mathcal{VBDD}$ fournit un vecteur de compteurs d'opérations de déconnexion et de reconnexion, une entrée par processus. Les déconnexions sont numérotées avec des numéros impairs et les reconnexions avec des numéros pairs. La diffusion des événements de déconnexion et de reconnexion est effectuée à l'aide d'une primitive réalisant un lien têtu [93] : si l'émetteur correct émet un message vers un récepteur correct et que l'émetteur retarde indéfiniment l'envoi du message suivant, alors le récepteur recevra ultimement le premier message. Par ailleurs, grâce à $\mathcal{HBFD}$, la mise en œuvre algorithmique de cette abstraction est silencieuse. Les propriétés de $\mathcal{VBDD}$ sont les suivantes :

— complétude : ultimement, tous les événements de déconnexion et de reconnexion d'un processus sont connus par tous les processus corrects dans sa partition ;

— précision : aucun processus ne connaît l'événement de déconnexion (respectivement, reconnexion) avant la déconnexion (respectivement, déconnexion) effective du processus concerné.

Par-dessus les détecteurs de défaillance et de déconnexion, le détecteur de partition ultimement parfait $\mathcal{EPPD}$ calcule l'ensemble des processus dans la même partition. $\mathcal{EPPD}$ est caractérisé par les propriétés qui suivent :

— complétude forte : si un processus q reste inatteignable pour toujours pour un processus correct p, alors ultimement p suspectera pour toujours q de ne pas être dans sa partition ;

— précision forte ultime : si un processus q reste atteignable pour toujours pour un processus correct p, alors ultimement p ne suspectera plus q de ne pas être dans sa partition.

$\mathcal{EPPD}$ se rapproche du détecteur $\Diamond P$ proposé par G.V. Chockler et coauteurs dans [57], avec en plus la possibilité de différencier les causes de non-atteignabilité, la prise en compte de liens unidirectionnels et enfin la propriété de silence. $\mathcal{EPPD}$ réalise une partie du détecteur de participants pour partitions réseau introduit par D. Cavin et coauteurs [45] : l'ensemble des participants est l'ensemble des processus atteignables et un consensus est relancé lorsque cet ensemble change ; la seconde propriété du détecteur de participants, qui est appelée *information inclusion* par les auteurs, n'est pas prise en compte dans notre proposition car nous supposons que l'ensemble des processus Π est connu au démarrage du système ; cette restriction est levée dans les travaux qui suivent.

4.3 Détection de participants et gestion de groupe partitionnable

Après la conception d'un service de détection de partition complètement stable dans les environnements à mobilité faible, nous nous intéressons à la conception d'un service de gestion de groupe partitionnable pour environnements très dynamiques. La gestion de groupe partitionnable est un service intergiciel de base pour les systèmes répartis tolérant les partitionnements du réseau.

Les spécifications existantes de ce service ne satisfont pas les deux exigences antagonistes suivantes : la spécification doit être *1)* forte pour faciliter la conception des applications réparties et *2)* faible pour être résoluble. Les deux spécifications les plus prometteuses sont proposées dans [15] et [57]. Elles se différencient par leur propriété de vivacité : la vivacité doit être seulement assurée dans les partitions complètement stables [57] ou la vivacité doit être assurée dans toutes les partitions [15]. Plus précisément,

dans [57], les auteurs définissent une partition ultimement complètement stable comme « un ensemble de processus qui sont ultimement corrects et mutuellement connectés, et pour lesquels tous les liens entre les processus dans cet ensemble sont actifs, et les liens entre les processus de cet ensemble et les autres processus se trouvant à l'extérieur de l'ensemble sont inactifs ». En conséquence, la vivacité ne peut pas être assurée lorsque deux participants sont connectés par intermittence pour toujours. Ce scénario d'instabilité disparaît dans la spécification de [15]. Pour cela, les auteurs considèrent que les liens de communication entre les participants d'une même partition (c'est-à-dire, qui sont mutuellement atteignables) sont équitables. Cependant, comme montré dans [168], la spécification de [57] peut être satisfaite par une implantation « triviale mais inutile » [5] et le modèle de système réparti enrichi avec les liens équitables de [15] est en contradiction avec les systèmes dynamiques et instables [6]. En outre, le modèle de système réparti de ces deux spécifications n'est pas adapté pour les MANETs car l'ensemble des processus dans le système est connu et fixe.

Dans nos travaux, nous proposons un modèle de système réparti construit au-dessus des réseaux mobiles spontanés, puis la spécification de la gestion de groupe partitionnable et sa mise en œuvre par transformation de la gestion de groupe en une séquence de consensus abandonnables. Le consensus abandonnable est la combinaison d'un détecteur ultime des α participants d'une partition et d'un registre ultime par partition.

Modèle de système réparti dynamique. Nous considérons le modèle d'arrivée infinie avec accès simultané borné [145]. Le système consiste en un ensemble infini dénombrable de participants qui ne se connaissent pas nécessairement les uns les autres avant de se rencontrer. Nous considérons un participant par nœud, et dans la suite, les mots « participant », « nœud » et « processus » sont interchangeables. Chaque processus peut accéder à une mémoire locale stable qui permet au processus de sauvegarder son état, puis de redémarrer après une défaillance en recouvrant son état.

Dans les MANETs, les nœuds communiquent en diffusant des messages à leurs voisins. Pour éviter les scénarios dégradés, nous supposons maintenant que les liens actifs par intermittence pour toujours, incluant les liens ultimement actifs, sont équitables. Nous définissons la relation d'atteignabilité comme suit : le processus q est atteignable par p si et seulement si le chemin de p à q est équitable. Ainsi, différemment de [15] et comme suggéré dans [168], les définitions de lien équitable et d'atteignabilité sont indépendantes du temps. Ensuite, une partition est un ensemble de processus mutuellement atteignables.

Avec uniquement des liens équitables, le système peut souffrir des pertes de messages et des délais de transmission de messages arbitraires. Aussi, nous introduisons le concept de lien SADDM (*Simple Average Delayed/Dropped of a Message*), qui est inspiré de [185]. Un lien SADDM autorise que des messages soient perdus et que d'autres soient transmis avec des délais de transmission non bornés, mais assure qu'un sous-ensemble des messages est ultimement reçu et que les messages de cet ensemble ne sont pas trop dispersés dans le temps. Un lien SADDM est défini comme suit : Soient β et δ deux constantes, et soit $I = [t_1, t_2]$ un intervalle de temps fini durant lequel le processus p diffuse un message m au processus correct q au moins β fois ; le lien $p \rightsquigarrow q$ est un lien SADDM si q reçoit le message m au moins une fois avant l'instant $t_1 + \delta$, avec $\delta > t_2 - t_1$. Par conséquent, une partition est stable durant une période lorsque tous les

5. Un ensemble de processus est autorisé à installer une vue capricieuse [9] à n'importe quel instant et sans aucune raison. Dans le pire des scénarios, comme montré dans [168], suite à une installation d'une nouvelle vue, chaque participant installe une vue singleton ne contenant que lui. Dans ce cas, le service de gestion de groupe partitionnable n'aide pas à obtenir des informations correspondant à la réalité de l'environnement.

6. Si un participant p envoie un message m à q à l'instant t alors q reçoit m si et seulement si q est atteignable par p à l'instant t. Or, comme expliqué dans [168], la relation d'atteignabilité n'est pas un invariant dans le temps.

processus de la partition peuvent communiquer les uns avec les autres via des chemins SADDM durant cette période.

Comme dans [147], nous découpons l'exécution d'une application répartie en intervalles de temps, considérons seulement une période, et définissons la stabilité d'une partition après un instant de stabilisation local (*Local Stabilisation Time*), qui est inconnu des processus. La partition stable de p est l'ensemble des processus corrects tel qu'il existe un instant LST_p après lequel il existe au moins un chemin SADDM de p vers chaque processus de la partition stable et des chemins SADDM pour le retour vers p.

Un processus est stable dans une partition s'il existe des chemins SADDM de ce processus vers tous les autres processus de la partition, et *vice versa*. Puisque des processus stables et instables peuvent coexister dans la même partition et puisque LST_p n'est pas connu des processus, il est souhaitable d'éviter que les processus instables empêchent la progression des processus stables. Par conséquent, l'application répartie devrait être exécutée seulement par un ensemble constitué d'au moins α processus stables dans la partition. α exprime le compromis entre l'accord et la progression parmi les processus d'une partition. Dans notre modèle, α est un paramètre dépendant de l'application. C'est l'application qui choisit pour chaque processus la valeur de α, c'est-à-dire le nombre minimum de processus stables nécessaire pour continuer « à travailler ». En résumé, nous définissons la condition de stabilité suivante : chaque processus choisit une valeur α dénotée α_p et toute partition de p doit contenir au minimum α_p processus.

La sélection des processus stables est basée sur un critère de stabilité utilisé afin de déterminer les processus mutuellement atteignables qui sont les plus stables, c'est-à-dire ceux qui peuvent être considérés comme faisant partie d'une éventuelle partition stable. En nous inspirant de nos travaux précédents (cf. section 4.2), nous considérons le critère de stabilité $HB(p, q)$, avec HB qui est une fonction qui dépend du nombre de battements de cœur multi-sauts reçus par p de la part de q, et une valeur seuil. En utilisant ce critère de stabilité, nous pouvons éliminer un processus lorsqu'il quitte cette partition tout en tolérant les déconnexions sporadiques.

La figure 7 illustre les concepts de chemin SADDM, de condition de stabilité et de partition stable. Les disques noirs représentent les processus instables tandis que les disques blancs sont les processus stables. Chaque partition stable est entourée par un cercle en trait plein. Sont dessinées cinq partitions stables avec leur valeur de α. La valeur de α pour le processus o est égale à 1 par exemple pour exprimer le fait que le processus de l'application s'exécutant sur o accepte de progresser seul. α de p exprime quant à lui que les participants de cette partition requièrent, selon p, au moins 4 membres pour progresser. Une partition stable n'est pas nécessairement isolée. Par exemple, le processus u dans la partition stable du processus w peut recevoir des messages diffusés par les processus dans la partition de p via des chemins SADDM, mais il n'existe pas de chemin SADDM des processus de la partition w vers u.

Spécification de la gestion de groupe partitionnable à base de consensus abandonnable. La figure 8 illustre l'architecture de la gestion de groupe partitionnable $\mathcal{PGM}$, qui est construit au-dessus du consensus abandonnable $\mathcal{AC}$. $\mathcal{AC}$ combine deux modules : le détecteur des α participants d'une partition $\Diamond\mathcal{PPD}$ et le registre ultime par partition $\Diamond\mathcal{RPP}$. $\Diamond\mathcal{PPD}$ et $\Diamond\mathcal{RPP}$ correspondent respectivement aux versions adaptées du leader ultime $\Diamond Leader$ et du registre ultime $\Diamond Register$ de [33]. Le rôle de $\Diamond\mathcal{PPD}$ est de construire l'ensemble αSet constitués d'au moins α processus stables et d'élire un leader parmi cet ensemble. Le rôle de $\Diamond\mathcal{RPP}$ est de proposer et de décider une valeur parmi les processus dans αSet. $\Diamond\mathcal{RPP}$ utilise le module de retransmission $\mathcal{RM}$ pour diffuser des messages de manière fiable via des

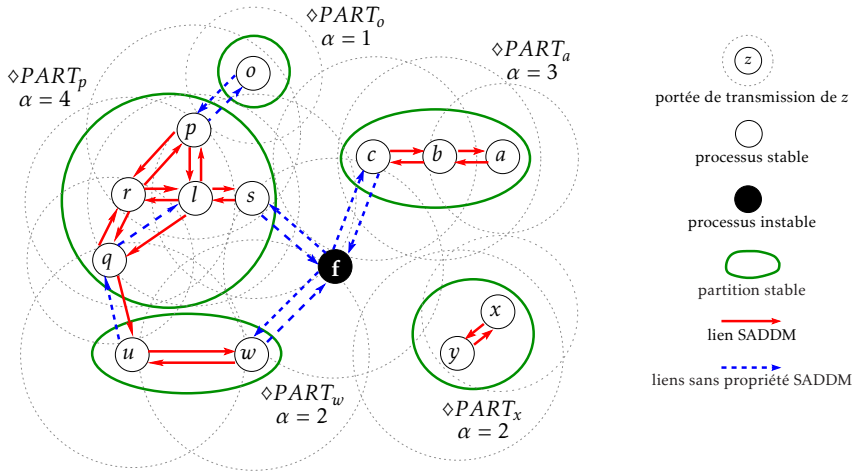


FIGURE 7 – Partitions stables et leur condition de stabilité durant une période

chemins SADDM. $\Diamond\mathcal{RPP}$ utilise aussi le module $\Diamond\mathcal{PPD}$ afin de décider si une valeur proposée doit être abandonnée lorsqu'un ou plusieurs participants disparaissent de la partition. La couche applicative décide d'inclure ou d'exclure les processus en proposant d'installer une nouvelle vue en utilisant $\mathcal{PGM}$. $\mathcal{PGM}$ utilise le module de retransmission pour diffuser les nouvelles vues installées. L'ensemble des membres potentiels de la nouvelle vue doivent être inclus dans αSet. Pour cela, $\Diamond\mathcal{PPD}$ notifie l'application lorsqu'il détecte un changement dans la composition de αSet.

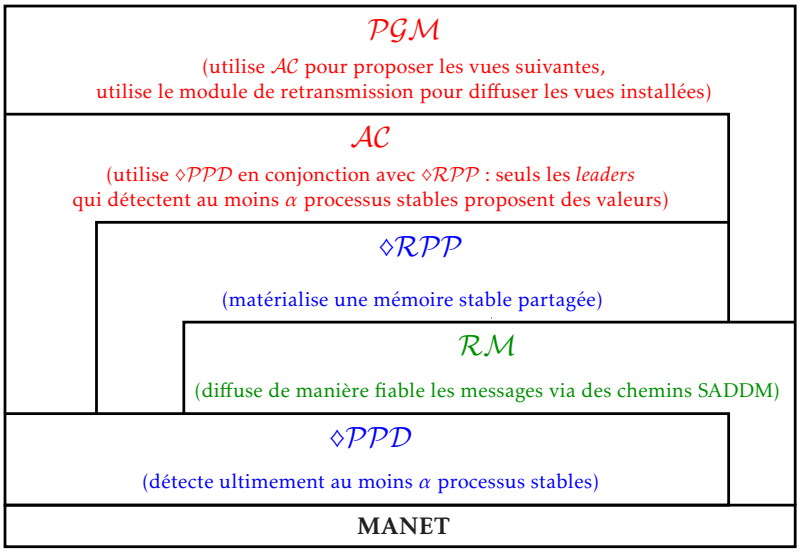


FIGURE 8 – Architecture de $\mathcal{PGM}$

Nous présentons maintenant les propriétés des différentes briques logicielles, en commençant par les couches basses de l'architecture.

Le détecteur ultime des α participants d'une partition $\Diamond\mathcal{PPD}$ est un oracle réparti qui détecte ultimement l'ensemble αSet des processus stables dans une partition. Les processus dans αSet sont sélectionnés selon le critère de stabilité $HB(p,q)$. En même temps, $\Diamond\mathcal{PPD}$ élit le leader de αSet. Puisque les processus

ne savent pas quand la composition de αSet se stabilise, plusieurs processus peuvent croire pendant un moment qu'ils sont leaders. Cependant, quand la condition de stabilité est satisfaite après LST_p (ou pour une période de temps suffisamment longue), un leader unique est élu. $\diamondsuit\mathcal{PPD}$ satisfait les deux propriétés suivantes : *1)* (stabilité du αSet) il existe un instant après lequel deux processus p et q d'un ensemble αSet de processus stables possèdent la même connaissance αSet ; *2)* (accord sur l'élection du leader) il existe un instant après lequel deux processus p et q d'un ensemble αSet de processus stables élisent le même processus $l \in \alpha Set$ comme le leader.

Le registre $\diamondsuit\mathcal{RPP}$ matérialise la mémoire stable partagée. À la différence de [33], le registre est partagé uniquement par les α processus stables de la partition, et la tentative de stocker une valeur dans le registre peut ne pas aboutir dans deux cas : *1)* contention ou *2)* condition de stabilité non satisfaite. Le premier cas est le même que celui du module $\diamondsuit\mathcal{R}egister$ de [33] : un proposeur (c'est-à-dire un processus qui propose une valeur dans l'algorithme de consensus [de la couche au-dessus]) peut abandonner une proposition s'il existe un autre proposeur dans la même partition qui a commencé à proposer une valeur de façon concurrente. Notons que dans ce cas, le consensus n'est pas abandonné. Dans le second cas, le proposeur n'abandonne pas seulement sa proposition, mais aussi l'instance du consensus associée car il n'existe pas α processus stables dans sa partition. $\diamondsuit\mathcal{RPP}$ satisfait les propriétés suivantes : *1)* (accord par partition) dans une partition, ultimement, deux processus de l'ensemble des processus stables αSet n'écrivent pas différentes valeurs ; *2)* (terminaison par partition) dans une partition, soit un processus p qui propose une valeur. S'il existe un ensemble αSet constitué d'au moins α processus stables incluant p, alors p écrit sa valeur ultimement. Sinon, p abandonne ; *3)* (non-trivialité d'abandon) dans une partition, s'il existe un ensemble αSet de processus stables constitué d'au moins α processus et si un processus p de cet ensemble propose une infinité de fois une valeur $val' = (vset, vid')$, avec $vset \subseteq \alpha Set \wedge |vset| \geqslant \alpha$, alors p décide ultimement $val = (vset, vid)$, avec $vid' \leqslant vid$; s'il n'existe pas un ensemble αSet constitué d'au moins α processus stables, alors p abandonne.

Le module de retransmission $\mathcal{RM}$ est utilisé pour permettre aux processus de diffuser des messages de manière fiable et têtue à travers les chemins SADDM. $\mathcal{RM}$ satisfait la propriété suivante : (diffusion fiable à un ensemble de destinataires) soit p un processus stable et *dest* un ensemble de processus stables inclus dans $\diamondsuit PART_p$; si p diffuse un message m β^n fois et si *dest* est l'ensemble des processus destinataires de m, alors le message m originellement diffusé par p arrive à chaque processus $q \in dest$ au plus en $\beta^n\eta + n\delta$ secondes, avec η la période de temps maximale qui sépare deux diffusions consécutives du message m et n la longueur du chemin SADDM le plus long.

Dans le consensus abandonnable $\mathcal{AC}$, par analogie avec le consensus Synod de Paxos [132, 133], seuls les processus qui pensent être des leaders proposent une valeur. Dans le rôle du proposeur, p propose un ensemble *vset* et un identifiant *vid* qui correspondent aux membres et à l'identifiant de la vue potentiellement successeur de la vue courante. La valeur retournée peut être ($\perp$, $\perp$), signifiant alors que le consensus a été abandonné. Si la valeur retournée est différente de $(\perp, \perp)$ alors le consensus est dit atteint. $\mathcal{AC}$ satisfait la propriété de validité par partition et les deux propriétés suivantes : *1)* (accord par partition) dans une partition, deux processus de l'ensemble des processus stables αSet ne décident pas ultimement différentes valeurs ; *2)* (terminaison par partition) dans une partition, soit un processus p qui propose une valeur. S'il existe un ensemble αSet constitué d'au moins α processus stables incluant p, alors p décide ultimement. Sinon, p abandonne ;

Les propriétés de la gestion de groupe partitionnable ainsi construite sont les suivantes :

— les propriétés concernant l'installation de vues :

— auto-inclusion : si le processus p installe la vue v alors p est un membre de v ;
— monotonicité locale : si le processus p installe la vue v après avoir installé la vue v' alors l'identifiant de v est plus grand que celui de v' ;
— événement de vue initiale : chaque événement applicatif est exécuté dans le contexte d'une vue ;

— les propriétés justifiant l'installation d'une vue et la stabilité :
— validité par partition : si le processus p installe la vue $v = (vset, vid)$ alors $v' = (vset, vid')$ a été proposée par un processus q (possiblement p) dans $vset$, avec $|vset| \geqslant \alpha_p$ et $vid' \leqslant vid$;
— accord sur la vue finale : dans une partition, s'il existe un ensemble $S \subseteq \alpha Set \subseteq \Diamond PART_p$ de processus stables, avec $\forall q \in \alpha Set, |S| \geqslant \alpha_q$, qui souhaitent installer la vue $v_{f'} = (vset, vid')$ avec $vset = S$, alors tous les processus dans S installent la même vue finale $v_f = (vset, vid)$, avec $vid' \leqslant vid$.

La propriété d'accord sur la vue finale garantit la vivacité d'une partition même si la partition n'est pas complètement stable. La particularité de notre approche réside dans la manière de détecter l'ensemble αSet. En outre, l'installation de vues capricieuses n'est pas autorisée car une vue ne peut être installée par un ensemble de processus S que si elle a été proposée par un processus de S et contient au moins α membres (le nombre minimum de processus stables requis par chacun des processus de S). Enfin, remarquons que la vue retournée possède le numéro vid et non vid', avec $vid' \leqslant vid$: le processus p propose une valeur et en attendant que la partition se stabilise, p peut être amené à proposer plusieurs fois sa valeur.

Toutes les briques logicielles de l'architecture du gestionnaire de groupe partitionnable ont été mises en œuvre dans des algorithmes répartis, et la correction des algorithmes a été prouvée.

4.4 Bilan

Dans cette section, nous avons présenté nos contributions en tolérance aux fautes pour les applications réparties en environnement mobile. Dans les premiers travaux, nous ciblons les systèmes répartis présentant des défaillances, des déconnexions, et des partitionnements du réseau. Nous présentons trois contributions : *1)* une version modifiée du détecteur de défaillance non fiable à base de « battements de cœur » [2], qui en plus d'offrir des informations sur la suspicion des participants et la possibillité de réaliser une primitive de diffusion fiable silencieuse [3] et têtue [93], fournit des informations sur les chemins de la topologie ; *2)* un détecteur de déconnexion qui diffuse les informations sur les déconnexions et les reconnexions ; *3)* un détecteur de partition non fiable qui détecte les participants dans les partitions complètement stables. Ces détecteurs peuvent être utilisés pour construire une primitive de consensus ou un service de gestion de groupe dans des partitions complètement stables [57].

Les seconds travaux présentés dans cette section concernent la gestion de groupe partitionnable, un service intergiciel de base pour les systèmes répartis tolérants aux partitionnements du réseau. Les spécifications existantes de ce service ne satisfont pas les deux exigences antagonistes suivantes : la spécification doit être *1)* assez forte pour faciliter la conception des applications réparties et *2)* assez faible pour être résoluble. Nous proposons un modèle de système réparti construit au-dessus des réseaux mobiles spontanés, puis la spécification de la gestion de groupe partitionnable et sa mise en œuvre algorithmique par transformation de la gestion de groupe en une séquence de consensus abandonnables. Le consensus abandonnable est la combinaison d'un détecteur ultime des α participants d'une partition et d'un registre ultime par partition. La spécification est assez forte car elle interdit l'installation des

vues capricieuses durant les périodes stables, et assez faible pour être résoluble car elle est mise en œuvre algorithmiquement.

Le travail sur la détection de déconnexion, de défaillance et de partition a eté effectué en collaboration avec Matthieu Bouillaguet, Luciana Arantes, Pierre Sens et Gaël Thomas, notamment dans le cadre d'un séjour d'étude dans l'équipe REGAL de septembre 2006 à février 2007, et avec Lynda Temal, Usman Bhatti et Tuan Dung Nguyen dans le cadre de leur stage de Master 2 Recherche. Les travaux sur la gestion de groupe partitionnable comprennent la contribution de la thèse de Léon Lim [140]. Les principales publications sur ce sujet sont les suivantes : NCA'2010 [252] et NCA'2008 [256] pour la première contribution, JPDC (2014) [241] et EDCC'2012 [248] pour la seconde.

5 Conclusion et perspectives

Cette section est l'occasion de dresser un bilan de mes activités de recherche présentées dans ce manuscrit et d'exposer des perspectives pour de nouvelles contributions. Dans la section 5.1, je présente une synthèse de mes contributions. Dans la section 5.2, je développe les perspectives à court et moyen termes.

5.1 Bilan

J'ai présenté dans ce manuscrit mes contributions en termes de conception et de mise en œuvre des applications réparties en environnements mobile et ubiquitaire, et dans l'Internet des objets à l'aide d'intergiciels. La ligne conductrice de ces travaux est la définition du rôle des intergiciels dans l'intégration des dispositifs mobiles et des objets connectés dans les architectures logicielles réparties. Les contributions sont exprimées en termes de :

— spécifications de modèles de systèmes répartis et de propriétés,
— mises en œuvre algorithmiques,
— architectures réparties,
— mises en œuvre logicielles dans des services intergiciels, et
— outils de développement d'applications réparties.

Les sujets abordés sont l'adaptation aux déconnexions, la sensibilité au contexte d'exécution et la tolérance aux fautes.

Les premières contributions relatées concernent chronologiquement nos premiers travaux dans le domaine de l'informatique mobile avec le sujet de l'adaptation aux déconnexions. Les causes des déconnexions proviennent des contraintes fortes en ressources des appareils mobiles et de la variabilité des communications sans fil. Lorsque nous avons débuté nos recherches sur le sujet, les principales solutions existantes visaient à rendre l'adaptation aux déconnexions transparente aux applications avec des mécanismes intégrés au niveau du système d'exploitation. Nous avons alors étudié la conception de stratégies d'adaptation collaboratives entre l'application et l'intergiciel. La première contribution est la structuration des applications sur appareils mobiles (au moment de l'étude, des assistants personnels numériques). Les objets assurant la continuité de service pendant les déconnexions sont appelés des objets déconnectés. Nous adressons dans ces premiers travaux les applications patrimoniales à base d'objets CORBA. Plus précisément, nous montrons comment les mécanismes comme les intercepteurs portables et le passage des objets par valeur sont utilisés pour séparer les préoccupations. Ensuite, nous nous intéressons aux composants CORBA et proposons une approche pour concevoir des applications à base de composants s'adaptant aux déconnexions. Nous décrivons comment l'apport du paradigme composant–conteneur améliore encore la séparation des préoccupations et aide à la conception de stratégies d'adaptation collaboratives : l'application répartie fournit des fonctionnalités, qui sont mises en œuvre par des composants interconnectés ; l'architecte identifie des points de variation, par exemple les connexions et les composants à activer ou désactiver pour la gestion des déconnexions, et spécifie la politique d'adaptation ; les développeurs mettent en œuvre la politique d'adaptation avec des mécanismes d'adaptation insérés dans les conteneurs ; et les utilisateurs finaux donnent des indications pendant l'exécution afin d'améliorer la qualité du service. Dans cette seconde contribution, nous proposons aussi des politiques de déploiement et de remplacement pour la gestion du cache de composants déconnectés sur l'appareil mobile. Ces deux contributions sont mises en œuvre dans le canevas logiciel DOMINT et validées dans un scénario appli-

catif de gestion de crise plan rouge. Les principales publications sur ce sujet sont les suivantes : ANTe (2006) [245], DOA'2003 [259], et IPDPS Workshops'2002 [289] et EDOC Workshop'2002 [288].

Nos travaux suivants s'inscrivent dans le domaine de l'informatique ubiquitaire dans lequel les environnements d'exécution sont ouverts et changeant avec de nombreux utilisateurs et objets connectés mobiles et volatiles. Afin de bénéficier des informations en provenance de leur contexte d'exécution, les applications deviennent sensibles au contexte et s'adaptent en fonction des situations. Dans ce cadre, nous étudions l'une des briques de base de la boucle d'autonomie : la gestion de contexte. Lorsque nous commençons nos recherches sur ce sujet, les solutions de la littérature sont soit issues des travaux en supervision système, donc avec une abstraction des informations de contexte de faible niveau ou très spécifique à la supervision système, soit construites de façon *ad hoc*, c'est-à-dire sans organisation de la fonctionnalité dans un service intergiciel. Nos contributions sont alors organisées selon deux axes de recherche, qui sont la structuration de la gestion de contexte dans un service intergiciel et l'ingénierie dirigée par les modèles des applications sensibles au contexte. Notre première contribution est la structuration du service intergiciel de gestion de contexte comme une forêt de nœuds de contexte, qui prennent en entrée des informations de contexte, appliquent un traitement, par exemple de filtrage, et produisent des informations de contexte de plus haut niveau d'abstraction. Nous définissons un méta-modèle de ces nœuds de contexte, fournissons un langage dédié permettant de les composer pour former des hiérarchies avec partage, et générons l'architecture du service ainsi que les squelettes des composants nœuds de contexte. La généricité de l'architecture des nœuds de contexte est démontrée en la spécialisant pour de nouvelles préoccupations comme la gestion de la qualité de contexte. La seconde contribution concerne la distribution des informations de contexte à l'aide d'un système réparti à base d'événements, qui est augmenté spécifiquement pour la gestion de contexte : par exemple avec du contrôle d'accès pour le respect de la vie privée. Nous mettons aussi en avant le concept de distribution multi-échelle (en anglais, *multiscoping distribution*) avec le concept de portée de distribution (en anglais, *scope*) pour exprimer que les informations de contexte des producteurs sont confinées aux consommateurs appartenant à la même portée de distribution. Une portée peut être récursivement membre d'autres portées et des filtres de visibilité contrôlent les transitions entre portées. La solution obtenue permet de structurer le réseau logique de courtiers du système réparti à base d'événements selon des préoccupations fonctionnelles comme l'organisation de groupes de clients et extrafonctionnelles comme la localisation géographique. La première contribution sur ce sujet de la gestion de contexte est mise en œuvre dans le canevas logiciel COSMOS et validée dans un scénario applicatif de commerce mobile. La seconde contribution est en cours de finalisation avec une spécification et une mise en œuvre algorithmique dans le canevas logiciel muDEBS, et est en cours de démonstration dans un scénario applicatif de ville intelligente avec transport multimodal. Les principales publications sur la première contribution sur ce sujet sont les suivantes : SCICO (2013) [242], TSI (2008) [244], IEEE DSOnline (2008) [243], DAIS'2007 [257], DOA'2009 [255], ICDIM'2010 [253], et DAIS'2011 [251]. Les premières publications sur la seconde contribution sont les suivantes : M4IoT'2014 [276], et MW4NG'2014 [277]. En outre, un article soumis à Annales des Télécommunications est sélectionné avec révision majeure.

Les derniers travaux présentés dans ce manuscrit concernent le domaine de la tolérance aux fautes des applications mobiles. Nous nous intéressons aux systèmes répartis très dynamiques construits avec des réseaux mobiles spontanés. Conjointement à nos travaux sur la gestion de contexte, nous nous intéressons aux mécanismes de détection des entraves, avec une attention particulière aux déconnexions et aux partitionnements réseau fréquents dans les réseaux mobiles. Ces mécanismes de détection sont utilisés pour construire des primitives comme le consensus ou la gestion de groupe à l'intérieur des partitions.

Contrairement aux systèmes dits de partition primaire dans lesquels seuls les participants de la partition regroupant une majorité de participants progressent, les systèmes partitionnables autorisent tout processus appartenant à une partition quelque peu stable de continuer à progresser, et donc à diverger par rapport aux processus des autres partitions. À notre avantage, si on considère les systèmes asynchrones sujets à des défaillances par arrêts francs, les résultats qui stipulent l'impossibilité du consensus et de la gestion de groupe pour les systèmes de partition primaire ne s'appliquent pas aux systèmes partitionnables. En revanche, dans les systèmes partitionnables, il est impossible de maintenir à la fois la cohérence (sous-entendu forte), la disponibilité (permettant de progresser) et la tolérance aux partitionnements réseau. Comme nous souhaitons tolérer les partitionnements réseau, nos contributions s'entendent comme étant complémentaires des résultats en réplication optimiste sur la gestion de la cohérence faible. Dans notre première contribution sur ce sujet de la détection, nous considérons les modèles de systèmes répartis avec des partitions dites complètement stables dans lesquels les partitions sont complètement isolées les unes des autres. Si ce n'est pas le cas, la stabilité des partitions non complètement isolées est compromise, ce qui entrave la progression des participants. Par rapport aux travaux de la littérature, nous proposons de distinguer de manière optimiste les défaillances par arrêts francs des déconnexions, ceci afin par exemple d'appliquer les résultats sur l'adaptation aux déconnexions : utilisation d'un cache d'entités déconnectées, gestion des opérations pendant les déconnexions, et réconciliation lors des reconnexions. Dans notre seconde contribution sur le sujet de la détection, nous levons la contrainte d'isolation complète des partitions pour autoriser que des liens unidirectionnels existent par intermittence entre participants de partitions différentes. L'objectif est de détecter les participants d'une partition et de construire un service de gestion de groupe partitionnable avec une spécification qui soit forte pour interdire la construction de partitions ne correspondant pas à la réalité de l'environnement à un instant donné et qui soit faible pour être mise en œuvre algorithmiquement. À notre connaissance, aucune solution satisfaisante n'existait dans la littérature. Pour obtenir notre solution, nous définissons un critère de stabilité (d'un participant dans une partition), qui s'exprime en fonction de la qualité des liens réseau. Ensuite, nous appliquons une stratégie collaborative entre l'application et le gestionnaire de groupe : chaque participant formule sa condition de stabilité (de la partition), qui est le nombre minimum de participants stables dans la partition pour accepter de progresser. Les modèles de systèmes répartis et les spécifications de ces deux contributions sont écrites formellement et des mises en œuvre algorithmiques sont proposées, avec leur preuve. Les principales publications sur ce sujet sont les suivantes : NCA'2010 [252] et NCA'2008 [256] pour la première contribution, JPDC (2014) [241] et EDCC'2012 [248] pour la seconde.

5.2 Perspectives

Les intergiciels offrent des paradigmes de conception, des modèles de programmation et des canevas logiciels pour aider le développement et l'exécution d'applications réparties. La construction de tels systèmes répartis reste un défi à cause de leur champ d'application très vaste, et de la taille et de la complexité des problèmes adressés. Dans cette section, je développe trois sujets comme perspectives à mes travaux décrits dans ce manuscrit. Le premier sujet concerne la dissémination des informations en provenance de l'Internet des objets. J'ai décrit dans la section 3.3 les premiers résultats en termes de contrôle de la dissémination des souscriptions et des publications dans un système à large échelle. Je prévois de continuer l'étude de l'approche multi-échelle et de m'intéresser à la mobilité des clients et des courtiers. Le deuxième sujet s'inscrit dans la continuité des travaux sur la tolérance aux fautes des applications mobiles. J'ai présenté dans la section 4.3 une solution au problème de la gestion de groupe

partitionnable dans laquelle les clients participent à l'établissement de la condition de stabilité pour la formation des groupes opérationnels. Je me propose d'approfondir l'utilisation en situation de forte dynamicité de la gestion de groupe partitionnable comme abstraction de base pour les communications de groupe, les machines à états ou les services de coordination dans les systèmes partitionnables. Enfin, le troisième sujet est nouveau par rapport à mes activités précédentes. Il s'agit de faciliter la conception d'algorithmes répartis par l'outillage autour du concept de langage dédié. Je compte m'appuyer sur les travaux dans le cadre des deux autres sujets pour participer au développement de logiciels libres pour la recherche, l'enseignement, et le transfert vers le monde industriel.

Dissémination des informations en provenance de l'Internet des objets. L'Internet des objets est considéré comme une évolution importante des systèmes ubiquitaires [204]. Des domaines applicatifs comme la logistique, la santé, la domotique, le contrôle environnemental, la ville intelligente, les jeux multijoueurs pervasifs, etc. sont dès aujourd'hui très demandeurs de solutions architecturales pour construire rapidement, à un coût raisonnable et avec une qualité contrôlée des applications à destination du grand public. Comme défendu dans l'ensemble des travaux présentés dans ce manuscrit, ces solutions requièrent des briques logicielles d'intermédiation, c'est-à-dire de l'intergiciel.

Historiquement, comme rappelé dans [206], les choix architecturaux oscillent de manière cyclique entre centralisation et répartition. Dans l'approche centralisée mettant en œuvre des solutions autour de l'informatique en nuage pour la récupération, le traitement et la présentation des informations en provenance de l'Internet des objets, le modèle d'affaire suppose que les applications tolèrent des latences possiblement élevées, des coûts de communication et énergétiques potentiellement élevés, et une centralisation des données personnelles par des tiers. Dans les années à venir, avec l'augmentation attendue du nombre d'objets connectés, ainsi que de leur utilisation, il est raisonnable de prévoir que des solutions plus fortement réparties seront demandées, qui impliquent par exemple des téléphones mobiles jouant le rôle de portails d'accès aux informations des capteurs de proximité, ces téléphones étant en interaction avec des nuages de proximité ou directement avec des pairs.

Parmi les défis soulevés, citons l'un d'entre eux qui est identifié depuis longtemps mais qui reste pertinent : le passage à l'échelle localisé (en anglais, *localised scalability*), c'est-à-dire un passage à l'échelle qui prend en compte les distances pour contrôler la distribution des données [186]. Dans nos premiers travaux sur muDEBS, nous dessinons une réflexion plus large en permettant de structurer une topologie de courtiers selon plusieurs dimensions de points de vue architecturaux complémentaires, donc pas uniquement la distance physique, mais aussi les caractéristiques des réseaux traversés, l'organisation sociale des utilisateurs, etc. Notons que ces structurations peuvent être effectuées selon des préoccupations fonctionnelles, comme par exemple la constitution de groupes d'utilisateurs, et extrafonctionnelles, comme par exemple des caractéristiques des réseaux traversés ou la localisation géographique. J'envisage donc de continuer nos investigations sur ce que des collègues de l'équipe nomment la caractérisation multi-échelle des systèmes répartis (cf. section 3.1) et sur l'utilisation de cette caractérisation pour améliorer et contrôler la dissémination des informations de contexte en provenance des objets connectés. L'objectif est de laisser à l'utilisateur et propriétaire d'informations de contexte (en provenance par exemple d'objets connectés qu'il possède ou contrôle) la possibilité de gérer son environnement ambiant tout en ayant un accès limité aux espaces ambiants d'autres utilisateurs.

Concernant cette problématique, je prévois de continuer nos recherches sur les concepts de portée de distribution (cf. section 3.3). Par ailleurs, je réfléchirai à la prise en compte de la mobilité des utilisateurs,

des objets connectés ainsi que des éléments internes de l'architecture comme les courtiers. Enfin, une troisième piste de réflexion sera l'étude de la complémentarité avec les travaux sur les traitement complexes d'événements [63, 207] (par exemple la prise en compte d'informations de contexte avec des vocabulaires hétérogènes : ontologies différentes [166, 205], corpus spécifiques à un domaine [98] ou folksonomie [99]).

Tolérance aux fautes des applications mobiles. Les systèmes informatiques pour les environnements ubiquitaires et l'Internet des objets sont construits dans des environnements fortement dynamiques : la variété, la fréquence et l'amplitude des changements de contexte d'exécution sont importants. Dans ces conditions, le défi est la définition de propriétés de sûreté et de progression ainsi que la mise en œuvre de solutions logicielles utilisables. Dans la section 4, nous avons vu que le théorème CAP, qui stipule qu'il est impossible de fournir à la fois des garanties de cohérence (forte), de (haute) disponibilité et de tolérance aux partitionnements, a été formulé dans le cadre de la fourniture de services Internet dans les réseaux à large échelle [81, 36, 90]. Comme indiqué par S. Gilbert et N. Lynch dans [89], le sujet requiert un réexamen dans le contexte des systèmes mobiles sans fil. Les partitionnements du réseau sont une caractéristique intrinsèque des environnements ubiquitaires et de l'Internet des objets, tant à cause de l'utilisation de communications par réseaux sans fil qu'à cause de la mobilité des participants et de la faible densité des appareils dans certaines zones. Par ailleurs, les applications dans les environnements ubiquitaires et dans l'Internet des objets sont *a priori* différentes des applications classiquement étudiées dans le cadre des systèmes informatiques construits à l'échelle de l'Internet.

En considérant les partitionnements du réseau inévitables, ce qui est vérifié dans les réseaux mobiles spontanés [96], quatre approches sont répertoriées dans la littérature pour gérer le compromis entre la cohérence et la disponibilité [89] : l'approche choisie par exemple par certains services de coordinations et de machines à états qui favorisent la cohérence au détriment de la disponibilité [50] ; l'approche choisie par certains gestionnaires de contenus à base de cache, par exemple pour le Web, pour lesquels la disponibilité prime aux dépens de la cohérence [153] ; la conciliation entre les exigences de cohérence et de disponibilité à l'aide de la théorie dite de la « cohérence continue » [215, 216] ; et la structuration du système pour définir des composants avec des exigences de cohérence et de disponibilité différentes (décomposition en fonction des données manipulées, des opérations effectuées, des services fournis, des utilisateurs, etc.).

Dans CAP, à la suite de Brewer [37], je pense que la tolérance aux partitions doit être gérée explicitement comme un compromis entre cohérence et disponibilité avec des mécanismes de détection et de recouvrement. Nos travaux présentés sur la gestion de groupe partitionnable sont un début de contribution pour la partie détection. En outre, nos travaux sur la gestion de déconnexion avec les concepts d'opération déconnectée et de mode de fonctionnement (connecté, faiblement connecté et déconnecté) concernent la partie recouvrement, et sont dans une moindre mesure une contribution à la problématique. La nouveauté dans les systèmes partitionnables est que la notion de partition primaire disparaît, laissant ouverte la question de la pertinence et de l'utilisabilité des services qu'il est possible de fournir lorsque tend à disparaître la notion de copie de référence (unique ou multiple) de la réplication optimiste. Autrement dit, il n'est pas garanti que la terminaison de l'activité intervienne dans une partition unique ou de référence, ou qu'un invariant global à la fois fort (pour être utilisable) et faible (pour être mis en œuvre) soit formulable. À ce propos, dans le même article, Brewer note l'intérêt d'une stratégie collaborative en citant le défi de montrer à l'utilisateur ou d'indiquer à l'application l'état d'avancement des activités en cours.

Comme point de départ sur ce sujet, je prévois d'approfondir l'utilisation de la stratégie collaborative à la suite de nos travaux sur la gestion de groupe partitionnable. Notre service de gestion de groupe

partitionnable fournit la constitution des participants ainsi que le leader d'une partition. Une première piste de réflexion est l'étude de l'utilisation de notre proposition pour des "services partitionnables" (c'est-à-dire sensibles aux partitionnements) à base de communications de groupes, de machines à états, ou encore de services de coordination. Les premières questions porteront sur la sémantique fonctionnelle ou extrafonctionnelle du critère de stabilité (par exemple utiliser en complément du nombre de battements de cœur, la durée de présence, la distance physique, ou encore le niveau d'énergie) et sur la condition de stabilité (par exemple introduire la dynamicité de la valeur α, qui est le nombre minimum de processus stables pour progresser dans une partition). En considérant ces premières réflexions et les travaux sur les modèles de mobilité individuelle et de groupe, je me propose d'étudier la définition des problèmes pré-cités (communication de groupe, machine à états et service de coordination, par exemple [141, 43, 50, 107]) dans les environnements dans lesquels l'ensemble des participants n'est pas connu et le système est partitionnable.

Ingénierie logicielle des algorithmes répartis. « La taille ou la composition d'un système n'est pas le seul facteur de complexité qu'il faut gérer. Un autre facteur concerne la diversité qui apparaît [...] dans toutes les activités de développement » [59]. C'est ainsi que je termine la présentation des perspectives par un sujet qui, pour des raisons multiples, concerne aussi bien la communauté des chercheurs que celle de l'enseignement, ou encore le transfert vers le monde industriel : l'ingénierie logicielle des algorithmes répartis. L'une des raisons expliquant la difficulté de l'adoption par l'industrie des résultats de la recherche en algorithmique répartie est le coût du développement logiciel d'une mise en œuvre algorithmique [30, 50, 92]. Dans mes activités de recherche et d'enseignement, je fais aussi les constats qui suivent.

En écrivant le pseudo-code d'un algorithme dans une forme proche du code exécutable, le lecteur capture mieux les abstractions mises en œuvre et possède une plus grande confiance dans le résultat grâce à une meilleure traçabilité vers la mise en œuvre. Pour cela, nous avons besoin d'un langage dédié [144]. Nous pensons que ce langage dédié doit être développé comme un langage interne à un langage généraliste, afin de bénéficier de l'écosystème et de la communauté du langage généraliste. À titre indicatif, le concepteur utilise fortement des abstractions comme : la manipulation d'ensembles, qui deviennent des collections dans les langages de la famille Java par exemple, et l'émission et la réception de messages, qui bénéficient de constructions telles que le filtrage par motif (en anglais *pattern matching*) des langages fonctionnels.

Par ailleurs, l'algorithme réparti, lorsque considéré dans une pile logicielle incluant d'autres algorithmes répartis, doit aussi tirer avantage de constructions pour la structuration du système complet : langage de définition d'architecture [143], et langage de programmation orienté objet, composant ou agent [39]. D'autres aspects comme l'instrumentation pour l'observation de l'exécution sont aussi utiles et leur mise en œuvre peut être facilitée en utilisant des *mixins* ou la programmation par aspect.

Une autre facette du sujet est l'outillage nécessaire pour appliquer les principes des méthodes agiles. Pour ce faire, le concepteur d'algorithmes répartis souhaite disposer d'outils d'exécution par « scriptage », de simulation [21, 234, 238] ou d'émulation dans des bancs d'essai [22, 139], et de *model checking* pour la recherche de contre-exemples [58]. Le langage dédié gagnera donc à être « scriptable » et les algorithmes à être insérés sans difficulté dans les plateformes de simulation. Concernant la modélisation formelle, la traduction vers un langage formel devra être peu coûteuse et permettre la traçabilité.

Comme point de départ pour ces travaux, nous étudierons les propositions suivantes : Spin [104], IOA [86], Splay [139, 189], Overlog [6] et Distal [28], en prenant comme exemples la gestion de groupe

partitionnable et la distribution multi-échelle. Dans le cadre du premier exemple, nous possédons la spécification formelle et les algorithmes avec les preuves écrites en langage naturel ainsi que la mise en œuvre d'une partie des algorithmes comme module OMNeT++ [234]. Dans le cadre du second exemple, nous possédons la spécification formelle et les algorithmes avec les preuves écrites en langage naturel ainsi que la mise en œuvre dans le canevas logiciel muDEBS. J'entrevois mes contributions dans le cadre de projet de développement de logiciels libres, le ou les canevas logiciels étant à déterminer.

Conclusion Les principes de base de la recherche sur les intergiciels sont la généricité et la séparation des préoccupations. Ces deux principes se traduisent dans mes activités de recherche par une forte synergie entre la théorie et la pratique. Les aspects théoriques concernent l'étude des abstractions, des paradigmes, des modèles de système, des spécifications, et des algorithmes. Les aspects pratiques ont trait à la conception d'architectures logicielles flexibles, à l'utilisabilité des concepts par les architectes et les utilisateurs finaux d'applications réparties, à l'évaluation des performances à l'aide de simulateurs ou sur bancs d'essai, et à l'évaluation de l'efficacité par la mise en place de démonstrateurs construits avec des canevas logiciels libres dans des projets avec des partenaires académiques et industriels. Cet équilibre entre théorie et pratique est important dans les domaines de l'informatique ubiquitaire et de l'Internet des objets qui bénéficient d'un engouement partagé par le grand public et le monde industriel, avec des défis à relever quand à la maîtrise de la complexité des solutions logicielles, à la sûreté de fonctionnement des systèmes répartis, et à la minimisation du délai de mise sur le marché des applications réparties innovantes.

Références

[1] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young. Mach : A New Kernel Foundation For UNIX Development. In *Proc. Summer 1986 USENIX Technical Conference*, pages 93–111, Atlanta, Georgia, USA, June 1986.

[2] M.K. Aguilera, W. Chen, and S. Toueg. Using the Heartbeat Failure Detector for Quiescent Reliable Communication and Consensus in Partitionable Networks. *Theoretical Computer Science*, 220(1) :3–30, June 1999.

[3] M.K. Aguilera, W. Chen, and S. Toueg. On Quiescent Reliable Communication. *SIAM Journal of Computing*, 29(6) :2040–2073, April 2000.

[4] M.K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Communication-efficient Leader Election and Consensus with Limited Link Synchrony. In *Proc. 23rd ACM Symposium on Principles of Distributing Computing*, pages 328–337, St. John's, Newfoundland, Canada, July 2004.

[5] M.K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. On implementing Omega in systems with weak reliability and synchrony assumptions. *Distributed Computing*, 21(4) :285–314, October 2008.

[6] P. Alvaro, T. Condie, N. Conway, J.M. Hellerstein, and R. Sears. I Do Declare : Consensus in a Logic Language. *ACM Operating Systems Review*, 43(4) :25–30, January 2010.

[7] Y. Amir, D. Dolev, S. Kramer, and D. Malki. Membership algorithms for multicast communication groups. In *Proc. 6th International Workshop on Distributed Algorithm*, volume 647 of *Lecture Notes in Computer Science*, pages 292–312, Haifa, Israel, October 1992. Springer-Verlag.

[8] AMQP Consortium. AMQP : Advanced Message Queuing Protocol, Version 0-9-1. Protocol specification, AMQP Consortium, November 2008.

[9] E. Anceaume, B. Charron-Bost, P. Minet, and S. Toueg. On the Formal Specification of Group Membership Services. Technical Report TR 95-1534, Department of Computer Science, Cornell University, Ithaca, New-York (USA), August 1995.

[10] B. Andersen, E. Jul, F. Moura, and V.M.P. Guedes. File System for Semiconnected Operation in AMIGOS. In *Proc. 2nd USENIX Symposium on Mobile and Location-Independent Computing*, December 1994.

[11] L. Arantes, F. Greve, P. Sens, and V. Simon. Eventual Leader Election in Evolving Mobile Networks. In *Proc. of the 17th International Conference On Principles Of DIstributed Systems*, number 8304 in Lecture Notes in Computer Science, pages 23–37, Nice, France, December 2013.

[12] J. Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2–3) :165–175, September 2003.

[13] L. Atzori, A. Iera, and G. Morabito. The Internet of Things : A survey. *Computer Networks*, 54(15) :2787–2805, June 2010.

[14] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1) :11–33, January 2004.

[15] Ö. Babaoğlu, R. Davoli, and A. Montresor. Group Communication in Partitionable Systems : Specification and Algorithms. *IEEE Transactions on Software Engineering*, 27(4) :308–336, April 2001.

[16] B. Badrinath, Armando Fox, Leonard Kleinrock, Gerald Popek, Peter Reiher, and Mahadev Satyanarayanan. A conceptual framework for network and client application. *ACM Mobile Networks and Applications*, 5(4) :221–231, December 2000.

[17] M. Baldauf, S. Dustdar, and F. Rosenberg. A Survey on Context Aware Systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4) :263–277, 2007.

[18] G. Banavar and A. Bernstein. Software Infrastructure and Design Challenges for Ubiquitous Computing Applications. *Communications of the ACM*, 45(12) :92–96, December 2002.

[19] R. Barazzutti, P. Felber, H. Mercier, E. Onica, J.F. Pineau, É Rivière, and C. Fetzer. Infrastructure Provisioning for Scalable Content-based Routing : Framework and Analysis. In *Proc. 12th IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, August 2012.

[20] G Barish and K. Obraczka. World Wide Web Caching : Trends and Techniques. *IEEE Communications Magazine*, pages 178–185, May 2000.

[21] R. Barr, Z.J. Haas, and R. van Renesse. JiST : an efficient approach to simulation using virtual machines : Research Articles. *Software—Practice and Experience*, 35(6) :539–576, May 2005.

[22] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, and S. Muir. Operating system support for planetary-scale network services. In *Proc. of the 1st USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, CA, USA, March 2004.

[23] G. Bell. Bell's law for the birth and death of computer classes. *Communications of the ACM*, 51(01) :86–94, January 2008.

[24] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini. A Survey of Context Data Distribution for Mobile Ubiquitous Systems. *ACM Computing Surveys*, 44(4) :24:1–24:45, August 2012.

[25] A. Belokosztolszki, D.M. Eyers, P.R. Pietzuch, J. Bacon, and K. Moody. Role-based Access Control for Publish/Subscribe Middleware Architectures. In *Proc. of the 2nd International Workshop on Distributed Event-Based Systems*, pages 1–8, San Diago, California, June 2003.

[26] G. Bernard, J. Ben-Hotman, L. Bouganim, G. Canals, B. Defude, J. Ferrié, S. Gançarski, R. Guerraoui, P. Molli, P. Pucheral, C. Roncancio, and P. Valduriez. Mobilité et base de données : État de l'art et perspectives. *Soumis à Technique et Science Informatiques*, 2002.

[27] G. Biegel, V. Cahill, and M. Haahr. A Dynamic Proxy Based Architecture to Support Distributed Java Objects in a Mobile Environment. In *Proc. 4th International Symposium on Distributed Objects and Applications*, volume 2519 of *Lecture Notes in Computer Science*, pages 809–826. Springer-Verlag, October 2002.

[28] M. Biely, P. Delgado, Z. Milosevic, and A. Schiper. Distal : A Framework for Implementing Fault-tolerant Distributed Algorithms. In *Proc. 43rd IEEE International Conference on Dependable Systems and Networks*, Budapest, Hungary, June 2013.

[29] K.P. Birman. Replication and Fault-Tolerance in the ISIS System. In *Proc. 10th ACM Symposium on Operating Systems Principles*, pages 79–86, Orcas Island, USA, December 1985.

[30] K.P. Birman, C. Chandersekaran, D. Dolev, and R. van Renesse. How the Hidden Hand Shapes the Market for Software Reliability. In *Proc. of the First IEEE Workshop on Applied Software Reliability*, Philadelphia, USA, June 2006.

[31] D.L. Black, D.B. Golub, D.P. Julin, R.F. Rashid, R.P. Draves, R.W. Dean, A. Forin, J. Barrera, H. Tokuda, G. Malan, and D. Bohman. Microkernel Operating System Architecture and Mach. In *USENIX Symposium on Micro-Kernels and Other Kernel Architectures*, pages 11–30, Seattle, USA, April 1992.

[32] G. Blair and P. Grace. Emergent Middleware : Tackling the Interoperability Problem. *IEEE Internet Computing*, 16(1) :78–81, January 2012.

[33] R. Boichat, P. Dutta, S. Frølund, and R. Guerraoui. Deconstructing Paxos. *Distributed Computing Column of ACM SIGACT News*, 34(1) :47–67, March 2003.

[34] C. Boutros Saab, X. Bonnaire, and B. Folliot. A flexible monitoring platform to build cluster management services. In *Proc. IEEE International Conference on Cluster Computing*, pages 258–265, Chemnitz, Germany, November 2000.

[35] G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4) :824–840, April 1985.

[36] E. A. Brewer. Towards robust distributed systems (abstract). In *Proc. 19th ACM Symposium on Principles of Distributing Computing*, page 7, Portland, Oregon, USA, July 2000.

[37] E.A. Brewer. Pushing the CAP : Strategies for Consistency and Availability. *IEEE Computer*, 45(2), February 2012.

[38] A. Bricker, M. Gien, M. Guillemont, J. Lipkis, D. Orr, and M. Rozier. Architectural issues in microkernel-based operating systems : The CHORUS experience. *Computer Communications*, 14(6) :347–357, July 1991.

[39] J.-P. Briot. Composants et agents : évolution de la programmation et analyse comparative. *Technique et Science Informatiques*, 33(1–2) :85–115, January 2014.

[40] É. Bruneton. *Un support d'exécution pour l'adaptation des aspects non-fonctionnels des applications réparties*. PhD thesis, INPG, Grenoble, France, 2001.

[41] É. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani. An Open Component Model and its Support in Java. In *Proc. 7th International Symposium on Component-Based Software Engineering*, volume 3054 of *Lecture Notes in Computer Science*, pages 7–22, May 2004.

[42] É. 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) :1257–1284, September 2006.

[43] M. Burrows. The Chubby Lock Service for Loosely-coupled Distributed Systems. In *Proc. of the 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 335–350, Seattle, Washington, USA, November 2006.

[44] L. Capra, G.S. Blair, C. Mascolo, W. Emmerich, and P. Grace. Exploiting Reflection in Mobile Computing Middleware. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4) :34–44, October 2002.

[45] D. Cavin, Y. Sasson, and A. Schiper. Reaching Agreement with Unknown Participants in Mobile Self-Organized Networks in Spite of Process Crashes. Technical Report IC/2005/026, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2005.

[46] U. Çetintemel, P.J. Keleher, B. Bhattacharjee, and M.J. Franklin. Deno : A Decentralized, Peer-to-Peer Object-Replication System for Weakly Connected Environments. *IEEE Transactions on Computers*, 52(7) :943–959, July 2003.

[47] E. Cecchet, H. Elmeleegy, O. Layaïda, and V. Quéma. Implementing Probes for J2EE Cluster Monitoring. *Studia Informatica*, 2005.

[48] S. Chabridon. Dimensions de la gestion des données en univers réparti et mobile : de la cohérence des données à la qualité de contexte. Mémoire d'habilitation à diriger des recherches de l'Université d'Évry Val d'Essonne, Évry, France, November 2014.

[49] S. Chabridon, R. Laborde, T. Desprats, A. Oglaza, P. Marie, and S. Machara Marquez. A survey on addressing privacy together with quality of context for context management in the internet of things. *Annales des Télécommunications*, 69(1) :47–62, February 2014.

[50] T.D. Chandra, R. Griesemer, and J. Redstone. Paxos Made Live : An Engineering Perspective. In *Proc. 26th ACM Symposium on Principles of Distributing Computing*, pages 398–407, Portland, Oregon, USA, August 2007.

[51] T.D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost. On the Impossibility of Group Membership. In *Proc. 15th ACM Symposium on Principles of Distributing Computing*, Philadelphia, USA, May 1996.

[52] T.D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2) :225–267, March 1996.

[53] H. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, Baltimore, USA, January 2003.

[54] L. Cherkasova. Proxies Performance with Greedy-Dual-Size- Frequency Caching Policy. Technical Report HPL-98-69, Computer Systems Laboratory, HPL, November 1998.

[55] M. Cherniack, M.J. Franklin, and S. Zdonik. Expressing User Profiles for Data Recharging. *IEEE Personal Communications*, 8(4) :32–38, August 2001.

[56] G.V. Chockler, D. Dolev, R. Friedman, and R. Vitenberg. Implementing a Caching Service for Distributed CORBA Objects. In J. Sventek and G. Coulson, editors, *Proc. IFIP/ACM/USENIX International Middleware Conference*, volume 1795 of *Lecture Notes in Computer Science*, pages 1–23. Springer-Verlag, April 2000.

[57] G.V. Chockler, I. Keidar, and R. Vitenberg. Group Communication Specifications : A Comprehensive Study. *ACM Computing Surveys*, 33(4) :427–469, December 2001.

[58] E. Clarke, E.A. Emerson, and J. Sifakis. Model Checking : Algorithmic Verification and Debugging. *Communications of the ACM*, 52(11) :75–84, November 2009.

[59] P. Collet, L. Du Bousquet, L. Duchien, and P.E. Moreau. Gdr génie de la programmation et du logiciel — défis 2025. `http://gdr-gpl.cnrs.fr/node/160`, February 2015.

[60] L. Courtrai, F. Guidec, N. Le Sommer, and Y. Mahéo. Resource Management for Parallel Adaptive Components. In *Proc. IEEE IPDPS Workshop on Java for Parallel and Distributed Computing*, pages 134–141, Nice, France, April 2003.

[61] J. Coutaz, J.L. Crowley, S. Dobson, and D. Garlan. Context is Key. *Communications of the ACM*, 48(3) :49–53, March 2005.

[62] J. Coutaz and G. Rey. Foundations for a Theory of Contextors. In C. Kolski and J. Vanderdonckt, editors, *Proc. 4th International Conference on Computer-Aided Design of User Interfaces*, pages 13–34, Valenciennes (France), May 2002. Kluwer.

[63] G. Cugola and A. Margara. Processing Flows of Information : From Data Stream to Complex Event Processing. *ACM Computing Surveys*, 44(3) :15:1–15:62, June 2012.

[64] R.C.A. da Rocha and M. Endler. Context Management in Heterogeneous, Evolving Ubiquitous Envrionments. *IEEE Distributed Systems Online*, 7(4), April 2006.

[65] P.-C. David. *Développement de composants Fractal adaptatifs : un langage dédié à l'aspect d'adaptation*. PhD thesis, Université de Nantes, École des Mines de Nantes (France), July 2005.

[66] P.-C. David and T. Ledoux. WildCAT : a generic framework for context-aware applications. In *Proc. 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing*, pages 1–7, Grenoble (France), November 2005.

[67] A.K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1) :4–7, 2001.

[68] A.K. Dey, D. Salber, and G.D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Special issue on context-aware computing in the Human-Computer Interaction Journal*, 16(2–4) :97–166, 2001.

[69] F. Douglis and J. Ousterhout. Transparent Process Migration : Design Alternatives and the Sprite Implementation. *Software - Practice and Experience*, 21(8) :757–785, August 1991.

[70] C. Dwork, N.A. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 1988.

[71] D. Dwyer and V. Bharghavan. A mobility-aware file system for partially connected operation. *ACM Operating Systems Review*, 31(1) :24–30, January 1997.

[72] E.N. Elnozahy and W. Zwaenepoel. Manetho : Transparent Rollback-Recovery with Low Overhead, Limited Rollback and Fast Output Commit. *IEEE Transactions on Computers*, 41(5) :526–531, May 1992.

[73] P. Eugster, B. Garbinato, and A. Holzer. Pervaho : A specialized middleware for mobile context-aware applications. *Electronic Commerce Research*, 9(4) :245–268, April 2009.

[74] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2), June 2003.

[75] P. Fahy and S. Clarke. CASS : Middleware for Mobile Context-Aware Applications. In *Proc. ACM MobiSys Workshop on Context Awareness*, Boston, USA, June 2004.

[76] L. Fiege, M. Cilia, and B.C. Mühl. Publish-subscribe grows up : Support for management, visibility control, and heterogeneity. *IEEE Internet Computing*, 10(1) :48–55, January 2006.

[77] L Fiege, M. Mezini, G. Mühl, and A. Buchmann. Engineering Event-Based Systems with Scopes. In B. Magnusson, editor, *Proc. 16th European Conference on Object-Oriented Programming*, number 2374 in Lecture Notes in Computer Science, pages 309–333, Málaga, Spain, June 2002. Springer-Verlag.

[78] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2) :374–382, April 1985.

[79] A. Flissi, C. Gransart, and P. Merle. Une Infrastructure à Composants pour des Applications Ubiquitaires. In *Actes de la 2è Conférence Francophone Mobilité et Ubiquité*, volume 120 of *ACM International Conference Proceeding Series*, pages 45–48, Grenoble, France, June 2005.

[80] I.R. Forman and S.H. Danforth. *Putting metaclasses to work*. Addison-Wesley, 1999.

[81] A. Fox and E.A. Brewer. Harvest, Yield and Scalable Tolerant Systems. In *Proc. 7th Workshop Hot Topics in Operating Systems*, pages 174–178, Rio Rico, Arizona, USA, March 1999.

[82] M.J. Franklin, S.R. Jeffery, S. Krishnamurthy, and F. Reiss. Design Considerations for High Fan-in Systems : The HiFi Approach. In *Proc. 2nd Biennial Conf. on Innovative Data Systems Research*, pages 290–304, Asilomar, Californie, January 2005.

[83] K.W. Froese and R.B. Bunt. Cache Management for Mobile File Service. *The Computer Journal*, 42(6) :442–454, June 1999.

[84] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[85] D. Garlan, D.P. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura : Toward Distraction Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2) :22–31, April 2002.

[86] S.J. Garland, N.A. Lynch, and M. Vaziri. IOA : A Language for Specifying, Programming, and Validating Distributed Systems. Technical report, MIT Laboratory for Computer Science, December 1997.

[87] F.C. Gärtner. Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments. *ACM Computing Surveys*, 31(1) :1–26, March 1999.

[88] D. Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems*, 7(1) :80–112, January 1985.

[89] S. Gilbert and N.A. Lynch. Perspectives on the CAP Theorem. *IEEE Computer*, 45(2), February 2012.

[90] S. Gilbert and Lynch ; N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News (Technical Column)*, 33(2) :51–59, June 2002.

[91] T. Gu, H.K. Pung, and D.Q. Zhang. A middleware for building context-aware mobile services. In *Proc. 59th IEEE Vehicular Technology Conference*, volume 5, pages 2656–2660, 2004.

[92] R. Guerraoui, N. Knezevic, V. Quéma, and M. Vukolić. The Next 700 BFT Protocols. In *Proc. 5th ACM SIGOPS/EuroSys European Conference on Computer Systems*, pages 363–376, Paris, France, April 2010.

[93] R. Guerraoui, R. Oliveira, and A. Schiper. Stubborn Communication Channels. Technical Report TR97, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1997.

[94] S. Gurun, C. Krintz, and R. Wolski. NWSLite : A Light-Weight Prediction Utility for Mobile Devices. In *Proc. 2nd ACM/USENIX International Conference on Mobile Systems, Applications and Services*, pages 2–11, Boston, USA, June 2004.

[95] N. Haderer, R. Rouvoy, C. Ribeiro, and L. Seinturier. APISENSE : Crowd-Sensing Made Easy. *ERCIM News, Special theme : Mobile Computing*, pages 28–29, 2013.

[96] J. Hähner, D. Dudkowski, P.J. Marrón, and K. Rothermel. Quantifying Network Partitioning in Mobile Ad Hoc Networks. In *Proc. 6th IEEE International Conference on Mobile Data Management*, pages 174–181, May 2007.

[97] R.S. Harbus. Dynamic process migration : To migrate or not to migrate. Technical report csri-42, University of Toronto, Canada, July 1986.

[98] S. Hasan and E. Curry. Approximate Semantic Matching of Events for the Internet of Things. *ACM Transactions on Internet Technology*, 14(1) :2.1–2.23, July 2014.

[99] S. Hasan and E. Curry. Thematic Event Processing. In *Proc. 14th IFIP/ACM/USENIX International Middleware Conference*, pages 109–120, Bordeaux, France, December 2014.

[100] K. Henricksen and J. Indulska. Modelling and Using Imperfect Context Information. In *Proc. IEEE PerComm Workshop on Context Modeling and Reasoning*, pages 28–33, Orlando, Florida, March 2004.

[101] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam. Middleware for Distributed Context-Aware Systems. In *Proc. 7th International Symposium on Distributed Objects and Applications*, Lecture Notes in Computer Science, Agia Napa (Cyprus), November 2005. Springer-Verlag.

[102] A. Hinze, K. Sachs, and A. Buchmann. Event-Based Applications and Enabling Technologies. In *Proc. 3rd ACM International Conference on Distributed Event-Based Systems*, pages 1–15, Nashville, TN, USA, July 2009.

[103] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger. Context-Awareness on Mobile Devices — the Hydrogen Approach. In *Proc. 36th Annual Hawaii International Conference on System Sciences*, Hawaii (USA), January 2003.

[104] G.J. Holzmann. The Model Checker Spin. *IEEE Transactions on Software Engineering*, 23(5) :1–17, May 1997.

[105] B.C. Housel, G. Samaras, and D.B. Lindquist. WebExpress : A client/intercept based system for optimizing Web browsing in a wireless environment. *ACM Mobile Networks and Applications*, 3(4) :419–431, December 1998.

[106] X. Hu, Y. Ding, N. Paspallis, P. Bratskas, G.A. Papadopoulos, P. Barone, and A. Mamelli. A Peer-to-Peer based infrastructure for Context Distribution in Mobile and Ubiquitous Environments. In *Proc. of 3rd International Workshop on Context-Aware Mobile Systems*, Vilamoura, Algarve, Portugal, 2007.

[107] P. Hunt, M. Konar, F.P. Junqueira, and B. Reed. ZooKeeper : Wait-free Coordination for Internet-scale Systems. In *Proc. of the USENIX Annual Technical Conference*, pages 1–14, Boston, USA, June 2010.

[108] INCOME. Architecture of Multiscalable Context Management, Version 1. Technical report, ANR INCOME Project, Deliverable D3.1.a.1, December 2013.

[109] N. Islam and R. Want. Smartphones : Past, Present, and Future. *IEEE Pervasive Computing*, 12(4) :89–92, December 2014.

[110] ISO/IEC. Information technology — Software product evaluation — Quality characteristics and guidelines for their use. International Standard ISO/IEC-9126, ISO/IEC Joint Technical Committee, December 1991.

[111] ISO/IEC/IEEE. Systems and software engineering — Architecture description. International Standard ISO/IEC/IEEE-42010 :2011, ISO/IEC/IEEE Joint Technical Committee, December 2011.

[112] J. Jing, A. Helal, and A. Elmagarmid. Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, 31(2) :117–157, June 1999.

[113] A.D. Joseph, J.A. Tauber, and M.F. Kaashoek. Mobile Computing with the Rover Toolkit. *IEEE Transactions on Computers*, 46(3) :337–352, March 1997.

[114] Anthony D. Joseph, Alan F. deLespinasse, Joshua A. Tauber, D.K. Gifford, and M. Frans Kaashoek. Rover : A Toolkit for Mobile Information Access. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 156–171, December 1995.

[115] C. Julien and G.-C. Roman. EgoSpaces : Facilitating Rapid Development of Context-Aware Mobile Applications. *IEEE Transactions on Software Engineering*, 32(5) :281–298, May 2006.

[116] D. Katsaros and Y. Manopoulos. Web Caching in Broadcast Mobile Wireless Envrionments. *IEEE Internet Computing*, 8(3) :37–44, May 2004.

[117] N. Kefalakis, N. Leontiadis, J. Soldatos, and D. Donsez. Middleware Building Blocks for Architecting RFID Systems. In *Proc. of the 1st International ICST Conference, Athens, Greece, May 18-20, 2009, Revised Selected Papers*, number 13 in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 325–336, Athens, Greece, May 2009. Springer-Verlag.

[118] J.O. Kephart and D.M. Chess. The Vision of Autonomic Computing. *IEEE Computer*, 36(1) :41–50, January 2003.

[119] A.-M. Kermarrec and P. Triantafillou. XL Peer-to-Peer Pub/Sub Systems. *ACM Computing Surveys*, 46(2) :16:1–16:45, November 2013.

[120] M. Kessis, C. Roncancio, and A. Lefebvre. DASIMA : A Flexible Management Middleware in Multi-Scale Contexts. In *Proc. 6th International Conference on Information Technology : New Generations, 2009*, pages 1390–1396, 2009.

[121] G. Kiczales, J. des Rivires, and D.G. Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.

[122] G. Kiczales, J. Irwin, J. Lamping, J ;-M. Loingtier, C. Videria Lopes, C. Maeda, and A. Mendhekar. Aspect-Oriented Programming. *ACM Computing Surveys*, 28(4es), December 1996.

[123] J.J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. In *Proc. 13th ACM Symposium on Operating Systems Principles*, pages 213–225, Pacific Grove, USA, October 1991.

[124] J.J. Kistler and M. Satyanarayanan. Transparent Disconnected Operation for Fault-Tolerance, Position Paper. *ACM Operating Systems Review*, 25(1), January 1991.

[125] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen, and E.J. Malm. Managing Context Information in Mobile Devices. *IEEE Pervasive Computing*, 2(3) :42–51, July 2003.

[126] G. Kortuem, S. Fickas, and Z. Segall. On-Demand Delivery of Software in Mobile Environments. In *Proc. IPPS Workshop on Nomadic Computing*, Geneva, Switzerland, April 1997.

[127] N. Kouici. *Gestion des déconnexions pour applications réparties à base de composants en environnements mobiles*. PhD thesis, Institut National des Télécommunications, en co-accréditation avec l'Université d'Evry Val d'Essonne, Évry (France), November 2005.

[128] S. Krakowiak. Middleware Architecture with Patterns and Frameworks. `http://sardes.inrialpes.fr/~krakowia/MW-Book`, February 2009.

[129] P. Kruchten. The 4+1 View Model of Architecture. *IEEE Software*, 12(6) :42–50, November 1995.

[130] G.H. Kuenning and G.J. Popek. Automated Hoarding for Mobile Computers. In *Proc. 16th ACM Symposium on Operating Systems Principles*, pages 1–22, Saint Malo, France, October 1997.

[131] C. Kyungmin, H. Inseok, K. Seungwoo, K. Byoungjip, L. Jinwon, L. SangJeong, P. Souneil, S. Junehwa, and R. Yunseok. HiCon : a hierarchical context monitoring and composition framework for next-generation context-aware services. 44(8) :34–42, August 2008.

[132] L. Lamport. The Part Time Parliament. *ACM Transactions on Computer Systems*, 16(2) :133–169, May 1998.

[133] L. Lamport. Paxos Made Simple. *ACM SIGACT News (Distributed Computing Column)*, 32(4) :18–25, December 2001.

[134] M. Langheinrich. Privacy by Design—Principles of Privacy-Aware Ubiquitous Systems. In *Proc. of the 3rd international conference on Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 273–291, Atlanta, Georgia, USA, October 2001. Springer-Verlag.

[135] J.-C. Laprie. Dependable Computing : Concepts, Limits, Challenges. In *Proc. 25th IEEE Symposium on Fault Tolerant Computing*, pages 42–54, Pasadena, California, USA, June 1995.

[136] Jean-Claude Laprie. Sûreté de fonctionnement des systèmes informatiques et tolérance aux fautes : concepts de base. *Technique et Science Informatiques*, 4(5) :419–429, September 1985.

[137] M. Leclercq, V. Quéma, and J.-B. Stefani. DREAM : a Component Framework for the Construction of Resource-Aware, Configurable MOMs. *IEEE Distributed Systems Online*, 6(9), September 2005.

[138] M. Leclercq, A.E. Özcan, V. Quéma, and J.-B. Stefani. Supporting Heterogeneous Architecture Descriptions in an Extensible Toolset. In *Proc. 29th ACM International Conference on Software Engineering*, (USA), May 2007.

[139] L Leonini, É. Rivière, and P. Felber. SPLAY : Distributed systems evaluation made simple (or how to turn ideas into live systems in a breeze). In *Proc. of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pages 185–198, Boston, MA, USA, April 2009.

[140] L. Lim. *Gestion de groupe partitionnable dans les réseaux mobiles spontanés*. PhD thesis, Institut Mines Télécom, Télécom SudParis, en co-accréditation avec l'Université d'Évry Val d'Essonne, Évry (France), November 2012.

[141] J. MacCormick, N. Murphy, M. Najork, C.A. Thekkath, and L. Zhou. Boxwood : Abstractions As the Foundation for Storage Infrastructure. In *Proc. of the 6th USENIX Symposium on Operating Systems Design and Implementation*, San Francisco, California, USA, December 2004.

[142] V. Marangozova. *Duplication et cohérence configurables dans les applications réparties à base de composants*. PhD thesis, Université Jospeh Fourier, Grenoble, France, June 2003.

[143] N. Medvidovic and R.N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, 26(1) :70–93, January 2000.

[144] M. Mernik, J. Heering, and A.M. Sloane. When and How to Develop Domain-Specific Languages. *ACM Computing Surveys*, 37(4) :316–344, December 2005.

[145] M. Merritt and G. Taubenfeld. Computing with Infinitely Many Processes Under Assumptions on Concurrency and Participation. In *Proceedings of the 14th International Symposium on Distributed Computing*, volume 1914 of *Lecture Notes in Computer Science*, pages 164–178, Toledo, Spain, October 2000. Springer-Verlag.

[146] M. Merritt and G. Taubenfeld. Computing with Infinitely Many Processes. *Information and Computation*, 233 :12–31, December 2013.

[147] A. Mostefaoui, M. Raynal, C. Travers, S. Patterson, D. Agrawal, and A. El Abbadi. From Static Distributed Systems to Dynamic Systems. In *Proc. 24th IEEE Symposium on Reliable Distributed Systems*, pages 109–118, Florianpolis (Brazil), October 2005.

[148] G.K. Mostefaoui, J. Pasquier-Rocha, and P. Brézillon. Context-aware computing : A guide for the pervasive computing community. In *IEEE/ACS International Conference on Pervasive Services*, pages 39–48, Novi Sad (Serbia/Montenegro), July 2004.

[149] L.B. Mummert, M.R. Ebling, and M. Satyanarayanan. Exploiting Weak Connectivity for Mobile File Access. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 143–155, December 1995.

[150] A.L. Murphy, G.P. Picco, and G.C. Roman. Lime : A Middleware for Physical and Logical Mobility. In *Proc. 21st IEEE International Conference on Distributed Computing Systems*, pages 524–533, Mesa, Arizona (USA), April 2001.

[151] G. Mühl, L. Fiege, and P.R. Pietzuch. *Distributed Event-Based Systems*. Springer, 2006.

[152] B.D. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, and K.R. Walker. Agile Application-Aware Adaptation for Mobility. In *Proc. 16th ACM Symposium on Operating Systems Principles*, pages 276–287, Saint Malo, France, October 1997.

[153] E Nygren, R.K. Sitaraman, and J. Sun. The Akamai Network : A Platform for High-Performance Internet Applications. *ACM Operating Systems Review*, 44(3) :2–19, March 2010.

[154] OMG. Minimum CORBA. OMG Document orbos/1998-08-04, Object Management Group, August 1998.

[155] OMG. Portable Interceptor. Omg document ptc/2001-03-04, Object Management Group, March 2001.

[156] OMG. The Common Object Request Broker - Architecture and Specifications. Revision 2.4.2. OMG Document formal/01-02-01, Object Management Group, February 2001.

[157] OMG. Wireless Access and Terminal Mobility in CORBA. OMG Document dtc/01-06-02, Object Management Group, June 2001.

[158] OMG. CORBA Components. Version 3.0. OMG Document formal/02-06-65, Object Management Group, June 2002.

[159] OMG. MDA Guide Version 1.0. Omg document, omg/2003-05-01, Object Management Group, May 2003.

[160] UML Profile for Enterprise Distributed Object Computing (EDOC). Specification number formal/2004-02-01, February 2004.

[161] UML Profile for CORBA Components. Specification number formal/2005-07-06, July 2005.

[162] H. Parzyjegla. *Engineering Publish/Subscribe Systems and Event-Driven Applications*. PhD thesis, University of Rostock, Germany, December 2012.

[163] A. Peddemors, H. Zandbelt, and M. Bargh. A Mechanism for Host Mobility Management supporting Application Awareness. In *Proc. 2nd ACM/USENIX International Conference on Mobile Systems, Applications and Services*, pages 231–244, Boston, USA, June 2004.

[164] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context Aware Computing for The Internet of Things : A Survey. *IEEE Communications Surveys and Tutorials*, 16(1) :414–454, January 2014.

[165] K. Petersen, M.J. Spreitzer, D.B. Terry, M.M. Theimer, and A.J. Demers. Flexible Update Propagation for Weakly Consistent Replication. In *Proc. 16th ACM Symposium on Operating Systems Principles*, pages 288–301, Saint Malo, France, October 1997.

[166] M. Petrovic, I. Burcea, and H.-A. Jacobsen. S-ToPSS : Semantic Toronto Publish/Subscribe System. In *Proc. of the 29th International Conference on Very Large Data Bases*, pages 1101–1104, Berlin, Germany, 2003.

[167] E. Pitoura and B. Bhargava. Building Information Systems for Mobile Environments. In *Proc. 3rd International Conference on Information and Knowledge Management*, pages 371–378, Gaithesburg, MD, November 1994.

[168] S. Pleish, O. Rütti, and A. Schiper. On the Specification of Partitionable Group Membership. In *Proc. 7th European Dependable Computing Conference*, Kaunas, Lithuania, May 2008.

[169] D. Preuveneers and Y. Berbers. Adaptive Context Management Using a Component-Based Approach. In L. Kutvonen and N. Alonistioti, editors, *Proc. 5th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, volume 3543 of *Lecture Notes in Computer Science*, pages 14–26, Athens (Greece), June 2005. Springer-Verlag.

[170] D. Ratner, P. Reiher, G.J. Popek, and G.H. Kuenning. Replication Requirements in Mobile Environments. *ACM Mobile Networks and Applications*, 6(6) :525–533, November 2001.

[171] A.M. Ricciardi and K.P. Birman. Using process groups to implement failure detection in asynchronous environments. In *Proc. 10th ACM Symposium on Principles of Distributing Computing*, pages 341–353, Montreal, QC, Canada, August 1991.

[172] M. Rodrig and A. LaMarca. Oasis : an architecture for simplified data management and disconnected operation. *Personal and Ubiquitous Computing*, 9(2) :108–121, March 2005.

[173] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt. Gaia : A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, 1(4) :74–83, October 2002.

[174] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt. Gaia : a Middleware Platform for Active Spaces. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4) :65–67, October 2002.

[175] D. Romero, G. Hermosillo, A. Taherkordi, R. Nzekwa, R. Rouvoy, and F. Eliassen. The DigiHome Service-Oriented Platform. *Software—Practice and Experience*, 43(10) :1143–1239, October 1999.

[176] R. Rottenberg, S. Leriche, C. Taconet, C. Lecocq, and T. Desprats. MuSCa : A Multiscale Characterization Framework for Complex Distributed Systems. In *Proc. 3rd Workshop on Model Driven Approaches in System Development*, Warsaw, Poland, September 2014.

[177] S. Rottenberg, S. Leriche, C. Taconet, C. Lecocq, and T. Desprats. MuScA : A Multiscale Distributed Systems Scale-Awareness Framework. In *Actes de Conférence francophone sur l'Architecture Logicielle*, Paris, France, May 2014.

[178] R. Rouvoy and P. Merle. Leveraging Component-Based Software Engineering with Fraclet. *Annals ofs Telecommunications, pecial Issue on Software Components—The Fractal Initiative*, 64(1–2) :65–79, January 2006.

[179] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Herrmann, C. Kaiser, S. Langlois, P. Léonard, and W. Neuhauser. Chorus Distributed Operating Systems. *Computing Systems Journal, The USENIX Association*, 1(4) :305–369, December 1988.

[180] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Herrmann, C. Kaiser, S. Langlois, P. Léonard, and W. Neuhauser. Overview of the Chorus Distributed Operating System. In *USENIX Symposium on Micro-Kernels and Other Kernel Architectures*, pages 39–69, Seattle, USA, April 1992.

[181] R. Ruggaber, J. Seitz, and M. Knapp. Π^2— a Generic Proxy Platform for Wireless Access and Mobility in CORBA. In *Proc. 19th ACM Symposium on Principles of Distributing Computing*, pages 191–198, Portland, Oregon, USA, 2000.

[182] F. Sadri. Ambient Intelligence : A Survey. *ACM Computing Surveys*, 43(4) :36:1–36:66, October 2011.

[183] F. Sailhan and V. Issarny. Cooperative Caching in Ad Hoc Networks. In *Proc. of the 4th International Conference on Mobile Data Management*, volume 2574 of *Lecture Notes in Computer Science*, pages 13–28, Melbourne, Australia, January 2003. Springer-Verlag.

[184] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Computing Surveys*, 37(1) :42–81, March 2005.

[185] S. Sastry and S.M. Pike. Eventually Perfect Failure Detectors Using ADD Channels. In *Proc. 5th International Symposium on Parallel and Distributed Processing and Applications*, volume 4742 of *Lecture Notes in Computer Science*, pages 483–496, Niagara Falls, Canada, August 2007. Springer-Verlag.

[186] M. Satyanarayanan. Pervasive Computing : Vision and Challenges. *IEEE Personal Communications*, 8(4) :10–17, August 2001.

[187] M. Satyanarayanan. The Many Faces of Adapatation. *IEEE Pervasive Computing*, pages 4–5, July 2004.

[188] Mahadev Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proc. 15th ACM Symposium on Principles of Distributing Computing*, pages 1–7, Philadelphia, USA, May 1996.

[189] V. Schiavoni, P. Felber, and É. Rivière. SplayNet : Distributed User-Space Topology Emulation. In *Proc. 14th IFIP/ACM/USENIX International Middleware Conference*, volume 8275 of *Lecture Notes in Computer Science*, pages 62–81, Beijing, China, December 2013. Springer-Verlag.

[190] B.N. Schilit, N.I. Adams, and R. Want. Context-Aware Computing Applications. In *Proc. of the Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz, CA, USA, December 1994.

[191] Stal Schmidt, D.C., M., H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture : Volume 2, Patterns for Concurrent and Networked Objects*. Wiley, 2000.

[192] B.A. Schroeder. On-Line Monitoring : A Turorial. *IEEE Computer*, 28(6) :72–78, June 1995.

[193] A. Segall. Distributed Network Protocols. 29(1) :23–35, January 1983.

[194] A. Senart, R. Cunningham, M. Bouroche, N. O'Connor, V. Reynolds, and V. Cahill. MoCoA : Customisable Middleware for Context-Aware Mobile Applications. In *Proc. 8th International Symposium on Distributed Objects and Applications*, volume 4275 of *Lecture Notes in Computer Science*, pages 1722–1738, Montpellier (France), November 2006. Springer-Verlag.

[195] C. Shiva, J. Al-Muhtadi, R. Campbell, and M.D. Mickunas. Mobile Gaia : a middleware for ad-hoc pervasive computing. In *Proc. 2nd IEEE Conference on Consumer Communications and Networking Conference*, pages 223–228, January 2005.

[196] C. Szyperski. *Component Software : Beyond Object-Oriented Programming, 2nd edition*. Addison-Wesley, 2002.

[197] C. Taconet. Intergiciels pour la sensibilité au contexte en environnement ubiquitaire. Mémoire d'habilitation à diriger des recherches de l'Université d'Évry Val d'Essonne, Évry, France, February 2011.

[198] C. Taconet and Z. Kazi-Aoul. Building Context-Awareness Models for Mobile Applications. *Journal of Digital Information Management*, 8(2) :78–87, April 2010.

[199] Fondation Télécom. L'internet des objets, objets de l'Internet. Cahier de Veille de la Fondation Télécom, April 2011.

[200] D.B. Terry, M.M. Theimer, K. Petersen, and A. Demers. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 172–183, December 1995.

[201] G. Thiel. LOCUS operating System, a transparent system. *Computer Communications*, 14(6) :336–346, July 1991.

[202] M. Vadet and P. Merle. Les conteneurs ouverts dans les plates-formes à composants. In *Actes des Journées Composants*, Besançon, France, October 2001.

[203] M. van Steen, G. Pierre, and S. Voulgaris. Challenges in very large distributed systems. *Journal of Internet Services and Applications*, 3(1) :59–66, May 2012.

[204] O. Vermesan, P. Friess, G. Woysch, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, M. Eisenhauer, and K. Moessner. Europe's IoT Strategic Research Agenda. Technical report, Internet of Things European Research Cluster, 2012.

[205] J. Wang, B. Jin, and J. Li. An Ontology-based Publish/Subscribe System. In H.-A. Jacobsen, editor, *Proc. IFIP/ACM/USENIX International Middleware Conference*, volume 3231 of *Lecture Notes in Computer Science*, pages 232–253, Toronto, Canada, October 2004. Springer-Verlag.

[206] R. Want, B.N. Schilit, and S. Jenson. Enabling the Internet of Things. *IEEE Computer*, 48(1), January 2015.

[207] S. Wasserkrug, A. Gal, O. Etzion, and Y. Turchin. Complex Event Processing over Uncertain Data. In *Proc. 2nd ACM International Conference on Distributed Event-Based Systems*, pages 253–264, Roma, Italy, July 2008.

[208] M. Weiser. The Computer for the 21st Century. *Scientific American*, pages 94–100, September 1991.

[209] S. Williams, M. Abrams, C. Standridge, and E. Fox. Removal policies in network caches for world-wide web documents. In *Proc. ACM SIGCOMM*, Stanford University, CA, USA, 1996.

[210] S. Williams, M. Abrams, C.R. Strandridge, G. Abdulla, and E.A. Fox. Removal Policies in Network Caches for World-Wide Web Documents. *ACM SIGCOMM Computer Communication*, pages 293–305, May 1996.

[211] T. Winograd. Architectures for Context. *Special issue on context-aware computing in the Human-Computer Interaction Journal*, 16(2–4) :401–419, 2001.

[212] J. Woodcock, P.G. Larsen, J. Bicarregui, and J. Fitzgerald. Formal Methods : Practise and Experience. *ACM Computing Surveys*, 41(4) :19:1–19:36, October 2009.

[213] eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard, `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html`, January 2013.

[214] S.S. Yau, F. Karim, Y. Wang, B. Wang, and S.K.S. Gupta. Reconfigurable Context-Sensitive Middleware for Pervasive Computing. *IEEE Pervasive Computing*, 1(3) :33–40, July 2002.

[215] H. Yu and A. Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Transactions on Computer Systems*, 20(3) :239–282, August 2002.

[216] H. Yu and A. Vahdat. The costs and limits of availability for replicated services. *ACM Transactions on Computer Systems*, 24(1) :70–113, February 2006.

Webographie

[217] Aspirerfid. http://wiki.aspire.ow2.org/xwiki/bin/view/Main/.

[218] CORBA. http://www.omg.org/gettingstarted/history_of_corba.htm.

[219] COSMOS. http://picoforge.int-evry.fr/projects/svn/cosmos/.

[220] Cosmos DSL. http://picoforge.int-evry.fr/websvn/filedetails.php?repname=cosmos&path=/trunk/cosmos/dsl/dsl/cosmos.dsl/documentation/grammar/grammar.pdf.

[221] Cosmos Type System. http://picoforge.int-evry.fr/websvn/filedetails.php?repname=cosmos&path=/branches/v0.2.x_cosmos/cosmos/api/Documentation/Alloy/specification.pdf.

[222] DREAM. http://dream.ow2.org/.

[223] EMF. http://www.eclipse.org/modeling/emf.

[224] FPath. http://fractal.ow2.org/fscript/.

[225] Fraclet. http://fractal.ow2.org/fraclet/index.html.

[226] Fractal. http://fractal.ow2.org.

[227] Fractal ADL. http://fractal.ow2.org/fractaladl/.

[228] Google GSON. https://code.google.com/p/google-gson/.

[229] Julia. http://fractal.ow2.org/julia/index.html.

[230] Juliac. http://fractal.ow2.org/juliac/index.html.

[231] JXTA. https://jxta.kenai.com/.

[232] muDEBS. https://fusionforge.int-evry.fr/www/mudebs/.

[233] NanoXML. http://nanoxml.sourceforge.net/orig/index.html.

[234] OMNeT++. http://omnetpp.org/.

[235] PERSEUS. http://perseus.objectweb.org.

[236] RabbbitMQ. http://www.rabbitmq.com.

[237] Siafu. http://siafusimulator.org/.

[238] SimGrid. http://simgrid.gforge.inria.fr/.

[239] TLA+ Proof System. https://tla.msr-inria.inria.fr/tlaps/content/Home.html.

[240] Xtext. http://www.eclipse.org/Xtext.

Liste des publications

Articles dans une revue internationale

[241] L. Lim and D. Conan. Partitionable group membership for Mobile Ad hoc Networks. *Journal of Paralled and Distributed Computing*, pages 2708–2721, March 2014.

[242] S. Chabridon, D. Conan, Z. Abid, and C. Taconet. Building Ubiquitous QoC-Aware Applications through Model-Driven Software Engineering. *Elsevier Journal of Science of Computer Programming*, 78(10), October 2013.

[243] R. Rouvoy, D. Conan, and L. Seinturier. Software Architecture Patterns for a Context Processing Middleware Framework. *IEEE Distributed Systems Online*, 9(6), June 2008.

Articles dans une revue nationale

[244] D. Conan, R. Rouvoy, and L. Seinturier. COSMOS : composition de nœuds de contexte. *Technique et Science Informatiques*, 27(9–10) :1189–1224, 2008.

[245] N. Kouici, D. Conan, and G. Bernard. État de l'art de la gestion de cache logiciel pour tolérer les déconnexions en environnements mobiles. *Annales des Télécommunications*, 61(11–12) :1458–1483, December 2006.

[246] N. Kouici, D. Conan, and G. Bernard. Adaptation des applications réparties à base de composants aux terminaux mobiles en environnement sans fil. *Journal en ligne Informations, Savoirs, Décisions & Médiations*, 2004.

Papier dans une conférence internationale

[247] J.-P. Arcangeli, A. Bouzeghoub, V. Camps, M.-F. Canut, S. Chabridon, D. Conan, T. Desprats, R. Laborde, E. Lavinal, S. Leriche, H. Maurel, A. Péninou, C. Taconet, and P. Zaraté. INCOME—Multi-Scale Context Management for the Internet of Things. In *Proc. International Joint Conference on Ambient Intelligence*, Pisa, Italy, November 2012.

[248] L. Lim and D. Conan. An Eventual α Partition-Participant Detector for MANETs. In *Proc. 9th European Dependable Computing Conference*, Sibiu, Romania, May 2012.

[249] S. Chabridon, Z. Abid, C. Taconet, and D. Conan. A Model Driven Approach for the QoC-Awareness of Ubiquitous Applications. In *Proc. 5th International Symposium on Ubiquitous Computing and Ambient Intelligence*, México, Mexico, December 2011.

[250] B. Jovanova, I. Arsov, D. Conan, D.-T. Anh, A. Ozanne, and M. Preda. Mobile Mixed Reality Games Creator Based on MPEG-4 BIFS. In *Proc. IEEE International Conference on Multimedia and Expo, Industrial Program*, Barcelona, Spain, July 2011.

[251] S. Chabridon, C.-C. Ngo, Z. Abid, D. Conan, C. Taconet, and A. Ozanne. Towards QoC-Aware Location-based Services. In *Proc. 11th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, Lecture Notes in Computer Science, Reykjavik, Iceland, June 2011. Springer-Verlag.

[252] L. Arantes, P. Sens, G. Thomas, D. Conan, and L. Lim. Partition Participant Detector with Dynamic Paths in Mobile Networks. In *Proc. 10th IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, July 2010.

[253] A. Bouzeghoub, C. Taconet, A. Jarraya, N.K. Do, and D. Conan. Complementarity of Process-oriented and Ontology-based Context Managers to Identify Situations. In *Proc. 5th International Conference on Digital Information Management*, pages 222–229, Thunder Bay, Canada, June 2010.

[254] J. Ribault, O. Dalle, D. Conan, and S. Leriche. OSIF : A Framework To Instrument, Validate, and Analyze Simulations. In *Proc. 3rd International ICST Conference on Simulation Tools and Techniques*, Malaga, Spain, March 2010.

[255] C. Taconet, Z. Kazi-Aoul, M. Zaier, and D. Conan. CA3M : A runtime model and a middleware for dynamic context management. In *Proc. 11th International Symposium on Distributed Objects and Applications*, volume 5870 of *Lecture Notes in Computer Science*, pages 513–530, Algarve, Portugal, November 2009. Springer-Verlag.

[256] D. Conan, P. Sens, L. Arantes, and M. Bouillaguet. Failure, Disconnection and Partition Detection in Mobile Environment. In *Proc. 7th IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, July 2008.

[257] D. Conan, R. Rouvoy, and L. Seinturier. Scalable Processing of Context Information with COSMOS. In J. Indulska and K. Raymonds, editors, *Proc. 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable*

Systems, volume 4531 of *Lecture Notes in Computer Science*, pages 210–224, Paphos, Cyprus, June 2007. Springer-Verlag.

[258] N. Kouici, L. Chateigner, D. Conan, S. Chabridon, and G. Bernard. Disconnection management for distributed component-based applications in mobile environments. In *Proc. 7th International Symposium on Programming and Systems*, Algiers, Algeria, May 2005.

[259] N. Kouici, D. Conan, and G. Bernard. Caching Components for Disconnection Management in Mobile Envrionments. In Z. Tari *et al*, editor, *Proc. 6th International Symposium on Distributed Objects and Applications*, volume 3291 of *Lecture Notes in Computer Science*, pages 1322–1339, Agia Napa, Cyprus, October 2004. Springer-Verlag.

[260] D. Conan, C. Taconet, D. Ayed, L. Chateigner, N. Kouici, and G. Bernard. A Pro-Active Middleware Platform for Mobile Environments. In *Proc. of IASTED International Conference on Software Engineering*, Innsbruck, Austria, February 2004.

[261] N. Kouici, D. Conan, and G. Bernard. Disconnection Metadata for Distributed Applications in Mobile Environments. In *Proc. International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, Nevada (USA), June 2003.

[262] D. Conan, P. Taponot, and G. Bernard. Rollback Recovery of PVM Applications. In *Proc. of ROSE '95 - The Romanian Open Systems Event*, pages 15–23, Bucharest, Romania, November 1995.

[263] G. Bernard and D. Conan. Flexible Checkpointing and Efficient Rollback-Recovery for Distributed Computing. In *Proc. SUUG '94 Conference*, Moscow, Russia, April 1994.

[264] G. Bernard and D. Conan. Making Distributed Applications Fault-Tolerant in Networks of Unix Workstations. In *Proc. I2U Convention '93*, Milano, Italy, May 1993.

Papier dans une conférence nationale

[265] N. Masmoudi and D. Conan. Contrats de contexte pour la gestion de contexte répartie. In *Actes de la 9ème Conférence Francophone Mobilité et Ubiquité*, Nancy, France, June 2013.

[266] S. Chabridon, D. Conan, Z. Abid, and Z. Taconet. Ingénierie dirigée par les modèles pour la construction d'applications ubiquitaires sensibles à la qualité du contexte. In *Actes de la 8ème Conférence Francophone Mobilité et Ubiquité*, pages 167–176, Toulouse, France, June 2012.

[267] D. Conan. Composition d'entités de contexte de ressources système. In *Actes de la 3ème Conférence Francophone Mobilité et Ubiquité*, ACM International Conference Proceeding Series, pages 111–114, Paris, France, September 2006.

[268] T.D. Nguyen and D. Conan. Gestion de groupes tolérant les défaillances et les déconnexions en environnement mobile. In *Proc. New Technologies for Distributed Systems*, Toulouse, France, 2006.

[269] M. Bhatti and D. Conan. Détection de partition pour la gestion de groupes en environnement mobile. In *Actes de la 2ème Conférence Francophone Mobilité et Ubiquité*, volume 120 of *ACM International Conference Proceeding Series*, pages 65–72, Grenoble, France, June 2005.

[270] L. Temal and D. Conan. Détections de défaillances, de connectivité et de déconnexions. In *Actes de la 1ère Conférence Francophone Mobilité et Ubiquité*, volume 64 of *ACM International Conference Proceeding Series*, pages 90–97, Nice, France, June 2004.

[271] N. Kouici, D. Conan, and G. Bernard. MADA : une approche pour le développement d'applications mobiles. In *Actes de la 1ère Conférence Francophone Mobilité et Ubiquité*, volume 64 of *ACM International Conference Proceeding Series*, pages 78–85, Nice, France, June 2004.

[272] N. Kouici, D. Conan, and G. Bernard. Adaptation des applications réparties à base de composants aux terminaux mobiles en environnement sans fil. In *Premier Congrès Francophone MAJECSTIC (MAnifestation des JEunes Chercheurs STIC)*, Marseille, France, October 2003.

Chapitres de livre

[273] D. Romero, R. Rouvoy, S. Chabridon, D. Conan, N. Pessemier, and N. Seinturier. *Enabling Context-Aware Web Services : A Middleware Approach for Ubiquitous Environments*, chapter 5, pages 111–138. Chapman and Hall/CRC, 2009.

[274] D. Conan and G. Bernard. La reprise sur erreur par recouvrement arrière automatique dans les systèmes répartis. In J.-F. Myoupo, editor, *Parallèlisme et Répartition*, chapter 4, pages 91–123. Hermès, April 1998.

[275] C. Bac, G. Bernard, D. Conan, Q. Nguyen, and C. Taconet. Experience with Chorus. In M. Bartosek, J. Staudek, and J. Wiedermann, editors, *Proc. of SOFSEM '95 : Theory and Practice of Informatics*, volume 1012 of *Lecture Notes in Computer Science*, pages 272–291. Springer-Verlag, December 1995.

Papiers dans un *workshop*

[276] P. Marie, L. Lim, A. Manzoor, S. Chabridon, D. Conan, and T. Desprats. QoC-Aware Context Data Distribution in the Internet of Things. In *Proc. Middleware Workshop on Middleware for Context-Aware Applications in the IoT*, Bordeaux, France, December 2014.

[277] L. Lim and D. Conan. Distributed Event-Based System with Multiscoping for Multiscalability. In *Proc. 9th Middleware Workshop on Middleware for Next Generation Internet Computing*, Bordeaux, France, December 2014.

[278] L. Lim and D. Conan. Toward a Solution to Partitionable Group Membership for MANETs. In *Proc. 1st OPODIS International Workshop on Dynamicity*, pages 25–36, Toulouse, France, December 2011.

[279] S. Chabridon, D. Conan, C. Taconet, C.K. Nguyen, C.C. Ngo, L. Lim, and Z. Abid. MDE, DSL and Tooling for Effective Context Management in Ubiquitous Computing. In *Proc. MobiCASE Workshop on Mobile Software Engineering*, Santa Clara, CA, USA, October 2010.

[280] Z. Abid, S. Chabridon, and D. Conan. Cohérence et qualité des informations de contexte en environnement pervasif. In *Proc. 3ème Workshop sur la Cohérence des Données en Univers Réparti - CDUR'09*, Toulouse, France, September 2009.

[281] Z. Abid, S. Chabridon, and D. Conan. A Framework for Quality of Context Management. In *Proc. First International Workshop on Quality of Context*, volume 5786 of *Lecture Notes in Computer Science*, Stuttgart, Germany, June 2009. Springer-Verlag.

[282] A. Beugnard, S. Chabridon, D. Conan, C. Taconet, F. Dagnat, and E. Kaboré. Towards context-aware components. In *Proc. ESEC/FSE Workshop on Context-Aware Software Technology and Applications*, Amsterdam, The Netherlands, August 2009.

[283] M.-C. Monget, D. Bouillet, and D. Conan. The IniTuX system : A new way to learn GNU/Linux. In *Proc. International Conference on Educational Multimedia, Hypermedia,& Telecommunications*, Montreal, Canada, June 2005.

[284] N. Kouici, D. Conan, and G. Bernard. An experience in adaptation in the context of mobile computing. In *Proc. of 2nd ECOOP workshop on Coordination and Adaptation Techniques for Software Entities*, pages 47–54, Glasgow, UK, July 2005.

[285] D. Bouillet, D. Conan, and M.-C. Monget. Un dispositif d'apprentissage innovant pour débuter avec GNU/Linux. In *Actes du 4è Colloque International sur les Technologies de l'Information et de la Connaissance dans l'Enseignement Supérieur et l'Industrie*, Compiègne, France, October 2004.

[286] M.U. Bhatti and D. Conan. Fault-tolerance in Mobile Environments : A Partition Detection System. In *Proc. of 2nd International Workshop on Frontiers of Information Technology*, Islamabad, Pakistan, December 2004.

[287] N. Kouici, D. Conan, and G. Bernard. Intégration d'un service de gestion des déconnexions dans les conteneurs des composants. In *Actes des Journées Composants*, Lille, France, March 2004.

[288] D. Conan, S. Chabridon, O. Villin, G. Bernard, A. Kotchanov, and T. Saridakis. Handling Network Roaming and Long Disconnections at Middleware Level. In *Workshop on Software Infrastructures for Component-Based Applications on Consumer Devices (in conjunction with EDOC'2002)*, Lausanne, Switzerland, September 2002.

[289] D. Conan, S. Chabridon, and G. Bernard. Disconnected Operations in Mobile Environments. In *Proc. 2nd IPDPS Work. on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Ft. Lauderdale, Florida, USA, April 2002.

[290] D. Conan, É. Putrycz, N. Farcet, and M. DeMiguel. Integration of Non-Functional Properties in Containers. In *Proc. 5th ECOOP Workshop on Component Oriented Programming*, Budapest, Hungary, June 2001.

[291] D. Conan, M. Coriat, and N. Farcet. A Software Component Development Meta-Model for Product Lines. In *Proc. 4th ECOOP Workshop on Component Oriented Programming*, Lisboa, Portugal, June 2000.

[292] D. Conan. Message delivery semantics for the rollback-recovery of undeterministic processes. In A. Schiper and M. Shapiro, editors, *Proc. of 2nd European Research Seminar in Distributed Systems*, pages 210–215, Zinal, Switzerland, 1997.

Rapports et autres publications

[293] D. Conan, S. Chabridon, O. Villin, and G. Bernard. Domint : Weak Connectivity and Disconnected CORBA Objects on Hand-Held Devices. Technical report, Institut National des Télécommunications, Évry, France, January 2003.

[294] D. Conan, S. Chabridon, O. Villin, and G. Bernard. Weak Connectivity and Disconnected CORBA Objects on Hand-Held Devices. In *Addendum to Proc. International Symposium on Distributed Objects and Applications*, Irvine, California, USA, October 2002.

[295] D. Conan, S. Chabridon, O. Villin, and G. Bernard. A Platform for Experimenting Disconnected Objects on Mobile Hand-Held Devices. In *Actes 18èmes Journées Bases de Données Avancées, Démonstrations*, Évry, France, December 2002.

[296] D. Conan, G. Bernard, and S. Chabridon. Gestion des déconnexions en environnement mobile. In *Poster du Colloque sur Mobiles-Services et réseaux mobiles de 3ème Génération*, Lyon, France, December 2001.

[297] D. Conan, B. Bretelle, S. Chabridon, and G. Bernard. Support pour l'exécution en mode déconnecté d'applications distribuées dans les environnements mobiles. In *Actes du Séminaire Objets Communicants*, Grenoble, France, October 2001.

[298] D. Conan. Product-line software components. Technical report, Common Laboratory Alcatel-CRC/Thomson-CSF, August 2000.

[299] D. Conan. Note on the link with Perco. Technical report, Common Laboratory Alcatel-CRC/Thomson-CSF, August 2000.

[300] D. Conan. Link with the computing infrastructure. Technical report, Common Laboratory Alcatel-CRC/Thomson-CSF, August 2000.

[301] D. Conan. *Tolérance aux fautes par recouvrement arrière dans les systèmes informatiques répartis*. PhD thesis, Université Pierre et Marie Curie, PARIS VI (France), September 1996.