CONTRIBUTIONS À LA RÉSILIENCE ET AU RESPECT DE LA VIE PRIVÉE DES SYSTÈMES MOBIQUITAIRES

Marc-Olivier Killijian

Manuscrit

présenté pour l'obtention de l'

Habilitation à diriger les recherches

de l'Université Toulouse 3 Paul Sabatier

Spécialité : Informatique

par

Marc-Olivier KILLIJIAN

LABORATOIRE D'ANALYSE ET D'ARCHITECTURE DES SYSTÈMES
École Doctorale Mathématiques, Informatique et Télécommunications de Toulouse

CONTRIBUTIONS À LA RÉSILIENCE ET AU RESPECT DE LA VIE PRIVÉE DES SYSTÈMES MOBIQUITAIRES

Soutenue le 20 février 2013, devant le jury :

Rapporteurs :	Valérie ISSARNY	Directeur de Recherche INRIA
	Philippe SCHNOEBELEN	Directeur de Recherche CNRS
	Hans-Peter SCHWEFEL	Prof. Aalborg University
Examinateurs :	Claude CASTELLUCCIA	Directeur de Recherche INRIA
	Abdelmalek BENZEKRI	Prof. Université de Toulouse 3 - Paul Sabatier
Parrain :	Yves DESWARTE	Directeur de Recherche CNRS

« *Le paradoxe de la condition humaine, c'est qu'on ne peut devenir soi-même que sous l'influence des autres.* »

Boris CYRULNIK (1937-)

Avant-propos

Les travaux présentés dans ce mémoire ont essentiellement été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS). La direction d'équipe et de laboratoire a un sens particulier au LAAS et j'aimerais remercier les personnes qui acceptent de se dévouer pour faire que cet énorme machine fonctionne. J'ai des pensées particulières pour plusieurs d'entre elles : Jean-Claude Laprie qui, à l'encontre de l'image qu'il pouvait parfois donner, m'a montré ce qu'était la liberté d'un chercheur, en m'accordant toute sa confiance ; Raja Chatila qui, grâce aux "projets LAAS", a permis à la thématique de la geoprivacy d'émerger au laboratoire ; Karama Kanoun qui, avec une touche toute personnelle, sait porter notre équipe avec bonheur.

J'aimerais remercier les collègues qui ont accepté de m'accompagner dans l'étape de ma vie professionnelle que constitue cette HDR. Je remercie en particuliers les rapporteurs, Valérie Issarny, Philippe Schoebelen et Hans-Peter Schwefel, et les membres du jury Claude Castelluccia et Abdelmalek Benzekri. Je remercie tout particulièrement Yves Deswarte pour ses relectures détaillées et ses bonnes idées. Merci à Maud et Marie Leriche pour leurs relectures efficaces de ce manuscrit.

Mes collègues du laboratoire sont également à "blâmer". D'énormes pans du travail présenté dans ce manuscrit s'est fait avec eux. Je ne vais pas tous les citer, ils se reconnaîtrons mais je donne néanmoins quelques mentions spéciales à Jean-Charles Fabre pour avoir accompagné mon envol de si belle manière et David Powell pour m'avoir fait confiance dans mes idées parfois loufoques.

Enfin, *but not least*, tout celà n'aurait pu être possible sans le soutien de ma femme Maud, merci à toi.

Table des matières

Table des figures

Introduction générale

Ce document essaie de résumer les travaux sur lesquels ma recherche a porté durant les 13 années passées. Durant ce temps, relativement long, j'ai évidemment abordé de multiples sujets, tous liés à la résilience des systèmes "interconnectés". Dans cette introduction générale, je propose un balayage rapide de ces travaux afin de mettre en lumière leurs points communs et leurs différences.

Le point de départ de mes travaux, commencés durant ma thèse, a concerné la tolérance aux fautes pour les systèmes distribués, sous un angle architecture et langage. Dans ces travaux, j'étudiais l'utilisation de la réflexivité à la compilation et à l'exécution afin de faciliter l'implémentation de mécanismes de tolérance aux fautes. Les techniques que j'ai développées permettaient l'adjonction de mécanismes de réplication de façon transparente à une application CORBA. D'autres résultats issus de ces travaux concernaient la sérialisation portable (C++ ou Java) d'objets : un objet C++ peut être sérialisé afin de sauvegarder son état, lequel état peut être restauré, par désérialisation, dans un objet Java. Ces travaux concernant l'utilisation du concept de réflexivité ont été poursuivis dans le cadre de travaux théoriques, concernant la *réflexivité multi-niveaux*, ou plus pratiques, comme la mise en œuvre de la réflexivité sur des composants sur étagères, dans une architecture logicielle embarquée, ou pour permettre l'adaptation de mécanismes de tolérance aux fautes à l'exécution. Ces différents travaux, dans leurs aspects les plus pratiques, ont été poussés jusqu'au transfert industriel. L'idée première qui était sous-jacente à ceux-ci concernait la séparation des problématiques : comment permettre, par des artifices au niveau du langage ou de l'intergiciel, à un développeur d'application d'ajouter simplement, et de façon la plus transparente possible, des mécanismes de tolérance aux fautes. Le chapitre 1 présente ces différents travaux.

Par la suite, durant mon post-doc, je me suis penché sur la tolérance aux fautes dans les systèmes mobiles. Dès le départ mon angle d'attaque a été de rechercher ce qui est spécifique dans le caractère mobile de ces systèmes. Les premiers travaux m'ont mené à étudier la notion de communication de groupes géographiques : comment définir un groupe d'entités communicantes en fonction de leur localisation respective ou en fonction de leur proximité. La question connexe de l'adressage géographique de nœuds a émergé, par ailleurs, sous le vocable de *géo-casting*, mais sans les propriétés associées aux communications de groupe telles qu'atomicité, ordre, etc. Néanmoins, ces travaux m'ont mis sur la piste de ce qui a constitué mon inspiration principale par la suite : comment tirer partie de la mobilité plutôt que de la

subir et de ne la voir que comme une source de problèmes (déconnexions, partitionnements, etc.) ? Pour répondre à cette question j'aurais pu prendre le train des réseaux tolérant aux délais (Delay-Tolerant Networks), mais j'ai plutôt choisi la piste de la décentralisation extrême, les réseaux pair-à-pair, qui me semblait une solution plus générale. Sous l'angle du pair-à-pair, j'ai donc proposé un système de sauvegarde coopérative de données, où les nœuds mobiles participants offrent un service de stockage sécurisé qu'ils peuvent utiliser afin de sauvegarder leurs données critiques. Cette solution a été également déclinée pour offrir un système de *boîte noire virtuelle* pour l'automobile. Ces travaux ont été traités sous des angles algorithmique et architecture, mais également sous l'angle de l'évaluation de la sûreté de fonctionnement. Ces travaux font l'objet du chapitre 2.

Dans le cadre de mes recherches sur la résilience des systèmes mobiquitaires, j'ai abordé l'étude de la mobilité humaine en équipant des collègues-cobayes de smartphones qui enregistraient notamment leurs déplacements. Ces travaux ont bien évidemment soulevé des questionnements d'ordre déontologique : comment exploiter ces données tout en préservant la vie privée des cobayes ? C'est à cette occasion que j'ai commencé à m'intéresser à ce que je nomme maintenant la *geoprivacy*. Ce domaine représente maintenant la majeure partie de mes travaux, tant sous l'angle des attaques que sous celui de la protection. Nous proposons un modèle Markovien de mobilité individuelle, outil à la fois compact, précis, intelligible et facilement adaptable pour représenter la mobilité d'un individu. Nous avons proposé plusieurs attaques qui reposent sur l'utilisation de ce modèle. Elles ciblent la prédiction des déplacements à venir, ou encore la des-anonymisation, et atteignent une efficacité au moins équivalente à l'état de l'art. En ce qui concerne la protection de la geoprivacy, nous travaillons actuellement sur des abstractions de niveau intergiciel, tel les locanymes ou les localisation vérifiées, afin de proposer une architecture sûre et respectueuse de la vie privée pour les systèmes géolocalisés. Le chapitre 3 aborde ces aspects de ma recherche.

Ces différents travaux se sont enchaînés dans un mélange de hasard, de sérendipité et de poursuite d'un objectif commun : fournir des moyens algorithmiques et architecturaux pour la résilience des systèmes informatiques actuels, à savoir distribués, mobiles, ubiquitaires. Les techniques et outils que j'utilise pour aborder cette problématique large auront été divers et variés, cela participe à mon expérience, sans cesse renouvelée. En effet, mon plaisir dans ces recherches passe par l'exploration et j'espère que le voyage que je vous propose dans ce manuscrit vous sera également agréable. Le chapitre 4 conclut ce document en livrant quelques réflexions et pistes de prospectives.

Il est très important de noter que les travaux présentés dans ce manuscrit sont pour la plupart, et comme l'immense majorité des travaux de recherche, au moins dans nos domaines, sont des travaux collectifs, même si je les présente comme étant personnels. Je profite de cette parenthèse pour remercier tous mes collègues, stagiaires, doctorants, ingénieurs, techniciens et, bien entendu, chercheurs pour leur aide, amitié, inspiration, etc.

Chapitre 1

Usage de la réflexivité pour la construction d'architectures sûres

Thèses	00	Marc-Olivier Killijian	[72]
	04	François Taïani	[133]
	05	Taha Bennani	[8]
	09	Caroline Lu	[99]
		Thomas Pareaud	[114]
Publications	98	SRDS	[76]
	99	Reflection	[86]
	00	SRDS	[74]
		Reflection and Software Engineering	[139]
	02	PRDC	[134]
		OOPSLA	[87]
	03	IEEE Trans. on Computers	[126]
		IPDPS	[75]
		DSN	[135]
	04	DSN	[9]
	05	DSN	[136]
	06	TSI	[137]
	08	PRDC	[115]
	09	SPE	[138]
		ERTFA	[101]
		RTNS	[100]
	10	EDCC	[45]

Les travaux résumés dans ce chapitre portent globalement sur l'utilisation de techniques logicielles telles que la réflexivité, les protocoles à méta-objets, les composants réflexifs, pour la construction d'architectures distribuées sûres de fonctionnement. Ces travaux ont commencé au LAAS un peu avant ma thèse, avec les travaux de Tanguy Pérennou [116], et ont perduré jusqu'à leur transfert vers l'industrie dans le cadre de la thèse de Caroline Lu [99] en contrat CIFRE avec Renault.

L'idée de base consiste à vouloir séparer le développement des mécanismes de tolérance aux fautes de celui du logiciel applicatif, et de lier ces deux aspects plus tard, à la compilation ou à l'exécution (section 1.3). Ces travaux ont mené à des recherches connexes comme celles touchant au transfert d'état portable (section 1.2), à l'utilisation de mécanismes sur étagère plutôt que pleinement réflexifs (section 1.5.1), ou encore à la prise en compte de l'aspect multi-couche des architectures actuelles (section 1.4). Dernièrement, nous nous sommes penchés, d'une part, sur l'utilisation de composants réfléxifs pour l'adaptabilité de la tolérance aux fautes (section 1.6) et, d'autre part, sur l'utilisation de la réflexivité pour la tolérance aux fautes dans les architectures embarquées automobiles (section 1.5.2).

Les travaux résumés dans ce chapitre ont été effectués dans le cadre de ma thèse de doctorat [72], ainsi que dans celles de François Taïani [133], Taha Bennani [8], Thomas Pareaud [114] et Caroline Lu [99], dont j'ai assuré l'encadrement conjointement avec Jean-Charles Fabre. Ces travaux ont été effectués, en partie, dans le cadre des projets européens DeVa, DSOS et ResIST et du projet ANR SCARLET.

1.1 Introduction : De la réflexivité pour la séparation des problématiques

A haut niveau d'abstraction, la réflexivité est une propriété qui permet à un élément de s'auto-contrôler et de s'auto-modifier. D'un point de vue conceptuel, on peut définir la réflexivité comme la propriété qui permet à un composant d'observer et de contrôler sa propre structure et son propre comportement. Un composant réflexif dispose donc d'un (méta-)modèle de lui-même. Ce méta-modèle peut porter sur des aspects statiques (comme la structure, l'héritage, la composition) ou dynamique (typiquement, son comportement). Ce méta-modèle est lié causalement au composant, ce qui implique qu'une modification de l'un entraîne la modification de l'autre. Enfin, ce méta-modèle du composant de base est contrôlé par un méta-composant qui dispose donc de mécanismes d'introspection, de réification et d'intercession pour agir sur le composant de base. Dans le monde orienté-objet, on parle de protocole à méta-objet, qui intervient entre les composants de base (les objets) et leurs méta-composants (les méta-objets). Les interactions entre objets et méta-objets sont régies par un protocole à méta-objets (PMO).

Les notions de réflexivité et les protocoles à méta-objets (ainsi que, plus récemment, la programmation orientée aspects) sont souvent utilisés pour promouvoir la séparation des problématiques (*separation of concerns*). Il nous a paru séduisant d'utiliser ces outils pour séparer mécanismes de tolérance aux fautes et logiciel applicatif. Dans la thèse de Tanguy Perennou [116], il est fait usage d'un protocole à méta-objets simples issu du compilateur réflexif OpenC++ v1, pour greffer à une application distribuée des mécanismes de réplication passive ou semi-active, et ce au moment de la compilation. La mécanique proposée est relativement simple : le développeur d'application utilise le compilateur OpenC++ pour compiler son application. Ce compilateur adjoint aux objets applicatifs un protocole à méta-objets défini, composé de méthodes de réification des appels de méthodes et des accès aux attributs de l'objet, de méthodes d'intercession permettant de réaliser des appels de méthodes ou de modifier des attributs de l'objet à partir du méta-objet, et de méthodes d'introspection permettant notamment de lire les attributs de l'objet. Le programmeur de tolérance aux fautes écrit donc le code des méta-objets en C++. Ces méta-objets interagissent avec les objets par le biais du protocole à méta-objet pour réaliser un mécanisme de tolérance aux fautes par réplication. Il crée également un fichier de configuration qui spécifie quels méta-objets sont à associer aux objets de l'application. A l'exécution, la création d'un objet de base, entraîne la création du méta-objet associé qui "capture" les appels de méthodes pour les envoyer aux répliques dans le cas de réplication active/semi-active, et qui utilise les méthodes d'introspection pour réaliser le transfert d'état entre les répliques.

Ces travaux ont montré que l'utilisation d'un protocole à méta-objet permettait de séparer, de façon élégante, code applicatif et mécanismes de tolérance aux fautes. Toutefois, la

relative pauvreté du protocole à méta-objet imposé était un facteur limitant et nous a incité à poursuivre ces recherches.

Dans ma thèse [72], nous nous sommes penché explicitement sur la conception d'un protocole à méta-objets spécifiquement dédié à la tolérance aux fautes dans les applications distribuées, dans le cadre d'une architecture CORBA, et en particulier par l'utilisation de la réflexivité à la compilation. L'approche que nous avons proposée permet donc de mettre en place, à la compilation, une architecture CORBA tolérante aux fautes, où clients et serveurs sont liés par un protocole à méta-objets dédié. Ce protocole à méta-objets permet, d'une part, le contrôle du comportement et de l'état interne des objets CORBA, et d'autre part, le contrôle des liens entre clients et serveurs ainsi qu'entre objets et méta-objets, le tout de façon dynamique. Dans la suite de cette section, nous présentons ces travaux sous deux aspects : tout d'abord, en section 1.2, des travaux qui ont porté sur la sérialisation d'objets portables, c'est-à-dire le transfert de l'état interne d'objets applicatifs entre Java et C++ ; puis, en section 1.3, la conception, l'implémentation et l'instanciation du protocole à méta-objet et de l'architecture CORBA proposées dans ma thèse.

1.2 Serialisation portable d'objets CORBA

Thèse : [72] - Publication majeure : [87]

La gestion de l'état des objets est une problématique importante en programmation distribuée, par exemple pour permettre la migration des objets, leur persistance, ou encore pour leur réplication dans le cadre de mécanismes de tolérance aux fautes. Alors que Java propose le mécanisme de sérialisation pour la sauvegarde de l'état des objets, il n'en va pas de même pour d'autres langages courants, en particulier C++. Alors que CORBA prône l'interopérabilité, il nous a paru opportun de proposer un mécanisme portable de sérialisation, entre des incarnations d'objets issus de différents langages de programmation. Dans [87] nous avons donc proposé une approche basée sur la réflexivité à la compilation qui permet de fournir un mécanisme de sauvegarde de l'état des objets CORBA de façon interopérable.

Notre approche se basait donc sur l'utilisation de deux compilateurs ouverts, l'un pour C++ [25] et l'autre pour Java [139], afin de générer à partir de la déclaration d'une classe (i) des conteneurs CORBA pour l'état des objets, et (ii) les mécanismes C++ et Java nécessaires pour sauvegarder et restaurer l'état des objets à partir de ces conteneurs. Cette approche comporte de nombreux avantages : les conteneurs d'état sont générés automatiquement ; grâce à l'utilisation de ces conteneurs, la notion d'état des objets est typée, ce qui implique une utilisation plus rigoureuse ; les conteneurs contiennent également la description de leur propre structure, ce qui permet d'envisager par exemple une exploration dynamique de l'état des objets.

La sérialisation portable des objets implique évidemment un modèle structurel commun aux différents langages envisagés, en l ?occurrence C++ et Java. Un tel modèle doit couvrir, d'une part, le système de typage, et d'autre part, les caractéristiques orientées-objet qu'il est capable de supporter. Le modèle que nous avons proposé se base sur la spécification du langage IDL de CORBA [111] pour le système de typage. Il supporte donc des types simples comme : les types de base (*short*, *int*, *char*, par exemple), les chaînes (*String*), les références aux objets CORBA, les classes pour la gestion de l'héritage (simple) et de la composition. Notre système de typage offre également le support de types composés comme les structures, tableaux, séquences ou encore des types définis par l'utilisateur (qui font référence à d'autres types simples ou composés).

Algorithme 1 Interface PSerializable pour la sérialisation portable

```
1: typedef sequence<octet> State ;
2: interface PSerializable {
3:       State get_state() ;
4:      void set_state(in State state) ;
5: }
```

Les détails d'implémentation peuvent être trouvés dans [87], le processus de compilation est illustré figure 1.1 : le programmeur d'application fournit la spécification de l'interface de

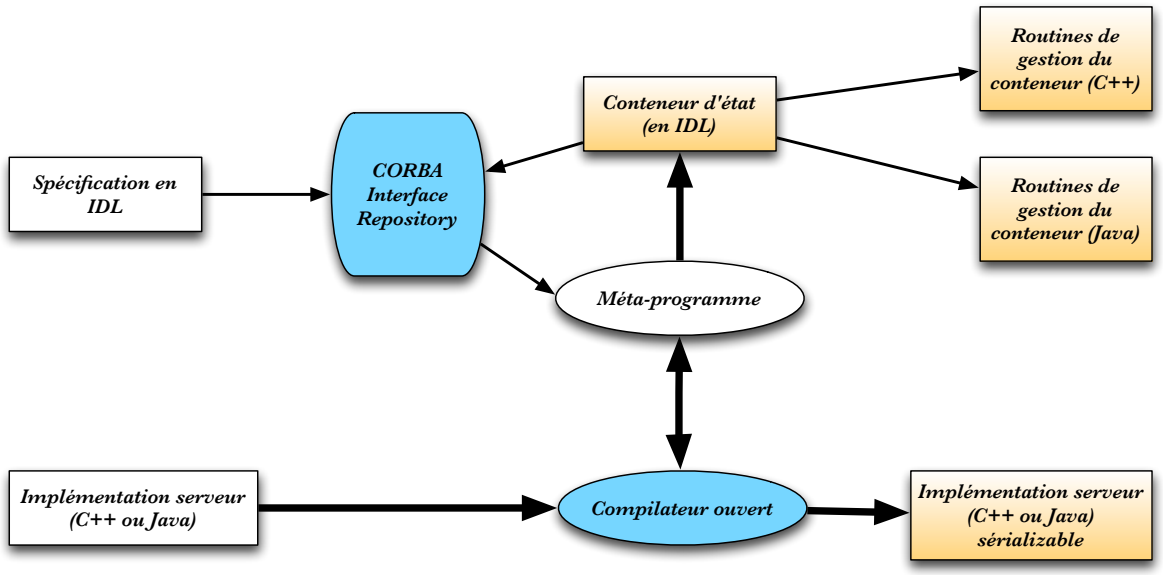


FIGURE 1.1 – Le processus de compilation pour la sérialisation portable

son serveur en IDL et sa réalisation en C++ ou en Java, le compilateur ouvert piloté par le méta-programme génère le conteneur d'état en IDL et une version augmentée de la sérialisation portable de l'implémentation du serveur, enfin, les compilateurs IDL vers C++ et/ou Java permettent de générer les routines de gestion du conteneur d'état. Le résultat de ce processus de compilation réalise l'interface *PSerializable* qui est présentée algorithme 1.

Nous illustrons, dans [87], l'utilisation de la sérialisation portable sur la base d'un service de messagerie instantanée implémenté en Java et en C++. Dans un premier temps, nous montrons qu'il est possible de réaliser un observateur externe d'état, en Java par exemple. Cet observateur est capable d'obtenir l'état de tout objet CORBA qui implémente l'interface *PSerializable*. L'observateur peut alors avoir une vue hiérarchique de l'état de l'objet, avec pour chaque attribut son type et sa valeur, et pour les types composés, ses composites. Dans un deuxième temps, nous présentons la migration de l'état d'un serveur de messagerie instantanée à partir d'une instance en Java vers une autre en C++ et vice-versa. Enfin, sur la même base, nous montrons qu'il est alors très facile de rendre un service persistent.

En résumé, la sérialisation portable d'objets CORBA est une caractéristique intéressante dans différents contextes (boîtes à outils logicielles [64], systèmes auto-* [4], migration d'objets [16]) mais c'est avant tout dans le cadre de la réplication pour les mécanismes de tolérance aux fautes que nous l'avons utilisé, cf. la section 1.3 ou d'autres travaux [142, 13].

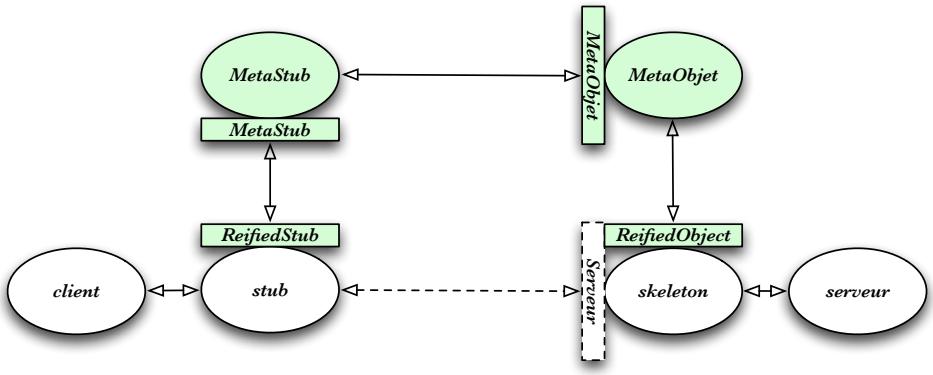


FIGURE 1.2 – Interactions entre objets et méta-objets CORBA

1.3 Un protocole à méta-objets pour la tolérance aux fautes

Thèse : [72] - Publication majeure : [126] - Autres publications : [76, 86, 139, 74, 75]

Dans mes travaux de thèse [72], nous avons conçu un protocole à méta-objets dédié à la tolérance aux fautes des applications distribuées en général, avec une implémentation basée sur l'intergiciel de communication CORBA en particulier. En nous basant sur l'expérience acquise lors des travaux précédents, nous étudions donc les mécanismes réflexifs nécessaires à la mise en œuvre de techniques de tolérances aux fautes par réplication (sauvegarde sur support stable, réplication passive, semi-active et active) et de confidentialité (par chiffrement des requêtes). Ces travaux sont simplement résumés dans cette section, le lecteur intéressé peut se référer à [126] et à [72] pour de plus amples détails.

Le protocole à méta-objet que nous avons défini régit les relations entre objets applicatifs et méta-objets responsables des caractéristiques non-fonctionelles. Objets applicatifs et méta-objets prennent place dans une architecture distribuée où tous sont des objets CORBA et interagissent donc à travers cet intergiciel. De cette façon, grâce à CORBA, ils bénéficient de la transparence du médium de communication et de la portabilité aux niveaux du langage, du système d'exploitation et du matériel. Sans entrer dans les détails de l'architecture CORBA, un client CORBA interagit avec un serveur à travers un *stub* et un *skeleton*. Le *stub* est l'entité représentant localement le serveur et est notamment responsable de la sérialisation des requêtes. Le *skeleton* est son pendant côté serveur et effectue en particulier la désérialisation des requêtes. Le cycle des requêtes émises par le client vers le serveur passe donc par cette chaîne : client-stub-ORB-skeleton-serveur et retour en sens inverse pour une éventuelle réponse. Notre protocole à méta-objet introduit des indirections dans cette chaîne pour y insérer les méta-objets. Comme illustré figure 1.2, celle-ci devient : client-stub-metastub-ORB-metaobjet-skeleton-serveur.

Le protocole à méta-objet est donc défini à travers les interfaces *ReifiedStub*, *ReifiedObject*, *MetaStub* et *MetaObject* respectivement présentées dans les algorithmes 2, 3, 4 et 5. C'est à travers ces interfaces que des mécanismes de tolérance aux fautes (tels que par réplication passive, semi-active, active avec vote, etc.) ont été mis en œuvre. Intuitivement, si l'on prend

l'exemple de la réplication passive, on peut aisément se convaincre qu'il est relativement aisé d'implémenter une sauvegarde périodique de l'état d'un serveur en utilisant la réification des méthodes entrantes, ainsi que la sérialisation (c'est-à-dire en implémentant la méthode *metaMethodCall* adéquate qui utilise *getState* pour la sauvegarde de l'état de l'objet).

Algorithme 2 Interface IDL des stubs

```
1:  interface ReifiedStub {
2:      // Gestion du comportement et des invocations
3:      void startUp ( in long constructorID, inout any arguments ) ;
4:      void methodCall ( in long methodID, inout any arguments ) ;
5:      void cleanUp () ;
6:      // Gestion des liens de metaniveau
7:      Metastub getMetastub() ;
8:      void setMetastub (in Metastub newMetastub) ;
9:      Object getReference() ;
10:     void setReference (in Object newObject) ;
11: }
```

Algorithme 3 Interface IDL des objets

```
1:  interface ReifiedObject {
2:      // Gestion de l'état
3:      any getState() ;
4:      void setState(any state) ;
5:      // Gestion du comportement et des invocations
6:      void baseStartUp ( in long constructorID, inout any arguments ) ;
7:      void baseMethodCall ( in long methodID, inout any arguments ) ;
8:      void baseCleanUp () ;
9:      // Gestion des liens de metaniveau
10:     Metaobject getMetaobject() ;
11:     void setMetaobject (in Metaobject newMetaobject) ;
12: }
```

La réalisation de ce protocole à méta-objet, que ce soit pour le langage Java ou pour le C++, s'appuie sur l'utilisation d'un compilateur ouvert afin de traduire le code applicatif de base en un code réflexif mettant en œuvre le protocole à méta-objet. Cette transformation de code se déroule en deux phases : une première phase de filtrage du code, et une seconde phase d'instrumentation à proprement parler. Le filtrage permet de s'assurer que le code de base est compatible avec de bonnes pratiques, que le modèle à objet est suffisamment propre. En ce qui concerne le langage C++, le filtrage vise essentiellement à assurer une bonne encapsulation des classes : protection des attributs et accès à ceux-ci via des accesseurs uniquement ;

Algorithme 4 Interface IDL des Metastubs

```
1: interface MetaStub {
2:     // Gestion du comportement et des invocations
3:     void metaStartUp ( in long constructorID, inout any arguments ) ;
4:     void metaMethodCall ( in long methodID, inout any arguments ) ;
5:     void metaCleanUp () ;
6:     // Gestion des liens stubs et metastubs
7:     ReifiedStub getStub() ;
8:     void getStub (in ReifiedStub newStub) ;
9: }
```

Algorithme 5 Interface IDL des Méta-objets

```
1: interface MetaObject {
2:     // Gestion du comportement et des invocations
3:     void metaStartUp ( in long constructorID, inout any arguments ) ;
4:     void metaMethodCall ( in long methodID, inout any arguments ) ;
5:     void metaCleanUp () ;
6:     // Gestion des liens objets et méta-objets
7:     ReifiedObject getObject() ;
8:     void setObject (in ReifiedObject newObject) ;
9: }
```

pas de classes et fonctions amies ; pas de variables globales ou de classes ; pas d'arithmétique des pointeurs ; pas de passage d'argument par pointeur mais seulement par référence. Ce filtrage permet de rapprocher le modèle à objet de C++ de celui de Java, où seule l'utilisation exclusive d'accesseurs a besoin d'être vérifiée. L'instrumentation des classes n'a qu'un objectif : celui de mettre en œuvre le protocole à méta-objets. Cela revient donc à réifier les créations et destructions d'objets, ainsi que les appels de méthodes. Pour ce faire, chaque fonction membre (c'est-à-dire les méthodes, constructeurs et destructeurs) est renommé. La méthode d'origine est remplacée par une méthode portant la même signature (nom de méthode, nombre, noms et types des arguments) qui empaquête les arguments dans une structure de donnée générique et appelle la méthode de réification adéquate (*metaStartup*, *metaMethodCall*, *metaCleanUp*). Des méthodes d'intercession sont crées : *baseStartup*, *baseCleanup* et *baseMethodcall*. Elles permettent de dépaqueter les paramètres et d'invoquer la méthode (ou le constructeur/destructeur) d'origine, puis de réempaqueter les éventuels résultats. Puis les méthodes de sérialisation *getState* et de désérialisation *setState* sont elles aussi implémentées, telles que décrites dans la section 1.2. Enfin des méthodes pour attacher et détacher un méta-objet aux objets sont crées.

Les différents objets CORBA correspondant à une application (*Clients*, *ReifiedStubs*, *MetaStubs*, *Metaobjects*, *ReifiedObjects*) prennent place au sein d'une architecture où nous avons défini différents services nécessaires à son fonctionnement : fabrique d'objets et fabrique de méta-objets. Le rôle de la fabrique d'objets est de permettre le lancement de nouvelles répliques d'objets applicatifs en fonction de la stratégie de réplication choisie. Celui de la fabrique de méta-objets est d'attacher le méta-objet adéquat à chaque objet applicatif. Il doit donc y avoir une fabrique d'objets et de méta-objets par machine (serveur) hôte du système distribué.

Les résultats expérimentaux conduits dans le cadre de ma thèse ont montré que le coût de la réflexivité était relativement faible. Ce coût est de l'ordre de la milliseconde pour ce qui est des invocations de méthodes et de l'ordre de la dizaine de milliseconde pour le lancement de l'application.

1.4 Réflexivité des systèmes multicouches

Thèse : [133] - Publication majeure : [136] - Autres publications : [134, 135, 137, 138, 46]

Dans les sections précédentes, nous avons montré que des mécanismes réflexifs, qu'ils soient adhoc ou sur étagère, permettaient de mettre en œuvre de façon élégante des stratégies de tolérance aux fautes au sein d'applications distribuées de type client-serveur. Nous avons pu également, au cours de ces travaux, observer les limites de ces approches. Certaines de ces limites proviennent du parti pris de se situer au niveau application/intergiciel de l'architecture logicielle. Par exemple, la capture de l'état au niveau applicatif ne permet que de traiter de la facette interne de l'état des objets, excluant notamment la sauvegarde d'état d'applications à brins d'exécution multiples [72].

Plus généralement, la nature des informations réfléxives disponibles dépendent fortement du "niveau" auquel on se situe :

- À haut niveau d'abstraction, les modèles de programmation ont une sémantique riche, associée à des primitives aux fonctions bien différenciées. Ils sont plus facilement programmables car plus proches du point de vue de l'utilisateur que de la réalisation concrète. Cette abstraction a un prix en terme de tolérance aux fautes : le niveau de granularité des notions manipulées à ce niveau est relativement grossier, tant du point de vue du contrôle que de l'observation. Certains aspects de bas niveau sont même complètement inaccessibles.
- Les modèles de programmation des couches les plus basses se situent, au contraire, à un niveau de granularité beaucoup plus fin. L'information disponible est donc bien plus importante, mais elle est également bien moins structurée et sa sémantique est, par conséquent, plus difficile à appréhender.

L'idée promue dans la thèse de François Taïani [133] est donc de combiner la puissance d'inférence sémantique des couches hautes avec la force de contrôle brute des niveaux les plus bas afin d'offrir une perception et un contrôle globaux du système. Prenons l'exemple de la réplication active d'un serveur multi-brins d'exécution. La réplication active nécessite un comportement déterministe des nœuds. C'est pourquoi le contrôle du non-déterminisme induit par les fils d'exécution multiple a été l'objet de tant de recherches dernièrement. Une approche satisfaisante consiste à s'assurer que les exclusions mutuelles (ou /textit mutexes pour /textit mutual exclusions) soient obtenues de façon identique sur toutes les répliques. Cela peut-être mis en œuvre de plusieurs façons : (i) en limitant le traitement séquentiel des requêtes [109], (ii) en utilisant un ordonnanceur déterministe [70], ou (iii) en imposant un consensus sur les décisions concernant la résolution des exclusions mutuelles [7, 108]. Ces solutions sont très difficiles, pour ne pas dire impossibles, à mettre en œuvre en se basant seulement sur des informations réflexives de haut-niveau car ni les changements de contexte, ni les sections critiques, ni les allocations de mutexes (sémaphores, verrous) ne sont visibles à ce niveau [74]. L'utilisation de primitives réflexives de bas niveau ne résoud pas le problème non plus. Parmi les approches mentionnées, la solution (iii) peut facilement être implémentée en utilisant un

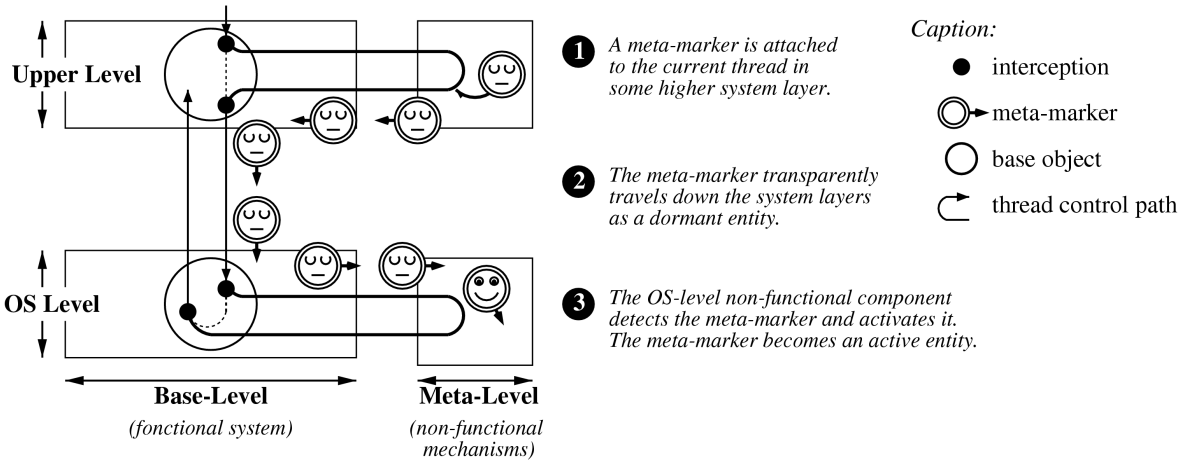


FIGURE 1.3 – Coordination multi-niveaux grâce aux meta-marqueurs

OS réflexif pour intercepter toute activité sur les mutexes et s'assurer que chaque réplique suit le même comportement. Toutefois, cette approche ne passe pas à l'échelle d'un système de taille "normale" car il y a bien trop d'activité sur les mutexes pour pouvoir tout synchroniser. À titre d'illustration, nous avons mesuré sur ORBACUS, une implémentation commerciale de la norme CORBA, jusqu'à 203 appels d'opérations sur les mutexes pour une simple requête. Ce problème de passage à l'échelle provient du manque de recul lorsque l'on se situe à ce niveau de détail : dans les faits, la plupart de ces prises de mutexes, dans l'intergiciel, n'ont pas d'effet sur le déterminisme observable au niveau applicatif. Par conséquent, Leur réplication n'est pas nécessaire. La difficulté est donc de discerner les prises de mutexes qui ont du sens pour les plus hauts niveaux, ce qui implique donc d'avoir une vision multi-niveaux de ces activités.

Nous avons proposé l'approche de la réflexivité multi-niveaux [134, 135] pour surmonter ces limitations. La réflexivité multi-niveaux est une approche réflexive qui combine les informations réflexives de niveau OS avec celles obtenues dans les niveaux supérieurs afin de bénéficier du meilleur des deux mondes : vue abstraite de haut-niveau et granularité fine. Dans l'exemple utilisé ci-dessus, la réflexivité multi-niveaux permet l'identification (et le contrôle) des mutexes qui sont vraiment importantes pour le déterminisme de l'application. La synchronisation des opérations de prises des mutexes sélectionnées uniquement permet de réduire drastiquement le nombre d'opérations à observer et à synchroniser sur les différentes répliques, ce qui permet alors de passer à l'échelle d'un système réel.

Dans [136], nous avons proposé une implémentation de la réflexivité multi-niveaux basée sur le principe des méta-marqueurs. Les méta-marqueurs s'inspirent de la biologie, et en particulier des phytohormones, pour convoyer de la méta-information à travers les différents niveaux d'abstraction d'un système logiciel. Ils peuvent être attachés et détachés d'un fil d'exécution afin de réaliser les mécanismes de réflexivité multi-niveaux. La figure 1.3 illustre comment les méta-marqueurs peuvent être utilisés pour coordonner les activités non-fonctionnelles à travers les niveaux multiples d'un système. Tout d'abord (1), dans une des couches hautes, un méta-marqueur est attaché au fil d'exécution courant. Ce méta-marqueur

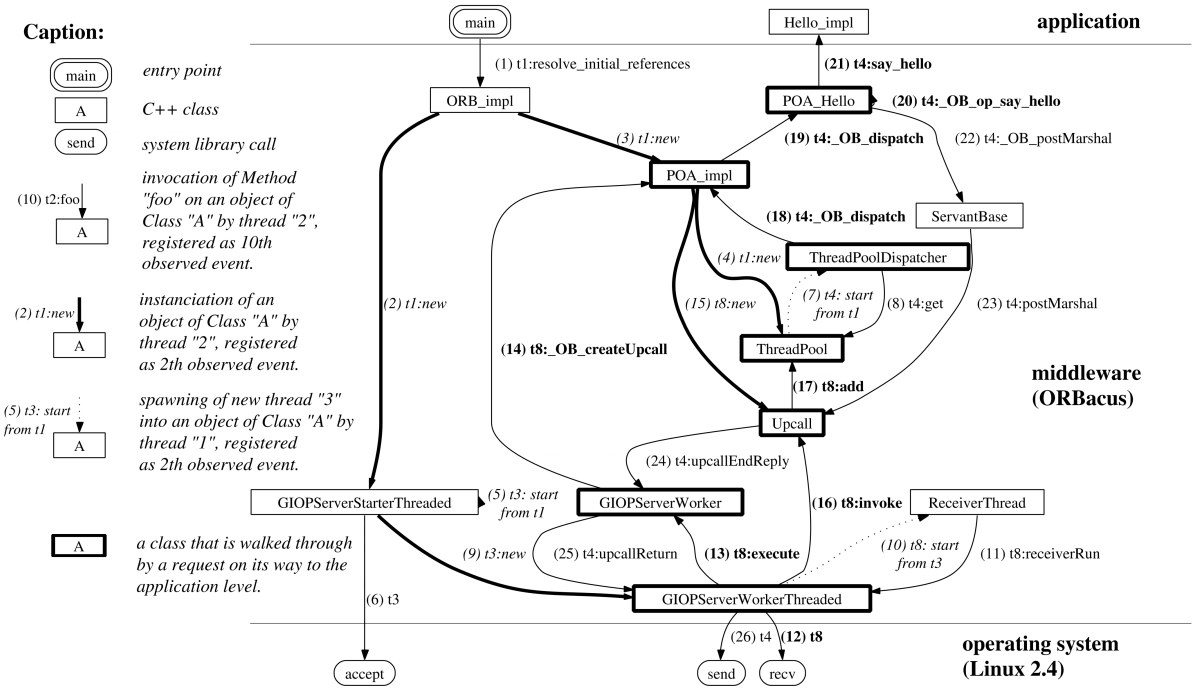


FIGURE 1.4 – Représentation de haut niveau du traitement d'une requête au niveau intergiciel

encapsule l'information réflexive concernant l'activité du système telle qu'observée à ce haut niveau d'abstraction. Alors que le fil d'exécution navigue à travers les niveaux logiciels du système, le méta-marqueur se propage (2). Il reste transparent pour les parties de code qui ne connaissent pas son existence, ce qui permet de garantir une forte séparation des problématiques entre parties fonctionnelles et non-fonctionnelles. Lorsque le fil d'exécution invoque une opération au niveau du système d'exploitation (OS), cette invocation est interceptée par la couche réflexive de l'OS. Les mécanismes réflexifs détectent la présence du méta-marqueur et l'activent (3). Une fois activé, et en fonction de l'information de haut-niveau qu'il transporte, le méta-marqueur peut mettre en œuvre les actions de niveau OS qui lui semblent nécessaires pour l'implémentation des mécanismes non-fonctionnels qu'il réalise. De cette manière, il crée un canal de coordination transparente entre les mécanismes non-fonctionnels de différents niveaux d'abstraction.

Toujours dans [136], nous avons utilisé le concept des méta-marqueurs pour aborder le problème de la réplication d'opération de synchronisation par mutexes, tel que décrit ci-dessus. Cette implémentation repose sur la synchronisation des opérations portant uniquement sur les mutexes ayant un impact sur le déterminisme perçu au niveau applicatif. Elle se base sur la rétro-conception de l'intergiciel afin d'identifier les relations entre *requêtes, sockets et mutexes*. Nous nous sommes appuyés sur l'outil CosmOpen [138], développé pour l'occasion, pour conduire cette rétro-conception sur 3 implémentations de la norme CORBA : ORBacus, TAO et OMNIORB. La figure 1.4 représente le modèle de haut-niveau obtenu après l'analyse d'ORBacus. Ce modèle représente le chemin parcouru par le traitement d'une requête CORBA. Grâce à ce modèle de haut-niveau, nous sommes en mesure d'identifier les points

de jonction exacts où introduire les méta-marqueurs. Grâce à cette approche, nous avons été capables d'insérer une couche de réflexivité multi-niveaux dans ORBAcus avec seulement 20 lignes de code supplémentaires (soit 0.002% du code original). Dans cette étude de cas, nous avons montré que seules 3 mutexes ont besoin d'être synchronisées pour assurer le déterminisme de la couche intergicielle. De plus, il y a eu en moyenne pour chaque requête seulement 3 opérations portant sur ces 3 mutexes, ce qui est à mettre en parallèle aux 203 opérations de mutex par requête dans un système qui répliquerait toutes les opérations.

1.5 Utilisation de mécanismes sur étagère

Thèses : [8, 99] - Publication majeure : [9] - Autres publications : [101, 100]

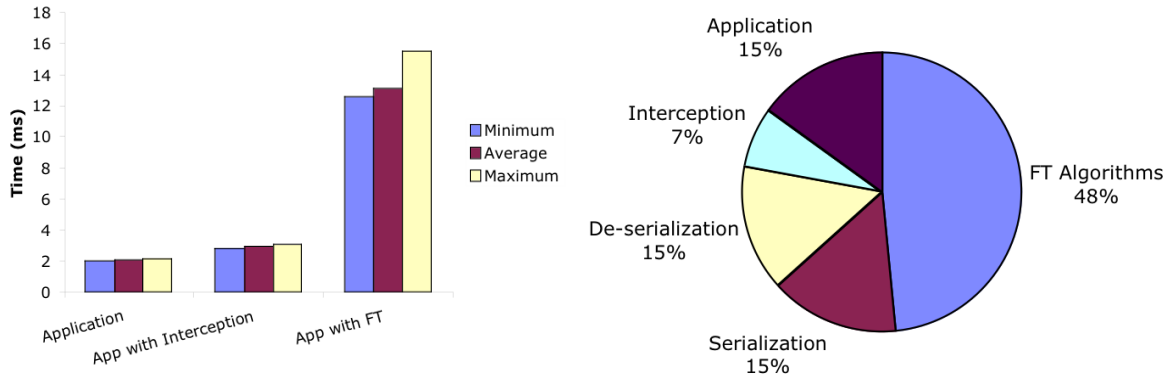
A l'opposée de l'approche présentée dans les sections précédentes, qui proposait la conception d'un protocole à méta-objet spécialisé pour la tolérance aux fautes des applications CORBA, et de mettre en œuvre ce protocole grâce à la transformation du code applicatif, nous nous sommes penchés, dans le cadre des travaux de thèse de Taha Bennani [8] et de Caroline Lu [99], sur l'utilisation de mécanismes réflexifs sur étagère. En effet, les évolutions conjointes de Java et des intergiciels, tant généralistes comme CORBA ou spécifiques comme AUTOSAR, ont permis d'envisager l'utilisation de mécanismes fournis directement par le support langage et par l'intergiciel pour l'implémentation de mécanismes de tolérance aux fautes.

1.5.1 Mécanismes sur étagère issus de CORBA

Dans [9], nous présentons un prototype dénommé DAISY pour *Dependable Adaptive Interceptors and Serialization-based sYstem*. Dans ce prototype, l'implémentation des mécanismes de tolérance aux fautes (en l'occurrence par réplication passive) repose donc, d'une part, sur la sérialisation des objets Java pour ce qui est de la capture de l'état des objets et, d'autre part, sur les *Portable Interceptors* en ce qui concerne la synchronisation des répliques. Il est hors de propos de présenter les détails d'implémentation dans ce manuscrit mais nous discutons ci-après des avantages et inconvénients de cette approche.

La sérialisation de la machine virtuelle Java donne accès à l'état interne des objets dans un format portable. Cette approche automatique évite au programmeur d'avoir à définir lui-même ces aspects ardus et sujets aux erreurs qui sont pourtant critiques pour la tolérance aux fautes. Dans les approches traditionnelles, le programmeur d'application aurait à définir lui même les méthodes pour la sauvegarde et la restauration de l'état de ses objets. Ce que l'on gagne en automatisation, on le perd bien entendu en adaptation aux besoins. En effet, l'état des objets comporte plusieurs facettes [87]. L'approche automatique ne peut traiter que de la facette interne, c'est-à-dire les attributs des objets, alors qu'une approche adaptée peut éventuellement s'appliquer aux facettes plus profondes comme les fichiers ouverts, ou les différents fils d'exécution.

Les *Portable Interceptors* (PIs) de CORBA permettent d'intercepter les requêtes envoyées au serveur par les clients et de les rediriger vers les mécanismes de tolérance aux fautes. Le bénéfice le plus notable est que les interactions entre clients et serveur peuvent être contrôlées de façon transparente, sans avoir recours à aucune modification du code applicatif. Cet aspect représente un atout quant à la simplicité de l'approche et à la maintenabilité du code. Les intercepteurs peuvent être insérés facilement sans interférer avec le développement de l'application. Ils peuvent être échangés dynamiquement, au chargement, selon la configuration désirée, pour choisir le mécanisme de tolérance aux fautes le plus adapté aux conditions



(a) Temps de réponse pour un client distant

(b) Répartition des coûts

FIGURE 1.5 – Coûts de la réflexivité et de la tolérance aux fautes dans DAISY

environnementales par exemple. Les intercepteurs sont des objets de l'intergiciel et, par conséquent, ils sont indépendants de l'application et du système opératoire. Ils sont par conséquent portables et réutilisables. Les performances sont acceptables et comparables à celles obtenues avec une approche adhoc, cf. fig. 1.5. En revanche, les intercepteurs présentent un certain nombre d'inconvénients :

1. Les PIs ne peuvent modifier les paramètres d'entrée ou de sortie des requêtes. Par conséquent, il se révèle impossible d'implémenter certains mécanismes non-fonctionnels tel que le chiffrement des requêtes par exemple. Il est toutefois possible d'insérer des données dans les entêtes des requêtes, ce qui peut être utilisé pour signer ces dernières, par exemple. Cela a également de sérieuses implications sur l'implémentation des mécanismes de tolérance aux fautes. Une réplique secondaire, qui reprendrait le traitement d'une requête après la défaillance du primaire par exemple, doit forcément recalculer la réponse une fois que le client lui a retransmis la requête. Elle ne peut pas se contenter de renvoyer la réponse qu'il aurait reçue du primaire avant sa défaillance.
2. Les PIs ne sont pas des objets CORBA. En conséquence, les PIs ne peuvent pas communiquer directement entre eux. Les mécanismes de tolérances aux fautes doivent donc tricher pour, par exemple, s'envoyer les copies des requêtes, les points de reprise, etc. La solution retenue implique donc d'imposer aux objets serveurs d'implémenter une interface de tolérance aux fautes, avec des méthodes vides, qui ne serviront que de moyen pour invoquer les méthodes correspondantes aux niveau des PIs. Cet inconvénient grève la transparence de la tolérance aux fautes au niveau de l'application, mais cela reste cantonné à la déclaration d'une interface aux méthodes vides. L'implémentation du client, quant à elle, n'est pas impactée.
3. Les PIs ne peuvent avoir leur propre fil d'exécution. Quelle que soit la stratégie d'ordonnancement des requêtes de l'intergiciel, le PI n'est activé que par l'arrivée d'une requête ; il ne peut avoir sa propre boucle et doit donc appliquer des stratégies réactives

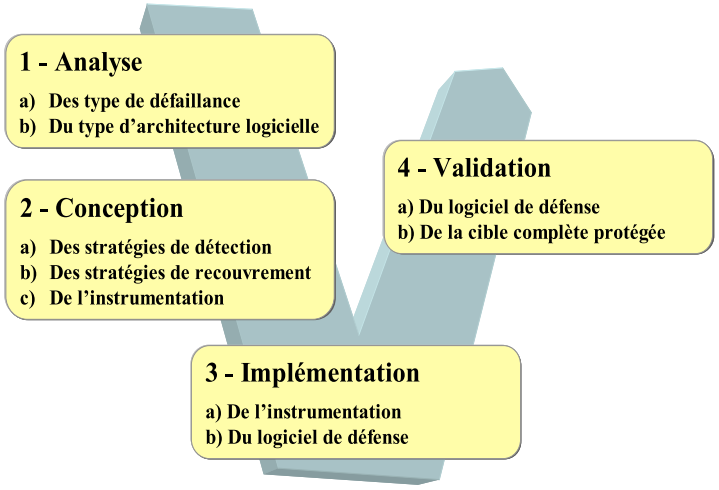


FIGURE 1.6 – Cycle de l'approche appliquée au domaine automobile

et non pas pro-actives. L'implémentation de la détection des fautes par messages de type *"I am alive"* est donc particulièrement difficile.

4. Les PIs doivent invoquer toutes les requêtes et ne peuvent les réordonner. La seule façon d'interrompre le flot d'exécution d'une requête est de lever une exception qui sera renvoyée au client. L'implémentation de mécanismes de réplication semi-active, où seule une réplique parmi n exécute la requête, est donc rendue plus difficile et moins transparente pour les clients. L'impossibilité de réordonner les requêtes a des implications très importantes en cas de défaillance du primaire alors que plusieurs clients ont des requêtes en cours. En effet, le PI secondaire qui reprend le rôle du primaire doit recevoir et suspendre l'exécution de toutes les requêtes en cours afin de décider s'il doit appliquer le dernier point de reprise reçu. Cela peut se révéler très compliqué selon les stratégies d'ordonnancement des requêtes de l'intergiciel. Une solution proposée dans [135] consiste à promouvoir les requêtes en tant qu'objets de premier ordre, cf. section 1.5.2.

Pour conclure, l'expérience menée dans ces travaux a consisté à s'astreindre à utiliser des mécanismes réflexifs sur étagères pour l'implémentation de stratégies de tolérance aux fautes. Si la simplicité de l'approche bénéficie essentiellement à la transparence pour le programmeur d'application, elle se paye en terme de capacité d'implémentation de stratégies évoluées de tolérance aux fautes. Dans la section suivante nous replongeons donc dans une approche adhoc, qui aborde non seulement la réflexivité de l'intergiciel, mais aussi la réflexivité des autres couches du système, à savoir l'applicatif et le système opératoire.

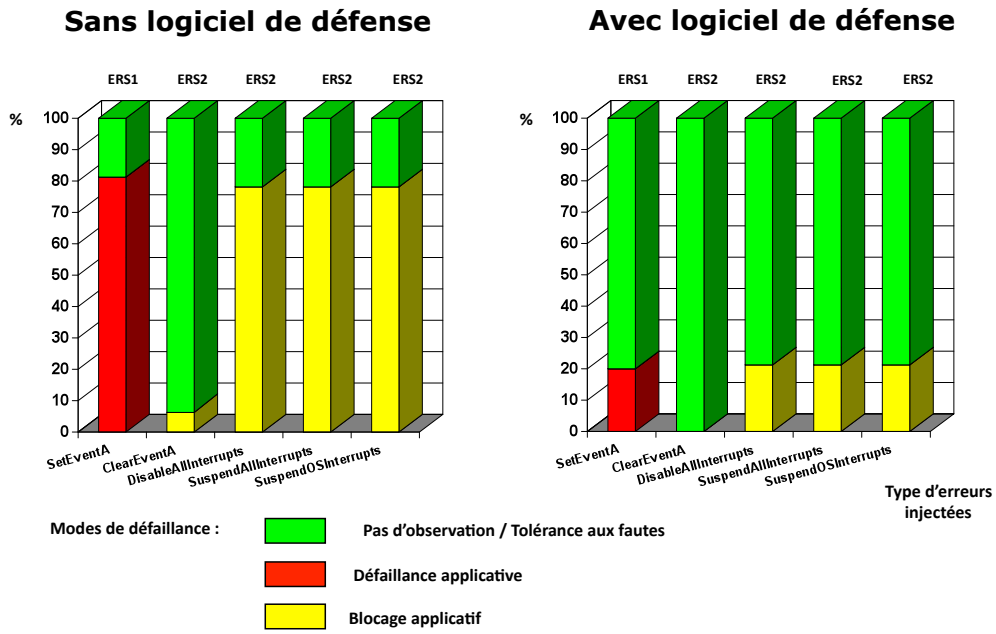


FIGURE 1.7 – Efficacité des mécanismes de tolérance aux fautes sur le module climatisation

1.5.2 Application à l'automobile

Dans le cadre de la thèse de Caroline Lu [99], en contrat CIFRE avec Renault, nous avons cherché à appliquer les concepts de la réflexivité multi-niveaux au domaine du logiciel embarqué automobile. Il s'est donc agit de caractériser le type d'architecture logicielle utilisée dans ce domaine, d'étudier les abstractions essentielles et d'identifier les modes de défaillance qui doivent être ciblés par des mécanismes de tolérance aux fautes afin de proposer des méta-modèles multi-niveaux adaptés à ce contexte, sur la base de l'expérience acquise dans le domaine et rapportée dans les sections précédentes. Il a fallu ensuite implémenter une étude de cas et procéder à son évaluation par injection de fautes. Dans cette thèse, nous proposons une méthodologie globale, qui va de la spécification (analyse de sûreté de fonctionnement) à la validation, cf. la figure 1.6. La méthode [101, 100] consiste à déterminer les fautes ciblées, les moyens de détection et de tolérance à mettre en œuvre. À partir de là, nous montrons comment instrumenter la plateforme système grâce à des mécanismes réflexifs multi-niveaux. Il est intéressant de noter que le standard AUTOSAR spécifie la notion de *hooks* qui permet d'implémenter facilement des mécanismes d'interception. Grâce à ces mécanismes, il est alors relativement aisé de réaliser l'instrumentation nécessaire à la détection et au recouvrement des fautes ciblées. La thèse propose ensuite une méthode de validation par injection de fautes classique. À titre illustratif, la figure 1.7 présente les résultats obtenus sur une application de climatisation rendue robuste par la méthode proposée dans la thèse.

1.6 Composants réflexifs pour l'adaptabilité

Thèse : [114] - Publication majeure : [45] - Autre publication : [115]

Certains systèmes critiques voient leur environnement et leur contexte d'exécution varier au cours de leur fonctionnement. Ces variations résultent des ressources disponibles et des modèles d'exécution, mais aussi des fautes auxquelles ils peuvent être soumis. Lorsque la survie du système est en jeu, ces variations doivent être prises en compte dans l'approche globale de la tolérance aux fautes. Les mécanismes de tolérance aux fautes doivent alors être adaptés à la situation, dans le sens où ils doivent répondre aux spécifications dans le contexte opérationnel courant. Cette adaptation nécessite la modification topologique, structurelle et algorithmique de la tolérance aux fautes, et ceci à l'exécution. Pour atteindre cet objectif, la tolérance aux fautes doit être conçue dès le départ, pour permettre son adaptation à l'exécution. Les technologies à base de composants favorisent la modularité des applications et donc la gestion de leur structure et de leur topologie. La réflexivité, quand à elle, permet de séparer la gestion algorithmique des autres problématiques. Ainsi, nous montrons comment fabriquer une stratégie de tolérance aux fautes avec un modèle à composant réflexif en vue de sa modification à l'exécution.

Notre démarche globale [114] s'appuie sur une approche réflexive récursive. Elle a impliqué en premier lieu la définition d'un méta-modèle pour la tolérance aux fautes à partir duquel nous proposons une décomposition de la tolérance aux fautes sous forme de services et de composants pour l'adaptation [115]. Les figures 1.8 et 1.9 montrent, par exemple, la décomposition en composants de mécanismes de réplication passive et semi-active.

Ensuite, nos travaux ont porté sur la détermination des états adaptables du système, c'est-à-dire comment décider si le système, ou un sous-ensemble de celui-ci est dans un état qui permet d'envisager une adaptation ? Si c'est le cas, comment s'assurer que le système reste dans cet état le temps de l'adaptation, tout en assurant la continuité du service ? Si ce n'est pas le cas, comment guider le système dans un tel état adaptable ? Ces travaux sont rapportés dans [45] où nous présentons une méthodologie pour (1) établir un modèle comportemental d'un système adaptable, (2) définir la notion d'état adaptable et les caractéristiques qui doivent y être associées (isolation, convergence, vivacité et conformité) et (3) recenser les états adaptables dans lesquels une adaptation de l'architecture à composants peut avoir lieu sans introduire d'incohérence.. Nous montrons également comment implémenter cette méthodologie sur le cas d'étude d'un pendule inversé. Le contrôleur du pendule est répliqué sur deux serveurs. Nous montrons qu'il est possible d'adapter le mécanisme de réplication (mécanismes passifs ou semi-actifs) de façon suffisamment efficace pour que le pendule reste en équilibre. Une adaptation, lors de ces expériences relatées dans [115], prenait de l'ordre de 100ms.

Ces travaux ont essentiellement montré la faisabilité de l'approche qui consiste à rendre adaptables les mécanismes de tolérances aux fautes grâce aux notions de réflexivité et de composants. Des mécanismes de tolérance aux fautes développés sous la forme de composants de

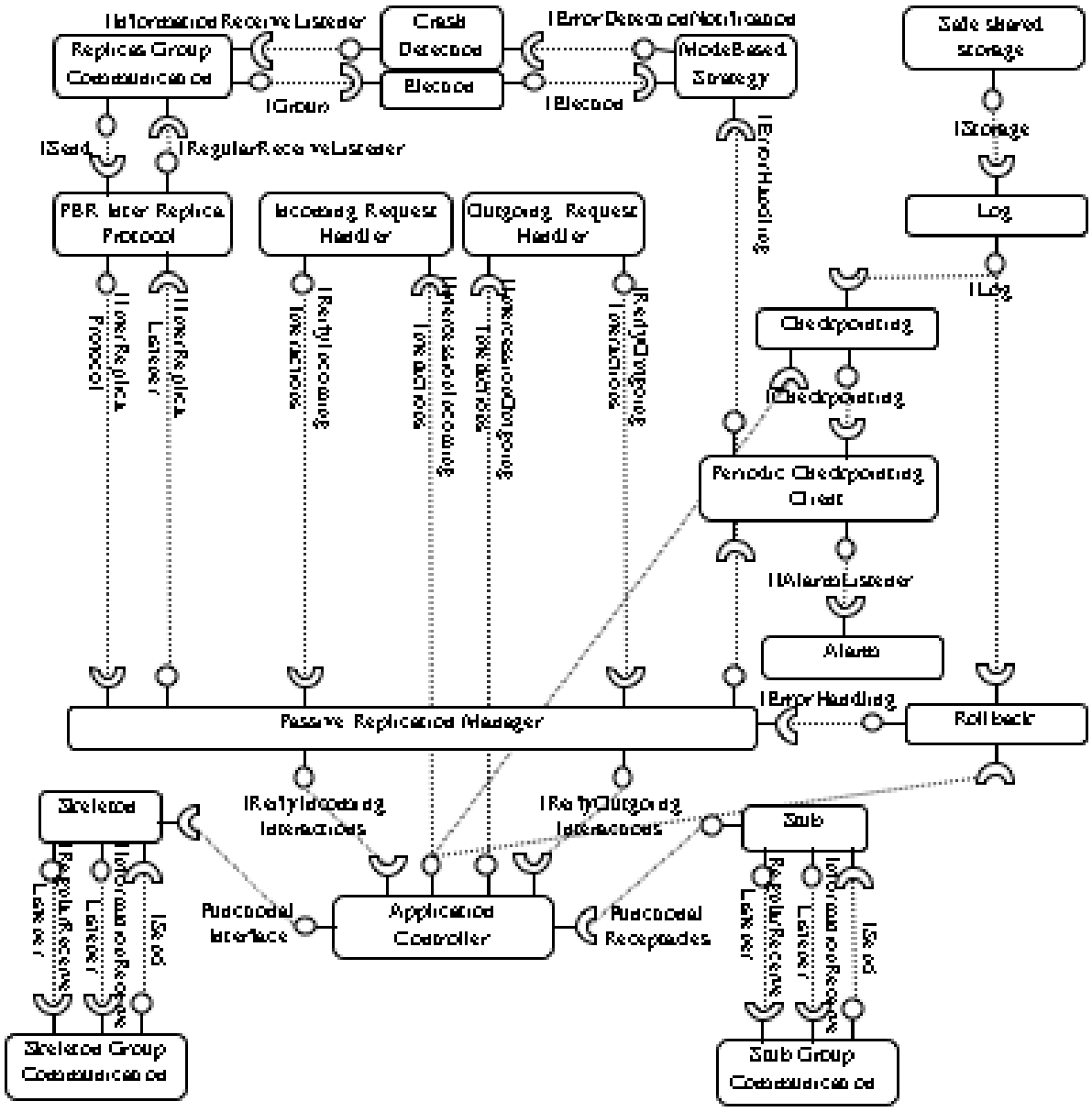


FIGURE 1.8 – Décomposition en composants d'un mécanisme de réplication passive

granularité fine peuvent être adaptés en ligne en fonction des conditions opérationnelles et de l'évolution de la configuration du système. Les limites de l'approche concernent essentiellement la complexité du modèle comportemental. Le passage à l'échelle implique une modélisation de plus haut niveau. Les expériences menées illustrent toutefois une forte corrélation entre le niveau de granularité du modèle et les performances de l'adaptation. Les arbitrages entre granularité et performance ne peuvent donc être faits qu'au cas par cas.

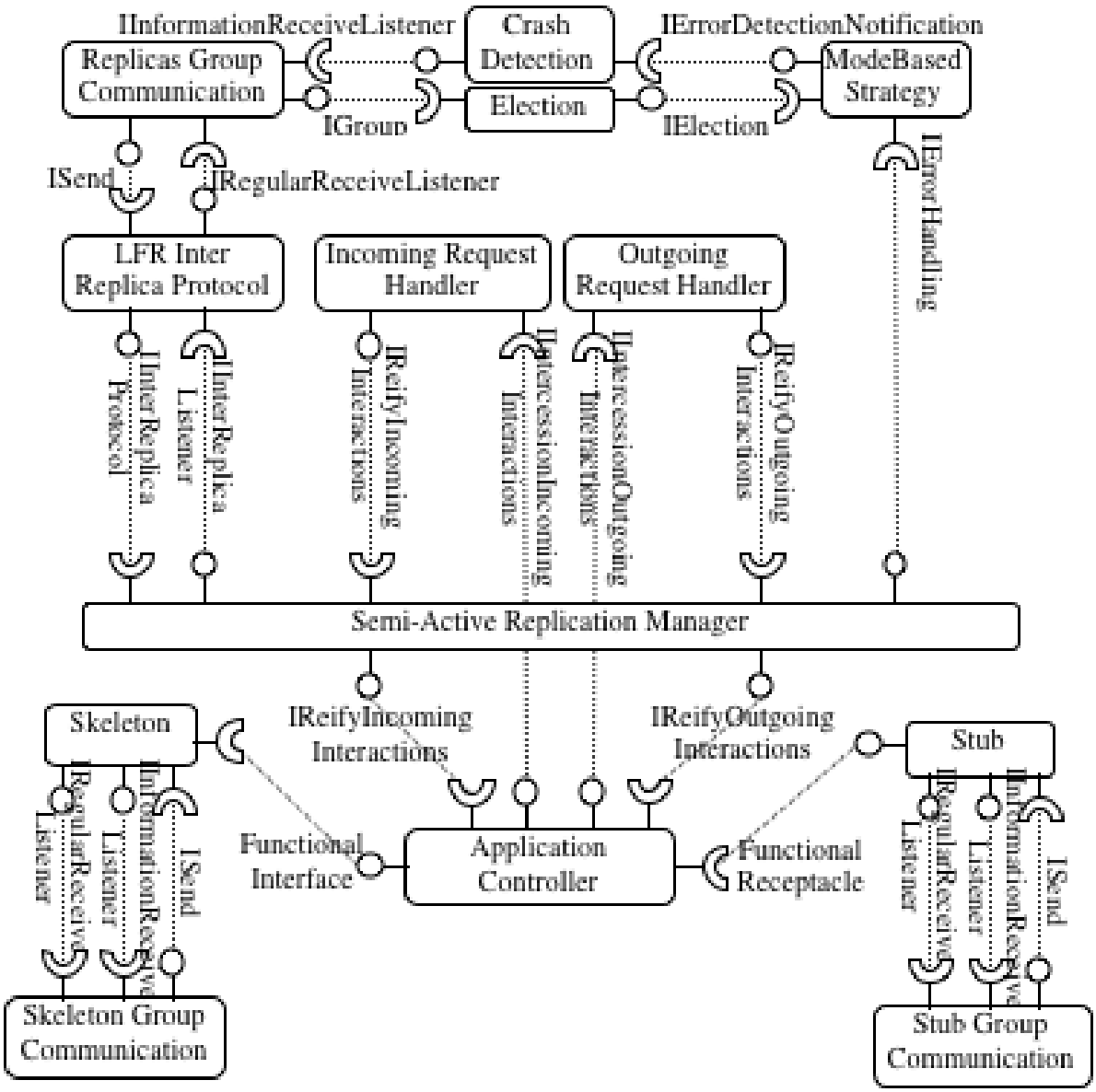


FIGURE 1.9 – Décomposition en composants d'un mécanisme de réplication semi-active

1.7 Conclusion

Ce chapitre a présenté mes premières activités de recherche. Celles-ci portaient essentiellement sur la question de la séparation des problématiques, entre le côté applicatif d'un système et ses facettes non-fonctionnelles (tolérance aux fautes, distribution, sauvegarde de l'état, adaptation, etc.). Ces recherches ont fait l'objet de travaux plutôt théoriques, comme la réflexivité multi-niveaux, et d'autres très pratiques, tels que leur application à une architecture logicielle automobile. Certaines des pistes ouvertes dans ces travaux n'ont pas été explorées, comme par exemple l'utilisation de la programmation orientée aspects, car j'ai estimé avoir suffisamment balayé le domaine et souhaitais me pencher sur d'autres sujets. Il est à noter que cette piste de la programmation orientée aspect est poursuivie par d'autres chercheurs de mon équipe de recherche. L'ensemble de ces travaux concernant la réflexivité s'est essen-

tiellement situé à un niveau intergiciel, entre l'application et le système d'exploitation. Cette recherche d'un niveau intergiciel adéquat, afin que les développeurs d'applications distribuées puissent facilement utiliser des mécanismes permettant de rendre leurs applications tolérantes aux fautes, se retrouve d'ailleurs dans le prochain chapitre qui aborde quant à lui la problématique des systèmes mobiles.

Chapitre 2

Résilience des systèmes mobiquitaires

Les travaux, relatés dans ce chapitre et le suivant, dans le domaine de la résilience des systèmes *mobiquitaires* (pour *mobiles et ubiquitaires*) sont ceux sur lesquels portent la plupart de mes efforts de recherche actuels. Ils constituent également les éléments principaux de ma prospective. Ces travaux peuvent être découpés en trois axes qui structurent ce chapitre et le suivant.

1. Architectures et algorithmes pour la résilience des systèmes mobiquitaires. Nos travaux, dans cet axe, portent d'une part sur la conception d'approches coopératives pour la disponibilité des données (cf. section 2.1.1), d'autre part sur la conception d'algorithmes de géo-communication (cf. section 2.1.2), et enfin, plus généralement, sur la conception d'architectures dédiées aux systèmes mobiquitaires coopératifs (cf. section 2.1.3) et robotiques (cf. section 2.1.4).
2. Évaluation analytique et expérimentale de la résilience des systèmes mobiquitaires. Dans cette section, nous abordons nos travaux concernant l'évaluation analytique de la disponibilité des données dans le cadre de la réplication coopérative de ces dernières (cf. section 2.2.1) ainsi que l'évaluation expérimentale des systèmes mobiquitaires (cf. section 2.2.2).
3. Protection et respect de la vie privée dans les systèmes mobiquitaires où nous traitons de nos travaux concernant la *geo-privacy* (cf. chapitre 3).

Ce chapitre se concentre donc sur les aspects algorithmiques, architecturaux et sur l'évaluation de la résilience des systèmes mobiquitaires. La table suivante présente les publications auxquelles ces travaux ont donné lieu.

Thèses	07	Ludovic Courtès	[33]
	07	Benjamin Lussier	[102]
	11	Hoang Nam Chu	[28]
Publications	01	POMC	[73]
	04	Middleware wkshop	[77]
		DRHE	[103]
	05	DRHE	[106]
	06	EDCC	[35]
	07	PRDC	[34]
		LADC	[36]
		ICAPS	[105]
		UbiComp wkshop	[79]
		DSN	[104]
	08	DEBS	[78]
		DEBS	[124]
		EDCC	[125]
		OPODIS	[122]
	09	PODC	[80]
		WADS	[84]
		EWDC	[29]
	10	NetMob	[85]
		WiMob	[82]
		Chapitre dans ADS	[81]
	11	AMIRE	[83]
	12	EDCC	[120]
		EDCC wkshop	[6]

2.1 Architectures et algorithmes pour la résilience des systèmes mobiquitaires

Nos travaux concernant l'algorithmique et l'architecture et l'intergiciel pour la résilience des systèmes mobiquitaires ont jusqu'alors porté sur quatre aspects différents : tout d'abord sur de nouveaux paradigmes de communication géo-centrée pour les systèmes mobiles, sur la réplication de données pour les systèmes mobiles, sur une architecture générale pour les systèmes mobiles coopératifs, et enfin sur la robustesse des architectures robotiques pour systèmes autonomes. Dans les sous-sections suivantes, ces différents axes sont développés.

2.1.1 Réplication coopérative de données

Thèse : [33] - Publication majeure : [36] - Autres publications : [77, 35, 34]

Les systèmes mobiles tels que les ordinateurs portables, assistants personnels (PDA) et téléphones portables sont de plus en plus utilisés et dans des contextes où ils risquent d'être abîmés, perdus ou volés. Ces outils sont souvent utilisés pour produire des données (contacts, photos, vidéos, etc.).

Malgré tout, relativement peu de mécanismes sont disponibles pour améliorer la disponibilité des données stockées sur ces appareils. En outre, les mécanismes disponibles tels que les mécanismes de « synchronisation » souffrent de limitations. Ils requièrent notamment une intervention manuelle de l'utilisateur, et leur utilisation est souvent contraignante (par exemple, l'utilisateur doit être dans l'environnement physique de son ordinateur de bureau). Cette situation laisse les utilisateurs de dispositifs mobiles avec, au mieux, des possibilités de sauvegarde de données par intermittence. Nous pensons que cette situation appelle de nouveaux mécanismes de sauvegarde tirant davantage parti des possibilités offertes par les dispositifs mobiles et leurs contextes d'utilisation. Ces dispositifs deviennent omniprésents et communicants. Avec l'arrivée de technologies de communication réseau ad hoc, les réseaux spontanés deviennent une réalité, permettant à des périphériques proches les uns des autres de communiquer entre eux sans aucune intervention humaine. Ces observations nous ont poussé à explorer les possibilités permettant de tirer parti des interactions spontanées dans l'objectif de créer un service de sauvegarde coopérative.

Dans cette section nous présentons brièvement l'approche algorithmique. Son évaluation analytique est présentée en section 2.2.1. Les algorithmes proposés ont été utilisés pour la mise en œuvre d'une boîte noire distribuée qui sert également d'exemple illustratif de l'évaluation expérimentale qui est traitée en section 2.2.2.

Objectifs. Le service que nous proposons a donc pour objectif d'améliorer la disponibilité des données stockées sur les dispositifs mobiles, en leur fournissant des moyens pour tolérer aussi bien les fautes permanentes (perte, vol, endommagement) que les fautes transitoires (effacement accidentel des données ou corruption). Pour tolérer les fautes permanentes, notre service doit être capable de stocker les données d'un dispositif sur d'autres dispositifs. Les

moyens de communication sans fil dorénavant ubiquistes fournissent le support à un tel service. Le service de sauvegarde envisagé est donc coopératif, décentralisé, et n'importe quel périphérique peut y participer. On attend d'un périphérique qui utilise le service qu'il y contribue en retour. Par la suite, nous utilisons le terme contributeur pour désigner un périphérique jouant le rôle de fournisseur d'espace de stockage ; on utilise le terme de propriétaire des données pour désigner un périphérique dans son rôle d'utilisateur ou de " client " du service. Évidemment, dans un souci d'équité, chaque dispositif doit jouer les deux rôles. Un mécanisme de sauvegarde coopérative pour dispositifs mobiles permet la sauvegarde de données même en l'absence d'accès à une infrastructure réseau.

Notre principal objectif est d'améliorer la sûreté de fonctionnement des dispositifs mobiles. Toutefois, la sûreté de fonctionnement d'un service coopératif ouvert tel que nous l'envisageons est elle aussi sujette à un certain nombre de menaces. Il s'agit donc de rendre le service lui-même également sûr de fonctionnement et en particulier vis-à-vis des menaces contre les attaques visant la confidentialité et le respect de la vie privée, contre l'intégrité et l'authenticité, et bien entendu contre celles visant la disponibilité des données et du service lui-même.

Principes de la Sauvegarde Coopérative. Nous supposons que les utilisateurs synchronisent leurs données avec leur ordinateur fixe, lorsqu'ils y ont accès. Par la suite, alors qu'ils se déplacent avec leurs dispositifs mobiles, ceux-ci devront spontanément se connecter aux dispositifs voisins participant au service dans le but d'échanger des données à sauvegarder, et ce de manière transparente à l'utilisateur. Cela est mis en œuvre à travers des protocoles de découverte de service. Une fois un contributeur découvert, un propriétaire lui envoie une requête de stockage de données. Un propriétaire peut sélectionner les contributeurs en fonction de la confiance qu'il leur accorde. En pratique, les requêtes sont composées de blocs de données de petite taille, plus adaptés au fait que les rencontres de contributeurs sont imprévisibles et potentiellement de courte durée. Les données sont chiffrées. Enfin, un contributeur peut lui-même choisir d'accepter ou non une requête en fonction des critères de son choix, tel que sa capacité de stockage disponible ou son niveau d'énergie. Dans la plupart des scénarios, il semble irréaliste de compter sur une rencontre entre le propriétaire et ses contributeurs pour la restauration de ses données. Par conséquent, nous supposons que les contributeurs transfèrent les données que leur ont fournies les propriétaires vers un support de stockage accessible sur Internet. Les propriétaires contactent ensuite ce support de stockage pour restaurer leurs données. Un tel support de stockage peut être mis en œuvre de différentes façons : un simple serveur FTP commun à tous les participants, un réseau pair-à-pair, la boîte aux lettres électronique du propriétaire, etc. Les contributeurs peuvent aussi faire suivre les données qu'ils ont collectées vers d'autres contributeurs. Cependant, cette approche n'apporte aucun avantage du point de vue de la disponibilité des données en l'absence d'informations supplémentaires sur les contributeurs (par exemple, il pourrait être profitable de transmettre les données à un contributeur qui a souvent accès à Internet, mais cette information n'est pas forcément connue des autres). De même, chaque contributeur peut aussi copier les données qu'il a reçues vers

d'autres contributeurs. Cette approche est attrayante, car elle augmente bien entendu la disponibilité des données. Cependant, elle pose plusieurs problèmes. D'abord, il est improbable qu'un propriétaire fasse confiance à ses contributeurs pour effectuer réellement cette copie, l'intérêt des contributeurs étant plutôt d'économiser leur énergie. Ensuite, sans davantage de coordination, cette approche pourrait rapidement conduire à une inondation du service.

Dans un scénario purement ad hoc, les participants peuvent restaurer leurs données à partir des contributeurs présents dans leur voisinage. Dans le cas général, ils récupèrent leurs données à partir d'un serveur accessible par Internet, comme nous l'avons mentionné. La plupart du temps, les propriétaires demandent la dernière sauvegarde disponible. Parfois, il peut aussi être utile de demander, par exemple, "la sauvegarde datant du 4 avril". Etant donné que chaque dispositif accède à Internet à son propre rythme, les sauvegardes peuvent mettre un certain temps à se propager jusqu'au serveur sur Internet. Ainsi, il n'est pas impossible que la sauvegarde disponible à un instant donné sur Internet ne soit pas la dernière effectuée. Parfois, il peut donc être possible pour un propriétaire de restaurer une version plus récente à condition d'attendre plus longtemps, ce qui illustre un compromis entre la fraîcheur et la disponibilité des données.

Conception et Mise en œuvre. Les objectifs que nous nous sommes fixés en termes de sûreté de fonctionnement (confidentialité, respect de la vie privée, intégrité, authenticité, et disponibilité) ont des implications sur les techniques de stockage à mettre en œuvre. Le service de sauvegarde coopérative que nous présentons dans les paragraphes qui suivent répond à ces exigences.

Efficacité du stockage. Afin de tirer le meilleur parti des rencontres avec les contributeurs, il convient d'exploiter le mieux possible l'espace de stockage qu'ils mettent à disposition. Également, afin de maîtriser les coûts énergétiques, il est préférable de réduire la quantité de données à transférer. Dans les services de stockage coopératifs à grande échelle, on économise souvent l'espace de stockage en ne stockant qu'une seule fois chaque donnée élémentaire. Cette propriété est connue sous le nom de stockage à instance unique [12]. Il a été montré que cette approche réduit sensiblement la quantité de données à stocker dans le cadre de nombreux systèmes de stockage de données (systèmes d'archivage, de partage de fichiers pair-à-pair, de la sauvegarde pair-à-pair, des systèmes de fichiers réseau et des outils de synchronisation des données à distance). Nous proposons une implémentation coopérative du stockage à instance unique au moyen de fonctions de hachage cryptographiques.

Blocs de données de petite taille. Les rencontres avec des contributeurs étant imprévisibles et potentiellement de courte durée, les transferts de données doivent se limiter à des petites tailles afin de maximiser leurs chances de réussite. Après avoir évalué différentes techniques de découpage de flux en blocs, nous proposons une implémentation basée sur le découpage en blocs de taille fixe, qui est l'approche la plus simple. Combinée au stockage à instance unique, elle offre un bon compromis coût/performance en terme de volume de stockage.

Structures des méta-données efficaces. Une fois les blocs créés, il s'agit de créer des méta-données indiquant comment reconstituer la donnée d'origine. Pour ce faire, il doit d'abord

être possible de nommer les blocs. Le schéma de désignation des blocs utilisé doit éviter les collisions entre noms de blocs (un nom de bloc ne doit pas être ré-attribué, même après une perte des données), et il doit être de préférence indépendant du contributeur stockant le bloc, ce qui permet de pré-calculer les noms de bloc. Après nos expérimentations, il s'est avéré que des condensés SHA-1 de 20 octets étaient un excellent compromis, même avec des blocs de taille assez faible.

Atomicité de la sauvegarde. Il est pratiquement impossible de stocker, par exemple, un fichier ou un ensemble de fichiers sur un seul contributeur, ce qui mène à une fragmentation et dissémination des données. Cependant, il est important que la sauvegarde reste dans un état cohérent (au sens des propriétés ACID des bases de données) : la sauvegarde doit être soit réalisée et recouvrable, soit non-recouvrable. Une manière simple de garantir l'atomicité des sauvegardes est de suivre une approche ajout-seulement des données : les données sont toujours ajoutées à l'espace de stockage et jamais modifiées sur place. Par conséquent, l'insertion de contenu dans l'espace de stockage est atomique. En outre, il a été montré que le coût de cette approche en termes de stockage est faible dans beaucoup de scénarios, à condition qu'une forme de stockage à instance unique soit utilisée.

Détection d'erreurs. Les modifications des données aussi bien accidentelles que malveillantes doivent pouvoir être détectées. Des codes détecteurs d'erreurs doivent être calculés, soit au niveau du flux d'entrée, soit au niveau des blocs individuels. Comme à la fois des modifications accidentelles et malveillantes doivent pouvoir être détectées, il nous faudra utiliser des fonctions de hachage cryptographiques. Outre l'intégrité des données, leur authenticité doit aussi pouvoir être vérifiée. A cette fin, une partie des méta-données peut être signée avec une signature électronique. Les fonctions de hachage cryptographiques ont souvent été proposées comme moyen de désignation de blocs de données : il s'agit de désigner les blocs par le résultat d'une fonction de hachage cryptographique appliquée à leur contenu. En outre, deux blocs dont le contenu est identique auront le même nom, ce qui procure un moyen simple et efficace pour mettre en œuvre le stockage à instance unique.

Chiffrement. La couche de stockage doit avoir recours au chiffrement des données pour garantir leur confidentialité. Le chiffrement des données peut être appliqué soit au flux d'entrée, soit aux blocs. Toutefois, si un chiffrement asymétrique est utilisé, cette approche empêche de mettre en œuvre du stockage à instance unique entre les blocs provenant de différents propriétaires. Le chiffrement convergent permet de résoudre ce problème [37], en utilisant un algorithme de chiffrement symétrique avec pour clef le condensé de la donnée en clair (condensé fourni par une fonction de hachage cryptographique).

Redondance. Bien entendu, les sauvegardes devront être suffisamment redondantes, d'autant plus qu'on ne fait pas nécessairement confiance aux contributeurs. Nous abordons dans la section 2.2.1 les différentes approches (redondance simple, codes d'effacement) permettant de stocker les blocs de données de manière redondantes.

2.1.2 Communication de proximité

Publication majeure : [81] - Autres publications : [73, 78, 124, 122]

Classiquement, dans le domaine de l'algorithmique distribuée, la mobilité est un facteur de risque, de difficultés, de déconnexions, etc. Il aura fallu attendre l'émergence des réseaux tolérant les retards (DTNs - Delay Tolerant Networks) pour que la mobilité puisse être prise en compte comme un aspect positif [47]. En l'occurrence, dans les DTNs, certains nœuds, appelés mules, peuvent littéralement transporter des paquets de données lors de leurs déplacements. L'exemple typique de la mule est celui d'une navette inter-sidérale qui peut emmener des paquets réseaux d'une galaxie à une autre, à la manière d'un postier. Les abstractions de l'informatique distribuée classique sont extrêmement difficile à appréhender dans un système avec des nœuds mobiles qui communiquent par un réseau adhoc. En effet, déjà lorsque le réseau de communication est simplement asynchrone et que les nœuds peuvent défaillir, une multitude d'abstractions sont impossibles à implémenter sans hypothèses supplémentaires. A fortiori lorsque les nœuds bougent, se déconnectent, que les messages peuvent être perdus, les choses sont encore plus difficiles. Dans ces travaux, nous proposons de prendre les choses à l'envers et de concevoir des abstractions logiques spécifiques et prenant en compte la localisation et de la mobilité des nœuds. De telles abstractions sont certes moins puissantes qu'une mémoire partagée ou qu'un algorithme de consensus, mais elles sont applicables et facilement réalisables dans un réseau constitué de nœuds mobiles à communication adhoc.

Groupes de Proximité. Dès 2001, dans [73], nous avons commencé à nous pencher sur la notion de communication de groupe dans les réseaux adhoc. Nous y proposions l'utilisation de la connaissance de la localisation pour contourner les difficultés rencontrées classiquement dans ce type de réseaux : déconnexions, routes invalides, etc. De plus, nous pensons que la localisation étant essentielle à certain domaines applicatifs de l'informatique ubiquitaire, il est important de proposer des primitives de communication adaptées à cette notion. La localisation est donc à la base de la définition de notre modèle de communication de groupe pour dispositifs mobiles qui repose sur la notion de *groupes de proximité*.

Un groupe de proximité G est défini par un nom, une forme, un point de référence et un nombril. Le nom correspond à la notion d'intérêt commun dans les groupes de communication classiques. La forme et le point de référence, relatif à cette forme, définissent la zone géométrique ou géographique sur laquelle porte le groupe de proximité. On distingue deux type de groupe de proximité : les groupes relatifs à un nœud, pour lesquels le nombril est ce nœud, et où le point de référence de la forme suit les déplacements de ce nombril, et les groupes de proximité absolus, pour lesquels le nombril est un point dans l'espace géographique auquel le point de référence est lié. Dans le premier cas, la zone d'intérêt "suit" les déplacements du nœud nombril ; dans le second, la zone est immobile.

Dans l'objectif d'aider à l'implémentation de tels groupes de proximité, nous avons défini un outil d'estimation de la couverture par les nœuds du système de la zone d'un groupe de proximité. Cet outil est basé sur une connaissance de la position des nœuds voisins qui est

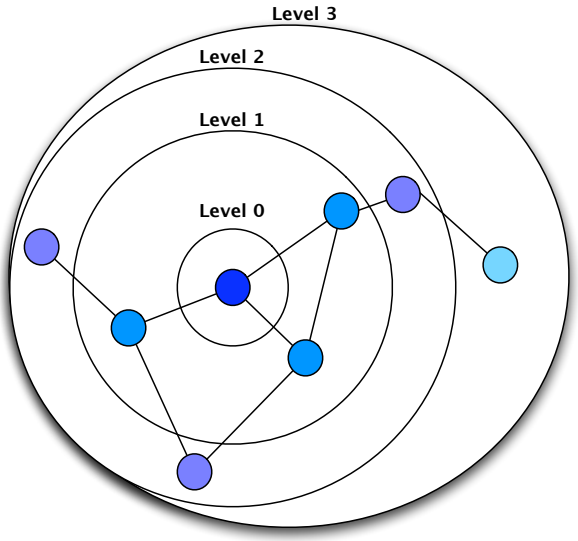


FIGURE 2.1 – Un graphe de carte de proximité

fournie par un service que nous avons dénommé *Carte de proximité* et qui fait l'objet de la sous-section suivante.

Carte de proximité. Dans la plupart des applications dédiées aux nœuds mobiles, et en particulier dans les applications basées sur la coopération des nœuds, chaque nœud doit interagir avec ses voisins. De plus, la qualité de service rendu par un composant du système peut dépendre de sa proximité : la quantité de nœuds dans le voisinage, de leurs positions respectives, de leur densité, etc. C'est le cas notamment des groupes de proximité. Il est donc nécessaire de formaliser la vue que peut avoir un nœud de son voisinage dans une représentation abstraite et compacte. C'est l'objet de la notion de *Carte de proximité* qui agrège à la fois l'information issue du positionnement et du réseau [73, 81]. Grâce à sa carte de proximité, un nœud voit les nœuds du voisinage (qui sont à un certain nombre H maximum de sauts), leur position et la fraîcheur de cette information.

Intuitivement, chaque nœud envoie périodiquement sa carte de proximité à ses voisins à un saut, et collecte cette même information de ses voisins directs. Lorsqu'il analyse l'information collectée, il est capable de mettre à jour sa carte avec les nœuds qui sont apparus comme voisins de ses voisins, les nœuds qui ont bougé, les nœuds dont la connectivité a changé, etc. La carte de proximité est représentée par un graphe tel que celui présenté figure 2.1. Les détails d'implémentation peuvent être trouvés dans [81].

La précision de la carte peut être évaluée face à deux critères, le premier temporel et l'autre relatif aux défaillances. Les aspect temporels sont relativement simples. Comme illustré par la figure 2.1, le niveau 0 de la carte représente la connaissance du nœud lui-même, le niveau 1 celle de ses voisins directs, le niveau n celle de ses voisins à n sauts. Si les cartes de proximité de chaque nœud sont envoyées périodiquement toutes les D unités de temps, alors l'information de niveau L est vieille de $L.D$ unités de temps. L'information concernant les nœuds lointains est donc logiquement moins précise que celle relative aux nœuds proches, à

la manière d'un œil de poisson qui voit flou à la périphérie. Le niveau maximal des cartes de proximité est donc limité à H sauts. En l'absence de fautes, à n'importe quel instant après $H.D$ unités de temps, un nœud connaît la position et la connectivité de ses voisins jusqu'à H sauts.

En ce qui concerne les fautes et défaillances, il est intéressant de voir que le protocole d'établissement des cartes de proximité est intrinsèquement tolérant aux fautes, qu'elles soient intermittentes ou par crash. Les fautes intermittentes, par exemple un GPS qui donne une position erronée ou une "mauvaise" carte de proximité, sont tolérées grâce à la forte redondance due à l'envoi périodique des cartes, dès qu'une information plus fraîche vient remplacer l'information erronée. Les défaillances de nœuds peuvent être détectées : lorsqu'un nœud n'émet plus de carte, il est effacé des cartes de ses voisins au bout d'un délai de grâce noté *failureDeltaTime*.

Les applications potentielles de la carte de proximité sont nombreuses et vont du service aux applications géolocalisées aux cartes de routage et de diffusion géo-localisée, en passant par les détecteurs de défaillances. La sous-section suivante traite de l'abstraction de *géo-registre* qui peut être implémentée sur la base des cartes de proximité.

Géo-registres. Dans un article fondateur [94], Leslie Lamport a défini différentes sémantiques de registres distribués : *safe*, *regular* ou *atomic*. Ces définitions ne tiennent pas compte des fautes qui peuvent toucher les nœuds, et ne sont pas directement utilisables dans un contexte mobile/adhoc. Nous proposons donc les géo-registres [125, 124, 122], accessibles uniquement depuis une zone géographique pré-déterminée et qui ont la particularité d'être ré-initialisés lorsqu'aucun nœud n'est présent au sein de la zone pour maintenir le registre. Cette sémantique particulière peut paraître faible mais nous montrons toutefois qu'elle est implémentable, peu coûteuse, et qu'elle peut être utile à certaines applications où la notion de densité de population dans la zone considérée a un sens. Par exemple, si le registre est utilisé pour stocker la densité des automobiles présentes dans une zone donnée, le registre peut être ré-initialisé à la valeur zéro s'il n'y a aucune voiture ; la valeur zéro aura du sens.

Le modèle théorique qui nous sert de cadre dans ces travaux est le suivant : nous considérons un système composé d'un nombre infini de nœuds qui évoluent dans l'espace géographique. Dans un premier temps, les nœuds ne crashent pas, sont anonymes (c'est-à-dire, dans ce contexte, qu'ils ne possèdent pas d'identifiant unique), et exécutent tous le même algorithme. Les nœuds sont capables de communiquer sans fil et de se localiser à tout moment avec une précision infinie. Ils peuvent se déplacer dans l'espace avec une vitesse maximale donnée.

Dans ce cadre, nous définissons une primitive de *geo-diffusion sûre* synchrone. Cette primitive permet de garantir que tous les nœuds corrects situés dans une zone géographique vont recevoir, dans un délai δ, tous les messages diffusés dans la zone. Nous définissons également la notion de *région centrale* dans laquelle tous les nœuds qui étaient dans la région centrale au moment de la diffusion d'un message sont assurés de le délivrer. Cette notion permet de s'abstraire un tant soit peu des mouvements des nœuds lors des diffusions de messages.

Sur la base de ces notions, nous définissons dans [122] la notion de geo-registre *régulier* (cf. [94]) sans écriture concurrente, et en proposons une implémentation. De façon intuitive, un geo-registre peut être vu comme une séquence de registres traditionnels dont l'utilisation est réduite à une zone géographique. En effet, le registre est remis à sa valeur initiale dès qu'aucun nœud n'est présent dans la zone pour le maintenir. Ces travaux se poursuivent encore actuellement dans différentes directions : avec des sémantiques de registre plus riches (multi-écrivains par exemple [123]), avec une implémentation dans un modèle de système localement synchrone, etc.

Nos travaux actuels dans le domaines des abstractions geo-inspirées se déroulent dans le cadre du projet AMORES [6] abordé dans le chapitre suivant. On notera toutefois que les abstractions sur lesquelles nous travaillons sont :

- les locanymes [50], qui permettent d'adresser de façon anonyme un nœud, ou une ressource, en fonction de sa localisation.
- les preuves de localisation, qui attestent de la localisation passée ou présente d'un nœud.
- la géo-cryptographie (ou *Position-based crypto*), qui associe des droits ou des clés à des localisations ou à des colocalisations.
- des primitives de stockage géo-localisé, généralisation des géo-registres décrits ci-dessus, comme par exemple un géo-buffer partagé.
- etc.

2.1.3 Éléments architecturaux

Publication majeure : [81] - Autres publication : [84]

Notre expérience concernant la résilience dans les systèmes mobiquitaires, relatée dans ce chapitre, nous a permis d'envisager la conception d'une architecture logicielle dédiée à la réalisation d'applications coopératives pour ce type de systèmes. En effet, les approches de type pair-à-pair semblent particulièrement bien adaptées pour gérer la dynamique de ces systèmes, tant en termes d'échelle, que de mobilité, déconnexions, défaillances, etc.

Dans cette section, tirée de [84, 81], nous présentons donc une architecture de niveau intergiciel pour la mise en place de services coopératifs pour les nœuds mobiles en général, et d'un service de sauvegarde coopérative comme celui présenté dans la section 2.1.1. Cette architecture est appliquée à la mise en œuvre d'une *boîte noire distribuée* qui implémente un dispositif virtuel, similaire aux boîtes noires de l'avionique, qui permet d'enregistrer l'historique d'un véhicule automobile de telle sorte que cet historique puisse être rejoué en cas d'accident.

L'architecture mise en place pour cette mise en œuvre est illustrée figure 2.2. Les aspects expérimentaux sont décrits en section 2.2.2. Son niveau intergiciel, dédié aux systèmes mobiles coopératifs, est basé sur quatre services intergiciels qui remplissent chacun un besoin de base :

1. Pour les besoins de **communication** : un service *réseau* qui satisfasse à la fois les besoins en termes de diffusion (*broadcast*) et de communication point-à-point (*unicast*).

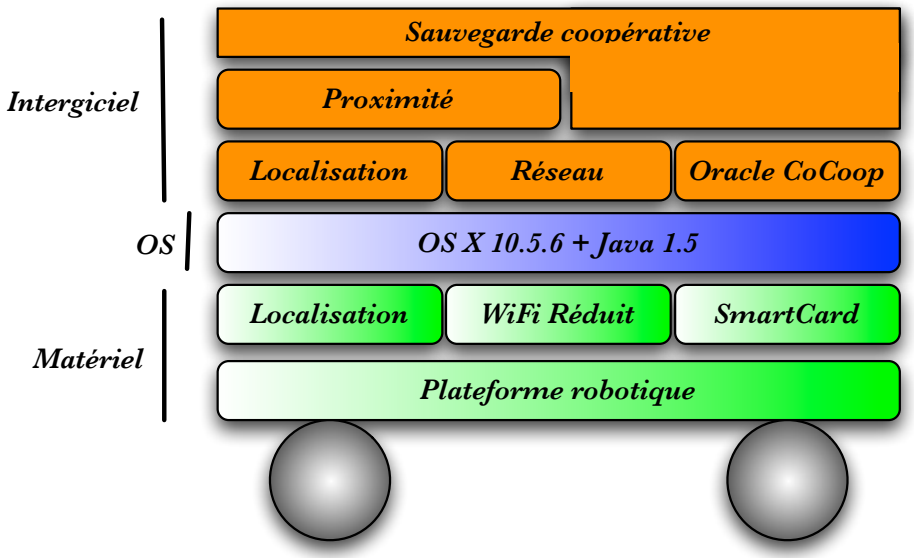


FIGURE 2.2 – Notre architecture pour les systèmes mobiquitaires coopératifs

Ces deux méthodes de communication sont nécessaires pour les raisons suivantes : dans un système coopératif, les nœuds doivent interagir de façon opportuniste avec d'autres nœuds qu'ils ne connaissent pas a priori. Ils doivent alors pouvoir diffuser des offres ou des requêtes de service pour explorer leur voisinage. Une fois qu'ils ont acquis suffisamment de connaissance sur ce voisinage, ils peuvent utiliser des communications point-à-point pour accéder aux ressources ou aux services nécessaires. C'est de notre point de vue la seule façon d'implémenter des services coopératifs sur un réseau mobile adhoc en tolérant la mobilité des nœuds et des ressources, ainsi que leurs défaillances.

2. Pour les besoins de **localisation et découverte de services** : un service de *carte de proximité* qui permette de connaître son voisinage, sa connectivité et sa localisation décrit en section 2.1.2. Ce service est lui-même basé sur un service de *localisation* qui permet de s'abstraire du matériel utilisé *in fine* pour la localisation : GPS, caméras infrarouges, ultra-sons ou UWB. La carte de proximité représente la connaissance locale qu'un nœud possède de son voisinage. Cette connaissance dépend essentiellement de deux paramètres : la largeur de la carte (c'est-à-dire le nombre de sauts maximal), et sa fraîcheur (soit sa fréquence de diffusion).

3. Pour les besoins d'**établissement de confiance** : un *oracle pour la confiance et la coopération* dont le rôle est d'évaluer le niveau de confiance et de coopération des pairs avec qui l'on veut pouvoir interagir. En effet, un service coopératif émerge de la coopération d'entités qui ne se connaissent en général pas. Par conséquent, ces entités n'ont a priori pas de relation de confiance pré-établie et peuvent être réticentes à coopérer. L'objectif de l'oracle pour la confiance et la coopération est d'évaluer localement le niveau de confiance entre entités voisines, et de gérer les éventuels mécanismes d'incitations à la coopération.

4. Pour les besoins de **résilience des données** : un service de *sauvegarde coopérative de données* décrit en section 2.1.1. Le problème de la sauvegarde coopérative peut-être réalisé en trois phases :
   (a) Découvrir les ressources de stockage disponibles dans le voisinage. Cette phase est réalisée grâce aux cartes de proximité.
   (b) Négocier un contrat avec les nœuds environnants pour l'utilisation de leurs ressources. Cela est réalisé par l'oracle pour la confiance et la coopération.
   (c) Transformer le flux de données à sauvegarder en blocs et assigner ces blocs aux ressources de stockage négociées. Cette transformation doit mettre en œuvre des techniques d'encodage de données capables de garantir les propriétés de confidentialité, respect de la vie privée, et disponibilité demandées. Ce service est également responsable de la restauration des données.

Sur la base de cette architecture, nous avons réalisé application de boîte noire distribuée. En quelques mots, cette application génère un flot de donnée sur chaque véhicule automobile. Ce flot consiste en la capture périodique de sa carte de proximité. Le service de sauvegarde coopérative est utilisé afin de répliquer ce flux de données sur les véhicules environnants, ou sur un réseau d'infrastructure lorsque la connectivité le permet. Le flux d'un véhicule (accidenté ou non) participant peut être restauré à partir des blocs éparpillés sur les véhicules ou sur l'infrastructure.

Plus exactement, l'application maintient la vue "boîte noire" de l'environnement du véhicule. Cette vue consiste en : la position du véhicule, sa carte de proximité élaguée à 10 sauts, une estampille temporelle. Cette vue est sérialisée et soumise au service de sauvegarde avec un identifiant (unique) qui consiste en la concaténation de l'identifiant du véhicule, en l ?occurrence son numéro d'immatriculation, et du temps courant. Lorsqu'un véhicule est accidenté, il faut alors soumettre le numéro d'immatriculation et intervalle temporel pour lequel on souhaite récupérer les données de la boîte noire. L'application va alors essayer de récupérer toutes les vues comprises dans cet intervalle. Lorsque les blocs de données correspondant nous parviennent, on peut rejouer le scénario de l'accident et l'analyser pour essayer de comprendre ce qui s'est passé. Une vidéo illustrant ce scénario est disponible à l'adresse suivante : http ://videotheque.cnrs.fr/doc=2712 ?langue=EN.

2.1.4 Robustesse des architectures robotiques

Thèses : [102, 28] - Publication majeure : [104] - Autres publications : [103, 106, 105, 29]

2.1.4.1 Introduction à la robustesse en robotique

Le concept de système autonome est une extension de la notion de robot développée dans le domaine de la robotique. Un robot est défini comme un appareil automatique capable de manipuler des objets ou d'exécuter des opérations selon un programme fixe ou modifiable. A travers cet agent, la robotique permet à l'homme d'étendre son emprise sur l'environnement par l'exécution reproductible de tâches répétitives et la possibilité d'agir à distance, évitant ainsi l'exposition d'êtres humains à un environnement dangereux.

Le développement d'une autonomie pour de tels systèmes, c'est-à-dire la présence de mécanismes de décision et d'action capables de s'adapter permettant de réagir avec flexibilité aux variations de leur environnement, augmente encore les possibilités qu'ils peuvent offrir. C'est le cas par exemple dans le domaine spatial où le temps de latence des communications rend les télé-opérations difficiles, mais de nombreux domaines sont concernés, du milieu médical jusqu'au divertissement. Cependant, l'utilisation des systèmes autonomes dans des applications critiques comme la chirurgie et l'espace pose le problème de leur sûreté de fonctionnement, en raison des conséquences potentiellement désastreuses de leur défaillance. Les études de fiabilité ne donnent pas des résultats très encourageant de ce point de vue : [17], menée pendant deux ans sur treize robots de sept modèles différents, donne un MTBF (*Mean Time Between Failure*) de 8 heures ; [141], menée pendant cinq mois sur une dizaine de robots autonomes d'un même modèle, donne un MTBF de 4,6 heures. Dans le *DARPA Grand Challenge* organisé en mars 2004, quinze véhicules terrestres autonomes ont eu pour tâche de parcourir 228 kilomètres dans le désert Mojave ; les deux compétiteurs les plus performants n'ont pas dépassé le douzième kilomètre. Fort heureusement, les éditions suivantes (*Grand Challenge 2005* et *Urban Challenge 2007*) ont eu plus de succès. Enfin, la norme IEC61508 (IEC61508, partie 3, clause 7.4.3) illustre le peu de confiance accordée envers les mécanismes décisionnels à base d'intelligence artificielle : l'utilisation de tels mécanismes comme stratégie de tolérance aux fautes est admise au niveau de criticité le plus faible (appelé SIL1) mais déconseillée pour les niveaux d'intégrité correspondant aux fonctions les plus critiques (SIL2 à SIL4).

Maintenant que l'autonomie des systèmes robotisés est mise en œuvre avec de plus en plus de succès dans le milieu de la recherche, et que l'on commence à en voir des concrétisations dans des musées, dans l'espace ou dans des foyers et des hôpitaux, il est important de développer des mécanismes améliorant leur sûreté de fonctionnement. C'est dans cet objectif que s'inscrit notre travail : en particulier introduire les notions pertinentes au développement de mécanismes de sûreté de fonctionnement informatique pour les systèmes autonomes critiques, étudier les solutions déjà existantes dans ce domaine, proposer de nouveaux mécanismes de tolérance aux fautes pour les architectures de systèmes autonomes critiques, et enfin évaluer

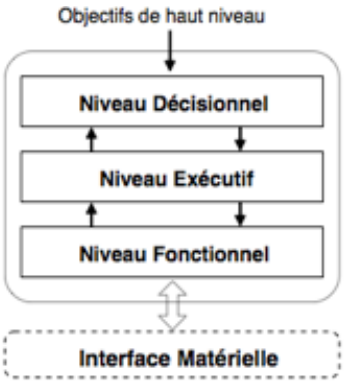


Figure 1 – Exemple d'architecture hiérarchisée

FIGURE 2.3 – Exemple d'architecture hiérarchisée

la robustesse et la résilience de ces systèmes dans un contexte ou l'évaluation expérimentale n'est pas envisageable.

Ces travaux ont débuté dans le cadre de la thèse de Benjamin Lussier [102], qui sont relatés ci-après. Nous y avons proposé l'utilisation de la redondance de la couche de planification dans l'architecture LAAS (pour *LAAS Architecture for Autonomous Systems*) à 3 niveaux. Nous introduisons, proposons différentes implémentations et évaluons cette approche. Les résultats obtenus montrent que le surcoût est minime pour un apport important en termes de tolérance aux fautes de développement.

Ces travaux se sont poursuivis dans le cadre de la thèse de Hoang Nam Chu [28], où nous avons procédé à l'évaluation de la robustesse de deux versions de la couche exécutive de l'architecture LAAS. La première est la version historique, à base de composants interconnectés grâce à l'intergiciel maison GENoM. La seconde version est à base de composants BIP [127] où des propriétés de sécurité-innocuité sont injectées directement dans la glu inter-composants.

2.1.4.2 Planification tolérante aux fautes pour robots critiques

Un système autonome est un système capable de percevoir, analyser, planifier, établir des décisions, et agir afin d'atteindre des objectifs assignés par un opérateur humain, malgré les incertitudes liées à l'environnement. La prise de décision des systèmes autonomes complexes (robots d'exploration spatiale ou sous-marine, guide de musée, assistant médical), actuellement majoritairement mise en place par la fonction de planification, nécessite d'une part des mécanismes décisionnels comprenant une fonction d'inférence utilisée pour résoudre différents types de problèmes, et d'autre part des modèles de l'environnement spécifiques à un type d'application. L'architecture la plus employée pour de tels systèmes est l'architecture hiérarchisée généralement composée d'un niveau décisionnel (planification, prise de décision), un niveau exécutif (supervision de l'exécution des actions de haut niveau), et un niveau fonc-

tionnel (exécution de tâches de bas niveau). Dans le domaine récent des systèmes autonomes, la sûreté de fonctionnement est encore peu employée : on retrouve principalement des techniques de prévention des fautes (modularité des composants, environnement de conception) et d'élimination des fautes (tests), mais leur utilisation n'est pas encore systématique. De plus, les fautes de développement ne sont que peu considérées par rapport aux fautes physiques. La notion de robustesse liée à la robotique, c'est-à-dire la tolérance vis-à-vis des incertitudes de l'environnement, peut apporter intrinsèquement une certaine tolérance aux fautes, mais reste loin d'être suffisante pour des applications critiques.

La partie théorique des travaux cherchait à améliorer la sûreté de fonctionnement des systèmes autonomes en proposant les premiers mécanismes de tolérance aux fautes applicables aux planificateurs. Ces mécanismes visent en particulier à tolérer des fautes de développement dans les modèles de planification. Les techniques traditionnelles de tolérance aux fautes sont difficiles à appliquer directement aux mécanismes décisionnels, notamment parce qu'il est difficile d'évaluer l'exactitude et la qualité d'un plan avant de l'avoir exécuté. Quatre mécanismes de détection complémentaires applicables aux systèmes autonomes hiérarchisés ont été proposés, basés sur des chiens de garde temporels, la vérification de règles dérivées des spécifications du système, et la détection d'échecs d'action pendant l'exécution du plan. Deux politiques de rétablissement (après détection de faute) sont données, toutes deux basées sur une diversification du modèle de planification, dans le but de tolérer les fautes de développement. Ces différents mécanismes sont implantés dans un composant de tolérance aux fautes appelé FTplan, capable de gérer l'exécution parallèle de plusieurs planificateurs chacun disposant d'un modèle diversifié.

Du point de vue pratique, nous avons réalisé une implémentation des mécanismes proposés et une évaluation de leur efficacité. Un composant FTplan, réalisant une politique de rétablissement et trois des quatre mécanismes de détection proposés, ont été implantés dans l'architecture de systèmes autonomes LAAS (cf. figure 2.4). Les principaux composants de cette architecture sont les modules fonctionnels GenoM, qui contrôlent les capteurs et actionneurs (ou, comme dans la figure, ceux simulés par l'environnement de simulation), l'exécutif procédural OpenPRS, et le planificateur IxTeT. La diversification d'un modèle de planification et des modifications au planificateur IxTeT ont été réalisées pour disposer de deux variantes, et permettre leur prise de contrôle par FTplan. L'environnement d'évaluation proposé est basé sur l'injection de fautes (pour évaluer l'efficacité des mécanismes de tolérance aux fautes proposés) et la simulation physique (pour éviter d'une part des comportements dangereux causés par les fautes injectées, et d'autre part permettre une automatisation des expériences). Quatre missions et quatre environnements différents (soit 16 jeux de test au total) ont été utilisés pour simuler la variabilité des environnements d'exécution d'un système autonome. En tout, plus de 3500 expériences ont été menées, pour plus de 120 heures de simulation. L'analyse des résultats montre un coût mineur des mécanismes de tolérance aux fautes au niveau des performances du système, et une amélioration de la fiabilité du système de 30 à 80% suivant les objectifs considérés.

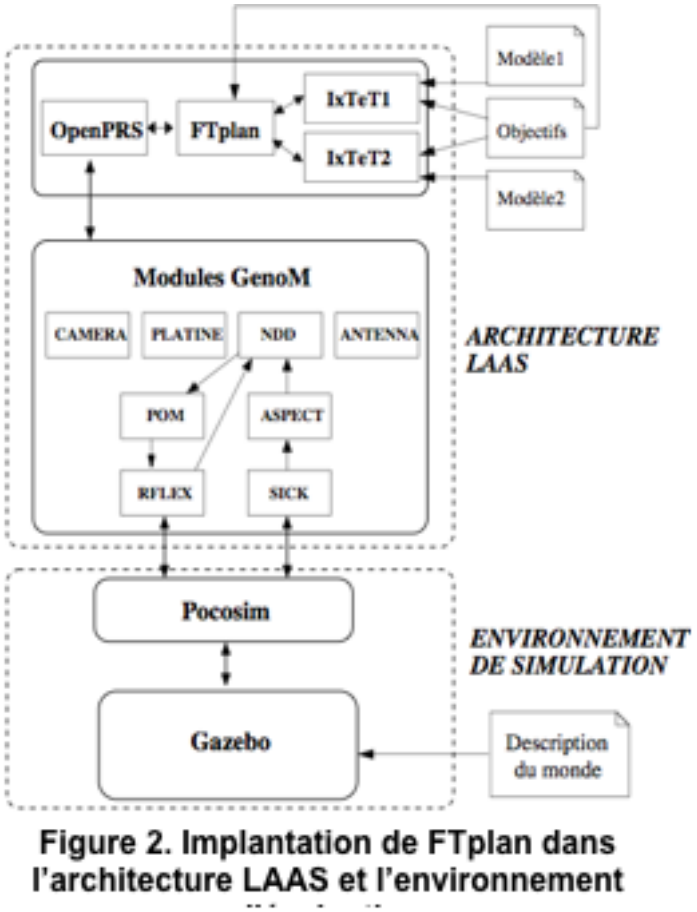


FIGURE 2.4 – Implantation de FTplan dans l'architecture LAAS

Plusieurs leçons instructives vis-à-vis de l'évaluation de systèmes autonomes ont également été tirées. Les mécanismes proposés, bien qu'améliorant objectivement le comportement du système cible en présence de fautes, ne répondent qu'à un aspect de la tolérance aux fautes dans les systèmes autonomes. Tout d'abord, ils se concentrent sur les fautes de développement dans les connaissances du planificateur, et ne couvrent pas toutes les fautes liées aux mécanismes décisionnels. En particulier, ils se concentrent sur l'aspect fiabilité du système (c'est-à-dire l'accomplissement de sa mission), plutôt que de s'intéresser à l'aspect sécurité. Ils doivent ainsi être utilisés de façon complémentaires à d'autres mécanismes, couvrant d'autres domaines de fautes. Néanmoins, les travaux menés montrent objectivement que des mécanismes de tolérance aux fautes appliqués aux planificateurs améliorent sensiblement le comportement d'un système autonome en présence de fautes. L'environnement d'évaluation des mécanismes de tolérance aux fautes s'est avéré apte à stresser l'autonomie du système, et à mesurer l'impact des fautes injectées sur le comportement du robot. Basée sur la simulation des composants matériels du robot et de son environnement, cette approche de simulation ne doit pas se substituer aux tests sur des plateformes réelles, mais doit permettre, à travers une approche complémentaire, d'effectuer des tests intensifs et une évaluation intégrée de mécanismes décisionnels. Malgré des difficultés liées à la quantification du résultat

des expériences, cet environnement d'évaluation s'est montré efficace pour la comparaison de systèmes différents.

2.1.5 Conclusion

Dans cette section, nous avons relaté nos travaux concernant la conception d'architectures et d'algorithmes pour la conception de systèmes mobiquitaires résilients. Ces travaux ont porté sur des aspects liés aux communications (diffusion, découverte de services et de localisation, géo-registres), et sur des aspects de plus haut niveau tels que l'incitation à la coopération ou la sauvegarde coopérative. De façon générale, ces travaux nous ont permis d'envisager la conception d'une architecture générique pour les systèmes mobiquitaires coopératifs et d'avancer vers des architectures robotiques sûres. Ces travaux ne sont certes pas terminés, nous en discutons la prospective en section 2.3, mais ils ont toutefois acquis un bon niveau de maturité [81]. Dans la suite de ce chapitre, nous nous posons donc la question de l'évaluation de la résilience des systèmes présentés dans cette section.

2.2 Évaluation de la résilience des systèmes mobiles

Lors de la conception de systèmes résilients se pose la question cruciale de leur évaluation. Dans cette section, nous abordons cette question sous différents angles. En effet, certains de nos travaux ont porté sur l'évaluation analytique, à base de modèles de Markov ou de réseaux de Petri. Ces travaux font l'objet de la sous-section 2.2.1. D'autres travaux ont également porté sur l'évaluation expérimentale et la section 2.2.2 présente la plateforme que nous avons développée à ce sujet.

2.2.1 Évaluation analytique de la sauvegarde coopérative

Thèse : [33] - Publication majeure : [34]

Dans [34], nous abordons l'évaluation analytique de la sûreté de fonctionnement du service de sauvegarde coopérative présenté en section 2.1.1. En effet, comme mentionné précédemment, plusieurs stratégies de réplication et de dispersion sont envisageables pour ce service. Par exemple, en termes de réplication, il est possible de créer de simples copies complètes des données utiles (réplication simple), ou encore d'envisager des techniques plus élaborées à base de *codes à effacement*. Ce choix de politique a un impact sur l'efficacité du stockage et sur la confidentialité des données, tel que décrit dans [36], mais son impact sur la disponibilité des données n'était pas facile à évaluer et son étude a donc fait l'objet de ces travaux.

Codes à effacement. Un *code à effacement (n,k)* produit un mot de n bits à partir d'une entrée de k bits. Pour recouvrer les données originales, m bits sont nécessaires et suffisants, avec $k \leq m \leq n$. Lorsque $m = k$, le code est dit optimal. Lorsque chaque k fragment est stocké sur un dispositif différent, un code optimal permet de tolérer $n-k$ défaillances, au-delà de la défaillance du client du service. Le coût du stockage est alors de $\frac{n}{k}$. Pour tolérer f fautes, il faut $n = k + f$, ce qui donne un coût de $1 + \frac{f}{k}$. Un code à effacement (avec $k \geq 2$) est donc plus efficace qu'une simple réplication (avec $k = 1$) pour ce qui concerne la quantité d'information à stocker.

Stratégies de réplication et opportunités de sauvegarde. Nous considérons le cas d'un client du service de sauvegarde qui doit répliquer un seul élément de données. Ce client suit une stratégie prédéfinie, à base d'un code à effacement (n,k) dont les paramètres sont également définis hors-ligne. Lorsque $k = 1$, la stratégie correspond à une réplication simple. Nous considérons une stratégie statique. En particulier, nous considérons que les clients ne détectent pas les défaillances des contributeurs du service. Les clients ne peuvent donc pas décider de créer des répliques supplémentaires d'un fragment lorsque le contributeur qui le stockait est défaillant. Nous considérons que chaque rencontre avec un nouveau dispositif est une nouvelle opportunité de sauvegarde. Précisément, cela implique qu'une relation de confiance peut être établie avec chaque contributeur et que ce dernier dispose de suffisamment d'espace de stockage. L'effet de cette hypothèse peut aisément être étudié en faisant

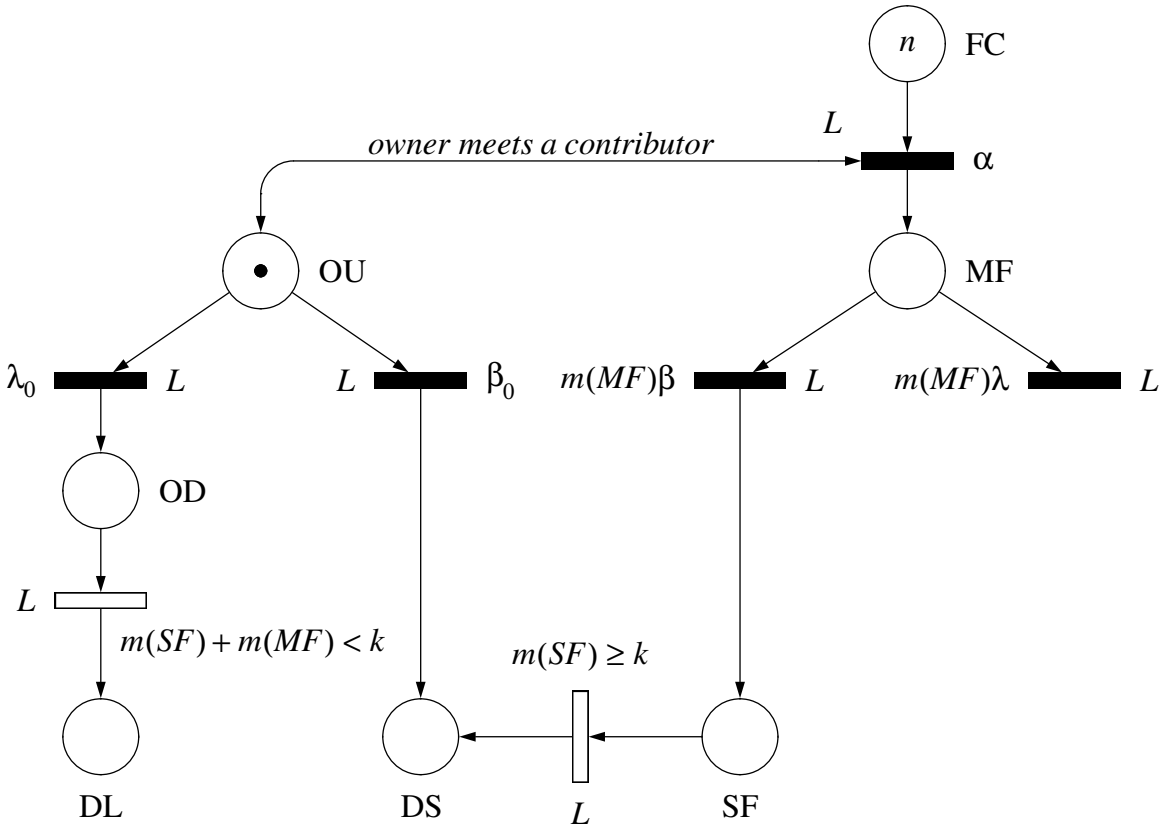


FIGURE 2.5 – Réseau de Petri de la réplication et dispersion d'un code à effacement (n, k)

varier le taux de rencontre des contributeurs. Nous considérons qu'une connexion à un réseau d'infrastructure n'est exploitée que lorsque la bande passante est abondante et peu chère, par conséquent, en cas de connexion à l'Internet, tous les fragments stockés sont transférés.

Modélisation. La figure 2.5 présente le réseau de Petri stochastique généralisé (GSPN) du service de sauvegarde coopérative avec un algorithme à base de code à effacement (n, k). Ce modèle se focalise sur la partie *ad hoc* de l'algorithme. Un fragment de donnée est considéré comme "sûr" (qu'il ne peut plus être perdu) lorsque son propriétaire ou le contributeur qui le stocke se connecte à Internet. Inversement, lorsqu'un dispositif participant défaille, tous les fragments qu'il stocke sont perdus. En conséquence, avec un code à effacement (n, k), une donnée est perdue lorsque son propriétaire défaille et moins de k contributeurs stockent encore un fragment ou en ont stocké un et l'ont transféré sur Internet. Ce modèle consiste en trois processus principaux représentés par des transitions temporisées :

- Un processus avec le taux α qui modélise la rencontre d'un contributeur par le propriétaire, où ce dernier envoie un fragment au contributeur.
- Un processus qui modélise la connexion d'un dispositif à Internet, avec le taux β_0 pour le propriétaire et β pour les contributeurs.

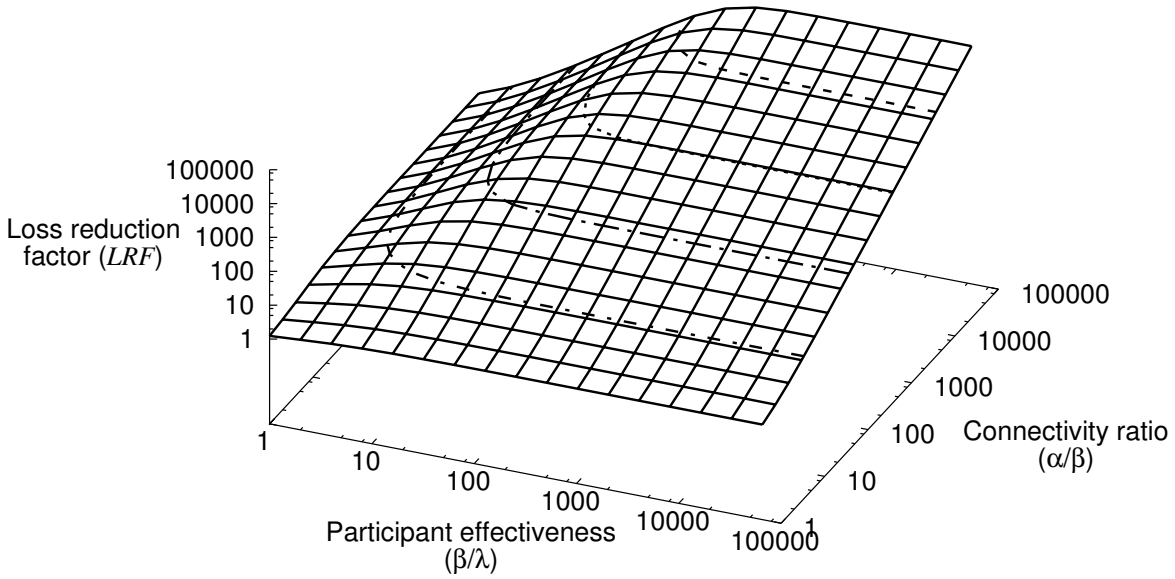


FIGURE 2.6 – Le LRF pour un code à effacement $(2, 1)$

– Un processus qui représente la défaillance d'un dispositif, λ_0 pour le propriétaire et λ pour les contributeurs.

Le réseau de la figure 2.5 est divisé en deux sous-réseaux. Le sous-réseau de gauche représente l'évolution d'une donnée chez son propriétaire : soit elle est perdue car le propriétaire défaille (avec le taux λ_0), soit elle est transférée sur Internet car il s'y connecte (avec le taux β_0). Le sous-réseau de droite décrit : (i) le processus de création de fragments de donnée lorsqu'un contributeur est rencontré (avec le taux α) et (ii) le processus de transfert du fragment sur Internet par le contributeur (avec le taux β) ou sa perte par défaillance du contributeur (avec le taux λ). Un tel modèle par GSPN peut être utilisé pour générer les chaînes de Markov associées avec n'importe quel code à effacement (n, k). Ces chaînes de Markovsont détaillées dans [33]. Le nombre total d'états dans une chaîne de Markov (n, k) est $O(n^2)$.

Analyse. Le paramètre d'analyse le plus important est ici la *probabilité de perte* des données, notée PL. On compare, pour un code (n, k) donné, PL avec la probabilité de perte de référence, notée PL_{ref}, qui correspond à un scénario sans service de sauvegarde coopérative où le propriétaire défaille avec le taux λ_0 et il se connecte à Internet avec le taux β_0. $PL_{ref} = \frac{\lambda_0}{\lambda_0+\beta_0}$. On note LRF le facteur de réduction de la probabilité de perte : $LRF = \frac{PL_{ref}}{PL}$. Un LRF de 100 veut dire qu'une donnée a une probabilité 100 fois plus faible d'être perdue avec la sauvegarde coopérative que sans elle.

La figure 2.6 donne le LRF pour un service de sauvegarde utilisant un code à effacement (2,1) et la stratégie de réplication résumée ci-dessus. Les propriétaires et contributeurs ont ici les mêmes caractéristiques : $\lambda_0 = \lambda$ et $\beta_0 = \beta$. On peut remarquer trois choses sur cette figure :

– Comme on pouvait s'en douter, l'utilisation d'un service de sauvegarde coopérative n'est pas très judicieux lorsque les accès à Internet sont fréquents ($\frac{\alpha}{\beta} \simeq 1$). Si on re-

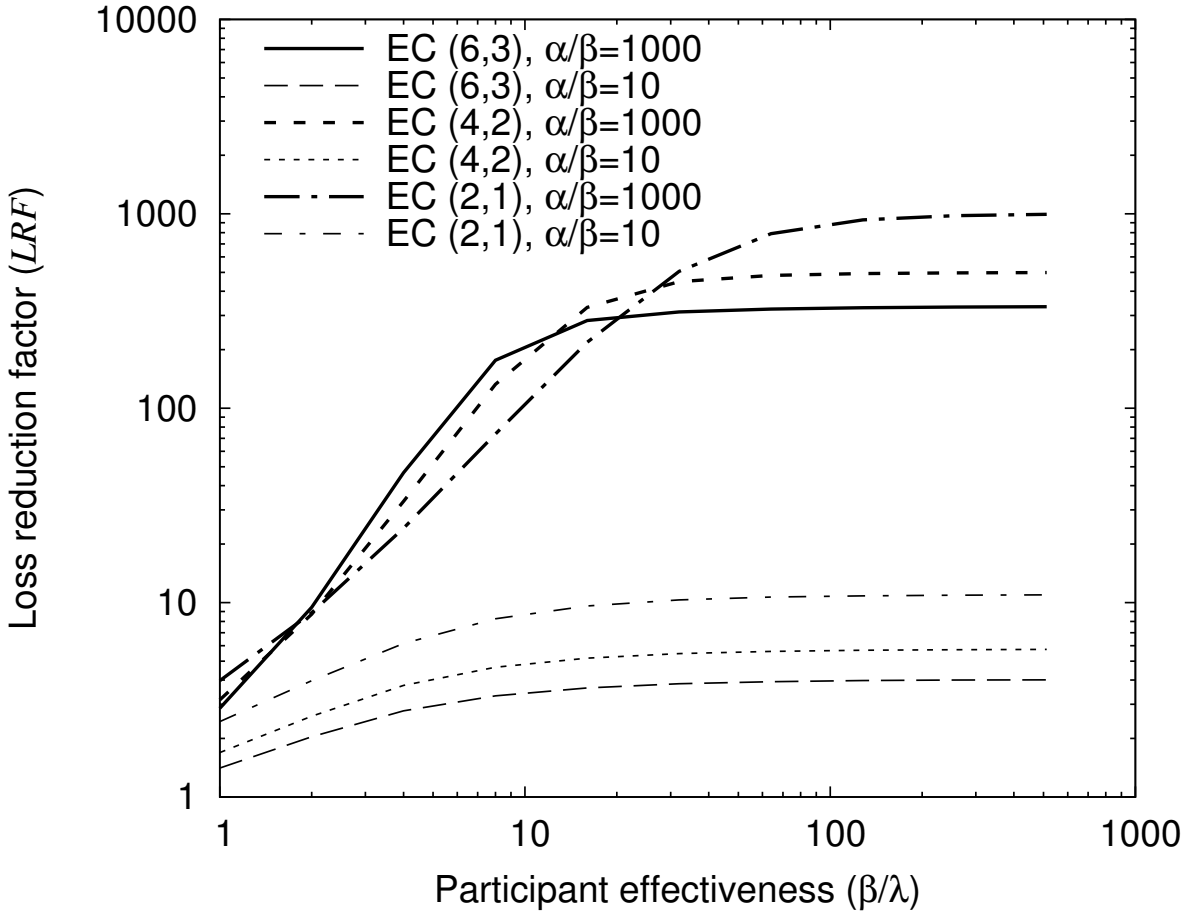


FIGURE 2.7 – Le LRF pour différents codes à effacement

cherche une amélioration d'un ordre de grandeur, l'environnement doit alors satisfaire $\frac{\beta}{\lambda} \geq 2$ et $\frac{\alpha}{\beta} \geq 10$. Ces conditions sont relativement peu contraignantes : des contributeurs qui se connectent deux fois plus souvent à Internet qu'ils ne défaillent et la rencontre de 10 contributeurs pour une connexion à Internet.

– Pour tout ratio $\frac{\alpha}{\beta}$, le LRF arrive à une asymptote après un certain seuil de $\frac{\beta}{\lambda}$. Augmenter $\frac{\beta}{\lambda}$ une fois passé ce seuil ne sert donc plus à rien.

– Le LRF croit proportionnellement à $\frac{\alpha}{\beta}$ jusqu'à un certain seuil puis suit une asymptote.

L'analyse du comportement asymptotique du LRF peut être trouvé dans [34].

La figure 2.7 permet de comparer les LRF pour différents codes à effacement. Chaque code à effacement a un coût de stockage équivalent $\frac{n}{k} = 2$. On y voit qu'à partir d'un certain ratio $\frac{\beta}{\lambda}$, la réplication simple est préférable ($\frac{\beta}{\lambda} \simeq 30$ pour les courbes avec $\frac{\alpha}{\beta} = 1000$. Si l'on affine cette analyse [34] on se rend compte que les codes à effacement à coût de stockage de 2 sont plus efficaces dans une zone où $\frac{\alpha}{\beta} \geq 100$ et $1 \leq \frac{\beta}{\lambda} \leq 100$. Cette zone se restreint lorsque le coût de stockage augmente. Toutefois il faut noter que l'utilisation des codes à effacement, si elle n'est pas particulièrement bénéfique pour la disponibilité, permet d'accroître notablement la confidentialité des données pour un coût de calcul bien inférieur à toute technique à base de chiffrement [36].

2.2.2 Évaluation expérimentale

Publication majeure : [81] - Autres publications : [79, 78, 80, 84, 82, 83]

Jusqu'à présent, relativement peu de travaux ont porté sur l'évaluation de la résilience dans les systèmes ubiquitaires. La plupart des travaux dans ce domaine ont porté sur l'expérience utilisateur et sur les interfaces hommes-machines. En ce qui concerne l'évaluation de protocoles et d'architectures, c'est généralement la simulation de réseau mobile qui est utilisée [38, 62]. Or les simulateurs utilisent un modèle physique parfois simpliste des différents composants du système (réseau sans fil, mobilité, etc.). Cela soulève le problème de la couverture des hypothèses sous-jacentes de la simulation, et de la représentativité des expériences conduites [20]. Pour autant, il existe très peu de travaux ayant porté sur l'évaluation de protocoles dans un environnement expérimental réaliste. Un état de l'art récent se trouve dans [83].

Les travaux relatés dans cette section ont donc porté sur la réalisation d'une plateforme réaliste qui permet, en laboratoire, d'évaluer et de valider des algorithmes de tolérance aux fautes pour systèmes mobiquitaires, tels que ceux présentés section 2.1. L'objectif était donc de concevoir une plateforme d'expérimentation qui permette de conduire des expériences reproductibles (y compris concernant les aspects mobilité) afin de compléter la validation par simulation de systèmes critiques.

La plateforme ARUM (*an Approach for the Resilience of Ubiquitous Mobile systems*) [1] est une plateforme d'évaluation expérimentale constituée de dispositifs fixes et mobiles [79, 78, 80, 84, 82, 83]. Chaque dispositif mobile est composé d'un robot programmable pour déplacer le dispositif, un ordinateur portable équipé de plusieurs interfaces réseaux, et d'un moyen de positionnement en intérieur ou en extérieur. Les éléments fixes de la plateforme concernent donc la contrepartie de l'infrastructure : un système de positionnement en intérieur, le support des communications sans-fil pour le mode *infrastructure*, ainsi que plusieurs serveurs (de stockage, de calcul, de positionnement, etc.). Cette plateforme est mise en place dans un laboratoire de $100m^2$ environ, où les dispositifs mobiles peuvent se déplacer. En faisant varier le facteur d'échelle de la plateforme, on peut émuler des systèmes de différentes tailles. Pour l'exemple applicatif que nous allons utiliser, et qui concerne la boîte noire distribuée décrite en section 2.1.3, le facteur de réduction d'échelle est de 50, comme le montre le tableau 2.1.

2.2.2.1 Localisation

La partie localisation est critique. En effet, il était nécessaire d'obtenir une précision importante en intérieur mais également un rafraîchissement fréquent de la position de façon à contrôler efficacement la trajectoire du robot. Nos spécifications étaient une précision de l'ordre du centimètre et une fréquence de rafraîchissement de l'ordre du Hertz. Il existe un

1. ARUM est un projet de recherche financé par le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS).

TABLE 2.1 – Réduction d'échelle dans ARUM : l'exemple automobile

Dispositif	**Vraie grandeur**	**Mise à l'échelle 1/50**
Communications sans-fil	portée : $100m$	portée : $2m$
Précision de la localisation	$5m$	$< 10cm$
Taille d'un nœud	quelques mètres	quelques dizaines de centimètres
Vitesse d'un nœud	quelques m/s	$< 1m/s$

certain nombre de technologies de positionnement en intérieur [65] qui différent tant par leurs caractéristiques, cf. la figure 2.8, que par leur prix. Nous utilisons actuellement le système *Hagisonic StarGazer technology* [2]. C'est un système à base de caméra infra-rouge embarquée qui observe et triangule une constellation de marqueurs réfléchissant qui sont positionnés au plafond. Ses performances est de l'ordre de quelques millimètres et sa fréquence de 10 Hz.

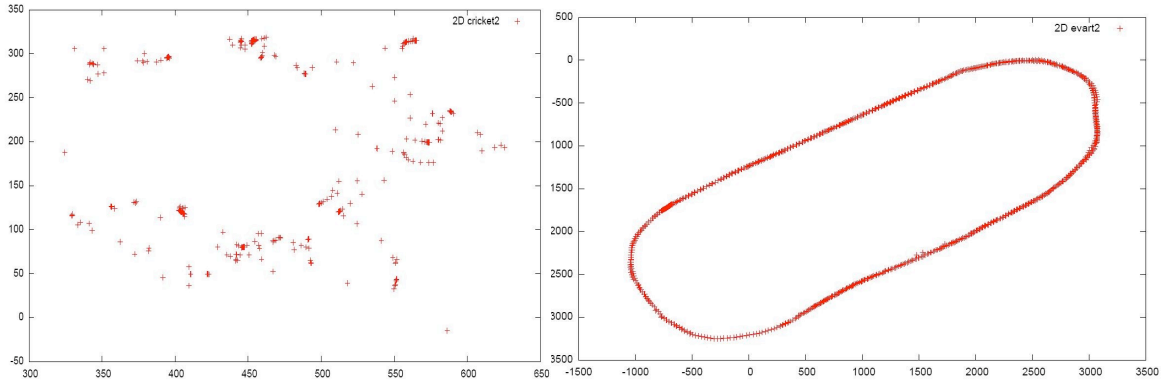


FIGURE 2.8 – Comparaison du positionnement avec deux technologies (à gauche ultrasons et à droite en infrarouge).

2.2.2.2 WiFi à portée réduite

La portée de transmission des dispositifs participants doit être ajustée de façon à respecter le facteur de réduction (ou d'augmentation) d'échelle. Dans notre scénario, il s'agit donc de réduire cette portée à 2m (cf. table 2.1) mais idéalement cette portée doit pouvoir être variable. Bien que quelques pilotes de cartes WiFi propose une API pour réduire la puissance de transmission, et en conséquence la portée, dans la plupart des cas cette fonctionnalité n'est pas utilisable. Une solution satisfaisante a consisté à utiliser des atténuateurs de signal [3] placés

2. Hagisonic - http ://www.hagisonic.com/

3. Un atténuateur de signal est un dispositif électronique qui réduit l'amplitude ou la puissance d'un signal, sans modifier sensiblement la forme de son onde

entre les cartes WiFi et leurs antennes. L'impédance nécessaire des atténuateurs dépend de nombreux paramètres, tels que la puissance d'émission et les caractéristiques des antennes mais aussi de la vitesse des robots, de la forme de la pièce, etc. Nous avons du procéder à une étude empirique[78] afin de calibrer notre système.

2.2.2.3 Mobilité

Afin de permettre la reproductibilité des expériences en terme de mobilité, nous avons mis en place une plateforme robotique qui se charge de mouvoir les dispositifs. Le rôle de ces robots est donc "d'implémenter" la mobilité des nœuds en fonction d'un certain scénario de mobilité. Nous utilisons un rover Lynxmotion 4WD qui peut transporter un ordinateur portable de 2kg pendant quelques heures à une vitesse d'1m/s sans nécessiter de recharger sa batterie. Nous les avons équipé de capteurs de proximité infra-rouge pour éviter les collisions. Cette plateforme est illustrée figure 2.9. Le logiciel de contrôle du mouvement tourne sur l'ordinateur portable transporté et est totalement indépendant du logiciel sous test. Il envoie des commandes de vitesse linéaire et angulaire au robot à travers une interface série.

FIGURE 2.9 – Les robots de la plateforme ARUM

2.2.2.4 Scénarios de mobilité

Le scénario de mobilité suivi par le logiciel de commande du robot doit être défini par un opérateur humain. Nous avons développé une interface de conception de trajets qui permet de dessiner des trajectoires, de configurer, de visualiser et de gérer des patrons de mobilité. Cette interface permet d'assigner des patrons différents pour chaque robot, patrons qui peuvent être synchronisés en des points de rendez-vous. Ce logiciel télécommande les expérimentations à partir d'un serveur connecté au réseau de supervision de la plateforme ARUM.

2.2.2.5 Application à la boîte noire distribuée

Nous avons pu, sur cette plateforme, évaluer l'implémentation de la boîte noire distribuée décrite section 2.1.3 sur trois véhicules automobiles et un véhicule de police. Pendant la pre-

mière partie du scénario, les trois véhicules suivent une trajectoire, se doublent ou se suivent, pendant que chacun participe à la sauvegarde des données de la boîte noire distribuée. Puis un des véhicules perd le contrôle, sort de sa trajectoire et a un accident. Lorsque l'accident est signalé, l'identification du véhicule et l'heure approximative de l'accident sont recueillies par la police, ce qui lui permet d'effectuer une requête de restauration des données du véhicule accidenté. Une fois que les données sont recueillies, la police est alors capable de rejouer le scénario de l'accident. Un film de cette démonstration est visible à l'adresse suivante : http ://videotheque.cnrs.fr/doc=2712 ?langue=EN.

2.3 Conclusion et Prospectives

Nous avons abordé, dans ce chapitre, mes travaux concernant l'algorithmique et l'architecture pour la résilience des systèmes mobiquitaires. Ces travaux, comme ceux du chapitre précédent, peuvent être très théoriques (comme ceux portant sur les géo-registres) ou très pratiques (l'évaluation expérimentale des systèmes mobiquitaires). Ils portent pour autant sur le même modèle de système : des nœuds légers et mobiles sont interconnectés par un réseau adhoc, et parfois par un réseau d'infrastructure. Ces nœuds coopèrent afin de réaliser un service, tel que la sauvegarde coopérative. Le service émerge donc, grâce à la coopération des nœuds, et est dépendant de celle-ci. Les aspects d'incitation à la coopération n'ont pas directement été abordés dans mes travaux mais, en revanche, le traitement des malveillances n'est jamais bien loin et sera d'ailleurs abordée dans le chapitre suivant, et dans ma prospective.

La problématique de la gestion de la mobilité, qu'elle soit vécue comme un élément positif (pour la dissémination des données dans le cadre le la sauvegarde coopérative) ou comme un élément négatif (lorsqu'un nœud disparaît d'un géo-registre, a-t-il défailli ou s'est-il déplacé ?), nous a amené à étudier la mobilité humaine [85]. Pour ces études, nous avons été amenés à équiper des individus de téléphones portables équipés d'un logiciel qui trace leurs déplacements, et d'autres paramètres liés à leur activité. Ces études ont tout de suite soulevé des questions d'ordre déontologique liées à la protection de la vie privée des cobayes. Ce sont ces questions qui m'ont conduit à me pencher sur les liens entre la mobilité et la vie privée. Le chapitre suivant relate les travaux qui ont découlé de ces questionnements.

Chapitre 3

Geoprivacy

Les travaux abordés dans ce chapitre concernent le respect de la vie privée dans un contexte géolocalisé. Pour plus de concision, j'emploie le terme de *geoprivacy*, veuillez m'excuser de l'anglicisme. Ces travaux représentent l'écrasante majorité de nos travaux actuels et de notre prospective. Par ce fait, il ne sont pas forcément au même degré de maturité que ceux présentés dans les chapitres précédents.

Nous assistons actuellement, et les travaux présentés dans le chapitre précédent y participent, à l'émergence de l'informatique ubiquitaire. L'instanciation actuelle de l'informatique ubiquitaire passe par l'équipement d'une majorité d'entre nous avec des dispositifs dotés de capacités de communication et de géolocalisation (*SmartPhones* et autres tablettes). Grâce à ces dispositifs, nous utilisons nombre de services géolocalisés, qui vont de la simple prévision météorologique locale, à la réservation de parking à sa destination, en passant par l'étiquetage spatial (*geo-tagging*) des photos. Dans la majeure partie des cas, les informations de localisation générées par le dispositif sont d'une part stockées sur l'appareil mais également transmises à une tierce partie (le fournisseur du service et/ou le fabricant de l'appareil). Ces données, si elles tombent entre les mains d'individus ou d'organisations mal intentionnés, peuvent être utilisées afin d'inférer des information d'ordre privé (identité, adresse, sexe, tranche de revenu annuel, etc.). D'ailleurs, certaines sociétés, comme SenseNetworks, ont construit leur modèle économique sur cette inférence de données privées ensuite vendues à des annonceurs spécialisés en publicité ciblée.

L'objet de ce chapitre est double : d'une part, présenter les dangers que représente la prolifération et la fuite, des données géolocalisées individuelles et d'autre part de présenter notre réflexion concernant la fourniture de systèmes géolocalisés respectueux de la vie-privée. La section 3.1 présente donc les risques et attaques concernant la geoprivacy. La section 3.2 présente, quant à elle, nos travaux visant à la fourniture d'une architecture pour les services localisés et coopératifs respectueux de la vie-privée. Enfin, la section 3.3 discute des pistes de travaux futurs dans ces domaines.

La table suivante présente les thèses encadrées et mes publications ayant trait à la geoprivacy, c'est bien entendu sur ces publications que se base ce chapitre.

Thèses	13	Miguel Nunez del Prado Cortez	en cours
	14	Moussa Traore	en cours
Publications	10	AINA	[51]
		SPRINGL	[52]
	11	Trans. on Data Privacy	[53]
		Dynam	[54]
		TR	[56]
	12	MPM	[55]
		ARMOR	[6]
		ARMOR	[50]

3.1 Attaques contre la geoprivacy

Thèse : Nunez del Prado Cortez, en cours - Publication majeure : [53] - Autres publications : [52] [51] [54] [55] [56]

Un système géolocalisé est un objet ou un dispositif auquel est associée une localisation. Il peut s'agir par exemple d'un *smartphone* ou d'un véhicule équipé d'un GPS. Habituellement, un système géolocalisé appartient à un individu (ou à un groupe d'individus comme une famille). Par conséquent, sa localisation correspond à la localisation de son propriétaire. Obtenir la localisation de ce système permet donc d'apprendre la localisation de l'individu qui le possède. Or, ces données de localisation sont distribuées assez largement et relativement faciles à obtenir, comme par exemple :

- chez les fabricants de smartphones qui récupèrent les traces de mobilité des appareils qu'ils ont vendus, cf. [30],
- dans les journaux de localisation stockés sur la plupart des appareils, cf. [2],
- chez les opérateurs de téléphonie mobile, ou les fournisseurs d'accès WiFi, qui stockent les cellules à partir desquelles leurs client se connectent, cf. [92],
- chez les fournisseurs d'applications et de services mobiles (basés sur la localisation ou non), ou encore par le biais des services de paiement (guichets automatiques, paiement par carte bancaire, télé-péages autoroutiers, etc.),
- sur les réseaux géo-sociaux, où de nombreux utilisateurs postent des informations sur les lieux où ils se trouvent, permettant par exemple d'en déduire lorsqu'ils ne sont pas à leur domicile, cf. [48],
- dans des bases de données publiques, comme Crawdad ou dans les répertoires OpenData et futurs MiData ou SmartDisclosure, initiatives des gouvernements britannique et américain, qui visent à libérer l'accès aux données personnelles gérées par les administrations et les entreprises,
- en installant un logiciel espion sur un ou plusieurs dispositifs, cf. SpyApps par exemple, ce qui est à la portée de n'importe qui, qu'il s'agisse de *script kiddie*, mari jaloux, entreprise soupçonneuse ou encore organisation criminelle,
- en développant un logiciel malveillant (*malware*), qui peut se répandre comme un virus, et qui collecte des traces. À l'heure où j'écris ces lignes, un tel logiciel n'a pas encore été signalé, mais au vu de l'actualité dans le domaine, ce n'est absolument pas de la science-fiction, voir par exemple le cas Find and Call qui collecte le carnet d'adresse, ou encore le rapport trimestriel Q1-2012 de MacAfee qui présente la croissance démesurée des logiciels malveillants pour smartphones.

Nous avons entrepris l'exploration et l'étude des différents types d'attaques d'inférence sur les données de localisation. Le résultat principal de ces travaux, en un mot, est que parmi les informations personnelles, apprendre la localisation d'un individu est une des plus grandes menaces pour le respect de sa vie privée. En effet, la connaissance de données spatio-temporelles concernant un individu peut être utilisée pour inférer la localisation de son domicile ou de son lieu de travail, pour tracer ses mouvements et ses habitudes, pour apprendre

ses centres d'intérêts, voire même pour détecter un changement soudain dans son comportement. Dans la section 3.1.1, nous donnons une classification et un bref aperçu des différentes attaques d'inférence, dont certaines seront illustrées dans la suite de ce chapitre : attaques par heuristiques en section 3.1.2.1, identification des points d'intérêts en section 3.1.2.2, modélisation de la mobilité en section 3.1.2.2, prédiction des mouvements en section 3.1.2.3 et dés-anonymisation en section 3.1.2.4. Enfin, en section 3.1.3, nous discutons de pistes pour des attaques futures.

3.1.1 Tentative de classification des attaques

Une attaque d'inférence cherche, à partir de données individuelles géolocalisées et parfois de connaissances auxiliaires, à inférer de nouvelles connaissances [90]. Par exemple, une attaque d'inférence peut consister à identifier le domicile ou le lieu de travail d'un individu. Les informations auxiliaires représentent la connaissance que l'adversaire a collectée préalablement (par exemple grâce à des attaques d'inférences précédentes ou grâce à l'accès à une source de données publiques) et qui peuvent l'aider à conduire une attaque d'inférence. Nous proposons de classer les attaques d'inférence selon trois dimensions : le type de données utilisées, l'objectif de l'attaque et, enfin, la technique utilisée.

3.1.1.1 Données Géolocalisées

Comme nous l'avons vu en introduction, la croissance rapide d'applications et services basés sur la localisation ont multiplié les sources potentielles de données géolocalisées. Les données générées par ces différentes sources varient dans leur forme et leur contenu, mais partagent également des caractéristiques importantes. Concernant le type de données, nous différencions principalement les traces de mobilité et les traces de contact. Une *trace de mobilité* est caractérisée par :

- Un *identifiant*, qui peut être l'identifiant du dispositif, un pseudonyme, ou encore être inconnu pour un anonymat complet. Un pseudonyme est en général utilisé quand on désire protéger l'identité réelle de l'individu ou du dispositif, tout en étant capable de lier les différentes actions effectuées par un même utilisateur.
- Des *coordonnées spatiales*, qui peuvent être une position GPS (latitude, longitude, altitude, par exemple), une zone spatiale (comme le nom d'une rue au sein d'une ville) ou aussi un label sémantique ("maison" ou "travail").
- Une *estampille temporelle*, qui peut être un horodatage précis ou simplement un intervalle de temps (par exemple entre 9h et 11h du matin).
- Des informations additionnelles et optionnelles, telles que la vitesse et la direction de déplacement, la présence d'autres dispositifs ou individus dans le voisinage, ou encore la précision de la position mesurée.

Des *traces de contact* sont une forme particulière de trace de mobilité qui consiste en l'enregistrement des rencontres entre différents dispositifs. Ce type de traces est donc composé des identifiants des dispositifs rencontrés, accompagnés d'une estampille temporelle. Ce type de données peut, par exemple, être collecté par un dispositif qui ne dispose pas de capacité de géolocalisation mais qui est capable de scanner son environnement radio (WiFi, Bluetooth, etc.) pour détecter la présence de voisins.

Un *ensemble de données géolocalisées* est un ensemble de données qui contient des traces de mobilité pour plusieurs individus (dispositifs). D'un point de vue technique, ces données peuvent avoir été collectées soit en enregistrant sur le dispositif lui-même ses déplacements durant une certaine période, soit par un serveur centralisé qui est capable de suivre la localisation de ces dispositifs en temps-réel. Une *suite de traces* est une collection de traces de mobilité qui correspondent aux déplacements d'un individu durant une certaine période. Un ensemble de données géolocalisées est généralement constitué d'un ensemble de suites de traces issues de différents individus.

3.1.1.2 Objectifs de l'Attaque

Un adversaire qui attaque des données géolocalisées peut avoir différents objectifs qui peuvent aller de l'identification du domicile de la cible jusqu'à la découverte de ses relations sociales, ou encore l'obtention de ses parcours de jogging favoris. Plus précisément, l'objectif d'une attaque d'inférence sur des données individuelles géolocalisées peut être :

1. *Identification des lieux importants* appelés les points d'intérêt ou POIs (pour /textitPoints-of-Interest), qui caractérisent les intérêts d'un individu [71]. Chaque individu, a, dans ses déplacements, des lieux favoris, où il revient souvent. Un POI peut, par exemple, être son domicile, son lieu de travail, ou encore son gymnase habituel, son théâtre favori ou le local de son parti politique. Révéler les POIs d'un individu implique une brèche importante de vie privée étant donné que ces informations peuvent être utilisées pour inférer des informations sensibles telles que les hobbys, croyances religieuses, préférences politiques, ou encore les maladies potentielles. Par exemple, si un individu visite régulièrement une clinique spécialisée dans un certain type de maladie, on peut déduire qu'il a une probabilité non-négligeable d'avoir cette maladie. Certains POIs, comme le domicile ou le lieu de travail, peuvent parfois être considérés comme étant de notoriété publique si ces informations peuvent être découvertes par d'autres moyens tels qu'une simple recherche sur le Web ou dans les pages jaunes ou blanches. Dans ce cas particulier, le risque en terme de respect de la vie privée est principalement que ces données peuvent être utilisées pour dés-anonymiser un individu particulier qui avait été pseudonymisé alors qu'une combinaison de certains de ses POIs le caractérise de façon unique. En particulier le couple domicile-travail peut souvent être considéré comme quasi-identifiant.

2. *Inférence des mouvements* ou des positions passées, présentes, et futures d'un individu [90]. Les déplacements d'un individu sont empreints d'une certaine régularité [128, 60]. Dans [128] par exemple, les auteurs ont exploré les limites de la prévisibilité dans la mobilité humaine en analysant les motifs de mobilité de 50000 individus dans un jeu de données anonymisé issu d'une compagnie de téléphonie mobile. En mesurant l'entropie des trajectoires individuelles, les auteurs ont montré que cette prévisibilité était potentiellement de 93%. On peut donc aisément imaginer qu'à partir d'une connaissance des déplacements passés d'un individu, on puisse essayer de prédire ses déplacements futurs.
3. *Apprendre la sémantique de la mobilité d'un individu* à partir de la connaissance de ses POIs et motifs de déplacements. Par exemple, certains modèles de mobilité, tels que les trajectoires sémantiques [3, 129] ne représentent pas seulement l'évolution des mouvements d'un individu dans le temps, mais attachent également des étiquettes sémantiques aux endroits visités. À partir de cette information sémantique, l'adversaire peut se construire une meilleure compréhension des intérêts et du comportement en termes de mobilité d'un individu que simplement en observant ses motifs de mobilité. Par exemple, l'adversaire peut inférer qu'en semaine, l'individu cible quitte généralement sa maison (POI_1) pour amener ses enfants à l'école (POI_2) avant d'aller au travail (POI_3), ce qui est une connaissance sensiblement plus riche que le simple motif $POI_1 \prec POI_2 \prec POI_3$.
4. La *dés-anonymisation* consiste à relier les enregistrements d'un même individu, qui peuvent être contenus dans un unique ou plusieurs jeux de données, de façon anonyme ou sous différents pseudonymes. C'est l'équivalent en termes de geoprivacy du risque de divulgation statistique où le respect de la vie privée est mesuré en fonction du risque de pouvoir lier les fiches d'un même individu dans deux bases de données (par exemple établir qu'un individu particulier inscrit sur telle liste électorale est aussi un patient spécifique d'un hôpital). Dans un contexte géolocalisé, l'objectif d'une telle attaque pourrait être d'associer les mouvements de la voiture d'Alice (contenus, par exemple, dans le jeu de données A) avec les localisations de son téléphone portable (contenues dans le jeu de données B). Puisque les POIs d'un individus peuvent être considérés comme une forme d'empreinte digitale, la simple anonymisation ou pseudonymisation d'un jeu de données n'est clairement pas suffisante pour se protéger de telles attaques. En effet, une combinaison de localisations peut jouer le rôle d'un quasi-identifiant si ces localisations caractérisent de façon unique, ou presque, un individu de la même façon que la combinaison de son prénom et de son nom. Par exemple, dans [59], les auteurs montrent que la simple paire domicile-travail est quasiment unique et peut être considérée comme quasi-identifiant si la précision est suffisante (en particuliers si l'on dispose du nom des rues, et pas seulement des lieu-dits).

5. *Découverte des relations sociales* entre individus en considérant que deux individus qui sont en contact pendant une durée non-négligeable partagent une forme de lien social (il faut bien entendu accepter les faux positifs) [69]. Cette information peut être dérivée des traces de mobilité en observant que deux individus sont dans le voisinage l'un de l'autre de façon récurrente ou encore directement à partir de traces de contact.

3.1.1.3 Techniques d'Inférence et Sources d'Information

Nous décrivons ici quelques méthodes et algorithmes d'apprentissage qui peuvent être utilisées comme techniques d'inférence :

- Le *clustering* est une forme d'apprentissage non supervisé qui essaye de regrouper les objets similaires dans le même *cluster* tout en séparant les objets dissemblables dans des clusters différents. Un algorithme de clustering nécessite une *mesure de distance* (ou une métrique de similarité) pour quantifier la similarité/différence entre les objets et ainsi mener le processus de clustering. Une mesure de distance intuitive entre deux localisations est la distance euclidienne mais des métriques plus complexes peuvent être utilisées, comme par exemple la longueur du plus court chemin sur une carte routière. Il existe de nombreux algorithme de clustering. Par exemple, *k-means* est un algorithme de clustering itératif, qui donne un résultat de k clusters accompagnés de leur centroïde relatif (qui représentent, dans les faits, la moyenne des localisations de chaque cluster). Cet algorithme peut-être utilisé simplement pour découvrir les POIs d'un individu en lui donnant en entrée les traces de mobilité de ce dernier, ou encore pour trouver des *hotspots* en lui donnant des jeux de données de toute une population. Des techniques plus complexes existent, comme le clustering basé sur la densité [145], et peuvent être utilisées afin de répondre aux points faibles de k-means, en particuliers : le fait de devoir fixer le nombre de clusters k, et la forme sphérique des clusters.
- Des *modèles de mobilité* peuvent être appris à partir des traces de mobilité des individus. Ils peuvent être utilisés soit pour identifier ces individus dans ces jeux de données, même s'ils sont anonymes, ou pour prédire leurs prochains mouvements. Dans [97], par exemple, les auteurs montrent qu'il est possible d'entraîner un réseau Markovien relationnel, de telle sorte qu'il puisse prédire avec une relativement bonne précision la prochaine localisation ou l'activité courante d'un individu. Il est aussi possible d'utiliser des algorithmes de suivi de cible [121] afin de reconstruire les chemins suivis par plusieurs individus dans un ensemble de données anonymisées. D'autres modèles de mobilité [3, 129] attachent une étiquette sémantique aux endroits visités afin d'enrichir l'information extraite.
- Des *heuristiques* peuvent donner de bons résultats en pratique [89] pour identifier des POIs de façon efficace. Une heuristique peut être aussi simple que de choisir la dernière position avant minuit pour découvrir le domicile, ou de prendre la localisation la plus stable en journée pour découvrir le lieu de travail.

– Les *données issues des réseaux sociaux* sont une source potentielle d'information que l'adversaire peut utiliser pour construire son attaque. Le site web « Please Rob Me » est un exemple illustrant parfaitement comment il est possible, à partir d'informations publiques (en l'occurrence des message de Twitter), de construire un classifieur qui peut prédire si un individu est actuellement à son domicile ou non. Un autre exemple de source de données est constitué par Google Latitude qui permet de suivre en temps réel les mouvements de ses proches sur une carte, à supposer qu'ils aient préalablement confirmé accepter de partager cette information en cliquant sur un lien reçu par SMS sur leur téléphone. Cette technique de confirmation est particulièrement sensible aux *shower attacks* où un conjoint peu scrupuleux profite de la douche de sa moitié pour faire la confirmation sur son téléphone. La cible n'a alors aucun indice lui indiquant qu'il ou elle est maintenant suivi.
– Les données publiques représentent une autre source de données potentielle pour un attaquant. Par exemple, des outils de géocodage inverse permettent de retrouver une adresse à partir de coordonnées spatiales, les pages blanches peuvent ensuite servir à identifier le nom de la ou des personnes domiciliées à cette adresse.

Souvent, le processus d'inférence ne consiste pas seulement en l'application d'une unique attaque d'inférence. Il peut être un processus incrémental pendant lequel l'adversaire collecte de plus en plus d'informations par la combinaison de plusieurs attaques d'inférence et de différentes sources de données.

3.1.2 Attaques contre la geoprivacy : Quelques Contributions

Dans cette section, nous exposons quelques contributions à des attaques en vie privée sur des données géolocalisées. Dans la section 3.1.2.1, nous commençons avec des attaques heuristiques simples de type *BeginEnd*. En particuliers, nous y montrons comment nous avons pu retrouver, avec l'aide de sources de données publiques, le domicile d'un individu dans un ensemble de données pseudonymisé. Au delà du domicile, et de façon générale, nous donnons des algorithmes pour l'identification de POIs d'un individu et pour la construction d'un modèle de mobilité de celui-ci, cf. 3.1.2.2. Ce travail constitue la base des attaques de prédiction des mouvements et de dés-anonymisation que nous présentons respectivement en section 3.1.2.3 et 3.1.2.4.

3.1.2.1 Attaques par heuristiques

Dans [51], nous illustrons l'utilisation de l'outil GEPETO, décrit section 3.2.2, pour l'évaluation d'une attaque d'inférence basée sur une heuristique simple que nous appelons *BeginEnd*. Cette heuristique consiste, dans les grandes lignes, à ne considérer qu'un sous-ensemble très restreint de traces de mobilité d'un individu. En effet, lorsqu'un adversaire désire identifier les POIs d'un individu, tels que son domicile par exemple, il semble naturel de ne considérer

que les positions où l'individu n'est pas en mouvement. En l'occurrence, l'heuristique *BeginEnd* pousse à l'extrême ce raisonnement et ne conserve que la première et la dernière trace de la journée, qui ont une forte probabilité d'être proche de son domicile. Dans cette étude, nous avons utilisé un jeu de données issu du dépôt CRAWDAD. Ce jeu de données contient les traces de mobilité de taxis dans la région de San Francisco [119]. Il contient les traces GPS d'environ 500 véhicules collectées pendant 30 jours par la société YellowCabs. Les positions ont été collectées toutes les minutes pendant environ 12h par jour pour chaque véhicule.

L'algorithme *BeginEnd* (cf. algorithme 6) considère donc que la première et la dernière trace collectées, pour chaque jour de travail des taxis, peut représenter une information sensible telle que le domicile des conducteurs ou encore l'adresse de la compagnie de taxis. Cependant, les taxis ne travaillent pas seulement en journée, observer simplement la première et la dernière trace d'une journée n'est donc pas la bonne approche. C'est pourquoi l'algorithme considère qu'une nouvelle période de travail commence après une pause d'une durée donnée en paramètre, par exemple après 2 heures. L'algorithme recherche donc de telles pauses dans les données et extrait la trace précédent et la trace suivant chaque pause.

Algorithme 6 l'heuristique BeginEnd

Require: Trail of (mobility) traces M, break duration t

1: Initialize N has being the number of records in the trail of traces M ($M.length$)
2: **for** $i = 1$ to $N - 1$ **do**
3: $\quad timeElapsed = timeDifferenc(M[i].time, M[i+1].time)$
4: $\quad$ **if** $timeElapsed \geq t$ **then**
5: $\quad\quad$ Add the mobility traces $M[i]$ and $M[i+1]$ to the resulting set R
6: $\quad$ **end if**
7: **end for**
8: **return** R, the list of begin and end traces discovered

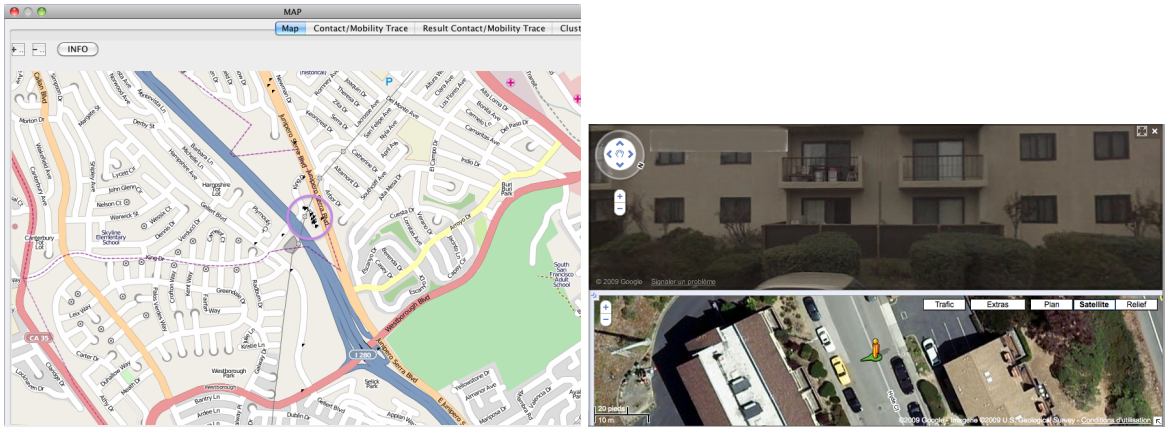


FIGURE 3.1 – Une attaque *BeginEnd* fructueuse.

Pendant cette étude, nous avons examiné les traces de 90 taxis, pris aléatoirement parmi les 500 disponibles. Pour 20 de ces 90 taxis, la visualisation sur une carte des résultats obtenus

après l'attaque *BeginEnd* pointe un quartier de petite taille qui correspond très probablement au quartier de leur domicile. En l'absence de l'adresse réelle des conducteurs de taxis, qui est fort heureusement absente des données, il ne nous est pas possible de valider formellement ces résultats. Toutefois, pour 10 des 90 taxis, le résultat pointait directement sur une adresse unique, ou sur une toute petite portion de rue, indiquant avec une forte probabilité que ces conducteurs allumaient leur GPS avant de démarrer leur véhicule et l'éteignaient juste avant d'en sortir. Cette adresse est très probablement celle du domicile des conducteurs. Pour confirmer cette intuition, nous avons trouvé dans Google StreetView un taxi jaune garé à l'adresse de l'un de ces 10 taxis, comme le montre la figure 3.1. Pour les autres taxis, l'attaque aura permis d'identifier le parking de la compagnie de taxis YellowCabs, ainsi que d'autres points d'intérêts de la ville de San Francisco, telle que ses stations de taxis.

3.1.2.2 Identification des points d'intérêt et modèles de mobilité markoviens

Dans [53], nous introduisons une forme de modèle de mobilité, que nous appelons chaîne de Markov de mobilité (*MMC* pour *Markov-Mobility Chain*). Une MMC peut représenter, de façon compacte et précise à la fois, la mobilité d'un individu. Une MMC peut être vue comme un automate probabiliste où chaque état représente un POI et où les transitions entre les états correspondent à un déplacement entre un POI et un autre. L'automate est probabiliste dans le sens où les transitions entre POIs ne sont pas déterministes mais, au contraire, il existe une distribution de probabilité entre les transitions qui quittent un POI.

Plus formellement, une MMC est composée de :

- *Un ensemble d'états* $P = \{p_1, \ldots, p_n\}$, dans lequel chaque état p_i représente un POI (ou un ensemble de POIs). Ces POIs peuvent, par exemple, être inférés par un algorithme de *clustering* œuvrant sur les traces de mobilité d'un individu. Chaque état est donc associé à une localisation spatiale, le POI. Pour la plupart des MMCs considérées ici, p_1 correspond au « domicile » de l'individu et p_2 à son lieu de travail. Il est donc également possible d'associer une étiquette sémantique à chaque état de la MMC. Les états sont ordonnés par importance décroissante du POI auquel ils sont associés. Le dernier état est souvent composé de ce que nous appelons les « POIs non fréquents », qui sont les POIs qui ont été visités régulièrement mais peu fréquemment.
- *Un ensemble de transitions*, $T = \{t_{1,1}, \ldots, t_{n,n}\}$, où chaque transition $t_{i,j}$ correspond au déplacement de l'état p_i à l'état p_j. Une probabilité est associée à chaque transition, elle représente la probabilité d'aller de l'état p_i à l'état p_j. Parfois, un individu peut quitter un POI pour aller à un endroit qui ne fait pas partie de ses POIs, puis revenir au POI original. Ce type de comportement est modélisé dans la MMC par une transition entre un état et lui-même. La somme des probabilités des transitions sortantes d'un état vaut 1 : $\sum_{j=1}^{n} t_{i,j} = 1$. Notons que la probabilité $t_{i,j}$ d'aller de p_i à p_j (pour $i \neq j$) est en général différente de la probabilité $t_{j,i}$ d'aller de p_j à p_i.

Une MMC peut être représentée sous la forme d'une *matrice de transitions* ou comme un *graphe dirigé* dans lequel les nœuds correspondent aux états et où il y a un arc dirigé et pondéré entre deux nœuds, si et seulement si la probabilité de transition entre ces deux nœuds est non nulle.

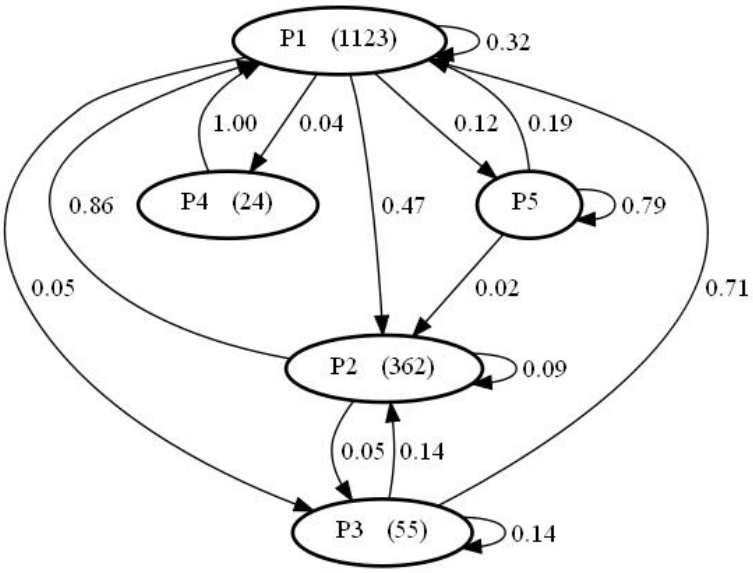


FIGURE 3.2 – Un exemple de chaîne de Markov de mobilité pour *Bob*.

Si l'on prend l'exemple d'un utilisateur dénommé *Bob*, qui a 4 POIs fréquent et un ensemble de POIs qui lui sont moins importants. On peut alors définir une MMC composée de 5 états, un pour chacun des 4 POIs fréquents et un dernier pour l'ensemble des autres POIs. Nous avons donc $P = \{p_1, p_2, p_3, p_4, p_5\}$ et les localisations (et/ou étiquettes sémantiques) associées à chaque état $L = \{l_1, l_2, l_3, l_4, l_5\}$. Supposons maintenant que nous ayons pu apprendre la MMC représentée par la figure 3.2 à partir des traces de mobilité de *Bob*. Ajoutons un poids à chaque état de la MMC. La description du calcul de ce poids sera présentée en détail dans la suite de cette section mais supposons pour l'instant que ce poids est relatif au temps passé dans chaque POI. Défini tel quel, ce poids donne donc une indication de l'importance du POI. Les états sont ordonnés par importance décroissante, sauf le POI correspondant à l'ensemble de POIs moins fréquents.

	P_1 - Domicile	P_2 - Travail	P_3 - Sport	P_4 - Loisirs	P_5 - Autres
P_1 - Domicile	0.321	0.469	0.049	0.037	0.124
P_2 - Travail	0.86	0.093	0.047	0.0	0.0
P_3 - Sport	0.714	0.143	0.143	0.0	0.0
P_4 - Loisirs	1.0	0.0	0.0	0.0	0.0
P_5 - Autres	0.2	0.02	0.0	0.0	0.78

FIGURE 3.3 – Chaîne de Markov de mobilité de *Bob* sous la forme de sa matrice de transition.

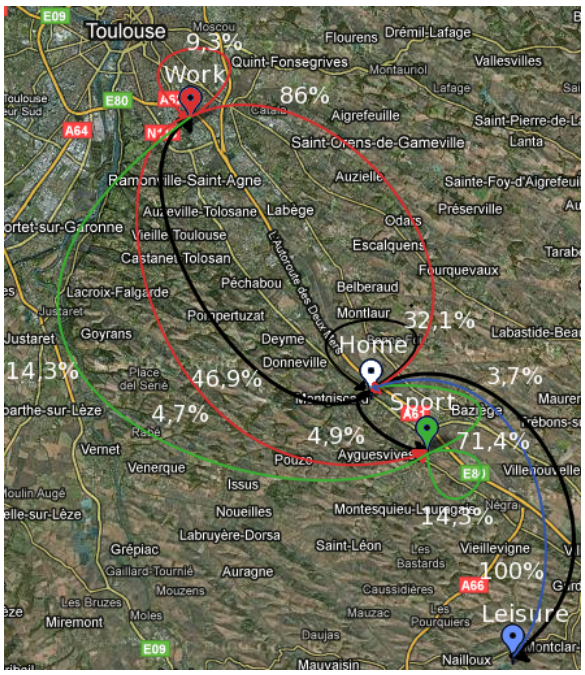


FIGURE 3.4 – Chaîne de Markov de mobilité de *Bob* sur une carte.

En observant la structure de la MMC, on peut se rendre compte que l'état p_1 est le seul qui peut être atteint à partir de chacun des autres. De plus, les transitions y menant ont une probabilité relativement élevée (sauf de l'état p_5). En associant ces différentes observations, on peut inférer avec un relativement bon niveau de confiance que cet état p_1 est probablement le "domicile" de *Bob*. Ensuite, si on veut essayer d'inférer le "lieu de travail" et en considérant que "les gens vont souvent de la maison au travail et vice-versa", on peut inférer que l'état p_2 est un bon candidat, ce qui est corroboré par le poids élevé de cet état. D'autres heuristique peuvent être utilisée pour continuer d'attacher des étiquettes sémantiques aux états p_3 et p_4. Par exemple, attachons l'étiquette "sport" à p_3 et "loisirs" à p_4. Si l'on oublie les poids des états, on peut alors représenter la MMC de *Bob* par la matrice de transition présentée en figure 3.3. En prenant en compte les localisations des POIs, on peut également représenter la structure de la MMC sur une carte réelle, voir la figure 3.4.

Pour la découverte des POIs, nous avons introduit dans [53] l'agorithme *Time-Density cluster* (TD cluster). Il s'agit d'un nouvel algorithme de clustering inspiré de *Density Time clustering* [63]. La motivation principale était d'aboutir à un algorithme aussi performant que ce dernier, tout en le rendant plus résistant à la distorsion (cf. la section 3.2.1 pour une discussion sur la distorsion des données). L'algorithme TD cluster est donné en algorithme 7. Il prend en entrée un rayon r, une fenêtre temporelle t, un taux de tolérance τ, un seuil de distance d et une suite de traces de mobilité M. L'algorithme commence par construire de façon itérative des clusters à partir des traces contenues dans M qui sont dans la même fenêtre temporelle t. Puis, pour chaque cluster, si une fraction des points (au dessus de τ) sont dans un rayon r du centroïde, le *cluster* est conservé. Dans le cas contraire, il est abandonné.

Finalement, l'algorithme fusionne les clusters dont les centroïdes (ou les médioïdes [1]) sont à une distance inférieure à d l'un de l'autre.

Algorithme 7 TD clustering

Require: Traces de mobilité M, fenêtre temporelle t, rayon r, taux de tolérance τ, seuil de distance d

1: $N = M.length$
2: $cumulTime = 0$
3: L = liste vide
4: C = cluster vide
5: **for** $i = 0$ to $N - 1$ **do**
6: $cumulTime = cumulTime + (M[i+1].time - M[i].time)$
7: **if** $cumulTime \leq t$ **then**
8: C.add($M[i]$)
9: **else**
10: $nbPtsOut = 0$
11: **for** $j = 0$ to $C.nbPts$ **do**
12: **if** $distance(C[j], C.centroid) > r$ **then**
13: $nbPtsOut = nbPtsOut + 1$
14: **end if**
15: **end for**
16: **if** $nbPtsOut/N < \tau$ **then**
17: Add the cluster C to L
18: **end if**
19: $cumulTime = 0$
20: C = cluster vide
21: **end if**
22: **end for**
23: Fusionner les clusters de L dont la distance entre centroïdes est $\leq d$
24: **return** L, la liste des POIs découverts

L'algorithme 8 décrit l'algorithme que nous proposons pour apprendre une MMC. L'algorithme commence (ligne 1) par appliquer un algorithme de clustering (TDCluster, cf. algorithme 7) aux traces de mobilité afin d'identifier les clusters de localisations importantes. Ensuite (ligne 2), afin de réduire le nombre de clusters, l'algorithme fusionne les clusters dont les centroïdes sont à une distance inférieure à d l'un de l'autre. Cette fusion n'est pas cumulative, les distances entre centroïdes sont calculées au préalable et tous les clusters d'un même groupe

1. Le centroïde est le centre géométrique de la zone délimitée par un cluster. Le médioïde est le point du cluster le plus proche de son centroïde. La plupart des algorithmes donnés dans ce manuscrit peuvent fonctionner avec les centroïdes ou les médioïdes. En général l'utilisation des médioïdes donnent de meilleurs résultats puisqu'il s'agit de points réels. Le calcul des médioïdes est cependant bien plus coûteux, $O(n^2)$.

Algorithme 8 apprentissage de MMC

Require: Traces de mobilité M, seuil de distance d, seuil de vitesse ϵ, seuil temporel $mintime$

listPOIs = TDCluster(M)
Fusionner les clusters de listPOIs qui sont à une distance $\leq d$ l'un de l'autre
for chaque cluster C dans $listPOIs$ **do**
  Calculer $interval\_temporel$ et $densite$ de C
  if $C.time\_interval > mintime$ **then**
    $freqPOIs$.add(C)
  else
    $infreqPOIs$.add(C)
  end if
end for
Trier $freqPOIs$ par densité décroissante
for chaque cluster dans $freqPOIs$ **do**
  Créer un état p_i dans la MMC
end for
Créer un état p_n représentant tous les clusters de $infreqPOIs$
Calculer M' : les traces de mobilité issues de M dont la vitesse, estimée par rapport à la précédente, est $\geq \epsilon$
for chaque trace t de M' **do**
  $closestPOI = t.closestPOI(freqPOIs)$
  if $t.distance(closestPOI) \leq d$ **then**
    t.addLabel("p_i")
  else
    t.addLabel("unknown")
  end if
end for
Concaténer toutes les traces successives qui portent le même label
Calculer toutes les probabilités de transition entre chaque paire d'état de la MMC
return MMC

sont fusionnés en même temps. Chaque cluster résultant est considéré comme un POI, le centroïde peut alors être considéré comme la localisation du POI. On calcule pour chaque cluster (ligne 4) la *densité* (le nombre de traces de mobilité dans le cluster) et l'*interval_temporel*, mesuré en jours, entre la première et la dernière trace du cluster. On sépare ensuite les POIs en deux catégories : les *POIs fréquents* qui correspondent aux POIs dont l'*interval_temporel* est supérieur ou égal à un seuil $mintime$ et les *POIs infréquents* dont l'*interval_temporel* est inférieur à ce seuil (lignes 5 à 9). On trie (ligne 11) l'ensemble *frequent POIs* par ordre de densité décroissante. Le premier POI sera donc le plus dense et le dernier le moins dense.

On peut alors construire la MMC en créant un état pour chaque POI de l'ensemble *frequent POIs* (lignes 12 à 14) ainsi qu'un dernier état représentant l'ensemble des POIs moins fréquents *infrequent POIs* (ligne 15). Chaque état est ensuite associé à un poids correspondant à sa densité calculée (ligne 4). Ensuite (ligne 16), on expurge les traces de mobilité de tous les points correspondants à un déplacement, calculé à partir de la trace précédente, dont la vitesse est supérieure à ϵ. On parcourt ensuite (lignes 17 à 24) l'ensemble M' de traces résultant de l'opération précédente afin de les affecter à un POI si elles sont à une distance inférieure à d d'un POI de *frequent POIs*, ou de les étiqueter « unknown » sinon. A partir de cet étiquetage, on peut extraire la séquence des localisations (POIs) qui ont été visitées par l'individu. Les traces successives partageant la même étiquette sont concaténées (ligne 25). A partir de cette séquence de localisations, on peut alors calculer les probabilités de transition entre chaque paire d'état en comptant les transitions entres POIs et en normalisant ces probabilités (ligne 26). Une séquence de la forme « $p_i \Rightarrow$ *unknown* $\Rightarrow p_i$ » est considérée comme une transition de p_i à lui-même.

Nous sommes donc capables, à partir des traces de mobilité d'un individu, de construire un modèle de sa mobilité sous la forme d'une MMC. Une telle MMC est un modèle à la fois compact, précis, lisible et versatile de la mobilité individuelle. Dans les sous-sections suivantes, nous explorons cette versatilité en utilisant les MMCs (ou certaines variantes) comme base de plusieurs attaques de geo-privacy.

3.1.2.3 Prédiction des mouvements

Dans [55], nous proposons une variante de l'algorithme 8 pour la construction de MMC afin que chaque état de la chaîne de Markov représente les n précédents POIs visités. Nous appelons cette variante les *nMMCs*. Nous proposons un algorithme de prédiction de la prochaine localisation d'un individu pour lequel nous disposons de la nMMC. Les performances de cet algorithme ont été évaluées sur 3 ensembles de données différents et démontrent une efficacité de 70% à 95% de la prédiction dès que $n \geq 2$. Ces travaux font l'objet de cette section.

L'algorithme 9 présente l'algorithme d'apprentissage d'une nMMC. Cet algorithme diffère de l'algorithme 8 par différents aspects. Tout d'abord, nous avons abandonné les POIs non fréquents, qui n'avaient que peu de sens dans un contexte où $n \geq 2$. Seuls les POIs

fréquents sont mémorisés (ligne 6). Un état de la nMMC est créé pour chaque paire de POI (ligne 10). Enfin, l'étiquetage des traces de mobilité n'impacte pas seulement la localisation courante, mais aussi les n-1 localisations suivantes (ligne 16 à 20). Le calcul des probabilités de transition n'est pas modifié.

Algorithme 9 apprentissage de n-MMC

Require: Traces de mobilité M, seuil de distance d, seuil de vitesse ϵ, seuil temporel $mintime$, nombre de localisations mémorisées n

1: listPOIs = TDCluster(M)
2: Fusionner les clusters de listPOIs qui sont à une distance $\leq d$ l'un de l'autre
3: **for** chaque cluster C dans $listPOIs$ **do**
4: Calculer $interval\_temporel$ et $densite$ de C
5: **if** $C.time\_interval > mintime$ **then**
6: $freqPOIs$.add(C)
7: **end if**
8: **end for**
9: Trier $freqPOIs$ par densité décroissante
10: **for** chaque paire de cluster dans $freqPOIs$ **do**
11: Créer un état p_i dans la MMC
12: **end for**
13: Calculer M' : les traces de mobilité issues de M dont la vitesse, estimée par rapport à la précédente, est $\geq \epsilon$
14: **for** chaque trace t de M' **do**
15: $closestPOI = t.closestPOI(freqPOIs)$
16: **if** $t.distance(closestPOI) \leq d$ **then**
17: Mettre à jour l'étiquette de la localisation courante et des $n-1$ précédentes avec $closestPOI$
18: Etiqueter la trace t avec la localisation courante et les $n-1$ précédentes
19: **else**
20: Étiqueter la trace t avec "unknown"
21: **end if**
22: **end for**
23: Concaténer toutes les traces successives qui portent le même label
24: Calculer toutes les probabilités de transition entre chaque paire d'état de la MMC
25: **return** MMC

Afin de prédire la localisation basée sur les n précédentes localisations, nous calculons alors la matrice de transition de la n-MMC. Prenons un exemple où $n = 2$ et où l'algorithme

Source/Dest.	H	W	L	O
H W	1,00	0,00	0,00	0,00
H L	1,00	0,00	0,00	0,00
H O	0,66	0,34	0,00	0,00
W H	0,00	0,84	0,08	0,08
L H	0,00	0,50	0,00	0,50
O H	0,00	1,00	0,00	0,00
O W	1,00	0,00	0,00	0,00

TABLE 3.1 – Exemple de matrice de transition d'une 2-MMC.

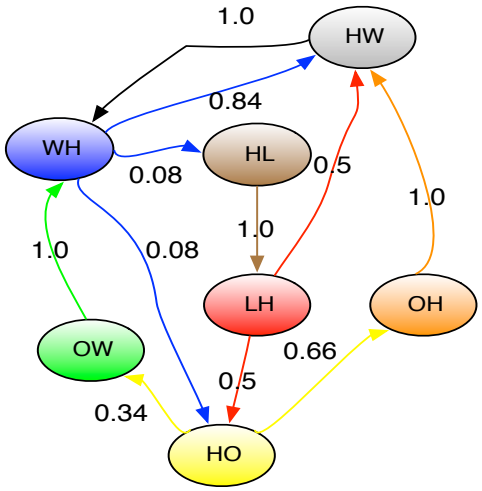


FIGURE 3.5 – Exemple de représentation graphique d'une 2-MMC.

de clustering a donné 4 POIs : « Home » (H), « Work » (W), « Leisure » (L) et « Opera » (O). La 2-MMC est donc, a priori, composée de 12 ($4 * 3$) états qui représentent chacun la visite en séquence de 2 POIs distincts : HW, HL, HO, WH, WL, WO, LH, LW, LO, OH, OW et OL. La matrice de transition est donc composée de 12 lignes (1 ligne par état de la 2-MMC) qui représentent l'état de départ et de 4 colonnes (1 colonne par POI) qui représentent le POI d'arrivée. La table 3.1 et la figure 3.5 donnent respectivement un exemple de la matrice de transition et de la représentation graphique d'une 2-MMC obtenue à partir des traces de mobilité d'un utilisateur du jeu de données Phonetic [85]. Toutes les paires de POIs ne sont pas représentées car certains cas de figure ne sont jamais survenus dans le jeu de données, par exemple la séquence « Work-Leisure » (WL).

L'algorithme de prédiction 10 est alors très simple. Il prend en entrée les n précédentes positions et la matrice de transition de la n-MMC et retourne le POI le plus probable à partir de la séquence des n précédentes localisations fournie en entrée. En cas d'égalité, le POI est choisi aléatoirement parmi les POIs les plus probables. Par exemple, pour la matrice de

transition donnée Table 3.1, si l'individu est en « HO », l'algorithme renvoie « H » qui est le POI le plus probable avec 66% des cas.

Algorithme 10 Prédiction de la prochaine localisation à partir d'une n-MMC

Require: Matrice de transition M d'une n-MMC, n précédentes localisations

1: Recherche dans M de la ligne l correspondant aux n précédentes localisations.
2: Recherche de la colonne c avec la plus forte probabilité de transition p_{max} dans la ligne l de M.
3: **if** Plusieurs colonnes avec p_{max} **then**
4: $nextPredictedPOI$=random(colonnes avec p_{max})
5: **else**
6: $nextPredictedPOI$=colonne avec p_{max}
7: **end if**
8: **return** $nextPredictedPOI$

Dans [55], nous avons évalué cet algorithme de prédiction sur deux jeux de données : *Geolife* [144] et *Phonetic* [85]. Geolife a été collecté par des chercheurs de Microsoft Asie, d'avril 2007 à octobre 2011, essentiellement dans la région de Shanghai, pour 175 utilisateurs. Phonetic, quant a lui, est composé de traces de mobilité issues de 6 chercheurs collectées d'octobre 2009 à janvier 2011. Notre évaluation consiste essentiellement à calculer la *précision* Acc et la prévisibilité $Pred$. La précision est le ratio entre le nombre de prédictions correctes $p_{correct}$ et le nombre de prédictions totales p_{total} :

$$Acc = p_{correct}/p_{total} \tag{3.1}$$

La prévisibilité $Pred$ est une mesure théorique qui représente le degré avec lequel les mouvements d'un individu sont prévisibles à partir de sa n-MMC, à la façon des travaux de Song et *al.* dans [128]. La prévisibilité $Pred$ d'une n-MMC est calculée comme le produit scalaire entre le vecteur stationnaire de la n-MMC et le vecteur des probabilités maximales de chaque ligne de la n-MMC :

$$Pred = \langle VStat \mid VPmax \rangle \tag{3.2}$$

Dans les évaluations de l'algorithme 10, nous avons séparé chaque suite de traces de mobilité en deux ensembles : l'*ensemble d'apprentissage*, qui est utilisé pour construire la n-MMC, et l'*ensemble de test*, qui est utilisé pour évaluer la précision de la prédiction. La figure 3.6 montre les résultats obtenus avec Phonetic pour $1 \leq n \leq 4$ et avec GeoLife pour $1 \leq n \leq 3$. Le succès de l'attaque (*Acc testing set* en losanges bleus) culmine autour de 70% pour GeoLife (figure de droite) et 96% pour Phonetic (figure de gauche). On remarquera que la mesure de la précision effectuée sur l'ensemble d'apprentissage (*Acc training set* en carrés rouges) donne des résultats supérieurs, ce qui était attendu, mais pas de façon très sensible, notamment dans le cas de Phonetic. On peut conclure que les individus des jeux de données n'ont pas particulièrement changé leurs motifs de mobilité entre les périodes d'apprentissage et de test.

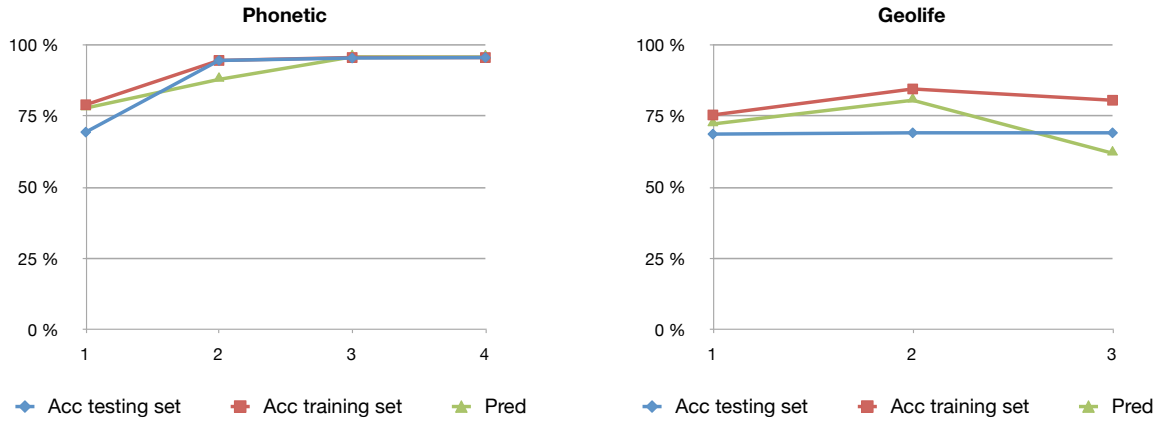


FIGURE 3.6 – Précision de l'attaque de prédiction de la prochaine localisation.

Toujours pour Phonetic, on remarquera que la courbe de précision rejoint la courbe de prévisibilité, ce qui n'est pas le cas des mesures de Geolife. C'est probablement dû à une mauvaise paramétrisation du clustering pour le second jeu de données, les individus représentés dans celui-ci semblent avoir un comportement plus chaotique, et donc plus difficile à modèliser et a fortiori à prédire. Remarquons que pour Phonetic, la précision augmente sensiblement (de 69 à 94%) avec $n = 2$ puis se stabilise. A l'inverse, pour Geolife, elle ne croît pas avec n. Ce comportement s'explique par le fait qu'avec de plus grandes valeurs de n, la n-MMC représente des cas qui ne surviennent que très occasionnellement et perd de sa généralité. Une piste de travaux futurs concerne justement une version de n-MMC où n est adaptatif, cf. section 3.1.3.

Les résultats présentés ici sont cohérents avec ceux obtenus dans [128] où les auteurs concluaient à une prévisibilité moyenne de 93% d'après l'analyse de l'entropie dans un jeu de données issu d'un opérateur de téléphonie mobile américain. La prédiction de la localisation d'un individu est un sujet particulièrement chaud. Pour preuve, Nokia a organisé en 2012 une compétition dont l'une des taches était justement de fournir un prédicteur de localisation [95]. Il faut noter toutefois que pour faire leur prédiction, les participants à la compétition ne disposaient que du jour et de l'heure à laquelle l'individu initiait son déplacement, et non pas de sa précédente localisation. Cela rend la tâche bien plus difficile pour la prédiction. Pour preuve, les gagnants de cette tâche particulière [42] n'ont obtenu qu'un taux de succès de 56,2%. Nous pensons toutefois que la prise en compte de la temporalité des déplacements constitue probablement un vecteur d'amélioration de notre prédicteur. C'est la raison pour laquelle nous avons commencé à travailler sur des MMC temporisées [54] où le temps est décomposé en tranches et où une MMC (ou une n-MMC) est produite pour chaque tranche temporelle.

3.1.2.4 Désanonymisation

Dans [56], nous proposons une nouvelle méthode basée sur l'utilisation des MMCs pour dés-anonymiser des données géolocalisées. Une MMC est construite à partir des traces de mobilité observées durant la phase d'apprentissage puis est utilisée pour l'attaque de désanonymisation dans la phase de test. Plus précisément, que ce soit dans la phase d'apprentissage ou dans la phase de test, la mobilité de chaque individu est représentée par une MMC. Ensuite, une mesure de distance est calculée entre les MMCs de la phase d'apprentissage et celles de la phase de test afin d'identifier les individus les plus proches en termes de mobilité. En substance, la mobilité d'un individu est utilisée comme une signature de celui-ci, jouant le rôle de quasi-identifiant. Avec cette méthode, si l'attaquant connaît *Alice* et une signature de sa mobilité (en l ?occurrence une MMC apprise précédemment), il peut essayer de l'identifier dans une base de données de mobilité anonymisée (ou pseudonymisée) en cherchant dans celle-ci une signature correspondante.

Dans la suite de cette section, nous décrivons trois métriques d'évaluation de similarité entre MMCs : la distance stationnaire et la distance de proximité ainsi qu'une combinaison de ces deux métriques, la distance *Stat-Prox*.

L'intuition derrière la *distance stationnaire* est que la distance entre deux MMCs correspond à la somme des distances entre les POIs les plus proches de chacune des MMCs. Afin de calculer la distance stationnaire, les états des MMCs sont appairés de façon à minimiser cette somme. De ce fait, il est possible qu'un état d'une MMC soit appairé avec plusieurs états de la seconde MMC, en particulier si les deux MMCs n'ont pas le même nombre d'états. L'algorithme 11 présente le calcul de la distance stationnaire entre deux MMCS M_1 et M_2. Tout d'abord, on calcule les vecteurs stationnaires des MMCs (ligne 1). Ensuite, pour chaque POI p_i de M_1 (lignes 3 à 11), l'algorithme cherche le POI de M_2 le plus proche de p_i, au sens de la distance euclidienne. Cette distance multipliée par la probabilité en régime stationnaire de p_i est ajoutée à la distance stationnaire accumulée (ligne 12). La valeur calculée représente la distance de M_1 à M_2. Cette distance n'étant pas symétrique, la distance de M_2 à M_1 est donc également calculée et la moyenne de ces deux distances représente la distance stationnaire symétrique, cf. algorithme 12.

L'intuition derrière la *distance de proximité* est que deux MMCs doivent être considérées comme très proches si elles partagent des POIS *importants*, tels que le domicile ou le lieu de travail. De plus, l'influence d'un POI dans le calcul de la distance doit être proportionnelle à son importance, c'est-à-dire à son poids. On commence donc par comparer les états les plus importants, ordonnés par poids décroissants, puis les deuxièmes, ainsi de suite. Le score de proximité obtenu si les deux premiers POIs sont communs est deux fois plus important que celui obtenu pour partager les deuxièmes POIs, qui lui même est le double de celui obtenu pour les troisièmes, etc. L'algorithme 13 présente le calcul de la distance de proximité entre deux MMCs M_1 et M_2 selon deux paramètres : $Rank$ le score obtenu pour le partage du premier POI (score qui est ensuite divisé par deux pour chaque saut dans la liste des POIs

Algorithme 11 $DistanceStationnaireAsymetrique(M_1, M_2)$

Require: MMCs M_1 et M_2

Calculer V_1 et V_2 les vecteurs stationnaires de respectivement M_1 et M_2
$CumulatedDistance = 0$
for i=1 to $|M_1|$ **do**
  $MinDistance = \infty$
  p_i = le i^{eme} POI de M_1
  for j=1 to $|M_2|$ **do**
    p_j = le j^{eme} POI de M_2
    if DistanceEuclidienne(p_i,p_j) $< MinDistance$ **then**
      $MinDistance$ = DistanceEuclidienne(p_i,p_j)
    end if
  end for
  $CumulatedDistance = CumulatedDistance + V_1[i] \times MinDistance$
end for
return $CumulatedDistance$

Algorithme 12 $DistanceStationnaire(M_1, M_2)$

Require: MMCs M_1 et M_2

$D_1 = DistanceStationnaireAsymetrique(M_1, M_2)$
$D_2 = DistanceStationnaireAsymetrique(M_2, M_1)$
$D = (D_1 + D_2)/2$
return D

comparés, ligne 8) et Δ la distance en dessous de laquelle deux POIs sont considérés comme identiques. L'algorithme compare un à un chaque couple POI de M_1 et M_2 (lignes 2 à 9), vérifie si leur distance euclidienne est inférieure à Δ (ligne 5) et si c'est le cas augmente le score de la valeur de $Rank$ (ligne 6). La distance de proximité est ensuite simplement l'inverse du score (ligne 11) : plus le score de proximité est grand, plus les MMCs sont proches et plus la distance est faible.

Algorithme 13 $DistanceProximite(M_1, M_2)$

Require: MMCs M_1 et M_2, $Rank$ le score obtenu pour le partage du POI le plus important, Δ la distance maximale pour partager deux POIs

1: $Score = 0$
2: **for** $k = 1$ to $min(|M_1|, |M_2|)$ **do**
3: p_i = le k^{eme} POI de M_1
4: p_j = le k^{eme} POI de M_2
5: **if** DistanceEuclidienne(p_i,p_j) $< \Delta$ **then**
6: $Score = Score + Rank$
7: **end if**
8: $Rank = Rank/2$
9: **end for**
10: **if** Score > 0 **then**
11: $Distance = 1/Score$
12: **else**
13: $Distance = \infty$
14: **end if**
15: **return** $Distance$

La distance de proximité, à l'inverse de la distance stationnaire qui est une distance géographique, est, elle, une distance symbolique entre deux MMCs et est basée sur la sémantique qui est attachée aux POIs. En effet, le premier POI d'une MMC est considéré comme très représentatif, comme le domicile, le second comme plutôt représentatif, comme le lieu de travail, et deux individus sont considérés comme très similaires s'ils partagent ces POIs. Nous proposons d'exploiter cette diversité dans le calcul des distances afin d'augmenter l'efficacité de la dés-anonymisation, c'est l'objet de l'heuristique *StatProx*. Cette heuristique considère qu'il existe des seuils à partir desquels il vaut mieux choisir la distance de proximité ou la distance stationnaire. L'algorithme 14 détaille ce mode de fonctionnement.

Afin d'évaluer les performances de ces trois distances entre MMCs pour conduire une attaque en dés-anonymisation, nous avons travaillé sur trois ensemble de données : les données GeoLife et Phonetic déjà décrites et utilisées en section 3.1.2.3 ainsi que le jeu de données *Nokia* qui est issu d'une campagne qui a eu lieu dans la ville de Lausanne sur 200 utilisateurs [88]. Lors de ces évaluations, nous séparons les traces de données des individus en deux

Algorithm 14 $DistanceStatProx(M_1, M_2)$

Require: MMCs M_1 et M_2, Seuils S_{stat}, S_{prox}

$R_{prox} = DistanceProximite(M_1, M_2)$
$R_{stat} = DistanceStationnaire(M_1, M_2)$
if $R_{prox} < S_{prox}$ **then**
  return R_{prox}
else
  if $R_{stat} < S_{stat}$ **then**
    return R_{stat}
  else
    if $R_{prox} < \infty$ **then**
      return R_{prox}
    else
      return R_{stat}
    end if
  end if
end if

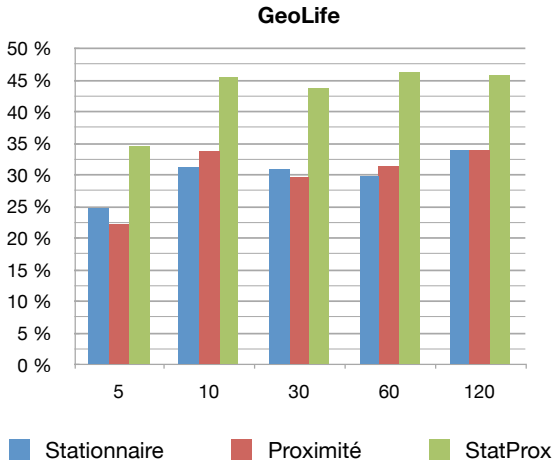


FIGURE 3.7 – Efficacité de la dés-anonymisation sur GeoLife pour différents taux d'échantillonnage.

ensembles, l'ensemble d'apprentissage et celui de test, comme nous l'avons fait dans les sections précédentes. L'ensemble d'apprentissage représente les connaissances de l'adversaire qui essaie de dés-anonymiser les individus de l'ensemble de test en les reliant à leur contrepartie dans l'ensemble d'apprentissage. Le taux de succès mesure donc les liens correctement établis entre les individus de l'ensemble d'apprentissage et celui de test et est normalisé par le nombre d'individus dans l'ensemble de test, qui correspond au nombre de tentatives de désanonymisation.

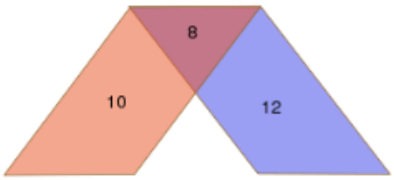


FIGURE 3.8 – Diagramme de Venn pour les individus correctement dés-anonymisés par distance stationnaire (à gauche), de proximité (à droite), en commun (au centre) et *StatProx* (total).

La figure 3.7 présente les résultats obtenus par les trois mesures de distances sur les données GeoLife et ce avec différents taux d'échantillonnage : 5 (les données d'origine), 10, 30, 60 et 120 secondes. On peut y observer que la distance *StatProx* surclasse les deux autres distances d'environ 10 points. Ceci provient du fait que chacune des mesures de distance « simple » désanonymise généralement des individus différents, comme le montre la figure 3.8. StatProx permet donc de bénéficier des désanonymisations fructueuses des deux autres distances. Son taux de succès culmine à environ 35% (81 individus) sans échantillonnage et va jusqu'à environ 45% (de 59 à 77 individus) avec échantillonnage.

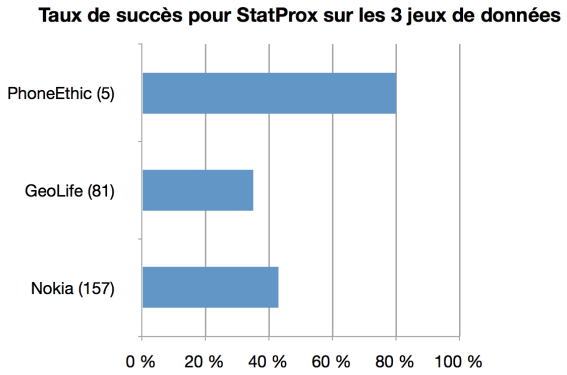


FIGURE 3.9 – Efficacité de la désanonymisation avec *StatProx* sur les différents jeux de données.

La figure 3.9 présente les résultats obtenus grâce à StatProx sur les trois jeux de données sans échantillonnage. Appliqué au jeu de données *Nokia*, l'attaque offre un taux de succès de 42% parmi les 157 individus dont il est possible de construire une MMC (195 individus au total dans le jeu de données). En ce qui concerne le jeu de données *Phonetic*, le taux monte à 80% mais sur la base de 5 utilisateurs seulement (sur les 7 individus d'origine, seules 5 MMCs peuvent être construites). Pourtant, une caractéristique importante du jeu de données *Phonetic* est que les individus sont tous de la même institution, travaillent dans le même laboratoire, et habitent la même métropole. Ils sont donc potentiellement plus difficiles à distinguer car leurs modèles de mobilité et leurs POIs sont très similaires.

La figure 3.10 est inspirée du concept de courbe ROC (*Receiver Operating Characteristics*). Une courbe ROC représente la sensibilité (le taux de vrais positifs vis-à-vis du taux de

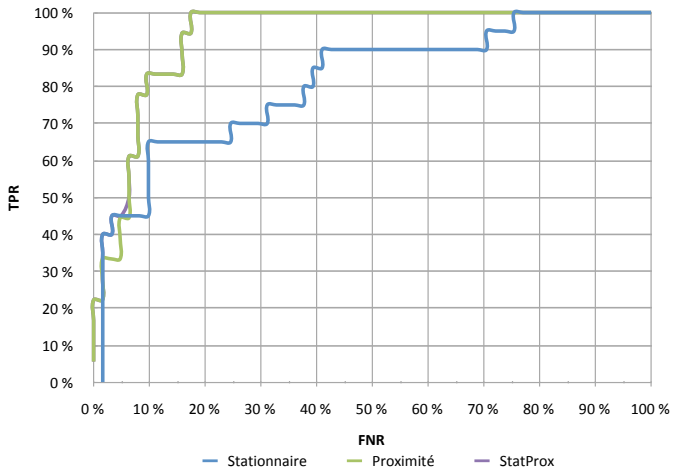


FIGURE 3.10 – Courbe « ROC » de la désanonymisation avec les trois distances sur les données GeoLife.

vrais négatifs) d'un système de classification. L'idée étant qu'entre deux classifieurs, il vaut mieux privilégier celui qui réussit plus rapidement (lorsque les candidats sont triés par score de confiance, ce qui correspond dans notre cas à un classement par distance décroissante). La forme de courbe ROC que nous obtenons présente le taux de vrais positifs (*True Positive Rate* - TPR) en fonction du taux de vrais négatifs (*True Negative Rate* - TNR) avec les candidats triés par distance décroissante, puisqu'une faible distance implique une confiance plus grande dans la prédiction. Cette courbe ROC confirme que *StatProx* offre de bonnes performances car il permet, grâce aux seuils de configuration, de calquer ses performances sur celles de la distance la plus performante, que ce soit la distance stationnaire ou de proximité.

Dans [39], les auteurs proposent une attaque similaire à celle conduite ici. Leurs travaux se basent sur des traces de mobilité dans un réseau GSM [41] qui contiennent notamment les identifiants des cellules GSM auxquelles se sont connectés les téléphones portables d'une centaine d'individus pendant les 9 mois de l'année scolaire 2004-2005. Sur la base de ces données, les auteurs construisent des modèles Markoviens de la mobilité (en termes de cellules GSM) des individus du jeu de données. Ils effectuent cet apprentissage sur une *période d'identification* puis tentent de retrouver chaque individu dans une *période antérieure à celle d'identification*, de la même manière que nous construisons nos ensembles d'apprentissage et de test. Le taux de succès de leur attaque est d'environ 39%, ce qui est relativement similaire à celui que nous obtenons. Ils développent également un autre modèle, basé sur la séquence des cellules visitées, qui obtient de meilleurs résultats mais qui n'est pas comparable à nos travaux. En effet, les cellules GSM sont les mêmes pour tous les utilisateurs, alors que les POIs appris par notre méthode sont spécifiques à chaque individu.

Ma *et al.* [107] proposent également une attaque de dés-anonymisation ainsi qu'un métrique pour quantifier la perte d'information privée d'un individu. Deux ensembles de données ont été utilisés dans ces travaux : celui des taxis de San Francisco que nous avons utilisé pré-

cédemment [119], un jeu de données issues des bus publics de la ville de Shanghai qui semble ne plus être disponible. Deux types d'attaquants sont définis : l'un *passif* qui se borne à collecter des données dans des bases de données publiques et l'autre *actif* qui peut participer ou perturber la collecte des données afin d'augmenter sa base de connaissance. Les différentes méthodes proposées dans [107] donnent d'excellents résultats avec un taux de succès entre 80 et 90%. Il est par contre essentiel de noter que dans ces travaux, les auteurs n'ont pas séparés les ensembles d'apprentissage et de test. À l'inverse, les attaquants peuvent disposer de la connaissance totale des données à ré-identifier, ce qui facilite énormément la chose. Dans les mêmes conditions, notre attaque obtient un taux de succès de 98,23% soit 111 individus sur 113 (sur un jeu de données différent du leur puisqu'il s'agit de GeoLife).

3.1.3 Conclusions et pistes de recherche

Dans cette section, nous avons présenté un ensemble de différentes attaques de geoprivacy, qui obtiennent des résultats à la hauteur de ceux obtenus dans l'état de l'art, et qui reposent sur les chaînes de Markov de Mobilité comme outil commun pour la représentation de la mobilité individuelle. Les MMCs semblent donc représenter un outil cohérent, compact, précis, lisible et versatile pour représenter la mobilité d'un individu, voire même pour servir de quasi-identifiant. Plusieurs pistes de nouvelles attaques restent à explorer : sémantique des mouvements, inférence des liens sociaux, etc. Ceci constitue une partie de notre prospective. Un autre aspect de notre prospective consiste à enrichir les MMCs : avec un caractère temporel, comme nous l'avons initié dans [54] ou avec mémoire des précédentes localisations avec n (le nombre de localisations mémorisées) variable en fonction des fenêtres temporelles afin d'optimiser la prévisibilité dans chaque fenêtre. Bien évidemment, nous ne nous contentons pas d'élaborer des attaques mais prenons à cœur de proposer des solutions pour la fourniture de services geolocalisés respectueux de la vie privée. Ceci fait l'objet de la section suivante.

3.2 Protection de la geoprivacy

Thèse : Traore, en cours - Publication majeure : [53] - Autres publications : [52] [6] [50]

Au delà du travail sur les attaques en geoprivacy, relatées dans la section 3.1, nous étudions également les différents moyens de protection possibles. Ces travaux sont architecturés autour de trois axes :

- *Évaluation des risques* : en s'inspirant des méthodes d'évaluation des risques de sécurité, nous définissons et appliquons un processus d'évaluation des risques en termes de vie-privée qui se base sur la définition des cas d'usages du système issus d'un modèle d'interaction décrit en UML. Le résultat de cette analyse consiste en l'identification de scénarios d'incidents, des différents items d'information à protéger (*assets*), des menaces et vulnérabilités ainsi que leur potentiel impact sur les informations privées. Ces travaux en sont à un stade très préliminaires et ne sont donc pas détaillés dans ce manuscrit. Ils sont toutefois mentionnés dans notre prospective, cf. section 3.3.
- *Méthodes de protection des données* : nous l'avons vu, les ensembles de données géolocalisées représentent une source d'information très sensible quant au respect de la vie privée. Dans la section 3.2.1, nous relatons nos travaux relatifs aux différentes méthodes d'assainissement de ces données. Nous décrivons également dans la section 3.2.2 l'outil GEPETO (pour *GEoPrivacy Enhancing Toolkit)* qui intègre à la fois les attaques présentées en section 3.1 et les méthodes d'assainissement présentées dans la section 3.2.1.
- *Algorithmes de services géolocalisés respectueux de la vie privée* : plutôt que de faire confiance à des systèmes centralisés pour fournir des services géolocalisés (pages jaunes, découverte de voisinage, etc.), au risque d'une collecte de grande ampleur de données sensibles, nous abordons en section 3.2.3 différents algorithmes (algorithmes distribués, cryptographiques) pour la fourniture de tels services de façon respectueuse de la vie privée.

3.2.1 Protection des données géo-localisées

Nous avons montré, en section 3.1, que les techniques de protection des données géolocalisées basées simplement sur la pseudonymisation des données n'étaient pas efficaces puisque les données elles-mêmes peuvent servir de pseudo-identifiant. Pour protéger des données, un processus d'assainissement (ou /textitsanitization), qui ajoute de l'incertitude aux données et/ou enlève certaines informations sensibles, doit être utilisé. Cette perte de données (ou de précision dans les données) pose un dilemme : elle protège mieux la vie privée mais au coût d'une moindre utilité des données due à la dégradation de celles-ci. Il s'agit donc d'évaluer le compromis entre protection et utilité des données. Dans cette section, nous présentons les grandes familles de techniques pour la protection de la vie privée dans les données géolocalisées [53].

Un *algorithme d'assainissement* S prend un jeu de données géolocalisées D en entrée, introduit de l'incertitude et retire de l'information de celui-ci afin de mieux protéger la vie-privée des individus contenus dans ce jeu de données. S produit un jeu de données assaini D' à partir de D. L'idée principale est qu'il est ensuite plus difficile à un attaquant d'obtenir des informations sensibles à partir de D' qu'à partir de D. Une procédure d'assainissement apporte en général des garanties en terme de respect de la vie privée. Elle peut garantir, par exemple, qu'à chaque pas temporel, il existe un nombre minimum d'individus k dans chaque zone spatiale (notion de *k-anonymat* [132]). Les différentes familles de techniques d'assainissement sont les suivantes :

- La *pseudonymisation* (voir par exemple [14, 11]) consiste à remplacer l'identifiant commun à plusieurs traces de mobilité soit par un pseudonyme généré aléatoirement (ce qui offre théoriquement l'anonymat mais pas la non-chaînabilité des données), soit par désidentification complète [131, 18, 112] (offrant donc théoriquement anonymat total et non-chaînabilité des données). [2] La pseudonymisation est en général effectuée préalablement à un processus d'assainissement mais n'est, bien souvent, pas suffisante pour protéger la vie-privée des individus.
- Les méthodes de *perturbation* (voir par exemple [5, 93, 96]) modifient les coordonnées spatiales d'une trace de mobilité en y ajoutant des perturbations aléatoires. Ce bruit peut, par exemple, être généré uniformément ou par une Gaussienne dans un rayon r centré sur les coordonnées originales. Si la géographie des environs n'est pas prise en compte, le résultat de la perturbation peut ne plus avoir aucun sens physique (en étant situé, par exemple, dans un fleuve ou sur une falaise).
- L'*agrégation* (voir par exemple [31, 19]) mêle plusieurs traces de mobilité en une seule coordonnée spatiale. Cette coordonnée peut, par exemple, être une zone plus large, telle qu'un quartier, ou être encore une moyenne des coordonnées d'origine. Un algorithme de *clustering* peut être utilisé pour regrouper ensemble les traces proches et ainsi servir de base pour le processus d'agrégation. Une autre possibilité est donc d'identifier les traces qui occupent une même zone spatiale et de les remplacer par les coordonnées de cette zone (comme un quartier, une rue, une ville, un pays).
- L'*échantillonnage* peut être vu comme une forme d'agrégation temporelle. Un mécanisme d'échantillonnage fusionne plusieurs traces de mobilité en un nombre moindre de traces. Les traces résultantes représentent un ensemble de traces qui ont eu lieu dans une même fenêtre temporelle. Elles peuvent être la médiane ou la moyenne des traces d'origine. Le processus d'échantillonnage offre en outre l'avantage de réduire le nombre de

2. L'*anonymat* peut être défini comme la capacité d'effectuer une action particulière sans avoir à révéler son identité alors que la *non-chaînabilité* est une notion plus forte qui implique de ne pas pouvoir relier différentes actions qui ont été effectuées par le même individu. Par exemple, le fait d'effectuer certaines actions sous un pseudonyme (plutôt que sous son identité réelle) offre l'anonymat mais pas la non-chaînabilité. Se référer à [118] pour de plus amples détails.

traces, en quelque sorte de *compresser* le jeu de données, ce qui induit un coût moindre pour d'autres techniques d'assainissement (voire, d'ailleurs, pour les attaques).

- Le *cloaking (spatial ou temporel)* [61, 57, 24] est une forme d'agrégation par extension du concept de k-anonymat [132] au domaine spatio-temporel. L'idée principale est de s'assurer qu'à chaque instant (ou tranche de temps), chaque individu est situé dans une zone spatiale partagée avec au moins $k-1$ autres individus. La zone spatiale est alors rendue publique plutôt que la localisation exacte de chaque individu. Cela garantit que, même si l'attaquant est capable de trouver le groupe dans lequel sa cible est située, le comportement de cette dernière ne sera pas distinguable de celui d'au moins $k-1$ autres individus. Le domaine du cloaking spatial et du k-anonymat dans les mouvements structurés, telles les trajectoires [140, 110], est un domaine de recherche très actif. Par exemple, dans [1], une approche au k-anonymat est proposée pour la sanitisation des trajectoires. Les trajectoires similaires, qui sont dans un rayon d'incertitude l'une de l'autre, sont regroupées de telle sorte que la taille de chaque groupe soit au minimum k. Seule la trajectoire la plus représentative de chaque groupe est alors publiée. De plus, les trajectoires les plus sensibles sont effacées du jeu de données assaini qui est publié.
- Les *Mix-zones* [10, 49] sont inspirées du concept de *mix-nets* de Chaum [23]. Les mix-zones sont des zones spatiales où (1) les localisations des individus ne sont pas disponibles et (2) chaque individu sortant d'une mix-zone a un pseudonyme différent de celui qu'il avait en entrant. L'objectif principal est de rendre plus difficile la chaînabilité des différentes actions effectuées par un individu. Les zones, immeubles, routes fortement peuplés sont habituellement de bons candidats pour être des mix-zones.
- L'*échange* consiste à échanger les traces de mobilité de deux individus pour une certaine période de temps.
- L'*effacement* des traces de mobilité qui sont jugées trop sensibles peut également être considéré comme procédure ultime d'assainissement. Dans le même esprit, on peut aussi procéder à l'*introduction de fausses traces* [143] dans le jeu de données assaini, afin de mélanger traces réelles d'individus et données artificielles.

Chaque technique d'assainissement possède ses avantages et inconvénients. Le lecteur peut se référer à [53] et à [32] pour plus de détails. De façon générale, l'assainissement des données mène à une perte d'information, il est donc important d'avoir recours à l'évaluation de l'utilité des données (via une *métrique d'utilité*) afin de comparer l'utilité des données d'origine D et des données assainies D'. Une telle métrique d'utilité peut être soit générique, liée par exemple à une propriété statistique du jeu de données, ou être dépendante de l'application. Dans ce dernier cas, la métrique mesure comment une application particulière peut fonctionner correctement avec D' plutôt qu'avec D. Dans la section suivante, nous présentons l'outil GEPETO, un outil que nous développons pour évaluer les techniques d'assainissement, leur effet sur l'utilité des données et l'efficacité des attaques.

3.2.2 GEPETO

GEPETO (pour *GEoPrivacy Enhancing TOolkit*) est un outil générique pour l'évaluation de méthodes d'assainissement et d'attaques d'inférence [51, 53]. Nous introduisons ici l'architecture de cet outil, que nous avons utilisé pour conduire toutes les expériences relatées jusque là dans ce chapitre.

L'objectif global de GEPETO est de fournir aux chercheurs intéressés par la geoprivacy des moyens d'évaluer les méthodes d'assainissement et les attaques d'inférence sur des jeux de données géolocalisées. GEPETO possède une interface de gestion des jeux de données et offre différents outils de manipulation de ces données : mécanismes d'assainissement, attaques d'inférence, visualisation sur des cartes réelles, etc. L'idée générale est d'offrir un outil générique et flexible afin que chacun puisse aisément y ajouter de nouveaux mécanismes. De plus, les composants de visualisation et de mesure de l'utilité permettent d'évaluer directement les bénéfices de l'assainissement, en particulier en regard des attaques d'inférence. C'est, à notre connaissance, un des seuls outils qui regroupe toutes ces fonctionnalités avec une approche unifiée, et on peut le comparer aux outils développés dans le cadre des projets GeoPKDD [3] [58] et MODAP [4] et aux modèles formels de la connaissance d'un attaquant et des contre-mesures possibles issues de [40].

GEPETO est conçu comme une architecture multi-couche, avec l'objectif de rendre le système fonctionnel, efficace, adaptable et sûr. La couche de présentation est composée d'un ensemble de classes qui gèrent la communication avec la base de données pour insérer, modifier ou effacer des données géolocalisées. La couche de contrôle est chargée de l'affichage, de la gestion locale et du contrôle des données et offre un modèle de celles-ci. La couche applicative regroupe les métriques d'utilité, les attaques d'inférence et les techniques d'assainissement. Enfin, la couche de visualisation constitue l'interface graphique de GEPETO, à partir de laquelle l'utilisateur peut accéder aux différentes fonctionnalités : affichage des données sur une carte, application d'attaques d'inférence, assainissement des données et visualisation des résultats de ces opérations. Cette architecture en couches a pour but de séparer clairement l'accès aux données, leur manipulation et leur présentation. Il est donc aisé d'incorporer de nouveaux algorithmes à la couche applicative sans se soucier de la façon dont les couches de contrôle et de présentation accéderont aux données. La couche de présentation utilise des services Web externes pour la visualisation des données, en l'occurrence *OpenStreetMap* [5]. Les choix de conception qui ont guidé cette architecture impliquent à la fois des atouts et des inconvénients. Par exemple, GEPETO ne peut être utilisé hors-ligne car il doit pouvoir accéder au serveur de base de données ainsi qu'au serveur *OpenStreetMap* pour la visualisation. Toutefois, l'ajout de nouvelles fonctionnalités et la maintenance sont facilités grâce à la séparation claire entre les couches.

3. http ://www.geopkdd.eu/
4. http ://modap.org/
5. http ://openstreetmap.fr/

Le prototype actuel de GEPETO est implémenté en Java (*JSDK6.0*) avec une base de données *MySQL 5.0* et représente 60kloc. Nous développons actuellement une nouvelle version qui se base sur Hadoop [6] pour la couche applicative et sur Google Web Toolkit (GWT) [7] pour la couche de présentation. Les objectifs de cette nouvelle version sont : l'efficacité des calculs qui pourront être effectués sur une grille, la portabilité et la légèreté de la couche de présentation à laquelle un navigateur Web pourra accéder directement. Ces travaux impliquent notamment une rétro-conception de la majorité des algorithmes d'assainissement et d'attaque afin qu'ils utilisent les fonctionnalités *Map* et *Reduce* [27]. Nous avons comme objectif à court terme de terminer le développement de la nouvelle version, afin de la rendre publique et ainsi de permettre à la communauté scientifique de l'utiliser et l'enrichir.

Tous les algorithmes présentés dans ce chapitre ont été implémentés dans GEPETO : attaques d'inférence, algorithmes d'assainissement, clustering, métriques d'utilité, etc. Les résultats présentés dans les sections précédentes sont donc issus d'expérimentations réelles sur le prototype de GEPETO, qui s'est révélé un outil fondamental pour la conduite de nos travaux. D'autres résultats sont présentés dans les articles [51, 52, 53, 56], notamment les effets de l'assainissement sur les attaques d'inférence, qui ne sont pas détaillés dans ce manuscrit.

3.2.3 Services géolocalisés respectueux de la vie-privée

Après nous être penchés sur les attaques contre la geoprivacy dans la section précédente, nous abordons maintenant des travaux concernant une architecture logicielle pour le développement de services géolocalisés respectueux de la vie privée. Ces travaux en sont à un stade préliminaire et ont commencé dans le cadre de deux projets : le projet AMORES [8][6] financé par l'ANR [9] débuté en octobre 2011 et le projet XPIR, financé par l'Université de Limoges et par le LAAS et débuté en mai 2012.

Le projet XPIR vise à fournir une boîte à outils pour l'implémentation d'un *Private Information Retrieval* (PIR) [26, 91]. Un protocole PIR permet à un client de questionner une base de données sans que le serveur de cette base de données ne soit capable de connaître quel est l'élément auquel le client a accédé. Une méthode simple mais très inefficace de PIR consiste à envoyer toute la base de données au client. Nous adoptons l'approche cryptographique qui consiste à faire l'hypothèse que les ressources de calcul du serveur sont bornées et nous utilisons un *système cryptographique homomorphe randomisé* [113]. Sans rentrer dans les détails, la caractéristique essentielle d'un système cryptographique homomorphe est qu'il conserve certaines propriétés dans le groupe. En particulier l'implémentation d'un PIR à base de chiffrement homomorphe [15, 98] utilise le fait que la multiplication de chiffrés correspond à l'addition des clairs et que la mise à la puissance des chiffrés correspond à la multiplication

6. http ://hadoop.apache.org
7. https ://developers.google.com/web-toolkit/
8. http ://www.amores-project.org
9. http ://www.agence-nationale-recherche.fr/

des clairs (sous un certain modulo). Le client envoie une requête chifrée, composée de 0 et de 1, 1 pour l'élément qu'il souhaite récupérer et 0 pour les autres éléments. Le chiffrement randomisé permet de ne pas pouvoir distinguer ces chiffrés. Le serveur met alors à la puissance les chiffrés de chaque élément de la base de donnée avec le morceau (chiffré) de requête qui lui correspond, par exemple $C(a)^{C(0)} = C'(0), C(b)^{C(1)} = C'(b)$, puis effectue le produit des valeurs obtenues, ici $C(a)^{C(0)}.C(b)^{C(1)} = C''(b)$. Il renvoie donc ce chiffré (qui n'a aucun sens pour lui) au client, qui le déchiffre pour obtenir l'élément b de la base de données. L'application d'un protocole PIR permettrait, par exemple, à des clients mobiles d'accéder à des services géolocalisés de type *Pages Jaunes* de manière privée. Ceci est un des éléments de notre prospective.

Le projet AMORES [6] se situe explicitement dans un contexte *mobiquitaire*, caractérisé à la fois par une mobilité forte des individus et par le fait que ces individus sont souvent porteurs d'appareils capables de se géolocaliser (smartphone ou voiture équipée d'un GPS). Ce projet propose d'améliorer la mobilité des usagers des services de transports grâce aux possibilités offertes par ces dispositifs, et ce, de façon respectueuse de la vie privée. Le projet est construit autour de trois applications ayant trait à la mobilité : co-voiturage dynamique, calcul d'itinéraires multimodaux en temps réel et réseau social mobile. Dans le cadre de ce domaine applicatif, l'objectif principal du projet est de définir les géo-primitives de communication au niveau intergiciel permettant de fournir les services géolocalisés requis, et cela tout en assurant le respect de la vie privée, en particulier au niveau de la localisation (geoprivacy). Les géo-primitives désignent l'ensemble des services servant aux échanges de données entre applications qui utilisent explicitement le contexte géographique dans leur fonctionnement. Dans le cadre d'AMORES, nous étudions plus particulièrement la fourniture de primitives géolocalisées, telles que le *geocasting* [73], les *géo-registres* [124, 125, 122], les *géo-requêtes*. De plus, afin de garantir l'authenticité des informations de localisation, nous étudions également les techniques permettant de vérifier la position géographique revendiquées par les entités du système (preuves de localisations ou *location proofs*). Enfin, afin d'offrir des garanties de protection de la vie privée, nous souhaitons proposer une approche d'anonymisation des données de localisation que nous appelons *locanymes* [50].

Par ailleurs, les géo-primitives requièrent des composants de base, tels que routage, primitives cryptographiques ou encore la géo-localisation (en extérieur ou à l'intérieur de bâtiments). La problématique du respect de la vie privée se pose également au niveau de ces composants de base et en particulier des traces numériques qui peuvent être générées par leur utilisation. Ainsi, le routage permet de *suivre* les déplacements d'un dispositif (et donc de son porteur), si celui-ci garde toujours la même adresse IP et/ou MAC. Dans le cadre d'AMORES, nous étudions le problème du routage anonyme, ainsi que celui de la génération des clés en prenant en compte les spécificités offertes par la géo-localisation. De façon transversale, ces primitives ne peuvent fonctionner que sur la base de la coopération des entités composant le réseau mobile. Nous développons des mécanismes incitant les entités à coopérer de façon respectueuse de la vie privée, mais également les services permettant de détecter les enti-

tés qui fourniraient délibérément de fausses informations ou qui auraient un comportement malveillant. L'approche envisagée consiste à définir des primitives génériques de gestion de la confiance et d'incitation à la coopération, conjointement avec des instances spécialisées de mécanismes d'observation du comportement des nœuds. Pour chacune des trois applications mentionnées, nous prévoyons de développer un prototype construit à partir des géo-primitives développées au niveau intergiciel. En ce qui concerne l'application de co-voiturage dynamique, elle sera intégrée par un des partenaires dans ses propres produits. L'intergiciel développé sera, par ailleurs, distribué en *open-source*. Enfin, la généralisation des résultats attendus sur le transport toulousain, en coopération avec Tisséo [10], nourrira les réflexions sur le devenir des systèmes de transport urbains et l'applicabilité réelle de l'approche. Pour résumer, les résultats du projet seront à la fois théoriques et pratiques : de nouvelles primitives de géo-communications pour les systèmes mobiquitaires, accompagnées d'un intergiciel et de prototypes, et enfin, un impact sur les systèmes de transport du futur.

Dans [50], nous abordons le problème du respect de la vie privée dans les systèmes géolocalisés par le biais de l'intégration, dès la conception de ces derniers, du concept de locanymes. Un locanyme est une notion qui intègre les concepts fondamentaux du respect de la vie privée (non-chaînabilité, in-falsifiabilité, responsabilité, non-répudiation et souveraineté). Nous y définissons un locanyme comme une version géolocalisée du concept de *pseudonyme*, lié à une position géographique. Ainsi, une entité peut être adressée par son locanyme, ce qui ne dévoile rien de son identité. À partir de cette notion de locanyme, nous travaillons actuellement sur la notion de preuve de localisation. Une preuve de localisation permet à une entité de prouver, a posteriori, qu'elle se trouvait à une certaine position géographique à un instant donné. L'entité qui désire obtenir une preuve de localisation, le *prouveur*, fait appel à des *témoins* afin de lui signer un certificat de localisation. Sans vouloir trop rentrer dans les détails de ces travaux en cours, les preuves de localisations sur lesquelles nous travaillons respectent totalement, sous certaines hypothèses, l'anonymat des différentes entités (prouveurs, témoins) et respectent la propriété de non-chaînabilité pour ces différentes entités également : en observant ou en participant au protocole, on ne peut pas *suivre* le parcours d'une entité. Notre proposition se base sur un schéma de signature de groupe déterministe : pour être valable, une preuve de localisation doit être signée par k témoins, ce qui permet de tolérer $k-1$ défaillances byzantines. Notre approche s'inspire des détecteurs de fautes [21] afin d'isoler la notion de *détecteur de proximité*. Nous étudions l'impact des propriétés de différentes classes de détecteurs de proximité sur la complexité et sur les propriétés des preuves de localisation construites au dessus.

Toujours dans le cadre du projet AMORES [6], nous avons initié une activité concernant l'évaluation de risques en termes de geoprivacy. L'idée est, en s'inspirant des normes sur la gestion des risques de sécurité (*ISO 27001* [66], *ISO 27002* [67], *ISO 27005* [68], *NIST Guide for conducting risk assessments* [130]), de proposer une méthode d'analyse permettant

10. le gestionnaire du réseau de transports en commun de l'agglomération toulousaine

de prendre en compte les risques spécifiques à la vie-privée en général et à la geoprivacy en particulier. Il est, en effet, intéressant de noter que les données géolocalisées vont faire leur apparition dans l'article 4 de la future réglementation européenne concernant la protection des données personnelles, qui devrait être adoptée prochainement [44]. Dans cette réglementation, qui remplacera la précédente [43], les données géolocalisées sont définies comme un identifiant personnel qui appartient aux individus dont elles sont issues. Dans un tel contexte, les applications géolocalisées doivent donc prendre en compte toutes les obligations concernant la protection des données issues de la nouvelle réglementation. Parmi ces obligations, on soulignera l'adjonction du *droit à l'oubli* et du *droit à la portabilité des données*. Dans [117], nous proposons donc une ébauche de méthode d'analyse des risques et l'appliquons au co-voiturage dynamique, qui est un des cas d'étude du projet AMORES. Cette méthode itérative s'appuie sur quatre étapes : l'établissement du contexte, l'identification, l'estimation, et l'évaluation des risques. Ces travaux sont toutefois très préliminaires et devraient faire l'objet de publications futures.

3.3 Conclusion et Prospectives

Dans ce chapitre, nous avons présenté nos travaux concernant la geoprivacy, la vie privée dans un contexte géolocalisé. Ces aspects de mes travaux ont émergé naturellement alors que je me proposais d'étudier la mobilité humaine à des fins de tolérance aux fautes : la sérendipité avait frappé. La geoprivacy représente maintenant la base de la plupart de mes recherches : attaques, moyens de protection, outils. Ce terrain de chasse se révèle passionnant et riche. La place, et le temps, me manquent afin de relater tous les résultats obtenus et attendus, mais j'espère avoir convaincu le lecteur de l'importance de ces travaux.

Les pistes de recherches principales concernant la geoprivacy ont, pour la plupart, déjà été évoquées dans ce chapitre. Il s'agit, en ce qui concerne les attaques, de suivre de nouvelles pistes, comme par exemple d'investiguer l'inférence de sémantique dans la mobilité - un POI situé dans un gymnase pourra très certainement être associé à une activité sportive - ou d'inférer les liens sociaux entre individus. Nous disposons également de plusieurs pistes pour améliorer le pouvoir d'expression des MMCs. Enfin, les avenues de recherche sont larges en ce qui concerne la protection. Dans ce domaine je compte poursuivre la voie des services coopératifs déjà abordés au chapitre 2, à l'opposé des services centralisés qui offrent pour moi un risque bien trop grand en termes de vie privée. Les travaux que nous avons commencés dans le cadre du projet AMORES, cf. section 3.2.3, sont dans cet axe : nouvelles abstractions pour adresser des nœuds de façon anonymes (locanymes), localisation prouvée, géo-cryptographie, communications anonymes à base de geo-casting, etc.

Chapitre 4

Eléments de prospective et Conclusion

4.1 Prospectives

Dans mes travaux passés, exposés dans ce manuscrit, j'ai entrepris l'exploration et l'étude des différents types d'attaques d'inférence sur les données de localisation. Le résultat principal de ces travaux, en un mot, est que parmi les informations personnelles, apprendre la localisation d'un individu est une des plus grandes menaces pour le respect de sa vie privée. En effet, la connaissance de données spatio-temporelles concernant un individu peut être utilisée pour inférer la localisation de son domicile ou de son lieu de travail, pour tracer ses mouvements et ses habitudes, pour apprendre ses centres d'intérêts, voire même pour détecter un changement soudain dans son comportement. Mon projet concernant la *geoprivacy* est articulé autour des attaques et des protection de la vie privée, les sections suivantes présentes brièvement quelques éléments de prospective dans chacun de ces axes.

4.1.1 Attaques contre la *geoprivacy*

– Proposer un modèle général des attaques de geoprivacy, i.e. consolider et enrichir la taxonomie proposée dans [53]. Cette taxonomie se focalise sur l'objectif de l'attaque : identification des lieux importants, inférence et prédiction des mouvements, sémantique de la mobilité d'un individu, dés-anonymisation ou encore découverte des relations sociales. Plusieurs pistes peuvent être explorées afin d'enrichir cette taxonomie. D'une part, on peut envisager de prendre en compte, non plus seulement les attaques ciblant un individu, mais des attaques ciblant des groupes d'individus. On peut également cataloguer les différents attributs ciblés par ces attaques, les différentes techniques qui peuvent être employées, les différents sous-objectifs qui peuvent être nécessaires de remplir, etc. D'autre part, il sera important d'identifier et de définir des métriques génériques de succès pour ces différentes attaques et des modèles précis des différents éléments constitutifs des attaques (capacité et connaissances des attaquants, attributs des données, qualité des données, etc.).

- Proposer de nouvelles attaques. En effet, mes travaux ont jusqu'à présent essentiellement porté sur la dés-anonymisation et la prédiction des mouvements. Ces attaques reposent sur les chaînes de Markov de Mobilité (MMC) comme outil commun pour la représentation de la mobilité individuelle. Les MMCs semblent donc représenter un outil cohérent, compact, précis, lisible et versatile pour représenter la mobilité d'un individu, voire même pour servir de quasi-identifiant. Plusieurs pistes de nouvelles attaques restent à explorer : inférence de la sémantique des mouvements, des liens sociaux, etc.
- Obtenir des données. Nous l'avons vu, les sources de données de mobilité individuelles sont légions pour les utilisateurs de téléphones portables notamment. Chaque intervenant de la chaîne est en capacité d'obtenir des données sur les individus utilisant leur produit (fabricant de téléphone, de système d'exploitation, d'application, de service, opérateur téléphonique, etc.). Chacun de ces intervenant représente une vulnérabilité exploitable pour qui voudrait obtenir frauduleusement des données de mobilité concernant un individu. De plus, on peut envisager d'autres sources, directement sur le dispositifs, sous la forme d'un logiciel malveillant, ou en attaquant les communications entre les différents intervenants de ces systèmes. Installer un logiciel espion sur un dispositif, cf. SpyApps par exemple, est à la portée de n'importe qui, qu'il s'agisse de *script kiddie*, mari jaloux, entreprise soupçonneuse ou encore organisation criminelle. L'avenir appartient aux logiciels malveillants (*malware*), qui se répandent comme un virus, et qui collectent des traces. À l'heure où j'écris ces lignes, un tel logiciel n'a pas encore été signalé, mais au vu de l'actualité dans le domaine, ce n'est absolument pas de la science-fiction, voir par exemple le cas Find and Call qui collecte le carnet d'adresse, ou encore le rapport trimestriel Q1-2012 de MacAfee qui présente la croissance démesurée des logiciels malveillants pour smartphones.
- Étudier et accompagner le déploiement de données ouverte dans le cadre des initiatives OpenData ou MiData. MiData ou SmartDisclosure sont des initiatives des gouvernements britannique et américain, qui visent à libérer l'accès aux données personnelles gérées par les administrations et les entreprises. Un libre accès doit non seulement être fortement sécurisé et régi par des politiques d'accès qui permettent à la fois finesse d'expression des règles mais aussi qui soient compréhensibles et utilisables par tout un chacun. Une problématique essentielle dans le cadre de cette libération des données porte sur le fait que l'on ne sait pas avec quelles autres données il sera possible, dans le futur, de recouper une donnée qui semble aujourd'hui anodine. Il s'agit donc d'étudier le danger, ou l'intérêt, potentiel futur d'une donnée ou d'un ensemble de données.

4.1.2 Protection de la *geoprivacy*

- Proposer un modèle général pour l'analyse des vulnérabilités. En s'inspirant des normes sur la gestion des risques de sécurité (*ISO 27001* [66], *ISO 27002* [67], *ISO 27005* [68], *NIST Guide for conducting risk assessments* [130]), j'envisage de proposer une mé-

thode d'analyse permettant de prendre en compte les risques spécifiques à la vie-privée en général et à la geoprivacy en particulier. Il est, en effet, intéressant de noter que les données géolocalisées vont faire leur apparition dans l'article 4 de la future réglementation européenne concernant la protection des données personnelles, qui devrait être adoptée prochainement [44]. Dans cette réglementation, qui remplacera la précédente [43], les données géolocalisées sont définies comme un identifiant personnel qui appartient aux individus dont elles sont issues. Dans un tel contexte, les applications géolocalisées doivent donc prendre en compte toutes les obligations concernant la protection des données issues de la nouvelle réglementation. Parmi ces obligations, on soulignera l'adjonction du *droit à l'oubli* et du *droit à la portabilité des données*.

– Proposer des solutions paire-à-paire pour les services basés sur la localisation. A mon avis, une première étape pour des services mobiquitaires plus respectueux de la vie privée passe par la décentralisation des services. En effet, il me parait dangereux de se référer toujours au(x) même(s) service(s) centralisé(s) pour toutes ses requêtes : recherches Web, courrier électronique, requêtes localisées de type *pages blanches*, etc. Un service pair-à-pair permet de diviser les informations qui sont disséminées et ainsi rendre plus difficile une collecte de ces informations. J'envisage donc de fournir des mécanismes décentralisés pour la construction de systèmes basés sur la localisation respectueux, par construction, de la vie privée. Par exemple, un locanyme est une notion qui intègre les concepts fondamentaux du respect de la vie privée (non-chaînabilité, in-falsifiabilité, responsabilité, non-répudiation et souveraineté). Je défini un locanyme comme une version géolocalisée du concept de *pseudonyme*, lié à une position géographique. Ainsi, une entité peut être adressée par son locanyme, ce qui ne dévoile rien de son identité. Une preuve de localisation est un autre exemple de mécanisme qui permet à une entité de prouver, a posteriori, qu'elle se trouvait à une certaine position géographique à un instant donné. L'entité qui désire obtenir une preuve de localisation, le *prouveur*, fait appel à des *témoins* afin de lui signer un certificat de localisation. Cette preuve peut ensuite être utilisée pour certifier, à toute fin utile, de façon anonyme à un *vérifieur* que le prouveur se trouvait bien à tel endroit à tel instant.
– Explorer le rapprochement entre l'informatique distribuée de la cryptographie. En effet, on peut envisager plusieurs ponts entre ces communautés. Je propose de commencer par exemple par les *Private Information Retrieval* (PIR) [26, 91]. Un protocole PIR permet à un client de questionner une base de données sans que le serveur de cette base de données ne soit capable de connaître quel est l'élément auquel le client a accédé. Une méthode simple mais très inefficace de PIR consiste à envoyer toute la base de données au client. L'approche cryptographique consiste à faire l'hypothèse que les ressources de calcul du serveur sont bornées et utilise un *système cryptographique homomorphe randomisé* [113]. Une première idée serait de réaliser un PIR géo-localisé et arborescent où un serveur PIR peut, s'il ne dispose pas de la donnée demandée (qu'il ne connait pas), faire appel à un autre serveur, jusqu'à éventuellement arriver à un serveur PIR

central. Un autre aspect particulièrement prometeur concerne la cryptographie basée sur la position, ou la geo-cryptographie [22]. On peut par exemple envisager d'utiliser une géo-clé, une clé cryptographique connue uniquement des entités présentes dans une certaine zone, afin de partager certains secrets entre les membres de cette zone.

– Contrôle d'accès respectueux de la vie privée. Il me semble essentiel de chercher des solutions de contrôle d'accès plus respectueuses de la vie privée que la collecte pure et simple de l'identité, comme c'est souvent le cas actuellement. En effet, pourquoi devrions nous pouvoir être complètement identifiés (de façon unique et traçable) lorsque nous prenons l'avion, louons une voiture, passons un péage autoroutier, etc. D'autre part, nous voyons apparaitre de plus en plus de controle d'accès à des services ou à des ressources basé sur la localisation : accès à la bibliothèque municipale si l'on est résident, tarifs préférentiels pour les évènements organisés par le conseil régional, accès à des fichiers informatiques protégés si l'on se trouve dans une zone sécurisée, etc. Les locanymes et preuves de localisation, issus de mes travaux actuels, représentent très certainement des premiers pas dans cette direction. En effet, la localisation d'un individu est l'un des attributs de son identité qui peuvent être utilisés pour effectuer du contrôle d'accès anonyme, au même titre que l'âge, l'adresse ou l'appartenance à un groupe tel que celui des assurés sociaux.

4.2 Conclusion

Dans ce manuscrit, j'ai décrit mes travaux de recherches de ces treize dernières années. Sous différents angles, ils ont tous concerné la tolérance aux fautes, la protection de la vie privée, et plus généralement la résilience des systèmes informatiques distribués et mobiles. La résilience d'un système informatique peut être définie comme la sûreté de fonctionnement de ce système face aux fautes accidentelles ou malveillantes, présentes et futures et aux changements dans l'environnement. Je suis intimement persuadé que la résilience des systèmes informatiques est un domaine de recherche qui va voir, dans les prochaines années, son importance croître de façon spectaculaire.

En effet, l'émergence de l'informatique ubiquitaire, à savoir omniprésente et interconnectée, implique la cohabitation, l'interaction et la coopération entre systèmes (matériels et logiciels) de natures différentes, du point de vue de leur ancienneté, de leur degré de maturité, de leur conception, etc. Ces différences vont très probablement être sources de nombreuses fautes d'interaction de nature et d'amplitude inconnues jusqu'alors. On peut également pronostiquer que l'utilisation de ces systèmes au sein d'un grand réseau hybride et ouvert sera également source de nombreuses malveillances, que celles-ci soient à but lucratif (chantage au déni de service par exemple), stratégique (militaire ou diplomatique), ou encore visant un bris de vie privée.

L'acceptation de l'informatique ubiquitaire n'est plus vraiment un problème si l'on en croit le taux de pénétration des smartphones et tablettes ou le nombre d'applications téléchargées sur les plateformes dédiées à ces dispositifs. La miniaturisation des gadgets, dans des lunettes de type *Smart Glasses* par exemple, va certainement encore accélérer l'accès à des services informatiques dans la vie de tous les jours. Si la tolérance aux fautes et la sécurité sont des caractéristiques qui sont attendues comme un dû par les utilisateurs de ces systèmes, il n'en est pas de même du respect de la vie privée. Si l'on en juge par le peu de précautions prises par les utilisateurs, le grand public n'est absolument pas conscient des risques qui pèsent sur sa vie privée du fait même de l'utilisation d'applications sur leur téléphones par exemple. Or, l'actualité regorge de faits divers relatant des atteintes à la vie privée : telle entreprise profile les utilisateurs de son application gratuite et revend ces profils pour des publicités ciblées ; telle entreprise revend pour quelques dollars les profils de plusieurs millions d'utilisateurs du plus grand réseau social ; telle autre entreprise se fait voler des millions de numéros de carte de crédit, etc.

La tolérance aux fautes et la sécurité sont maintenant des domaines bien identifiés et bien financés, par les entreprises elles-mêmes, et par les agences de financement de la recherche. Il n'en va pas de même de la protection de la vie privée, qui reste un parent pauvre. La protection de la vie privée a donc besoin d'être vulgarisée afin de devenir un également une caractéristique standard, attendue par les usagers, et elle mérite d'être financée à hauteur des enjeux qu'elle représente, non pas pour les entreprises, mais pour la société elle-même. Je pense contribuer à la prise de conscience sur les dangers concernant la vie privée et apporter ma pierre en ce qui concerne des moyens de protection de celle-ci, et j'espère pouvoir continuer de le faire dans les prochaines années, par tous les moyens à ma disposition en tant que scientifique, comme par exemple par la vulgarisation, par des publications scientifiques et par la formation de jeunes docteurs.

Bibliographie

[1] Abul, O., Bonchi, F., et Nanni, M. (2008). Never walk alone : Uncertainty for anonymity in moving objects databases. Dans *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, p. 376–385. IEEE.

[2] Allan, A. et Warden, P. (2011). Got an iphone or 3g ipad ? apple is recording your moves. *O'Reilly Radar*.

[3] Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., et Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. Dans *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, GIS '07, p. 22 :1–22 :8. New York, NY, USA : ACM. `http://doi.acm.org/10.1145/1341012.1341041`.

[4] Andras, P. et Charlton, B. (2005). Self-Aware Software–Will It Become a Reality ? *Self-star Properties in Complex Information Systems*, p. 229–259.

[5] Armstrong, M., Rushton, G., Zimmerman, D., et al. (1999). Geographically masking health data to preserve confidentiality. *Statistics in medicine*, 18(5), p. 497–525.

[6] Artigues, C., Deswarte, Y., Guiochet, J., Huguet, M.-J., Killijian, M.-O., Powell, D., Roy, M., Bidan, C., Prigent, N., Anceaume, E., Gambs, S., Guette, G., Hurfin, M., et Schettini, F. (2012). Amores : an architecture for mobiquitous resilient systems. Dans *Proceedings of AppRoaches to MObiquitous Resilience (ARMOR'12), a EDCC workshop.*.

[7] Basile, C., Kalbarczyk, Z., et Iyer, R. (2003). A preemptive deterministic scheduling algorithm for multithreaded replicas. *Dependable Systems and Networks, International Conference on*, 0, p. 149.

[8] Bennani, M. T. (2005). *Tolérance aux fautes dans les systèmes répartis à base d'intergiciels réflexifs standards*. Thèse de doctorat, Institut National des Sciences Appliquées, Toulouse, 146p. Doctorat.

[9] Bennani, T., Blain, L., Courtès, L., Fabre, J.-C., Killijian, M.-O., Marsden, E., et Taïani, F. (2004). Implementing simple replication protocols using corba portable interceptors and java serialization. Dans *DSN*, p. 549–554. IEEE Computer Society.

[10] Beresford, A. et Stajano, F. (2003). Location privacy in pervasive computing. *Pervasive Computing, IEEE*, 2(1), p. 46–55.

[11] Bettini, C., Mascetti, S., Wang, X., Freni, D., et Jajodia, S. (2009). Anonymity and historical-anonymity in location-based services. *Privacy in Location-Based Applications*, p. 1–30.

[12] Bolosky, W. J., Douceur, J. R., Ely, D., et Theimer, M. (2000). Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. *SIGMETRICS Perform. Eval. Rev.*, 28(1), p. 34–43. `http://doi.acm.org/10.1145/345063.339345`.

[13] Bouchenak, S., Boyer, F., De Palma, N., et Hagimont, D. (2003). Can aspects be injected ? experience with replication and protection. *On The Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE*, p. 1402–1420.

[14] Buttyán, L., Holczer, T., et Vajda, I. (2007). On the effectiveness of changing pseudonyms to provide location privacy in vanets. Dans *Proceedings of the 4th European conference on Security and privacy in ad-hoc and sensor networks*, p. 129–141. Springer-Verlag.

[15] Cachin, C., Micali, S., et Stadler, M. (1999). Computationally private information retrieval with polylogarithmic communication. Dans *Advances in Cryptology - EUROCRYPT'99*, p. 402–414. Springer.

[16] Cala, J. (2007). Migration in CORBA Component Model. Dans *Distributed applications and interoperable systems : 7th IFIP WG 6.1 international conference, DAIS 2007, Paphos, Cyprus, June 6-8, 2007 : proceedings*, p. 139. Springer-Verlag New York Inc.

[17] Carlson, J. et Murphy, R. R. (2005). How ugvs physically fail in the field. *IEEE Transactions on Robotics*, 21(3), p. 423–437.

[18] Cassa, C., Wieland, S., et Mandl, K. (2008). Re-identification of home addresses from spatial locations anonymized by gaussian skew. *International journal of health geographics*, 7(1), p. 45.

[19] Castelluccia, C. (2007). Securing very dynamic groups and data aggregation in wireless sensor networks. Dans *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, p. 1–9. IEEE.

[20] Cavin, D., Sasson, Y., et Schiper, A. (2002). On the accuracy of manet simulators. Dans *POMC '02 : Proceedings of the second ACM international workshop on Principles of mobile computing*, p. 38–43. New York, NY, USA : ACM Press.

[21] Chandra, T. et Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2), p. 225–267.

[22] Chandran, N., Goyal, V., Moriarty, R., et Ostrovsky, R. (2009). Position based cryptography. *Advances in Cryptology-CRYPTO 2009*, p. 391–407.

[23] Chaum, D. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), p. 84–90.

[24] Cheng, R., Zhang, Y., Bertino, E., et Prabhakar, S. (2006). Preserving user location privacy in mobile data management infrastructures. Dans *Privacy Enhancing Technologies*, p. 393–412. Springer.

[25] Chiba, S. (1995). A metaobject protocol for c++. Dans *OOPSLA*, p. 285–299.

[26] Chor, B., Goldreich, O., Kushilevitz, E., et Sudan, M. (1995). Private information retrieval. Dans *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, p. 41–50. IEEE.

[27] Chu, C., Kim, S., Lin, Y., Yu, Y., Bradski, G., Ng, A., et Olukotun, K. (2007). Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19, p. 281.

[28] Chu, H.-N. (2011). *Test and Evaluation of the Robustness of the Functional Layer of an Autonomous Robot*. Thèse de doctorat, Institut National Polytechnique, Toulouse, 119p. Doctorat.

[29] Chu, H.-N., Arlat, J., Killijian, M.-O., Lussier, B., et Powell, D. (2009). Robustness Testing of Robot Controller Software. Dans *Proceedings of the 12th European Workshop on Dependable Computing, EWDC 2009*, edité par H. WAESELYNCK, p. 2 pages. Toulouse, France. `http://hal.archives-ouvertes.fr/hal-00381686/en/`. Rapport LAAS n˚09184.

[30] CNIL (2011). *Géolocalisation : l'iPhone bavarde pendant votre sommeil ?*. Rapport technique, CNIL. `http://www.cnil.fr/la-cnil/actualite/article/article/geolocalisation-liphone-bavarde-pendant-votre-sommeil/`.

[31] Cohen, N., Purakayastha, A., Turek, J., Wong, L., et Yeh, D. (2001). Challenges in flexible aggregation of pervasive data. *IBM Research Report RC*, 21942, p. 98646.

[32] Cottrill, C. (2011). Location privacy : Who protects ? *URISA Journal-Urban and Regional InformationSystems Association*, 23(2), p. 49.

[33] Courtès, L. (2007). *Sauvegarde coopérative de données pour dispositifs mobiles*. Thèse de doctorat, Institut National Polytechnique, Toulouse, 149p. Doctorat.

[34] Courtès, L., Hamouda, O., Kaaniche, M., Killijian, M., et Powell, D. (2007). Dependability evaluation of cooperative backup strategies for mobile devices. Dans *Pacific Rim Dependable Computing*, p. 139–146.

[35] Courtès, L., Killijian, M., et Powell, D. (2006). Storage tradeoffs in a collaborative backup service for mobile devices. Dans *Proceedings of the 6th European Dependable Computing Conference*, LAAS Report #05673, p. 129–138.

[36] Courtès, L., Killijian, M.-O., et Powell, D. (2007). Security rationale for a cooperative backup service for mobile devices. Dans *Proceedings of the Latin-American Symposium on Dependable Computing (LADC)*, p. 212–230. Springer-Verlag.

[37] Cox, L. P., Murray, C. D., et Noble, B. D. (2002). Pastiche : making backup cheap and easy. *SIGOPS Oper. Syst. Rev.*, 36(SI), p. 285–298. `http://dx.doi.org/10.1145/844128.844155`.

[38] Das, S. R., Castañeda, R., et Yan, J. (2000). Simulation-based performance evaluation of routing protocols for mobile ad hoc networks. Dans *Mob. Netw. Appl.*, tome 5, p. 179–189. Hingham, MA, USA : Kluwer Academic Publishers.

[39] De Mulder, Y., Danezis, G., Batina, L., et Preneel, B. (2008). Identification via location-profiling in gsm networks. Dans *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, WPES '08, p. 23–32. New York, NY, USA : ACM. `http://doi.acm.org/10.1145/1456403.1456409`.

[40] Duckham, M., Kulik, L., et Birtley, A. (2006). A spatiotemporal model of strategies and counter strategies for location privacy protection. *Geographic Information Science*, p. 47–64.

[41] Eagle, N. et (Sandy) Pentland, A. (2006). Reality mining : sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4), p. 255–268. `http://dx.doi.org/10.1007/s00779-005-0046-3`.

[42] Etter, V., Kafsi, M., et Kazemi, E. (2012). Been There, Done That : What Your Mobility Traces Reveal about Your Behavior. Dans *Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing*.

[43] European Parliament and the Council of the European Union (1995). Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union*, L 281, p. 0031–0050.

[44] European Parliament and the Council of the European Union (2011). Proposal for General Data Protection Regulation. *Official Journal of the European Union*.

[45] Fabre, J.-C., Killijian, M.-O., et Pareaud, T. (2010). Towards on-line adaptation of fault tolerance mechanisms. Dans *EDCC*, p. 45–54. IEEE Computer Society.

[46] Fabre, J.-C., Killijian, M.-O., et Taiani, F. (2011). Robustness of automotive applications using reflective computing : lessons learnt. Dans *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, p. 230–235. New York, NY, USA : ACM. http://doi.acm.org/10.1145/1982185.1982237.

[47] Farrell, S. et Cahill, V. (2006). *Delay- and disruption-tolerant networking*. Artech House telecommunications library. Artech House.

[48] Freni, D., Ruiz Vicente, C., Mascetti, S., Bettini, C., et Jensen, C. S. (2010). Preserving location and absence privacy in geo-social networks. Dans *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, p. 309–318. New York, NY, USA : ACM. http://doi.acm.org/10.1145/1871437.1871480.

[49] Freudiger, J., Raya, M., Félegyházi, M., Papadimitratos, P., et Hubaux, J. (2007). Mix-zones for location privacy in vehicular networks. Dans *Proceedings of the 1st International Workshop on Wireless Networking for Intelligent Transportation Systems (WiN-ITS 07)*.

[50] Gambs, S., Killijian, M.-O., et andM. Traore, M. R. (2012). Locanyms : Towards privacy-preserving location-based services. Dans *Proceedings of AppRoaches to MObiquitous Resilience (ARMOR'12), a EDCC workshop.*.

[51] Gambs, S., Killijian, M.-O., et del Prado Cortez, M. N. (2010). GEPETO : A GEoPrivacy-Enhancing TOolkit. Dans *AINA Workshops*, p. 1071–1076. IEEE Computer Society.

[52] Gambs, S., Killijian, M.-O., et del Prado Cortez, M. N. n. (2010). Show me how you move and i will tell you who you are. Dans *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, p. 34–41. New York, NY, USA : ACM. http://doi.acm.org/10.1145/1868470.1868479.

[53] Gambs, S., Killijian, M.-O., et del Prado Cortez, M. N. n. (2011). Show me how you move and i will tell you who you are. *Transactions on Data Privacy*, 4(2), p. 103–126.

[54] Gambs, S., Killijian, M.-O., et del Prado Cortez, M. N. n. (2011). Towards temporal mobility markov chains. Dans *Proceedings of the 1st International Workshop on Dynamicity collocated with OPODIS'2011*.

[55] Gambs, S., Killijian, M.-O., et del Prado Cortez, M. N. n. (2012). Next place prediction using mobility markov chains. Dans *Proceedings of EuroSys 2012 Workshop on Measurement, Privacy, and Mobility (MPM)*.

[56] Gambs, S., Killijian, M.-O., et Nunez Del Prado Cortez, M. (2011-10). *De-anonymization attack on geolocated datasets*. Rapport technique, LAAS-CNRS. http://hal.archives-ouvertes.fr/hal-00718763. 12.

[57] Gedik, B. et Liu, L. (2004). Mobieyes : Distributed processing of continuously moving queries on moving objects in a mobile system. *Advances in Database Technology-EDBT 2004*, p. 523–524.

[58] Giannotti, F., Giannotti, G., et Pedreschi, D. (2008). *Mobility, data mining, and privacy : geographic knowledge discovery*. Springer.

[59] Golle, P. et Partridge, K. (2009). On the anonymity of home/work location pairs. Dans *Proceedings of the 7th International Conference on Pervasive Computing*, Pervasive '09, p. 390–397. Berlin, Heidelberg : Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-01516-8_26.

[60] Gonzalez, M. C., Hidalgo, C. A., et Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *Nature*, 453(7196), p. 779–782.

[61] Gruteser, M. et Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. Dans *Proceedings of the 1st international conference on Mobile systems, applications and services*, p. 31–42. ACM.

[62] Hamida, E. B., Chelius, G., et Gorce, J. M. (2008). On the complexity of an accurate and precise performance evaluation of wireless networks using simulations. Dans *11th ACM-IEEE Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*.

[63] Hariharan, R. et Toyama, K. (2004). Project lachesis : parsing and modeling location histories. *Geographic Information Science*, p. 106–124.

[64] Hericko, M., Juric, M., Rozman, I., Beloglavec, S., et Zivkovic, A. (2003). Object serialization analysis and comparison in java and .net. *ACM SIGPLAN Notices*, 38(8), p. 44–54.

[65] Hightower, J. et Borriello, G. (2001.). A survey and taxonomy of location systems for ubiquitous computing. *IEEE Computer*.

[66] ISO (2005). *ISO/IEC 27001 :2005 - Information technology – Security techniques – Information security management systems – Requirements*. Rapport technique, ISO.

[67] ISO (2005). *ISO/IEC 27002 :2005 - Information technology – Security techniques – Code of practice for information security management*. Rapport technique, ISO.

[68] ISO (2008). *ISO/IEC 27005 :2008 - Information technology – Security techniques – Information security risk management*. Rapport technique, ISO.

[69] Jedrzejczyk, L., Price, B., Bandara, A., et Nuseibeh, B. (2009). I know what you did last summer : risks of location data leakage in mobile and social computing. *Department of Computing Faculty of Mathematics, Computing and Technology The Open University*.

[70] Jimenez-Peris, R., Patio-Martinez, M., et Aravalo, S. (2000). Deterministic scheduling for transactional multithreaded replicas. *Reliable Distributed Systems, IEEE Symposium on*, 0, p. 164.

[71] Kang, J. H., Stewarta, B., Borriello, G., et Welbourne, W. (2004). Extracting places from traces of locations. Dans *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, p. 110–118.

[72] Killijian, M.-O. (2000). *Tolérance aux fautes sur Corba par protocole à métaobjets et langages réflexifs*. Thèse de doctorat, Institut National Polytechnique de Toulouse. `http://www.laas.fr/~mkilliji/publications.html`. Doctorat.

[73] Killijian, M.-O., Cunningham, R., Meier, R., Mazare, L., et Cahill, V. (2001). Towards group communication for mobile participants. Dans *Proceedings of Principles of Mobile Computing (POMC)*, p. 75–82.

[74] Killijian, M.-O. et Fabre, J.-C. (2000). Implementing a reflective fault-tolerant corba system. Dans *SRDS*, p. 154–163.

[75] Killijian, M.-O. et Fabre, J.-C. (2003). Adaptive fault tolerant systems : Reflective design and validation. Dans *IPDPS*, p. 212. IEEE Computer Society.

[76] Killijian, M.-O., Fabre, J.-C., Ruiz-Garcia, J.-C., et Chiba, S. (1998). A metaobject protocol for fault-tolerant corba applications. Dans *SRDS*, p. 127–134.

[77] Killijian, M.-O., Powell, D., Banâtre, M., Couderc, P., et Roudier, Y. (2004). Collaborative backup for dependable mobile applications. Dans *Proceedings of 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (Middleware 2004)*, p. 146–149. New York, NY, USA : ACM Press. `http://www.laas.fr/mosaic/`.

[78] Killijian, M.-O., Powell, D., Roy, M., et Severac, G. (2008). Experimental evaluation of ubiquitous systems. Dans *Fast abstract at DEBS'08*. `http://debs08.dis.uniroma1.it/pdf/fa-killijian.pdf`.

[79] Killijian, M.-O., Rivière, N., et Roy, M. (2007). Experimental evaluation of resilience for ubiquitous mobile systems. Dans *Proc. of UbiComp, Workshop on Ubiquitous Systems Evaluation (USE)*, p. 283–287.

[80] Killijian, M.-O. et Roy, M. (2009). Brief announcement : a platform for experimenting with mobile algorithms in a laboratory. Dans *PODC*, edité par S. Tirthapura et L. Alvisi, p. 316–317. ACM.

[81] Killijian, M.-O. et Roy, M. (2010). Data backup for mobile nodes : A cooperative middleware and an experimentation platform. Dans *Architecting Dependable Systems VII*, edité par A. Casimiro, R. de Lemos, et C. Gacek, tome 6420 de *Lecture Notes in Computer Science*, p. 53–73. Springer Berlin / Heidelberg.

[82] Killijian, M.-O., Roy, M., et Séverac, G. (2010). Arum : a cooperative middleware and an experimentation platform for mobile systems. Dans *Proc. of WiMob'2010*, p. 8.

[83] Killijian, M.-O., Roy, M., et Séverac, G. (2011). The arum experimentation platform : an open tool to evaluate mobile systems applications. Dans *Proc. of Amire'11*.

[84] Killijian, M.-O., Roy, M., Severac, G., et Zanon, C. (2009). Data backup for mobile nodes : a cooperative middleware and experimentation platform. Dans *Proc. of the Workshop on Architecting Dependable Systems of the IEEE International Conference on Dependable Systems and Networks (DSN-2009), Lisboa, Portugal*.

[85] Killijian, M.-O., Roy, M., et Trédan, G. (2010). Beyond san francisco cabs : building a *-lity mining dataset. Dans *Proceedings of Workshop on the Analysis of Mobile Phone Networks (NetMob). Sattelite of NetSci 2010, Cambridge, MA, USA*.

[86] Killijian, M.-O., Ruiz-Garcia, J.-C., et Fabre, J.-C. (1999). Using compile-time reflection for objects'state capture. Dans *Reflection*, edité par P. Cointe, tome 1616 de *Lecture Notes in Computer Science*, p. 150–152. Springer.

[87] Killijian, M.-O., Ruiz-Garcia, J.-C., et Fabre, J.-C. (2002). Portable serialization of corba objects : a reflective approach. Dans *OOPSLA*, p. 68–82.

[88] Kiukkonen, N. (2009). *Technical Report : Data collection campaign*. Rapport technique, Nokia Research Center, Lausanne, Switzerland.

[89] Krumm, J. (2007). Inference attacks on location tracks. Dans *Proceedings of the 5th international conference on Pervasive computing*, PERVASIVE'07, p. 127–143. Berlin, Heidelberg : Springer-Verlag. `http://dl.acm.org/citation.cfm?id=1758156.1758167`.

[90] Krumm, J. (2009). A survey of computational location privacy. *Pers Ubiquit Comput*, p. 391–399.

[91] Kushilevitz, E. et Ostrovsky, R. (1997). Replication is not needed : Single database, computationally-private information retrieval. Dans *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, p. 364–373. IEEE.

[92] Kwan, M., Cartwright, W., et Arrowsmith, C. (2012). Tracking movements with mobile phone billing data : A case study with publicly-available data. Dans *Advances in Location-Based Services*, edité par G. Gartner, F. Ortag, W. Cartwright, G. Gartner, L. Meng, et M. P. Peterson, Lecture Notes in Geoinformation and Cartography, p. 109–117. Springer Berlin Heidelberg. `http://dx.doi.org/10.1007/978-3-642-24198-7_7`.

[93] Kwan, M., Casas, I., et Schmitz, B. (2004). Protection of geoprivacy and accuracy of spatial information : How effective are geographical masks ? *Cartographica : The International Journal for Geographic Information and Geovisualization*, 39(2), p. 15–28.

[94] Lamport, L. (1986). On interprocess communication. *Distributed Computing*, 1, p. 77–101.

[95] Laurila, J. K., Gatica-Perez, D., Aad, I., Blom, J., Bornet, O., Do, T., Dousse, O., Eberle, J., et Miettinen, M. (2012). The mobile data challenge : Big data for mobile computing research. Dans *Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing*. Newcastle, UK.

[96] Leitner, M. et Curtis, A. (2006). A first step towards a framework for presenting the location of confidential point data on mapsÑresults of an empirical perceptual study. *International Journal of Geographical Information Science*, 20(7), p. 813–822.

[97] Liao, L., Fox, D., et Kautz, H. (2005). Location-based activity recognition using relational markov networks. Dans *In : Proceedings of the Nineteenth International Conference on Artificial Intelligence, IJCAI'05*.

[98] Lipmaa, H. (2005). An oblivious transfer protocol with log-squared communication. *Information Security*, p. 314–328.

[99] Lu, C. (2009). *Robustesse du logiciel embarqué multicouche par une approche réflexive : application à l'automobile*. Thèse de doctorat, Institut National des Sciences Appliquées, Toulouse, 142p. Doctorat.

[100] Lu, C., Fabre, J.-C., et Killijian, M.-O. (2009). An approach for improving fault-tolerance in automotive modular embedded software. Dans *17th International Conference on Real-Time Systems (RTNS 2009), Paris (France), 26-27 Octobre 2009*, p. 132–147.

[101] Lu, C., Fabre, J.-C., et Killijian, M.-O. (2009). Robustness of modular multi-layered software in the automotive domain : a wrapping-based approach. Dans *14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009), Majorque (Espagne)*.

[102] Lussier, B. (2007). *Tolérance aux fautes dans les systèmes autonomes*. Thèse de doctorat, Institut National Polytechnique, Toulouse, 131p. Doctorat.

[103] Lussier, B., Chatila, R., Ingrand, F., Killijian, M. O., et Powell, D. (2004). On fault tolerance and robustness in autonomous systems. Dans *Proc. of 3rd IARP/IEEE-RAS Joint Workshop on Technical Challenge for Dependable Robots in Human Environments, Manchester, Great Britain*.

[104] Lussier, B., Gallien, M., Guiochet, J., Ingrand, F., Killijian, M.-O., et Powell, D. (2007). Fault tolerant planning for critical robots. Dans *DSN*, p. 144–153. IEEE Computer Society.

[105] Lussier, B., Gallien, M., Guiochet, J., Ingrand, F., Killijian, M.-O., et Powell, D. (2007). Planning with diversified models for fault-tolerant robots. Dans *ICAPS*, edité par M. S. Boddy, M. Fox, et S. Thiébaux, p. 216–223. AAAI.

[106] Lussier, B., Lampe, A., Chatila, R., Guiochet, J., Ingrand, F., Killijian, M.-O., et Powell, D. (2005). Fault tolerance in autonomous systems : How and how much ? Dans *Proc. of 4th IARP/IEEE-RAS Joint Workshop on Technical Challenge for Dependable Robots in Human Environments, Nagoya, Japan*.

[107] Ma, C. Y., Yau, D. K., Yip, N. K., et Rao, N. S. (2010). Privacy vulnerability of published anonymous mobility traces. Dans *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, MobiCom '10, p. 185–196. New York, NY, USA : ACM. `http://doi.acm.org/10.1145/1859995.1860017`.

[108] Napper, J., Alvisi, L., et Vin, H. (2003). A fault-tolerant java virtual machine. *Dependable Systems and Networks, International Conference on*, 0, p. 425.

[109] Narasimhan, P., Moser, L., et Melliar-Smith, P. (1999). Enforcing determinism for the consistent replication of multithreaded corba applications. *Reliable Distributed Systems, IEEE Symposium on*, 0, p. 263.

[110] Nergiz, M., Atzori, M., et Saygin, Y. (2008). Towards trajectory anonymization : a generalization-based approach. Dans *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, p. 52–61. ACM.

[111] Object Management Group (2006). *CORBA Component Model 4.0 Specification*. Specification Version 4.0, Object Management Group. `http://www.omg.org/docs/formal/06-04-01.pdf`.

[112] Ohm, P. (2009). Broken promises of privacy : Responding to the surprising failure of anonymization. *UCLA Law Review*.

[113] Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. Dans *Advances in Cryptology - EUROCRYPT'99*, p. 223–238. Springer.

[114] Pareaud, T. (2009). *Adaptation en ligne de mécanismes de tolérance aux fautes par une approche à composants ouverts*. Thèse de doctorat, Institut National Polytechnique, Toulouse, 166p.

[115] Pareaud, T., Fabre, J.-C., et Killijian, M.-O. (2008). Componentization of fault tolerance software for fine-grain adaptation. Dans *PRDC*, p. 248–255. IEEE Computer Society.

[116] Pérennou, T. (1997). *Une architecture à métaobjets pour systèmes répartis tolérant les fautes*. Thèse de doctorat, Institut National Polytechnique, Toulouse, 168p. Doctorat.

[117] Petcana, L. (2011). *GeoPrivacy Risk Assessment - Application to a Carpooling System*. Rapport technique, LAAS.

[118] Pfitzmann, A. et Hansen, M. (2010). A terminology for talking about privacy by data minimization : Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. *URL : http ://dud. inf. tu-dresden. de/literatur/Anon_Terminology_v0*, 34.

[119] Piorkowski, M., Sarafijanovic-Djukic, N., et Grossglauser, M. (2009). CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http ://crawdad.cs.dartmouth.edu/epfl/mobility.

[120] Powell, D., Arlat, J., Chu, H. N., Ingrand, F., et Killijian, M.-O. (2012). Testing the input timing robustness of real-time control software for autonomous systems. Dans *EDCC*, edité par C. Constantinescu et M. P. Correia, p. 73–83. IEEE.

[121] Reiter, S. et Rigoll, G. (2005). Multimodal meeting analysis by segmentation and classification of meeting events based on a higher level semantic approach. Dans *Proc. IEEE ICASSP*, tome 2, p. 161–164.

[122] Roy, M., Bonnet, F., Querzoni, L., Bonomi, S., Killijian, M.-O., et Powell, D. (2008). Geo-registers : An abstraction for spatial-based distributed computing. Dans *Int. Conf. On Principles Of DIStributed computing (OPODIS), LNCS 5401*, p. 534–537.

[123] Roy, M., Bonnet, F., Querzoni, L., Bonomi, S., Killijian, M.-O., et Powell, D. (2009). *Geo-registers : An Abstraction for Spatial-Based Distributed Computing*. Rapport technique 09190, LAAS-CNRS.

[124] Roy, M., Killijian, M.-O., Querzoni, L., Bonnet, F., Piergiovanni, S. T., et Bonomi, S. (2008). Fada : Formalisms and algorithms for resilient services design in ambient systems. Dans *Fast abstract at DEBS'08*. `http://debs08.dis.uniroma1.it/pdf/fa-killijian.pdf`.

[125] Roy, M., Killijian, M.-O., Querzoni, L., Bonnet, F., Piergiovanni, S. T., et Bonomi, S. (2008). FADA : Formalisms and Algorithms for Resilient Services Design in Ambient Systems. Dans *Proceedings of the 7th European Dependable Computing Conference (EDCC) - Fast Abstracts*. `http://www.dis.uniroma1.it/~midlab`.

[126] Ruiz-Garcia, J.-C., Killijian, M.-O., Fabre, J.-C., et Thévenod-Fosse, P. (2003). Reflective fault-tolerant systems : From experience to challenges. *IEEE Trans. Computers*, 52(2), p. 237–254.

[127] Sifakis, J. (2009). Component-based construction of real-time systems in bip. Dans *CAV*, p. 33–34.

[128] Song, C., Qu, Z., Blumm, N., et Barabasi, A.-L. (2010). Limits of predictability in human mobility. *Science*, 327(5968), p. 1018–1021.

[129] Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., et Vangenot, C. (2008). A conceptual view on trajectories. *Data Knowl. Eng.*, 65(1), p. 126–146. `http://dx.doi.org/10.1016/j.datak.2007.10.008`.

[130] Stoneburner, G., Goguen, A., et Feringa, A. (2002). Risk management guide for information technology systems. *Nist special publication*, 800(30), p. 800–30.

[131] Sweeney, L. (2000). Uniqueness of simple demographics in the us population. *LIDAP-WP4. Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA*.

[132] Sweeney, L. et al. (2002). k-anonymity : A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5), p. 557–570.

[133] Taïani, F. (2004). *La réflexivité dans les architectures multi-niveaux : application aux systèmes tolérant les fautes*. Thèse de doctorat, Université Paul Sabatier, Toulouse, 155p. Doctorat.

[134] Taïani, F., Fabre, J.-C., et Killijian, M.-O. (2002). Principles of multi-level reflection for fault tolerant architectures. Dans *PRDC*, p. 59–66. IEEE Computer Society.

[135] Taïani, F., Fabre, J.-C., et Killijian, M.-O. (2003). Towards implementing multi-layer reflection for fault-tolerance. Dans *DSN*, p. 435–444. IEEE Computer Society.

[136] Taïani, F., Fabre, J.-C., et Killijian, M.-O. (2005). A multi-level meta-object protocol for fault-tolerance in complex architectures. Dans *DSN*, p. 270–279. IEEE Computer Society.

[137] Taïani, F., Killijian, M.-O., et Fabre, J.-C. (2006). Intergiciels pour la tolérance aux fautes. *Technique et Science Informatiques*, 25(5), p. 599–630.

[138] Taïani, F., Killijian, M.-O., et Fabre, J.-C. (2009). Cosmopen : dynamic reverse engineering on a budget. how cheap observation techniques can be used to reconstruct complex multi-level behaviour. *Softw., Pract. Exper.*, 39(18), p. 1467–1514.

[139] Tatsubori, M., Chiba, S., Itano, K., et Killijian, M.-O. (1999). Openjava : A class-based macro system for java. Dans *Reflection and Software Engineering*, edité par W. Cazzola, R. J. Stroud, et F. Tisato, tome 1826 de *Lecture Notes in Computer Science*, p. 117–133. Springer.

[140] Terrovitis, M. et Mamoulis, N. (2008). Privacy preservation in the publication of trajectories. Dans *Mobile Data Management, 2008. MDM'08. 9th International Conference on*, p. 65–72. IEEE.

[141] Tomatis, N., Terrien, G., Piguet, R., Burnier, D., Bouabdallah, S., Arras, K. O., et Siegwart, R. (2003). Designing a secure and robust mobile interacting robot for the long term. Dans *In Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, p. 4246–4251.

[142] Veiga, L., Santos, N., Lebre, R., et Ferreira, P. (2004). Loosely-coupled, mobile replication of objects with transactions. Dans *Tenth International Conference on Parallel and Distributed Systems, 2004. ICPADS 2004. Proceedings*, p. 675–682.

[143] You, T., Peng, W., et Lee, W. (2007). Protecting moving trajectories with dummies. Dans *Mobile Data Management, 2007 International Conference on*, p. 278–282. IEEE.

[144] Zheng, Y., Li, Q., Chen, Y., Xie, X., et Ma, W.-Y. (2008). Understanding mobility based on gps data. Dans *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, p. 312–321. New York, NY, USA : ACM. `http://doi.acm.org/10.1145/1409635.1409677`.

[145] Zhou, C., Frankowski, D., Ludford, P., Shekhar, S., et Terveen, L. (2004). Discovering personal gazetteers : an interactive clustering approach. Dans *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, GIS '04, p. 266–273. New York, NY, USA : ACM. `http://doi.acm.org/10.1145/1032222.1032261`.

CONTRIBUTIONS À LA RÉSILIENCE ET AU RESPECT DE LA VIE PRIVÉE DES SYSTÈMES MOBIQUITAIRES

Résumé : Les travaux présentés dans ce mémoire résument l'ensemble de mes activités dans le domaine de la résilience et du respect de la vie privée dans les systèmes ubiquitaires mobiles. Ils s'orientent sur trois axes principaux : l'utilisation de la réfléxivité pour la construction d'architectures sûres, la résilience des systèmes mobiquitaires (architectures, algorithmes et leur évalution), et la *geoprivacy*.

Le premier axe concerne la tolérance aux fautes pour les systèmes distribués, sous un angle architecture et langage. Dans ces travaux, j' ai étudié l'utilisation de la réflexivité à la compilation et à l'exécution afin de faciliter l'implémentation de mécanismes de tolérance aux fautes indépendemment de l'application. L'utilisation de la réflexivité a été étudiée dans le cadre de travaux théoriques, concernant la *réflexivité multi-niveaux*, ou plus pratiques, comme la mise en œuvre de la réflexivité sur des composants sur étagères, dans une architecture logicielle embarquée, ou pour permettre l'adaptation de mécanismes de tolérance aux fautes à l'exécution. Le chapitre 1 présente ces différents travaux.

Le deuxième axe concerne la tolérance aux fautes dans les systèmes mobiles. Mon approche a été dd'aborder la mobilité comme un atout et non pas comme une difficulté. Cette approche m'a mené à étudier la notion de communication de groupes géographiques : comment définir un groupe d'entités communicantes en fonction de leur localisation respective ou en fonction de leur proximité. J'ai ensuite, sous l'angle du pair-à-pair, proposé un système de sauvegarde coopérative de données, où les nœuds mobiles participants offrent un service de stockage sécurisé qu'ils peuvent utiliser afin de sauvegarder leurs données critiques. Cette solution a été également déclinée pour offrir un système de *boîte noire virtuelle* pour l'automobile. Ces travaux ont été traités sous des angles algorithmique et architecturaux, mais également sous l'angle de l'évaluation de la sûreté de fonctionnement, à la fois analytique et expérimentale. Ces travaux font l'objet du chapitre 2.

Dans le cadre de mes recherches sur la résilience des systèmes mobiquitaires, des questions d'ordre déontologique ont été soulevées : comment exploiter des données de mobilité individuelles tout en préservant la vie privée des individus ? C'est à cette occasion que je me suis intéréssé à ce que l'on peut nommer la *geoprivacy*. Ce domaine représente maintenant la majeure partie de mes travaux, tant sous l'angle des attaques que sous celui de la protection. Nous proposons un modèle Markovien de mobilité individuelle, outil à la fois compact, précis, intelligible et facilement adaptable pour représenter la mobilité d'un individu. Sur la base de ce modèle de mobilité, nous proposons plusieurs attaques qui ciblent par exemple la prédiction des déplacements futurs, ou encore la des-anonymisation. En ce qui concerne la protection de la geoprivacy, nous travaillons actuellement sur des abstractions de niveau intergiciel, tel les locanymes ou les localisation vérifiées, afin de proposer une architecture sûre et respectueuse de la vie privée pour les systèmes géolocalisés. Le chapitre 3 aborde ces aspects de ma recherche.

Ces différents travaux se sont enchaînés dans un mélange de hasard, de sérendipité et de poursuite d'un objectif commun : fournir des moyens algorithmiques et architecturaux pour la résilience des systèmes informatiques actuels, à savoir distribués, mobiles, ubiquitaires. Les techniques et outils que j'utilise pour aborder cette problématique large auront été divers et variés, cela participe à mon expérience, sans cesse renouvelée. De nombreuses pistes de recherche sont encore ouvertes et sont exposées dans le chapitre 4.

TOWARDS RESILIENT AND PRIVATE MOBIQUITOUS SYSTEMS

Abstract : The works presented in this dissertation are representative of my activities in the field of resilience and privacy for ubiquitous mobile systems. They are organized according to three research areas : the use of reflection for the construction of dependable architectures, resilient mobiquitous systems (architectures, algorithms and evaluation) and geo-privacy.

The first research area concerns fault tolerance for distributed systems, from an architectural and language viewpoint. In this work, I have investigated the use of compile-time reflection to facilitate the implementation of fault tolerance mechanisms independently of the application. The use of reflection has been studied in the context of both theoretical work on multi-level reflexivity, and more practical work on the implementation of reflective off-the-shelf components in embedded software architectures.

The second research area concerns fault tolerance in mobile systems. My angle of attack was to consider mobility as an asset and not as a potential difficulty. This approach led me to explore the concept of geographical communication groups : how to define a group of communicating entities according to their respective location or as a function of their proximity. Then, I proposed a cooperative backup service, where the participating nodes offer a p2p secure storage service that they can use to back up their critical data. This approach was also followed to provide a virtual black box service for cars. These works were treated from both algorithmic and architectural viewpoints, and also in terms of analytical and experimental dependability assessment.

As part of my research on the resilience of mobiquitous systems, ethical issues were raised : how to exploit mobility data of individuals, while preserving their privacy ? This raised my interest in what can be called geo-privacy. This third area is now the major focus of my research, both in terms of attacks and protection. We propose a Markov mobility model as a compact, accurate, understandable and easily adaptable tool to represent the mobility of an individual. Based on this mobility model, we propose several attacks that target, e.g., prediction of future mobility, and de-anonymisation. Regarding the protection of geo-privacy, we are currently working on middleware-level abstractions, such as locanyms and location proofs to provide a secure and private architecture for location-based systems.

These works were linked with each other through a mixture of chance and serendipity, and the pursuit of a common goal : providing means for architectural and algorithmic resilience of current computer systems, namely distributed, mobile, ubiquitous systems. Many avenues of research are still open and are discussed.