Adaptation dans les systèmes dynamiques : une vision informatique de la résilience

Matthieu Roy

MÉMOIRE

En vue de l'obtention de

L'HABILITATION À DIRIGER LES RECHERCHES DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National Polytechnique de Toulouse (INP Toulouse)*

Présentée et soutenue le *11/06/2018* **par :**

MATTHIEU ROY

Adaptation dans les systèmes dynamiques : une vision informatique de la résilience

---

JURY

PASCAL FELBER	Université de Neuchâtel, Suisse	Rapporteur
RAYNOMD NAMYST	Université de Bordeaux, France	Rapporteur
PIERRE SENS	Université de Paris 6, France	Rapporteur
LAURENCE DUCHIEN	Université de Lille, France	Examinateur
ANTONIO FERNÁNDEZ ANTA	IMDEA, Madrid, Espagne	Examinateur
MAURICE HERLIHY	Brown University, RI, USA	Examinateur
PHILIPPE QUEINNEC	INP Toulouse, France	Examinateur
JEAN-CHARLES FABRE	INP Toulouse, France	Correspondant

École doctorale :
EDSYS : Informatique 4200018
Unité de Recherche :
LAAS-CNRS (UPR 8001)
Rapporteurs :
Pascal Felber, *Raynomd Namyst* et *Pierre Sens*

Préface

> Il n'existe rien de constant si ce n'est le changement
>
> Bouddha

La notion de résilience a été utilisée dans de nombreux domaines : en écologie, elle désigne la capacité d'un écosystème à retrouver un fonctionnement normal après une perturbation ; en psychologie, elle correspond à la récupération après un stress post-traumatique ; en art, elle mesure la capacité d'une œuvre d'art à conserver son attrait esthétique malgré les évolutions de la subjectivité.

En informatique, ce terme est généralement utilisé comme synonyme de tolérance aux fautes. Or, comme l'a justement remarqué Jean-Claude Laprie [1], le concept de résilience n'est pas statique comme la tolérance aux fautes, mais inclut une composante dynamique, imprévue voire imprévisible. Ainsi, il propose de définir la résilience comme « la persistance de la sûreté de fonctionnement en dépit des changements ».

Table des matières

Chapitre 1

Introduction

> Ne dis pas peu de choses en beaucoup de mots, mais dis beaucoup de choses en peu de mots
>
> Pythagore

Faire face à l'incertitude et aux changements

Considérons un moment les sources de l'informatique et son modèle le plus emblématique, la machine de Turing [2]. Ce concept abstrait d'un opérateur muni d'un ruban à cases infini, effectuant de manière répétée les actions « regarder la case en cours », « changer d'état et écrire la case en cours » et « changer de case » dans cet ordre définit de facto l'algorithmique, puis l'informatique, comme un processus intrinsèquement séquentiel.

Supposons qu'on puisse observer l'exécution d'un système tel un observateur omniscient, qui aurait accès —en continuant l'analogie avec une machine de Turing— à l'état global du système : état complet du ruban, connaissance de la machine à états et de la position de case observée. Même en connaissant l'état du système à tout instant, un observateur omniscient serait incapable de répondre à une question aussi simple que « l'exécution se terminera-t-elle ? ».

La science informatique ne s'intéresse pas, contrairement aux mathématiques, à l'ensemble des théorèmes (qu'ils soient prouvables ou non [3]), mais uniquement aux objets que l'on peut construire [4] avec un algorithme. Suivant cette vue constructiviste, il n'est pas raisonnable d'imaginer connaître l'état global complet d'une machine de Turing, ou d'un système informatique en exécution. Pis encore, cette vision « globale » n'est pas clairement définie si le système est réparti physiquement dans l'espace [5].

Or, les systèmes informatiques —probablement une des plus fascinantes créations de l'espèce humaine— envahissent le monde, le contrôlent de plus en plus et deviennent toujours plus connectés. Dans ce contexte, comment est-il possible pour la science informatique de fournir des garanties sur les services rendus par ces systèmes évoluant dans un environnement par définition changeant et non observable

de manière non équivoque ?

Ce mémoire décrit les travaux que j'ai conduits dans ce sens depuis plus de dix années passées au LAAS, le Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS.

Organisation du document

Dans la vision défendue dans ce mémoire, un système informatique résilient est un système fiable doté de capacités d'adaptation, fiables et adaptables elles aussi.

Il est commun de considérer un système adaptable comme une boucle de rétroaction organisée selon une séquence répétée des trois actions « observer », « calculer » et « contrôler », ou en anglais *SCC : "Sense", "Compute" and "Control"*.

Ce manuscrit est organisé en suivant cette terminologie dans le désordre.

Le chapitre 2 décrit les travaux que j'ai encadrés pour organiser un système informatique afin de pouvoir le contrôler et l'adapter à l'exécution. L'implémentation de l'adaptation à l'exécution y est déclinée selon deux axes : l'adaptation des mécanismes de tolérance aux fautes, et l'adaptation dans un système embarqué.

Le chapitre 3 se focalise sur la fourniture de capacités d'observation, et leur utilisation pour le contrôle de l'exécution. La vérification à l'exécution fournit des déclencheurs pour les mécanismes d'adaptation du chapitre 2.

Le chapitre 4 décrit des mécanismes algorithmiques pour le calcul fiable en environnement incertain. Deux sources d'incertitude y sont considérées : l'asynchronie, c'est-à-dire l'incertitude temporelle entre les entités composant le système, et les systèmes geo-localisés, où l'incertitude majeure est de nature spatio-temporelle.

De manière transverse, le chapitre 5 décrit les travaux conduits pour contenir et comprendre les changements dans les systèmes réels en s'intéressant à la dynamique induite par l'humain, à la fois créateur et utilisateur des systèmes informatiques.

Chaque chapitre se conclut par un bilan et une mise en perspective des travaux présentés.

Enfin, le chapitre 6 propose un ensemble de pistes de recherche en relation avec les travaux présentés.

Contributions majeures de ce document

Je tiens à me placer en porte à faux avec deux tendances actuelles dans le monde de la recherche : l'hyper spécialisation des chercheurs, et l'évaluation à outrance et essentiellement comptable des travaux, au détriment de leurs qualités. Je défends une vision de la recherche « traditionnelle », dans laquelle le croisement de résultats et idées de différents domaines permet de faire émerger des idées originales. Pour moi, les travaux majeurs de ce document sont souvent à la frontière de disciplines différentes, ou de branches de l'informatique. Est-ce parce que ce type de travaux, malgré les grands discours récurrents sur la nécessaire pluridisciplinarité de la recherche, sont difficiles à financer et à publier ? Est-ce la fierté de publier des articles après un grand nombre de refus au motif que le travail est intéressant mais n'est pas orthodoxe par rapport à telle ou telle communauté ?

S'il ne fallait retenir que quelques travaux sur ceux que je mène depuis plus de dix années au LAAS, je privilégierais les cinq thèmes suivants :

— les travaux sur l'adaptation des mécanismes de tolérance aux fautes [J5], résumés dans la Section 2.1, mêlant des concepts venant des architectures orientées services (SOA), des intergiciels, du génie logiciel et des systèmes critiques.
— la surveillance à l'exécution des logiciels pour les architectures à multiples niveaux de criticité [J2, C19]. Ces travaux relient architecture avancée des processeurs, langages, compilation et calcul de pire temps d'exécution dans une approche formelle, prouvée, et expérimentale : ils fournissent une approche élégante pour garantir la sûreté en s'adaptant à l'imprévisibilité des accès mémoire concurrents des architectures modernes, complexes.
— la mise en lumière du lien entre la concurrence minimale et l'asynchronie du système, à la croisée de la théorie de la concurrence, des systèmes répartis et des mécanismes de tolérance aux fautes active rencontrés dans la *state machine replication* [C14].
— le développement d'outils pour capturer les mobilités humaines afin de comprendre les sources de stabilité dans un environnement mobile. Des expérimentations permettant de générer des jeux de données, jusqu'à l'analyse des données pour en comprendre les motifs sous-jacents, ces travaux ont nécessité la coopération de chercheurs venant d'horizons variés (biologie, physique théorique, économie) [J6, C9, C27].
— l'étude des schémas généraux qui gouvernent l'évolution des réseaux induits par l'activité humaine [J4, C23] : l'approche propose des outils mathématiques utilisant les centralités, un concept utilisé en biologie et par les moteurs de recherche Internet, et propose une analyse statistique sur de nombreux exemples issus d'un domaine traditionnellement étudié par les physiciens.

Notes sur ce document et sur l'accessibilité des résultats

J'ai rédigé ce document en français, bien que la langue maintenant unique des chercheurs est l'anglais. Je défends très fortement la diversité linguistique dans le monde, car ce sont les différences subtiles entre les langues qui permettent de faire évoluer la connaissance. Mon expérience de l'anglais « worldish » tend à l'appauvrissement, non seulement de l'anglais, mais surtout de la pensée qui se standardise. Écrire en français a été finalement une difficulté que je n'avais pas anticipé, me montrant à quel point nous sommes maintenant formattés par l'anglais scientifique. Cela a pour moi été un acte de résistance au processus d'uniformisation et d'appauvrissement actuel.

Mon second acte de résistance visible dans le document concerne la visibilité et l'accessibilité des résultats. Je milite depuis de nombreuses années contre le modèle actuel de publication, basé sur des éditeurs scientifiques qui n'en ont que le nom et profitent de l'activité d'une communauté pour extorquer de l'argent aux organismes publics. Tous les articles que j'ai publiés et qui sont référencés dans ce document sont disponibles librement. Le mémoire électronique utilise des liens hypertexte qui renvoient à ces publications libres d'accès.

Ainsi, le but du mémoire lui même est de fournir un résumé le plus concis possible de mes travaux afin de fournir une vision informatique de la résilience. Le lecteur

pourra, pour plus de détails, suivre la référence idoine qui le mènera au résumé puis à la version libre de la description des travaux concernés. Pour des raisons légales, ce mécanisme est limité aux articles dont je suis un des auteurs.

Chapitre 2

Organiser : Architectures pour l'adaptation

> Choisissez un travail que vous aimez et vous n'aurez pas à travailler un seul jour de votre vie
>
> Confucius

A priori, un système informatique est prévu pour fonctionner dans un environnement donné. Si l'on désire le faire évoluer durant sa vie opérationnelle, ce qui semble incontournable pour les systèmes qui ne doivent pas être arrêtés, alors son architecture doit avoir été conçue dans ce sens. Si l'on considère maintenant un système sûr de fonctionnement [6] qui doit évoluer, s'adapter à de nouveaux types de défaillance ou à l'évolution des ressources disponibles, cette évolution doit se faire sans mettre en défaut les propriétés de sûreté de fonctionnement. D'un point de vue de la tolérance aux fautes, on peut ainsi rapprocher la résilience de la tolérance aux fautes adaptative [7].

2.1 Conception de l'architecture pour l'adaptation

Dans le cadre des travaux de thèse de Miruna Stoicescu [Th9], nous avons proposé et évalué une approche pour développer des systèmes *sûrs de fonctionnement agiles*. Par opposition aux travaux existants qui proposent l'adaptation des mécanismes de sûreté de fonctionnement en utilisant un approche préprogrammée [8, 9], notre proposition autorise toute modification lors de l'exécution en utilisant des concepts et mécanismes issus du génie logiciel : les concepts qui sous-tendent ce travail sont inspirés des Architectures Orientées Services [10], de la programmation orientéee aspect [11] et des intergiciels réflexifs à base de composants [12].

2.1.1 Modèle de fautes, modèle d'évolution

Afin de poser un cadre théorique à notre approche, la première nécessité pour appréhender l'adaptation d'un système aux changements est de se fixer un cadre

d'évolution, un modèle des changements qui peuvent survenir lors de l'exécution.

Pour formaliser le problème de l'évolution des mécanismes de tolérance aux fautes (*FTM –Fault Tolerance Mechanism*), nous avons tout d'abord dû définir clairement dans quelle mesure un système peut évoluer durant sa vie opérationnelle. En se focalisant uniquement sur la tolérance aux fautes, on peut isoler trois paramètres ou axes essentiels du système [C37] pouvant être amenés à évoluer pendant l'exécution :

— Le modèle de fautes considéré (FT),
— Les caractéristiques de l'Application (A), dont les plus importantes vis-à-vis de la tolérance aux fautes sont *i*) le déterminisme de l'application, *ii*) l'existence ou non d'un état de l'application en vue de sa restauration en cas de défaillance, *iii*) l'accès à l'état de l'application,
— Les Ressources disponibles (R) (bande passante, énergies, ressources de calcul).

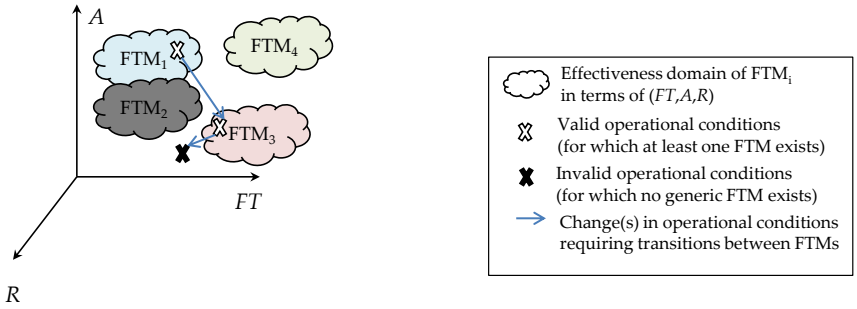


FIGURE 2.1 – Modèle de référence et chemin d'évolution des mécanismes de tolérance aux fautes

À tout instant, le système considéré se trouve sur un point de cet espace, et une exécution du système correspond à un chemin dans cet espace. De plus, un mécanisme de tolérance aux fautes donné ne fonctionne correctement que dans un domaine qui lui est associé dans le repère *(FT, A, R)*, schématisé par les « nuages » de la Figure 2.1.

Lors d'une évolution des conditions opérationnelles, représentée par une flèche bleue sur la Figure 2.1, si le système doit sortir du domaine de validité du mécanisme de tolérance aux fautes considéré, il est nécessaire de changer dynamiquement ce mécanisme.

Notre approche pour mettre en œuvre cet élément essentiel de la résilience, à savoir l'adaptation en-ligne des FTMs, s'est effectuée en plusieurs étapes. Tout d'abord, nous avons identifié comme indiqué ci-dessus les paramètres qui dictent le choix d'un FTM particulier parmi un ensemble de mécanismes de tolérance aux fautes disponibles. Les valeurs de ces paramètres indiquent, au moment de la conception, quel est le FTM le plus approprié à adjoindre à une application, si un tel FTM existe. La variation de ces paramètres peut invalider ce choix initial et doit déclencher une transition vers un nouveau FTM, cohérent avec les nouvelles valeurs des paramètres *(FT, A, R)*. Ces paramètres représentent ainsi un référentiel qui permet de capturer la dynamique nécessaire pour permettre l'évolution des FTMs.

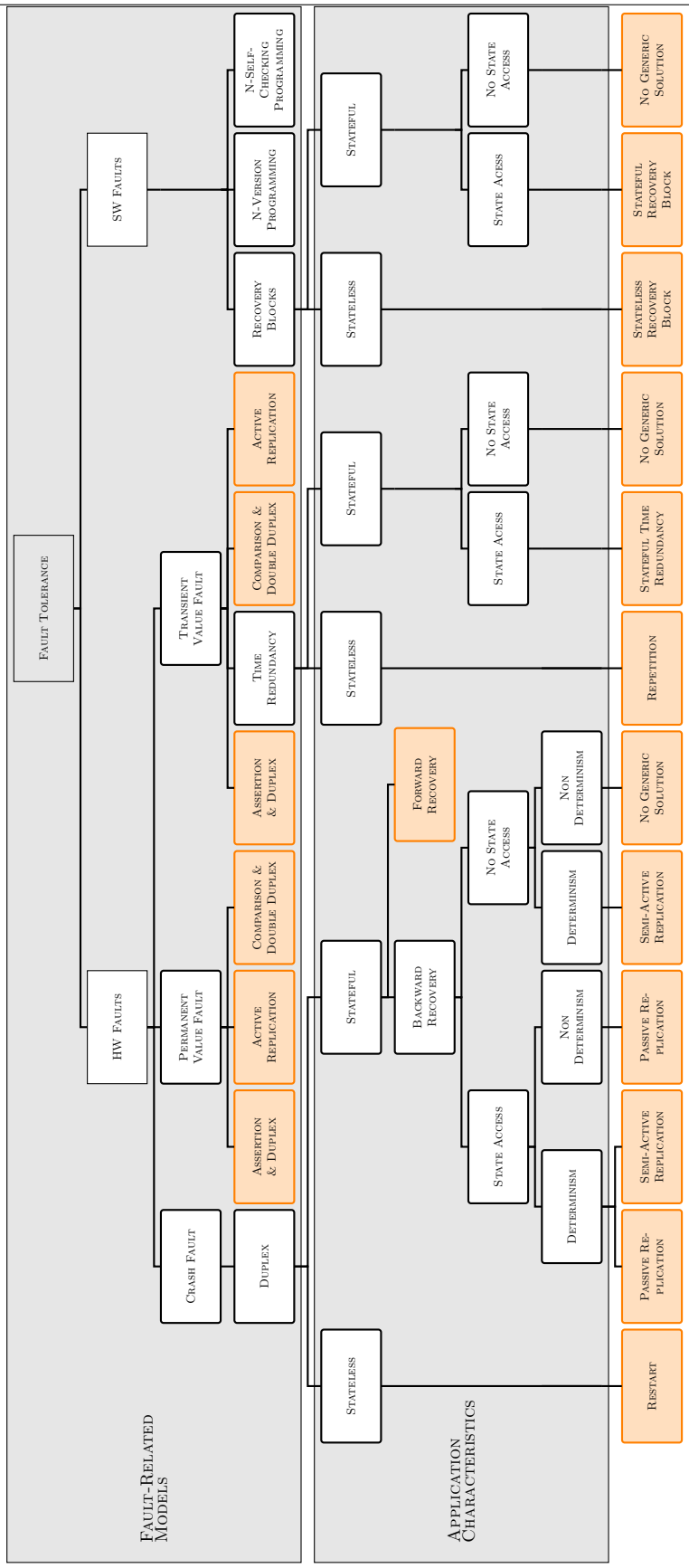


FIGURE 2.2 – Classification des FTM selon le modèle de fautes et les caractéristiques de l'application

En suivant ce référentiel, nous avons analysé et classé un ensemble représentatif des mécanismes de tolérance aux fautes les plus utilisés [C35], comme représenté sur la Figure 2.2.

2.1.2 Factorisation des mécanismes de tolérance aux fautes

Suite à la classification des différents mécanismes pour la tolérance aux fautes en fonction des paramètres opérationnels du système, nous nous sommes focalisés sur l'architecture à mettre en œuvre afin de permettre des transitions fiables et rapides entre différents mécanismes de tolérance aux fautes.

La première étape vers une architecture permettant l'adaptation des mécanismes de tolérance aux fautes à l'exécution a été l'analyse détaillée de la structure et du fonctionnement d'un ensemble de FTMs. Le but de cette analyse est d'identifier leurs éléments communs et leurs points de variabilité, afin d'offrir une adaptation en-ligne à grain fin.

Cette étape d'analyse est la pierre angulaire d'une approche de « conception pour l'adaptation » : en isolant les caractéristiques communes (factorisation) et les points variables, on minimise les modifications à apporter lors d'une transition d'un mécanisme à l'autre.

De manière intéressante, cette analyse permet non seulement l'adaptation efficace à l'exécution, mais est aussi utile lors du développement d'un mécanisme de tolérance aux fautes car elle permet d'identifier les briques de base fondamentales communes à de nombreux mécanismes.

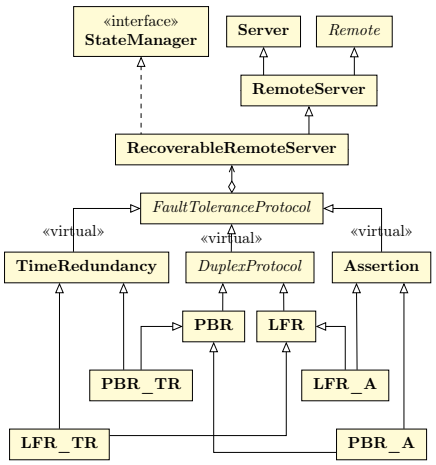


FIGURE 2.3 – Extrait du framework de patrons de conception pour la tolérance aux fautes. LFR : *Leader-Follower Replication*, PBR : *Primary Backup Replication*

Tout mécanisme de tolérance aux fautes est ainsi implémentable dans une « forme normale » qui découpe le traitement en trois phases distinctes, *before, proceed, after*, qui correspondent respectivement aux traitements préparatoires, à la mise en œuvre effective du mécanisme de tolérance aux fautes, et au traitements de consolidation du mécanisme de tolérance aux fautes [J5].

Le résultat final de ces travaux est un système de patrons de conception (*design*

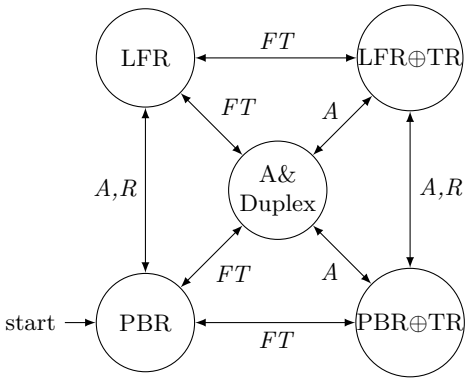


FIGURE 2.4 – Transitions possibles entre les FTMs. Les labels sur les arcs correspondent aux changements qui induisent la transition. LFR : *Leader-Follower Replication*, PBR : *Primary Backup Replication*, TR : *Time Redundancy*, A&Duplex : *Assertion & Duplex*

patterns) pour la tolérance aux fautes [C26], dont une partie est représentée sur la Figure 2.3.

2.1.3 Intergiciel pour l'adaptation de la tolérance aux fautes

Transitions L'étape suivante a été d'exploiter des outils spécifiques pour l'adaptation à l'exécution, c'est-à-dire un intergiciel réflexif à base de composants, et projeter les patrons de conception pour la tolérance aux fautes identifiés sur cette architecture à composants. Ici encore, les choix de conception sont extrêmement importants car ils vont déterminer la granularité des modifications en-ligne : les caractéristiques variables doivent être projetées sur des composants individuels afin de pouvoir les modifier à l'exécution, pour profiter de l'analyse faite lors de la « conception pour l'adaptation ».

Nous avons défini et mis en œuvre des scénarios de transition, comme ceux de la Figure 2.4, pour prouver la faisabilité de notre approche et illustrer la dynamique qu'apportent ces outils provenant du génie logiciel à la gestion des mécanismes de tolérance aux fautes. L'approche a été évaluée en comparant le temps et l'espace utilisés pour effectuer une transition différentielle entre deux mécanismes avec ceux nécessaires dans le cas d'une évolution monolithique

Ce travail se focalise uniquement sur l'adaptation des mécanismes de sûreté de fonctionnement (aspects non fonctionnels). Une modification doit donc pouvoir se faire sans violer les contraintes de sûreté du système, ce qui impose en particulier des algorithmes de transition fiables.

Support système minimal nécessaire Au delà des aspects techniques de la mise en œuvre, ce travail a permis de définir le support système minimal nécessaire à la mise en œuvre de l'approche différentielle tolérante aux fautes [J5]. Ce support minimal doit inclure :

— le contrôle sur le cycle de vie des composants : `start`, `stop`, `add`, `remove`,
— l'accès à l'état des composants,
— le contrôle des interactions entre les composants,

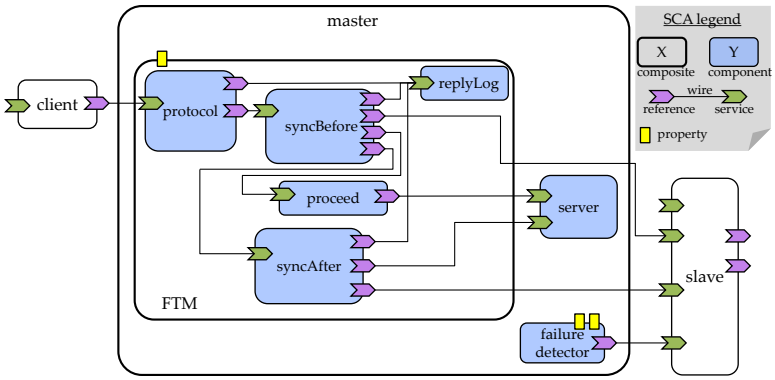


FIGURE 2.5 – Architecture à base de composants de PBR *Primary-Backup Replication*

— l'atomicité dans les scripts de transition.

L'atomicité est un point critique : l'aspect transactionnel des modifications qui sont effectuées garantit la tolérance aux fautes du processus d'adaptation lui-même.

La Figure 2.5 montre l'architecture à composant obtenue pour un mécanisme de réplication de type primaire-secondaire.

Adaptation en-ligne et préparation hors-ligne Ces travaux montrent que l'adaptation à l'exécution doit être séparée en deux phases : la résilience à froid, qui inclut toutes les opérations à effectuer hors ligne (développement de mécanismes et de leurs transitions), et la résilience à chaud, comprenant l'adaptation en elle même et toutes les interactions faites en ligne.

La Figure 2.6 représente ces deux facettes de la résilience et leurs liens. La partie hors ligne (résilience froide), fournit des packages en vue de l'adaptation future du système. Les mesures à l'exécution (résilience chaude) vont fournir des données pour enrichir et améliorer la partie hors ligne, et ainsi former une boucle de rétroaction.

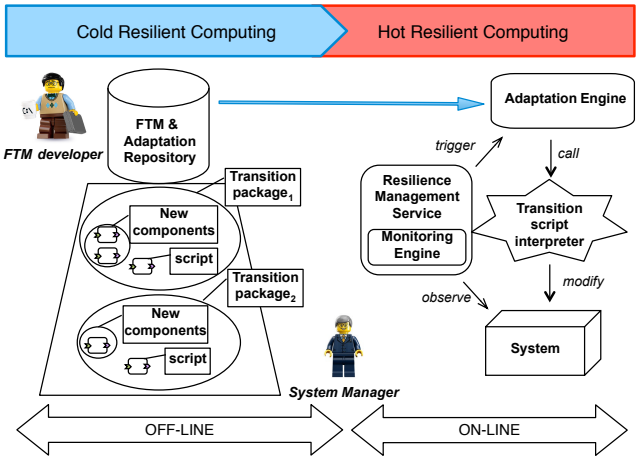


FIGURE 2.6 – Architecture à composants pour la résilience : découpage hors-ligne et en-ligne

Évaluation Enfin, nous avons évalué la facilité d'utilisation de ce processus dans deux exemples d'intégration. Dans le premier, nous nous sommes intégrés à un processus de développement dirigé par la conception dédié aux applications *Sense-Compute-Control*. Cette suite logicielle, `DiaSuite`, n'incluait pas initialement de support pour la tolérance aux fautes. Cette intégration a montré que l'approche est suffisamment générique pour être attachée à un processus de développement existant, et a permis d'ajouter des exigences de tolérance aux fautes de manière transparente pour les utilisateurs du système.

La deuxième intégration a été l'ajout des mécanismes de tolérance aux fautes adaptatifs dans Srijan, un kit de développement pour les réseaux de capteurs sans fil, donc dans un environnement à ressources très limitées. Le travail de conception en amont (factorisation, minimisation des modifications à apporter lors d'une transition) permet de minimiser les ressources utilisées, ce qui a autorisé une intégration dans un environnement contraint en ressources.

La première intégration [C26] a été conduite dans le cadre du projet SERUS [Pr13] avec les centres INRIA de Bordeaux et de Lille. La seconde intégration a donné lieu à une étude de cas sur des réseaux de capteurs en collaboration avec INRIA centre Rocquencourt [C24] dans le cadre du projet ANR Blanc MURPHY [Pr12]

2.2 Rendre adaptable une architecture existante

2.2.1 Pourquoi rendre adaptable un système existant

La section précédente présentait des mécanismes pour concevoir un système en vue de son adaptabilité ultérieure. Bien entendu, les solutions proposées ne sont pas applicables dans tous les systèmes. Dans le cadre d'un système temps réel embarqué, par exemple, une solution basée sur un intergiciel à composant n'est pas envisageable. Cependant, les mises à jour et la personnalisation des systèmes embarqués sont de plus en plus demandées par leurs utilisateurs, voire nécessaires. En effet, la complexité grandissante des systèmes exige de déployer des moyens pour permettre leur maintenance et leur évolution lors de leur vie opérationnelle.

Dans le cadre des travaux de thèse d'Hélène Martorell [Th8], effectués dans le cadre d'une CIFRE avec Renault [Pr7], nous avons exploré les possibilités d'inclure des mises à jour et de l'adaptation dans le contexte industriel des systèmes enfouis dans l'automobile qui suivent le standard AUTOSAR.

2.2.2 Un cas concret : AUTOSAR

Un des inconvénients majeurs de l'architecture AUTOSAR est son manque de flexibilité, une contrepartie qui semble nécessaire pour assurer la prédictibilité et garantir les propriétés temps-réel du système. Nous avons étudié les modifications nécessaires pour intégrer des mises à jour partielles dans le cadre du standard AUTOSAR, aux niveaux architecture [C29] et processus de développement [C21].

Les containers : prévoir l'adaptation future. Dans le processus de développement d'AUTOSAR, le système est compilé et chargé sur le calculateur en un bloc

monolithique et, pour faire une modification après le déploiement initial, la seule solution est de recompiler et de recharger totalement le code binaire.

Pour ajouter des mécanismes d'adaptation à l'exécution du logiciel, nous avons proposé une approche *pré-câblée*, offrant des capacités d'adaptation déterminées au moment de la compilation. La première étape consiste en l'analyse de l'espace possible pour l'adaptation, en particulier en temps de calcul et ressources mémoire disponibles. Nous avons proposé le mécanisme de *container* [C29], un composant logiciel initialement vide, inclus lors de la compilation du système.

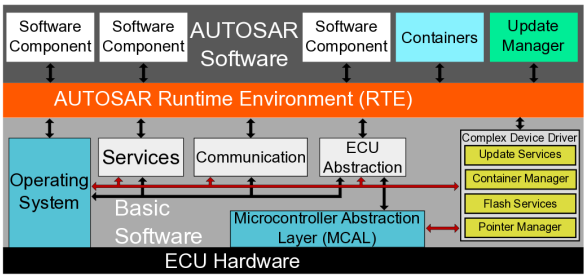


FIGURE 2.7 – Architecture AUTOSAR modifiée pour l'adaptation à l'exécution

Dans la version pré-câblée simple, chaque *container* embarque un budget temps, mémoire et une fréquence d'activation définie statiquement. L'analyse de l'ordonnancement est faite une fois pour toutes à la compilation, en utilisant ces données statiques. Lors de l'exécution, il est possible d'ajouter un nouveau composant si celui-ci entre dans un des containers disponibles. Dans ce cas, le gestionnaire de mise à jour (*Update Manager* sur la Figure 2.7, qui s'appuie sur les services représentés en jaune) charge le composant, puis libère éventuellement un autre composant s'il s'agit d'une mise à niveau.

Dans une version plus adaptative [C12], les budgets temps et mémoire sont partagés entre tous les containers. Lors de l'ajout d'une fonctionnalité, on vérifie d'abord si le budget temps et mémoire le permet. Si c'est le cas, le système calcule la matrice de précédence entre les tâches du système et vérifie si la mise à jour est compatible, puis vérifie si le système reste ordonnançable en effectuant une analyse de sensibilité, inspirée du facteur d'élasticité défini par Bini et al. [13]. Le facteur d'élasticité permet de vérifier sans recalculer l'ordonnancement si la tâche peut être ajoutée tout en respectant les contraintes de précédence.

On obtient ainsi une méthode efficace pour modifier à la volée le code exécuté par un système AUTOSAR, mais les difficultés liées à la rigidité de ces systèmes [Th8] appellent à plus de flexibilité dans le futur standard Adaptive AUTOSAR.

2.3 Bilan et notes

Pour permettre l'adaptation à l'exécution d'un système, un ensemble de mécanismes doivent être inclus dès son déploiement. Les travaux présentés dans ce chapitre montrent que la prise en compte le plus tôt possible, dans la conception d'un système, des futures capacités d'adaptation autorise une plus grande maîtrise et de meilleures performances de cette adaptation.

Dans le cadre des travaux de thèse de M. Stoicescu [Th9] pour le projet ANR MURPHY [Pr12], nous avons considéré l'adaptation comme une composante primaire d'un système résilient, et nous avons proposé et évalué une architecture logicielle basée composants pour implémenter l'adaptation sûre des mécanismes de tolérance aux fautes.

Les points marquants de ces travaux sont :

— la formalisation de l'évolution des paramètres impactant la validité des mécanismes de tolérance aux fautes dans un référentiel décrivant le modèle de fautes, les caractéristiques de l'application et les ressources disponibles, et la classification des mécanismes de tolérance aux fautes suivant ce référentiel,
— l'analyse de nombreux mécanismes de tolérance aux fautes pour factoriser leurs points semblables, et normaliser leur conception sous la forme d'un motif *before, proceed, after*,
— la détermination du support système minimal pour permettre une adaptation à l'exécution fiable,
— le découpage clair du processus d'adaptation en une partie hors ligne, la résilience froide, et une partie exécutive, la résilience chaude,
— l'implémentation et l'évaluation de l'approche dans un intergiciel à composants, et l'intégration dans deux environnements de développements.

Dans le cadre d'une collaboration avec Renault [Pr7] et de la thèse d'Hélène Martorell [Th8], nous avons développé des mécanismes permettant d'ajouter des capacités d'adaptation à un système basé sur le standard automobile AUTOSAR. Le mécanisme de *container* développé dans ces travaux permet d'inclure des applications vides dans ce système où tout doit être défini à la compilation, et permet de fournir certaines capacités d'adaptation à l'exécution. Nous avons montré que les contraintes d'ordonnancement temps réel peuvent être prises en compte de manière efficace en effectuant une analyse de sensibilité, ce qui permet de modifier le système à l'exécution tout en garantissant l'ensemble des contraintes temporelles.

De nombreuses publications sont associées à ces travaux [C12, C21, C24, C26, C29, C35, C37], et deux thèses sont actuellement en cours au LAAS pour poursuivre ce travail dans des environnements embarqués [C6, C8, C10]. Les travaux sur l'adaptation dans AUTOSAR sont la source de nombreuses interactions avec des acteurs majeurs de l'automobile pour la définition du futur standard Adaptive AUTOSAR. Les résultats majeurs du travail sur l'architecture pour l'adaptation ont été publiés dans ACM CBSE [C26] et dans le Journal of Software Architectures [J5].

Chapitre 3

Observer : Vérification à l'exécution de propriétés

> Observez l'ordre naturel des choses. Travaillez avec lui plutôt que contre lui, car essayer de changer ce qui est ne pourra que faire surgir une résistance.
>
> Lao Tseu

Pour affirmer qu'un système donné est fiable, il est nécessaire de vérifier son bon fonctionnement. Une approche communément utilisée dans les systèmes critiques est la vérification statique de programmes. Nous nous sommes focalisés dans nos travaux sur un aspect complémentaire à la vérification statique, la *vérification à l'exécution*, qui consiste à surveiller le système pendant son exécution et à veiller à ce qu'il se conforme bien à une spécification donnée. La vérification à l'exécution peut aussi générer des évènements pour déclencher une reconfiguration du système.

Ce chapitre se focalise sur la vérification à l'exécution de propriétés liées à la sûreté, et met particulièrement l'accent sur les aspects temporels : la section 3.1 introduit la vérification de contraintes temporelles à partir d'automates temporisés, la section 3.2 ajoute la répartition du calcul, en se basant sur des réseaux de Petri ; la section 3.3 traite de l'utilisation de la vérification en ligne de temps d'exécution pour garantir les propriétés temporelles dans un système temps réel multicœurs.

3.1 Exprimer et vérifier des contraintes temporelles

3.1.1 Observation à l'exécution d'un système

Historique et processus d'observation

Une étape essentielle dans la conception d'un système adaptatif est sa capacité d'observation. Nous nous focalisons ici sur l'auto-observation, c'est-à-dire les mesures sur l'état du système que le système peut calculer par lui-même. Les travaux décrits ici ont été conduits lors des travaux de thèse de Thomas Robert [Th10].

Chaque processus d'un système donné étant séquentiel, il est naturel de considérer qu'un historique local calculable est une séquence d'évènements. Plus précisément, pour introduire la notion de temps sur un processus localisé dans l'espace, indépendemment d'une référence extérieure, nous avons basé nos travaux sur la notion de trace temporisée [14].

Définition 1 (Trace temporisée) *Soit Σ l'ensemble des évènements observables. Une* trace temporisée *u est une séquence alternée d'évènements e_i et de durées t_i :*

$$u = t_1.e_1.t_2.e_2.\ \ldots\ t_\ell.e_\ell \qquad \textit{telle que} \qquad \ell \geq 1, \forall i \in [1,\ell], e_i \in \Sigma \wedge t_i > 0$$

Comme nous l'avons montré lors des travaux de thèse de Thomas Robert [Th10], l'ensemble des traces temporisées peut être muni d'opérations similaires aux opérations sur les langages rationnels (concaténation, union ensembliste, étoile de Kleene). Dans ce résumé nous n'utiliserons que le concept fondamental de préfixe. Une trace u_1 est préfixe d'une trace u_2 s'il existe une trace v telle que $u_2 = u_1.v$.

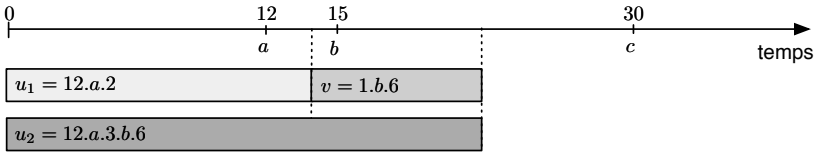


FIGURE 3.1 – Traces temporisées : u_1 est préfixe de u_2

Du point de vue de l'observation de l'état du système, la notion de préfixe est essentielle : à tout instant, le système de vérification n'aura à sa disposition qu'un préfixe de l'exécution totale.

Détection d'erreur à partir d'une spécification

Comment définir si un système est dans un état correct ou dans un état erroné ? En suivant les définitions données ci-dessus, on peut définir l'ensemble des états acceptables du système comme un ensemble de traces temporisées correspondant à une spécification $\mathcal{S}$ (cet ensemble est habituellement infini, et la partie 3.1.2 précisera une possibilité de description de cet ensemble par un automate temporisé fini).

Lors de l'exécution, l'observation du système est une trace temporisée qui est toujours *finie*. Il ne sera donc pas possible, à partir de cette information partielle, de détecter la violation d'une propriété telle que « au bout d'un moment, une action a va être exécutée ». Ce type de propriété de vivacité étant exclu, nous pouvons définir les symptômes d'erreur associés à $\mathcal{S}$:

Définition 2 (Symptômes d'erreur) *Les symptômes d'erreur associés à une spécification $\mathcal{S}$ sont l'ensemble des traces finies qui n'appartiennent pas à $\mathcal{S}$.*

Monotonie de la vérification, détection au plus tôt

On peut remarquer que le processus de détection d'erreur est monotone : si s est un symptôme d'erreur, alors toutes ses continuations finies sont des symptômes d'erreur (de la même manière, tout préfixe d'une exécution correcte est correct).

Un problème pratique essentiel pour la détection d'erreur est celui de la détection *au plus tôt* : étant donné une spécification temporisée $\mathcal{S}$, calculer les plus petits symptômes d'erreur, c'est-à-dire caractériser les états qui nous indiqueront de manière sûre que le système dévie de son comportement normal dès que c'est possible.

Définition 3 (Détection au plus tôt) *L'ensemble des plus petits symptômes d'erreur pour $\mathcal{S}$ est :*

$$S_{\min}(\mathcal{S}) = \{u : \forall m, u.m \notin \mathcal{S} \qquad \textit{(défaillance)}$$
$$\wedge\, \forall (v,w) : u = v.w, \exists m : v.m \in \mathcal{S}\} \qquad \textit{(minimalité)}$$

Soit la propriété P de « réponse en délai borné » suivante : tout événement a est suivi d'un événement b dans les 2 ms. Dans l'exemple de la Figure 3.1, u_1 est un plus petit symptôme d'erreur pour la propriété P et, en observant u_1, on sait que le système ne satisfera pas la propriété P.

Il est important de comprendre dans le cas des systèmes temps-réels qu'une erreur peut survenir lorsqu'aucun évènement n'a eu lieu, par exemple parce qu'une horloge a dépassé une certaine échéance (*deadline*). La définition ci-dessus semble similaire à celle donnée dans [15] mais présente la différence fondamentale suivante : elle autorise la description de fragments d'exécution ne finissant pas obligatoirement avec un événement. Ainsi, contrairement à d'autres travaux similaires, notre définition permet d'englober *explicitement* les deux causes pratiques de violation d'une propriété P [C43] :

— Une défaillance sur un événement interdit
— Une défaillance sur dépassement d'échéance (*deadline*).

3.1.2 Synthèse d'un détecteur

Automates temporisés

L'expression des comportements autorisés, la spécification du système $\mathcal{S}$, a été faite en utilisant des automates temporisés [16]. À titre d'exemple, considérons un protocole d'accès à un périphérique.

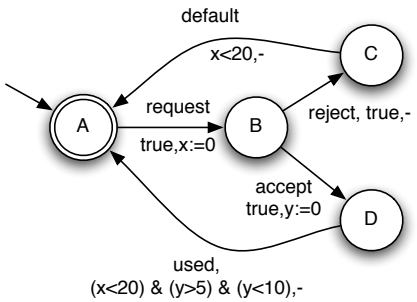


FIGURE 3.2 – Protocole d'accès à un périphérique temps réel.

Il y a deux comportements décrits dans ce modèle :

1. Si la ressource est obtenue (accept), elle doit être utilisée pendant **au moins** 5 ms (contrainte physique pour stabiliser l'action) et **au plus** 10 ms (contrainte d'équité). Dans les deux cas, il s'agit d'une contrainte sur l'horloge y.

2. Le traitement complet d'une requête doit prendre au plus 20 ms (contrainte sur l'horloge x).

Si l'on considère l'exécution tronquée 20.request.16.accept, on se retrouve dans l'obligation de violer l'une des deux contraintes : soit le traitement complet se fait bien en moins de 20ms mais la ressource n'est pas contrôlée pendant assez longtemps (moins de 4 ms), soit la boucle complète prend plus de 20 ms. Bien qu'il y ait une transition de D vers A, la transition sortant de D est définitivement inhibée si, lorsque accept a lieu, l'horloge x dépasse 15 ms.

Génération d'un détecteur à partir d'une spécification

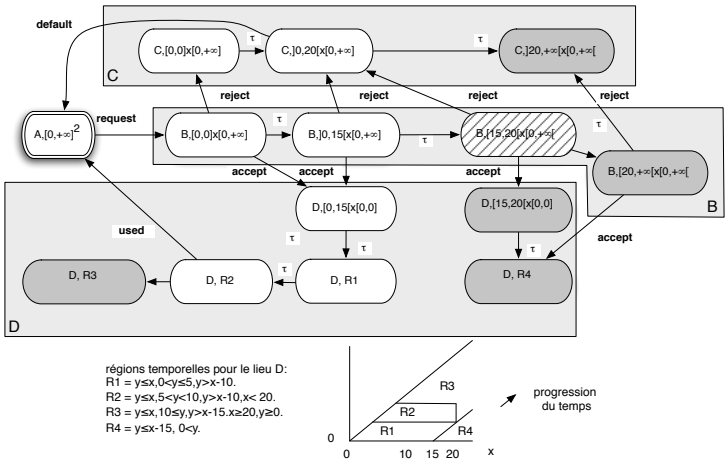


FIGURE 3.3 – Abstraction du graphe des configurations : le graphe quotient

Le graphe des régions [16] permet d'abstraire le temps en groupant les transitions temporelles par familles et les configurations par régions. Abstraire le graphe des configurations revient à définir une procédure telle que l'on puisse générer un graphe fini, appelé graphe quotient [17], permettant de déterminer pour chaque nœud si ce dernier est co-accessible, c'est-à-dire reste dans la spécification. Lors des travaux de thèse de Thomas Robert [Th10], nous avons utilisé le graphe quotient pour obtenir un automate compact, utilisable efficacement à l'exécution pour détecter les plus petits symptômes d'erreur et implémenter la détection au plus tôt. La Figure 3.3 montre l'automate obtenu à partir de la spécification de la Figure 3.2. On peut remarquer que la trace évoquée ci-dessus 20.request.16.accept mène à un état interdit en gris : le graphe calculé permet de « déplier » la spécification pour identifier les plus petits symptômes d'erreur.

Utilisation de la détection pour le recouvrement

Le dernier point important de ces travaux a été de coupler la détection d'erreur au plus tôt avec une procédure de recouvrement.

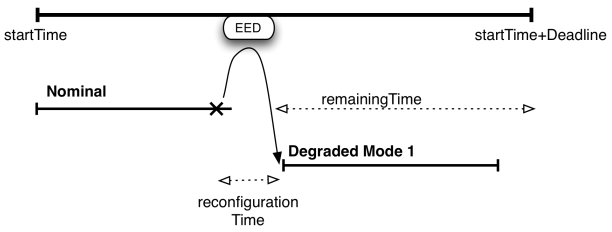


FIGURE 3.4 – Mode dégradé d'exécution

L'approche, schématisée sur la Figure 3.4, se base sur les mécanismes de détection d'anomalie décrits précédemment et générés automatiquement à partir d'une description formelle du système sous la forme d'un automate temporisé pour offrir un mécanisme de recouvrement d'erreur dans un système temps réel.

On suppose qu'un système embarque un ensemble d'implémentation variées d'un même service avec des qualités de service différentes : un mode nominal, et un ensemble de modes dégradés dont les temps d'exécutions au pire cas sont inférieurs au mode nominal. Nous avons développé un système de surveillance qui, dès que le service nominal dévie de sa spécification, lève une alarme. L'intérêt de ce système est qu'il donne, en plus de la présence d'une erreur, le temps d'exécution qu'il reste avant que le système n'arrive dans un état erroné, et implémente donc une forme *proactive* de la tolérance aux fautes. Le système est capable à la volée de basculer vers un mode dégradé dans le temps restant, et, si cela n'est pas possible, de lever une exception globale au niveau système [C40, C42].

3.2 Exprimer une contrainte temporelle et répartie

La suite de ces travaux a été conduite lors de la thèse d'Olivier Baldellon [Th7]. Ces travaux ont porté sur la *distribution* de la vérification selon deux axes, d'une part la vérification distribuée d'une propriété, et d'autre part la vérification de propriétés distribuées.

Réseaux de Petri temporisés

Les formules logiques et les automates sont des abstractions intrinsèquement centralisées qui se prêtent assez mal à l'expression de propriétés réparties et à leur évaluation décentralisée. Les réseaux de Petri n'ont pas cette limitation : leur sémantique ne dépend que des évènements, de leur ordre et du temps. Pour utiliser les réseaux de Petri A-temporisés dans le cadre de la vérification à l'exécution de propriétés réparties, nous avons étendu ce formalisme pour rendre possible son évaluation dans un contexte d'observation imparfaite, en particulier concernant l'ordre dans lequel les évènements observables du système peuvent êtres reçus par un système de supervision.

Ces travaux se basent sur trois contributions majeures : le mécanisme de jetons négatifs, la définition de propriétés à vérifier suivant une approche par assemblage de motifs, et la transcription automatique vers un ensemble d'algorithmes distribués.

Jetons négatifs dans les réseaux de Petri

Pour être capable de vérifier une propriété distribuée nous avons proposé le concept de *jeton négatif*, qui permet une exécution spéculative d'un réseau de Petri, condition nécessaire à son exécution en ligne [C31]. Intuitivement, autoriser des jetons négatifs dans un réseau de Petri permet d'exécuter la vérification au fur et à mesure que les évènements animant le réseau arrivent sur le détecteur. La vérification des contraintes temporelles et la validité de la vérification sont effectuées a posteriori.

Expression de propriétés par assemblage de motifs

La plupart des approches développées dans d'autres équipes partent d'une formule donnée, et le travail consiste à la découper en sous-parties vérifiables dans des sous-composants distribués. Il s'agit d'approches top-down. En revanche, dans notre approche, nous proposons une conception modulaire, bottom-up, des propriétés à vérifier : nous autorisons l'écriture de propriétés complexes à partir de propriétés simples en les composant (approche modulaire). Cela a permis de développer une bibliothèque de motifs composables classiques (causalité, synchronisation, k parmi n, mode normal et dégradé, etc.) rendant l'expression des propriétés à vérifier plus aisée [Th7].

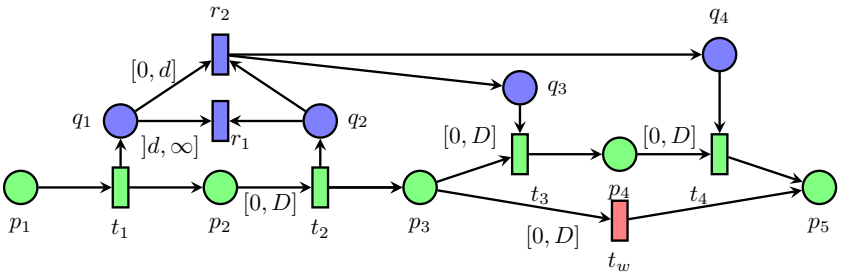


FIGURE 3.5 – Réseau de Petri assemblé pour la vérification d'un système ferroviaire [C25], composé d'un mode nominal en vert, et d'un mode dégradé en bleu

Implémentation sur un système réparti

La dernière étape de ce travail a été le développement de MINOTOR [C25], un compilateur qui permet la génération automatique et le déploiement du système de supervision. L'utilisateur décrit une spécification à vérifier sous forme d'un réseau de Petri, et fournit la description physique du système : topologie du système, propriétés de synchronie et sources des évènements surveillés. À partir de ces deux descriptions, l'outil MINOTOR génère automatiquement les différentes parties à distribuer du programme de vérification, et effectue son déploiement sur le réseau.

3.3 Vérification à l'exécution pour garantir la sûreté

Dans la continuité des travaux décrits ci-dessus, on peut remarquer que la définition même de *résilience*, la continuité de la sûreté de fonctionnement en dépit des changements, peut être implémentée en informatique comme un système de surveillance couplé à une adaptation en vue de garantir la sûreté du système. Cette partie se concentre sur des travaux visant précisément une telle implémentation, en se focalisant sur l'aspect temporel des systèmes temps-réel exécutés sur une plateforme multicœurs.

Garanties temporelles & WCET *vs.* multicœurs

Les systèmes temps-réel critiques se doivent de garantir des propriétés temporelles strictes, et en particulier doivent garantir que les tâches s'exécutent toutes avant une échéance temporelle donnée par la spécification. Ces garanties sont de manière classique prouvées en calculant pour chaque tâche son pire temps d'exécution (*WCET — Worst Case Execution Time*), et en montrant qu'il est possible d'ordonnancer toutes les tâches à exécuter sur l'architecture matérielle donnée en respectant les dépendances entre tâches et les contraintes d'échéances (*deadlines*).

Avec l'arrivée des architectures multicœurs, ce modèle consistant à évaluer le pire temps d'exécution de chaque tâche en isolation puis de prouver qu'il existe un ordonnancement compatible a montré ses limites. En effet, les propriétés temporelles de l'exécution ne *composent* pas aussi simplement : lors d'un accès mémoire par une tâche s'exécutant sur un cœur, le pire scénario pour cet accès est qu'il est en concurrence avec d'autres accès sur les autres cœurs, ce qui génère une contention sur le bus d'accès mémoire et donc une mise en attente dictée par la politique d'accès du bus mémoire. Ainsi, les pire temps d'exécution calculés de manière classique seront très élevés pour une telle architecture, rendant l'intérêt des multicœurs pour le cadre temps-réel très limité.

Garanties dynamiques sur les temps d'exécution

Dans le cadre d'un groupe de travail sur les systèmes temps-réel fiables [Pr14], nous nous sommes intéressés, lors des travaux post-doctoraux d'Angeliki Kritikakou, aux mécanismes *dynamiques* permettant de contourner ces problèmes de calcul trop pessimistes des pire temps d'exécution sur les multicœurs. Le problème principal étant la grande variabilité des temps d'accès mémoire due au partage par les différents cœurs, nous nous sommes focalisés sur la surveillance à l'exécution des propriétés temporelles des applications.

Ainsi, plutôt que de calculer un pire temps d'exécution pour une tâche donnée, par définition très pessimiste, nous avons proposé de calculer le pire temps d'exécution restant en différents points d'un programme, et, à l'exécution, de comparer le temps effectif d'exécution (dynamique) avec le pire temps (statique) restant au point courant.

Plus précisément, le but recherché est de limiter le pessimisme extrême des approches classiques temps-réel. Supposons qu'une tâche critique donnée τ_c peut être ordonnancée sur un multicœurs dans lequel un seul cœur est actif, mais que l'analyse

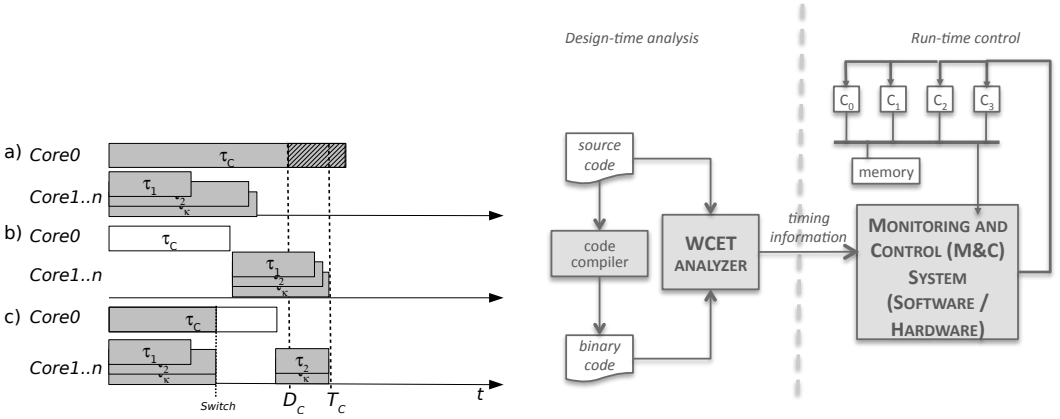


FIGURE 3.6 – **Gauche :** D'après l'analyse pessimiste classique avec 1 tâche critique τ_c sur un cœur, et n tâches non critiques τ_i sur les autres cœurs, *a*) le système n'est pas ordonnançable en parallélisme total, *b*) en cas d'isolation de la tâche critique, le système est sous utilisé, *c*) l'approche proposée augmente l'utilisation des ressources en garantissant les échéances temporelles. **Droite :** Processus de développement de l'approche criticité mixte garantie

du pire temps d'exécution, lorsque les autres cœurs sont utilisés, rend cette tâche statiquement non ordonnançable, comme représenté sur la figure 3.6 à gauche. Une approche statique possible serait alors d'ordonnancer d'abord la tâche critique et, lorsque celle ci est terminée, de lancer l'exécution des autres tâches.

Scénarios : isolation vs. pleine charge

Le but de l'approche proposée est de lancer de manière optimiste toutes les tâches et de surveiller le temps d'exécution de la tâche critique (qui dépend des motifs d'accès à la mémoire de l'ensemble des tâches s'exécutant), et d'arrêter les tâches non critiques si les interférences mémoire impactent trop le temps d'exécution de la tâche critique.

Dans cette approche simplifiée, deux scénarios d'exécution sont envisagés : *i*) la charge maximale, qui correspond à l'exécution en parallèle de toutes les tâches du système, et *ii*) l'isolation, lorsque seule la tâche critique s'exécute.

Si l'on lance initialement toutes les tâches en parallèle, le problème principal revient à déterminer, pendant l'exécution, le moment le plus tard où arrêter les tâches non critiques tout en garantissant les échéances de l'application critique envisagée —le switch sur la figure 3.6-*c*).

Analyse hors ligne pour la surveillance en ligne

Un programme pour lequel il est possible de calculer un pire temps d'exécution est, en général, un programme totalement dépliable dans lequel l'ensemble des branches possibles d'exécution forme un arbre fini. Il est donc aisé, mais coûteux, de calculer le pire temps d'exécution depuis n'importe quel point du programme. Cependant, le stockage de ces pire temps d'exécutions pour son utilisation à l'exécution est rédhibitoire et inenvisageable dans le cadre d'un système à ressources contraintes.

Pour fournir une évaluation des pire temps d'exécution à différents points d'un programme qui soit utilisable à l'exécution, il faut pouvoir stocker une structure de données compacte qui permette de déterminer rapidement à l'exécution le temps nécessaire à la complétion de la tâche considérée.

Boucles et fonctions

Considérons en particulier les nids de boucle, structure de programmation souvent employée dans les programmes temps-réel implémentant des filtres. Plutôt que de déplier totalement un nid de boucles et calculer en tout point le pire temps d'exécution restant, l'approche développée propose d'évaluer le pire temps restant dynamiquement : à partir du WCET en début de boucle et du WCET d'une itération de la boucle, calculés hors ligne, on peut calculer très rapidement en ligne le WCET à chaque itération de la boucle. Ce raisonnement peut s'appliquer récursivement à plusieurs boucles imbriquées en nid, à condition de mémoriser à l'exécution pour chaque niveau de boucle le compteur d'itération de chaque niveau.

Le cas des appels de fonctions est similaire aux boucles imbriquées (un appel de fonction est empilé de la même manière qu'une boucle à l'intérieur d'une autre est "empilée"), à ceci près que la structure d'appel d'un ensemble de fonctions est moins régulière et plus dynamique qu'un nid de boucle qui est statiquement défini.

Cette observation est à la base de l'approche développée car elle permet, lors d'une analyse hors ligne du programme, de produire une structure compacte permettant de calculer efficacement le pire temps d'exécution restant, ou RWCET [C28] (*Remaining Worst Case Execution Time*).

Pour ce faire, il a été nécessaire de se limiter à une sous-partie des programmes formalisés sous forme d'une variante des graphes de flot de contrôle (*CFG - Control Flow Graph*). Les programmes pour lesquels l'approche s'applique doivent ainsi se conformer à la grammaire schématisée en figure 3.7, où C est une conditionnelle, N un groupe d'instructions séquentielles, B un non terminal et F_i une fonction. Les deux caractéristiques remarquables sont *i*) la normalisation des boucles (la conditionnelle est avant l'itération, comme dans le cas d'un `while`) *ii*) la fermeture des `if/then/else` (le flot de contrôle doit converger vers un point de sortie).

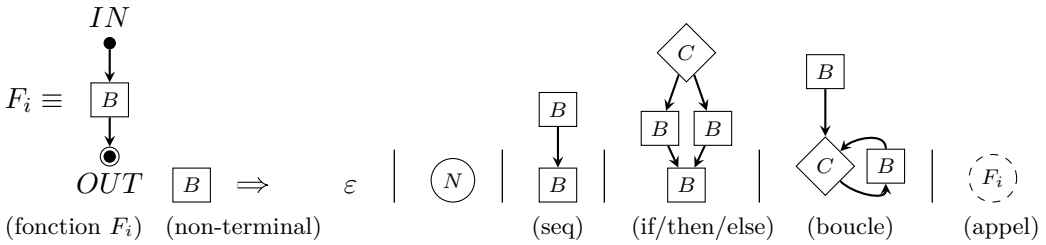


FIGURE 3.7 – Grammaire des fonctions pour l'évaluation du pire temps à l'exécution.

En se basant sur cette grammaire, il est possible de montrer que l'approche garantit que les contraintes temporelles de l'application sont respectées [C19], ce qui permet d'envisager cette approche dans une optique de certification. Au delà de la preuve formelle de correction, de nombreuses évaluations expérimentales ont été

menées pour tester la robustesse de l'approche et évaluer le compromis idéal entre le coût ajouté par la surveillance et le gain obtenu en parallélisme d'exécution.

Extension au cas réparti

Une extension naturelle des travaux décrits ci-dessus consiste à ne plus se limiter à un unique cœur exécutant des tâches critiques. Dans le cas où c cœurs doivent exécuter des tâches critiques, il est possible d'utiliser l'approche proposée ci dessus pour l'analyse hors ligne —en étendant l'analyse en isolation par une analyse avec c cœurs critiques uniquement— mais le contrôleur en-ligne doit maintenant prendre en compte la répartition des tâches critiques sur plusieurs cœurs.

Nous avons proposé une approche simple qui utilise un cœur dédié, le *master*, qui possède une vue globale de l'exécution. Chaque cœur critique exécute son contrôleur local comme décrit ci-dessus et vérifie sa condition de sûreté temporelle. Lorsqu'un contrôleur détecte qu'il est nécessaire d'arrêter les tâches non critiques pour diminuer les interférences, il envoie une requête au *master*. La Figure 3.8 illustre comment le *master*, en fonction des demandes d'exécution en isolation et de l'état d'exécution des tâches critiques, orchestre l'arrêt et le redémarrage des tâches non critiques [C20].

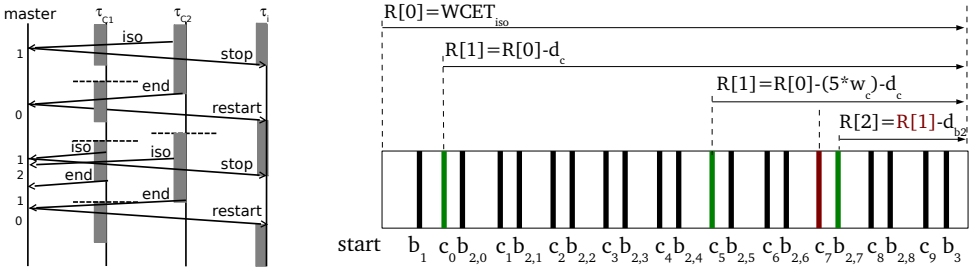


FIGURE 3.8 – **Gauche :** orchestration par le cœur *master* de l'exécution des tâches non critiques en fonction des requêtes d'isolation des tâches critiques. **Droite :** gain de l'approche adaptative/dynamique de la surveillance par rapport à une approche statique. En noir : points d'observation statiques, en vert : points d'observation effectivement exécutés par l'approche dynamique.

Une plus grande dynamicité de la vérification Le placement des points de surveillance temporelle de l'exécution est un paramètre essentiel de cette approche. Une surveillance trop fréquente implique un surcoût dû à l'exécution du contrôleur et diminue les performances. Placer les points de surveillance trop éloignés impliquera un passage en isolation anticipé dû à l'incertitude du temps d'exécution jusqu'au prochain point de vérification.

Il est possible de profiler une application donnée pour déterminer avant l'exécution un placement efficace des points de surveillance. Cependant, un profilage ne permettra que d'estimer un bon placement, et le placement optimal dépendra des conditions de l'exécution, par essence non prévisibles.

Une extension importante de ce travail a ainsi été l'étude du placement dynamique des points d'observation des propriétés temporelles [J2] afin de s'adapter aux conditions opérationnelles. À partir d'un placement statique pessimiste des points

d'observations, il est possible à chaque vérification du temps d'exécution de calculer le nombre maximal de points d'observation qu'il est possible d'ignorer sans mettre en danger l'échéance temporelle de la tâche en cours. Cette approche permet ainsi d'augmenter l'utilisation efficace d'une architecture multicœurs exécutant des tâches de niveaux de criticité différents.

3.4 Bilan et notes

Détecter une violation de la spécification ou, mieux, un risque de déviation d'un programme avant qu'il ne soit trop tard est essentiel pour permettre la reconfiguration et continuer à garantir la sûreté de fonctionnement d'un système.

Dans le cadre de la thèse de Thomas Robert [Th10] puis d'Olivier Baldellon [Th7], une grande partie des travaux réalisés s'est concentrée sur l'expression des propriétés à vérifier. Nous avons considéré deux formalismes incluant le temps et les évènements pour exprimer puis vérifier une spécification : les *automates temporisés* pour les systèmes centralisés, et les *réseaux de Petri A-temporisés* pour les systèmes répartis.

Dans le cas centralisé, nous avons défini la notion de plus petit symptôme d'erreur, qui permet d'implémenter la détection au plus tôt, c'est-à-dire la détection de la déviation de la spécification dès que cela est possible. Nous avons montré que cette notion est implémentable efficacement en utilisant le graphe quotient issu de l'automate temporisé décrivant la propriété à vérifier, qui donne une représentation finie et compacte de l'ensemble des exécutions possibles. Nous avons montré que cette approche permet d'implémenter la résilience proactive, dans laquelle une reconfiguration est effectuée avant même que le système ne sorte de sa spécification. La description de l'approche pour la détection au plus tôt est publiée dans PRDC'08 [C43], et les mécanismes pour la résilience proactive ont été publiés dans RTNS'10 [C40]

Les réseaux de Petri, de par leur nature non centralisée, permettent quant à eux d'exprimer et d'implémenter la vérification d'un spécification répartie. Nous avons développé la notion de *jeton négatif* dans un réseau de Petri, qui permet une vérification en-ligne malgré les délais et incertitudes introduits par la répartition. Pour simplifier et fiabiliser l'écriture de spécifications sous forme de réseau de Petri, nous avons développé une bibliothèque de motifs qui peuvent être assemblés pour former des propriétés complexes. Ces travaux ont été concrétisés dans un prototype, Minotor, qui compile une spécification et déploie l'outil de supervision dans un environnement réparti. Minotor est décrit dans PRDC'13 [C25], et l'analyse des jetons négatifs pour la vérification en ligne est publiée dans SRDS'12 [C31].

Dans un système temps-réel multicœur à mémoire partagée, la variabilité des temps d'accès à la mémoire nécessite de repenser la notion même de pire temps d'exécution. Dans le cadre d'une collaboration [Pr14] avec C. Pagetti (ONERA) et C. Rochange (IRIT), les travaux post-doctoraux d'A. Kritikakou [P2] se sont concentrés sur l'utilisation de la vérification à l'exécution des propriétés temporelles pour garantir l'exécution de tâches critiques dans les temps, malgré la présence d'autres applications non critiques s'exécutant en concurrence.

Ces travaux s'appuient sur la séparation entre une analyse hors-ligne du code et

une vérification en-ligne. L'application est analysée en calculant le WCET de la tâche en plusieurs points du programme en considérant l'application seule et l'application en concurrence maximale. En ligne, un système de supervision surveille l'exécution, vérifie les contraintes temporelles et décide si besoin d'interrompre les tâches non critiques pour réduire les interférences mémoire. Ces travaux proposent pour la première fois de considérer une variante dynamique du pire temps d'exécution, couplé à un système d'ordonnancement dynamique qui garantit les contraintes temporelles de l'application. Grâce à l'approche formelle sous forme d'une grammaire des programmes autorisés, les garanties temporelles du système sont prouvées et ce type d'approche pourrait donc être utilisé dans un cadre de certification. Parmi les publications associées [J2, C7, C19, C20, C28], les résultats les plus importants de ces travaux ont été publiés dans ECRTS'13 [C19] et ACM Trans. on Design Automation and Embedded Systems [J2]. Par ailleurs, l'approche est utilisée par Thales RT dans leurs développements de futures architectures pour les systèmes avioniques.

Les travaux issus de la thèse de Thomas Robert sur l'automate des régions et la résilience proactive ont inspiré le concept de *safety bag* pour la surveillance de propriétés liées à la sûreté [Pr10], dont la description dépasse le cadre de ce résumé. La supervision pour la sûreté de systèmes autonomes a été développé dans la thèse de A. Mekhi-Mokhtar [C33, C34], puis dans celle de M. Machin [J7] et lors des travaux post-doctoraux de Fanny Dufossé [C11, P1], en collaboration avec J. Guiochet (LAAS).

La surveillance à l'exécution des temps d'exécution dans les systèmes multicœurs a inspiré une collaboration en cours avec Renault [Pr2] pour la thèse de D. Loche [Th2] sur les architectures permettant l'exécution sûre de tâches de niveaux de criticité différents sur une architecture utilisant des processeurs multicœurs.

Chapitre 4

Calculer : Algorithmes pour systèmes répartis dynamiques

> Le hasard, c'est Dieu qui se promène incognito.
>
> Albert Einstein

Les systèmes informatiques répartis sont formés à partir d'entités de calcul qui communiquent et se coordonnent à travers une interface qui les relie. Comme soulevé dès l'introduction de ce document, les sources d'incertitude dans un tel système sont multiples, et il n'existe pas de modèle de calcul simple et unifié de système réparti, tant les sources d'incertitude (défaillance, asynchronie, communication, etc) sont nombreuses et variées.

Dans la jungle de ces systèmes, deux sources d'incertitude et donc de dynamicité me tiennent particulièrement à cœur :

— les systèmes répartis à communication par *mémoire partagée*. Ces systèmes, faiblement répartis, englobent les systèmes multicœurs actuels, organisés autour d'une mémoire globale,

— les systèmes répartis *géo-localisés*. Depuis l'avènement de l'informatique mobile (smartphones, véhicules communicants), il apparaît qu'aucun modèle de système réparti ne prend correctement en compte la géographie et la mobilité des nœuds de calcul.

Ce chapitre se focalise d'abord sur les algorithmes sans-attente, ou *wait-free*, qui forment une sous-classe des algorithmes pour mémoire partagée qui doit fonctionner quelque soit l'incertitude temporelle, puis décrit mes travaux sur les algorithmes géo-localisés, dans lesquels la source majeure d'incertitude est le déplacement des nœuds de calcul.

4.1 Algorithmes wait-free

Les algorithmes sans attente (wait-free [18]) représentent la sous-classe d'algorithmes répartis s'exécutant sur une mémoire partagée commune —on peut aussi

parler d'algorithmes concurrents— pour lesquels chaque processus de calcul participant termine son exécution en un nombre borné d'étapes de calcul, quel que soit le comportement des autres processus.

Contrairement à une programmation classique par verrou assurant la cohérence des données (qui est par définition sujette à des blocages si le processus détenant le verrou est défaillant), la classe des algorithmes wait-free est, par nature, adaptative et robuste car tout processus progresse dans son exécution et n'est pas ralenti par les autres processus : la seule hypothèse sur laquelle repose un algorithme wait-free est l'atomicité [19] des lectures et écritures en mémoire.

Est-il possible d'implémenter des outils de synchronisation non triviaux au dessus d'un modèle sans verrous qui capture d'une certaine manière les hypothèses minimales qu'on peut faire sur un système multi-thread s'exécutant sur un multicœurs ? Dans le cadre de collaborations [Pr9] avec Eli Gafni (UCLA), Sergio Rajsbaum (UNAM) et Pierre Fraigniaud (IRIF), nous avons montré que malgré la relative faiblesse des hypothèses de travail, il est possible de construire des algorithmes qui s'adaptent à l'incertitude temporelle et offrent des services utiles au développement d'application sûres de fonctionnement.

4.1.1 S'adapter à l'asynchronie

Simulations en mémoire partagée La BG-simulation, du nom de ses auteurs Borowsky et Gafni [21, 20], est un algorithme permettant de simuler l'exécution de n processus de calcul, parmi lesquels au plus t peuvent défaillir par arrêt, par un ensemble de $t+1$ processus tous susceptibles de défaillir —ce dernier modèle est exactement un modèle de calcul *wait-free*.

La BG-simulation a permis de prouver des résultats d'impossibilité et de calculabilité en montrant de manière effective qu'un système distribué à mémoire partagé tolérant t fautes est équivalent à un système *wait-free* de $t+1$ processus.

En réécrivant pour simplifier les concepts qui sous-tendent la BG-simulation [Ch1], il est apparu que les techniques employées sont très similaires à une version wait-free de la *state machine replication*, ou réplication active [23, 24, 22], une technique centrale de la tolérance aux fautes.

Les mécanismes de réplication active utilisent le consensus comme brique de base pour garantir que l'ensemble des répliques sont cohérentes. Or, il est impossible de résoudre le problème du consensus si le système est asynchrone et au moins un processus peut être défaillant [25].

Safe-agreement : un consensus affaibli wait-free Cependant il existe des problèmes moins contraignants que la réplication active, comme le partage virtuel de mémoire [26], qui ne nécessitent pas d'utiliser le consensus. Une question essentielle dans les systèmes wait-free est d'exhiber un mécanisme d'accord qui ait une solution, et qui puisse être utile au programmeur. Le *safe-agreement* est une telle variante affaiblie du consensus.

Définition 4 *Le* safe-agreement *est une tâche dans laquelle des processus proposent chacun une valeur, et doivent décider (wait-free) une valeur de sortie respectant les trois propriétés :*

Terminaison : *Chaque processus non défaillant doit renvoyer une valeur proposée, ou* $\perp$ *;*

Validité : *Si tous les processus sont corrects, alors au moins un processus décide d'une valeur non-*$\perp$ *;*

Accord : *Tous les processus qui décident une valeur différente de* $\perp$ *décident de la même valeur.*

En comparaison du problème du consensus, le safe-agreement autorise que certains processus retournent une valeur indéfinie $\perp$. Cependant il n'est pas possible que tous renvoient cette valeur $\perp$ dans le cas où aucun processus n'est défaillant, d'après la propriété de validité. Ce problème a une solution wait-free, décrite dans l'Algorithme 1 [C14].

Algorithm 1 Wait-free SAFE-AGREEMENT : code du processus p proposant v

1: write $(Id(p), v)$ ▷ premier write
2: snapshot memory, to get $view = \{(i_1, v_{i_1}), \ldots, (i_k, v_{i_k})\}$ ▷ premier snapshot
3: write $(Id(p), view)$ ▷ second write
4: snapshot memory, to get $setofview = \{(j_1, view_{j_1}), \ldots, (j_\ell, view_{j_\ell})\}$ ▷ second snapshot
5: $minview \leftarrow \bigcap_{r=1}^{\ell} view_{j_r}$ ▷ minimum par inclusion
6: **if** for every i such that $(i, v_i) \in minview$ we have $view_i \in setofview$ **then**
7: decide minimum value w in $minview$ ▷ $w = \min\{v_i : (i, v_i \in minview)\}$
8: **else**
9: decide $\perp$ ▷ Il existe un processus bloqué entre la ligne 1 et la ligne 3
10: **end if**

Cet algorithme, en apparence simpliste, fournit aux processus d'un système wait-free ce qui semble être l'accord maximal atteignable malgré l'asynchronie et les défaillances possibles. En particulier, on peut remarquer que si un processus p renvoie $\perp$, c'est qu'il existe un autre processus qui n'est pas synchrone avec p. Ce lien entre l'asynchronie observée et la qualité de l'accord obtenu est à la base des travaux décrits ci-dessous.

Dégradation de la réplication active La réplication active permet de simuler un thread sur un ensemble de répliques réparties. Dans un système dans lequel le consensus n'a pas de solution, comme ici les systèmes wait-free, la question de savoir s'il est possible de construire un système réparti de réplication qui garantit la cohérence tout en offrant des propriétés de vivacité était ouverte.

Nous avons montré que, dans un système wait-free, l'équivalent de la réplication active correspond à l'exécution simultanée de k threads, où k est une mesure de l'asynchronie au sein du système [C14]. Ainsi si le système est synchrone, la simulation proposée est un algorithme de réplication active.

RC-simulation et BG-simulation La première étape de ce travail a été de formaliser sous forme algorithmique les idées de fond de la BG-simulation, dont la description dans l'article original est assez informelle [20].

De manière assez similaire à ce que fait un algorithme de réplication active, il est possible de formaliser la simulation en considérant les threads à exécuter comme une

séquence d'instructions. La manière la plus naturelle pour ce faire est de supposer que les threads sont écrits sous forme normale, comme décrit dans l'algorithme 2. Dans cette forme, un thread effectue une séquence de lecture/écriture en mémoire jusqu'à ce qu'un prédicat de terminaison φ soit vrai, et applique une fonction de décision f sur l'ensemble de l'information collectée. Les lectures sont faites à l'aide de snapshots, qui prennent un instantané global de la mémoire, et les écritures stockent en mémoire l'information accumulée par le processus depuis le début de son exécution —la forme normale correspond à une implémentation à information maximale.

Algorithm 2 Forme normale pour un thread $\tau = (Id, A = (\varphi, f), v)$.

```
1: view ← (Id, v)
2: write(view)
3: while φ(view) do                    ▷ φ : prédicat de terminaison de A
4:     view ← snapshot()
5:     write(view)
6: end while
7: decide f(view)                      ▷ f : fonction de sortie de A
```

Ainsi, les instructions des threads à simuler forment une séquence de snapshot/write [27]. On peut représenter l'état de la simulation BG comme une matrice d'instruction M : chaque thread correspond à une colonne, et $M_{i,j}$ représente le $i^{\text{ème}}$ snapshot du thread j.

Pour garantir la cohérence des vues entre les différents simulateurs, chaque cellule $M_{i,j}$ est associée à une instance du safe-agreement, comme décrit sur la Figure 4.1 : les simulateurs exécutent pour chaque instruction l'algorithme 1 qui peut converger sur une valeur unique, ou renvoyer $\perp$ si un des simulateurs est ralenti ou défaillant.

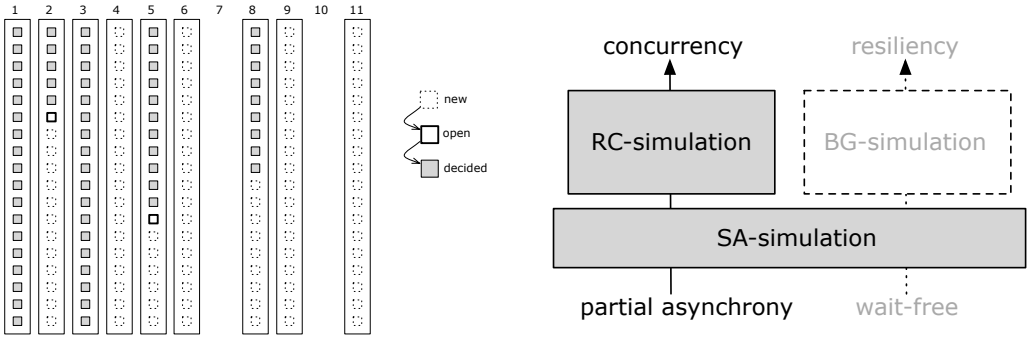


FIGURE 4.1 – **Gauche :** la SA-simulation se base sur la matrice d'instructions M des threads à exécuter —chaque thread est représenté par une colonne. **Droite :** la RC-simulation et la BG-simulation sont deux instances de la SA-simulation qui visent deux buts différents.

Cette représentation de la simulation est en fait une simulation basée sur le safe-agreement, ou SA-simulation, de la même manière que la réplication active est basée sur le consensus.

Pour aller plus loin dans cette analyse, et en remarquant qu'un processus défaillant par arrêt n'est pas distinguable d'un processus très lent, il est possible d'instancier la SA-simulation sous une forme différente de la BG-simulation originale (voir Figure 4.1) : si les threads simulés sont choisis en utilisant un vecteur de safe-agreement, alors il est possible de garantir que le nombre de threads simulés sera fonction de la mesure de l'asynchronie dans le système.

La BG-simulation sélectionne les threads de manière arbitraire, mais les simule selon un ordonnancement de type tourniquet pour garantir la vivacité de la simulation. La RC-simulation (RC : *Reduce Concurrency*) simule les threads dans un ordre arbitraire, mais sélectionne les threads simulés pour garantir un nombre minimum de threads ouverts [C14].

Renommage dépendant de l'asynchronie Pour illustrer l'intérêt de la RC-simulation, prenons comme threads à simuler des instances de l'algorithme classique de *renaming* [28] : dans ce problème, un ensemble de processus possédant des identifiants dans un domaine de grande taille se coordonnent pour obtenir des identifiants uniques dans un domaine plus petit —dans le cas wait-free, n processus peuvent se renommer dans l'espace $[1, 2n-1]$, mais pas dans $[1, 2n-2]$.

En simulant l'algorithme classique du renommage sans attente (*wait-free renaming*) à l'aide de la RC-simulation, alors la taille de l'espace des noms alloués s'adaptera à la synchronie dans le système. Plus précisément, l'espace des noms en sortie sera $[1, n+k]$, avec k représentant l'asynchronie dans le système ($k = 0$ est un système synchrone, plus de détails sur la définition exacte de k dans [C14]).

En conclusion, la simulation proposée permet de transformer automatiquement tout algorithme wait-free en une version dont la qualité de la solution sera fonction de l'asynchronie dans le système.

4.1.2 Allocation de ressources contraintes

La communauté des chercheurs en algorithmes répartis est traditionnellement divisée en deux sous-communautés : d'un côté, les recherches sur les algorithmes ou tâches en supposant que le support de communication est global (le renaming fait partie de cette classe de recherches), et de l'autre les algorithmes sur les graphes, pour lesquels on recherche la meilleure solution possible en interaction locale (la recherche d'un stable maximal –*maximal independent set*– en est un exemple).

Dans le renaming, les processus doivent obtenir un nouvel identifiant unique dans un ensemble le plus petit possible. Il est possible de considérer ce problème comme l'instance d'un problème de coordination dans lequel les processus doivent choisir de manière exclusive des ressources pour les utiliser. Considérons maintenant que certains de ces choix sont mutuellement incompatibles. Ce type de contrainte peut être représenté par un graphe où les ressources sont les nœuds du graphe, et les arcs représentent des choix incompatibles. À titre d'exemple, la Figure 4.2 représente des transactions à exécuter et les arcs dénotent les dépendances entre ces transactions. Un ensemble de serveurs exécutant ces transactions a tout intérêt à choisir un ensemble indépendant de nœuds afin d'exécuter des transactions indépendantes.

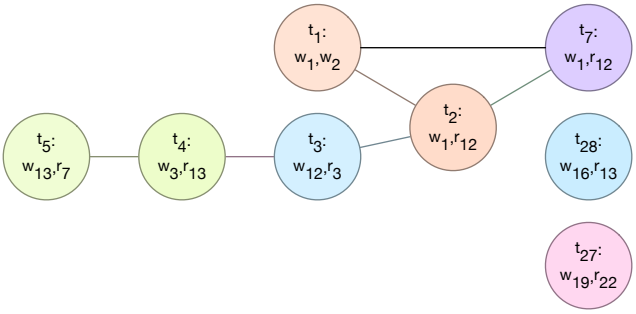


FIGURE 4.2 – Exemple de graphe représentant des transactions et leurs dépendances

Ce problème de coordination asynchrone avec contraintes [C3, C4] est l'application de la vision « tâches » à un problème classique sur les graphes, le Maximal Independent Set.

Définition 5 (coordination asynchrone avec contraintes) *Soit $G = (V, E)$ un graphe à n nœuds modélisant les ressources et leurs contraintes. Dans le problème de la coordination asynchrone avec contraintes, chaque processus a en entrée un nœud du graphe et doit retourner un nœud de G tel que l'ensemble des sorties des processus forme un stable (*Independent Set*) de G.*

Nous avons étudié ce problème en cherchant à déterminer, pour un graphe G donné, le nombre maximal de processus qui peuvent résoudre le problème de la coordination asynchrone avec contraintes.

Borne inférieure sur les lignes Considérons la ligne P_n de n nœuds. J'ai obtenu la borne inférieure suivante, prouvée dans [C3], par réduction à un autre problème classique, le renaming avec préférence initiale. Il est aussi possible de démontrer ce résultat de manière directe en utilisant les outils de la topologie algébrique.

Théorème 1 *Soit k un entier positif. Le plus petit n tel que le problème de la coordination asynchrone avec contraintes sur P_n a une solution pour k processus satisfait $n = 4k - 3$*

On peut remarquer que la présence des contraintes a une conséquence « dramatique » sur la borne inférieure, qui est quasiment le double de la borne inférieure $2k - 1$ de la version sans contrainte (le renaming).

Cette borne est atteinte par l'algorithme simple suivant : *i*) choisir un stable maximal I de P_n, *ii*) numéroter les nœuds de I de 1 à $2n - 1$, *iii*) exécuter un algorithme optimal de renaming sur I.

Le cas des anneaux Le cas des anneaux est plus troublant. Pour un anneau C_n de taille n, la borne inférieure issue de la preuve pour un chemin s'applique et montre que le nombre k de processus pouvant résoudre ce problème sur le cycle C_n vérifie $k - 3 \leq n$.

Cependant, l'algorithme basé sur un choix statique (avant l'exécution) d'un stable maximal I sur lequel on exécute un algorithme de renaming n'atteint pas la borne inférieure et fonctionne jusqu'à $n = 4k - 2$.

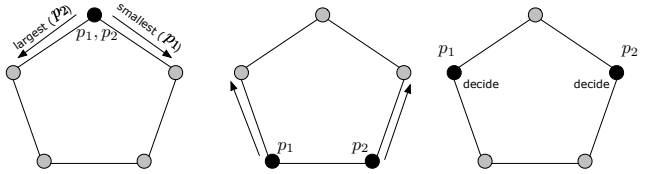


FIGURE 4.3 – Algorithme optimal de coordination pour deux processus sur un pentagone

De manière intéressante, les algorithmes optimaux pour 2 et 3 processus sur l'anneau sont radicalement différents des algorithmes classiques de renaming. L'application d'un algorithme de renaming permet d'obtenir une solution sous-optimale au problème, alors que l'algorithme optimal est très dynamique et s'appuie sur les propriétés du graphe. L'algorithme optimal pour 2 processus sur un anneau de taille 5 utilise les identifiants des processus et la topologie du réseau pour résoudre les conflits, comme représenté sur la Figure 4.3.

De nombreux problèmes restent ouverts sur ce travail qui montre que toute solution à la coordination asynchrone avec contraintes doit s'adapter aux propriétés de l'exécution (identifiants, asynchronie) et à la structure du graphe sous-jacent pour être efficace.

4.2 Algorithmes géo-localisés

Dans cette partie, je décris des algorithmes et mécanismes répartis dans lesquels les participants connaissent de manière explicite leur localisation, d'où le terme « algorithme géo-localisé ».

4.2.1 Geo-registre : le partage géo-localisé

Dans le cadre du réseau d'excellence ReSIST [Pr16], j'ai dirigé un groupe de travail sur la définition et l'implémentation d'une solution de partage de données (un registre [29], dans la terminologie classique des systèmes répartis) pair-à-pair et localisée dans l'espace.

Un geo-registre [C44] est une abstraction permettant aux utilisateurs de partager des données (lecture/écriture) à proximité d'une zone localisée dans l'espace. Deux sources majeures de difficulté rendent ce problème difficile : 1) l'incertitude due aux déplacements par définition imprévisibles des utilisateurs, et 2) la prise en compte de la spatialisation, non prise en compte dans les modèles habituels de systèmes répartis.

Les travaux développés proposent deux concepts permettant de résoudre les difficultés ci-dessus : 1) la définition d'une primitive de communication à garanties spatio-temporelles, le $(\delta, A)-$geo-reliable broadcast, et 2) le raffinement de l'espace

où les opérations de lecture et écriture sont possible, séparant la région en une zone de lecture écriture (*core region*) et une zone de relais pour la fiabilité.

Plus précisément, un geo-registre est associé à une zone géographique donnée A, comme décrit sur la Figure 4.4, dans laquelle on implémente une diffusion fiable géo-localisée.

Définition 6 ($(\delta, A)-$geo-reliable broadcast) *est une primitive de communication telle que*

- *tout processus $p \in A$ peut exécuter* `broadcast`(m)
- *si un processus correct exécute* `broadcast`(m) *au temps t et reste dans A de t à $t + \delta$, alors tous les processus corrects présents dans A entre t et $t + \delta$ délivrent m avant $t + \delta$.*

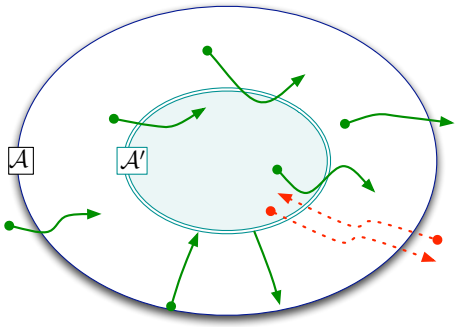


FIGURE 4.4 – Définition de la *core region* A' liée au $(\delta, A)-$geo-reliable broadcast : pour garantir la cohérence des vues, un processus dans A' n'a pas le temps de quitter A' pendant un broadcast (flèche rouge impossible)

Définition 7 (Core region) *Une* core region *A' associée à A est un sous-ensemble de A tel que tout message m envoyé au temps t par un processus p dans A' en utilisant un $(\delta, A)-$geo-reliable broadcast sera délivré (avant $t + \delta$) par tout processus q qui était dans A' au temps t.*

Geographically controlled thread :
when p enters A :
  $R_p \leftarrow void$;
  wait for
    □ ($W(x)$ is received) : $R_p \leftarrow x$; exit ;
    □ (2δ time delay elapsed)
  RB_send(REQ)
  wait for
    □ ($REP(v)$ is received) : $R_p \leftarrow v$;
    □ ($W(x)$ is received) : $R_p \leftarrow x$;
    □ (2δ time delay elapsed) : $R_p \leftarrow \perp$;
when p leaves A :
  free(R_p) ;

Communication controlled thread :
upon reception of (REQ) : **if** ($R_p \neq void$) **then RB_send**($REP(R_p)$)
upon reception of ($W(x)$) : $R_p \leftarrow x$

Read and Write operations :

When p is in A :
`read`() : **wait until** ($R_p \neq void$) **then return**(R_p) ;

When p is in A' :
`write`(x) : **RB_send**($W(x)$) ;

FIGURE 4.5 – Implémentation d'un geo-registre

Ces deux définitions ont pour intérêt majeur de masquer de nombreux paramètres physiques du système (vitesse des entités, coordonnées géographiques) et permettent de raisonner en faisant abstraction de certains paramètres physiques, d'une manière similaire aux détecteurs de défaillances qui masquent les hypothèses temporelles dans une abstraction.

À titre d'illustration, l'implémentation d'un geo-registre qui se base sur ces mécanismes est décrite dans la Figure 4.5. L'algorithme semble simple, mais c'est justement parce que les hypothèses et les mécanismes sous-jacents capturent les paramètres essentiels pour simplifier le raisonnement.

Les idées développées lors de la conception de cet algorithme, en particulier la gestion des délais, la gestion du groupe dynamique de communication, et des outils basés sur les deux définitions données ici ont été repris dans des travaux d'autres équipes de recherche [30, 31] qui cherchent à implémenter des services de stockage dans des environnements très dynamiques.

4.2.2 Boîte noire distribuée : principes et algorithmes

Comment fournir des services fiables répartis à un ensemble de véhicules communicant au travers d'une interface réseau pair-à-pair tel que le futur réseau sans fil pour véhicules IEEE 802.11*p* ? C'était l'enjeu du projet européen HIDENETS [Pr15].

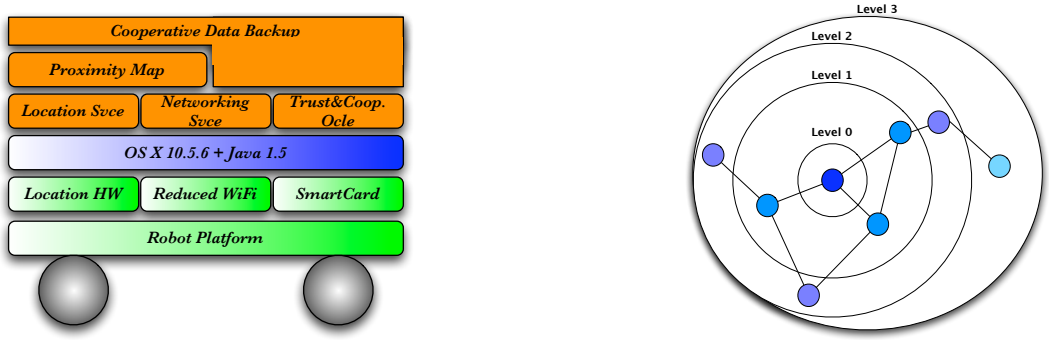


FIGURE 4.6 – **Gauche :** Architecture HIDENETS. **Droite :** Structure de données fournie par le service *proximity map*

J'ai participé à la spécification et au développement de trois modules originaux, représentés sur la Figure 4.6 à gauche, qui encapsulent des aspects propres aux réseaux sans fil portés par les humains :

— *Proximity map* représente la connaissance qu'une entité a sur son voisinage. Les informations sur les voisins directs sont les plus à jour, et la pertinence des informations sur les voisins diminue avec le nombre de sauts réseau nécessaires. La structure de données résultante a une forme d'oignon, comme représenté sur la Figure 4.6 à droite.

— *Trust and cooperation oracle* fournit un service réparti d'évaluation du niveau de confiance entre les participants du système. Il a été implémenté de manière simple en vérifiant que le logiciel s'exécutant est intègre à l'aide d'un système de cartes à puces.

— *Cooperative data backup* est un service de sauvegarde de données entre les différents utilisateurs du réseau, implémenté en utilisant une approche de type fragmentation-redondance-dissémination.

Au dessus de cette architecture, nous avons développé une application de *boîte noire répartie*, qui utilise les véhicules à proximité pour disséminer et sécuriser le flot de paramètres sur les trajets effectués (position, vitesse, etc). En cas d'accident, il est ainsi possible de rejouer le scénario de l'accident en utilisant les données sauvegardées sur les autres véhicules [Ch2, Ch4].

4.2.3 Preuves réparties de localisation

Suite aux travaux développés dans le cadre du projet européen PRIME [Ch3, Pr18], nous avons monté au LAAS le projet ANR AMORES [Pr6] autour de la fourniture de services collaboratifs géo-localisés préservant la vie privée. Dans le cadre de ce projet, nous avons travaillé sur les abstractions et algorithmes nécessaires à l'implémentation d'un service de co-voiturage dynamique.

Dans ce projet, le but était de mettre en relation des utilisateurs qui partagent un but commun (ou, tout au moins, leurs buts ont une intersection non vide), tout en garantissant la protection de leur vie privée et de leur sécurité. Au delà de cette application, l'idée est de proposer un ensemble de *building blocks* spécialisés sur les services géo-localisés préservant la sûreté et la sécurité.

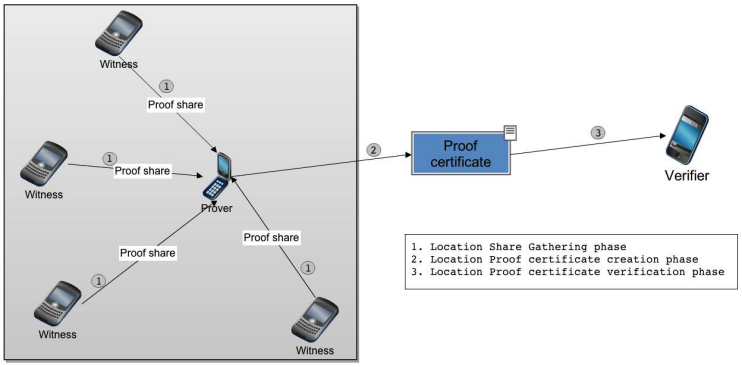


FIGURE 4.7 – Architecture du système de preuve de localisation PROPS

Dans ce cadre, j'ai co-dirigé (avec Marc-Olivier Killijian et Sébastien Gambs) la thèse de Moussa Traoré [Th6] sur les preuves de localisation respectant la vie privée [C32].

Nous avons développé un système collaboratif de preuves de localisation PROPS (*PRivacy-preserving lOcation Proof System*) [C18] qui permet à un utilisateur de collecter une preuve des localisations qu'il a visitées en interagissant avec d'autres utilisateurs, le tout en offrant des propriétés de respect de la vie privée et sécurité fortes pour ces mêmes utilisateurs. Le système est totalement décentralisé (voir la Figure 4.7) et utilise des schémas cryptographiques asymétriques et répartis pour permettre à un *Prover* de prouver à un *Verifier* qu'il se se trouvait à un instant donné à un endroit donné. Vu la nature très physique (localisation) de la propriété

à prouver, ce premier travail s'est appuyé sur une brique mêlant algorithmique et physique de délimitation de distance.

Cette brique de délimitation fiable de la distance a ensuite été implémentée dans le protocole VSSDB (*Verifiable Secret-Sharing based Distance-Bounding protocol*) [C17], un nouveau protocole délimiteur de distance permettant à un utilisateur de convaincre un autre utilisateur qu'il est situé dans sa proximité. Ces travaux s'appuient sur plusieurs mécanismes physiques (délimitation des temps de transfert entre terminaux) et cryptographiques, et proposent une approche prouvée formellement et un prototype sur terminal Android garantissant la sécurité des utilisateurs dans un environnement mobile et ouvert.

4.3 Bilan et notes

Maîtriser l'incertitude est au cœur de la plupart des travaux sur les systèmes répartis. Les travaux présentés dans ce chapitre traitent de l'incertitude temporelle dans les systèmes répartis à mémoire partagée, puis de l'incertitude spatio-temporelle des systèmes répartis mobiles.

Le théorème le plus emblématique de l'algorithmique répartie, l'impossibilité du consensus en environnement asynchrone [25], s'appuie sur la propriété qu'il est impossible de différencier un processus lent et un processus défaillant. L'asynchronie peut donc être vue comme une forme affaiblie d'une défaillance par arrêt.

Les travaux décrits dans la première partie de ce chapitre traitent des algorithmes wait-free, les algorithmes sur mémoire partagée dont le temps d'exécution de chaque processus est borné, quelque soit le comportement des autres processus. Ces algorithmes ont un intérêt pratique pour les architectures multicœurs et capturent par nature le lien identifié ci-dessus entre défaillance par arrêt et asynchronie.

La première section décrit la RC-simulation, qui peut être vue comme une variante d'un autre algorithme, la BG-simulation. La RC-simulation transforme l'asynchronie en concurrence (RC=*reduce concurrency*), et correspond à la transposition dans le monde wait-free de la réplication active, ou *state machine replication* : placée dans un système synchrone, la RC-simulation implémente un mécanisme de réplication active, et, plus le système est asynchrone, plus les performances du système en termes de réplication se dégradent. La formalisation présentée montre que ces deux algorithmes sont deux instances d'une classe d'algorithmes de simulation basée sur un version affaiblie du consensus , le safe-agreement, dont une implémentation wait-free a été proposée. Ces travaux proposent aussi pour la première fois une définition claire de la notion d'asynchronie dans un système réparti. Ce lien identifié entre l'asynchronie du système et la qualité de service d'un protocole de réplication a aussi des conséquences générales sur les algorithmes wait-free, car il montre qu'il existe des algorithmes qui s'adaptent à l'asynchronie, dans le sens où la qualité de la solution obtenue est fonction de l'asynchronie de l'exécution.

Dans le cadre de ces collaborations avec des chercheurs d'autres laboratoires — S. Rajsbaum & A. Castañeda (UNAM, Mexique), E. Gafni (UCLA, US) et P. Fraigniaud (IRIF) — la deuxième contribution sur le sujet des algorithmes wait-free concerne l'étude d'un problème classique de l'algorithmique sur les graphes, le *Maximal Independent Set*, dans un modèle wait-free. Nous avons montré que ce problème,

qui peut être vu aussi comme une extension du renommage dans un graphe, est un problème fondamentalement différent d'autres problèmes wait-free : les algorithmes optimaux, que nous n'avons pu déterminer que dans quelques cas, présentent une forme particulière qui s'adapte dynamiquement aux propriétés temporelles de l'exécution et à la structure du graphe. L'étude fine des propriétés de ce problème en utilisant la topologie algébrique, qui sort du cadre de ce mémoire, a permis de dériver des familles de graphes « difficiles » et « faciles » pour ce problème.

Les travaux sur la RC-simulation sont publiés dans DISC'14 [C14] ; la reformulation de la BG-simulation simplifiée en utilisant les structures décrites ici est publiée dans l'*Encyclopedia of Algorithms* [Ch1]. Entre temps, l'article original sur la BG-simulation a obtenu le prix Dijkstra 2017, montrant l'impact de ces travaux sur la communauté. Les travaux sur la coordination asynchrone dans les graphes ont été publiés dans SIROCCO'16 [C3], et une version étendue est en cours de préparation. Le problème dual de la coordination asynchrone, utilisant le dual d'un graphe, est la version discrète d'un problème classique de robotique collaborative, le *gathering*, traité dans [J1, C5]

La deuxième grande classe de problèmes étudiés dans ce chapitre est formée par les algorithmes pour systèmes mobiles. Ajouter la composante géographique, physique, à un système réparti augmente de façon considérable l'incertitude dans le système et concevoir des algorithmes pour de tels systèmes est une tâche difficile.

Dans ce cadre, l'essentiel de mes travaux s'est porté sur la détermination des « bonnes » abstractions permettant de raisonner sur ces systèmes —une « bonne » abstraction permet de masquer certains paramètres physiques pour simplifier la programmation tout en capturant des propriétés réalistes sur l'exécution. En suivant cette ligne de recherche, les premiers travaux que j'ai conduits identifient des briques fondamentales, en particulier la « carte de proximité » qui agrège l'information dans une structure récursive capturant le degré de connaissance maximale que peut avoir un participant dans un système mobile.

Les travaux sur les geo-registres, ou stockage géo-localisé, montrent qu'il est possible de proposer un service de stockage localisé dans l'espace au dessus d'un service de diffusion fiable possédant des garanties spatio-temporelles. Ces travaux proposent de séparer l'espace en deux zones concentriques de niveau de cohérence différent, une zone de lecture et une d'écriture, définissant le concept de *core region* pour garantir la cohérence. Les briques sur lesquelles s'appuient l'implémentation du service de stockage permettent d'écrire l'algorithme sans référence à des propriétés physiques comme la vitesse des entités et sont donc de « bonnes » abstractions.

Enfin, une composante essentielle d'un système géo-localisé coopératif est la confiance que chacun peut avoir envers les autres entités, et tout particulièrement pour garantir que la localisation annoncée par un participant est authentique. Dans la thèse de M. Traore [Th6] dans le projet AMORES [Pr6], nous avons développé VSSDB, un protocole délimiteur de distance collaboratif infalsifiable. VSSDB s'appuie sur les propriétés physiques de la transmission radio et sur un mécanisme collaboratif utilisant le chiffrement pour prouver que la distance entre deux entités du système est bornée. En s'appuyant sur ce protocole, nous avons pu développer PROPS, un système de preuve de localisation respectant la vie privée ; PROPS n'utilise aucun service centralisé et ouvre la possibilité d'implémenter des services basés

sur la localisation fiable de manière totalement décentralisée. Ces travaux ont été réalisés en collaboration avec S. Gambs et C. Lauradoux pour la partie cryptographique.

Les premiers travaux sur les architectures pour les systèmes mobiles ont été décrits dans un chapitre d'*Architecting Dependable Systems* [Ch4]. Les travaux sur les geo-registres sont publiés dans OPODIS'08 [C44]. VSSDB est publié dans BalkanCryptSec'14 [C17] et PROPS a été publié à SRDS'14 [C18].

Chapitre 5

Comprendre et mesurer la dynamique humaine dans les systèmes

> La terre est ma patrie. Et l'humanité, ma famille.
>
> Khalil Gibran

Une des sources majeures d'incertitude des systèmes répartis qui se sont développés dans les dix dernières années est la mobilité des terminaux informatiques. Contrairement aux systèmes répartis développés jusqu'à la fin des années 2000, l'avènement des technologies sans fil combiné à la miniaturisation des systèmes et aux progrès des dispositifs de stockage d'énergie a permis le développement de grands systèmes répartis. Deux caractéristiques majeures de ces grands systèmes nécessitent une attention particulière, car elles ne sont prises en compte par aucun modèle de système réparti développé jusqu'à maintenant :

— la *géo-localisation* des dispositifs, rendue possible par le développement des technologies de localisation de type GPS,

— la *mobilité* des dispositifs, autonomes en énergie et en lien direct avec l'utilisateur qui les transporte (cas des smartphones ou des véhicules communicants).

Ainsi, je me suis intéressé, dès mon arrivée au LAAS, à l'étude de tels systèmes répartis portés par les humains. Dans un système réparti classique, les algorithmes développés s'appuient sur des hypothèses minimales et réalistes de stabilité, comme le nombre maximal de défaillances, pour fournir des mécanismes garantis malgré l'incertitude du système. Dans un système réparti mobile, porté par l'humain, le problème majeur est de déterminer des sources de stabilité, pas nécessairement statiques, qui vont fournir la « graine » sur laquelle il sera possible de construire des algorithmes offrant des services prouvés.

Ce chapitre expose mes travaux en vue de comprendre et d'utiliser la dynamique humaine selon deux échelles : les déplacements humains macroscopiques, et les micro-mobilités observées dans les foules d'humains. Dans l'ensemble de ces travaux, contrairement aux approches de type robotique, le but a été de comprendre

et d'utiliser la mobilité humaine : la mobilité doit être une composante du modèle du système réparti considéré, mais ne peut être contrôlée par l'algorithme.

Cette capture de données est un travail essentiellement expérimental. La première partie de ce chapitre décrit les deux points de départ de ces expériences : le développement d'un simulateur de flotte de véhicules pour le projet européen HIDENETS [Pr15], et le développement d'un outil de suivi de l'activité humaine sur smartphone. La combinaison des outils logiciels et matériels de ces deux expériences initiales a donné naissance à la plateforme SOUK [Pr11], décrite dans la seconde partie, qui vise à étudier finement les déplacements et interaction humaines. La troisième partie présente quelques travaux d'analyse liés aux données capturées.

5.1 Macro-mobilité

5.1.1 Plateforme pour l'évaluation d'algorithmes inter-véhiculaires

Comment évaluer les qualités d'algorithmes développés pour un réseau de véhicules communicants ? Il est irréaliste de mener des campagnes de test extensives avec de vrais véhicules. De plus, avoir des expériences reproductibles permettant de comparer plusieurs algorithmes est impossible dans l'environnement ouvert où évoluent des véhicules.

FIGURE 5.1 – Plate-forme d'évaluation d'algorithmes inter-véhiculaires

J'ai développé avec d'autres collègues du LAAS une plateforme d'évaluation expérimentale d'algorithmes pour systèmes mobiles [C36, C38, C41], qui a servi de démonstrateur au projet européen HIDENETS [1] [Pr15]. Le but de la plateforme étant de reproduire dans un laboratoire des scénarios réalistes d'un système de véhicules communicants, ces travaux ont porté sur la réduction d'échelle cohérente de tous les paramètres d'un tel système.

Plus précisément, considérons un terrain expérimental de $250000\,\mathrm{m}^2$ qui doit être reproduit dans une salle d'expérimentation de $100\,\mathrm{m}^2$. La réduction d'échelle en résultant est un facteur 50 en distance (2500 en surface). La table 5.1 résume, pour chaque périphérique à considérer, les contraintes résultant de cette réduction d'échelle.

1. `http://homepages.laas.fr/~mroy/hidenets/hidenets-demo.mp4`(2008)

Périphérique	**Précision réelle**	**Précision simulée**
Communication sans fil	portée : 100 m	portée : 2 m
GPS	5 m	2 cm
Taille véhicule	quelques mètres	quelques décimètres
Vitesse véhicule	quelques $^{m}/_{s}$	moins de 1 $^{m}/_{s}$

TABLE 5.1 – Réduction d'échelle : émuler un système inter-véhiculaire dans un laboratoire

Comme décrit dans les articles, la plate-forme est un modèle réduit de réseau de communication inter-véhiculaire comprenant 4 robots équipés de Wifi « maison » et de différents systèmes de localisation exécutant un programme réparti. Une attention particulière a été portée au développement d'une interface réseau sans fil à portée variable, en utilisant un système d'atténuateurs de signal et des « chaussettes de Faraday » visible sur la Figure 5.1. Sur cette plateforme, l'utilisateur peut programmer des motifs de mobilité qui seront suivis par les robots. On peut donc évaluer un algorithme donné avec différents motifs de mobilité programmés, mais aussi comparer plusieurs algorithmes dans les mêmes conditions expérimentales, chose qui est impossible en environnement réel.

5.1.2 Déplacement macroscopique des humains

Pour produire des jeux de données de mobilité, qui peuvent entre autre être injectés dans la plateforme décrite ci-dessus, je me suis intéressé à la modélisation de la mobilité des utilisateurs par la mesure des déplacements. J'ai développé le logiciel `phonEthic`, sur plateforme Symbian, traçant l'activité d'utilisateurs sur des téléphones portables équipés de GPS [C39]. Ainsi, nous avons équipé un groupe de volontaires de ce logiciel pour enregistrer l'ensemble des activités et évènements observables (GPS, wifi, bluetooth) de cet ensemble d'utilisateurs afin d'obtenir des traces réelles d'activité humaine.

Les caractéristiques majeures de ce développement ont été :

— le développement de l'application garantissant une utilisation minimale des ressources du smartphone considéré, en particulier pour l'utilisation de la batterie et de la mémoire,
— le déploiement d'une plateforme répartie de capture de données de l'environnement, en plus des informations de mobilité (accéléromètre, voisins bluetooth, points d'accès wifi),
— la fourniture d'une architecture respectant la vie privée des utilisateurs du service tout en permettant la capture de données hautement sensibles. Le service de capture pouvait être arrêté à tout moment par l'utilisateur. Les données étaient chiffrées localement puis transmises et stockées sur un serveur du laboratoire sous forme anonymisée.

5.2 Micro-mobilité et interactions : le cas des foules

5.2.1 Capture des déplacements et interactions dans les foules

Si les déplacements des humains peuvent être désormais relativement facilement capturés en utilisant un smartphone, cette approche possède plusieurs limitations dues à la technologie de localisation employée qui empêche une étude fine de la localisation et des interactions entre utilisateurs. Pourtant, pour comprendre la dynamique de systèmes portés par l'humain, il est nécessaire de relier les aspects sociaux et spatiaux des déplacements humains.

La capture sur smartphone décrite ci-dessus ne peut être faite qu'en extérieur et n'offre pas la précision nécessaire à l'étude des interactions inter-utilisateurs. Ainsi, en 2012, en tirant profit de l'expérience acquise sur les systèmes de localisation en intérieur utilisés dans la plate-forme d'évaluation de systèmes véhiculaires, nous avons développé avec Gilles Trédan la plate-forme SOUK [Pr11]—*Spatial & Social Observation of hUman Kinetics*— que nous avons déployée pour la première fois lors de l'inauguration d'un bâtiment au LAAS.

SOUK vise à suivre les déplacements et interactions au sein d'une foule dense d'individus. Pour ce faire, nous avons couplé un système de localisation sur étagères développé par Ubisense [2] à des logiciels que nous avons développés pour analyser le flot de positions reçu en temps réel.

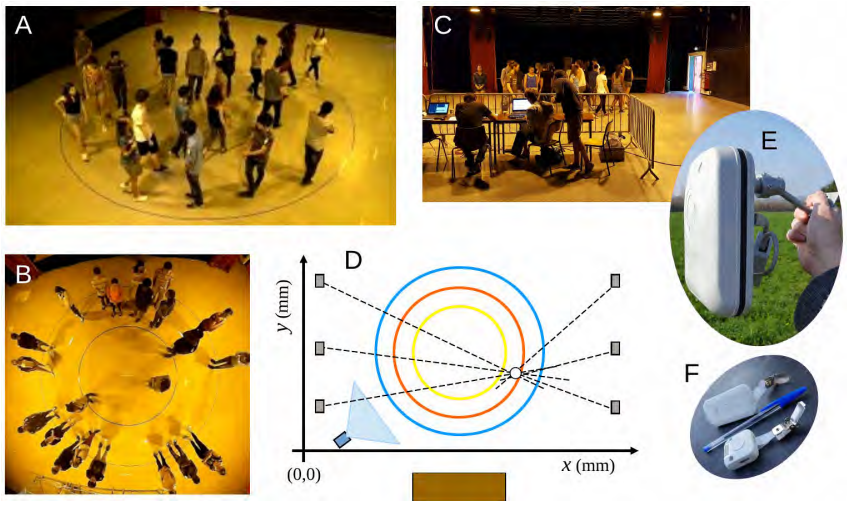


FIGURE 5.2 – **A, B :** Capture des positions d'un groupe, **C :** Régie de contrôle, **D :** Configuration des capteurs, **E :** Capteurs UWB, **F :** Tags UWB. **Images :** Ramon Escobedo Martinez (CRCA)

Plateforme de capture de positions La plateforme matérielle est constituée de deux types d'éléments : un réseau de capteurs fixes, et un ensemble de tags placés sur les utilisateurs. La Figure 5.2 montre les éléments du système de capture lors d'une expérience.

2. `https://www.ubisense.net/product/dimension4`

Lors du déploiement, les capteurs sont fixés et localisés très précisément afin de servir de support à la localisation. Tous les capteurs sont synchronisés de manière très précise par un réseau câblé ethernet, et reliés à un serveur qui va collecter et analyser les données des capteurs. Les tags émettent à tour de rôle des trains d'onde UWB, qui sont capturés et analysés par les capteurs. Les trains d'onde UWB ont la particularité d'être très robustes aux réflexions et aux obstacles, rendant cette technologie très adaptée à la capture dans les environnements denses. Pour une meilleure précision, nous attachons les tags aux épaules des participants, visibles sur la Figure 5.3 à gauche.

Un système bien configuré fournit une précision pour chaque tag de l'ordre de 10 cm, ce qui nous permet de calculer l'orientation des utilisateurs, et d'analyser les interactions réelles entre les participants.

Les tags se partagent le réseau en utilisant un mécanisme de type TDMA (*Time Division Multiple Access*) pour éviter les collisions. Si c est le nombre d'émissions par seconde autorisées par le TDMA, f la fréquence de rafraîchissement des positions, et n le nombre d'utilisateurs, alors la fréquence est déterminée par $f = \frac{c}{2n}$. La version (modifiée) du matériel que nous avons permet 137 émissions par seconde, ce qui permet de suivre 64 utilisateurs à une fréquence d'au moins 1 Hz.

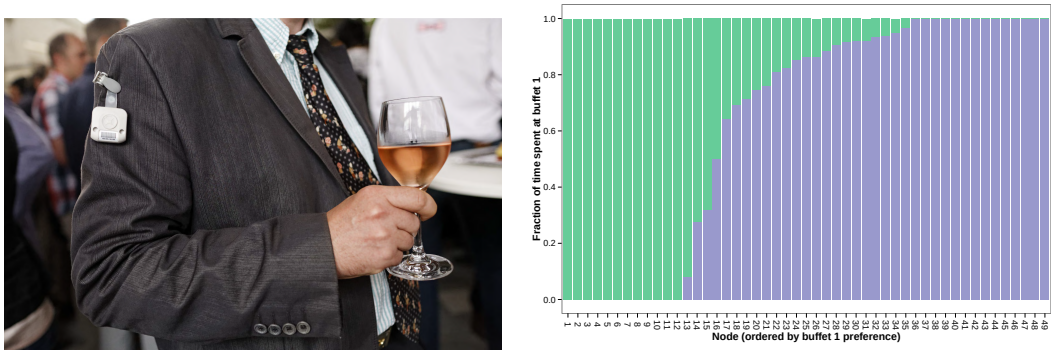


FIGURE 5.3 – **Gauche :** Placement des équipements de capture, **Droite :** Temps passé aux deux buffets pour chaque participant au cocktail d'inauguration du bâtiment ADREAM au LAAS.

Analyse des données À partir des données capturées, de nombreuses analyses sur le comportement et les interactions des participants peuvent être conduites en utilisant le pipeline logiciel que nous avons développé [J6, C27]. Lors de l'inauguration du bâtiment ADREAM au LAAS, nous avons capturé les interactions pendant le cocktail d'inauguration. La Figure 5.3 à droite montre le temps passé sur chaque zone de buffet, et l'on observe des préférences non uniformes entre les participants : alors que l'affluence aux deux buffets était équivalente, l'analyse par participant suggère que chaque participant a une préférence ou une spécialisation sur un buffet donné, probablement liée aux groupes de discussion près des buffets, ce qui est précisément le type de « graine » de stabilité que l'on recherche pour l'utiliser dans des algorithmes.

Pour ce qui est des modèles de mobilité, le modèle le plus employé dans la littérature est le modèle *random way-point (RWP)* dans lequel chaque humain tire

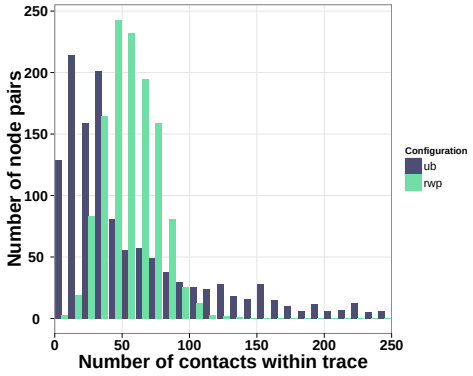


FIGURE 5.4 – **Gauche :** Comparaison des distributions de contact entre un modèle *Random Way Point* et les données mesurées. **Droite :** Expériences sur les comportements en groupe

au hasard une direction, s'y rend, et recommence. La Figure 5.4 à gauche montre une différence frappante entre la mobilité et les interactions mesurées et celle donnée par un modèle RWP calibré en utilisant les données mesurées sur les participants. Supposons que deux dispositifs portés par les utilisateurs peuvent communiquer s'ils sont à une distance d'au plus 2 m.La Figure 5.4 montre la distribution des contacts : on peut voir qu'un modèle RWP donne une distribution normale centrée sur 50, alors que les comportements mesurés donnent une distribution à queue lourde, certains utilisateurs étant connectés la plupart du temps, et d'autres ne se voyant quasiment jamais.

Le but premier de la plateforme SOUK est la capture des interactions « libres » au sein d'un évènement social, qui permet d'obtenir des jeux de données pour comprendre le lien entre le comportement social et spatial des utilisateurs. De nombreux travaux sont encore en cours, et les analyses déjà menées montrent clairement que, pour être réaliste, un modèle de mobilité humain doit prendre en compte une composante sociale : il ne semble pas réaliste de modéliser chaque humain indépendamment, et c'est justement dans les interactions avec les autres humains que la mobilité prend tout son sens.

Analyse des comportements de groupe Une autre application de la plateforme SOUK est d'étudier les comportements collectifs de groupes d'humains. Un exemple issu de la physique est l'étude du comportement de ségrégation en deux groupes, de la même manière qu'une vinaigrette finit toujours par se séparer en huile et vinaigre.

Dans une expérience préliminaire, menée en collaboration avec Guy Théraulaz (Centre de Recherche sur la Cognition Animale, Toulouse) et Clément Sire (Laboratoire de Physique Théorique, Toulouse), nous avons placé un groupe de volontaires dans une arène circulaire. Chaque participant possède deux bonnets, un rouge et un bleu. Un programme contrôlant l'expérience tire périodiquement une distribution aléatoire équilibrée attribuant à chaque participant une couleur qui est affichée sur un écran. Chaque participant met alors le bonnet correspondant et les participants doivent se séparer en deux groupes rouge et bleu. Nous avons utilisé les capacités

de la plateforme pour calculer le temps nécessaire à la ségrégation en deux groupes (voir Figure 5.4), en faisant varier la taille du groupe.

Partant de ces données représentant le comportement en présence d'une information globale (les utilisateurs voient les bonnets de tous les participants), nous avons décidé d'étudier la quantité d'information optimale pour résoudre le problème de la ségrégation en deux groupes. Pour ce faire, nous avons développé un mécanisme de retour auditif sur les tags portés par l'utilisateur, illustré sur la Figure 5.5. Dans chaque expérience, chaque participant a une couleur (rouge ou bleu) qu'il ne connaît pas. Le système développé calcule à chaque instant la couleur dominante dans le voisinage de chaque participant, et le tag « bippe » si le voisinage est majoritairement d'une couleur différente de celle du participant. L'expérience se termine lorsqu'aucun des participants ne bippe, c'est-à-dire que les participants sont bien séparés en deux groupes de même couleur. Nous avons conduit plus de 100 expériences réparties sur deux semaines pour déterminer la taille optimale du voisinage à considérer pour minimiser le temps nécessaire à ségréguer les participants en deux groupes.

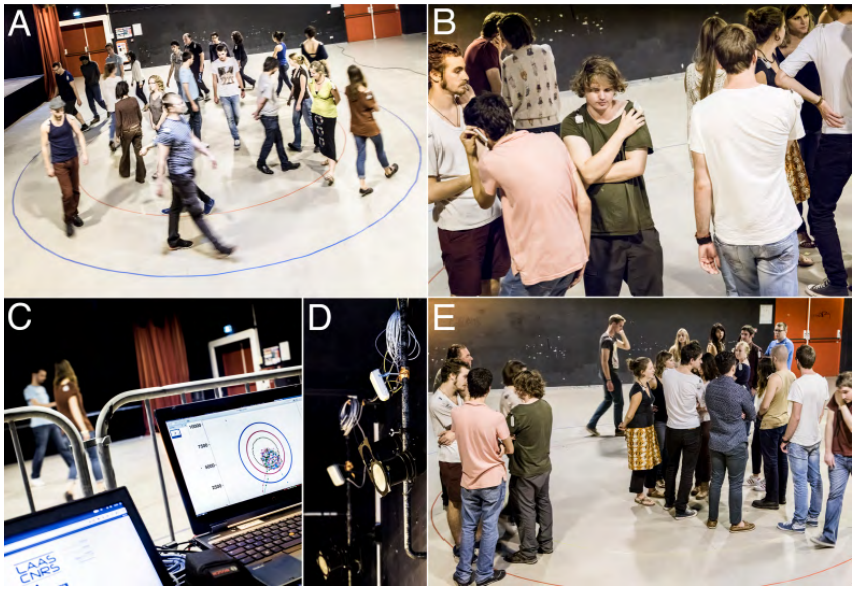


FIGURE 5.5 – **A :** Déplacement libre, **B :** Phase de ségrégation basée sur un retour auditif, **C :** Régie de contrôle, **D :** Capteurs ubisense, **D :** Ségrégation en 2 phases

Les résultats de ces travaux, en cours de finalisation, montrent qu'il existe une quantité d'information agrégée optimale pour résoudre ce problème, en l'occurrence il faut considérer les 7 plus proches voisins.

Originalité de la plateforme De nombreuses campagnes de capture de données de mobilité ont été réalisées par le passé, et publiées sur le projet CRAWDAD [3]. La Table 5.2 montre une sélection de jeux de données disponibles librement et contenant des informations de mobilité. Étant donné que ces campagnes utilisent du matériel sur étagères (des smartphones en général) pour capturer les données, la position est

3. `http://crawdad.cs.dartmouth.edu`

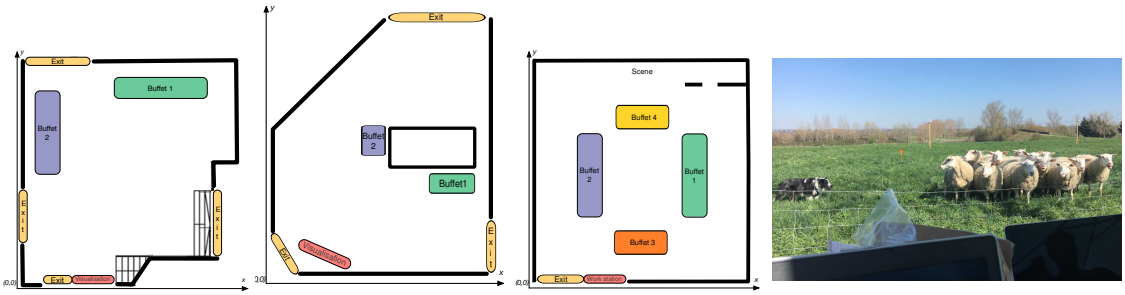


FIGURE 5.6 – De gauche à droite : cocktails ADREAM, Milan, Le Cap et moutons.

en général donnée par un GPS ou une interface réseau sans fil.

En comparaison, notre plateforme SOUK se place sur une *échelle* différente : en raison de contraintes d'installation complexes, SOUK capture les positions des utilisateurs dans un environnement spatio-temporel plus réduit que les autres jeux de données, mais fournit une précision inégalée dans la capture, rendant possible l'analyse fine des interactions dans une foule.

Jeu de données	**Nombre d'utilisateurs**	**Durée**	**Résolution fréquentielle**
Yellow cabs	100	1 mois	GPS : 1/min
Reality Mining	100	9 mois	Bluetooth : 12/h
UIUC-UIM	28	3 semaines	BlueTooth : 1/min, WiFi : 1/h
Nokia	200	1 an	GPS : défini par utilisateur
Yonsei/Lifemap	8	2 mois	GPS, WiFi 12 – 30/h
SOUK	50	1.5h	Ultra-Wide Band : 2/s

TABLE 5.2 – Comparaison de SOUK avec d'autres jeux de données disponibles

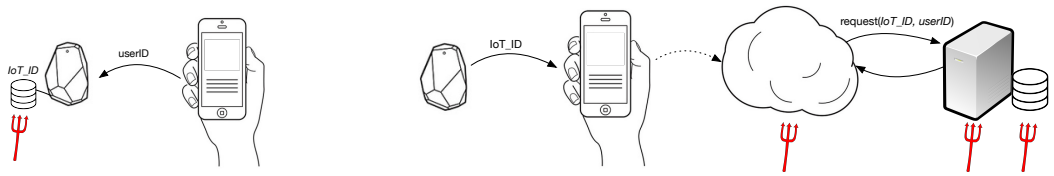
En plus de ce jeu de données librement disponible sur les interactions dans un cocktail [Pr11], nous avons capturé les interactions et déplacements dans plusieurs contextes différents : suivi d'un groupe de danseurs avec un chorégraphe travaillant sur le mouvement primaire à Milan (Matteo Lafranchi, compagnie Effetto Larsen en mai 2015 [4], en collaboration avec Hugues Chaté (CEA)), suivi d'évènements sociaux à Milan et dans la salle de spectacle Le Cap en septembre 2015, et même le suivi des interactions entre un troupeau de moutons, un berger et un chien de berger, en collaboration avec l'INRA Toulouse.

5.2.2 Analyse des interactions dans les foules

Modéliser les comportements

Les travaux de thèse de Roberto Pasqua [Th4] ont porté sur la capture et l'analyse de données de mobilité et d'interaction dans les foules, dans le cadre du projet SOUK. En particulier, nous avons utilisé les jeux de données de mobilité de SOUK pour simuler ce que pourrait capturer un ensemble de périphériques espions bluetooth [C9], et déployer une attaque visant à découvrir le réseau social d'un évènement à l'aide de capteurs bon marché.

4. `http://effettolarsen.it/old/Stormo_revolution_sci.html`



Périphérique espion bluetooth

Corruption du fournisseur de service ou de l'infrastructure

FIGURE 5.7 – Scénario d'attaque. La surface d'attaque est représentée par les tridents.

Considérons un environnement classique de l'internet des objets (IoT), où des capteurs de type iBeacon sont disséminés dans l'environnement. Classiquement, ils émettent un signal reçu par le téléphone porté par l'utilisateur qui relaie cet identifiant au fournisseur de service, ou écoutent les périphériques bluetooth à portée. Dans ces deux cas décrits sur la Figure 5.7, le système génère une liste d'évènements de co-localisation de dispositifs IoT et d'utilisateurs. À partir d'un ensemble de K logs de ce type, nous avons proposé une méthode de traitement permettant d'estimer la distance entre les objets IoT ayant généré les logs, et de re-calculer le réseau social entre utilisateurs, comme décrit succinctement sur la Figure 5.8.

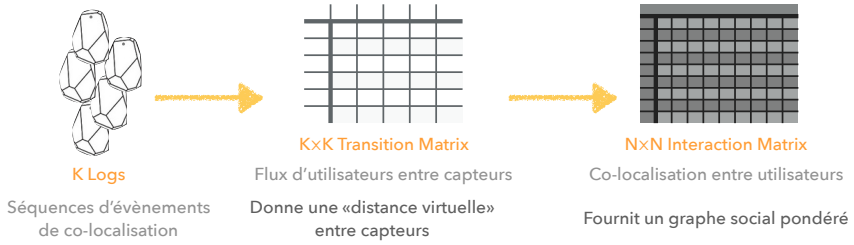


FIGURE 5.8 – Pipeline de traitement des logs

L'algorithme proposé n'utilise pas la localisation des utilisateurs, contrairement aux autres travaux sur le sujet [32], et fournit de meilleurs résultats que d'autres approches similaires.

Les co-localisations et interactions entre humains contiennent tellement d'information qu'il est possible de recalculer une « carte » de l'environnement, comme représenté sur la Figure 5.9.

Les jeux de données de SOUK ont été utilisés 1) pour simuler ce qu'un périphérique bluetooth placé en un point de l'espace verrait au cours de l'expérience, et 2) pour calculer une vérité de terrain sur le réseau social des personnes en présence. Cela montre à quel point fournir des jeux de données de traces de mobilité réelles à haute précision est une aide précieuse pour l'évaluation d'algorithmes et de mécanismes.

Évaluation d'algorithmes classiques sur une foule

Une autre application possible de ces jeux de données est l'évaluation des performances d'algorithmes « classiques » sur des systèmes mobiles. Nous avons ainsi

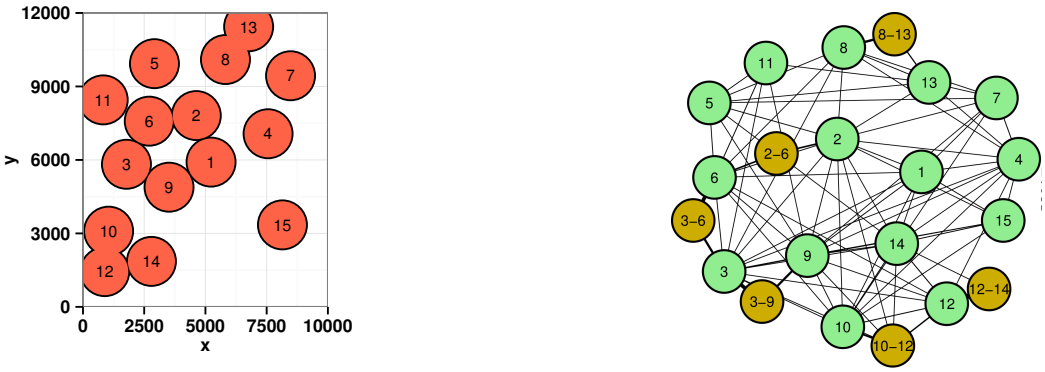


FIGURE 5.9 – **Gauche :** Déploiement de 15 capteurs en utilisant une stratégie basée sur les points d'intérêt. **Droite :** Représentation de la matrice de transition comme un graphe en utilisant l'algorithme Fruchterman et Reingold.

évalué les performances de l'algorithme le plus emblématique de partage de données : l'implémentation d'un service de partage de mémoire fournissant deux opérations, lire et écrire [26].

À partir des données réelles, il est relativement simple d'évaluer la distribution du temps de propagation d'une information en utilisant une communication pair-à-pair, de calculer la distribution des temps d'aller-retour pour un motif requête-réponse, voire même de faire varier la taille des quorums mis en œuvre afin d'optimiser une implémentation pour un système mobile porté par les utilisateurs [C16].

5.3 Analyser les évolutions des réseaux issus de l'activité humaine

Les réseaux induits par l'activité humaine (réseaux d'interaction, réseaux de transport...) ont des caractéristiques particulières qui ne sont pas capturées par des graphes aléatoires simples [33]. L'étude des *complex networks*, de leur structure topologique et des mécanismes permettant de générer des graphes synthétiques ayant des caractéristiques similaires aux graphes observés est une branche très active de la recherche en physique [34, 35].

Cependant, peu de résultats existent sur la *dynamique* de ces réseaux, qui sont pourtant en évolution permanente : tout réseau complexe observé est le résultat d'un processus évolutif qui est encore incompris.

Comparaison de graphes et distance On peut remarquer que peu d'outils permettent de comparer deux graphes. La mesure classique de similarité entre deux graphes est la distance d'édition d_{GED} [36] ou *graph edit distance*.

Étant donnés deux graphes G_1 et G_2, $d_{GED}(G_1, G_2)$ est définie par le nombre minimal d'opérations d'édition nécessaires pour passer de G_1 à G_2. En fonction du contexte, les opérations d'édition autorisées sont variables, mais incluent en général l'insertion et la suppression de nœuds et d'arcs.

La distance d'édition joue un rôle important pour les applications d'analyse de motif et de reconnaissance, mais son intérêt pour l'étude de l'évolution de graphes issus de l'activité humaine est limitée : l'ensemble des graphes à une distance d'édition donnée d'un graphe initial est très hétéroclite — la distance d'édition n'a aucune sémantique, et ne capture aucune structure que l'on rencontre généralement dans les graphes réels.

Distance basée sur les centralités Les centralités [37] forment une famille de mesures permettant d'extraire des informations sur la structure d'un réseau, et plus particulièrement sur le rôle qu'un nœud du réseau a vis-à-vis du reste du réseau. Il existe de nombreuses centralités qui capturent des aspects différents du réseau : *degree*, *closeness*, *betweenness*, *pagerank*, *eigenvector*...

Définition 8 (Centralité) *Une* centralité C *est une fonction* $C\colon (G, v) \to \mathbb{R}+$ *qui associe à un graphe* $G = (V, E)$ *et un nœud* $v \in V(G)$ *une valeur non-négative* $C(G, v)$.

Nous avons défini la notion de distance en centralité [J4, C23], dans le but d'étendre la notion de centralité à la comparaison de deux graphes et de fournir une distance liée aux propriétés structurelles des nœuds au sein des graphes :

Définition 9 (Distance en centralité) *Soit une centralité* C*. La distance en centralité* $d_C(G_1, G_2)$ *entre deux graphes est la somme sur les nœuds des différences en centralité dans les deux graphes :*

$$d_C(G_1, G_2) = ||C(G_1) - C(G_2)||_1 = \sum_{v \in V} |C(G_1, v) - C(G_2, v)|$$

Évolution de réseaux dynamiques Nous avons montré l'utilité de la distance proposée en étudiant des séquences de réseaux issus du monde réel. La méthodologie utilisée dans cette partie expérimentale de validation présente, à mon sens, un intérêt en soi [J4].

L'intuition de la méthode d'évaluation est la suivante : à partir d'un réseau donné G_t et de son évolution G_{t+1}, une évolution purement aléatoire de G_t à une distance d'édition $d_{GED}(G_t, G_{t+1})$ représente un ensemble de graphes de comparaison S_{t+1}, ou null-model [38]. Ensuite, nous vérifions si G_{t+1} est une aberration par rapport au *null model* S_t par rapport à la centralité C. Plus précisément, G_{t+1} est une aberration si sa distance à G_t moins la distance moyenne de S_{t+1} à G_t est supérieure à la déviation standard, c'est-à-dire :

$$|d_C(G_t, G_{t+1}) - \ \mu(\{d_C(G_t, x), x \in S_{t+1}\})| \ > 2\sigma(\{d_C(G_t, x), x \in S_{t+1}\}).$$

Cette évaluation permet de mesurer la déviation de la dynamique par rapport à une évolution purement aléatoire. Cette méthodologie nous a aussi permis de définir des *signatures* qui utilisent un ensemble de centralités pour qualifier la dynamique d'une séquence de graphes : la signature est définie comme la fraction d'aberrations dans une séquence pour un ensemble de centralités. De par la méthodologie utilisée, la signature ne dépend ni de l'échantillonnage temporel du réseau étudié, ni de sa

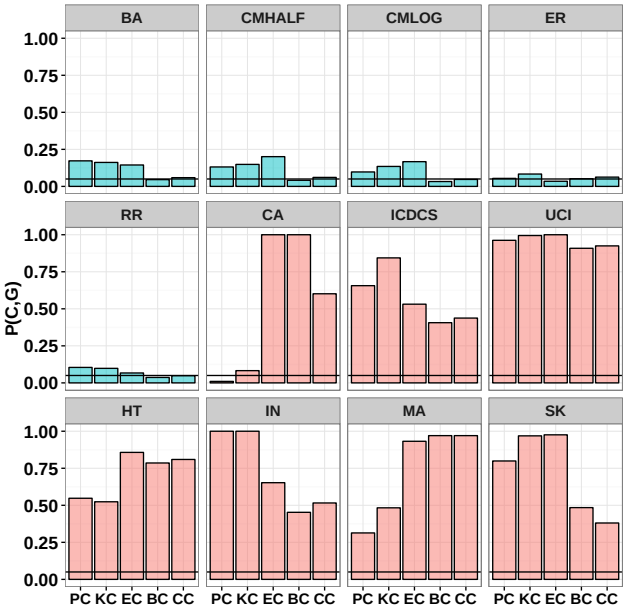


FIGURE 5.10 – Représentation des *signatures de dynamicité* sur des jeux de données synthétiques et issus du monde réel. Pour chaque jeu de données, l'histogramme montre pour chaque centralité $p_{C,T}$, la probabilité que G_{t+1} soit une aberration vis à vis du *null model* de cette centalité. La ligne horizontale à 5% correspond au *null model* de la centralité (le pourcentage de graphes à distance au moins 2σ de la moyenne d'une distribution normale).
Centralités : PC : pagerank KC : closeness EC : ego BC : betweenness CC : closeness
Données synthétiques (bleu) : BA : Barabasi-Albert, CMHALF : attachement préférentiel équiprobable CMLOG : Attachement préférentiel, distribution logarithmique, ER : Erdös-Rényi, RR : Random Regular.
Données réelles (rouge) : CA : réseau des AS de l'Internet, ICDCS : co-auteurs ICDCS, UCI : Réseau social de l'UCI, HT : interactions pendant la conférence Hypertext, IN : Interactions pendant l'évènement "Infectious : Stay away" MA : Mails échangés dans une entreprise, SK : données de cocktails SOUK.

taille. La Figure 5.10 présente quelques résultats concernant les réseaux étudiés. Elle montre en particulier que les distances en centralité rendent compte de la dynamique observée, et que, suivant le jeu de données considéré, certaines centralités rendent mieux compte de la dynamique du réseau, en particulier selon le caractère local ou global de la centralité considérée. On peut remarquer aussi que les jeux de données générés par des modèles sont plus proches du *null model*, ce qui appelle à développer de nouveaux modèles prenant mieux en compte la dynamique structurelle des réseaux observés.

5.4 Bilan et notes

Ce chapitre se concentre sur la mesure des mobilités humaines pour la compréhension de la dynamique des structures induites par cette mobilité —la cible principale est ici un système réparti porté par les utilisateurs.

Historiquement, la partie expérimentale de ces recherches a été la convergence de

deux projets : une plateforme d'émulation de systèmes automobiles dans un laboratoire, et un projet de mesure de l'activité humaine sur smartphones. J'ai développé dans le cadre du projet HIDENETS [Pr15] une plateforme en « modèle réduit » d'un système de véhicules communicants. Les défi majeur relevé dans ce projet a été la réduction de l'échelle dans tous les composants (véhicule, capacités de communication, localisation).

Dans un système informatique collaboratif communicant sans infrastructure globale, la connexion entre deux terminaux est induite par la co-localisation des utilisateurs qui portent les terminaux, elle même influencée par les interactions sociales entre les utilisateurs. La compréhension de la dynamique des systèmes répartis portés par les utilisateurs nécessite donc la compréhension des interactions sociales.

C'est pourquoi j'ai développé avec G. Trédan la plateforme Souk, qui permet d'étudier le lien entre la mobilité spatiale et la mobilité sociale. Souk s'appuie sur les technologies de localisation très précises utilisées dans la plateforme d'évaluation de systèmes mobiles précédemment développée. Souk utilise la technologie radio ultra-large bande pour suivre finement les déplacements et interactions d'individus en groupes denses, ou foules.

Souk a d'abord été utilisé pour capturer des jeux de données d'interactions « libres », c'est-à-dire les mouvements de personnes qui interagissent dans des évènements sociaux, afin de comprendre la dynamique sociale du déplacement. Ces jeux de données ont été utilisés dans un second temps pour évaluer les risques portés sur la vie privée par les équipements IoT capables de connaître les utilisateurs à leur portée. Ils nous ont aussi permis de mener des analyses préliminaires de performances d'algorithmes répartis classiques dans un cadre mobile.

Les analyses menées grâce à Souk dépassent le cadre de l'informatique, et ce projet a fédéré localement des recherches sur les comportements collectifs, en collaboration avec G. Théraulaz (Centre de Recherches sur la Cognition Animale) et C. Sire (Laboratoire de Physique Théorique). Souk permet pour la première fois de faire des expériences avec des humains pour étudier la réponse d'un groupe à l'information locale donnée aux participants d'une tâche collective. Nous avons développé une plateforme expérimentale pour étudier la quantité optimale d'information à donner pour optimiser le temps de réalisation d'une tâche simple de séparation en deux groupes.

En collaboration avec S. Schmid (U. Vienne) et Y-A. Pignolet (ABB), nous avons étudié le dynamisme des graphes d'interactions évoluant dans le temps. En remarquant qu'il n'y a pas d'outil de mesure de distance entre deux graphes d'interaction prenant en compte la sémantique de ces graphes, nous avons défini et utilisé une distance basée sur les centralités dans les graphes, qui capture les rôles qu'ont les nœuds dans un graphe. Nous avons montré que ces distances capturent effectivement bien la dynamique observée, que ce soit dans les jeux de données de mobilité que nous avons capturés, ou dans d'autres graphes dynamiques issus de l'activité humaine (réseaux sociaux, réseaux d'infrastructure...) Ces travaux ouvrent de nombreuses perspectives sur l'analyse des réseaux dynamiques, en particulier pour la prédiction de leurs évolutions.

Publications de référence La plateforme d'expérimentation de systèmes mobiles est décrite succinctement dans ACM PODC'09 [C41] et dans AMiRE'11 [C36]. La plateforme Souk est décrite dans ACM Ubicomp'13 [C27] et dans *Communication Networks* [J6]. L'utilisation des jeux de données pour évaluer les dangers de l'IoT sur la vie privée est publiée dans ACM Ubicomp'16 [C9]. L'étude des dynamiques des réseaux d'interaction en utilisant les centralités est publiée dans une version simplifiée dans ACM FOMC'14 [C23], puis en version complète dans le *Journal of Statistical Mechanics : Theory and Experiment* [J4], un journal de physique orienté vers l'analyse des systèmes complexes.

Chapitre 6

Perspectives

> Le but n'est pas seulement le but, mais le chemin qui y conduit.
>
> Lao Tseu

Les pistes de recherche présentées dans ce chapitre se concentrent sur trois grands thèmes qui couvrent la théorie et la pratique des systèmes informatiques résilients : *i*) le *développement d'outils* pour la capture fiable de données dans les grands systèmes répartis, pour leur diffusion vers la société et leur utilisation pour modéliser ces systèmes, *ii*) les *architectures* pour implémenter la résilience dans les environnements complexes, et *iii*) les *algorithmes* et outils mathématiques pour les systèmes répartis, à grande échelle ou sur une puce.

6.1 Évolution des réseaux complexes

6.1.1 Développer des outils de mesure répartis ouverts

Ouvrir les données pour lutter contre la monétisation

Les données de mobilité des utilisateurs (piétons, vélos et voitures) sont une sous-partie critique des données personnelles, essentielles pour la recherche dans de nombreux domaines : sociologie, urbanisme, informatique, etc. Cependant, les grands acteurs d'Internet, communément nommés GAFA(M) (Google, Apple, Facebook, Amazon et Microsoft), et les opérateurs téléphoniques qui possèdent ces données refusent généralement de coopérer avec les chercheurs, et utilisent ces données comme source de revenus.

Il est crucial de lutter contre cette monétisation des données personnelles et rendre les données accessibles pour servir la société —hors de la volonté mercantile d'acteurs privés.

Je milite pour mettre le citoyen au centre de la capture de ces données pour plusieurs raisons :

— *Protection des données privées.* Un argument majeur des sociétés privées pour ne pas diffuser, même protégées, les données dont elles disposent est d'ordre

sécuritaire, soulignant que ces données ont un caractère trop sensible pour être diffusées. Or, ces mêmes sociétés utilisent voire même revendent ces données pour des services marchands. Mettre les producteurs des données (les citoyens) face à leurs propres traces, leur montrer l'ensemble des données privées qu'il génèrent, et surtout montrer ce qu'on peut en faire a comme effet secondaire bénéfique la prise de conscience chez les utilisateurs de la valeur et des dangers associés à ces données.

— *Collecte de grandes masses de données.* Dans plusieurs projets menés actuellement [Pr4, Pr11], de nombreux citoyens se portent naturellement volontaires. Participer à la capture de données « pour la science » est un acte de résistance citoyen pour de nombreuses personnes, ce qui garantira la collecte d'un grand volume de données.

— *La ville comme objet expérimental.* Les villes sont des zones à forte densité de population, et la collecte de données sur une fraction même réduite des habitants permettra d'avoir des informations pertinentes sur les interactions entre les habitants et les infrastructures de la ville, ainsi qu'entre les habitants eux-mêmes.

Capturer les données de manière répartie

Comme indiqué ci-dessus, impliquer les utilisateurs est une manière efficace de collecter de grandes quantités de données incluant la mobilité et la dynamique induite par les humains.

Je développe actuellement avec Gilles Tredan le projet CLUE [Pr4], dont le but est d'implémenter un outil de mesure réparti à l'échelle de la ville, porté par des vélos. Le but de ce projet est de capturer des traces de déplacement, couplées à d'autres paramètres mesurés dans l'environnement : mesures de pollution atmosphérique, température, état des routes via un accéléromètre, etc. Les participants sont volontaires et le système est invisible pour l'utilisateur, car autonome en énergie via la dynamo et embarqué sur le vélo.

Dans un cadre relativement similaire, je débute dans le cadre d'une thèse CIFRE avec Continental un projet de capture et de diffusion de données sur les réseaux de véhicules connectés, dont le but est d'utiliser le réseau des véhicules comme un capteur/actionneur réparti à grande échelle.

Collecte répartie : optimisation des ressources

Quel que soit le support de la collecte —les capteurs répartis peuvent être un réseau de bicyclettes, de smartphones ou de véhicules motorisés—, un point difficile à traiter est celui de l'optimisation des ressources utilisées par le processus de capture et de collecte.

Qu'il s'agisse de ressources énergétiques, réseau ou processeur, le processus doit être optimisé pour minimiser les désagréments pour l'utilisateur. Les défis techniques qui devront être pris en compte incluent :

— La quantité de données produites par chaque capteur étant potentiellement importante, le système doit coordonner les échanges de données pour s'adapter à une charge maximale donnée, évoluant dans le temps,

— La prise en compte de la dimension géographique du calcul, du traitement et du transfert des données est nécessaire, pour équilibrer la charge portée par les capteurs,
— Le système doit s'adapter à la charge globale, aux variations dans l'espace et dans le temps, en particulier aux heures de forte densité ou à la nuit, aux évènements non prédictibles, aux disparités dues à la variabilité des zones géographiques.
— La redondance potentielle des captures d'une même donnée doit être utilisée pour pallier la relative imprécision des mesures faites avec des dispositifs fragiles et placés dans un environnement incertain.

On peut ainsi remarquer que transférer un tel flux d'évènements avec certaines garanties dans un environnement contraint est très proche des problématiques d'ordonnancement typiquement rencontrées dans les systèmes embarqués temps-réel qui pourraient être étendus et adaptés au problème du transfert d'informations au sein du système. Ainsi, il serait intéressant de s'inspirer des travaux actuels de la communauté temps réel qui étendent les algorithmes traditionnels pour prendre en compte de nouvelles dimensions dans les problèmes d'ordonnancement. En particulier, des travaux récents se focalisent sur la dimension énergétique [39], c'est-à-dire au respect des contraintes de temps couplé à la minimisation de la consommation énergétique.

Diffuser et utiliser des données sensibles

Ces données étant porteuses de nombreux risques pour la vie privée des utilisateurs les ayant générées, une question cruciale est de développer des mécanismes permettant de les rendre utilisables.

Dans le cadre du projet européen PRIME [Pr18], j'ai travaillé sur la protection des informations privées en développant un système d'accès aux données multi-rôles : les accès aux données dépendent des droits et utilisations faites par les usagers de la base de données.

Ainsi, un urbaniste voudra accéder à des statistiques globales sur les trajets des vélos ou des voitures mais n'a pas besoin de connaître les trajectoires des utilisateurs. Un utilisateur peut obtenir ses données mais pas celles des autres utilisateurs sans leur consentement. Un sociologue sera intéressé par les co-localisations d'utilisateurs sans nécessairement savoir qui est impliqué.

Résoudre le problème de l'accès à ces données doit à mon sens se baser sur un contrôle d'accès multi-rôle [Ch3], qui doit être étendu à la gestion des traces de mobilité, qui est une source très importante de fuite d'informations privées.

6.1.2 Étudier les humains comme un système réparti

Dans le cas de l'étude des comportements collectifs des humains, évoqué dans la partie 5.2, un agenda de recherche raisonnable se base sur les quatre points suivants :

1. étudier l'influence de l'information « globale », dans sa qualité et sa quantité, fournie à un groupe d'utilisateurs pour réaliser une tâche collective simple ;
2. faire réaliser à des groupes de personnes des tâches collaboratives basées sur le déplacement et les interactions. Dans ce cadre, je compte évaluer les solu-

tions développées par des groupes de volontaires de problèmes classiques de l'informatique répartie (renaming, coloration de graphe, etc.), afin de comparer les stratégies d'un « cerveau réparti » composé d'humains qui collaborent, avec les solutions optimales pour un système réparti informatique ;

3. contrôler l'environnement, en développant des outils de réalité virtuelle permettant de contrôler et de mesurer les informations diffusées, les choix réalisés par les groupes de volontaires, et leurs influences respectives ;
4. coupler les points 2 et 3 pour faire réaliser à des groupes d'humains des tâches collectives dans un environnement augmenté, dans lequel la mobilité et les interactions réelles sont mesurées, et l'information contrôlée est ajoutée à l'environnement sous forme d'une sur-couche au monde réel.

Dans le cadre du projet Souk [Pr11] et de collaborations avec des chercheurs d'autres domaines (Clément Sire, physique théorique, Paul Seabright, sciences économiques, Guy Théraulaz, biologie), cet agenda est en cours de réalisation : la première étape est en cours, les étapes 2 et 3 sont en développement. La difficulté de cet axe de recherche vient de la complexité du développement et de la gestion des outils matériels, et ce projet ne peut se faire que dans un cadre pluridisciplinaire, aux frontières de la sociologie, de l'éthologie, de la théorie des jeux...

6.1.3 Modéliser les déplacements humains

Mes travaux précédents ont montré que, dans la mobilité des humains en groupe, les co-localisations sont un marqueur très important de la composante sociale du déplacement. Ce constat plaide pour le développement de modèles de mobilité de groupes de personnes, et non plus de personnes en isolation. Cependant, développer un modèle de calcul incluant l'espace, le temps et la dimension sociale semble intéressant mais difficilement atteignable directement.

Le points suivants détaillent les pistes que j'envisage pour tendre vers ce but.

Mobilité explicite : corrélations multi échelles

Les jeux de données de mobilité humaine capturés incluent plusieurs échelles, décrivant les interactions et micro-mobilités dans les environnements denses de type foule d'individus, et les déplacements macroscopiques à l'échelle de la ville. Pour aller vers la compréhension des dynamismes et des motifs stables qui sont contenus dans ces traces, il me semble pertinent de commencer par rechercher des corrélations entre les différentes échelles de granularité.

Abstraire les déplacements

Un modèle, au sens informatique du terme, n'a pas vocation à être précis mais doit être suffisamment représentatif de la réalité pour être utile, tout en étant suffisamment abstrait pour être utilisable dans un algorithme ou une preuve.

Un grande partie des recherches présentées dans la première partie du manuscrit se focalisent sur la mobilité, mais utilisent in fine les co-localisations (qui peuvent être recalculées à partir des jeux de données de mobilité). Ainsi, un modèle utilisable en

informatique qui permettrait de tirer profit de la réalité complexe, spatio-temporelle, et sociale des déplacements ne doit pas nécessairement décrire les déplacements mais les abstraire.

La mobilité et l'aspect géographique des systèmes n'est que le support des communications possibles entre les entités de calcul associées aux utilisateurs. Dans un système réparti, ce sont les interactions entre humains qui sont importantes car elles déterminent l'ensemble des communications locales possibles.

L'ensemble des interactions possibles à un instant donné est un graphe, une structure facile à utiliser en informatique, où un modèle discret est préférable à un modèle continu. Un modèle de motifs qui se retrouvent dans les séquences des graphes d'interactions offre des garanties sur le système sous jacent permettant de développer des mécanismes fiables.

Trois niveaux d'abstraction doivent être considérés, que je classe selon la complexité croissante de leur étude : 1) les modèles de co-localisation, 2) les modèles de regroupement (sous graphes complets), dont la stabilité et la connectivité permet d'imaginer des mécanismes de réplication et de pérennisation de stockage d'information, 3) les modèles de re-co-localisation (la répétition entre rencontres d'utilisateurs), qui décriraient les localités sociales, une autre source de stabilité qui pourrait être utilisée par des services informatiques.

Générer la dynamique des systèmes complexes

Dans la suite des travaux présentés sur les évolutions des réseaux issus de l'activité humaine [J4], et pour compléter les pistes évoquées ci dessus, il faut dépasser les approches classiques issues de la branche « systèmes complexes » qui se focalisent sur la détermination des processus engendrant les réseaux complexes observés, comme les processus de génération d'attachement préférentiel.

Analyser et prévoir la dynamique des réseaux complexes issus de l'activité humaine doit se faire en définissant des processus qui vont *générer la dynamique observée* et non plus le résultat de cette dynamique.

6.2 Architectures pour garantir la sûreté en environnement complexe

6.2.1 Architectures adaptables pour systèmes embarqués

Le concept même d'évolution et d'adaptation est antagoniste aux concepts qui ont traditionnellement sous-tendu la conception des systèmes embarqués. Comme montré lors des travaux de thèse d'Hélène Martorell [Th8], les capacités d'adaptation à l'exécution d'un système conçu de manière monolithique et incluant des contraintes temps réel impose de nombreuses limitations.

L'augmentation de la densité des puces en unités de calcul, couplée à l'augmentation de la quantité des services fournis par les systèmes embarqués qu'on rencontre dans les futurs systèmes automobiles nécessite de repenser les architectures pour prendre en compte l'exécution concurrente et fiable d'applications à multiple niveaux de criticité.

Les travaux présentés pour l'exécution garanties de tâches critiques en concurrence ouvrent des perspectives dans cette direction. L'approche proposée considère des tâches critiques et des tâches non critiques, ce qui est trop simpliste pour être appliqué dans un vrai système embarqué à multiple niveau de criticité.

Approche multi couches pour la multi criticité

Une première approche pour étendre ces résultats serait de considérer l'ensemble des niveaux de criticité s'exécutant en concurrence comme un ensemble concentrique de niveaux de qualité de service, à la manière d'un oignon. Il s'agirait d'organiser les tâches en strates offrant différents niveaux de fonctionnalité, depuis un mode dégradé dans lequel seules les fonctionnalités essentielles sont exécutées jusqu'au système complet, exécutant toutes les tâches.

Il s'agirait ensuite de précalculer les paramètres, essentiellement temporels, de ces différentes strates d'exécution afin de fournir un support lors de l'exécution permettant d'implémenter un contrôleur qui prendra la décision de passer d'une configuration à l'autre et ainsi modifier l'ordonnancement des tâches.

De manière pratique, ce contrôleur devra être implémenté efficacement en utilisant le support matériel disponible, en particulier un système d'hyperviseur, pour surveiller le système à l'exécution afin d'exécuter le système dans un mode optimisant les ressources disponibles tout en garantissant la sûreté de fonctionnement des applications.

6.2.2 Surveillance pour la résilience proactive

La surveillance à l'exécution est un outil pour déclencher une adaptation du système. Idéalement, le système de surveillance doit permettre d'implémenter la *résilience proactive*, c'est à dire détecter les potentiels problèmes avant qu'ils n'aient un impact sur l'exécution correcte du système, et reconfigurer le système avant qu'il ne soit trop tard.

Ce concept a été développé dans le cadre de la thèse de Thomas Robert [Th10], et étendu dans le cadre de collaborations —non décrites dans ce mémoire— autour des systèmes autonomes [J7, C11, C33, C34].

En supposant que le mécanisme d'adaptation lui même est fiable, alors le système de surveillance qui déclenche l'adaptation ne doit pas nécessairement être parfait : il suffit qu'il détecte toutes les déviations, car les faux positifs induisent une reconfiguration non nécessaire qui n'a d'impact que sur les performances. Une piste prometteuse dans ce sens serait de développer un mécanisme de surveillance d'anomalie basée sur le *machine learning* [C1], configuré pour détecter au moins toutes les déviations.

6.2.3 Le cas de l'intelligence artificielle

L'utilisation de plus en plus intensive de l'intelligence artificielle dans les systèmes pose de nombreux problèmes pour la sûreté de fonctionnement —le fonctionnement en « boîte noire » des mécanismes basé sur l'apprentissage est en contradiction avec l'approche basée sur les preuves de bon fonctionnement.

Comme évoqué dans le paragraphe précédent, l'IA peut être utilisée dans un système sûr de fonctionnement si son utilisation ne met pas en péril la sûreté du système.

Dans le cas général, il faut néanmoins développer des mécanismes permettant l'inclusion sûre de services basés sur l'apprentissage, que je déclinerais selon deux axes :

— pour l'inclusion fiable, « en aveugle », de mécanismes d'intelligence artificielle dans les architectures critiques, l'architecture logicielle du système doit fournir un mécanisme d'encapsulation (*wrapper*) fiable qui permet d'éviter les interférences qui pourraient mettre en danger la fiabilité du système. Ce cas se ramène aux points évoqués en partie 6.2.1 pour la gestion fiable des systèmes à multiple niveaux de criticité et est pour moi similaire au problème d'intégration de composants sur étagères non-fiables au sein d'un système fiable. Un exemple classique de ce cas est l'inclusion d'algorithmes de vision, qui ne sont prouvés ni fonctionnellement ni temporellement, dans un système embarqué automobile.

— pour garantir des réponses fiables de mécanismes d'apprentissage, les recherches récentes sur les exemples adversariaux et les GAN (*Generative Adversarial Nets* [40]) permettent de mettre le doigt sur les limites des approches de *deep learning*. Développer des méthodes d'apprentissage et de surapprentissage de réseaux qui permettraient d'augmenter les garanties fournies par les mécanismes d'IA, voire même d'implémenter un « cordon sanitaire » de sûreté —une zone grise clairement identifiée dans laquelle la réponse est incertaine— est une voie de recherche prometteuse.

6.3 Algorithmes et outils pour les systèmes répartis

De par leur grande variabilité et la complexité de leur architecture, les systèmes répartis offrent un terrain d'exploration quasiment infini d'un point de vue mathématique et algorithmique.

Cette section présente les pistes de recherches que j'ai identifiées dans ce domaine en me focalisant sur trois architectures particulièrement représentatives : les architectures multicœurs, les architectures manycœurs (*manycore*) et les réseaux dynamiques.

6.3.1 Multicœurs

Depuis une vingtaine d'année se développe la théorie de la calculabilité répartie basée sur des outils de topologie algébrique [41]. Cette théorie s'applique directement aux architectures multicœurs, car elle a été développée pour les algorithmes *wait-free*, qui ont une structure très régulière et permet ainsi d'avoir une théorie « simple ».

L'ensemble des résultats de cette théorie [42] s'intéresse à la calculabilité, et ne traite pas de la complexité. Pourtant, les outils topologiques permettent de mesurer la complexité d'une tâche de synchronisation à résoudre, mais ces mesures ne sont pas directement reliées au coût d'un algorithme en nombre d'opérations en mémoire.

L'outil de base utilisé pour simplifier cette théorie est un objet de haut niveau, le `snapshot`, qui fournit une vue globale de la mémoire, bien loin d'une simple lecture ou écriture en mémoire. Pour obtenir des résultats de complexité et des bornes inférieures en utilisant les outils de la topologie algébrique, deux voies peuvent être suivies :

— une première voie consiste à déterminer la borne inférieure en complexité de l'objet de base utilisé, le `snapshot`, et utiliser ce résultat pour voir si une bonne mesure de complexité en lecture/écriture pourrait être déduite de la théorie classique. Ce problème reste ouvert depuis plus de 20 ans.

— une seconde voie, plus directe, serait de raffiner la théorie en s'appuyant sur les objets de base des architectures multi-cœurs, les mots mémoire, ou registres.

6.3.2 Algorithmes et théorie pour les manycœurs

Les architectures multicœurs ont une limitation physique importante : le bus mémoire central est partagé par tous les cœurs. Même si actuellement le nombre de cœurs par puce augmente, ces architectures ne pourront pas offrir une telle simplicité avec un nombre de cœurs très important et seront rapidement limitées.

Les architectures manycœurs, développées par exemple par Intel et Kalray, proposent une architecture régulière mais d'une philosophie radicalement différente : les cœurs possèdent chacun une mémoire indépendante, et sont reliés par des liens réseau en grille ou en tore en deux dimensions. Cette architecture passe à l'échelle et permet d'augmenter le nombre de cœurs tant que la finesse de gravure diminue.

Très peu de résultats théoriques ou algorithmiques existent sur ces architectures réparties pourtant très régulières. Dans le domaine des systèmes répartis, le modèle Local [43] prend explicitement en compte le graphe de communication et peut être appliqué directement aux manycœurs.

Cependant, la majorité des résultats dans le modèle Local traite des anneaux, et le cas des tores n'a pas été étudié. Deux points méritent d'être investigués :

— Est-il possible d'étendre la théorie de la calculabilité utilisant la topologie algébrique à des structures locales de calcul, sans mémoire partagée ? La structure la plus simple à considérer serait un anneau, mais le cas intéressant pour les manycœurs serait la grille ou le tore.

— Quels résultats peuvent être montrés dans le modèle Local de calcul instancié sur sur grille ou un tore ?

Idéalement, il faudrait développer une « boîte à outils » de synchronisation de base implémentable sur les manycœurs, l'équivalent des algorithmes wait-free pour les multi-cœurs.

6.3.3 Systèmes répartis dynamiques

Robotique mobile théorique

Dans le but de relier des travaux issus de communautés différentes, j'ai travaillé récemment à la compréhension fine d'un modèle classique de robotique théorique

—le *look-compute-move*— qui a une structure similaire au *Sense-Compute-Control* des systèmes auto-adaptables.

Dans le cadre d'une collaboration avec Sergio Rajsbaum et Armando Castañeda (UNAM, Mexico), nous avons montré [J1, C5] que ce modèle peut être ramené à un modèle de topologie algébrique classique, ce qui permet entre autres d'en déduire quels algorithmes peuvent y être implémentés.

Ce modèle est d'une certaine manière trop théorique (un robot peut instantanément avoir une vue globale du système), et, pour être utilisable en pratique pour des robots mobiles, il est nécessaire d'affaiblir la primitive de vision (*look*). Dans ces modèles de robotique, la communication se fait uniquement par vision, et faire des ponts entre les résultats des deux communautés permettra d'utiliser de manière croisée les contributions issues des deux domaines.

Calculabilité dans les systèmes répartis dynamiques

Dans la partie 6.1.3, j'indiquais la nécessité de développer des modèles de dynamisme qui rendent compte de la mobilité sociale des humains, afin de pouvoir développer des algorithmes au dessus de ces modèles.

Une question duale est de déterminer, pour un problème donné, quelle dynamique maximale du graphe de communication autorise l'implémentation d'une solution.

Des travaux montrent les conditions sur le dynamisme maximal [44] d'un réseau de communication permettant d'implémenter un mot mémoire/registre. Ces résultats montrent quel dynamisme maximal permet de transposer l'ensemble de la théorie du wait-free aux systèmes dynamiques.

Il serait intéressant d'avoir le même type de résultats pour d'autres abstractions utiles pour implémenter des services fiables : le problème du consensus, le k-set-agreement, une variante plus simple du consensus...

De plus, plusieurs types de dynamiques sont envisageables pour un même problème à résoudre ; parmi toutes ces dynamiques, les plus pertinentes sont celles qui sont les « plus proches » des dynamiques observées sur le jeux de données et sur les modèles disponibles, comme indiqué en partie 6.1.3.

Confiance globale et blockchains

Enfin, dans un système décentralisé basé sur les interactions entre utilisateurs, une question cruciale est d'assurer une confiance dans le système réparti, malgré la méfiance que l'on peut avoir vis-à-vis d'un participant a priori.

Dans ce contexte, les systèmes de *blockchains* fournissent des algorithmes, non prouvés, qui implémentent une forme de confiance globale. Récemment, ce sujet est devenu un sujet de recherche, dans le but de formaliser et de mieux comprendre les algorithmes impliqués.

En plus de fournir un système transactionnel classiquement implémenté par les monnaies virtuelles, certains systèmes de blockchains, comme le système Ethereum, embarquent un langage Turing complet.

Au delà de l'embrasement médiatique autour de ces systèmes et de leurs déboires (en particulier pour Ethereum), comprendre, prouver et développer ces approches

totalement décentralisées permettrait d'implémenter des systèmes dynamiques et mobiles de collaboration de confiance.

Thèses encadrées

[Th1] J. Ibarz. « Équilibrage de charge efficace et adaptatif avec garanties spatio-temporelles pour les systèmes de véhicules connectés ». CIFRE Continental. 2018-2021.

[Th2] D. Loche. « Architecture Électrique/Électronique (EE) pour le véhicule du futur autonome et connecté ». CIFRE Renault. 2017-2020.

[Th3] C. Bertero. « BICycle-based Laboratory of Urban Evolutions ». Financement Univ. Toulouse. 2016-2019.

[Th4] R. Pasqua. « Inférence et modèles de données personnelles : mobilité sociale, proximité spatiale ». Financement Ministériel. Thèse de doct. Nov. 2016. URL : `http://thesesups.ups-tlse.fr/3339/1/2016TOU30195.pdf`.

[Th5] L. Pintard. « From safety analysis to experimental validation by fault injection - Case of automotive embedded systems ». CIFRE Valeo. Thèse de doct. 2015. URL : `http://www.theses.fr/2015INPT0052/document`.

[Th6] M. Traore. « Privacy-preserving and secure location authentication ». ANR AMORES. Thèse de doct. 2015. URL : `http://www.theses.fr/2015INPT0053/document`.

[Th7] O. Baldellon. « Supervision en ligne de propriétés temporelles dans les systèmes distribués temps-réel ». Financement Ministériel. Thèse de doct. 2014. URL : `http://www.theses.fr/2014INPT0098/document`.

[Th8] H. Martorell. « Architecture et processus de développement permettant la mise à jour dynamique de systèmes embarqués automobiles ». CIFRE Renault. Thèse de doct. 2014. URL : `http://www.theses.fr/2014INPT0108/document`.

[Th9] M. Stoicescu. « Architecting Resilient Computing Systems : a Component-Based Approach ». ANR MURPHY. Thèse de doct. 2013. URL : `http://www.theses.fr/2013INPT0120/document`.

[Th10] T. Robert. « Détection d'erreur au plus tôt dans les systèmes temps réel : une approche basée sur la vérification en ligne ». Financement Ministériel. Thèse de doct. Juin 2009. URL : `https://tel.archives-ouvertes.fr/tel-00420480`.

Postdoc encadrés

[P1] F. Dufossé. *Théorie des jeux et model checking pour la vérification de propriétés de sûreté*. Financement RTRA STAE, avec Jérémie Guiochet (LAAS). 2013.

[P2] A. Kritikakou. *Exécution fiable de systèmes à multiple niveaux de criticité sur multi-cœurs*. Financement RTRA STAE, avec Claire Pagetti (ONERA). 2013.

Publications — journaux

[J1] A. Castañeda, S. Rajsbaum et M. Roy. « Convergence and covering on graphs for wait-free robots ». In : *Journal of the Brazilian Computer Society* 24.1 (jan. 2018). url : https://hal.laas.fr/hal-01740338.

[J2] A. Kritikakou, T. Marty et M. Roy. « DYNASCORE : DYNAmic Software COntroller to increase REsource utilization in mixed-critical systems ». In : *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 23 (2) (jan. 2018). url : https://hal.archives-ouvertes.fr/hal-01559696.

[J0] A. Mostefaoui, M. Raynal et M. Roy. « Time-Efficient Read/Write Register in Crash-prone Asynchronous Message-Passing Systems ». In : *Computing* (mai 2018). url : https://hal.archives-ouvertes.fr/hal-01784210.

[J3] M. Lauer, M. Amy, J.-C. Fabre, M. Roy, W. Excoffon et M. Stoicescu. « Resilient Computing on ROS using Adaptive Fault Tolerance ». In : *Journal of Software : Evolution and Process* 00 (nov. 2017), p. 1-18. url : https://hal.laas.fr/hal-01703968.

[J4] Y. A. Pignolet, M. Roy, S. Schmid et G. Trédan. « The many faces of graph dynamics ». In : *Journal of Statistical Mechanics : Theory and Experiment* 2017 (juin 2017). url : https://hal.archives-ouvertes.fr/hal-01559708.

[J5] M. Stoicescu, J.-C. Fabre et M. Roy. « Architecting resilient computing systems : A component-based approach for adaptive fault tolerance ». In : *Journal of Systems Architecture* 73 (fév. 2017), pp.6-16. url : https://hal.archives-ouvertes.fr/hal-01472877.

[J6] M.-O. Killijian, R. Pasqua, M. Roy, G. Trédan et C. Zanon. « Souk : Spatial Observation of hUman Kinetics ». In : *Computer Networks* (2016). url : https://hal.archives-ouvertes.fr/hal-01372329.

[J7] M. Machin, J. Guiochet, H. Waeselynck, J.-P. Blanquart, M. Roy et L. Masson. « SMOF - A Safety MOnitoring Framework for Autonomous Systems ». In : *IEEE Transactions on Systems, Man, and Cybernetics : Systems* (déc. 2016). url : https://hal.archives-ouvertes.fr/hal-01394139.

[J8] Y. Deswarte, C. A. Melchor, V. Nicomette et M. Roy. « Protection de la vie privée sur internet ». In : *Revue de l'Electricité et de l'Electronique (REE)* 9 (2006), p. 65-74.

[J9] A. Mostéfaoui, S. Rajsbaum, M. Raynal et M. Roy. « Condition-based consensus solvability : a hierarchy of conditions and efficient protocols ». In : *Distributed Computing* 17.1 (2004), p. 1-20. url : https://doi.org/10.1007/s00446-003-0093-9.

Publications — chapitres

[Ch1] M. Roy. « The BG Distributed Simulation Algorithm ». In : *Encyclopedia of Algorithms*. Sous la dir. de M.-Y. Kao. Springer, 2016. Chap. BG Distributed Simulation Algorithm, p. 199-203. URL : https://doi.org/10.1007/978-3-642-27848-8_611-1.

[Ch2] I. Astic, C. Aunis, J. Dupire, V. Gal, E. Gressier-Soudan, C. Pitrey, M. Roy, F. Sailhan, M. Simatic, A. Topol et E. Zaza. « Pervasive Games and Critical Applications ». In : *Computer Science and Ambient Intelligence*. Wiley-Blackwell, 2013. Chap. 12, p. 263-284. ISBN : 9781118580974. URL : https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118580974.ch12.

[Ch3] A. A. E. Kalam, C. A. Melchor, S. Berthold, J. Camenisch, S. Clauss, Y. Deswarte, M. Kohlweiss, A. Panchenko, L. Pimenidis et M. Roy. « Further Privacy Mechanisms ». In : *Digital Privacy - PRIME - Privacy and Identity Management for Europe*. Sous la dir. de J. Camenisch, R. Leenes et D. Sommer. T. 6545. Lecture Notes in Computer Science. Springer, 2011, p. 485-555.

[Ch4] M.-O. Killijian et M. Roy. « Architecting Dependable Systems VII ». In : sous la dir. d'A. Casimiro, R. de Lemos et C. Gacek. T. 6420. Lecture Notes in Computer Science. Springer, 2010. Chap. Data Backup for Mobile Nodes : a Cooperative Middleware and Experimentation Platform, p. 53-73. ISBN : 978-3-642-17244-1. URL : https://doi.org/10.1007/978-3-642-17245-8.

Publications — conférences

[C1] C. Bertero, M. Roy, C. Sauvanaud et G. Trédan. « Experience Report : Log Mining using Natural Language Processing and Application to Anomaly Detection ». In : *28th International Symposium on Software Reliability Engineering (ISSRE 2017)*. Toulouse, France, oct. 2017, 10p. URL : https://hal.laas.fr/hal-01576291.

[C2] A. Castañeda, Y. Moses, M. Raynal et M. Roy. « Early Decision and Stopping in Synchronous Consensus : A Predicate-Based Guided Tour ». In : *International Conference on Networked Systems (NETYS)*. T. 96. Proceedings of NETYS 2017 : the 5th International Conference on Networked Systems. Marrakech, Morocco, mai 2017, pp.167-221. URL : https://hal.archives-ouvertes.fr/hal-01559723.

[C3] A. Castañeda, P. Fraigniaud, E. Gafni, S. Rajsbaum et M. Roy. « Asynchronous Coordination Under Preferences and Constraints ». In : *23rd International Colloquium on Structural Information and Communication Complexity*. SIROCCO 2016 pre-proceedings. Helsinki, Finland, juil. 2016. URL : https://hal.archives-ouvertes.fr/hal-01341710.

[C4] A. Castañeda, P. Fraigniaud, E. Gafni, S. Rajsbaum et M. Roy. « Asynchronous Coordination with Constraints and Preferences ». In : *35th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2016)*. Proceedings of the 35th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2016). Chicago, United States, juil. 2016. URL : https://hal.archives-ouvertes.fr/hal-01326911.

[C5] A. Castañeda, S. Rajsbaum et M. Roy. « Two Convergence Problems for Robots on Graphs ». In : *7th IEEE Latin-American Symposium on Dependable Computing*. LADC 2016. Cali, Colombia, oct. 2016. URL : https://hal.archives-ouvertes.fr/hal-01386628.

[C6] J.-C. Fabre, M. Lauer, M. Roy, M. Amy, W. Excoffon et M. Stoicescu. « Towards Resilient Computing on ROS for Embedded Applications ». In : *8th European Congress on Embedded real Time Software and Systems (ERTS 2016)*. Toulouse, France, jan. 2016. URL : https://hal.archives-ouvertes.fr/hal-01288113.

[C7] A. Kritikakou, T. Marty, C. Pagetti, C. Rochange, M. Lauer et M. Roy. « Multiplexing Adaptive with Classic AUTOSAR ? Adaptive Software Control to Increase Resource Utilization in Mixed-Critical Systems ». In : *Workshop CARS 2016 - Critical Automotive applications : Robustness & Safety*. CARS 2016 - Critical Automotive applications : Robustness & Safety. Göteborg, Sweden, sept. 2016. URL : https://hal.archives-ouvertes.fr/hal-01375576.

[C8] M. Lauer, M. Amy, J.-C. Fabre, M. Roy, W. Excoffon et M. Stoicescu. « Engineering Adaptive Fault-Tolerance Mechanisms for Resilient Computing on ROS ». In : *HASE 2016 - IEEE 17th International Symposium on High Assurance Systems Engineering Symposium*. HASE 2016 - IEEE High Assurance Systems Engineering Symposium. Orlando, FL, United States : IEEE, jan. 2016, p. 94-101. URL : https://hal.archives-ouvertes.fr/hal-01288098.

[C9] R. Pasqua, M. Roy et G. Trédan. « Loca : A Location-Oblivious Co-location Attack in Crowds ». In : *2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. Ubicomp 2016. Heidelberg, Germany : ACM, sept. 2016. URL : https://hal.archives-ouvertes.fr/hal-01372317.

[C10] M. Lauer, M. Amy, W. Excoffon, M. Roy et M. Stoicescu. « Towards Adaptive Fault Tolerance : From a Component-Based Approach to ROS ». In : *CARS 2015 - Critical Automotive applications : Robustness & Safety*. Paris, France, sept. 2015. URL : https://hal.archives-ouvertes.fr/hal-01193039.

[C11] M. Machin, F. Dufossé, J. Guiochet, D. Powell, M. Roy et H. Waeselynck. « Model-Checking and Game theory for Synthesis of Safety Rules ». In : *2015 IEEE 16th International Symposium on High Assurance Systems Engineering (HASE)*. Daytona Beach Shores, United States, jan. 2015, pp.36-43. URL : https://hal.archives-ouvertes.fr/hal-01164981.

[C12] H. Martorell, J.-C. Fabre, M. Lauer, M. Roy et R. Valentin. « Partial Updates of AUTOSAR Embedded Applications - To What Extent ? » In : *11th European Dependable Computing Conference (EDCC 2015)*. Paris, France, sept. 2015. URL : https://hal.archives-ouvertes.fr/hal-01194832.

[C13] L. Pintard, M. Leeman, A. Ymlahi-Ouazzani, J.-C. Fabre, K. Kanoun et M. Roy. « Using Fault Injection to Verify an AUTOSAR Application According to the ISO 26262 ». In : *SAE 2015 World Congress & Exhibition*. Journal Articles from SAE 2015 World Congress & Exhibition. Detroit, United States : SAE International, avr. 2015. URL : https://hal.archives-ouvertes.fr/hal-01221422.

[C14] P. Fraigniaud, E. Gafni, S. Rajsbaum et M. Roy. « Automatically Adjusting Concurrency to the Level of Synchrony ». In : *International Symposium on Distributed Computing (DISC)*. T. LNCS. 8784. Austin, United States, oct. 2014, p. 1-15. URL : https://hal.archives-ouvertes.fr/hal-01062031.

[C15] P. Fraigniaud, S. Rajsbaum, M. Roy et C. Travers. « The Opinion Number of Set-Agreement ». In : *OPODIS 2014 - 18th International Conference on Principles of Distributed Systems*. Sous la dir. de M. K. Aguilera, L. Querzoni et M. Shapiro. T. 8878. LNCS - Lecture Notes in Computer Science. Cortina d'Ampezzo, Italy : Springer, déc. 2014, p. 155-170. URL : https://hal.inria.fr/hal-01073578.

[C16] J. Friginal, M.-O. Killijian, R. Pasqua, M. Roy et G. Trédan. « Does Mobility Matter ? An Evaluation Methodology for Opportunistic Apps ». In : *IEEE International Symposium on Network Computing and Applications (NCA)*. Cambridge, United States, août 2014. URL : https://hal.archives-ouvertes.fr/hal-01242260.

[C17] S. Gambs, M.-O. Killijian, C. Lauradoux, C. Onete, M. Roy et M. Traoré. « Vssdb : A Verifiable Secret-Sharing Distance-Bounding Protocol ». In : *International Conference on Cryptography and Information security (BalkanCryptSec'14)*. BalkanCryptSec 2014. Istanbul, France, oct. 2014. URL : https://hal.archives-ouvertes.fr/hal-01242265.

[C18] S. Gambs, M.-O. Killijian, M. Roy et M. Traoré. « PROPS : A PRivacy-Preserving Location Proof System ». In : *33rd IEEE International Symposium on Reliable Distributed Systems, SRDS 2014*. 33rd IEEE International Symposium on Reliable Distributed Systems, SRDS 2014. Nara, Japan, oct. 2014. URL : https://hal.archives-ouvertes.fr/hal-01242266.

[C19] A. Kritikakou, O. Baldellon, C. Pagetti, C. Rochange et M. Roy. « Run-time Control to Increase Task Parallelism in Mixed-Critical Systems ». In : *26th Euromicro Conference on Real-Time Systems (ECRTS14)*. Madrid, Spain, juil. 2014, 11p. URL : https://hal.archives-ouvertes.fr/hal-01015476.

[C20] A. Kritikakou, C. Pagetti, M. Roy, C. Rochange, M. Faugère, S. Girbal et D. Gracia Pérez. « Distributed run-time WCET controller for concurrent critical tasks in mixed-critical systems ». In : *22nd International Conference on Real-Time Networks and Systems*. Versailles, France, oct. 2014. URL : https://hal.archives-ouvertes.fr/hal-01096102.

[C21] H. Martorell, J.-C. Fabre, M. Roy et R. Valentin. « Improving Adaptiveness of AUTOSAR Embedded Applications ». In : *ACM Symposium on Applied Computing*. T. Volume I : Artificial intelligence & agents, distributed systems, and information systems : dependable. Gyeongju, South Korea, mar. 2014, p. 384-390. URL : https://hal.archives-ouvertes.fr/hal-01062054.

[C22] L. PINTARD, J.-C. Fabre, M. Leeman, K. Kanoun et M. Roy. « From Safety Analyses to Experimental Validation of Automotive Embedded Systems ». In : *PRDC 2014*. 20th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2014). Singapore, Singapore : IEEE, nov. 2014, p. 125-134. URL : https://hal.archives-ouvertes.fr/hal-01265454.

[C23] M. Roy, S. Schmid et G. Trédan. « Modeling and Measuring Graph Similarity : The Case for Centrality Distance ». In : *FOMC 2014, 10th ACM International Workshop on Foundations of Mobile Computing*. Philadelphia, United States, août 2014, p. 53. URL : https://hal.archives-ouvertes.fr/hal-01010901.

[C24] M. Stoicescu, J.-C. Fabre, M. Roy et A. Pathak. « From Resilient Computing Architectural Concepts to Wireless Sensor Network-Based Applications ». In : *European Dependable Computing Conference (EDCC)*. 2014, p. 46-49. URL : https://hal.archives-ouvertes.fr/hal-00938389.

[C25] O. Baldellon, J.-C. Fabre et M. Roy. « Minotor : Monitoring Timing and Behavioral Properties for Dependable Distributed Systems ». In : *The 19th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2013)*. Vancouver, Canada, déc. 2013, 10p. URL : https://hal.archives-ouvertes.fr/hal-00978475.

[C26] Q. Enard, M. Stoicescu, E. Balland, C. Consel, L. Duchien, J.-C. Fabre et M. Roy. « Design-Driven Development Methodology for Resilient Computing ». In : *CBSE'13 : Proceedings of the 16th International ACM Sigsoft Symposium on Component-Based Software Engineering*. Vancouver, Canada, juin 2013. URL : https://hal.inria.fr/hal-00814298.

[C27] M.-O. Killijian, M. Roy, G. Trédan et C. Zanon. « SOUK : Social Observation of hUman Kinetics ». In : *Ubicomp 2013 (2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing)*. Zurich, Switzerland, sept. 2013, 4p. URL : https://hal.archives-ouvertes.fr/hal-00839520.

[C28] A. Kritikakou, O. Baldellon, C. Pagetti, C. Rochange, M. Roy et F. Vargas. « Monitoring On-line Timing Information to Support Mixed-Critical Workloads ». In : *IEEE Real-Time Systems Symposium 2013*. Vancouver, Canada, déc. 2013, p. 9-10. URL : https://hal.archives-ouvertes.fr/hal-01015455.

[C29] H. Martorell, J.-C. Fabre, M. Roy et R. Valentin. « Towards Dynamic Updates In AUTOSAR ». In : *SAFECOMP 2013 - Workshop CARS (2nd Workshop on Critical Automotive applications : Robustness & Safety) of the 32nd International Conference on Computer Safety, Reliability and Security*. Sous la dir. de M. ROY. Toulouse, France, sept. 2013, NA. URL : https://hal.archives-ouvertes.fr/hal-00848361.

[C30] L. Pintard, J.-C. Fabre, K. Kanoun, M. Leeman et M. Roy. « Fault Injection in the Automotive Standard ISO 26262 : An Initial Approach ». In : *European Workshop on Dependable Computing (EWDC)*. 2013.

[C31] O. Baldellon, J.-C. Fabre et M. Roy. « Distributed Monitoring of Temporal System Properties using Petri Nets ». In : *31st IEEE International Symposium on Reliable Distributed Systems (SRDS 2012)*. Irvine, United States, oct. 2012, 10p. URL : https://hal.archives-ouvertes.fr/hal-01015494.

[C32] S. Gambs, M.-O. Killijian, M. Roy et M. Traoré. « Locanyms : Towards Privacy-Preserving Location-Based Services ». In : *1st European Workshop on AppRoaches to MObiquiTous Resilience*. Sibiu, Romania, mai 2012, p. 6. URL : https://hal.archives-ouvertes.fr/hal-00699742.

[C33] A. Mekki-Mokhtar, J.-P. Blanquart, J. Guiochet, D. Powell et M. Roy. « Elicitation of Executable Safety Rules for Critical Autonomous Systems ». In : *Embedded Real Time Software and Systems (ERTS2012)*. Toulouse, France, fév. 2012, 10p. URL : https://hal.archives-ouvertes.fr/hal-01282237.

[C34] A. Mekki-Mokhtar, J.-P. Blanquart, J. Guiochet, D. Powell et M. Roy. « Safety Trigger Conditions for Critical Autonomous Systems ». In : *The 18th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2012)*. Niigata, Japan, nov. 2012, 10p. URL : https://hal.archives-ouvertes.fr/hal-01282203.

[C35] M. Stoicescu, J.-C. Fabre et M. Roy. « From Design for Adaptation to Component-Based Resilient Computing ». In : *IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2012)*. 10p. Niigata, Japan, nov. 2012, 10p. URL : https://hal.archives-ouvertes.fr/hal-00747469.

[C36] M.-O. Killijian, M. Roy et G. Severac. « The ARUM Experimentation Platform : an Open Tool to evaluate Mobile Systems Applications ». In : *AMiRE 2011 : 6th International Symposium on Autonomous Minirobots for Research and Edutainment*. Bielefeld, Germany, mai 2011, p. 1. URL : https://hal.archives-ouvertes.fr/hal-00667832.

[C37] M. Stoicescu, J.-C. Fabre et M. Roy. « Architecting Resilient Computing Systems : Overall Approach and Open Issues ». In : *Software Engineering for Resilient Systems - Third International Workshop, SERENE*. T. LNCS 6968. Sept. 2011. URL : https://hal.archives-ouvertes.fr/hal-01615018.

[C38] M.-O. Killijian, M. Roy et G. Séverac. « ARUM : A cooperative middleware and an experimentation platform for mobile systems ». In : *IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*. Oct. 2010, p. 442-449. URL : https://hal.archives-ouvertes.fr/hal-01615017.

[C39] M.-O. Killijian, M. Roy et G. Tredan. « Beyond San Fancisco Cabs : Building a *-lity Mining Dataset for Social Traces Analysis ». In : *Workshop on the Analysis of Mobile Phone Networks*. Cambridge, MA, United States, mai 2010. URL : https://hal.laas.fr/hal-01740343.

[C40] T. Robert, M. Roy et J.-C. Fabre. « Early Error Detection for Fault Tolerance Strategies ». In : *18th International Conference on Real-Time and Network Systems*. Toulouse, France, nov. 2010, p. 159-168. URL : https://hal.archives-ouvertes.fr/hal-00546934.

[C41] M.-O. Killijian et M. Roy. « Brief announcement : A Platform for Experimenting with Mobile Algorithms in a Laboratory ». In : *Principles Of Distributed Computing*. Calgary, Canada, août 2009, p. 316-317. URL : https://hal.archives-ouvertes.fr/hal-00394002.

[C42] T. Robert, J.-C. Fabre et M. Roy. « Application of Early Error Detection for Handling Degraded Modes of Operation ». In : *12th European Workshop on Dependable Computing, EWDC 2009*. Sous la dir. de H. WAESELYNCK. Toulouse, France, mai 2009, 3 pages. URL : https://hal.archives-ouvertes.fr/hal-00381913.

[C43] T. Robert, J. Fabre et M. Roy. « On-line Monitoring of Real Time Applications for Early Error Detection ». In : *14th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2008, 15-17 December 2008, Taipei, Taiwan*. 2008, p. 24-31. URL : https://doi.org/10.1109/PRDC.2008.31.

[C44] M. Roy, F. Bonnet, L. Querzoni, S. Bonomi, M. Killijian et D. Powell. « Georegisters : An Abstraction for Spatial-Based Distributed Computing ». In : *Principles of Distributed Systems, 12th International Conference, OPODIS 2008, Luxor, Egypt, December 15-18, 2008. Proceedings*. 2008, p. 534-537. URL : https://hal.archives-ouvertes.fr/hal-01741220.

[C45] E. Anceaume, X. Défago, M. Gradinariu et M. Roy. « Towards a Theory of Self-organization ». In : *Principles of Distributed Systems, 9th International Conference, OPODIS 2005, Pisa, Italy, December 12-14, 2005, Revised Selected Papers*. 2005, p. 191-205. URL : https://doi.org/10.1007/11795490_16.

[C46] E. Anceaume, X. Défago, M. Gradinariu et M. Roy. « Towards a Theory of Self-organization ». In : *Distributed Computing, 19th International Conference, DISC 2005, Cracow, Poland, September 26-29, 2005, Proceedings*. 2005, p. 505-506. URL : https://doi.org/10.1007/11561927_43.

[C47] L. Courtès, M. Killijian, D. Powell et M. Roy. « Sauvegarde coopérative entre pairs pour dispositifs mobiles ». In : *Actes des 2èmes journées francophones Mobilité et Ubiquité 2005, UBIMOB'05, 31 mai - 3 juin 2005, Grenoble, France*. 2005, p. 97-104. URL : http://doi.acm.org/10.1145/1102613.1102635.

[C48] M. Raynal et M. Roy. « Allowing Atomic Objects to Coexist with Sequentially Consistent Objects ». In : *Parallel Computing Technologies, 8th International Conference, PaCT 2005, Krasnoyarsk, Russia, September 5-9, 2005, Proceedings*. 2005, p. 59-73. URL : https://doi.org/10.1007/11535294_6.

[C49] M. Raynal, M. Roy et C. Tutu. « A Simple Protocol Offering Both Atomic Consistent Read Operations and Sequentially Consistent Read Operations ». In : *19th International Conference on Advanced Information Networking and Applications (AINA 2005), 28-30 March 2005, Taipei, Taiwan*. 2005, p. 961-966. URL : https://doi.org/10.1109/AINA.2005.64.

[C50] E. Anceaume, R. Friedman, M. Gradinariu et M. Roy. « An Architecture for Dynamic Scalable Self-Managed Persistent Objects ». In : *On the Move to Meaningful Internet Systems 2004 : CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, Agia Napa, Cyprus, October 25-29, 2004, Proceedings, Part II*. 2004, p. 1445-1462. URL : https://doi.org/10.1007/978-3-540-30469-2_39.

[C51] A. Mostéfaoui, S. Rajsbaum, M. Raynal et M. Roy. « A Hierarchy of Conditions for Asynchronous Interactive Consistency ». In : *Parallel Computing Technologies, 7th International Conference, PaCT 2003, Novosibirsk, Russia, September 15-19, 2003, Proceedings*. 2003, p. 130-140. URL : https://doi.org/10.1007/978-3-540-45145-7_11.

[C52] M. Roy et A. Mostéfaoui. « Single-Write Safe Consensus using Constrained Inputs ». In : *SIROCCO 10 : Proceedings of the 10th Internaltional Colloquium on Structural Information Complexity, June 18-20, 2003, Umeå Sweden*. 2003, p. 291-306.

[C53] D. Agrawal, A. El Abbadi, A. Mostéfaoui, M. Raynal et M. Roy. « The Lord of the Rings : Efficient Maintenance of Views at Data Warehouses ». In : *Distributed Computing, 16th International Conference, DISC 2002, Toulouse, France, October 28-30, 2002 Proceedings*. 2002, p. 33-47. URL : https://doi.org/10.1007/3-540-36108-1_3.

[C54] A. Mostéfaoui, S. Rajsbaum, M. Raynal et M. Roy. « Condition-Based Protocols for Set Agreement Problems ». In : *Distributed Computing, 16th International Conference, DISC 2002, Toulouse, France, October 28-30, 2002 Proceedings*. 2002, p. 48-62. URL : https://doi.org/10.1007/3-540-36108-1_4.

[C55] A. Mostéfaoui, M. Raynal, M. Roy, D. Agrawal et A. El Abbadi. « Towards a formal model for view maintenance in data warehouses ». In : *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24, 2002*. 2002, p. 129. URL : http://doi.acm.org/10.1145/571825.571845.

[C56] A. Mostéfaoui, S. Rajsbaum, M. Raynal et M. Roy. « A hierarchy of conditions for consensus solvability ». In : *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, PODC 2001, Newport, Rhode Island, USA, August 26-29, 2001*. 2001, p. 151-160. URL : http://doi.acm.org/10.1145/383962.384006.

[C57] A. Mostéfaoui, S. Rajsbaum, M. Raynal et M. Roy. « Efficient Condition-Based Consensus ». In : *SIROCCO 8, Proceedings of the 8th International Colloquium on Structural Information and Communication Complexity, Vall de Núria, Girona-Barcelona, Catalonia, Spain, 27-29 June, 2001*. 2001, p. 275-292.

Projets

[Pr1] *CIFRE Continental : Équilibrage de charge efficace et adaptatif avec garanties spatio-temporelles pour les systèmes de véhicules connectés.* Co-responsable LAAS, direction de la thèse de Jean Ibarz, 2018-2021.

[Pr2] *CIFRE Renault : Architecture Électrique/Électronique (EE) pour le véhicule du futur autonome et connecté.* Responsable LAAS, direction de la thèse de Daniel Loche, 2017-2020.

[Pr3] *RTRA STAE SYSCOB : Systèmes Complexes Bio-inspirés.* Co-direction du projet, 2017-2019. URL : `http://xsys.fr/evenements/wisdrones/`.

[Pr4] *CLUE : Cycle-based Laboratory of Urban Evolutions.* Co-direction du projet, 2016-. URL : `https://www.laas.fr/projects/clue`.

[Pr5] *H2020 CPSELabs : CPS Engineering Labs - expediting and accelerating the realization of cyber-physical systems.* Participant, 2015-2018. URL : `http://www.cpse-labs.eu`.

[Pr6] *ANR AMORES : Architecture for Mobiquitous Resilient Systems.* Responsable WP algorithmes répartis, 2012-2015. URL : `http://projects.laas.fr/AMORES/`.

[Pr7] *CIFRE Renault : Architectures et processus de développement permettant la mise à jour dynamique de systèmes embarqués automobiles.* Co-responsable LAAS, co-direction de la thèse d'Hélène Martorell, 2012-2015.

[Pr8] *CIFRE Valeo : Méthodes et techniques d'injection de fautes pour une couverture optimale des exigences de sûreté des applications automobiles embarquées.* Co-responsable LAAS, co-direction de la thèse de Ludovic Pintard, 2012-2015.

[Pr9] *CNRS/UNAM DACOR : Distributed Algorithms : Complexity evaluation and On-line veRification and checking using algebraic topology tools.* Responsable du projet, 2012-2015.

[Pr10] *SMOF : Safety Monitoring Framework.* Participant, 2012-. URL : `https://www.laas.fr/projects/smof`.

[Pr11] *SOUK : Social Observation of Human Kinetics.* Co-direction du projet, 2012-. URL : `http://projects.laas.fr/souk`.

[Pr12] *ANR BLANC Murphy : Dependability-focused Evaluation of Sensor Networks.* Responsable LAAS, 2011-2013. URL : `http://murphy.cnam.fr`.

[Pr13] *INRIA ARC Serus : Software Engineering for Resilient Ubiquitous Systems.* Participant, 2011-2012. URL : `http://serus.bordeaux.inria.fr`.

[Pr14] *RTRA STAE TORRENTS : Time-oriented reliable embedded networked systems.* Co-direction du projet, 2010-2016. URL : `http://www.irit.fr/torrents/`.

[Pr15] *EU FP6 HIDENETS : Highly dependable ip-based Networks and services.* Responsable plateforme mobilité et algorithmes répartis, 2006-2008. URL : `http://cordis.europa.eu/project/rcn/79303_en.html`.

[Pr16] *EU NoE Resist : Resilience for survivability in IST.* Responsable projet systèmes mobiles, 2006-2009. URL : `http://www.resist-noe.org`.

[Pr17] *EU FP6 ASSERT : Automated proof-based System and Software Engineering for Real-Time applications*. Responsable LAAS, 2004-2007. URL : `https://cordis.europa.eu/project/rcn/71564_en.html`.

[Pr18] *EU FP6 PRIME : Privacy and Identity Management for Europe*. Responsable intégration serveur et contrôle d'accès, 2004-2008. URL : `http://doi.org/10.1007/978-3-642-19050-6`.

Références

[1] J.-C. Laprie. « From dependability to resilience ». In : *38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks*. 2008, G8-G9.

[2] A. M. Turing. « On computable numbers, with an application to the Entscheidungsproblem ». In : *Proceedings of the London Mathematical Society* (1936).

[3] K. Gödel. « Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I ». In : *Monatshefte für Mathematik und Physik* 38 (1931), p. 173-198.

[4] A. Church. « An Unsolvable Problem of Elementary Number Theory ». In : *American Journal of Mathematics* 58.2 (1936), p. 345-363.

[5] H. Minkowski. « Espace et Temps ». In : *Annales scientifiques de l'É.N.S.* T. 26. 3è série. 1909, p. 499-517.

[6] J.-C. Laprie. « Sûreté de fonctionnement des systèmes : concepts de base et terminologie ». In : *REE. Revue de l'électricité et de l'électronique* 11 (2004), p. 95-105.

[7] K. Kim et T. F. Lawrence. « Adaptive fault tolerance : Issues and approaches ». In : *Distributed Computing Systems, 1990. Proceedings., Second IEEE Workshop on Future Trends of*. IEEE. 1990, p. 38-46.

[8] L. C. Lung, F. Favarim, G. T. Santos et M. Correia. « An Infrastructure for Adaptive Fault Tolerance on FT-CORBA ». In : *Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC '06)*. IEEE, 2006.

[9] O. Marin, P. Sens, J.-P. Briot et Z. Guessoum. « Towards Adaptive Fault-Tolerance for Distributed Multi-Agent Systems ». In : *Proceedings of The Fourth European Research Seminar on Advances in Distributed Systems*. ERSADS '01. 2001, p. 195-201.

[10] J. Marino et M. Rowley. *Understanding SCA (Service Component Architecture)*. Addison-Wesley Professional, 2009.

[11] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier et J. Irwin. « Aspect-Oriented Programming ». In : *ECOOP'97 – Proceedings of the European Conference on Object-Oriented Programming* (1997), p. 220-242.

[12] C. Szyperski. *Component Software : Beyond Object-Oriented Programming*. 2nd. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2002. isbn : 0201745720.

[13] E. Bini, M. D. Natale et G. Buttazzo. « Sensitivity analysis for fixed-priority real-time systems ». In : *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*. 2006, p. 10-22.

[14] E. Asarin, P. Caspi et O. Maler. « Timed regular expressions ». In : *Journal of the ACM* 49 (2002), p. 172-206.

[15] A. Bauer, M. Leucker et C. Schallhart. « Monitoring of Real-Time Properties ». In : *FSTTCS*. Sous la dir. de S. Arun-Kumar et N. Garg. T. 4337. Springer, 2006, p. 260-272.

[16] R. Alur. « Timed Automata ». In : *Proc. 11th International Computer Aided Verification Conference*. 1999, p. 8-22.

[17] S. Tripakis et S. Yovine. « Analysis of Timed Systems Using Time-Abstracting Bisimulations ». In : *Formal Methods in System Design* 18 (2001), p. 25-68.

[18] M. Herlihy. « Wait-free Synchronization ». In : *ACM Trans. Program. Lang. Syst.* 13.1 (jan. 1991), p. 124-149. issn : 0164-0925. url : `http://doi.acm.org/10.1145/114005.102808`.

[19] M. P. Herlihy et J. M. Wing. « Linerizability : A Correctness Condition for Atomic Objects ». In : *TOPLAS* 12.3 (juil. 1990), p. 463-492.

[20] E. Borowsky et E. Gafni. « Generalized FLP Impossibility Result for T-resilient Asynchronous Computations ». In : *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*. STOC '93. San Diego, California, USA : ACM, 1993, p. 91-100. isbn : 0-89791-591-7.

[21] E. Borowsky, E. Gafni, N. Lynch et S. Rajsbaum. « The BG distributed simulation algorithm ». In : *Distributed Computing* 14.3 (juil. 2001), p. 127-146.

[22] F. B. Schneider. « Implementing Fault-Tolerant Services Using the State Machine Approach : A Tutorial ». In : *ACM Computing Surveys (CSUR)* 22.4 (1990). `http://www.cs.cornell.edu/fbs/publications/smsurvey.pdf`, p. 299-319.

[23] L. Lamport. « The Part-time Parliament ». In : *ACM Trans. Comput. Syst.* 16.2 (mai 1998), p. 133-169. issn : 0734-2071.

[24] M. Chérèque, D. Powell, P. Reynier, J.-L. Richier et J. Voiron. « Active Replication in Delta-4 ». In : *The Twenty-Second Annual International Symposium on Fault-Tolerant Computing (Digest of Papers : FTCS-22)*. IEEE, 1992, p. 28-37.

[25] M. J. Fischer, N. A. Lynch et M. S. Paterson. « Impossibility of Distributed Consensus with one Faulty Process ». In : *Journal of the ACM* 32.2 (avr. 1985), p. 374-382.

[26] H. Attiya, A. Bar-Noy et D. Dolev. « Sharing Memory Robustly in Message-passing Systems ». In : *J. ACM* 42.1 (1995), p. 124-142. issn : 0004-5411.

[27] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt et N. Shavit. « Atomic Snapshots of Shared Memory ». In : *J. ACM* 40.4 (1993), p. 873-890. issn : 0004-5411.

[28] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg et R. Reischuk. « Renaming in an asynchronous environment ». In : *J. ACM* 37.3 (1990), p. 524-548.

[29] L. Lamport. « On Interprocess Communication. Part I & II ». In : *Distributed Computing* 1.2 (1986), p. 77-101.

[30] A. Klappenecker, H. Lee et J. L. Welch. « Dynamic regular registers in systems with churn ». In : *Theoretical Computer Science* 512 (2013). Stabilization, Safety and Security, p. 84-97. issn : 0304-3975.

[31] R. Baldoni, S. Bonomi, A. M. Kermarrec et M. Raynal. « Implementing a Register in a Dynamic Distributed System ». In : *2009 29th IEEE International Conference on Distributed Computing Systems*. Juin 2009, p. 639-647.

[32] I. Bilogrevic, K. Huguenin, M. Jadliwala, F. Lopez, J.-P. Hubaux, P. Ginzboorg et V. Niemi. « Inferring social ties in academic networks using short-range wireless communications ». In : *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM. 2013, p. 179-188.

[33] P. Erdős et A. Rényi. « On the evolution of random graphs ». In : *Publ. Math. Inst. Hungar. Acad. Sci* 5 (1960), p. 17-61.

[34] M. E. Newman, D. J. Watts et S. H. Strogatz. « Random graph models of social networks ». In : *Proceedings of the National Academy of Sciences* 99 (2002), p. 2566-2572.

[35] A.-L. Barabási et R. Albert. « Emergence of Scaling in Random Networks ». In : *Science* 286 (1999).

[36] X. Gao, B. Xiao, D. Tao et X. Li. « A survey of graph edit distance ». In : *Pattern Anal. Appl.* 13.1 (2010), p. 113-129.

[37] S. Borgatti et M. Everett. « A graph-theoretic perspective on centrality ». In : *Social Networks* 28.4 (2006), p. 466-484.

[38] M. Newman et M. Girvan. « Finding and evaluating community structure in networks ». In : *Phys. Rev. E.* 69.2 (2004).

[39] M. Bambagini, M. Marinoni, H. Aydin et G. Buttazzo. « Energy-Aware Scheduling for Real-Time Systems : A Survey ». In : *ACM Trans. Embed. Comput. Syst.* 15.1 (jan. 2016), 7 :1-7 :34. issn : 1539-9087.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville et Y. Bengio. « Generative Adversarial Nets ». In : *Advances in Neural Information Processing Systems 27*. Sous la dir. de Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence et K. Q. Weinberger. Curran Associates, Inc., 2014, p. 2672-2680.

[41] M. Herlihy et N. Shavit. « The Topological Structure of Asynchronous Computability ». In : *J. ACM* 46.6 (nov. 1999), p. 858-923. issn : 0004-5411.

[42] M. Herlihy, D. Kozlov et S. Rajsbaum. *Distributed Computing Through Combinatorial Topology*. 1st. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2013.

[43] D. Peleg. *Distributed Computing : A Locality-sensitive Approach*. Philadelphia, PA, USA : Society for Industrial et Applied Mathematics, 2000. isbn : 0-89871-464-8.

[44] Y. Afek et E. Gafni. « A Simple Characterization of Asynchronous Computations ». In : *Theor. Comput. Sci.* 561.PB (jan. 2015), p. 88-95. issn : 0304-3975.

[45] « Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings ». In : sous la dir. d'A. F. Anta, G. Lipari et M. Roy. T. 7109. Lecture Notes in Computer Science. Springer, 2011. isbn : 978-3-642-25872-5.

[46] A. Avižienis, J.-C. Laprie, B. Randell et C. Landwehr. « Basic Concepts and Taxonomy of Dependable and Secure Computing ». In : *IEEE Trans. Dependable Secur. Comput.* 1 (1 jan. 2004), p. 11-33.