HAL
open science

Systèmes d'information collaboratifs et auto-organisants pour réseaux de capteurs large-échelle : "De la théorie à la pratique"

Yann Busnel

To cite this version:

Yann Busnel. Systèmes d'information collaboratifs et auto-organisants pour réseaux de capteurs large-échelle : "De la théorie à la pratique". Réseaux et télécommunications [cs.NI]. Université Rennes 1, 2008. Français. NNT : . tel-00365659

HAL Id: tel-00365659

https://theses.hal.science/tel-00365659v1

Submitted on 4 Mar 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre: 3769

THÈSE

Présentée devant

devant l'Université de Rennes 1

pour obtenir

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

par

Yann BUSNEL

Équipe d'accueil : ASAP - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :

Systèmes d'information collaboratifs et auto-organisants pour réseaux de capteurs large-échelle

« De la théorie à la pratique »

soutenue le 18 novembre 2008 devant la commission d'examen

M.	Luc	BOUGÉ	Président
MM.	Carole	DELPORTE-GALLET	Rapporteurs
	Pierre	SENS	
MM.	Isabelle	GUÉRIN-LASSOUS	Examinateurs
	Vivien	QUÉMA	
Mme	Anne-Marie	KERMARREC	Directrice de Thèse
M.	Marin	BERTIER	Co-encadrant

Les thèses les plus fausses sont souvent les plus belles.
Pierre DANINOS. (1913–2005)

À Flavie et Virgile.

Membres du jury

Rapporteurs

Docteur Carole DELPORTE-GALLET,
Professeur des Universités à l'Université Paris 7 – Denis Diderot

Professeur Pierre SENS,
Professeur des Universités à l'Université Paris 6 – Pierre et Marie Curie

Examinateurs

Professeur Luc BOUGÉ (Président du jury),
Professeur des Universités à l'Ecole Normale Supérieure de Cachan – Bretagne

Professeur Isabelle GUÉRIN-LASSOUS,
Professeur des Universités à l'Ecole Normale Supérieure de Lyon

Docteur Vivien QUÉMA,
Chargé de Recherche au CNRS – L.I.Grenoble

Encadrants

Docteur Anne-Marie KERMARREC, (Directrice de Thèse)
Directrice de Recherche à l'INRIA Rennes – Bretagne Atlantique

Docteur Marin BERTIER, (Co-encadrant)
Maître de Conférence à l'INSA Rennes

Remerciements

Établir un ensemble exhaustif de remerciements est une tâche utopique, tant les personnes qui ont compté pour moi et m'ont soutenu au cours de ces trois dernières années sont nombreuses.

Pour commencer, je souhaite remercier Luc Bougé, Professeur à l'ENS Cachan – Bretagne, pour avoir accepté de présider mon jury de doctorat, mais également pour sa confiance et son accompagnement dans mes choix et projets depuis 2002.

De toute évidence, je remercie grandement Pierre Sens, Professeur à l'université Paris VI et Carole Delporte-Gallet, Professeur à l'université Paris VII, d'avoir accepté la lourde tâche qu'incombe le rôle de rapporteur. Leur lecture attentive, leurs remarques et critiques pertinentes ont permis d'améliorer sensiblement cette thèse.

Merci également à Vivien Quéma, Chargé de Recherche au CNRS Grenoble, et Isabelle Guerin-Lassous, Professeur à l'ENS Lyon, d'avoir accepté de jouer le rôle d'examinateur et d'avoir eu une curiosité sans limite concernant mes travaux de thèse.

La majeure partie de ces remerciements revient bien entendu à mes deux encadrants. Ma directrice de thèse, Anne-Marie Kermarrec, qui a su allier direction efficace, intuition pertinente, enthousiasme communicatif et encadrement d'une qualité exceptionnelle. Marin Bertier a été un co-encadrant qui a su manier avec brio une alliance entre esprit critique et soutien sans faille. Tous deux ont été des guides extrêmement enrichissants me permettant de prendre désormais mon envol de chercheur.

En outre, je tiens particulièrement à remercier tous ceux avec qui j'ai eu l'honneur de collaborer lors de ces trois années de doctorat : Roberto Baldoni, Ken Birman, Eric Fleury, Ali Ghodsi, Konrad Iwanicki, Hugo Miranda, Leonardo Querzoni, Cigdem Segul, Gilles Tredan, Maarten Van Steen, Aline Viana, Spyros Voulgaris et Hakim Weatherspoon.

Un remerciement chaleureux aux membres de l'équipe-projet ASAP pour leur gaieté et leur convivialité avec, en outre, Achour, Antoine, Corentin, Damien, Davide, Erwan, Francois, Gilles, Guang, Michel, Nicolas, Sathia, Vincent, Vincent et Xiao. Un remerciement particu-

lier pour Etienne Rivière et Kevin Hugenin pour avoir supporté mes frasques de co-bureau, et une mention spéciale pour Fabrice Le Fessant qui saura pourquoi. Merci également à nos assistantes dévouées et efficaces, à savoir Cécile et Fabienne. Je tiens également à remercier Thierry Priol, ainsi que l'équipe PARIS, pour m'avoir accueilli à l'occasion de mon stage de Master 2 Recherche et pendant les premiers mois de mon doctorat, avec un clin d'œil particulier à Loïc Cudennec.

J'ai eu le plaisir d'enseigner en tant que moniteur à l'ENS Cachan – Bretagne ainsi qu'à l'IFSIC durant plus de trois ans, et j'ai fortement apprécié l'environnement et les collègues avec lesquels j'ai eu l'occasion de sévir, parmi lesquels François Bonnet, Luc Bougé, Anne Bouillard, David Cachera, Philippe Ingels, Claude Jard, Tristan Le Gall, Sébastien Macé, Aomar Maddi, Fabienne Moreau, Murielle Pressigout, Louis Rilling, Corentin Travers. Merci aussi à mes collègues de l'ADOC, du conseil de laboratoire, et des stages du CIES Grand-Ouest et de l'école doctorale Matisse.

Un pensée sincère à mes amis matheux et danseurs sans le soutien desquels cette thèse n'aurait pu voir le jour, avec, dans le désordre, Mikaël, Jean-Romain, Viktoria, Mathilde, Clément, Olivier, Nathalie, Fanny, Richard, Sébastien, Thomas, Jean-Louis et également Carole, Sylvain, Marine, Agnès, Anthony, Olivier, Vanessa, Philippe, Sylvaine, Yannick, Odran, Adrien et tous les autres...

Derniers remerciements mais pas les moindres : je tiens à exprimer mon dévouement complet à ma famille. En premier lieu, celle qui m'a supporté avec patience et bienveillance tous les jours de ces trois années, ma femme Flavie, et qui m'a offert le plus beau des cadeaux il y a quelques semaines : mon fils Virgile. Merci également à mes parents : Michèle, qui a toujours su être présente et qui a été ma principale source de soutien depuis ma plus tendre enfance et Bernard, qui, à sa façon, m'a préservé et soutenu dans mes projets. Merci également à Cécile, Sophie, Christophe et Valentin ainsi qu'à ma belle-famille Carmen, Gérard, Marie, Romain, Martial et Yvette pour leur soutien et leur présence.

Je remercie finalement toutes les personnes auxquelles je pense régulièrement, mais qui ne peuvent pas toutes apparaître ici. Qu'elles ne m'en tiennent pas rigueur...

Table des matières

Membres du jury **1**

Remerciements **3**

Table des matières **4**

Introduction **9**
Objectifs de cette thèse 11
Organisation du document 12
Contributions 12
Liste des publications 15

1 Introduction générale aux réseaux de capteurs **17**
1.1 Contraintes physiques d'un capteur 18
1.1.1 Interface de communication 19
1.2 Défis intrinsèques aux réseaux de capteurs 21
1.2.1 Consommation énergétique 22
1.2.2 Tolérance aux défaillances 23
1.2.3 Couverture et connexité 24
1.3 Différents aspects de ces réseaux 25
1.3.1 Déploiement des nœuds 25
1.3.2 Modèle de données 25
1.3.3 Hétérogénéité 25
1.3.4 Critères d'évaluation 26
1.3.5 Agrégation des données 26
1.4 Ouverture et positionnement 26

PARTIE I : Réseaux de capteurs statiques **27**

2 Suivi et identification de trajectoires sur un réseau de capteurs binaires **29**
2.1 Introduction . . . 29
2.2 État de l'art succinct . . . 31
2.3 Modélisation du système . . . 31
2.3.1 Localisations et mouvements . . . 32
2.3.2 État du système . . . 32
2.3.3 L'observateur . . . 33
2.4 Identification d'objets et suivi de trajectoire . . . 33
2.4.1 Le problème MOTI . . . 34
2.4.2 Un résultat d'impossibilité . . . 34
2.5 Conditions de résolubilité de MOTI . . . 37
2.5.1 Caractérisation d'infaillibilité . . . 37
2.5.2 Un algorithme de génération du graphe des états SG . . . 39
2.5.3 Caractérisation de MOTI . . . 40
2.6 Une condition suffisante rendant MOTI $\mathbb{P}$-résoluble (*a priori*) . . . 41
2.7 Contraindre le mouvement des objets en temps réel avec un observateur actif . 44
2.7.1 Algorithme simple avec omniscience . . . 44
2.7.2 Algorithme sans connaissance du graphe des états . . . 45
2.8 Algorithmes répartis large-échelle sans observateur . . . 51
2.8.1 Vers une utilisation *a posteriori* . . . 54
2.9 Conclusion . . . 54

3 Structure générique multi-couches à faible consommation **55**
3.1 Introduction . . . 55
3.2 Prérequis du système . . . 56
3.3 La collection **-cast* . . . 57
3.4 Le cœur de SOLIST . . . 58
3.4.1 Une structure multi-couche . . . 58
3.4.2 Structure de couche : LIGH-t-LAYER . . . 59
3.4.3 Relier les mondes : les points d'entrées . . . 61
3.4.4 Résumé des prérequis . . . 63
3.5 La collection **-cast* dans SOLIST . . . 64
3.5.1 Anycast . . . 64
3.5.2 Broadcast . . . 64
3.5.3 k-cast . . . 66
3.6 Évaluation des performances . . . 68
3.6.1 Environnement de simulation . . . 68
3.6.2 Présentation des algorithmes de comparaison . . . 69
3.6.3 Fiabilité de SOLIST . . . 70
3.6.4 Consommation d'énérgie . . . 72
3.7 Comparatif de travaux relatifs . . . 76
3.8 Conclusion . . . 77

PARTIE II : Réseaux de capteurs mobiles **79**

4 Modèles de calcul pour réseaux de capteurs mobiles **81**
4.1 Contexte général 81
4.2 Protocoles de population 83
4.2.1 Fonctionnalités de bases 83
4.2.2 Formalisation du modèle 83
4.2.3 Exemples de protocoles de population 85
4.2.4 Puissance du modèle 86
4.2.5 Travaux connexes 87
4.3 Protocoles de communauté 87
4.3.1 Motivation 87
4.3.2 Formalisation du modèle 88
4.3.3 Puissance du modèle 88
4.3.4 Travaux connexes 88
4.4 Conclusion 89

5 Prise en compte de la mobilité dans les protocoles de population **91**
5.1 Motivation 91
5.2 Mobilité avec les protocoles de population et de communauté 92
5.2.1 Les modèles MAPP et MAPC 92
5.2.2 Pertinence du modèle 93
5.2.3 Analyse fondamentale de la convergence 94
5.3 Impact des modèles de mobilité sur la vitesse de convergence 99
5.3.1 Echantillon des distributions de probabilité sur Λ 99
5.3.2 Estimation en temps discret : les modèles de rencontre 100
5.3.3 Estimation en temps continu : modèles d'inter-rencontre 105
5.4 Une borne inférieure optimale de la vitesse de convergence 108
5.5 À propos de la pertinence du modèle de RWP 114
5.6 Conclusion 116

6 Équivalence des protocoles épidémiques et de population **117**
6.1 Les protocoles épidémiques 117
6.1.1 Historique des protocoles épidémiques 118
6.1.2 Protocole épidémique générique 118
6.1.3 Deux grandes classes de protocoles épidémiques 119
6.2 Motivation 121
6.3 Une classification des protocoles épidémiques 122
6.3.1 Entre synchronisme et asynchronisme 122
6.3.2 Puissances des classes de protocoles épidémiques 123
6.4 Combler le fossé entre les protocoles épidémiques et de population 124
6.4.1 Equivalence entre les protocoles de population et de asyncPENA 125
6.4.2 Equivalence entre les protocoles de communauté et de PENI 128
6.4.3 Un impact potentiel pour les deux domaines 130

6.5 Optimalité du RPS en terme de vitesse de convergence 133
6.6 Conclusion 133

Conclusion **135**
Contexte d'étude 135
Contributions 136
Perspectives 137

Bibliographie **141**

Table des figures **157**

Liste des algorithmes **159**

Lexique **161**
Acronymes 161
Notations mathématiques 162
Notations algorithmiques 163

Introduction

Depuis l'apparition des premiers calculateurs au début du siècle dernier, l'évolution des systèmes informatiques a été jalonnée de grandes étapes de miniaturisation. En effet, l'apparition du transistor, suivie de la puce micro-électronique, des ordinateurs personnels, des ordinateurs transportables, puis de poche sont autant de révolutions qui ont marqué ce processus de progression. À l'aube du XXIe siècle, certains téléphones portables sont plus puissants que les ordinateurs de bureau commercialisés à la fin des années 1980. Désormais, les systèmes électroniques de taille réduite sont omniprésents dans notre environnement.

A contrario de cette quête de diminution de la taille des composants, et *a fortiori* de celle des machines dites intelligentes, l'évolution de l'informatique a nécessité, pour sa part, un accroissement sans pareil des besoins, en terme de puissance de calcul et de stockage pour le traitement de l'information et la communication. Les systèmes centralisés sont arrivés logiquement à leur limite intrinsèque et l'apparition des réseaux communicants a permis d'atteindre une puissance de calcul comparable aux super-calculateurs pour un coût minime. Par la suite, la nécessité du partage d'informations à distance et l'évolution des mœurs de notre société vers une hyper-communication a permis la démocratisation de ces systèmes dit *répartis*. Ces systèmes permettent de distribuer les puissances de calcul et de stockage parmi un nombre grandissant d'entités participantes. De l'ordre de la dizaine au moment de la démocratisation du réseau Internet, à la fin des années 1980, ces systèmes répartis atteignent désormais communément la centaine de millions de participants avec la popularisation des *systèmes pair-à-pair*, tels que Napster [Nap08], Skype [Sky08], ou autres BitTorrent [Bit08].

Le désir de miniaturisation, jumelé avec les technologies de ces systèmes communicants à calculs répartis, a fait récemment éclore un ensemble de nouveaux systèmes dit d'*informatique diffuse* ou *embarquée*. La réduction des entités électroniques se trouvant face à une limitation physique des composants, la distribution de la puissance au sein d'un système dit *collaboratif* a permis de pallier cette limitation. Equipés d'un périphérique de communication sans-fil, les éléments de ces nouveaux systèmes peuvent donc être aisément déployés, sans nécessiter

d'infrastructure matérielle coûteuse, encombrante et pénible à mettre en place et à entretenir. La restriction concernant la puissance et le déploiement s'écartant, la réduction de taille des entités physiques s'est intensifiée permettant de voir apparaître des ordinateurs plus petits qu'une pièce d'un centime d'Euro. Ces nouveaux micro-ordinateurs, qui sont au cœur de notre étude, se situent actuellement à la frontière avec un nouveau domaine émergent : celui des *nano-technologies*.

Réseaux de capteurs En parallèle, les applications de surveillance, de protection et d'observation de notre environnement se sont accrues ces dernières années, aussi bien dans le milieu médical qu'écologique ou militaire. Malheureusement, l'intervention humaine n'est pas réalisable dans tous les lieux d'investigation. Les années 2000 ont alors vu la naissance de «*poussières intelligentes*», ensemble de plate-formes matérielles miniaturisées couplées avec un module d'acquisition de données permettant de capter et de collecter des stimuli ou événements issus de l'environnement local de ce *capteur*. Les faibles possibilités technologiques d'un capteur unique ne lui permettent pas d'avoir une utilité propre, mais la mise en réseau d'un grand nombre de ceux-ci ouvre des opportunités de progrès et de fonctionnement palpitantes.

L'intérêt des communautés issues de la recherche et de l'industrie pour ces *réseaux de capteurs sans fil* (*RCsF*) s'est accru par la potentielle fiabilité, précision, flexibilité, faible coût ainsi que la facilité de déploiement de ces systèmes. Les caractéristiques des RCsF permettent d'envisager un grand nombre d'applications d'observation répartie dans l'espace. Ces dernières peuvent se déployer dans de nombreux contextes : observation de l'environnement naturel (pollution, inondation, ...), suivi d'écosystèmes (surveillance de niches d'oiseaux, croissance des plantes, ...), militaire (télésurveillance de champs de bataille, détection d'ennemis, ...), biomédical et surveillance médicale (détection de cancer, rétine artificielle, taux de glucose, diabètes, ...), *etc.* La spontanéité, l'adaptabilité du réseau et la dynamicité de sa topologie dans le déploiement des RCsF soulèvent de nombreuses questions encore ouvertes. L'une des limitations contraignantes de ces réseaux est la faible ressource énergétique des capteurs due essentiellement à leur taille minimaliste et leur indépendance filaire. Cette contrainte doit être considérée comme la préoccupation de premier ordre dans la conception et le déploiement d'un RCsF.

Auto-organisation Compte tenu du nombre potentiellement important de capteurs participants et de leurs ressources réduites, il apparaît indispensable de développer des solutions entièrement décentralisées, répartissant la charge entre les participants et permettant de passer à l'échelle en terme de nombre d'entités déployées. Par conséquent, dans le contexte des RCsF, l'approche collaborative dans la conception de systèmes dédiés apparaît nécessaire. En effet, ces derniers sont la plupart du temps développés dans un cadre applicatif précis afin d'optimiser la consommation d'énergie locale et globale, dans le but d'optimaliser la durée de vie du réseau. Enfin, cet ensemble de capteurs devant être capable de délivrer un service sans nécessiter d'intervention extérieure (humaine ou matérielle), la collaboration doit être conçue de manière *auto-organisante*. Cette dernière notion a pour optique l'émergence d'un comportement général et global du système, à partir d'acteurs indépendants et possédant uniquement des informations restreintes et locales du système. Ainsi, la charge est répartie sur l'ensemble

du système, et l'augmentation du nombre de participants n'entraîne pas la formation de goulot d'étranglement, ni de point de défaillance critique (par l'unicité du serveur), couramment observés sur les systèmes communicants dit centralisés.

Limitations des systèmes existants Depuis leur récente apparition, les RCsF ont connu une effusion d'intérêt permettant en quelques années de réunir un ensemble de connaissances considérable. Nombreuses sont les applications déployées fournissant des services fondés sur un RCsF de nos jours. Par exemple, nous pouvons citer la surveillance de zone sécurisée, ou la domotique[1]. Malheureusement, la grande majorité des résultats scientifiques et techniques dans ce domaine a été développée dans un contexte applicatif. À ce jour, encore très peu de travaux présentent des analyses fondamentales et/ou théoriques des RCsF. Une modélisation générique communément admise, permettant d'explorer les enjeux et les limitations de ce domaine, manque toujours à cette communauté en plein essor.

Objectifs de cette thèse et positionnement

Cette thèse s'inscrit dans l'apport de services et de formalisation de résultats fondamentaux pour les RCsF à large échelle, avec une approche collaborative et totalement décentralisée issue des systèmes répartis de grande taille. Plus particulièrement, deux objectifs principaux sont poursuivis :

1. la proposition de fondements et de mécanismes permettant d'analyser *a priori* un système dans le contexte des RCsF. Nous étudions pour cela différents problèmes et modèles avec une approche formelle, à la fois dans le contexte des RCsF statiques (*e.g.* suivi déterministe de trajectoires) et mobiles (*e.g.* formalisation des modèles de mobilité) ;
2. la proposition de services en temps réel permettant d'obtenir une meilleure fiabilité et efficacité, tels que la mise en œuvre d'une structure générique à la majorité des RCsF, ou l'adaptation de solutions épidémiques issues des réseaux filaires aux problèmes rencontrés sur les RCsF.

Comme introduit ci-avant, de nombreuses solutions ont déjà été proposées dans le contexte des RCsF. Cependant, de manière générale, le cheminement de la plupart des avancées significatives sur ces systèmes est commun. En effet, les solutions proposées sont majoritairement évaluées de manière empirique sur des plates-formes de simulation numérique ou sur des bancs de test, dit d'environnement réel, mais malgré tout figés et limités au matériel disponible sur ceux-ci. Ainsi, la validation des contributions s'effectue en général sur un sous-ensemble de cas de figures potentiellement représentatif, mais néanmoins restreint.

Afin d'atteindre les objectifs sus-cités, nous proposons dans un premier temps de poursuivre une approche théorique des réseaux de capteurs afin d'identifier les possibilités, les obstacles et les frontières de ce domaine. Dans un second temps, les résultats démontrés et admis antérieurement nous permettent de proposer des solutions robustes et efficaces dans le contexte de nos modèles envisagés et selon les critères identifiés. Nous appliquerons cette méthodologie

[1] La domotique est l'ensemble des technologies de l'électronique, de l'informatique et des télécommunications utilisées dans le contexte du domicile de particuliers.

Cadre \ Environnement	Réseaux statiques	Réseaux mobiles
Caractère fondamental	MOTI [BQB+08a, BQB+08b]	MAPP [BBK08b] PE ≅ PP et PC [BBK08a]
Caractère applicatif	SOLIST [BBK07, BBK08c, BBK08d]	GCP [BBFK07b, BBFK07c]

TAB. 1 – Situation des contributions personnelles

passant idéologiquement *de la théorie à la pratique* et *du particulier au général* au sein des deux grandes classes de réseaux de capteurs existantes, à savoir les réseaux statiques et les réseaux mobiles.

Organisation du document

Structurellement, ce document s'articule autour de ces deux grandes classes. La première partie traite uniquement des problématiques liées aux réseaux de capteurs dits statiques, les acteurs de ces systèmes étants déposés ou fixés de façon pérenne dans l'environement d'observation. *A contrario*, la seconde partie de ce document est consacrée aux réseaux de capteurs dits mobiles.

En premier lieu, nous introduisons ce manuscrit par un chapitre disjoint présentant de manière générique les caractéristiques et les défis relatifs à tout réseau de capteurs, et plus précisément ceux de très grande taille. Ensuite, à l'exception du chapitre 4, chaque partie sera composée de chapitres décrivant les contributions proposées au cours de mes travaux de recherche. Effectivement, en introduction à la seconde partie, le chapitre 4 présente un ensemble de modélisations de réseaux de capteurs mobiles. Cet ensemble de notions est commun et nécessaire aux derniers chapitres de cette partie. À l'image des adages « *de la théorie à la pratique* » et « *du particulier au général* », chacune des deux grandes parties présente en premier lieu des contributions à caractère fondamental permettant d'analyser théoriquement et *a priori* un système donné, puis, en second lieu, un ensemble de solutions plus d'ordre pratique et appliqué, issues des résultats antécédents.

Contributions et publications

L'ensemble de mes contributions présentées dans ce document est ordonné au sein du tableau 1, permettant de cibler l'environnement et les enjeux de celles-ci. Nous introduisons ici une courte présentation de ces différentes contributions. En premier lieu, dans le contexte des réseaux de capteurs statiques, nous proposons :

MOTI – Chapitre 2 Étant donné un ensemble de capteurs binaires permettant de suivre les trajectoires d'objets mobiles anonymes, ce chapitre étudie le problème de l'association déterministe d'un chemin révélé par un réseau de capteurs statiques avec la trajectoire

réelle d'un unique objet. Ce problème est nommé *Suivi d'Objets Multiples et Identification* (SOMI, ou son équivalent anglo-saxon *MOTI* utilisé dès lors). La difficulté de MOTI réside dans le fait que les trajectoires de deux objets – ou plus – peuvent être si proches qu'elles en deviennent indiscernables, rendant impossible l'association déterministe des chemins observés avec les trajectoires réelles des objets. Nous montrons que MOTI ne peut être résolu dans le cas général, même en présence d'un observateur omniscient externe. Par l'étude de différentes restrictions du système, nous identifions les limites de solvabilité du problème et proposons des solutions *a priori*, *en temps réel* et *a posteriori* pour résoudre ce problème. Ces travaux ont été effectués en collaboration avec Roberto Baldoni, Professeur (Université « La Sapienza » de Rome) et Leonardo Querzoni.

SOLIST – Chapitre 3 Les cœurs des systèmes de gestion de données dans les systèmes répartis reposent sur certaines fonctionnalités de base telles que la diffusion ou la recherche de capteurs particuliers. Dans ce chapitre, nous proposons la mise en œuvre décentralisée d'une collection de primitives appelées **-cast* (*anycast*, *k-cast* et *broadcast*). Pour cela, nous présentons SOLIST, une structure multi-couche, largement inspirée des réseaux pair-à-pair structurés, limitant la consommation d'énergie dans le cadre des RCsF. SOLIST est évaluée par simulation sur SeNSim [BBT08], un simulateur générique de réseaux de capteurs que nous avons développé, permettant d'attester la pertinence de l'approche.

D'autre part, nous présentons les contributions suivantes, dans le contexte des réseaux de capteurs mobiles :

Modèles et paradigmes – Chapitre 4 Nous présentons deux modélisations de réseaux de capteurs mobiles. Les *protocoles de population* et de *communauté* fournissent des fondements théoriques pour modéliser et ainsi analyser formellement le comportement et l'évolution des RCsF mobiles.

MAPP – Chapitre 5 Nous proposons dans ce chapitre une extension de ces modèles permettant d'analyser théoriquement l'influence des modèles de mobilité sur la convergence des algorithmes réalisés par des protocoles de populations : *MAPP* (*Mobilité Appliquée aux Protocoles de Population*). Nous conjecturons de manière empirique l'existence d'impacts significatifs dus à certaines caractéristiques de ces modèles de mobilité. À l'aide de ces résultats, nous montrons l'existence d'une borne inférieure sur la vitesse de convergence moyenne de n'importe quel protocole de population (respectivement de communauté), correspondant également à celle du modèle de mobilité dominant pour l'évaluation de systèmes mobiles, dit *points de navigation aléatoires* (*RWP* issu de *Random WayPoint* pour les anglo-saxons).

PE $\cong$ PP et PC – Chapitre 6 Les protocoles épidémiques fournissent une interface commune, fondée sur l'échange périodique d'informations entre les participants d'un système. Convaincu de l'analogie avec les protocoles de population (et de communauté), nous montrons dans ce chapitre que la complexité et la puissance de calcul des protocoles épidémiques (quelle que soit la classe considérée – *cf.* chapitre 4) sont strictement équivalentes à celles des protocoles de population et de communauté. Pour cela, nous proposons une classification des protocoles épidémiques en fonction de l'anonymat des nœuds et du

synchronisme du canal de communication. La preuve des équivalences permet d'apparenter l'ensemble de ces modèles et de réunir les travaux menés sur ceux-ci. Nous illustrons cette assertion par deux exemples succincts. De même, nous concluons en montrant que des résultats issus du chapitre précédent peuvent être ainsi directement appliqués aux protocoles épidémiques.

Hors du contexte de ce document, nous pouvons citer trois autres contributions personnelles.

- D'une part, nous avons développé un simulateur de RCsF, permettant de prendre en compte la mobilité des nœuds d'un réseau : *SeNSim* [BBT08]. Ce logiciel, dont le code source est écrit en Java^{TM}, permet la synthétisation d'un réseau de capteurs statiques ou mobiles large-échelle et l'analyse comportementale des protocoles et algorithmes simulés sur ce dernier. Fondé sur un système à événements discrets, SeNSim permet de simuler différents scénarios de mobilités, de défaillances et d'événements.
- D'autre part, nous avons évalué les enjeux, les avantages et les limitations des systèmes de mémoire répartie à large-échelle et nous mettons en exergue les qualités des protocoles épidémiques dans ce cadre [WMI^{+}07].
- Enfin, nous avons proposé une mise en pratique de l'utilisation d'un protocole épidemique dans le contexte des réseaux de capteurs mobiles [BBFK07b]. Nous proposons une approche de mise à jour automatique de logiciel sans mécanisme d'identification prérequis. Ces travaux présentent donc la conception et l'évaluation du protocole proposé : *GCP* (*Gossip-based Code Propagation*) [BBFK07c]. De même que SOLIST, ce protocole a été évalué par simulation sur SeNSim et comparé avec les performances des algorithmes classiques de dissémination de données. La pertinence de GCP a été mise à l'épreuve selon des modèles de mobilités stochastiques et des traces de mobilités réelles. Ces travaux ont été effectués en collaboration avec Eric Fleury, Professeur à l'Ecole Normale Supérieure de Lyon.

Liste chronologique des publications

2005

[Bus05a] Y. Busnel. **Prise en compte d'aspects sémantiques dans la contruction d'un réseau pair-à-pair.** Dans la *3ème MAnifestation des JEunes Chercheur francophones dans les domaines des STIC (MajecSTIC '05)*, volume 3, pages 435–438, Rennes, France, Novembre 2005.

2006

[Bus06a] Y. Busnel. **ProxSem : Interest-based proximity measure to improve search efficiency in P2P systems.** Dans *1st conference of the European Chapter of ACM SIGOPS,* Poster Session *(EuroSys '06)*, Leuven, Belgique, Avril 2006.

[Bus06b] Y. Busnel. **Système d'information pair-à-pair pour les réseaux de capteurs larges échelles.** Dans la *Lettre de la Fondation Michel Métivier*, 5 :3–5, Octobre 2006.

[BK06] Y. Busnel et A.-M. Kermarrec. **ProxSem : Mesure de proximité sémantique pour les systèmes de partage de fichiers pair-à-pair.** Dans la *5ème Conférence Française en Systèmes d'Exploitation (CFSE '5)*, volume 5, pages 37–48, Perpignan, France, Octobre 2006.

2007

[BK07] Y. Busnel et A.-M. Kermarrec. **ProxSem : Interest-based proximity measure to improve search efficiency in P2P systems.** Dans *4th European Conference on Universal Multiservice Networks (ECUMN '07)*, pages 62–71, Toulouse, France, Février 2007.
(Rapport de recherche intégral disponible [BK05])

[BBFK07a] Y. Busnel, M. Bertier, E. Fleury, et A.-M. Kermarrec. **GCP : Gossip-based code propagation for large-scale mobile wireless sensor network.** Dans *2nd conference of the European Chapter of ACM SIGOPS,* Poster Session *(EuroSys '07)*, Lisbonne, Portugal, Mars 2007.

[Bus07] Y. Busnel. **Le modèle de pair à pair profite aux réseaux de capteurs très étendus.** Dans *Interstices : Découvrir la recherche en informatique*, page 22096, Mai 2007.

[WMI+07] H. Weatherspoon, H. Miranda, K. Iwanicki, A. Ghodsi, et Y. Busnel. **Gossiping over storage systems is practical**. Dans *ACM Operating System Review*, Octobre 2007.

[BBFK07b] Y. Busnel, M. Bertier, E. Fleury, et A.-M. Kermarrec. **GCP : Gossip-based code propagation for large-scale mobile wireless sensor networks.** Dans *1st International Conference on Autonomic Computing and Communication Systems (Autonomics '07)*, Rome, Italy, Octobre 2007.
(Rapport de recherche intégral disponible [BBFK07c])

2008

[BBK08c] Y. Busnel, M. Bertier, et A.-M. Kermarrec. **Solist : Structure multi-couche pair-a-pair à faible consommation pour les réseaux de capteurs sans-fil.** Dans la *6ème Conférence Française en Systèmes d'Exploitation (CFSE '6)*, Fribourg, Suisse, Février 2008.

[BQB+08b] Y. Busnel, L. Querzoni, R. Baldoni, M. Bertier, et A.-M. Kermarrec. **Suivi et identification de trajectoires sur un réseau de capteurs binaires.** Dans la *10ème Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (AlgoTel '08)*, Saint-Malo, France, Mai 2008.

[BQB+08a] Y. Busnel, L. Querzoni, R. Baldoni, M. Bertier, et A.-M. Kermarrec. **On the deterministic tracking of moving objects with a binary sensor network.** Dans *4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '08)*, volume LNCS 5067, pages 46–59, Ile Santorin, Grèce, Juin 2008.

[BBK08b] Y. Busnel, M. Bertier, et A.-M. Kermarrec. **On the impact of the mobility on convergence speed of population protocols.** *Rapport de Recherche RR-6580*, Institut National de Recherche en Informatique et Application (INRIA), Rennes, France, Juillet 2008.

[BBK08d] Y. Busnel, M. Bertier, et A.-M. Kermarrec. **SOLIST or how to look for a needle in a haystack ?** Dans *4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '08)*, Avignon, France, Octobre 2008.
(Rapport de recherche intégral disponible [BBK07])

[BBK08a] Y. Busnel, M. Bertier, et A.-M. Kermarrec. **Bridging the Gap between Population and Gossip-based Protocols.** *Rapport de Recherche RR-6720*, Institut National de Recherche en Informatique et Application (INRIA), Rennes, France, Novembre 2008.

[BBT08] M. Bertier, Y. Busnel et G. Tredan. **SeNSim (Sensor Network Simulator)** : `http://sensim.gforge.inria.fr/`. Équipe-Projet ASAP, INRIA Rennes – Bretagne Atlantique, France, 2005–2008.

CHAPITRE 1

Introduction générale aux réseaux de capteurs

Les progrès significatifs en électronique de capture ont permis une nette amélioration des capteurs depuis une décennie. En effet, les premiers capteurs, de taille plus conséquente, étaient souvent placés loin de l'*événement* à percevoir, nécessitant souvent des techniques complexes de différentiation des cibles à observer par rapport au bruit environnant. Dorénavant, un RCsF est composé d'un nombre important de petits nœuds, lesquels sont déployés au cœur même du phénomène à observer, sinon à une distance très courte. La position des capteurs ne nécessite plus une étude préalable et n'a pas à être pré-déterminée. Ceci permet par exemple un déploiement chaotique sur des zones inaccessibles ou dévastées. Un panel de quelques nœuds capteurs récemment mis en œuvre est présenté en figure 1.1.

Par conséquent, les algorithmes utilisés sur un RCsF actuel nécessitent une capacité d'*auto-organisation*, mise en œuvre par une approche *collaborative*. Ainsi, les nœuds d'un RCsF ne transmettent qu'un sous-ensemble des informations collectées, lesquelles sont d'abord traitées localement par de simples calculs.

Deux grandes classes de réseaux Le déploiement de RCsF soulève un certain nombre de questions conceptuelles telles que l'auto-organisation, l'adaptabilité ou la dynamique de la topologie du réseau. Parmi les différentes caractéristiques d'un RCsF, la *mobilité* des nœuds dans le temps et dans l'espace scinde ces réseaux en deux grandes classes. Alors que des solutions se reposant sur la connaissance des coordonnées géographiques sont réalisables simplement dans un contexte de RCsF statique (*cf.* chapitres 2 et 3), elles deviennent irréalisables, ou très coûteuses en énergie, en présence de forte mobilité. De plus, toute application à fondement collaboratif repose sur la communication des participants. Pour cela, l'efficacité d'une solution dépend notamment du protocole de routage sous-jacent, de l'anonymat des nœuds, ou encore du modèle de déplacements de ceux-ci.

Ainsi, il paraît naturel de scinder en deux parties la présentation des propositions existantes pour les RCsF. De la même manière, l'organisation de ce document s'articule également autour de deux grandes parties, l'une dans le contexte des RCsF statiques, et la seconde dans celui des

Type SensLAB [Sen08] © SensLAB | Type Mica2dot © Crossbow Tech. Inc. | Type Telos © UC Berkeley | Type MicaZ © Crossbow Tech. Inc.

FIG. 1.1 – Echantillon de spécimens de capteurs existants.

RCsF mobiles.

Dans ce chapitre, nous proposons, en premier lieu, une présentation générique d'un nœud capteur classique ainsi que la modélisation de communication sans-fil choisie pour nos expérimentations. Puis, nous exposons différents défis et enjeux des réseaux de capteurs de manière globale, en identifiant les contraintes en présence de mobilité ou non. Enfin, nous introduisons un inventaire des paramètres spécifiques à prendre en compte dans la conception de systèmes sur les RCsF, mobile ou non. Afin d'illustrer les différents travaux présentés dans la suite, nous conservons dans ce chapitre une vision large-échelle des RCsF, avec une approche collaborative et auto-organisante.

1.1 Contraintes physiques d'un capteur

Un capteur est composé de quatre unités de base, à l'instar de la représentation en figure 1.2 : un *module de capture*, un *module de traitement*, un *module de communication* et un *module d'alimentation* [ASSC02b, VDSJCJDM03].

Le module de capture, permettant la collecte de données issus de *stimuli* externes, est le plus souvent constitué de deux sous-composants : des capteurs classiques combinés à un *convertisseur analogique-vers-numérique* (*ADC* pour *Analog-to-Digital Converter*). Le signal analogique, généré par le phénomène observé perçu par les capteurs, est converti en un signal numérique par l'ADC et transmi au module de traitement. L'étude de ce module se trouve être hors du contexte de ce document.

Le module de traitement est composé d'un processeur – ou *Unité Centrale de Traitement* (*UCT*, ou *CPU* pour *Central Processing Unit*) – généralement associé à un périphérique de stockage de taille réduite. Il exécute le processus permettant la collaboration des nœuds entre eux afin de mener à bien les tâches de capture dévolues. Il représente le principal support matériel utilisé dans ce manuscrit.

Le module de communication permet de connecter les nœuds du réseau, quels que soient le support et le média utilisé par celui-ci. Le plus souvent, il est constitué principalement d'un périphérique de communication sans-fil de type Bluetooth, ZigBee ou même WiFi (*cf.* paragraphe 1.1.1). Il représente la principale source de consommation d'énergie et doit ainsi être largement pris en compte. Nous ne consacrons pas une étude approfondie de ce module dans ce

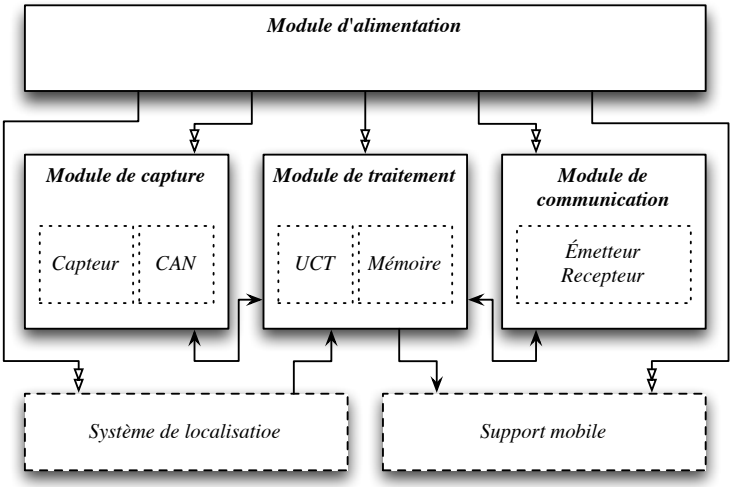

FIG. 1.2 – Constitution standard d'un capteur.

document, mais toutes nos conceptions utilisent ce module et sont motivées par une limitation d'utilisation afin d'économiser l'énergie.

Le module d'alimentation est quant à lui l'un des composants les plus importants d'un nœud. Il assure la durée de vie du capteur en approvisionnant en énergie tous les modules internes et externes du capteur. De même que pour le module de communication, toute conception de systèmes sur des RCsF doit prendre en compte l'utilisation de ce module. Nous cherchons dans toutes nos contributions à limiter son utilisation et ainsi améliorer la durée de vie du nœud capteur.

Il existe également de nombreux autres sous-modules ou périphériques additionnels. La figure 1.2 présente notamment les deux qui sont utilisés dans ce document. Dans le cadre des RCsF statiques, il est usuel de disposer d'un système de localisation, permettant d'obtenir la position géographique du capteur considéré. Ce module sera nécessaire dans le cadre des chapitres 2 et 3. Dans la seconde partie de ce document, nous considérons que les capteurs seront embarqués sur un support mobile, imposant alors une toute autre conception de système. Avant de présenter ces différences dans les systèmes d'information pour les RCsF, nous étendons la description du module de communication et présentons notre modélisation de celui-ci utilisée dans le reste de ce document.

1.1.1 Interface de communication

Concernant les médias de communication sans-fil, les différentes alternatives communément admises sont fondées soit sur l'onde radioélectrique, l'optique ondulatoire (Laser) ou l'infrarouge. L'utilisation du Laser requiert une énergie minime mais exige un alignement des lignes de visée entre deux capteurs (récepteur unidirectionnel) et reste très sensible aux conditions atmosphériques environnantes. L'infrarouge, comme le Laser, ne nécessite pas d'antenne mais reste cependant très limité quant à la taille de la bande passante disponible. À cet égard,

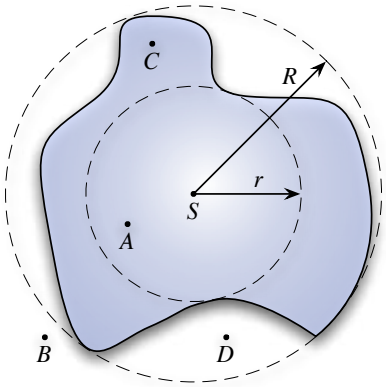

FIG. 1.3 – Zone de transmission d'un capteur standard, équipé d'une antenne omnidirectionnelle.

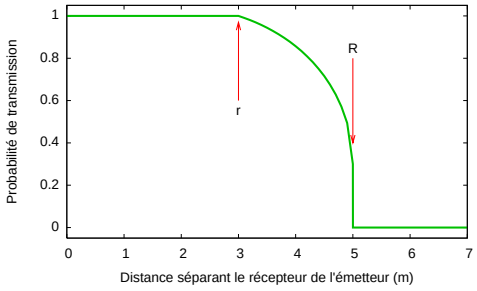

FIG. 1.4 – Probabilité de réussite d'une transmission en fonction de la distance séparant les deux capteurs.

l'onde radioélectrique s'impose comme étant le support de communication le plus approprié pour la majorité des applications sur les RCsF. C'est le type de communication que nous considérons dans l'intégralité de ce document. Les modules d'émission et de réception sont, la plupart du temps, combinés sur un même périphérique de communication. Par la suite, pour représenter ce module de communication, nous utiliserons le néologisme *transcepteur* (contraction de *transmetteur* et *récepteur*, à l'instar des anglo-saxons utilisant couramment le néologisme *transceiver* issus de *transmitter* et *receiver*).

Les transcepteurs peuvent fonctionner dans quatre modes différents : *émission*, *réception*, *en attente* et *éteint*. Il est important de préciser qu'un transcepteur dans un état d'attente consomme environ autant que dans un état de réception [XHE01]. Il est donc préférable d'éteindre complètement le module lorsqu'il n'envoie, ni ne reçoit, de paquets.

En outre, différents types de standard de communication existent et leurs utilisations varient selon les besoins des applications, en termes de bande passante, de distance de transmission, de robustesse du signal, *etc.* Alors que l'informatique classique regorge d'une profusion de standards, il n'existe qu'un unique standard officiel adopté pour les RCsF : le *WirelessHART* [HAR07]. De nombreux standards sont pour autant utilisés très largement dans la communauté, à savoir le *WiFi* (IEEE 802.11x[1]), le *Bluetooth* (IEEE 802.15.1), le *ZigBee* (IEEE 802.15.4) ou encore, le futur prometteur *Wibree* (encore à l'état de *draft IETF* à ce jour). Chacun de ceux-ci possède leurs avantages et inconvénients propres. Le WiFi permet une large couverture de zone ainsi qu'un haut-débit de transmission de donnée. Cependant, la consommation d'énergie nécessaire à son utilisation l'exclut souvent lors d'une mise en œuvre réelle. Le Bluetooth et le ZigBee ont une consommation plus faible, mais également un débit et une zone de transception limités. Le premier permet des communications point-à-point tandis que ZigBee facilite la communication « un-vers-tous ».

[1] À ce jour, 5 standards WiFi, dépendant des caractéristiques de taux de transfert ou de portée du signal du standard, sont représentés par les lettres : a, b, g, n ou y.

L'interface communément admise de transmission par radiodiffusion a pour nature une communication omnidirectionnelle. Dans la suite de ce document, pour toutes nos simulations, nous considérons le modèle suivant. Les nœuds peuvent communiquer uniquement de proche en proche, par diffusion locale, limitée au rayon de transception du capteur. Considérons un nœud S représenté au centre de la figure 1.3. Nous spécifions r et R, deux rayons de longueurs différentes, avec les propriétés suivantes : (1) $0 \leq r \leq R$ et (2) $\mathcal{S}_S(r) \subseteq \mathcal{S}_S(R)$[2]. r représente le rayon de la sphère pour laquelle le taux de transmission est uniforme, et à l'intérieur de laquelle, tout nœud situé à une distance inférieure à r reçoit de façon certaine les messages envoyés par S (*e.g.* le nœud A sur la figure 1.3). Le second rayon R représente la distance pour laquelle la tansception peut ne pas être uniforme. Aucun nœud séparé d'un autre par une distance d'au moins R ne peut recevoir les transmissions issues de ce dernier (*e.g.* le nœud B sur la même figure). Par conséquent, les nœuds séparés par une distance comprise entre r et R pourront, ou ne pourront pas, recevoir les messages émis de l'un ou de l'autre (*e.g.* les nœuds C et D par rapport à S en figure 1.3). Plus formellement, nous proposons une modélisation du comportement réel des transmissions : la probabilité de réception d'un message émis par un nœud à une distance d est donnée en équation (1.1).

$$\mathbb{P}[\text{réception}|d] = \begin{cases} 1 & \text{si } d < r \\ P_{min} + \sqrt{\frac{R-d}{R-r}} \cdot \left(5 - \frac{R-d}{R-r}\right) \cdot \frac{1-P_{min}}{4} & \text{si } r < d < R \\ 0 & \text{si } d > R \end{cases} \tag{1.1}$$

Dans cette équation, P_{min} représente la borne inférieure de probabilité de recevoir un message à une distance égale à R. Une représentation graphique de l'évolution de la probabilité en fonction de la distance séparant l'émetteur considéré d'un récepteur potentiel est donnée en figure 1.4, avec les paramètres suivants : $r = 3$, $R = 5$ et $P_{min} = 0{,}3$.

Dans le reste de ce chapitre, nous présentons les différents défis et caractéristiques communs à tous les RCsF, en introduisant à chaque fois les différences existantes pour le contexte statique et le contexte mobile.

1.2 Défis intrinsèques aux réseaux de capteurs

Alors qu'un grand nombre d'applications met en œuvre des RCsF, ces derniers possèdent plusieurs restrictions inhérentes à prendre en compte dans leur conception, qu'ils soient statiques ou mobiles. En plus du fait qu'un capteur souffre de limitations locales, telles qu'une faible puissance de calcul, une réserve d'énergie limitée et une bande passante de communication réduite, il faut également prendre en compte des aspects globaux du système. Nous nous présentons ci-après un ensemble de défis à considérer d'un point de vue global lors du développement d'une application [AKK04].

[2] $\mathcal{S}_S(x)$ représente la sphère de centre S et de rayon x.

1.2.1 Consommation énergétique

Un capteur sans-fil est le plus souvent équipé d'une source énergétique de faible capacité (*i.e.* moins de 0,5 Ah pour 1,2 V) [ASSC02b]. Dans de nombreux contextes applicatifs, le renouvellement des ressources énergétiques est simplement impossible. La durée de vie d'un capteur, et par conséquent celle du réseau, est donc étroitement liée à celle de son module d'alimentation (communément nommé, par abus de langage, sa *batterie*).

La gestion et la conservation d'énergie tiennent une place d'autant plus primordiale. C'est essentiellement pour cette raison que la contrainte énergétique est considérée comme étant celle de premier ordre par la communauté. Les travaux de recherche se focalisent sur la conception de protocoles et d'algorithmes à faible consommation pour les RCsF. D'un autre côté, dans le contexte des *réseaux mobiles ad hoc* (dénotés *MANET* pour *Mobile Ad-hoc NETwork*), la consommation d'énergie représente un facteur de conception d'autant moins contraignant que les ressources peuvent être remplacées ou renouvelées par l'utilisateur. Dans ces réseaux, l'accent est donc mis plus sur la qualité de service (*QoS*) que sur la faible consommation. Dans les RCsF mobiles qui peuvent s'apparenter aux MANET, les protocoles sont souvent conçus spécifiquement pour une application, afin d'optimiser notamment l'efficacité en terme de consommation d'énergie.

La fonction principale d'un nœud d'un RCsF est de détecter des événements, d'effectuer quelques traitements rapides sur les données collectées et de transmettre ses résultats. La consommation d'énergie d'un capteur peut donc être classifiée selon trois axes, quelque soit le contexte de mobilité : la *capture*, le *traitement* des données et la *communication*. Celles-ci sont traitées indépendamment ci-dessous.

La durée de vie d'un capteur est liée à sa réserve d'énergie et à la façon dont celle-ci est utilisée [HCB00]. Mais, fréquemment, la disparition d'un nœud, dont l'énergie est épuisée, n'implique pas la mort du réseau dans son ensemble. Ainsi, certains algorithmes dans un contexte statique n'hésitent pas à sacrifier quelques nœuds en épuisant l'intégralité de leur réserve sur une période donnée pour augmenter significativement la durée de vie globale du RCsF [MVG+06]. Par contre, ces algorithmes doivent assurer que la perte de ces nœuds continue cependant de garantir la condition de connexité du RCsF (*cf.* 1.2.3). À l'inverse, en présence de mobilité, le plus souvent, un capteur est embarqué sur chaque unité mobile. La perte d'un capteur entraîne alors la disparition d'une des unités dans le système. Ainsi, ce type d'heuristique est souvent peu utilisé dans ce contexte.

Capture de données

Le module de capture et ses éléments varient fortement en fonction de l'application considérée. Par conséquent, leur consommation d'énergie dépend étroitement de celle-ci. Par exemple, pour une application médicale de surveillance de diabète, des capteurs sont placés sur des patients à risque et relèvent périodiquement le taux de glycémie. Une capture sporadique (*i.e.* une fréquence de capture relativement faible) peut souvent amener à une consommation plus réduite qu'avec un contrôle d'événements permanent. Cependant, ce dernier est nécessaire dans le cas d'un suivi de trajectoire d'objets. En sus de la fréquence de capture, la complexité de la détection d'événements joue également un rôle primordial dans la détermination de la dépense

énergétique. Par exemple, un fort bruit ambiant entraîne souvent des captures inexactes, et en augmentera donc la complexité.

Communication sans-fil

Des trois classes de consommation d'énergie, la plus gourmande est celle de la communication. Elle comprend à la fois la transmission et la réception de données. Il est également important de prendre en compte, non seulement la dépense énergétique en mode actif du transcepteur, mais également la consommation due à l'initialisation du circuit électronique de transception. Dans un contexte de RCsF statique, il est possible d'éteindre régulièrement le transcepteur afin d'économiser de l'énergie. Dans un contexte à forte mobilité, la mise en veille est plus difficile en raison des messages périodiques émis pour détecter continuellement son voisinage de communication direct. En effet, après chaque extinction du transcepteur, le temps de démarrage, de l'ordre de la centaine de micro-secondes entraîne un coût non-négligeable. Par conséquent, étant donné que le temps de transmission d'un message de petite taille est très court, l'alternance marche/arrêt n'est pas toujours efficace sur un RCsF mobile.

Traitement des données

Comparativement au module communication, la consommation issue du traitement des données est bien moindre. En se basant sur la théorie de la diffusion Rayleigh et de l'affaiblissement de l'onde en fonction de la distance, le coût énergétique pour transmettre 1 kilo-octet sur une distance de 100 mètres est approximativement le même que celui pour exécuter 3 millions d'instructions sur un processeur a 100 MIPS/W (millions d'instructions par seconde et par watt) [PK00]. Par conséquent, le traitement local des données est crucial pour réduire la consommation d'énergie d'un RCsF à routage multi-saut.

Dans un routage sur une infrastructure réseau fixe (capteurs immobiles), l'agrégation de messages et la construction d'arbres de diffusion nécessitent un traitement local plus important, mais dont l'économie d'énergie est non négligeable. À l'inverse, dans un contexte mobile, ce type de mécanisme n'est pas réalisable, mais l'agrégation reste possible par l'instauration d'un délai de retransmission. Durant ce délai, tous les messages reçus seront agrégés avant d'être transmis d'un bloc à la destination.

1.2.2 Tolérance aux défaillances

Dans un réseau de capteurs à routage multi-saut ou *ad hoc*, chaque nœud joue à la fois le rôle d'initiateur de message ainsi que celui de *routeur* (ou *retransmetteur*). Le dysfonctionnement de quelques nœuds peut générer des variations topologiques manifestes dans un contexte statique et entraîner alors une réorganisation du réseau.

La globalité de la tâche principale du RCsF déployé ne doit pas être affectée par la défaillance ou la perte temporaire de certains capteurs du réseau, causés par un dommage physique, un manque d'énergie ou des interférences. La tolérance aux défaillances est définie par la capacité de maintenir les fonctionnalités du RCsF sans interruption dues à des défauts de nœuds [SSJ01], qu'ils soient mobiles ou statiques. La conception d'algorithmes ou de protocoles pour ces réseaux doit évaluer le degré de tolérance requis. En effet, si l'environnement de

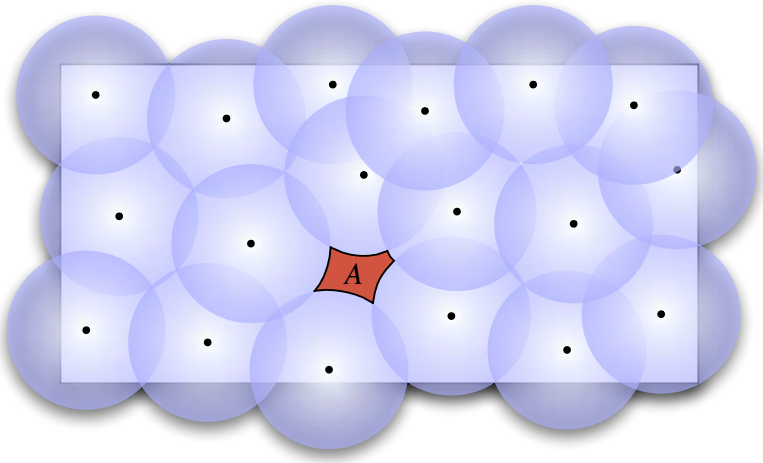

FIG. 1.5 – Surface de couverture d'un RCsF.

déploiement d'un RCsF est à faible interférence, les solutions développées peuvent être plus permissives. Deux applications foncièrement opposées en terme d'environnement de mesure auront deux politiques opposées de tolérance aux défaillances : par exemple, la mesure du taux d'hygrométrie dans une maison particulière et la surveillance de soldats sur un champs de bataille. Alors que le premier peut souffrir d'une perte de qualité temporaire, bien que le bruit de l'environnement ne varie pas subitement, le second s'appuie sur des données captées critiques, ainsi que sur des capteurs qui peuvent être anéantis brusquement par le milieu hostile de déploiement.

1.2.3 Couverture et connexité

Dans un RCsF, chaque nœud perçoit une vision limitée et purement locale de son environnement, relative uniquement à sa *zone de perception*. Cette dernière doit être mise en relation avec la zone de transmission de ce capteur, afin de déterminer la densité optimale de capteurs à déployer.

En effet, considérons une zone déterminée à surveiller par un RCsF statique. Sur celle-ci, les nœuds doivent pouvoir s'échanger des données de proche en proche. Pour cela, la proximité des nœuds doivent être suffisante les uns des autres pour assurer la connexité du réseau. De même, les zones de perception des capteurs doivent couvrir l'ensemble de la zone à surveiller. La couverture d'un vaste espace est donc composée de l'union de nombreuses couvertures de petites tailles comme illustré en figure 1.5. Sur cette figure sont représentées les zones de perception de tous les capteurs participants. La zone A étant située hors de toute zone de perception, tout événement intervenant dans cette zone ne sera donc ni observé ni pris en compte. C'est typiquement le cas de figure à éviter lors du déploiement des nœuds pour une application de surveillance.

De surcroît, la couverture en terme de surface ne suffit pas à surveiller efficacement une

zone donnée. Il est nécessaire de prendre en compte la longévité de couverture et par conséquent d'administrer correctement les extinctions de nœuds surveillant une zone déjà observée intégralement par d'autres capteurs.

Dans le cadre des RCsF mobiles, la connexité peut ne pas être assurée continuellement, et le réseau peut souffrir de partitionnement temporaire. Il est ainsi nécessaire de mettre en œuvre des solutions, non plus fondées sur la répartition spatiale des capteurs, mais également sur la connexité au cours du temps. Pour cela, l'étude des modèles de mobilité est un axe de recherche primordial.

1.3 Différents aspects de ces réseaux

Dans ce dernier paragraphe, nous présentons des caractéristiques qui ne sont pas nécessairement communes à la conception de tous les RCsF mais qui représentent des défis déterminants [ASSC02a, WP02]. À titre d'exemple, nous considérons ici l'impact de ces facteurs pour le problème du routage dans les RCsF, et présentons succinctement chaque objectif à atteindre dans ce contexte.

1.3.1 Déploiement des nœuds

C'est un facteur dépendant de l'application qui affecte grandement les protocoles de routage dans un RCsF statique. Le déploiement peut être déterminé ou aléatoire. Dans la première stratégie, les capteurs sont placés manuellement et les données peuvent donc être acheminées via des chemins prédéterminés. En revanche, avec une approche aléatoire, les capteurs sont éparpillés au hasard (*e.g.* lâchés d'un avion). Une approche auto-organisante est donc indispensable au bon fonctionnement du routage.

1.3.2 Modèle de données

La capture d'information et la couverture des données requises dépendent essentiellement de l'application. Elles peuvent être classifiées selon différents modèles : en fonction du temps (surveillance périodique), des événements (réaction à l'occurrence d'un évènement particulier), des requêtes (réponse à la demande d'une station de base[3]) ou de manière hybride (combinaisons des précédentes approches) [BGS01, YG02]. Dans un contexte statique, un mécanisme d'alerte est souvent mis en place. À l'inverse, en présence de mobilité, un routage ad-hoc est nécessaire afin d'agréger *rapidement* les messages vers une station de base.

1.3.3 Hétérogénéité

Dans de nombreuses études, tous les capteurs d'une application sont considérés homogènes (*i.e.* même capacité de calcul, de communication et d'énergie). Néanmoins, en fonction de l'application, certains capteurs peuvent avoir des rôles différents, ou des périphériques spécifiques, générant une architecture hétérogène en terme de type de nœuds et de connectivité.

[3]Une *station de base* est une entité, le plus souvent considérée comme ayant des ressources infinies, permettant de collecter et diffuser des données sur le RCsF.

Par exemple, les réseaux de capteurs vidéo contiennent divers types d'équipements tels que des microphones, des caméras vidéo, des sources énergétiques supplémentaires, *etc.* De même, dans un RCsF hiérarchique, certains capteurs sont déclarés « meneurs » de leur groupe. Le routage vers les stations de base est alors traité par ces derniers.

1.3.4 Critères d'évaluation

Dans diverses applications, les données doivent être transmises dans une certaine plage de temps, après quoi, elles deviennent caduques. Pourtant, dans la plupart des applications, la durée de vie du réseau est favorisée, au détriment de la qualité de fréquence d'émission des données. Les protocoles de routage assurant une qualité de service et prenant en compte la gestion de l'énergie, représentent un défi nouveau et stimulant, tant dans un contexte statique que mobile, où les réseaux à tolérance de délais (*cf.* paragraphe 1.2.1) sont en pleine expansion.

1.3.5 Agrégation des données

Un nombre important de capteurs peut générer de la redondance dans les mesures effectuées. Comme il a été abordé à plusieurs reprises, l'agrégation de paquets similaires en provenance de différents nœuds permet de réduire le nombre de transmissions (*e.g.* suppression de duplicata, minimum, moyenne, maximum, ...) Un traitement du signal peut également être envisagé dans un but d'agrégation, par fusion de données (*e.g.* conformation de faisceaux, ou *beamforming*). Il permet une économie d'énergie (*cf.* paragraphe 1.2.1) et une robustesse du mécanisme de routage.

1.4 Ouverture et positionnement

Domaine émergeant, aux frontières de l'étude des systèmes informatiques, des réseaux numériques et du traitement du signal, l'attrait pour les réseaux de capteurs sans fil s'accroît considérablement. Dans ce chapitre, nous avons présenté les propriétés communes à ces réseaux, tout en mettant en évidence leurs limitations (physiques, technologiques, ...). Nous avons présenté différents défis à la fois dans un contexte de RCsF statique et mobile.

Dans la suite de ce document, nous présentons en détails les différentes contributions de l'auteur pour l'aide à la conception d'algorithmes et de protocoles dans les RCsF large-échelle. En effet, nous sommes convaincus que, dorénavant, le passage à l'échelle est une caractéristique clé dans le déploiement d'applications. De même, nous sommes également persuadés que la conception de systèmes sur les RCsF, qu'ils soient mobiles ou statiques, doit suivre indubitablement une approche collaborative et auto-organisante.

Toutes les contributions présentées dans le reste de ce document soutiennent cette thèse.

Première partie

Réseaux de capteurs statiques

CHAPITRE 2

Suivi et identification de trajectoires sur un réseau de capteurs binaires

Nous séparons nos contributions concernant les RCsF statiques et mobiles. Comme nous l'avons introduit au chapitre précédent, à l'origine, l'utilisation des RCsF était centrée sur les applications de surveillance. Puis, la diversification des applications potentielles ont permis de montrer la puissance et les interfaces nécessaires aux RCsF.

Dans cette partie, nous présentons deux contributions suivant cette évolution des réseaux de capteurs statiques. En premier lieu, dans ce chapitre, nous présentons un problème de suivi d'objets mouvants anonymes ainsi que diverses caractérisations et solutions de ce problème, dans différents contextes d'applications. En second lieu, nous présentons une infrastructure générique à toutes les applications utilisant un RCsF statique.

Auparavant, et avant de présenter le formalisme du problème sus-cité, nous introduisons le contexte ainsi que le modèle de l'environnement considéré. Enfin, nous proposons des caractérisations et solutions possibles dudit problème dans un contexte d'étude *(i) a priori*, *(ii)* en *temps réel* et *(iii) a posteriori*, de la surveillance à proprement dîte.

2.1 Introduction

Contexte «*Suivre les mouvements d'objets anonymes et indiscernables, et associer à chaque trajectoire d'un objet un identifiant unique*» est un problème de base dans de nombreux contextes applicatifs tels que la surveillance [OS05, CRZ04], le sauvetage [KGSL05], l'étude de trafic [BIL+07], *etc.* Sa résolution est pourtant complexe et ce, pour deux raisons intrinsèques. Dans un premier cas, considérons l'aspect matériel du système de détection. La faillibilité des RCsF (*e.g.* [ABC+03]) et la complexité du calcul d'un système de localisation (*e.g.* [AGY04]) renforcent la difficulté de suivre un objet mouvant, en raison de fausses détections, ou d'observations manquantes par exemple.

En second lieu, considérons alors un matériel de détection infaillible. Malgré cela, même si nous considérons des capteurs idéaux permettant d'éliminer ces effets, associer un identi-

fiant unique à une trajectoire devant refléter exactement celle d'un unique objet réel reste un problème, dû à la fusion potentielle de chemins. En d'autres termes, deux trajectoires peuvent devenir si proches l'une de l'autre qu'elles en deviennent indiscernables, et donc impossibles à identifier de manière *déterministe* une fois séparées à nouveau. Dans la suite, cette association bijective entre les trajectoires observées et réelles des objets sera identifiée comme le problème de *suivi d'objets multiples et leur identification* soit *MOTI* (signifiant *Multiple Objects Tracking and Identification*).

Nous cherchons donc à résoudre MOTI d'un point de vue déterministe. Ainsi, nous considérons dans cette étude un réseau de capteurs binaires parfaits (*i.e.* chaque capteur retourne uniquement une valeur booléenne indiquant si un objet est situé dans sa zone de capture, ou non). Le choix de cibler l'étude sur de telles entités, simples et minimalistes, est motivé par la réflexion suivante : soit les capteurs sont capables d'identifier précisément quel objet se trouve dans leur zone de capture, et dans ce cas la résolution de MOTI est triviale, soit ils n'en sont pas capables et dans ce cas, quelle que soit la capacité de ces capteurs (telles que la puissance de calcul, les ressources réseaux, la sensibilité de capture, *etc.*), l'information supplémentaire ne permet pas d'éviter la confusion de trajectoire [LLR+03].

Enfin, le RCsF est représenté par un graphe connexe épars, dénommé *graphe de connectivité des chemins* [OS05] (GCC), dans lequel (1) chaque sommet représente un capteur binaire et (2) il existe un arc entre deux capteurs u et v seulement si un objet peut passer de la zone de réception de u à celle de v sans activer d'autres capteurs. Il est ainsi possible de modéliser l'ensemble des mouvement possibles des objets. Ainsi, outre les limitations intrinsèques de ce modèle[1], il schématise parfaitement un grand nombre d'applications d'intérieur telles que la traque de mouvements de personnes dans un bâtiment, ainsi que d'extérieur telles que le suivi de gondoles sur les canaux de la ville de Venise [BIL+07]. Pour chacun de ces deux exemples, des capteurs sont placés à des positions stratégiques de l'environnement à surveiller (*e.g.* à chaque croisements de routes), et les arcs représentent les couloirs ou les canaux existants. Ainsi, le GCC est donc une représentation suffisamment réaliste pour ne pas être permissive.

Contributions Dans ce contexte, nous montrons qu'il est impossible de résoudre le problème MOTI dans un graphe quelconque, et ceci, même avec l'aide d'un observateur omniscient possédant la connaissance complète de l'état du système. Nous montrons que l'impossibilité de MOTI provient d'un problème topologique du GCC.

Ensuite, nous présentons quelques restrictions qu'il est possible d'imposer *a priori* à la fois sur le graphe, sur les mouvements des objets ou sur les deux, afin de rendre MOTI toujours résoluble (c'est le cas d'un GCC acyclique par exemple). Plus spécifiquement, si le GCC contient des cycles de longueurs supérieures ou égales à ℓ, il est toujours possible de résoudre MOTI si le nombre maximum d'objets en mouvements simultanés est strictement plus petit que $\lceil \frac{\ell}{2} \rceil$.

Enfin, en fonction de ces résultats, nous proposons diverses approches centralisées, puis réparties, afin d'assurer la réalisation de l'identification et du suivi d'objets multiples, en *temps réel* ainsi qu'*a posteriori*. Pour cela, nous formalisons le problème de manière globale, puis locale, et introduisons plusieurs algorithmes résolvant MOTI sous des conditions précises.

[1] Ce modèle ne permet pas de représenter les intersections de zones de couvertures par exemple.

2.2 État de l'art succinct

Pister des objets mobiles à l'aide RCsF est un problème possédant un large spectre d'applications [ABC+03, HXTG04, SMK+07]. Cette thématique a été traitée dans la littérature sous de nombreux aspects. Néanmoins, la plupart de ces travaux se rapportent au problème de suivi correct d'un [ABC+03] ou plusieurs [SMK+07] objets, avec un réseau de capteurs binaires, caractérisé par le bruit environnant et les erreurs de détections sous différentes contraintes (*e.g.* niveaux de bruit, consommation d'énergie [GM04], capacité de calcul et/ou de communication limitées [LLR+03], *etc.*). Par exemple, considérons les travaux menés dans [SMK+07]. Singh *et al.* proposent un algorithme probabiliste permettant un résultat de suivi significatif par observation de l'état des capteurs au cours du temps.

Dans nos travaux, dans un but de généralité, nous empruntons une approche du problème déterministe par une analyse théorique, avec un cadre technologique idéal composé de capteurs *parfaits* (pas de bruit, d'interférence, d'erreur de détection, ou de limitation de communication). Ainsi, nous montrons qu'il est impossible d'associer de façon déterministe une trajectoire observée à des objets mouvants anonymes.

Comme introduit dans le paragraphe précédent, ce résultat est une conséquence de la possibilité de fusion puis scission de trajectoires durant la période d'observation [LLR+03]. Une fois que deux trajectoires ont fusionné, dû à la forte proximité de deux objets, il est impossible de maintenir de façon déterministe l'identité de chacun des objets dès l'instant où leurs trajectoires se séparent. À notre connaissance, il s'agit de la première étude fondamentale sur la résolubilité du problème générique.

De surcroît, dans un environnement au sein duquel plusieurs objets peuvent se déplacer librement, les fusions et scissions de trajectoires sont fréquentes et universelles. Toutefois, à l'instar de [OS05], nous limitons ici notre analyse à un environnement spécifique représenté par le GCC, sur lequel les mouvements des objets sont partiellement contraints.

2.3 Modélisation du système

Nous considérons un système composé d'un ensemble d'objets génériques, se déplaçant dans un environnement où des capteurs peuvent détecter leur présence. Un tel environnement est donc modélisé par le graphe de connectivité des chemins (GCC) $G(V, E)$ défini formellement par l'ensemble des sommets V (qui représente des capteurs binaires), et pour chaque arc $e_{i,j} \in E$ reliant deux sommets $v_i, v_j \in V$, $e_{i,j}$ représente la possibilité pour un objet de se déplacer d'une position détectée par v_i à une autre détectée par v_j, sans activer aucun autre capteur binaire. Nous notons $\mathbb{G}$ l'ensemble de ces graphes. De plus, nous considérons le GCC comme non-orienté : les mouvements sont toujours possibles dans les deux directions. Enfin, nous supposons qu'il existe au plus un chemin direct pour se déplacer d'une position à une autre sans activer d'autres capteurs : *i.e.* au plus un arc peut relier deux mêmes sommets du GCC. Pour chacun des sommets v_i, il existe dans E un arc propre $e_{i,i}$ représentant la possibilité pour un objet de rester à la même position.

2.3.1 Localisations et mouvements

L'ensemble des objets $\{o_1, \ldots, o_x\}$ est dénoté $\mathbb{O}$. Le temps est représenté comme une suite discrète et infinie $T = \langle t_0, t_1, \ldots \rangle$ à partir de t_0. À tout instant t, chaque objet o occupe une position représentée par un sommet $v_i \in V$. La position d'un objet o au temps t est donnée par la fonction $loc : T \times \mathbb{O} \to V$, qui retourne le sommet v_i correspondant. Les seuls déplacements autorisés pour les objets sont les arcs de E (un objet situé en v_i ne peut passer en v_j que s'il existe un arc $e_{i,j} \in E$ – Nous considérons que $e_{i,j} = e_{j,i}$). L'ensemble des destinations possibles à partir d'un sommet spécifique est donné par la fonction[2] $adj : \mathbb{G} \times V \to \mathcal{P}(V)$. Le mouvement d'un objet o entre deux temps consécutifs t et $t+1$ est donnée par la fonction $mov : T \times \mathbb{O} \to E$, laquelle retourne l'arc $e_{i,j}$ visité durant le mouvement donné, pour lequel $v_i = loc_t(o)$, $v_j = loc_{t+1}(o)$ et $v_j \in adj_G(v_i)$.

Nous considérons également les deux hypothèses suivantes : $\forall t \in T$, $\forall o_i, o_j \in \mathbb{O}$,

(i) $o_i \neq o_j \Rightarrow loc_t(o_i) \neq loc_t(o_j)$ *i.e.* deux objets distincts ne peuvent se trouver à la même position au même moment ;

(ii) $o_i \neq o_j \Rightarrow mov_t(o_i) \neq mov_t(o_j)$ *i.e.* deux objets distincts ne peuvent se déplacer sur le même arc en même temps.

Sans ces deux contraintes, le problème MOTI est trivialement impossible. En effet, si deux objets peuvent se situer sur le même sommet au même moment, il sera impossible de connaître de manière déterministe lequel de ces deux objets a quitté le sommet le premier. De même, si la contrainte *(ii)* n'est pas vérifiée, il sera impossible de déterminer si les objets situés respectivement en $loc_t(o_i)$ et $loc_t(o_j)$ ont échangé leur place, ou s'ils sont restés au même endroit au temps suivant.

Chaque objet $o \in \mathbb{O}$ se déplace dans le système en suivant une trajectoire. Cette trajectoire, décrite entre les temps $t_i, t_j \in T$ (pour $j > i$), est définie par $P_{t_i,t_j,o} = \langle v^{t_i}, \ldots, v^{t_j} \rangle$ où $v^{t_k} = loc_{t_k}(o), k \in [\![i, j]\!]$. Pour t_i et t_j donnés, la *trajectoire globale* est définie comme l'ensemble des trajectoires de tous les objets de $\mathbb{O}$. Celle-ci est notée $P_{t_i,t_j} = \left\{P_{t_i,t_j,o}\right\}_{o \in \mathbb{O}} \in \mathbb{P}_{t_i,t_j}$, ou plus simplement P si le contexte ne traduit pas d'ambiguïté sur l'intervalle de temps considéré. Toute trajectoire globale P_{t_i,t_j} appartient à $\mathbb{P}_{t_i,t_j}$, l'ensemble de toutes les trajectoires possibles de x objets dans le GCC G considéré, durant la période $[\![t_i, t_j]\!]$ donnée.

2.3.2 État du système

L'*état du système* au temps t, noté S_t, est décrit comme l'état de chaque capteur (présence ou non d'un objet) à cet instant. Il est représenté par un vecteur de valeurs booléennes, associées à chaque sommet du GCC :

Définition 2.1 *Vecteur d'état Le* vecteur d'état *au temps t, noté S_t, est un vecteur de taille $|V|$ tel que :*

$$\forall v \in V, S_t[v] = \begin{cases} 1 & \textit{si } \exists\, o \in \mathbb{O} : loc_t(o) = v \\ 0 & \textit{sinon} \end{cases} \qquad \diamond$$

Par la suite, nous notons $\mathbb{S}$ l'ensemble de tous les vecteurs d'état possibles pour x objets dans un GCC G donné. Trivialement, le cardinal de $\mathbb{S}$ est égal à $\binom{|V|}{x}$.

[2] $\mathcal{P}(\cdot)$ représente l'ensemble des parties d'un ensemble donné.

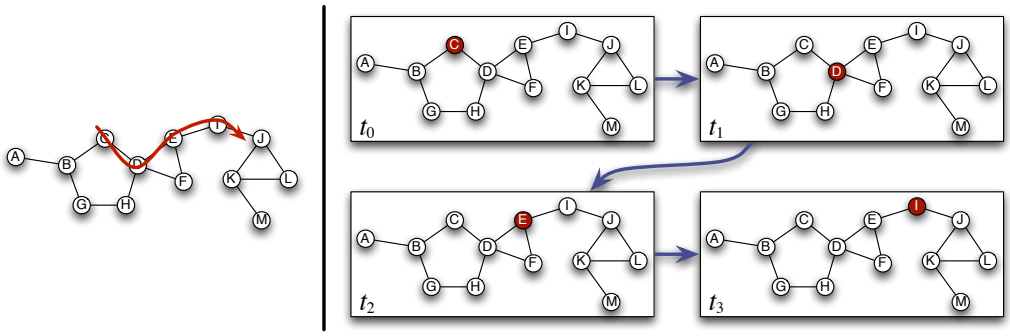

FIG. 2.1 – Illustration d'une trajectoire réelle et observée d'un objet o
Durant la période de temps $[\![t_0, t_3]\!]$*, la trajectoire réelle est* $P_{t_0,t_3,o} = [C, D, E, I]$.
Le but de MOTI est d'extrapoler $\overline{P}_{t_0,t_3,o}$ *par l'observation de sorte que* $P_{t_0,t_3,o} = \overline{P}_{t_0,t_3,o}$

2.3.3 L'observateur

Le système possède un observateur capable de lire, à tout temps t, le vecteur d'état S_t. Le but de cet observateur est d'identifier les objets et de suivre leur trajectoire à travers le temps. Étant donné le vecteur d'état S_t, l'observateur utilise une fonction tag qui lui permet d'associer un identifiant unique $\overline{o} \in \overline{\mathbb{O}}$ à chaque sommet v occupé (*i.e.* $S_t[v] = 1$). Pour tous les autres sommets (*i.e.* $\forall v' \in V, S_t[v'] = 0$) est associé une valeur prédéfinie $\perp$. La seule contrainte est que deux sommets ne peuvent partager le même identifiant (à l'exception de $\perp$). Par exemple, si $S_t = [1, 0, 0, 1, 0, 1, 1, 0, 0]$ et $\overline{\mathbb{O}} \subset \mathbb{N}$, il est possible d'avoir $tag(S_t) = [3, \perp, \perp, 2, \perp, 1, 4, \perp, \perp]$.

Chaque identifiant représente un objet, considéré comme unique par l'observateur. Par souci de commodité, nous introduisons la fonction $\overline{loc} : T \times \overline{\mathbb{O}} \rightarrow V$ qui associe le sommet v correspondant à l'étiquette $\overline{o} \in \overline{\mathbb{O}}$ associé par la fonction tag au temps $t \in T$.

Le vecteur contenant la liste des sommets consécutifs au cours du temps, pour un identifiant unique donné, correspond donc à une *trajectoire observée* : $\overline{P}_{t_i,t_j,o} = \langle v^{t_i}, \ldots, v^{t_j} \rangle$ où $v^{t_k} = \overline{loc}_{t_k}(\overline{o}), k \in [\![i, j]\!]$. De la même façon, nous pouvons définir la *trajectoire globale observée* par $\overline{P}_{t_i,t_j} = \left\{\overline{P}_{t_i,t_j,\overline{o}}\right\}_{\overline{o} \in \overline{\mathbb{O}}}$. Celle-ci correspond donc à l'ensemble de toutes les trajectoires perçues et extrapolées par l'observateur. Trivialement, les domaines des trajectoires réelles et observées sont identiques : $\overline{P}_{t_i,t_j}$ appartient à $\mathbb{P}_{t_i,t_j}$.

2.4 Identification d'objets et suivi de trajectoire

Nous montrons dans ce paragraphe que même avec une connaissance globale constante de l'état du système, il est impossible de déterminer de manière infaillible l'ensemble des trajectoires d'objets anonymes.

2.4.1 Le problème MOTI

Considérons un intervalle de temps $[\![t_i, t_j]\!]$. Le problème de *Suivi d'Objets Multiples et leur Identification* (*MOTI*) correspond à la définition d'une fonction tag permettant d'identifier correctement toute trajectoire réelle. Plus formellement, MOTI est résolu si :

$$\forall o \in \mathbb{O},\ \exists \overline{o} \in \overline{\mathbb{O}} : P_{t_i,t_j,o} = \overline{P}_{t_i,t_j,\overline{o}}. \tag{2.1}$$

D'un point de vue global, nous pouvons donc définir MOTI selon la condition suivante :

$$P_{t_i,t_j} = \overline{P}_{t_i,t_j}. \tag{2.2}$$

Ceci signifie que l'ensemble des trajectoires observées est identique à celui des trajectoires réelles (*cf.* figure 2.1). Étant donné que ces trajectoires (aussi bien réelles qu'observées) sont directement apparentées à la localisation des différents objets mouvants, nous pouvons définir le problème MOTI à une granularité plus fine par l'expression suivante :

$$\text{MOTI est résoluble } \textit{ssi } \forall o \in \mathbb{O},\ \exists \overline{o} \in \overline{\mathbb{O}},\ \forall t \in [\![t_i, t_j]\!] : \overline{loc}_t(\overline{o}) = loc_t(o)$$

La difficulté de ce problème provient du fait qu'il existe des situations pour lesquelles l'observateur peut confondre au moins deux trajectoires, et ainsi rendre un résultat erroné.

2.4.2 Un résultat d'impossibilité

Les deux théorèmes présentés dans ce paragraphe permettent de conclure à l'impossibilité de résolution de MOTI dans un contexte général. En premier lieu, nous caractérisons les cas pour lesquels la résolubilité de MOTI est impossible. Les cas d'impossibilité correspondent aux cas où il existe deux trajectoires réelles distinctes pour lesquels l'observateur retourne le même ensemble de trajectoires observées. Dans un second temps, nous montrerons qu'il existe au moins un cas pour lequel la condition du théorème suivant est vérifiée.

Theorème 2.1 (Insolubilité de MOTI) *Soient un intervalle de temps* $[\![t_i, t_j]\!]$*, un GCC* $G(V, E)$*, un ensemble* $\mathbb{O}$ *de* x *objets et une fonction* tag. *MOTI est insoluble* ssi [3]*,*

$$\exists P, P' \in \mathbb{P}_{t_i,t_j} :\ P \neq P' \wedge \overline{P} = \overline{P'} \tag{2.3}$$

où $\overline{P}$ *et* $\overline{P'}$ *sont obtenus par l'observateur en utilisant la fonction* tag *à partir des trajectoires réelles* P *et* P'*.*

Preuve – Considérons un intervalle de temps $[\![t_i, t_j]\!]$, un GCC $G(V, E)$ ainsi qu'un ensemble $\mathbb{O}$ de x objets.

– Préalablement, prouvons par l'absurde que si la condition 2.3 est respectée alors MOTI est insoluble.

Supposons que (1) $\exists P, P' \in \mathbb{P}_{t_i,t_j} :\ P \neq P' \wedge \overline{P} = \overline{P'}$ et que (2) MOTI est résoluble. Étant donné cette dernière condition, nous avons $P = \overline{P}$ ainsi que $P' = \overline{P'}$ (*cf.* la définition du problème MOTI donnée au paragraphe 2.4.1). Ainsi, en utilisant la condition (1), nous avons $P \neq P'$ et $\overline{P} = \overline{P'}$,

[3] Dans le reste de ce document, nous utilisons indistinctement « *si et seulement si* » et son abréviation *ssi*.

ce qui conduit à une contradiction.

– Inversement, prouvons à présent par la contraposée (*i.e. modus tollens*) que si MOTI est insoluble, alors la condition 2.3 est toujours vérifiée.

Soit l'hypothèse suivante : $\forall P, P' \in \mathbb{P}_{t_i,t_j} : \; P \neq P' \Longrightarrow \overline{P} \neq \overline{P'}$ ou autrement dit $P = P' \vee \overline{P} \neq \overline{P'}$. Soit $map : \mathbb{P}_{t_i,t_j} \rightarrow \mathbb{P}_{t_i,t_j}$ la fonction qui associe la trajectoire réelle à la trajectoire observée, par l'utilisation de la fonction tag donnée. Toute fonction map bijective vérifie l'expression suivante :

$$\forall P, P' \in \mathbb{P}_{t_i,t_j} : \; P \neq P' \Longrightarrow map(P) \neq map(P') \Longrightarrow \overline{P} \neq \overline{P'}$$

où $map(\cdot)$ correspond aux trajectoires observées. Supposons que map représente la fonction identité. Alors, nous avons : $\forall P \in \mathbb{P}_{t_i,t_j} : \; P = map(P) = \overline{P}$. Ainsi, selon la définition donnée au paragraphe 2.4.1, MOTI est résoluble. Donc, la contraposition de ce raisonnement nous donne :

$$\text{MOTI est insoluble} \Longrightarrow \exists P, P' \in \mathbb{P}_{t_i,t_j} : \; P \neq P' \wedge \overline{P} = \overline{P'}.$$

La combinaison des deux points précédents démontre l'équivalence du théorème 2.1, *i.e.* une caractérisation de l'insolubilité de MOTI. ■

Corollaire 2.1 (Résolubilité de MOTI) *Soient un intervalle de temps* $[\![t_i, t_j]\!]$*, un GCC* $G(V, E)$*, un ensemble* $\mathbb{O}$ *de* x *objets et une fonction* tag*. MOTI est résoluble* ssi*,*

$$\forall P, P' \in \mathbb{P}_{t_i,t_j} : \; P \neq P' \Longrightarrow \overline{P} \neq \overline{P'}.$$

Par conséquent, il existe une impossibilité de résoudre MOTI *si, et seulement si*, il existe au moins deux trajectoires globales respectant la condition 2.3 du théorème 2.1. Ainsi, nous pouvons établir le résultat suivant :

Theorème 2.2 (Impossibilié de MOTI) *Étant donné le modèle du système donnée au paragraphe 2.3, MOTI est* impossible *à résoudre.*

Preuve – Considérons le GCC G à 4 sommets, reliés par une boucle de 4 arcs, illustré en figure 2.2.a. De même, considérons deux objets se déplaçant dans ce graphe G ainsi qu'un intervalle constitué uniquement de deux temps consécutifs $[\![t_0, t_1]\!]$ avec $t_1 = t_0 + 1$.

Examinons deux trajectoires globales $P, P' \in \mathbb{P}_{t_0,t_1}$, présentées respectivement en figure 2.2.b et 2.2.c. Celles-ci sont définies comme suit :

$$\left\{ \begin{array}{lcl} P & = & \left\{ \begin{array}{c} \left[\begin{array}{c} A, B \\ D, C \end{array} \right] \end{array} \right\} \quad (2.2.b) \\ P' & = & \left\{ \begin{array}{c} \left[\begin{array}{c} A, C \\ D, B \end{array} \right] \end{array} \right\} \quad (2.2.c) \end{array} \right.$$

Trivialement, nous avons $P \neq P'$.

Cependant, conformément aux caractéristiques du système et aux capacités de l'observateur, ce dernier dispose exactement des mêmes informations, présentées en figure 2.2.d, que ce soit pour P ou P'. Supposons que t_0 est le temps initial d'observation. Alors, l'observateur n'a pas d'autres informations disponibles sur le système que les deux vecteurs d'états suivants, lesquels sont identiques pour P et P' :

$$S_{t_0} = \left[\begin{array}{c} 1, 0, 0, 1 \end{array} \right] \text{ et } S_{t_1} = \left[\begin{array}{c} 0, 1, 1, 0 \end{array} \right]$$

Étant donné que la relation tag est une application (*i.e.* une image unique pour un antécédent donné), il existe un étiquetage unique pour un état du système, extrapolé à partir du vecteur d'état courant et

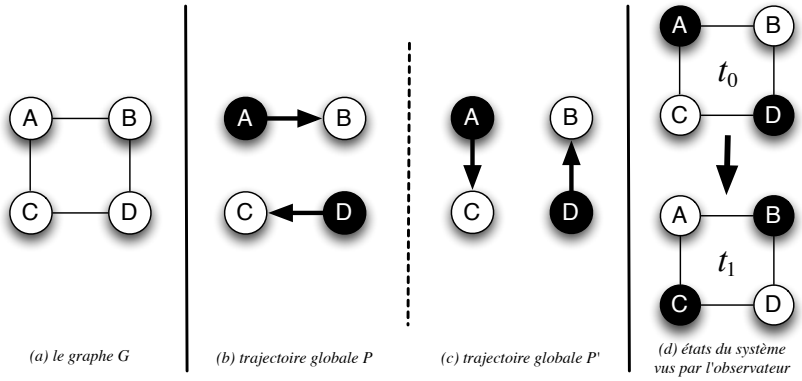

FIG. 2.2 – Concrétisation de l'impossibilité de MOTI

Simple exemple pour lequel l'observateur ne peut pas déduire précisément la trajectoire des objets : Pour le GCC G, présenté en (a)*, avec deux objets, deux trajectoires globales P et P' peuvent advenir, présentées respectivement en* (b) *et* (c)*. L'observateur n'est pas capable de les distinguer en connaissance uniquement des informations présentées en* (d)*.*

du précédent étiquetage du système. Ainsi, supposons que :

$$tag(S_{t_0}) = \left[\ 1, \perp, \perp, 2\ \right].$$

Étant donnés $tag(S_{t_0})$ et S_{t_1}, seulement deux possibilités sont envisageables pour $tag(S_{t_1})$:

$$(1)\ tag(S_{t_1}) = \left[\ \perp, 1, 2, \perp\ \right] \text{ ou } (2)\ tag(S_{t_1}) = \left[\ \perp, 2, 1, \perp\ \right].$$

Pour chacun de ces cas, $tag(S_{t_1})$ est identique pour l'observation de P et de P'. Donc, pour une fonction tag donnée, $\overline{P} = \overline{P'}$. D'où, dû au théorème 2.1, MOTI est insoluble. ■

Pour une meilleure compréhension de la dernière étape de la preuve, considérons plus précisément chacun des deux cas de figure :

Cas (1) Dans ce cas, la trajectoire globale observée, extrapolée par l'observateur, est la suivante :

$$\overline{P} = \overline{P'} = \left\{ \begin{bmatrix} A, B \\ D, C \end{bmatrix} \right\}.$$

Ainsi, nous avons $P = \overline{P} = \overline{P'} \neq P'$ et MOTI n'est pas résolu.

Cas (2) Par symétrie, dans ce cas, la trajectoire globale observée, extrapolée par l'observateur, est la suivante :

$$\overline{P} = \overline{P'} = \left\{ \begin{bmatrix} A, C \\ D, B \end{bmatrix} \right\}.$$

Donc, nous avons $P' = \overline{P'} = \overline{P} \neq P$ et MOTI n'est pas résolu.

En définitive, il existe au moins un cas pour lequel MOTI n'est pas résoluble. D'où, l'existence de l'impossibilité de résolution et la preuve du théorème 2.2.

2.5 Conditions de résolubilité de MOTI

Le problème MOTI étant insoluble dans le cas générique, il est intéressant de caractériser les cas de résolubilité. L'objectif de ce paragraphe est d'imposer des contraintes *a priori* pour que les trajectoires puissent être suivies infailliblement. Cependant, identifier ces contraintes n'est pas trivial.

Avant d'entrer plus précisément dans les détails permettant de contraindre les caractéristiques du système afin de rendre MOTI résoluble, il est nécessaire d'introduire d'autres notations.

2.5.1 Caractérisation d'infaillibilité

Soient un objet o et sa trajectoire décrite au cours du temps dans le GCC. Nous pouvons identifier ses mouvements unitaires comme suit.

Définition 2.2 (Mouvement unitaire) *Soit $o \in \mathbb{O}$ un objet se déplaçant dans le système représenté par un GCC $G(V, E)$. Pour chaque temps $t \in T$, son* mouvement unitaire *est défini par $m_{t,o} = P_{t,t+1,o}$.* ◇

Définition 2.3 (Mouvement unitaire global) *Soit un système représenté par le GCC $G(V, E)$ sur lequel les objets de l'ensemble $\mathbb{O}$ se déplacent. Pour chaque temps $t \in T$, le* mouvement unitaire gobal *est dénoté : $M_t = P_{t,t+1}$.* ◇

Le mouvement unitaire global est défini, à chaque instant t, comme l'ensemble des mouvements unitaires de tous les objets. Par conséquent, il représente comment le système «évolue» immédiatement après l'instant t donné.

Si nous considérons l'état du système à un temps t spécifique, il est possible d'identifier toutes les potentialités de mouvements unitaires que les objets peuvent être amenés à effectuer. Chaque combinaison possible de ces mouvements correspond à un mouvement unitaire global différent. De même, tous ces mouvements globaux peuvent être définis à partir de la position des objets dans le GCC (*i.e.* étant donné un vecteur d'état, il est possible d'extrapoler tous les mouvements possibles immédiatement après).

Par conséquent, les mouvements unitaires globaux ne sont pas nécessairement liés à une notion de temps. Ils peuvent être vus comme l'ensemble des mouvements unitaires *possibles* que les objets peuvent effectuer, à partir d'un état du système donné. Hors du contexte temporel, il est donc possible de définir le *graphe des états*. Celui-ci rassemble tous les états possibles du système (en terme de vecteurs d'état) ainsi que les mouvements unitaires globaux permettant de passer de l'un à l'autre.

Définition 2.4 (Graphe des états) *Soit $G(V, E)$ un GCC représentant l'environnement de déplacement de x objets dans $\mathbb{O}$. Le graphe des états correspondant est défini par $SG(\mathbb{S}, \mathbb{M})$, où $\mathbb{S}$ est l'ensemble de tous les vecteurs d'états possibles et $\mathbb{M}$ celui de tous les mouvements unitaires globaux possibles.* ◇

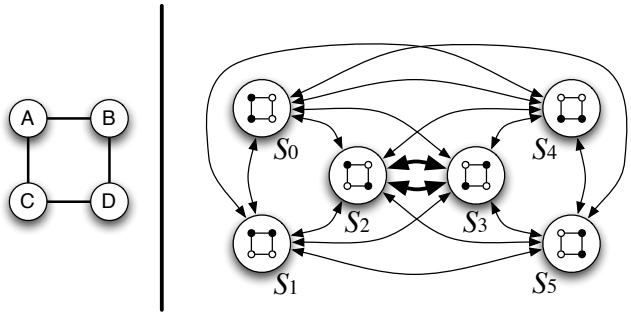

FIG. 2.3 – Graphe des états d'un système à 2 objets mouvants sur un GCC à 4 sommets.

La figure 2.3 présente un exemple de graphe des états, représenté sur la droite de la figure, dans le cas de 2 objets mouvants sur le graphe à quatre sommets introduit précédemment et rappellé sur la partie gauche de la figure. À présent, il est nécessaire d'introduire une notion de *faillibilité*. Nous pouvons définir quels sont les arcs et les sommets du graphe des états qui peuvent être considérés comme *faillibles*, par rapport à la résolubilité de MOTI.

Définition 2.5 (Mouvement faillible) *Soit un système représenté par $G(V, E)$ sur lequel les objets de $\mathbb{O}$ peuvent se déplacer, et le graphe des états $SG(\mathbb{S}, \mathbb{M})$ correspondant. Considérons deux états $S, S' \in \mathbb{S}$ tels qu'il existe un mouvement global $M \in \mathbb{M}$ reliant ces deux états dans SG : $S \xrightarrow{M} S'$. M est* faillible *ssi $\exists M' \in \mathbb{M}$ tel que $S \xrightarrow{M'} S'$ et $M \neq M'$.* ◇

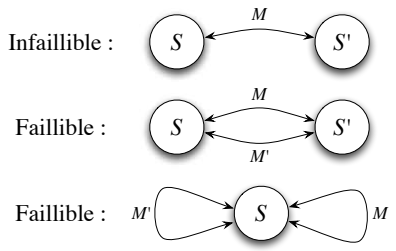

FIG. 2.4 – Faillibilité des états et des mouvements globaux.

Sont considérés comme *faillibles* tous les mouvements globaux reliant deux même états du système déjà reliés par un autre mouvement global différent. Cette notion de faillibilité permet de traduire les cas de figure pour lesquels l'observateur est incapable de distinguer quel mouvement s'est réellement produit, parmi les possibilités de mouvements faillibles reliant deux états consécutifs du système (les trajectoires ne sont extrapolées qu'en connaissance des états du système). La présence d'un mouvement faillible dans le graphe des états rend MOTI insoluble pour le système considéré.

De la même manière, nous pouvons définir la faillibilité d'un état du système.

Définition 2.6 (État faillible) *Soit un système représenté par $G(V, E)$ sur lequel les objets de $\mathbb{O}$ peuvent se déplacer, et le graphe des états $SG(\mathbb{S}, \mathbb{M})$ correspondant. Un état $S \in \mathbb{S}$ est* faillible *ssi $\exists M \in \mathbb{M}, \exists S' \in \mathbb{S}$ tels que $S \xrightarrow{M} S'$ et M est faillible.* ◇

La figure 2.4 présente les différents cas de figure de faillibilité des états et des mouvements. Considérons à nouveau le graphe des états représenté en figure 2.3, les états S_2 et S_3

Algorithme 2.1 : Construction du graphe des états

Données : G, un graphe ; x, le nombre d'objets mouvants ;
Résultat : $SG(\mathbb{S}, \mathbb{M})$, le graphe des états associé ;

1 $\mathbb{S} \leftarrow \{S \in \{0,1\}^{|V|} | \sum_{v \in V} S[v] = x\}$;
2 $\mathbb{M} \leftarrow \varnothing$;
3 **pour chaque** $S \in \mathbb{S}$ **faire**
4 **pour chaque** $S' \in \mathbb{S} \backslash \{S\}$ **faire**
5 **si** $\exists!(v, v') \in V^2$ *tel que* $S[v] = S'[v'] = 1$ *et* $S[v'] = S'[v] = 0$ **alors**
6 $\mathbb{M} \leftarrow \mathbb{M} \cup \{S \xrightarrow{\{[v-v']\}} S'\}$;
7 **pour chaque** $S \in \mathbb{S}$ **faire**
8 **pour chaque** $S' \in \mathbb{S} \backslash \{S\}$ **faire**
9 **pour chaque** $S'' \in \mathbb{S} \backslash \{S\}$ **faire**
10 **si** $(S' \rightarrow S \in \mathbb{M}) \wedge (S \rightarrow S'' \in \mathbb{M})$
11 **et** $(\forall S' \rightarrow S'' \in \mathbb{M},\ e_{S',S''} \neq e_{S',S} \cup e_{S,S''})$
12 **et** $\forall [v', v_1] \in e_{S',S},\ \forall [v_2, v''] \in e_{S,S''},\ v_1 \neq v_2$ **alors**
13 $\mathbb{M} \leftarrow \mathbb{M} \cup \{S' \xrightarrow{e_{S',S} \cup e_{S,S''}} S''\}$;
14 **si** G *est non-orienté* **alors**
15 fusionner les arcs symétriques dans $\mathbb{M}$;
16 **retourner** $(\mathbb{S}, \mathbb{M})$;

sont tous deux faillibles, en raison de l'existence des deux arcs distincts (deux mouvements globaux faillibles) les reliant (flèches épaisses). Tous les autres états (et, *a fortiori*, tous les autres mouvements globaux) sont *infaillibles*.

2.5.2 Un algorithme de génération du graphe des états SG

Dans ce paragraphe, nous proposons un algorithme de construction du graphe des états, fondé sur la fermeture transitive conditionnelle du graphe des états réduit aux mouvements isolés[4]. L'aspect conditionnel permet d'affirmer qu'aucun mouvement incohérent ne pourra être décelé dans le graphe des états généré par cet algorithme.

Cet algorithme se décompose en deux phases principales. La première permet de générer le graphe des états réduit aux mouvements isolés. Au préalable, l'ensemble des états est initialisé avec l'intégralité de tous les états possibles du système, et celui des mouvements globaux avec l'ensemble vide. Puis, à partir de la ligne 3, et jusqu'à la ligne 6, l'algorithme enrichit le graphe des états avec tous les mouvements isolés possibles. La seconde phase de l'algorithme adjoint au graphe précédent tous les mouvements globaux non-isolés possibles. De la ligne 7 à la ligne 13, inspiré du mécanisme utilisé dans l'algorithme de détermination de la fermeture transitive de Floyd-Warshall [Flo62, War62], trois boucles imbriquées calculent itérativement tous les mouvements possibles, avec un nombre quelconque de mouvements unitaires concomi-

[4]Un mouvement isolé signifie qu'au plus un unique objet parmi tous se déplace dans le mouvement global considéré.

tants[5]. Enfin, les dernières lignes (de la ligne 14 à 15) permettent de fusionner les mouvements symétriques (inversement identiques) dans le cadre d'un GCC non-orienté, et par conséquent, retourne un graphe des états non-orienté.

Preuve de correction et de terminaison de l'algorithme 2.1 – La terminaison de cet algorithme est triviale étant donné que celui-ci n'est composé que de boucles imbriquées de longueur finie. La complexité de cet algorithme est évidemment $O(|\mathbb{S}|^3)$ puisqu'il contient exactement trois boucles imbriquées sur l'ensemble $\mathbb{S}$.

La preuve de correction s'inspire fortement de celle de l'algorithme de Floyd-Warshall [Flo62, War62]. Ce dernier assure que tous les chemins possibles entre deux états apparaissent dans la fermeture transitive du graphe. La condition de la ligne 12 permet quant à elle de ne pas introduire de mouvement incohérent dans le graphe des états. Étant donné que les arcs dans SG sont générés à partir de l'intégralité des mouvements isolés et respectent la condition sus-citée, aucun mouvement global possible ne peut être ignoré. ■

2.5.3 Caractérisation de MOTI

À présent, nous sommes donc en mesure de générer, et donc d'exploiter, le graphe des états. À partir des définitions du paragraphe 2.5.1, il est possible de réviser le théorème 2.1 (page 35) et ainsi proposer deux définitions de la résolubilité de MOTI. En premier lieu, nous considérons les cas pour lesquels la trajectoire globale est connue (*e.g.* lors d'une analyse de l'évolution du système *a posteriori*).

Theorème 2.3 (P-résolubilité) *Soit $SG(\mathbb{S}, \mathbb{M})$ un graphe des états défini pour un GCC de $\mathbb{G}$, $G(V, E)$, contenant les objets mouvants de $\mathbb{O}$. Pour une trajectoire spécifique $P \in \mathbb{P}_{t_i,t_j}$ donnée : MOTI est résoluble ssi $\forall t \in [\![t_i, t_{j-1}]\!]$, M_t est infaillible.*

Preuve – Soit une trajectoire spécifique $P \in \mathbb{P}_{t_i,t_j}$.

– Commencons par prouver par la contraposée l'implication directe. Supposons que $\exists t \in [\![t_i, t_{j-1}]\!]$ tel que M_t est *faillible*. Notons $P^* = P_{t,t+1}$ la sous-trajectoire de P_{t_i,t_j} donné tel que $M_t = P^*$. De part la définition 2.5, il existe au moins un mouvement infaillible $M' \in \mathbb{M}$ qui permet de passer du même état initial S au même état final S', noté $M' = P'^*$. Trivialement, $P^* \neq P'^*$ mais ils partagent les mêmes vecteurs d'états initial et final S_t et S_{t+1}. Étant donné que la relation tag utilisé par l'observateur est une application, il existe une unique étiquette image pour ces états. Donc, nous avons :

$$\exists P^*, P'^* \in \mathbb{P}_{t,t+1} : P^* \neq P'^* \wedge \overline{P^*} = \overline{P'^*}.$$

Par conséquent, dû au théorème 2.1, MOTI est insoluble pour P^* et P'^*. Ainsi, vu que P^* est une sous-trajectoire de P_{t_i,t_j}, MOTI est insoluble pour P_{t_i,t_j}.

– Supposons à présent que $\forall t \in [\![t_i, t_{j-1}]\!]$, M_t est *infaillible*. Nous avons donc : $\forall t \in [\![t_i, t_{j-1}]\!]$, $\nexists\, M \neq M_t$ tel que $S_t \xrightarrow{M} S_{t+1}$. D'où, $\forall t \in [\![t_i, t_{j-1}]\!], \exists! P \in \mathbb{P}_{t,t+1}$ de S_t à S_{t+1} et, de par son unicité, ce P est une sous-trajectoire du P_{t_i,t_j} considéré entre les instants t et $t+1$. Ainsi, $\forall P' \in \mathbb{P}_{t,t+1}$ tel que $P \neq P'$, l'état initial du système (ou respectivement final) pour P' n'est pas le même que l'état initial (ou respectivement final) du système pour P. Alors, par l'emploi d'une fonction bijective map quelconque telle qu'introduite dans la preuve du théorème 2.1, nous avons : $\forall P' \in \mathbb{P}_{t,t+1} : P \neq$

[5] La notation $e_{S,S'}$ représente l'ensemble des mouvements unitaires permettant de passer de S à S'.

$P' \Longrightarrow map(P) \neq map(P')$. Si map est la fonction identité, nous avons : $\forall P' \in \mathbb{P}_{t,t+1} : P \neq P' \Longrightarrow \overline{P} \neq \overline{P'}$. Ainsi, par l'utilisation du corollaire 2.1, MOTI est résoluble. ■

Bien que la caractérisation de la résolubilité de MOTI pour une trajectoire globale donnée soit utile et nécessaire, et cela pour tous les systèmes pour lesquels il doit être décidé à chaque instant si MOTI est résoluble, il est également possible de définir un ensemble de cas pour lesquels MOTI est toujours résoluble, et ce, quelle que soit la trajectoire des objets. Cette seconde caractérisation généralise le théorème 2.3 pour toutes les trajectoires globales possibles du système.

Theorème 2.4 ($\mathbb{P}$-résolubilité) *Soit $SG(\mathbb{S}, \mathbb{M})$ un graphe des états défini pour un GCC de $\mathbb{G}$, $G(V, E)$, contenant les objets mouvants de $\mathbb{O}$. $\forall P \in \mathbb{P}_{t_i,t_j}$: MOTI est résoluble ssi $\forall S \in \mathbb{S},\ S$ est infaillible.*

Preuve – Supposons que $\forall P \in \mathbb{P}_{t_i,t_j}$, MOTI est résoluble. Alors, $\forall P \in \mathbb{P}_{t_i,t_j}, \forall t \in [\![t_i, t_{j-1}]\!]$: M_t est infaillible (théorème 2.3) et donc $\forall P \in \mathbb{P}_{t_i,t_j}, \forall t \in [\![t_i, t_{j-1}]\!]$: S_t est infaillible (définition 2.6). Étant donné que chaque état du système peut apparaître comme l'état initial d'une trajectoire considérée, nous avons : $\forall S \in \mathbb{S}, \exists P \in \mathbb{P}_{t_i,t_j} : S_{t_i} = S$. Ainsi, comme tous les états du système sont infaillibles pour toutes les trajectoires possibles, il en découle que $\forall S \in \mathbb{S},\ S$ est infaillible.

– Supposons maintenant que $\forall S \in \mathbb{S},\ S$ est infaillible. De part la définition 2.6, nous avons $\forall M \in \mathbb{M}, M$ est infaillible. Donc, $\forall P \in \mathbb{P}_{t_i,t_j}, \forall t \in [\![t_i, t_{j-1}]\!], M$ est infaillible. Le théorème 2.3 permet de conclure que $\forall P \in \mathbb{P}_{t_i,t_j}$, MOTI est résoluble. ■

2.6 Une condition suffisante rendant MOTI $\mathbb{P}$-résoluble (*a priori*)

Dans cette partie, nous montrons comment MOTI peut être résolu par l'ajout de contraintes sur les caractéristiques du système. Plus précisément, nous montrons comment le problème devient résoluble s'il est possible de supposer que les mouvements des objets soient « limités ».

Comme il a été préalablement décrit, le graphe des états associé à un système peut contenir un ou plusieurs états faillibles, rendant alors MOTI insoluble. Une méthode simple, pour éviter la présence d'états faillibles dans un graphe des états, revient à éliminer tous les mouvements faillibles parmi l'ensemble des mouvements globaux possibles, permettant ainsi d'obtenir uniquement des mouvements effectifs infaillibles. D'un point de vue pratique, cela revient à limiter les mouvements des objets dans l'environnement, à modifier l'environnement lui-même, ou à adjoindre des contraintes sur les déplacements. Par exemple, prenons le cas d'un bâtiment dans lequel des personnes peuvent se déplacer librement d'une pièce à l'autre en passant par des portes. Les restrictions peuvent être effectuées par un blocage temporaire de l'ouverture d'une porte. Dans notre modèle, cela revient à éliminer les arcs correspondants à ces mouvements dans le GCC donné, et ainsi, à modifier le graphe des états.

Fondamentalement, l'unique cause d'insolubilité de MOTI repose sur la présence de cycles dans le GCC représentant le système considéré. Lorsque deux objets, ou plus, se déplacent dans un cycle, il peut être impossible de déterminer leurs trajectoires par simple observation des vecteurs d'état. Sur la figure 2.5, l'observateur ne peut déterminer si tous les objets se sont déplacés dans le sens horaire ou si un seul objet s'est déplacé dans le sens trigonométrique.

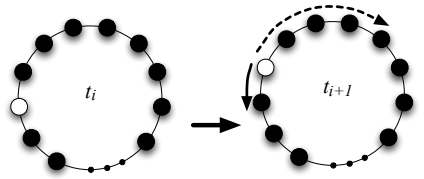

FIG. 2.5 – Confusion possible entre les deux trajectoires.

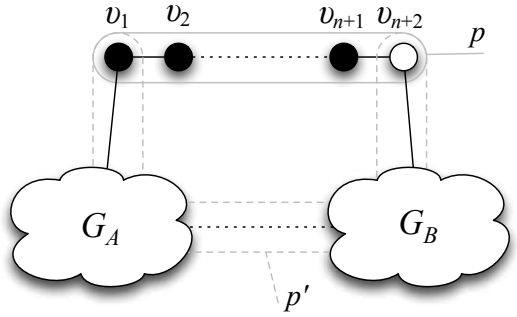

FIG. 2.6 – Schématisation du GCC illustrant l'existence potentielle de chemins entre v_1 et v_{n+2}.

Le problème peut être évité en limitant à k déplacements le nombre d'objets de $\mathbb{O}$ pouvant se mouvoir simultanément. Cette dernière valeur k dépend fortement de la topologie du GCC.

Dans ce cas, nous pouvons garantir que MOTI est résoluble quelque soit la trajectoire P, respectant à tout instant la condition d'*au plus k déplacements simultanés*. L'ensemble de toutes ces trajectoires est une fraction de $\mathbb{P}$, dénoté $\mathbb{P}^k$ dans la suite.

Theorème 2.5 *Soient $k \leq |\mathbb{O}|$ le nombre maximum d'objets se déplaçant simultanément, $\ell > 1$ un entier et un GCC ne contenant pas de cycle de longueur $1 < l < \ell$.*

$$\forall P \in \mathbb{P}^k : \textit{MOTI est } P\textit{-résoluble ssi } \left\lceil \frac{\ell}{2} \right\rceil > k.$$

Preuve – Prouvons de prime abord la première implication : $k < \lceil \frac{\ell}{2} \rceil \Rightarrow \forall P \in \mathbb{P}^k$, P ne contient aucun mouvement faillible. Pour cela, une preuve par induction est conduite sur le nombre $n < k$ d'objets se déplaçant simultanément à chaque instant. Par la suite, étant donnés S et S' deux vecteurs d'états, *diff*(S, S') représente le nombre de sommets sur G dont l'état a changé entre S et S'. Remarquons que *diff*(S, S') est toujours un nombre pair, attendu que pour chaque mouvement d'un objet ou d'une chaîne d'objets, deux sommets changent d'état dans l'état suivant.
Étape initiale ($n = 1$) : Considérons l'algorithme de construction du graphe des états SG proposé au paragraphe 2.5.2. À la première phase itérative de cet algorithme, les arcs étiquetés par les mouvements isolés sont adjoints au graphe, initialement sans arc. Ces arcs relient tous les couples possibles de vecteurs d'états tels que *diff*$(S, S') = 2$ et qu'il existe un arc sur G reliant les deux sommets ayant inversé leurs états. Pour chacun de ces couples (S, S'), un unique arc est ajouté car au plus un objet a effectué le mouvement associé aux sommets changeant d'état entre S et S' (*cf.* la définition d'un mouvement isolé). Ainsi, pour un système sur lequel un seul objet ne se déplace à la fois (*i.e.* $n = 1$), le graphe des états résultant ne contient aucun mouvement faillible.
Hypothèse d'induction : Supposons que si $n < k$ objets se déplacent simultanément, aucune trajectoire $P \in \mathbb{P}^n$ possible ne contient un mouvement faillible.
Étape d'induction (pour $n + 1$) : À présent, considérons le cas où $n + 1$ objets peuvent se déplacer simultanément. Il doit être prouvé que, à partir de l'hypothèse ci-dessus, aucun mouvement faillible n'est adjoint à SG. Plus précisément, $\forall S, S' \in SG$ tels que $2 \leq$ *diff*$(S, S') \leq 2(n + 1)$, si un arc étiqueté par $n+1$ mouvements simultanés est adjoint à SG entre S et S', alors il n'existait pas d'autres

arcs entre ces deux états, étiquetés par au plus $n+1$ mouvements simultanés.

Au préalable, considérons le cas où *diff*$(S, S') = 2$. Supposons que les deux sommets changeant d'état entre S et S' sont étiquetés respectivement v_1 et v_{n+2}. Un arc étiqueté par $n+1$ mouvements peut être ajouté à SG entre S et S' seulement si le chemin $p = v_1, v_2, \ldots, v_{n+1}, v_{n+2}$ existe dans G et que $\forall v \in \{v_1, \ldots, v_{n+1}\}, S[v] = 1$. Dans ce cas, montrons que le seul mouvement possible menant l'état du système de S à S' est celui où chaque objet situé en v_i, avec $i \in [\![1, n+1]\!]$, se déplace sur v_{i+1}. Supposons, sans perdre en généralité, qu'il existe un autre mouvement possible qui n'implique pas les objets localisés sur les sommets $v_2, \ldots, v_{n+1}$.

La figure 2.6 représente ce dernier cas, où l'objet situé en v_1 doit se déplacer sur un sous-graphe G_A de G et un objet situé sur un sous-graphe G_B doit se déplacer en v_{n+2}. Si le seul chemin reliant les sommets de G_A avec les sommets de G_B est p (*i.e.* il n'existe pas de lien p' comme représenté en figure 2.6), alors $|\{v \in G_A : S'[v] = 1\}| > |\{v \in G_A : S[v] = 1\}|$ (et respectivement, $|\{v \in G_B : S'[v] = 1\}| < |\{v \in G_B : S[v] = 1\}|$). Ceci implique que $\exists v \in G_A \cup G_B : S[v] \neq S'[v]$, et donc, *diff*$(S, S') > 2$, lequel est impossible au vu de nos hypothèses initiales. À l'opposé, si il existe un chemin $p' \neq p$ permettant de connecter G_A à G_B, alors p fait partie d'un cycle $c \subset G$. La longueur de ce cycle c est, par hypothèse, au moins ℓ. Afin d'assurer que *diff*$(S, S') = 2$, sur S, il doit y avoir un objet situé sur tous les sommets de c excepté v_{n+2}. De plus, tous les objets de ces sommets, à l'exception de ceux localisés sur $v_2, \ldots, v_{n+1}$, doivent se déplacer (sinon *diff*$(S, S') > 2$). Auquel cas, cela signifie que m $(\geq \ell - n)$ objets doivent bouger simultanément. Or, étant donné que $\lceil \frac{\ell}{2} \rceil > k \geq n+1$, nous avons $m > 2(n+1) - n > n+1$, *i.e.* tout autre arc connectant S à S' dans SG doit être étiqueté par plus de $n+1$ mouvements.

Présentement, considérons le cas où *diff*$(S, S') = 2(x+1)$ avec $x \leq n$. Ici, il y a $n+1$ objets distincts se déplaçant sur $x+1$ chemins distincts. Chacun de ces derniers est caractérisé par la présence d'un objet sur chacun des sommets le composant, excepté le dernier (à l'instar de p sur la figure 2.6). Un raisonnement identique au cas précédent peut être mené séparément sur chacun des $x+1$ chemins, compte tenu qu'ils contiennent respectivement moins de n objets.

– En second lieu, prouvons la seconde implication de ce théorème par contradiction : $\forall P \in \mathbb{P}^k$, MOTI est P-résoluble $\Rightarrow \lceil \frac{\ell}{2} \rceil > k$. Supposons que $k \geq \lceil \frac{\ell}{2} \rceil$. Notons c le plus petit cycle du GCC G, constitué par les sommets $v_1, \ldots, v_\ell$. Dès lors, sans perdre en généralité, considérons un vecteur d'état S tel que $\forall i \in \{2 \cdot k - 1 | k \in [1, \lceil \frac{\ell}{2} \rceil]\} : S[v_i] = 1$ et $\forall i \in \{2 \cdot k | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\} : S[v_i] = 0$. Considérons également le vecteur d'état S' identique à S mais où $\forall i \in \{2 \cdot k - 1 | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\} : S[v_i] = 0$ et $\forall i \in \{2 \cdot k | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\} : S[v_i] = 1$ (*cf.* illustration en figure 2.7). Remarquons que si ℓ est impaire alors $S'[v_\ell]$ reste inchangé et égal à 1. Ainsi, dans ce cas, l'état de tous les sommets – excepté le dernier dans le cas où ℓ est impair – indéxés de v_1 à v_ℓ sont inversés. Ces deux vecteurs d'états sont assurément dans SG, en raison de l'hypothèse $k \geq \lceil \frac{\ell}{2} \rceil$. Considérons alors les deux mouvements globaux suivants : $M = \{[v_i, v_{i+1 \mod \ell}\}_{i \in \{2 \cdot k - 1 | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\}}$ et $M' = \{[v_i, v_{i-1 \mod \ell}\}_{i \in \{2 \cdot k - 1 | k \in [1, \lceil \frac{\ell}{2} \rceil]\}}$. Chacun de ces mouvements relie S à S' dans SG et est étiqueté par un ensemble de mouvements simultanés de cardinal inférieur ou égal à k. Par ce fait, il existe des mouvements faillibles dans SG. Ceci contredit l'hypothèse initiale selon laquelle $\forall P \in \mathbb{P}^k$, MOTI est P-résoluble. ■

D'un point de vue pratique, deux méthodes sont envisageables pour garantir *a priori* les conditions du théorème 2.5 dans tous les cas. La première consiste à choisir le GCC selon une topologie caractérisée par des cycles de longueur strictement supérieure à $2 \cdot x$. En conséquence, MOTI est $\mathbb{P}$-résoluble pour tout système caractérisé par un graphe acyclique, comme le justifie le théorème 2.6 ci-après. Une seconde méthode nécessite de limiter le nombre d'objets pouvant se déplacer simultanément dans le système (*e.g.* naïvement, un moyen simple consiste à ne pas autoriser plus de $\lceil \frac{\ell}{2} \rceil - 1$ objets dans le GCC).

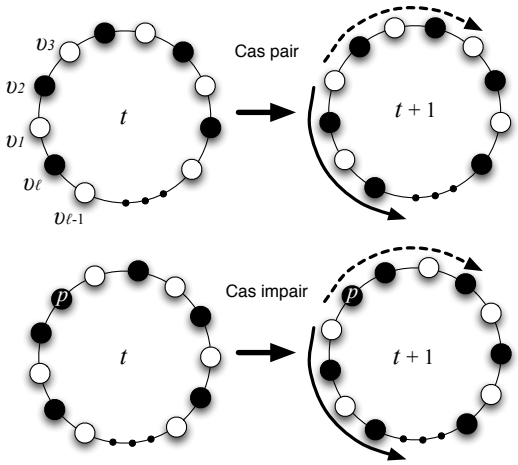

FIG. 2.7 – Illustration de la faillibilité des cycles dans le GCC.

Theorème 2.6 *MOTI est $\mathbb{P}$-résoluble pour n'importe quel système caractérisé par un GCC $G(V, E)$ acyclique.*

Preuve – En considérant G comme un graphe ayant un plus petit cycle de longueur infinie, le théorème 2.5 permet de conclure que MOTI est $\mathbb{P}$-résoluble tant qu'un nombre fini d'objets se déplacent simultanément dans le système. Étant donné que $\mathbb{O}$ est supposé fini, la $\mathbb{P}$-résolubilité est toujours garantie. ■

2.7 Contraindre le mouvement des objets en temps réel avec un observateur actif

Contraindre le mouvement des objets *a priori* nécessite le plus souvent d'imposer des restrictions trop fortes sur le GCC, et sur le système, pour être réalisable en pratique. Dans cette partie, nous développons des méthodes pour assurer la résolubilité de MOTI par une analyse en temps réel de la faillibilité de la situation. Inspiré par l'algorithme naïf présenté aussitôt après, nous proposons une version décentralisée de cet algorithme, permettant une caractérisation et une résolution de MOTI, sans nécessiter le lourd calcul du graphe des états.

Dans toute cette partie, nous considérons un *observateur actif*, *i.e.* qui peut agir, ou faire agir, sur la topologie du GCC ou sur le déplacement des objets mouvants.

2.7.1 Algorithme simple avec omniscience

A chaque instant de temps, l'observateur reçoit le vecteur d'état S_t. À partir de celui-ci, il exécute l'algorithme 2.2 présenté ci-dessous, et agit en conséquence.

Cet algorithme vérifie en premier lieu que le système est dans un état infaillible. Si tel est le cas, tous les mouvements des objets sont autorisés, et l'observateur attend donc le temps

Algorithme 2.2 : Contrainte des mouvements des objets en temps réel

Données : G, un GCC ; SG, le graphe des états correspondant ; S_t, un vecteur d'état ;
1 **si** S_t *est faillible dans* SG **alors**
2 $Q \leftarrow$ Sélectionner $|\mathbb{O}| - k$ objets dans $\mathbb{O}$;
3 Paralyser les déplacements de tous les objets $o \in Q$ dans le prochain mouvement;
4 **ou**
5 **si** $\exists S \in adj_{SG}(S_t)$ *tel que* S *est infaillible* **alors**
6 Astreindre le système à retourner dans un état infaillible;

suivant. À l'opposé, l'état est faillible et l'observateur doit donc agir pour éviter un mouvement faillible, rendant impossible la résolution de MOTI. Deux possibilités sont alors envisageables :

- limiter le mouvement global à k mouvements unitaires d'objets simultanément par rapport au théorème 2.5 ;
- essayer de forcer le système à revenir dans un état infaillible en contrôlant les mouvements unitaires de certains objets.

Le problème de cet algorithme réside dans la connaissance intrinsèque du graphe des états. En effet, ce graphe est composé de $n = \binom{|V|}{x}$ sommets. Le paragraphe 2.5.2 propose un algorithme permettant de construire efficacement ce graphe, mais celui-ci (algorithme 2.1) a pourtant une complexité en $O(n^3)$. Cette dernière n'est pas envisageable pour un GCC avec un grand nombre de sommets et un nombre d'objets tel que $1 \ll x \ll |V|$.

2.7.2 Algorithme sans connaissance du graphe des états

Dans cette sous-partie, nous proposons une caractérisation de la faillibilité, uniquement par l'observation d'un ou deux états consécutifs du système. Ainsi, il n'est pas nécessaire pour l'observateur de connaître, et *a fortiori* de construire, le graphe des états. Consécutivement, nous étendons l'algorithme précédent avec l'utilisation de ces nouvelles caractérisations.

(a) Théorèmes fondamentaux

Dès lors et dans le reste de ce document, pour tout cycle c d'un graphe G, nous noterons ℓ_c sa longueur en terme de sommets. Par exemple, pour le graphe présenté en partie gauche des figures 2.2 et 2.3, sa longueur est $\ell_G = 4$.

Theorème 2.7 (Caractérisation d'état faillible) *Soit G un GCC et $S \in \mathbb{S}$ un vecteur d'état. S est faillible ssi $\exists c \subseteq G$ un cycle tel que $\forall v \in c, S[v] = 0 \Rightarrow \forall v' \in adj_c(v), S[v'] = 1$.*

Remarque : *Ce théorème affirme que, étant donné un vecteur d'état S, celui-ci est infaillible* ssi *dans chaque cycle de G, il existe deux sommets inoccupés consécutifs.*

Preuve du Théorème 2.7 – Premièrement, considérons que la seconde hypothèse est vérifiée. Étant donné que le cas acyclique est trivial (*cf.* théorème 2.6), considérons le cas d'un GCC G contenant au moins un cycle, noté c dans le reste de la preuve. Supposons que les ℓ_c sommets du cycle c

sont étiquetés de v_1 à v_{ℓ_c} dans le sens horaire (*cf.* figure 2.7).

Définition 2.7 *Soit* $\kappa_c(S) = \{i \in [1, \ell_c] | S[v_i] = 0\}$ *(il est évident que* $\forall k \in \kappa_c(S), v_k \in c$*).* $\diamond$

Afin de mieux appréhender cette preuve, un exemple simple est présenté préalablement ($|\kappa_c(S)| = 1$), suivi du cas générique ($|\kappa_c(S)| = l$).

• $|\kappa_c(S)| = 1$: Dans ce cas, $\exists! k \in [1, \ell_c]$ tel que $S[v_k] = 0$. Considérons les deux états suivants :

$$\begin{array}{rcccccccc} & & & & k & & & & \\ & & & & \downarrow & & & & \\ S = & [1 & \cdots & 1 & 0 & 1 & 1 & \cdots & 1] \\ \text{et} \quad S' = & [1 & \cdots & 1 & 1 & 0 & 1 & \cdots & 1] \end{array}$$

Donc, il existe deux mouvements globaux $P, P' \in \mathbb{P}_{t,t+1}$ tels que :

- $P \backslash \{[v_i, v_j]\}_{i \in [1,\ell_c]} = P' \backslash \{[v_i, v_j]\}_{i \in [1,\ell_c]}$: tous les objets de $G \backslash c$ (*i.e.* les objets situés hors de c) ont le même déplacement dans P et P' ;
- $P \supseteq \{[v_i, v_i]\}_{i \in [1,\ell_c] \backslash \{k,k+1\}} \cup \{[v_{k+1}, v_k]\}$: un unique objet dans c s'est déplacé dans le sens trigonométrique ;
- $P' \supseteq \{[v_i, v_{i+1}]\}_{i \in [1,\ell_c] \backslash \{k\}}$: tous les objets dans c se sont déplacés dans le sens horaire.

Donc, il existe au moins deux mouvements globaux distincts permettant de passer de S à S'. D'où, ces mouvements sont faillibles (*cf.* définition 2.5) et S l'est également (*cf.* défintion 2.6).

• $|\kappa_c(S)| = l$ avec $l < \ell_c$: Ce cas correspond simplement à la généralisation de l'exemple suscité. La figure 2.7 illustre les cas extrêmes pour un cycle de longueur paire ou impaire. Par définition, $\forall i \in \kappa_c(S), S_t[i] = 0$. Considérons un vecteur d'état S' tel que $\forall i \in [2, \ell_c], S'[i] = S[i-1]$ et $S'[1] = S[\ell_c]$. Notons $\kappa = \kappa_c(S)$ et $\kappa^+ = \{i + 1 \mod \ell_c | i \in \kappa_c(S)\}$. Ainsi, nous pouvons décrire simplement les deux mouvements globaux $P, P' \in \mathbb{P}_{t,t+1}$ suivant :

- $P \backslash \{[v_i, v_j]\}_{i \in [1,\ell_c]} = P' \backslash \{[v_i, v_j]\}_{i \in [1,\ell_c]}$;
- $P \supseteq \{[v_i, v_i]\}_{i \in [1,\ell_c] \backslash (\kappa \cup \kappa^+)} \cup \{[v_{i+1}, v_i]\}_{i \in \kappa}$: seulement $|\kappa|$ objets sur c se sont déplacés dans le sens trigonométrique ;
- $P' \supseteq \{[v_i, v_{i+1}]\}_{i \in [1,\ell_c] \backslash \kappa}$: tous les objets sur c se sont déplacés dans le sens horaire.

Donc, de la même manière, il existe au moins deux mouvements globaux distincts permettant de passer de S à S'. Ces mouvements sont alors faillibles et S l'est également.

– À présent, en second lieu, montrons par la contraposée la première implication du théorème. Nous avons l'hypothèse suivante :

$$\forall c \subseteq G \text{ un cycle tel que } \exists v \in c, S[v] = 0 \wedge \exists v' \in adj_c(v), S[v'] = 0$$

$$\iff \forall c \subseteq G \text{ un cycle tel que } \exists v \in c, \exists v' \in adj_c(v), S[v] = S[v'] = 0. \tag{2.4}$$

Si G est acyclique, le théorème 2.6 assure que S est infaillible. Dès lors, considérons un GCC G avec au moins un cycle, dénoté c dans le reste de la preuve. Nous montrons alors par induction sur la longueur du cycle que tout mouvement global issu de S est infaillible.

Étape initiale ($\ell_c = 3$) : Dans ce contexte, $\exists! k \in [1, 3]$ tel que $S[v_k] = 1$ (dans le cas contraire, l'hypothèse 2.4 ne serait pas vérifiée). Alors, pour chaque mouvement unitaire de cet objet isolé, celui-ci est trivialement unique et infaillible. Donc, tous les mouvements sur c issus de S sont infaillibles si $\ell_c = 3$;

Étendons ce résultat aux cycles de taille $\ell_c = 4$: En utilisant le graphe des états de la figure 2.3 (page 38), les seuls états vérifiant l'hypothèse 2.4 sont les sommets S_0, S_1, S_4 and S_5. $\forall i \in \{0, 1, 4, 5\}$, la figure 2.3 montre que tous les mouvements sur c ayant pour origine S_i sont infaillibles.

Hypothèse d'induction : Étant donnée ℓ_c, $\forall x' \in [1, \ell_c - 2]$ représentant le nombre d'objets localisés dans le cycle c : si le vecteur d'état S vérifie l'hypothèse 2.4, alors tous les mouvements sur c issus de S sont infaillibles.
Étape d'induction (pour ℓ_c+1) : Considérons un cycle c de longueur ℓ_c+1 vérifiant l'hypothèse 2.4. Nous procédons en deux étapes :
1. $\forall x' \in [1, \ell_c - 2]$ représentant le nombre d'objets dans c, le résultat est équivalent à un cycle de longueur ℓ_c augmenté d'un sommet supplémentaire v tel que $S[v] = 0$. De façon immédiate, il est évident qu'ajouter un sommet inoccupé dans un cycle ne peut infirmer l'hypothèse 2.4. Donc, d'après l'hypothèse d'induction, tous les mouvements issus de S sont infaillibles.
2. Soit $x' = \ell_c - 1$ de sorte que l'hypothèse 2.4 est vérifiée. Dans ce cas, tous les objets sont groupés d'un coté du cycle, et les deux sommets inoccupés sont adjacents de l'autre coté. Ce cas est trivialement équivalent à celui d'un cycle de longueur ℓ_c contenant exactement $\ell_c - 2$ objets. D'après l'hypothèse d'induction, tous les mouvements issus de S sont infaillibles.

Ainsi, nous pouvons conclure que $\forall c \subseteq G$ un cycle, $\forall x' \in [1, \ell_c - 2]$ le nombre d'objets localisés sur c : si S vérifie l'hypothèse 2.4, alors tous les mouvements issus de S sont infaillibles, et par la définition 2.6, S est lui-même infaillible. ■

Corollaire 2.2 *Soit un GCC G et S un vecteur d'état. Si S est faillible, alors $\exists c \subseteq G$ un cycle tel que $|\{v \in c | S[v] = 1\}| \geq \frac{\ell_c}{2}$ (*i.e. *il y a plus de $\frac{\ell}{2}$ objets sur un cycle de longueur ℓ).*

Preuve – Soit un vecteur d'état faillible S. Par le théorème 2.7, on a :

$$\exists c \subseteq G \text{ un cycle tel que } \forall v \in c, S[v] = 0 \Rightarrow \forall v' \in adj_c(v), S[v'] = 1.$$

En d'autres termes, chaque sommet inoccupé est entouré par des sommets hébergeant un objet :

$$\forall v \in c, S[v] = 0 \Rightarrow \exists v', v'' \in c, S[v'] = S[v''] = 1$$

$$\Longrightarrow |\{v \in c | S[v] = 0\}| \leq |\{v \in c | S[v] = 1\}|$$

D'autre part, comme un vecteur d'état est défini sur $\{0; 1\}$ (*cf.* définition 2.1), nous avons :

$$|\{v \in c | S[v] = 0\}| + |\{v \in c | S[v] = 1\}| = \ell_c \Longrightarrow 2 \cdot |\{v \in c | S[v] = 1\}| \geq \ell_c$$

$$\Longrightarrow |\{v \in c | S[v] = 1\}| \geq \frac{\ell_c}{2}.$$ ■

De manière immédiate et évidente, la contraposée nous donne :

Corollaire 2.3 *Soit un GCC G et $S \in \mathbb{S}$. $\forall c \subseteq G$ un cycle tel que $|\{v \in c | S[v] = 1\}| < \frac{\ell_c}{2}$, S est infaillible (*i.e. *si il y a moins de $\frac{\ell_c}{2}$ objets dans tout cycle c de G, alors S est infaillible).*

À présent, à l'instar du théorème 2.7 pour les états du système, il est nécessaire de caractériser un mouvement faillible sans utiliser le graphe des états. Dans le reste de cette partie, nous écartons de l'étude délibérément les GCC acycliques, pour lesquels le théorème 2.6 assure que MOTI est $\mathbb{P}$-solvable, et donc que tout état ou mouvement est infaillible.

Définition 2.8 (Longueur de chaîne) *Soit un chemin p entre deux sommets dans un graphe G. La* longueur de chaîne *$l_p(v, v')$ est le nombre de sommets de p de v à v' non-inclus.* ◇

Lemme 2.1 *Par extension, en considérant un cycle c sur lequel les sommets sont étiquetés consécutivement de v_1 à v_{ℓ_c}, nous avons :*

$$l_p(v_i, v_j) = \begin{cases} j - i - 1 & \text{si } i < j \\ \ell_c + j - i - 1 & \text{si } i \geq j \end{cases}$$

Définition 2.9 (Chaîne minimale continue d'objets) *La* chaîne minimale continue d'objets issue de v_i *sur un cycle c au temps t, notée $\mu_{c,t}(v_i)$, est la plus plus petite longueur de chaîne sur c, partant de v_i et ne contenant que des sommets occupés au temps t :*

$$\begin{cases} \mu^+_{c,t}(v_i) & = & \min_{j \in \kappa_c(S_t)} l_c(v_i, v_j) & \textit{dans le sens trigonométrique} \\ \mu^-_{c,t}(v_i) & = & \min_{j \in \kappa_c(S_t)} l_c(v_j, v_i) & \textit{dans le sens horaire.} \end{cases}$$

La généralisation sur tout le cycle c permet de définir la chaîne minimale continue d'objets *sur ce cycle au temps t :* $\mu_{c,t} = \min_{i \in \kappa_c(S_t)} \min(\mu^+_{c,t}(v_i), \mu^-_{c,t}(v_i)).$ ◇

Sans perdre en généralité, pour un cycle c donné, nous considérons dès lors que l'hypothèse d'étiquetage consécutif $(v_1, \ldots, v_{\ell_c})$ de c est toujours vérifiée. Considérons également que pour un $k \in \{-1, 1\}$, la notation $\mu^k_{c,t}(v_i)$ représente la chaîne minimale dans le sens de k ($\equiv$ signe).

Theorème 2.8 (Caractérisation de mouvement faillible) *Soient un GCC G et S_t un vecteur d'état au temps t tel que S_t est faillible. Soit $c \subseteq G$ un cycle tel que c vérifie la condition du théorème 2.7 et aucun objet n'entre ou ne quitte c pendant le mouvement M_t. M_t est faillible ssi $\exists k \in \{-1, 1\}, \forall v_i \in c, S_t[v_i] = 0 \Rightarrow \exists j \in [\![1, \mu^k_{c,t}(v_i)]\!], S_{t+1}[v_{i+k \cdot j[\ell_c]}] = 0$.*

Preuve – Considérons que les hypothèses du théorème sont toutes valides pour un cycle c de G. Comme nul objet n'est entré, ou n'a quitté, c durant $[\![t, t+1]\!]$, nous avons $|\kappa_c(S_t)| = |\kappa_c(S_{t+1})|$.

– Supposons que la dernière hypothèse est vérifiée : $\exists k \in \{-1, 1\}, \forall v_i \in c, S_t[v_i] = 0 \Rightarrow \exists j \in [\![1, \mu^k_{c,t}(v_i)]\!], S_{t+1}[v_{i+k \cdot j[\ell_c]}] = 0$. Soit $J^k_{c,t}$ le vecteur contenant, pour tout sommet de c, la valeur de j dans $[\![1, \mu^k_{c,t}(v_i)]\!]$ s'il est inoccupé au temps t, et $\perp$ sinon.

Considérons les deux mouvements $P, P' \in \mathbb{P}_{t,t+1}$ suivant :

- $P \backslash \{[v_i, v_d]\}_{i \in [1, \ell_c]} = P' \backslash \{[v_i, v_d]\}_{i \in [1, \ell_c]} = M_t \backslash \{[v_i, v_d]\}_{i \in [1, \ell_c]}$: tous les objets de $G \backslash c$ (*i.e.* objets situés hors de c) ont les mêmes déplacements que dans M_t ;
- $P \supseteq \begin{cases} \{[v_i, v_{i-k}]\}_{i \in \{d+m | d \in \kappa_c(S_t), m \in [\![1; J^k_{c,t}[v_d]]\!]\}} \\ \{[v_i, v_i]\}_{i \in \{d+m | d \in \kappa_c(S_t), m \in [\![J^k_{c,t}[v_d]+1, \mu^k_{c,t}(v_d)]\!]\}} \end{cases}$: ici, $\sum_{i \in \kappa_c(S_t)} J^k_{c,t}[v_i]$ objets de c se déplacent «dans le sens contraire de k» ;
- $P' \supseteq \begin{cases} \{[v_i, v_{i+k}]\}_{i \in \{d+m | d \in \kappa_c(S_t), m \in [\![J^k_{c,t}[v_d], \mu^k_{c,t}(v_d)]\!]\}} \\ \{[v_i, v_i]\}_{i \in \{d+m | d \in \kappa_c(S_t), m \in [\![1, J^k_{c,t}[v_d]-1]\!]\}} \end{cases}$: ici, $\ell_c - \sum_{i \in \kappa_c(S_t)} J^k_{c,t}[v_i]$ objets de c se déplacent «dans le sens de k» ;

Ainsi, P et P' mènent au même ensemble $\kappa_c(S_{t+1})$. D'où, P et P' correspondent à deux mouvements différents de S_t à S_{t+1}. Donc, il existe au moins un mouvement global différent de M_t entre S_t et S_{t+1}. La définition 2.5 traduit donc que M_t est faillible.

– En second lieu, montrons l'autre implication par la contraposée. Supposons que :

$$\forall k \in \{-1, 1\}, \exists v_i \in c, \forall j \in [\![1, \mu^k_{c,t}(v_i)]\!], S_t[v_i] = 0 \wedge S_{t+1}[v_{i+k \cdot j[\ell_c]}] = 1. \tag{2.5}$$

Partant de cette hypothèse, nous prouvons que M_t est infaillible.

Afin de faciliter la lecture de cette preuve, supposons que tout calcul sur les étiquettes des sommets du cycle c se fait, dès lors, modulo ℓ_c.

Si $\forall v_i \in c, S_t[v_i] = 1$, étant donné que $|\kappa_t| = |\kappa_{t+1}|$, alors $\forall v_i \in c, S_{t+1}[v_i] = 1$. Ce résultat est contradictoire puisque $\exists v_i \in c$ tel que $S_t[v_i] = 0$. Ainsi, $|\kappa_t| = |\kappa_{t+1}| \neq 0$.

Supposons que pour ce même v_i, $S_{t+1}[v_i] = 0$. D'après les caractéristiques de notre modèle (*cf.* paragraphe 2.3), un objet ne peut se déplacer qu'au plus d'un sommet entre deux temps consécutifs. Ainsi, aucun objet n'a transité par v_i entre t et $t+1$. En se basant sur ce sommet inoccupé pendant deux temps consécutifs, l'observateur est toujours capable d'extrapoler les mouvements des objets de c de manière déterministe. Ainsi, M_t est infaillible.

Ainsi, dès lors, supposons que $\forall i \in \kappa_c(S_t), S_{t+1}[v_i] = 1$, sinon, de la même manière, M_t serait infaillible. D'après cette hypothèse et la condition 2.5, il est possible d'inférer que $\exists i \in \kappa_c(S_t), \forall j \in [\![-\mu^-_{c,t}(v_i), \mu^+_{c,t}(v_i)]\!], S_{t+1}[v_{i+j}] = 1$. De plus, il est évident que $\forall i \in \kappa_c(S_t), (i + \mu^+_{c,t}(v_i) + 1) \in \kappa_c(S_t)$ et $(i - \mu^-_{c,t}(v_i) - 1) \in \kappa_c(S_t)$, étant donné la définition 2.9. Les sommets étiquetés par ces deux entiers sont donc occupés au temps $t+1$, étant inoccupés au temps t. D'où, $\exists i \in \kappa_c(S_t), \forall j \in [\![-\mu^-_{c,t}(v_i) - 1, \mu^+_{c,t}(v_i) + 1]\!], S_{t+1}[v_{i+j}] = 1$.

Dans ce cas, comme v_i est occupé au temps $t+1$, supposons, sans perdre en généralité, au vue de la symétrie du problème, qu'un objet s'est déplacé dans le sens trigonométrique de v_{i-1} à v_i. Ainsi, comme $S_{t+1}(v_{i-1}) = 1$, un autre objet a dû se déplacer de v_{i-2} à v_{i-1}, et ainsi de suite jusqu'à $v_{i-\mu^-_{c,t}(v_i)}$. Donc, un objet s'est nécessairement déplacé de $v_{i-\mu^-_{c,t}(v_i)-1}$ à $v_{i-\mu^-_{c,t}(v_i)}$ étant donné l'hypothèse $S_{t+1}[v_{i-\mu^-_{c,t}(v_i)}] = 1$. Or, $(i - \mu^-_{c,t}(v_i) - 1) \in \kappa_c(S_t)$. Donc, aucun objet n'a pu se déplacer de $v_{i-\mu^-_{c,t}(v_i)-1}$ à $v_{-\mu^-_{c,t}(v_i)}$ entre t et $t+1$. Ce mouvement est donc impossible. La condition $\exists i \in \kappa_c(S_t), \forall j \in [\![-\mu^-_{c,t}(v_i) - 1, \mu^+_{c,t}(v_i) + 1]\!], S_{t+1}[v_{i+j}] = 1$ n'est donc jamais vérifiée. D'où, par contradiction, $\exists i \in \kappa_c(S_t), S_{t+1}[v_i] = 0$.

Ainsi, comme il existe un même sommet inoccupé sur c en t et $t+1$, de la même manière que présenté ci-dessus, le mouvement M_t est infaillible.

D'où, l'équivalence suivante :

M_t est faillible ssi $\exists k \in \{-1, 1\}, \forall i \in \kappa_c(S_t), \exists j \in [\![1, \mu^k_{c,t}(v_i)]\!], S_{t+1}[v_{i+k\cdot j[\ell_c]}] = 0$. ■

Le théorème précédent permet d'obtenir simplement le résultat suivant.

Corollaire 2.4 *Soient un GCC G et $P \in \mathbb{P}_{t_i,t_j}$ une trajectoire. Si $\exists t \in [\![t_i, t_j - 1]\!], \exists c \subseteq G$ un cycle tels que $\forall v \in c, S_t[v] = S_{t+1}[v] = 1$ alors MOTI n'est pas P-résoluble.*

Preuve – Supposons que, étant donné $P \in \mathbb{P}_{t_i,t_j}$, $\exists t \in [\![t_i, t_j]\!], \exists c \subseteq G$ un cycle tels que $\forall v \in c, S_t[v] = S_{t+1}[v] = 1$. Alors, $\mu_{c,t} = \mu_{c,t+1} = \ell_c$ et $\kappa_c(S_t) = \kappa_c(S_t) = \varnothing$. Donc, $\forall v \in c, \forall v' \in adj_c(v), S_t[v] = S_{t+1}[v']$. D'où, pour $k = 1, j = 1, \forall v_i \in c, S_t[v_i] = S_{t+1}[v_{i+k\cdot j[\ell_c]}] = 1$. Par le théorème 2.8, M_t est donc faillible. Ainsi, par la contraposée du théorème 2.3 (page 40), MOTI n'est pas P-résoluble. ■

(b) Algorithme exécuté par l'observateur

Dès à présent, nous pouvons donc établir un algorithme utilisable par l'observateur, qui, reprenant les concepts de l'algorithme 2.2, permet d'éviter les situations faillibles sans construire le graphe des états. A chaque instant, l'observateur reçoit le vecteur d'état S_t et exécute la routine décrite dans l'algorithme 2.3.

Algorithme 2.3 : Prévention de mouvements infaillibles

Données : G, un GCC ; S_t, un vecteur d'état ;
1 $\ell \leftarrow 2$;
2 **répéter**
3 **pour chaque** *cycle* $c \in G$ *de taille* ℓ **faire**
4 marquer(c);
5 **si** $|\{v \in c | S_t[v] = 1\}| \geq \ell/2$ **alors**
6 **si** $\exists v \in c, \exists v' \in adj_c(v)$ *tels que* $S_t[v] = S_t[v'] = 0$ **alors**
7 continuer sur l'itération suivante;
8 **else**
9 **pour chaque** *objet* $\overline{o} \in \overline{\mathbb{O}}$ *tel que* $\overline{loc}_t(\overline{o}) \in c$ **faire**
10 $\gamma_c(\overline{o}) \leftarrow \{v \in adj_G(\overline{loc}_t(\overline{o})) \backslash c | S_t[v] = 0\}$;
11 $\Gamma_c \leftarrow \{\overline{o} \in \overline{\mathbb{O}} | \gamma_c(\overline{o}) \neq \varnothing\}$;
// Γ_c *est l'ensemble des objets potentiellement déplaçables hors de* c
12 **si** $\exists \overline{o} \in \Gamma_c, \exists v \in \gamma_c(\overline{o})$ *tels que* $(\forall$ *cycle* $c' \neq c, v \notin c') \vee$
13 $(\exists$ *cycle* $c' \neq c, \exists v' \in c'$ *tels que* $v \in c' \wedge v \neq v' \wedge S_{t+1}[v'] = 0)$ **alors**
14 déplacer o sur v;
15 **sinon**
16 $k \leftarrow |\kappa_c(S_t)|$;
17 Contraindre les mouvements unitaires sur c à $k-1$ objets;
18 $\ell \leftarrow \ell + 1$;
19 **jusqu'à** *ce qu'il ne reste plus de cycles non marqués dans* G ;

Theorème 2.9 *L'algorithme 2.3 garanti que MOTI est résoluble sur* G *durant l'intervalle* $[\![t_i, t_j]\!]$ *si* $\forall c \in G, \exists v \in c$ *tel que* $S_{t_i}[v] = 0$ *(*i.e. *à l'initialisation, tout cycle n'est pas plein).*

Ce théorème traite de la correction de l'algorithme 2.3. De manière immédiate, il impose un extremum quant au nombre d'objets, celui-ci dépendant évidemment de la topologie du GCC considéré. De plus, plus le nombre d'objets présents est élevé, moins la liberté de mouvement de ceux-ci est importante, et donc, le nombre de trajectoires réalisables réduit d'autant, en raison des contraintes du modèle.

La terminaison de cet algorithme est quand à elle triviale : le nombre de cycle dans G est fini par définition (E est un ensemble fini). ℓ admet donc un maximum, correspondant au plus long cycle de G. Pour toute valeur de ℓ, les lignes 3 et 4 garantisent que tout cycle de longueur ℓ sera visité et marqué. Enfin, chaque itération de la boucle 2–19, ℓ croît strictement, et le nombre de cycles non-marqués décroît invariablement. Le nombre d'itérations de cette boucle est donc fini et l'algorithme termine donc toujours.

Preuve du théoreme 2.9 – La condition initiale permet d'assurer qu'au premier temps t_i, la condition du corollaire 2.4 ne peut être vérifiée. Ainsi, au pire, l'état initial du système (S_{t_i}) peut être faillible mais MOTI peut rester résoluble si aucun mouvement faillible n'advient. Prenons un temps quelconque $t \in [\![t_i, t_j - 1]\!]$, et montrons que M_t ne peut être faillible et que S_{t+1} ne peut entrer dans le cadre du corollaire 2.4. Ainsi, par induction sur t, nous pourrons conclure que l'algorithme garanti la résolubilité de MOTI sur l'intervalle de temps $[\![t_i, t_j]\!]$.

Les deux boucles imbriquées de cet algorithme (initiées aux lignes 2 et 3) garantissent que tout les cycles de G seront pris en compte. Le reste du graphe peut être ignoré, de part le fait qu'aucun mouvement faillible ne peut survenir hors d'un cycle (*cf.* théorème 2.6). Pour chacun des cycles, le ligne 5 permet de vérifier si ce cycle peut amener le système dans un état faillible, sous les conditions du corollaire 2.2. Si c'est le cas, la ligne 6 vérifie la faillibilité exacte de cet état par le théorème 2.7 et ce cycle peut donc être ignoré. Sinon, le cycle considéré rend l'état faillible et un traitement est effectué à partir de la ligne 8 (*cf.* théorème 2.7).

Dans ce cas, l'algorithme recense tous les objets pouvant sortir du cycle c donné (*cf.* ligne 9 à 11). S'il est possible de déplacer un objet hors de c et dans une partie sans cycle de G (ligne 12), ce mouvement sera privilégié afin de retourner dans un état S_{t+1} infaillible, par le corollaire 2.3. Dans le cas ou aucune partie acyclique de G n'est accessible, considérons c' le cycle herbergeant potentiellement l'objet $\overline{o}$ au temps $t+1$. L'algorithme évite à la ligne 13 de remplir complètement c' au temps $t+1$ afin d'exclure une retombée dans le cadre du corollaire 2.4. Pour cela, il vérifie que quelque soit la trajectoire locale à c' possible, il restera au moins un sommet inoccupé à l'instant $t+1$ si $\overline{o}$ se déplace en $v \in c'$.

Si aucune des conditions précédentes ne peut être vérifiée, alors contraindre un seul objet demeure infaisable sur c. Les lignes 16 et 17 permettent donc de limiter le mouvement global sur c à $k-1$ objets ; ceci dans l'optique d'éviter que les conditions du théorème 2.8 ne soient vérifiées.

Ainsi, pour chaque cycle, le mouvement suivant l'instant t est forcément infaillible sur ce cycle. Étant donné que tous les cycles sont pris en compte dans cet algorithme, il permet de garantir que $\forall t \in [\![t_i, t_j - 1]\!]$, M_t est infaillbile. D'où, MOTI est résoluble sur l'intervalle $[\![t_i, t_j]\!]$. ■

2.8 Algorithmes répartis large-échelle sans observateur

À partir du précédent algorithme, il est naturel d'imaginer une version aux conditions encore plus allégées et locales, évitant d'une part la construction du graphe des états, mais également, la nécessité d'un observateur global. Le système pouvant être très grand, il est important de décentraliser la détection de mouvements faillibles et donc, d'évaluer la faillibilité de la situation d'un point de vue strictement local. Pour cela, nous proposons deux algorithmes, lesquels effectuent la même tâche mais se complètent en fonction de la topologie du GCC et du nombre d'objets se déplaçant sur celui-ci.

(a) Algorithme réparti de détection 1 – Sans information topologique :

Cet algorithme est une version répartie de base de l'algorithme de détection d'états faillibles.

La routine présentée dans l'algorithme 2.4 est exécutée sur chaque sommet, à chaque temps discret. Elle consiste en un échange des états sur tous les voisinages afin de vérifier si un sommet n'est pas entouré de deux sommets occupés (lignes 1 à 2). Si tel est le cas (ligne 3), le sommet bordé lance une *sonde de détection* aux sommets occupés (lignes 4 à 5).

Ensuite, chaque fois qu'un sommet occupé reçoit une sonde inconnue, il la retransmet à tout son voisinage dans le GCC (lignes 7 à 9). Dans le cas d'un sommet inoccupé, celui-ci ne retransmet la sonde qu'à ses voisins occupés (lignes 11 à 12). En effet, une transmission aux sommets inoccupés serait inutile, en raison de la remarque du théorème 2.7, page 45. Ainsi, si deux sommets consécutifs inoccupés sont présents sur un chemin, l'état de ce chemin est infaillible, et la sonde est arrêtée. Dans tous les autres cas, la sonde fera une boucle complète et reviendra au sommet initiateur.

Algorithme 2.4 : Détection répartie d'états faillibles – Sans information topologique

Données : i, l'identifiant local ; s_i, son état ; $\mathcal{N}$, le voisinage de i dans le GCC ;
1 **Envoyer** $(\texttt{État}, s_i)$ **à** $\mathcal{N}$;
2 $S \leftarrow \langle s \in \{0,1\} \mid$ **Recevoir** $(\texttt{État}, s)$ **de** $j \rangle_{j \in \mathcal{N}}$;
3 **si** $\sum_{j \in \mathcal{N}} S[j] > 1$ **alors**
4 $\quad \mathcal{Q} \leftarrow \{j \in \mathcal{N} \mid S[j] = 1\}$;
5 $\quad$ **Envoyer** `Sonde[`i`]` **à** $\mathcal{Q}$;
6 **À la réception de** `Sonde[`j`]` **du** *sommet* k **faire**
7 $\quad$ **si** $i \neq j$ **alors**
8 $\quad\quad$ **si** $s_i = 1$ **alors**
9 $\quad\quad\quad$ **Envoyer** `Sonde[`j`]` **à** $\mathcal{N} - \{k\}$;
10 $\quad\quad$ **sinon**
11 $\quad\quad\quad \mathcal{Q} \leftarrow \{j \in \mathcal{N} - \{k\} \mid S[j] = 1\}$;
12 $\quad\quad\quad$ **Envoyer** `Sonde[`j`]` **à** $\mathcal{Q}$;
13 $\quad$ **sinon**
14 $\quad\quad$ Lancer une alerte d'état faillible

Lorsqu'un initiateur reçoit sa propre sonde, alors il existe un cycle faillible sur le système. Celui-ci lance donc une alerte informant la faillibilité de l'état courant (ligne 14), par l'appel d'une primitive spécifique de l'application[6].

Ce premier algorithme génère une forte charge de messages sur le réseau dans le cas d'une topologie de GCC avec peu de cycles. En effet, le théorème 2.6 traduit l'inutilité de vérifier l'état des chemins hors des cycles du GCC. L'objectif du second algorithme de détection est de limiter la charge du réseau, en connaissant *a priori* l'ensemble des cycles du GCC.

(b) Algorithme réparti de détection 2 – Avec information topologique :

Dans ce cas, nous considérons que chaque sommet connaît l'ensemble des cycles au sein desquels il apparaît. Celui-ci peut-être inscrit manuellement sur chaque sommet au début de l'observation, ou construit de manière automatique lors d'une phase d'initialisation du réseau.

Ainsi, tout capteur n'appartenant à aucun cycle ne participera pas à l'algorithme de détection, et logiquement, ne générera pas de messages superflus sur le réseau.

Tout autre sommet du GCC, appartenant au moins à un cycle, exécutera alors la routine de l'algorithme 2.5, laquelle est décrite ci-dessous :

- Si aucun objet n'est hébergé par le sommet considéré au temps t, celui-ci n'a aucune action à effectuer jusqu'au prochain temps où un objet se présentera ;
- Dans le cas contraire (ligne 1), si un objet est présent, il vérifie à une distance de ℓ sauts si il existe au moins deux sommets inoccupés consécutifs dans le sens trigonométrique sur chaque cycle auquel il appartient (lignes 2 à 6). S'il existe un cycle pour lequel cette condition n'est pas vérifiée, il lance alors une sonde sur le cycle donné dans le sens trigonométrique (ligne 7).

[6] La méthode d'alerte sort du contexte d'étude de ce problème et n'est donc pas explicitée dans ce manuscrit.

Algorithme 2.5 : Détection répartie d'états faillibles – Avec informations topologiques

Données : i, l'identifiant local ; s_i, son état ; $\mathcal{C}$, la liste des cycles contenant i dans le GCC ; ℓ, une borne du nombre de retransmission ;

1 **si** $s_i = 1$ **alors**
2 $\quad \mathcal{Q} \leftarrow \{j \mid \forall c \in \mathcal{C}, \forall j \in c, l_c(i,j) < \ell\}$;
3 $\quad$ **Envoyer** `AppelÉtat` **à** $\mathcal{Q}$;
4 $\quad S \leftarrow \langle s \in \{0;1\} \mid$ **Recevoir** $(\texttt{État}, s)$ **de** $j\rangle_{\{j \mid \exists c \in \mathcal{C}, l_c(i,j) < \ell\}}$;
5 $\quad$ **pour chaque** $c \in \mathcal{C}$ **faire**
6 $\quad\quad$ **si** $\nexists k \in [i+1; i+\ell-1]$ *tel que* $S[c[x]] = S[c[x+1]] = 0$ **alors**
7 $\quad\quad\quad$ **Envoyer** `Sonde[`i, c`]` **à** $c[i+1]$;

8 **À la réception de** `Sonde[`j, c`]` **du** *sommet* k **faire**
9 $\quad$ **si** $k = c[i-1]$ **alors**
10 $\quad\quad$ **si** $i = j$ **alors**
11 $\quad\quad\quad$ Lancer une alerte d'état faillible
12 $\quad\quad$ **sinon**
13 $\quad\quad\quad$ **si** $S[c[i]] = S[c[i-1]] = 0$ **alors**
14 $\quad\quad\quad\quad$ **Envoyer** `Sonde[`j, c`]` **à** $c[i-1]$;
15 $\quad\quad\quad$ **sinon**
16 $\quad\quad\quad\quad$ **Envoyer** `Sonde[`j, c`]` **à** $c[i+1]$;
17 $\quad$ **sinon**
18 $\quad\quad$ **si** $k = c[i+1] \wedge i \neq j$ **alors**
19 $\quad\quad\quad$ **Envoyer** `Sonde[`j, c`]` **à** $c[i-1]$;

Deux cas de figure peuvent alors se produire en fin de parcours de la sonde. (1) L'initiateur reçoit sa propre sonde provenant du même côté du cycle sur lequel il l'a envoyé. Dans ce cas, cela signifie que la sonde a rencontré deux sommets inoccupés sur le cycle, celui-ci étant alors infaillible. (2) L'initiateur l'a reçu de l'autre côté (ligne 9–10). Dans ce cas, aucune paire de sommets consécutifs inoccupés n'existe sur le cycle c, entraînant alors, à la ligne 11, l'émission d'une alerte d'état faillible (*cf.* théorème 2.7).
Considérons à présent l'évolution du parcours de la sonde. Le sommet exécutant la routine n'est donc pas l'initiateur de la sonde ($i \neq j$). À chaque étape, le sommet courant i compare son état s_i ($= S[c[i]]$) avec l'état de son prédécesseur sur le parcours. Si ces deux sommets sont inoccupés (ligne 13), la sonde fait alors demi-tour (ligne 14), et sera retransmise sans condition jusqu'à l'initiateur de celle-ci (lignes 18 et 19). Dans le cas contraire, la sonde poursuit sa route le long du cycle (ligne 16) jusqu'à trouver soit deux sommets inoccupés consécutifs, soit l'initiateur de la sonde.

(c) Algorithme réparti de prévention – Avec ou sans information topologique :

Les deux algorithmes ci-dessus permettent de détecter la présence d'un état faillible au cours d'une observation. Considérant des capteurs actionneurs, il est possible d'imaginer un algorithme de prévention de mouvement faillible par l'utilisation d'un des deux précédents al-

gorithmes. En effet, le sommet lançant l'alerte d'état faillible pourrait modifier temporairement le GCC pour rendre tout mouvement faillible impossible (*e.g.* par suppression d'un arc sur le cycle vérifiant le théorème 2.7).

2.8.1 Vers une utilisation *a posteriori*

Dans les cas coutumiers où les algorithmes de détection sus-cités ne sont pas associés à une interface de prévention, le traitement des données collectées peut effectivement être effectué *a posteriori*. Au vu des preuves du précédent paragraphe, les mouvements faillibles advenus durant l'observation pourront être aisément extrapolées à partir du traitement des données. Ainsi, entre chacun de ces mouvements faillibles, de manière sûre, les trajectoires observées seront corrélées aux trajectoires réelles. Les utilisateurs du système pourront alors extrapoler un ensemble de résultats infaillibles, sur des périodes plus ou moins longues.

2.9 Conclusion

Dans ce chapitre, nous avons considéré le problème particulier d'associer de manière déterministe des trajectoires observées d'objets mouvants dans un graphe connexe épars. Afin de présenter l'impossibilité d'attribuer un unique identifiant à une trajectoire observée par un RCsF binaire dans le cadre général[7] (*i.e.* le problème MOTI), nous avons mis en évidence la dépendance forte de la résolubilité de ce problème avec la topologie du graphe ainsi qu'avec le nombre d'objets mouvants.

À partir de ce résultat, nous avons présenté diverses restrictions permettant de rendre le problème possible, d'un point de vue fondamental. À mesure de l'avancement des résultats, nous avons cherché également à initier un parallèle avec la mise en œuvre pratique de ces résultats en proposant plusieurs algorithmes de détection et de prévention d'occurrences insolubles dans le contexte du problème MOTI, pour différents cas de figure.

Nous avons finalement proposé deux algorithmes décentralisés permettant de détecter localement, et sans nécessiter d'actions extérieures au RCsF, les situations dangereuses, et de potentiellement les éviter dans le contexte de capteurs-actionneurs. Ce chapitre illustre ainsi clairement l'utilité d'une analyse théorique afin de proposer des solutions pratiques permettant de résoudre un problème donné.

[7] selon des hypothèses très fortes : capteurs binaires parfaits, couverture idéale de la zone, existence d'un observateur omniscient de l'état du graphe, *etc.*

CHAPITRE 3

Structure générique multi-couches décentralisée et auto-organisante à faible consommation

Dans la continuité de l'esprit de ce document, nous progressons dans chacune des deux grandes parties par une méthode allant *du particulier au général*. Le chapitre précédent montre l'intérêt, pour un problème donné, d'une conception d'algorithmes fondée sur un résultat théorique. Dans ce chapitre, nous élargissons le spectre d'étude à toutes les applications sur des RCsF statique, mais avec une approche plus pratique. Nous proposons pour cela une infrastructure générique et adaptable, limitant la consommation d'énergie dans le cadre de ces RCsF.

3.1 Introduction

Contexte Rechercher une information spécifique dans un réseau large-échelle non-structuré revient à chercher une aiguille dans une botte de foin sans détecteur de métaux ! La recherche de fichiers rares dans les systèmes de partage de fichiers pair-à-pair (PàP) [BK07, FHKM04, HKLF+06] relève de la même difficulté. Alors que l'inondation est déconseillée dans les réseaux filaires, elle se révèle résolument inexploitable dans les RCsF, où le souci d'économiser l'énergie domine.

Conformément aux propriétés introduites au chapitre 1, et contrairement aux systèmes répartis classiques, les RCsF sont déployés et configurés en général pour les besoins d'une application spécifique. Cependant, certaines fonctionnalités de base se révèlent être communes à la plupart de ces applications. Notamment, nous avons identifié un ensemble de primitives de communication, celles-ci correspondant à des briques de base, partagées par ces applications. Baptisé **-cast*, cet ensemble est composé des primitives de *broadcast*, d'*anycast* et de k*-cast*. La primitive d'*anycast* consiste à contacter un nœud quelconque dans un sous-ensemble spécifique du réseau. La primitive de k*-cast* consiste à contacter, sans distinction, k nœuds d'un sous-ensemble du réseau. Enfin, la primitive de *broadcast* nécessite de contacter tous les nœuds de ce sous-ensemble. Par exemple, considérons une application d'inventaire, où chaque capteur représente une entité d'un type donné. Envoyer un message à tous les capteurs d'un tel type,

ou interroger le système sur la présence d'un objet de type spécifique, sont des tâches usuelles, réalisables directement par l'utilisation de la collection *-*cast*.

Contribution Nous considérons ici un système dans lequel chaque capteur possède un type. Dans ce chapitre, nous proposons SOLIST (*i.e. Self-Organized Large-scale and lightweight Information-based Sensor Technology*), un réseau structuré multi-couches créant un support au *-*cast*, efficace en terme d'économie d'énergie et de fiabilité[1] pour les RCsF. SOLIST propose donc une interface générique pour les applications dans le contexte des RCsF (gestion de groupement par type et de la collection *-*cast*).

Afin de mettre en oeuvre les primitives de la collection *-*cast*, SOLIST contient une sur-couche logique par type existant dans le système, et y regroupe tous les capteurs de même type. Les primitives de broadcast et k-cast sont mises en œuvre au niveau de ces sur-couches. De surcroît, SOLIST propose une primitive efficace d'anycast, laquelle peut être utilisée seule pour les besoins de l'application, ainsi que pour déterminer un point d'entrée dans une sur-couche particulière.

Ce chapitre est organisé comme suit. Les paragraphes 3.2 et 3.3 introduisent respectivement le modèle du système et l'interface générique fournie. La structure de SOLIST est introduite dans le paragraphe 3.4 et la mise en œuvre de la collection *-*cast* au sein de SOLIST est présentée en détail au paragraphe 3.5. Dans un but d'évaluation, différents scénarios ont été simulés afin de comparer SOLIST avec des mécanismes traditionnels de recherche (marche aléatoire et inondation). Nous présentons les résultats de simulation dans le paragraphe 3.6. Avant de conclure en section 3.8, à titre de comparaison, nous terminerons ce chapitre par un état de l'art succinct.

3.2 Prérequis du système

Dans ce chapitre, nous considérons un réseau composé de n capteurs sans-fil, répartis dans un espace géographique donné. Bien qu'aucune hypothèse ne soit imposée sur la répartition des nœuds, la topologie doit cependant assurer la connexité du réseau. Nous ne considérons que des capteurs statiques, mais ceux-ci peuvent cependant devenir indisponibles, en raison d'une défaillance ou de la consommation totale de leur batterie. En dépit de ces défaillances, le réseau doit rester connexe, le système ne supportant pas le partitionnement (même temporaire) du réseau. Nous considérons également un canal de communication idéal (*i.e.* sans collision, ni perte de message). De plus, le temps de propagation d'un message entre deux nœuds du réseau doit être borné, et cette borne connue.

Chaque nœud connaît sa position géographique relative dans un système de coordonnées virtuelles, ainsi que la taille de ce système. Nous ne proposons pas dans ces travaux une méthode de construction de ce système de coordonnées, mais, étant donné un RCsF statique, les coordonnées relatives peuvent être calculées localement lorsque qu'un nœud rejoint le réseau, comme proposé dans [KMS+05, OW05]. Au demeurant, aucun ensemble pré-défini de types n'est requis dans notre système.

[1] Nous entendons ici, par *fiabilité*, la réussite d'une requête d'une primitive de *-*cast*, avec une réponse à jour.

Afin de découvrir son voisinage, tout nœud diffuse localement et périodiquement un message d'existence. Dans la nécessité d'obtenir une communication multi-saut, un protocole de routage géographique doit être utilisé sur le réseau. L'unique condition imposée sur ce protocole consiste en la nécessité qu'un nœud puisse envoyer des messages à un autre nœud, ne connaissant que sa position dans le système de coordonnées virtuelles. Dans le cadre de nos expérimentations, une version allégée du protocole GPSR [KK00] a été utilisée, permettant notamment de trouver efficacement le nœud le plus proche d'un point de coordonnées virtuelles données.

3.3 La collection *-*cast*

Une grande majorité des applications réparties repose sur les mêmes fonctionnalités. Plus spécifiquement, dans les RCsF, il est plus courant de contacter des nœuds en fonction de leur catégorie, ou de leur rôle, qu'à partir de leur identité. Cette spécificité, assimilable à l'identification d'un groupe de capteurs, plutôt que celle d'un capteur unique, est mise en œuvre par la structure de groupe de SOLIST. À un temps donné, chaque nœud possède un *type* représentant son état. Celui-ci peut être soit *statique* (si le réseaux est composé de capteurs hétérogènes, par exemple, chaque technologie de capteur sera lié à un identifiant particulier pour toute la durée de l'application), soit *dynamique* (pouvant présenter le niveau d'énergie restante ou les valeurs des données captées). Dans le premier cas, la valeur du type est fixée pour toute la durée de l'application, et dans le second, cette valeur peut changer dynamiquement au cours de l'exécution. De plus, un nœud peut ne pas avoir de type (temporairement ou non) et ne participera donc qu'au routage sur le réseau. SOLIST assure que tous les nœuds de même type sont regroupés ensemble dynamiquement.

Nous avons identifié les trois briques de base suivantes :

Primitive d'anycast ANYCAST($type$) ;

Cette primitive permet d'obtenir le contact d'un nœud parmi tous les nœuds d'un $type$ donné.

Par exemple, cette fonctionalité peut être utilisée afin de vérifier si au moins une instance d'un $type$ donné existe dans le système, et dans l'affirmative, de localiser une de celles-ci.

Primitive de k-cast KCAST($type, k$) ;

Cette primitive permet de contacter k nœuds d'un $type$ donné, s'il en existe suffisamment dans le réseau. Si le groupe contient plus de k nœuds, la primitive retournera VRAI*, ainsi que des informations optionnelles dépendantes de l'application, et* FAUX *sinon.*

Par exemple, un gestionnaire de stock a communément besoin de savoir si une marchandise est disponible en quantité suffisante. Une autre application classique des RCsF est la surveillance de feux de forêt. Il peut être utile de connaître si le réseau contient au moins 10 capteurs ayant mesuré une température supérieure à 40 degrés Celsius ou une hygrométrie inférieure à 2 %vol. Dans ce cas, une valeur moyenne sur un échantillon de nœuds d'un type donné pourrait être requise. La primitive k-cast répond idéalement à ces types de demande.

Primitive de broadcast BROADCAST($type$) ;

Cette primitive permet de contacter tous les nœuds d'un $type$ donné. Des informations optionnelles sur les noeuds de cet ensemble peuvent être attachées au message réponse, telles que le nombre de nœuds de ce $type$, la moyenne de leur valeur mesurée, etc.

Dans SOLIST, la primitive de broadcast diffuse un message uniquement aux nœuds du même $type$. Ce service peut donc être vu comme du *multicast*. Ce type de fonctionnalité peut être utilisé pour disséminer de l'information à un ensemble de nœuds de $type$ donné comme dans une application de publication/abonnement par exemple. Il peut également être utilisé pour comptabiliser le nombre d'entités d'un même $type$ disponibles dans un stock.

Chaque nœud possède deux primitives supplémentaires :

Rejoindre le système JOIN($type$) ;

Lors de l'arrivée d'un nœud dans un RCsF existant, celui-ci doit contacter un nœud appartenant déjà au réseau, et exécuter des tâches classiques d'initialisation, d'annonce de sa présence à son voisinage direct, de connexion à d'éventuelles structures, *etc.* Plus de détails sur cette primitive sont fournis dans la suite.

Quitter le système LEAVE($type$) ;

Lors du départ déterminé d'un nœud, celui-ci doit effectuer une procédure de départ afin d'éviter de potentielles incohérences de structure, de mettre à jour l'information sur ses voisins, *etc.* Si un nœud quitte le réseau sans exécuter cette procédure, il est considéré comme défaillant.

Dans le cas de types dynamiques, à chaque changement de type d'un nœud, ce dernier commencera par quitter le réseau afin de quitter son groupe actuel, puis de revenir avec un type mis à jour et rejoindre un groupe différent, le cas échéant.

3.4 Le cœur de SOLIST

Dans ce paragraphe, nous présentons l'architecture de SOLIST. L'implémentation technique de la collection **-cast* dans SOLIST est détaillée dans le paragraphe 3.5 suivant, page 64. La structure de SOLIST permet de mettre en œuvre cette collection combinant à la fois efficacité, fiabilité et économie d'énergie.

3.4.1 Une structure multi-couche

La conception de SOLIST repose sur une structure multi-couche, imaginée pour l'ensemble des primitives de **-cast*.

Premièrement, tous les nœuds font partie d'une couche de base, laquelle assure la connectivité globale du réseau et le routage géographique[2]. Cette couche est appelée *couche de routage* par la suite. Elle permet de contacter une destination quelconque en utilisant uniquement ses coordonnées virtuelles, introduites dans la partie 3.2, lesquelles représentent la position géographique d'une entité.

[2]Le protocole de routage utilisé dans notre implémentation est une version allégée de GPSR [KK00].

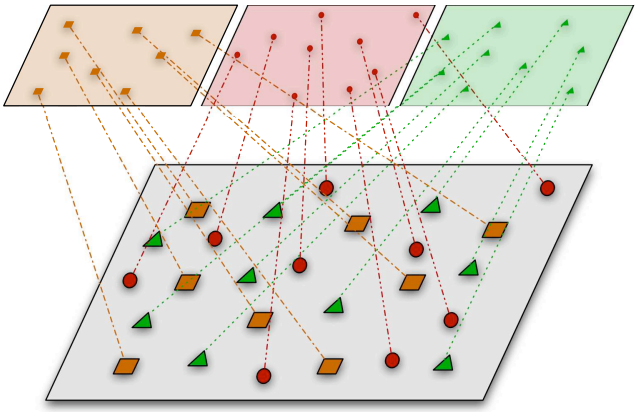

FIG. 3.1 – Projection en sur-couches

De plus, au-dessus de cette couche de routage, SOLIST met en œuvre différentes sur-couches logiques légères (une par groupe existant dans le réseau). La structure de ces sur-couches est présentée ci-après, en section 3.4.2. Cette structure multi-couche fournit un groupement logique des noeuds du réseau. Par exemple, dans la figure 3.1, n nœuds de trois types différents sont disposés dans un RCsF. À chacun des types données correspond une sur-couche spécifique (*i.e.* celle des triangles, des ronds et des carrés), au-dessus de la couche de routage. Un mécanisme de liaison inter-couche par *points d'entrée* est proposé au paragraphe 3.4.3.

3.4.2 Structure de couche : LIGH-t-LAYER

La structure de sur-couche à faible consommation énergétique dans SOLIST est réalisée par découpage rectangulaire de l'espace. Nous avons observé précédemment [BBFK07b] de nombreuses similitudes entre les RCsF et les systèmes pair-à-pair (PàP), en terme de propriétés et de fonctionnalités. Notre structure de sur-couche s'est donc naturellement inspirée des propositions de réseaux logiques structurés PàP, notamment les tables de hachage distribuées classiques telles que CAN [RFH+01]. Dans [CJK+03], Castro *et al.* propose une comparaison de celles-ci dans le contexte du *multicast*. La structure de CAN nous est apparue comme idéale pour les primitives identifiées des RCsF. CAN divise l'espace en zones logiques de responsabilité, réparties sur les différents nœuds du système. Chaque nœud possède alors un voisinage virtuel au sein de cette structure, correspondant aux nœuds responsables des zones adjacentes à celle du nœud donné. Inspirés par cette technique, nous avons modifié sensiblement la structure et la construction de CAN afin de l'adapter au contexte des RCsF. Les principales différences résident dans les contraintes de communication et d'économie d'énergie, qui doivent impérativement être prises en compte. Nous n'avons donc conservé que le découpage logique de l'espace. Les coordonnées virtuelles d'un nœud restent les mêmes quelle que soit la couche considérée[3] (couche de routage et sur-couche de groupe). Les zones de responsabilité d'un

[3]Les fonctionnalités de hachage et de répartition uniforme des nœuds par le calcul de nouvelles coordonnées logiques dans CAN sont coûteuses et ne sont pas nécessaires dans SOLIST.

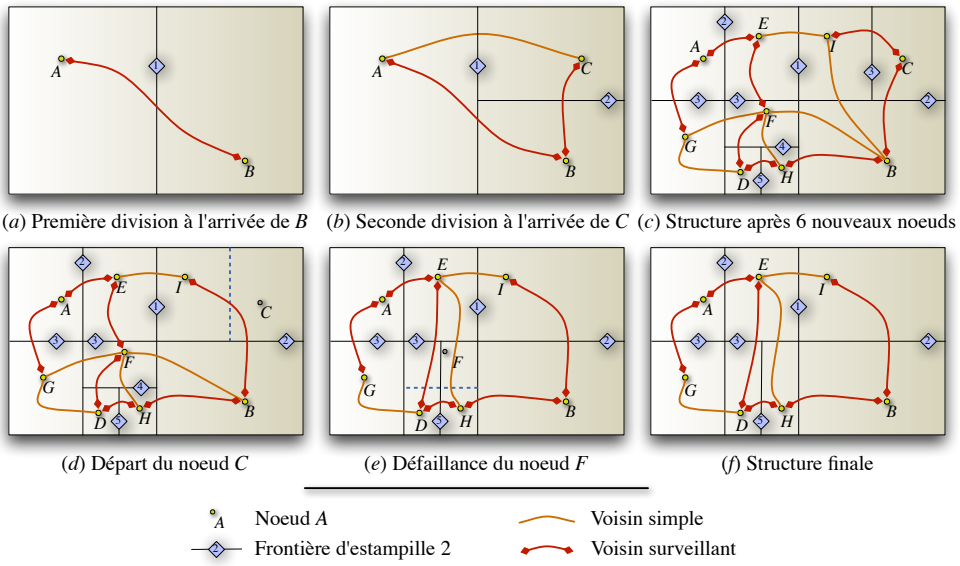

FIG. 3.2 – Évolution de la structure d'un LIGH-t-LAYER suivant différents évenements

nœud ne sont utilisées que pour l'obtention d'une implémentation efficace des primitives de k-cast et broadcast. Nous appellerons ces sur-couches LIGH-t-LAYER par la suite[4].

Ces sur-couches sont construites graduellement et dynamiquement. Le premier nœud d'un type t' donné rejoignant le réseau devient *responsable* de la totalité de l'espace du LIGH-t'-LAYER correspondant. A l'arrivée d'un autre nœud de type t' dans le réseau, ce dernier contacte le nœud responsable[5] de la zone contenant ses coordonnées, en utilisant un routage glouton dans le LIGH-t'-LAYER. Cette dernière zone est alors divisée en deux, chacun des deux nœuds devenant responsable de la partie correspondante à ses coordonnées, à l'instar de CAN. Une nouvelle frontière est alors créée entre les deux zones. Celle-ci sera estampillée par la valeur d'un compteur local, afin de conserver l'ordonnancement des créations de frontières. Cette estampille permet de réorganiser la structure en cas de départ ou de défaillance d'un nœud, comme indiqué ci-après.

Ainsi, chaque nœud appartenant à une sur-couche spécifique doit maintenir une liste de voisins appelée *vue* dans le LIGH-t-LAYER correspondant. Chaque entrée de cette vue contient (1) l'identifiant du voisin logique ; (2) ses coordonées dans SOLIST ; (3) les coordonnées des extrémités de la frontière commune ; et (4) l'estampille de cette frontière.

La figure 3.2 présente les différents cas d'évolution de la structure d'un LIGH-t-LAYER à partir de sa création. Le nœud A de type t a rejoint le réseau en premier, et il est donc responsable de la totalité de l'espace du LIGH-t-LAYER. Dans la figure 3.2.a, B contacte A, qui divise sa zone en deux et envoi à B les coordonnées de la zone de ce dernier ainsi que celles de leur frontière commune. Puis, à l'arrivée du nœud C, B divise à son tour sa propre

[4] t correspond à l'identifiant du type des nœuds regroupés dans cette sur-couche.
[5] Le mécanisme de recherche de celui-ci est décrit au paragraphe 3.4.3.

zone, et estampille la nouvelle frontière par un entier strictement plus grand, comme l'illustre la figure 3.2.*b*. Après plusieurs arrivées, la figure 3.2.*c* présente l'état de la structure du LIGH-t-LAYER avec 9 nœuds (identifiés de A à I). La plus grande estampille de frontière est 5.

La partie inférieure de la figure 3.2 présente comment conserver la cohérence de la structure en cas de départ (figure 3.2.*d*) ou de défaillance (figure 3.2.*e*). Un départ et une défaillance sont traités de la même façon par le système. Quand un nœud quitte le LIGH-t-LAYER (sur un départ définitif ou un changement de type), celui-ci envoie sa vue aux nœuds situés de l'autre coté de la frontière la plus récente. Ces derniers deviennent alors responsables de l'union des deux zones, mettent à jour leur propre vue (avec des nouveaux voisins potentiels ou des allongements de frontières) et finalement, informent les voisins concernés du changement d'état de la structure.

Par exemple, dans la figure 3.2.*d*, le nœud C doit quitter le LIGH-t-LAYER. C envoie donc sa vue à I, qui se situe derrière la frontière d'estampille 3 (3 étant la plus grande estampille). I devient donc responsable des deux zones (celles de I et de C). I met à jour sa vue en étendant sa frontière commune avec B, et, enfin, prévient ce dernier de cette modification.

La détection de défaillance est effectuée par un mécanisme de surveillance commune, selon les hypothèses du modèle, décrites dans la partie 3.2. Le temps maximal de propagation d'un message étant connu, il ne peut pas y avoir de fausse suspicion. À chaque étape d'arrivée dans le LIGH-t-LAYER, les deux nœuds impliqués dans la division d'une zone initient une surveillance symétrique. Périodiquement, chaque nœud du LIGH-t-LAYER envoi un message $jeSuisVivant$! à son (ou ses) surveillant(s). Si celui-ci n'en reçoit pas pendant un laps de temps déterminé, il vérifie l'état de son voisin par un message $toujoursEnVie$? S'il n'obtient toujours aucune réponse, le surveillant considère le nœud comme fautif et le supprime du LIGH-t-LAYER.

En cas de défaillance du nœud F comme dans la figure 3.2.*e* par exemple, celui-ci doit être retiré de SOLIST. F ne peut évidemment pas envoyer sa vue à un de ses voisins, comme le nécessite la procédure de départ volontaire. Dans ce cas, D, qui est un des nœuds surveillants de F, envoie un message $reconstitutionDeVue$ qui va tourner autour de la zone de F, afin de reconstruire la vue de ce dernier à partir des informations disponibles sur chacun de ses voisins. Avec cette vue ainsi reconstituée, D exécute la procédure standard de départ, contactant chaque nœud au-delà de la frontière la plus récente (dans ce cas, H et lui-même), et ceux-ci allongent leur zone avec la partie correspondante de l'ancienne zone de F. La figure 3.2.*f* présente l'état de la structure du LIGH-t-LAYER après l'exécution de ces départs.

3.4.3 Relier les mondes : les points d'entrées

Chaque LIGH-t-LAYER étant autonome et indépendant, en terme de fonctionnement, de chacun des autres LIGH-t-LAYERs ainsi que de la couche de routage, reste le problème de relier toutes ces couches les unes aux autres. Comment atteindre, à partir de n'importe quel nœud du RCsF, un LIGH-t-LAYER spécifique sans autre information que l'identifiant t du type voulu ?

Nous avons calqué l'espace de coordonnées virtuelles sur la couche de routage (*cf.* partie 3.2), et utilisons celui-ci comme proposé dans des travaux antérieurs [REG+02]. Nous utilisons deux fonctions de hachage communes à tous les nœuds du système. Le but de ces fonctions est de répartir des *points d'entrées virtuels* dans l'espace de coordonnées, uniformé-

ment, en fonction du nombre de type existant. Soit t un identifiant de type donné, pour lequel un nœud recherche le LIGH-t-LAYER correspondant ; $f_x : \mathbb{N} \mapsto [0;1[$ et $f_y : \mathbb{N} \mapsto [0;1[$ les deux fonctions de hachage ; et ws_x, ws_y la taille respectivement horizontale et verticale de l'espace de coordonnées. Les coordonnées du point d'entrée correspondant au LIGH-t-LAYER (nommés ep_t) sont calculés de la manière suivante :

$$ep_t = (x, y) : \begin{cases} x &= f_x(t) \times ws_x \\ y &= f_y(t) \times ws_y \end{cases} \tag{3.1}$$

Comme f_x et f_y sont communes à tous les nœuds du réseaux, les coordonnées des points d'entrées sont uniques pour tout type. Un point d'entrée est donc représenté par ses coordonnées, identiques pour tous les nœuds. Malheureusement, il est probable qu'aucun nœud physique ne se trouve exactement au point correspondant à ces coordonnées. Le protocole de routage géographique permettant de déterminer le nœud physique le plus proche d'un point donné dans l'espace virtuel (*cf.* partie 3.2), nous appellerons dès lors indistinctement *point d'entrée* le point de coordonnées ep_t obtenues par l'équation 3.1 et le nœud le plus proche de celui-ci. Ce dernier doit connaître au moins un nœud de type t (et donc, connaître l'existence du LIGH-t-LAYER). Si ce LIGH-t-LAYER existe, ce nœud doit être en mesure de donner l'identifiant et les coordonnées d'un nœud de ce LIGH-t-LAYER. Par la suite, les nœuds référencés sur les points d'entrées seront appelés *nœuds de contact*.

La disposition de ces points d'entrées dans l'espace peut amener un nœud excentré à envoyer une requête vers un point d'entrée situé à l'exact opposé du réseau. Afin d'éviter qu'un message n'ait à traverser entièrement le réseau et de diminuer la charge des nœuds les plus proches d'un point d'entrée, l'espace des coordonnées est divisé selon une grille : m_x divisions horizontales (par rapport à la coordonnée x) et m_y divisions verticales (par rapport à la coordonnée y). L'espace des coordonnées est alors calqué sur chacune de ces sous-divisions, appelés *cellules* par la suite, mais uniquement dans le cadre du calcul des coordonnées des points d'entrées. Lorsqu'un nœud doit atteindre un LIGH-t-LAYER, il envoie sa requête au point d'entrée le plus proche de lui, lequel lui renverra les informations d'un nœud de contact dans le LIGH-t-LAYER.

La figure 3.3 présente une topologie $m_x \times m_y = 3 \times 2$ contenant 5 types déclarés. Pour chaque cellule, les 5 points d'entrées ($et_1, \ldots, et_5$) sont répliqués à la même position relative. Dans ce cas, le nœud i émet une requête afin d'accéder au LIGH-1-LAYER (sur-couche du groupe de type 1). i connaît, grâce aux fonctions de hachages communes, les coordonnées des 6 points d'entrées (flèches pleine et pointillées), mais ne transmet sa requête qu'au plus proche d'entre eux (ici, $ep_{1(2)}$ représenté par une flèche pleine). Plus formellement, soit $d(\cdot,\cdot) : (\mathbb{R} \times \mathbb{R})^2 \mapsto \mathbb{R}$ la distance entre deux points de l'espace de coordonnées sur la couche de routage, et cs_x, cs_y la taille des cellules respectivement horizontales et verticales. L'équation 3.2 présente le calcul pour trouver le point d'entrée de type t le plus proche. Soit les deux ensembles suivants :

$$\begin{cases} \Gamma_{x,t} = \{(k_x + f_x(t)) \times cs_x | k_x \in [\![0; m_x - 1]\!]\} \\ \Gamma_{y,t} = \{(k_y + f_y(t)) \times cs_y | k_y \in [\![0; m_y - 1]\!]\} \end{cases}$$

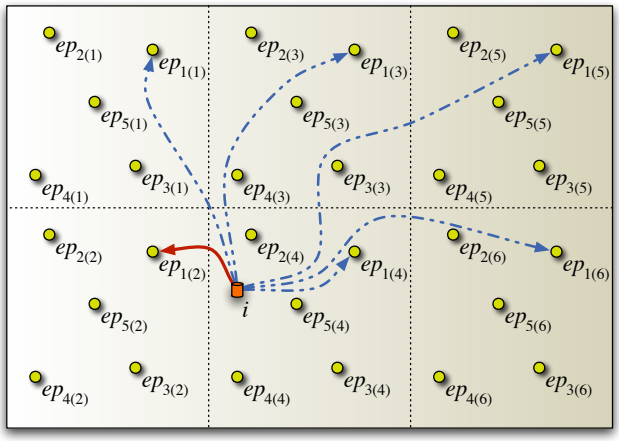

FIG. 3.3 – Exemple d'une division de l'espace en 3×2 cellules.
Le nœud i envoi sa requête de recherche de LIGH-1-LAYER *au point d'entrée le plus proche :* $ep_{1(2)}$.

Les coordonnées du point d'entrée le plus proche sont les suivantes :

$$ep_t = (x, y) \quad \text{tel que} \quad d(i, (x, y)) = \min_{\substack{x' \in \Gamma_{x,t} \\ y' \in \Gamma_{y,t}}} d(i, (x', y')) \tag{3.2}$$

Afin de mettre à jour dynamiquement les informations disponibles sur les points d'entrées, à chaque arrivée d'un nœud dans un LIGH-t-LAYER, celui-ci en informe le point d'entrée de type t le plus proche de sa position. Ainsi, ce point d'entrée connaît le nœud de contact le plus proche de lui, sans avoir à interroger le LIGH-t-LAYER. Cela permet à l'initiateur d'une requête d'obtenir un nœud de contact qui ne soit pas trop éloigné. De même, en cas de départ ou de défaillance, les points d'entrées sont informés de ce départ afin de mettre à jour leurs informations sur les nœuds de contact. Néanmoins, comme le choix du nœud de contact se fait localement sur chaque point d'entrée, un nœud n'est pas informé s'il a été choisi comme nœud de contact. Tout départ d'un type t donné doit donc être notifié à tous les points d'entrée de ce type.

3.4.4 Résumé des prérequis

Avant de poursuivre plus en avant vers les détails de conception des primitives de **-cast*, nous résumons dans ce paragraphe l'intégralité des informations qu'un nœud doit maintenir à jour immuablement :

- un identifiant unique (id) ;
- ses coordonnées dans l'espace virtuel (x_{id}, y_{id}), correspondant à sa position géographique dans l'espace considéré ;
- les dimensions de cet espace virtuel (ws_x, ws_y) ;
- son identifiant de type, s'il en est (t_{id}) ;

– deux fonctions de hachage mutuelles (f_x, f_y) ainsi qu'une fonction de distance d, afin de calculer la position des points d'entrées ;
– la distribution des cellules divisant la couche de routage (m_x, m_y)[6].

De plus, si un nœud est régi par un type t particulier, celui-ci doit maintenir en sus une vue $view_t$ à jour (*cf.* paragaphe 3.4.2 pour plus de détails).

Enfin, si un nœud est un point d'entrée du type t', il doit toujours avoir connaissance de l'identifiant et des coordonnées dans l'espace virtuel d'un nœud appartenant au LIGH-t'-LAYER correspondant.

3.5 La collection *-*cast* dans SOLIST

À partir de la structure de SOLIST, il est trivial d'implémenter la collection *-*cast*. Nous présentons dans cette section les détails d'une implémentation optimisée pour préserver l'énergie des capteurs en réduisant le plus possible le nombre de messages redondants.

3.5.1 Anycast

Un nœud génère une requête d'anycast afin de contacter *un* nœud quelconque du type donné. Ainsi, le mécanisme de recherche d'un LIGH-t-LAYER, présenté au paragraphe 3.4.3, peut être immédiatement appliqué pour répondre aux besoins de cette primitive. Celui-ci retourne à l'émetteur de la requête, soit l'identifiant et les coordonnées d'un nœud de contact, appartenant au LIGH-t-LAYER, soit `NOT_FOUND` si ce dernier n'existe pas sur le réseau.

Étant donné que ce mécanisme met continuellement à jour l'information des nœuds de contact localisée sur les points d'entrées, une réponse à une requête d'anycast correspond à un nœud de contact proche du point d'entrée. Par conséquent, plus les cellules sont de dimension réduite, plus proche sera le nœud de contact transmis. Il est toutefois nécessaire d'identifier les quelques cas de figure apportant une réponse périmée, et ainsi caractériser la fiabilité du mécanisme. En effet, à l'arrivée du premier nœud d'un type donné sur le réseau, celui-ci doit informer tous les points d'entrées correspondant de la création d'un nouveau LIGH-t-LAYER. Ainsi, durant la période de propagation de l'information sur tous les points d'entrées, un de ceux-ci, non-encore informé de la création, peut potentiellement répondre un *faux négatif*. À l'opposé, lors du départ du dernier nœud d'un LIGH-t-LAYER, l'information de destruction de ce dernier à tous les points d'entrées peut entraîner l'existence de *faux positif*. Néanmoins, ces deux cas restent rares et peuvent être détectés par l'insertion d'un délai à chaque requête. En effet, la période durant laquelle une occurrence de ces *faux* peut advenir est éphémère et bornée par le produit du diamètre du système avec la borne temporelle de transmission d'un message (*cf.* paragraphe 3.2).

Nous présentons le *pseudo-code* de ce mécanisme dans l'algorithme 3.1.

3.5.2 Broadcast

La primitive de broadcast garantit une diffusion générale de l'information dans un LIGH-t-LAYER. À l'initialisation d'une diffusion, le nœud initiateur émet préalablement une instance

[6]La dimension des cellules cs_x et cs_y est extrapolée simplement par respectivement ws_x/m_x et ws_y/m_y.

Algorithme 3.1 : ANYCAST(t)

Données : Toute information présentée au paragraphe 3.4.4
Résultat : Un couple $(id_t, (x_{id_t}, y_{id_t}))$

```
[ Calcul des coordonnées des points d'entrées                    ]
```

1 $(x_e, y_e) \leftarrow$ **nil**;
2 $d_e \leftarrow ws_x \cdot ws_y$;
3 $(x_h, y_h) \leftarrow (f_x(t), f_y(t))$;
4 **pour** *i = 0* **à** *m - 1* **faire**
5   **pour** *j = 0* **à** *n -1* **faire**
6         $(x_{tmp}, y_{tmp}) \leftarrow ((i + x_h) \cdot cs_x, (i + y_h) \cdot cs_y)$;
7   **si** $d((x_{id}, y_{id}), (x_{tmp}, y_{tmp})) < d_e$ **alors**
8             $(x_e, y_e) \leftarrow (x_{tmp}, y_{tmp})$;
9             $d_e \leftarrow d((id_x, id_y), (x_e, y_e))$;

```
[ Contacter le point d'entrée et attendre                        ]
```

10 **Envoyer** (`Anycast`, *t*) **à** (x_e, y_e);
11 **retourner** *réponse* **de** (x_e, y_e);

d'anycast afin de localiser un nœud de contact de type t, appartenant par conséquent au LIGH-t-LAYER, et transmet à ce dernier la requête de broadcast.

Dans SOLIST, nous tirons profit de la structure des LIGH-t-LAYERs afin de limiter la redondance des messages émis. Ainsi, nous garantissons qu'aucun nœud ne reçoit deux fois le même message d'une requête donnée. Les zones de responsabilité étant de forme rectangulaire, nous désignons les frontières en fonction de leur direction par rapport au nœud considéré (*Nord*, *Sud*, *Est* et *Ouest*).

La figure 3.4 illustre le comportement de propagation d'un message de broadcast dans un LIGH-t-LAYER. Contrairement à l'algorithme initial de broadcast dans CAN, lequel entraîne une redondance de messages [RHKS01], cet algorithme assure qu'un nœud est contacté une seule et unique fois. Cette condition limite intrinsèquement la résistance aux défaillances de notre algorithme. Cependant, étant donné le traitement pro-actif des défaillances dans SOLIST, la perte d'un message au cours d'une diffusion de broadcast reste rare. L'algorithme 3.2 de broadcast accomplit les étapes suivantes :

[Étape 1] Le nœud de contact (dans notre exemple, le nœud A) transmet le message dans les quatre directions, respectivement au :

Nord et Sud *À chaque nœud localisé dans l'alignement de la frontière correspondante.*
En figure 3.4, A transmet le message à L. Cependant, A ne le transmet pas à F étant donné que celui-ci n'est pas dans l'alignement de leur frontière commune. Cette notion d'alignement signifie qu'un nœud est visible sur la largeur de la frontière (*e.g.* considérons la sous-zone de F correspondant à l'intersection avec l'extension virtuelle de celle de A vers le sud : F n'est pas situé dans cette sous-zone, et n'est donc pas dans l'alignement de la frontière du point de vue de A, mais l'est du point de vue de G) ;

Est et Ouest *À exactement un nœud pris au hasard dans chacune de ces directions.*
Sur la figure 3.4, du coté *est*, A a le choix entre C et J pour transmettre le message. A choisi donc arbitrairement d'envoyer le message à J. De l'autre coté (*i.e.* ouest), A ne

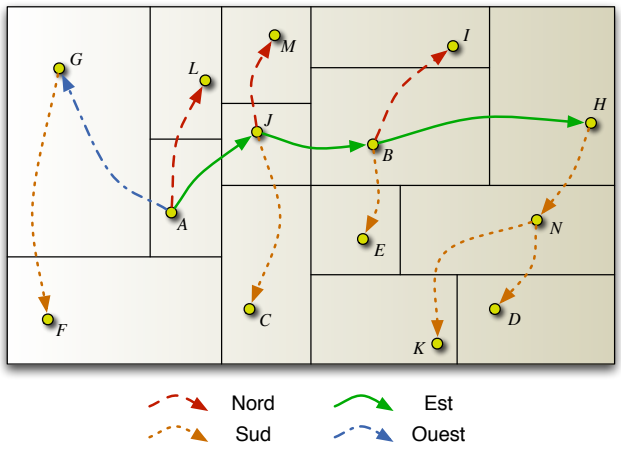

FIG. 3.4 – Instance de broadcast dans un LIGH-t-LAYER.

possède qu'un voisin et n'a donc pas d'autre choix que de transmettre le message à G.

[Étape 2] Chaque nœud recevant le message le retransmet au :

Nord *À chaque nœud localisé dans l'alignement de la frontière s'il l'a reçu en provenance du sud, de l'est ou de l'ouest.*
Representé par les flèches cadratines sur la figure 3.4 ;

Sud *À chaque nœud localisé dans l'alignement de la frontière s'il l'a reçu en provenance du nord, de l'est ou de l'ouest.*
Representé par les flèches pointillées sur la figure 3.4 ;

Est *À exactement un nœud pris au hasard s'il l'a reçu en provenance de l'ouest.*
Representé par les flèches pleines sur la figure 3.4 ;

Ouest *À exactement un nœud pris au hasard s'il l'a reçu en provenance de l'est.*
Representé par les flèches pointillées-cadratines sur la figure 3.4.

L'algorithme 3.2 présente en détail le pseudo-code du comportement sus-cité. Dans cette représentation, N, S, E et O symbolisent respectivement les directions du Nord, Sud, Est et Ouest. La fonction $estAligné(\cdot)$ retourne une valeur booléenne traduisant si le nœud considéré est dans l'alignement de la frontière commune, et la fonction $loc(\cdot)$ la direction relative d'un voisin donné et $choisirUn(\cdot)$ un élément tiré aléatoirement dans l'ensemble passé en paramètre. Enfin, $majObjet(\cdot)$ est une fonction fournie par l'application, permettant de mettre à jour l'information à inclure dans le message retour, le cas échéant.

3.5.3 k-cast

Dans l'optique d'implémenter la fonctionnalité de k-cast dans SOLIST, deux possibilités peuvent être envisagées, par extension de l'algorithme de broadcast non-redondant ci-avant. La première consiste en une approche *probabiliste*, par émission de sous-requêtes j-cast parallèles dans tous les directions, avec $j < k$. Par cette approche, il est possible d'obtenir des

Algorithme 3.2 : BROADCAST(t)

Données : Nœud transmetteur id_f et informations du paragraphe 3.4.4
Résultat : Un objet o, le cas échéant

```
[ Construction de l'ensemble des voisins auxquels transférer    ]
Γ ← ∅;
pour chaque n ∈ view_t[N] faire
    si estAligné(n) et loc(id_f) ≠ N alors
        Γ ← Γ ∪ {n};
pour chaque n ∈ view_t[S] faire
    si estAligné(n) et loc(id_f) ≠ S alors
        Γ ← Γ ∪ {n};
si loc(id_f) = E alors
    Γ ← Γ ∪ {choisirUn(view_t[O])};
si loc(id_f) = O alors
    Γ ← Γ ∪ {choisirUn(view_t[E])};
[ Transmission du message                                        ]
Envoyer (Broadcast, t) à Γ;
[ Optionel : Mise à jour et envoi de la réponse                  ]
pour chaque réponse d'un nœud de Γ faire
    majObjet(o);
Envoyer (BroadcastRéponse, o) à id_f;
```

faux négatifs, j étant déterminé probabilistiquement. En conséquence, nous lui avons préféré une approche *déterministe*, permettant d'obtenir de meilleurs résultats en terme de fiabilité et d'économie d'énergie. En effet, avec cette approche, au plus k nœuds différents sont atteints. Néanmoins, cette méthode implique un surcoût de délai dû au séquencement de la propagation.

L'algorithme de k-cast proposé consiste en un parcours en profondeur dans le graphe logique du LIGH-t-LAYER. L'ordre de traitement des directions est donné par l'heuristique suivante : Nord → Sud → Ouest → Est.

Considérons à nouveau l'arbre de diffusion de la Figure 3.4. Le nœud A reçoit une requête de 10-cast. En premier lieu, il envoie donc un message de 9-cast message à L et attend la réponse de ce dernier (une requête de 9-cast suffit étant donné que A se compte lui-même). Étant donné que L n'a pas de voisins au nord, il retransmet directement sa réponse à A en prenant soin de faire décroître la valeur de k à 8. Comme A ne possède pas de voisin dans l'alignement de frontière au sud, il transmet donc sa requête à travers sa frontière ouest, en l'occurrence au nœud G, et patiente jusqu'à obtention de la réponse. G n'a pas de voisin au nord, mais en possède un dans l'alignement de frontière au sud. Il lui envoie donc le message et ainsi de suite. La requête 10-cast issue de A progresse donc comme suit :

$$A \xrightarrow{k=9} L \xrightarrow{k=8} A \xrightarrow{k=8} G \xrightarrow{k=7} F \xrightarrow{k=6} G \xrightarrow{k=6} A$$
$$\xrightarrow{k=6} J \xrightarrow{k=5} M \xrightarrow{k=4} J \xrightarrow{k=4} C \xrightarrow{k=3} J \xrightarrow{k=3} B$$
$$\xrightarrow{k=2} I \xrightarrow{k=1} B \xrightarrow{k=1} E \xrightarrow{k=0} B \xrightarrow{k=0} J \xrightarrow{k=0} A$$

Algorithme 3.3 : KCAST(t)

Données : Nœud transmetteur id_f, paramètre k et informations du paragraphe 3.4.4
Résultat : Un objet o, le cas échéant

```
[ Construction de l'ensemble des voisins auxquels transférer ]
1  Γ ← ∅;
2  pour chaque n ∈ view_t[N] faire
3      si estAligné(n) et loc(id_f) ≠ N alors
4          Γ ← Γ ∪ {n};
5  pour chaque n ∈ view_t[S] faire
6      si estAligné(n) et loc(id_f) ≠ S alors
7          Γ ← Γ ∪ {n};
8  si loc(id_f) = E alors
9      Γ ← Γ ∪ {choisirUn(view_t[O])};
10 si loc(id_f) = O alors
11     Γ ← Γ ∪ {choisirUn(view_t[E])};
[ Transmission du message à k nœuds ]
12 Γ' ← Γ;
13 tant que k > 0 et Γ ≠ ∅ faire
14     n ← choisirUn(Γ) selon l'ordre N, S, O, E;
15     Envoyer (Kcast, k) à n;
16     Γ ← Γ − {n};
17     k ← Recevoir k de n;
[ Optionel : Mise à jour et envoi de la réponse ]
18 pour chaque réponse d'un nœud de Γ' faire
19     majObjet(o);
20 Envoyer (KcastRéponse, k, o) à id_f;
```

À la réception de la réponse de J, A peut désormais transmettre un message `True` à l'initiateur de la requête 10-cast. Les derniers *aller-retour* entre B et ses voisins peuvent être évités s'il n'est pas nécessaire de joindre physiquement les nœuds (B sachant qu'il a plus de 3 voisins).

L'algorithme 3.3 présente le pseudo-code de cette primitive, exécuté par chaque nœud contacté du LIGH-t-LAYER. Le paragraphe 3.5.2 introduit la définition des fonctions utilisées dans cet algorithme.

3.6 Évaluation des performances

Dans cette partie, nous présentons au préalable l'environnement de simulation, puis les protocoles utilisés à des fins de comparaison et enfin, l'évaluation de SOLIST.

3.6.1 Environnement de simulation

Nous avons simulé avec SeNSim [BBT08] plusieurs topologies de réseaux différentes, respectant le modèle défini préalablement (*cf.* paragraphe 3.2). Nous avons utilisé une version

allégée du protocole de routage géographique GPSR : le calcul des *graphes planaires* étant trop coûteux, et désireux d'évaluer le coût énergétique de SOLIST, notre version de GPSR ne contient que le routage glouton avec la *règle de la main droite* en présence d'un trou de densité sur une route[7].

Un exemple de topologie utilisée est présenté en figure 3.13 (page 74) : un réseau de 1 000 nœuds, répartis parmi 10 types statiques (10 nœuds de type 1, 30 de type 2, 50 de type 3, ..., 190 de type 10), dans un espace de 96×96 mètres carrés. Cette instance de topologie à répartition aléatoire et uniforme des nœuds est considérée comme référence dans la suite pour tout résultat non moyenné. En outre, cette répartition uniforme s'avère être le pire des cas dans le contexte de SOLIST. En effet, dans le cas extrême où tous les nœuds du même type sont situés dans la même région de l'espace, la propagation des requêtes reste localisée. Ici, dans le cas d'une distribution aléatoire uniforme, les requêtes doivent le plus souvent traverser tout l'espace, et donc entraînent un surcoût énergétique (la couche de routage investissant tous les nœuds quel que soit leur type). Pour chacun des scénarios présentés ci-après, chaque nœud arrive dans le réseau, émet une requête de k-cast et une de broadcast (et donc, par définition, 3 requêtes d'anycast), avant de finalement quitter le réseau. Les dates d'arrivée, de départ ou de requête sont générées aléatoirement, ainsi que les valeurs de t et de k (ces dernières sont piochées respectivement dans $[1; 12]$ et $[1; 200]$ afin d'obtenir des résultats hétérogènes).

3.6.2 Présentation des algorithmes de comparaison

Nous avons utilisés deux protocoles classiques, souvent utilisés dans les RCsF, afin de comparer deux des trois membres de la collection *-cast : la *marche aléatoire* pour la primitive anycast, et l'*inondation* pour celle de broadcast. Divers autres algorithmes de broadcast ont été proposés tels que LPBcast [EHG+03] mais ceux-ci sont probabilistes, contrairement à celui proposé dans ce chapitre. De plus, nous ne proposons pas de comparatif pour la primitive de k-cast étant donné qu'aucun algorithme simple sur des réseaux non-structurés n'a été proposé auparavant pour cette fonctionnalité. Présentons succinctement les deux algorithmes comparatifs.

Marche aléatoire L'initiateur de la requête d'anycast la transmet à l'un de ses voisins, choisi aléatoirement (*i.e.* dans le contexte des RCsF, un nœud situé dans son rayon de transmission). Si ce nœud est une entité du type recherché, celui-ci renvoie alors directement sa position. Dans le cas contraire, il choisit un autre nœud aléatoirement dans sa propre vue, à l'exception de l'émetteur initial, et lui transmet la requête. Le message se transmet de nœud en nœud de cette manière jusqu'à trouver un nœud du type considéré. Si, après un nombre prédéterminé de retransmissions, aucun n'est trouvé, le dernier receveur envoie un message `NOT_FOUND` à l'initiateur de la requête.

Inondation Mécanisme classique des systèmes répartis non-structurés, celui-ci consiste à transmettre le message reçu à tous les nœuds de son voisinage. Comme une multiple réception peut advenir, un nœud donné ne retransmet le message qu'une unique fois. Ainsi,

[7] Le message contourne le trou dans le sens trigonométrique jusqu'à reprendre une approche gloutonne dès que possible – *cf.* [KK00] pour plus de détails.

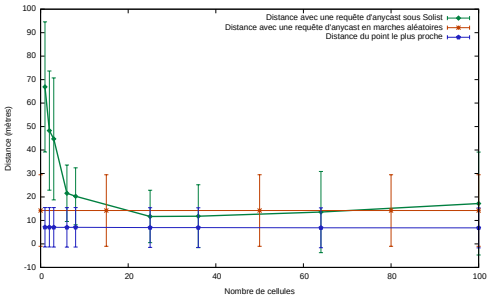

FIG. 3.5 – Distance moyenne entre l'initiateur et le nœud de contact d'une requête d'anycast.

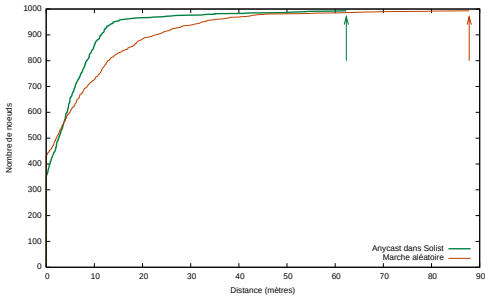

FIG. 3.6 – FR de la distance moyenne entre l'initiateur et le nœud de contact d'une requête d'anycast pour une topologie à 5×5 cellules.

un arbre de diffusion est construit dynamiquement et peut-être utilisé pour agréger de l'information si une réponse est requise (ceci est en dehors du contexte de ce document).

Nous évaluons SOLIST selon deux grands axes, présentés dans les paragraphes suivants : la fiabilité des primitives de la collection *-cast, et la consommation d'énergie au paragraphe 3.6.4.

3.6.3 Fiabilité de SOLIST

Considérons en premier lieu la correction et la précision des réponses fournies par chacune des fonctionnalités de SOLIST.

Évaluation de l'anycast La qualité de la primitive d'anycast se mesure selon plusieurs critères : la distance entre l'émetteur d'une requête et le nœud de contact répondu, *a fortiori* la distance avec le point d'entrée, ainsi que la précision de la réponse. Tout d'abord, la figure 3.5 présente la distance moyenne et l'écart type de celle-ci entre l'émetteur de la requête et le nœud de contact correspondant, en fonction du nombre de cellules dans SOLIST. La borne inférieure (*Distance du point le plus proche*) correspond à la distance moyenne entre l'émetteur de la requête et le nœud le plus proche du type requis. La distance moyenne par l'utilisation de marches aléatoires est également présentée ici. Pour de grande cellules dans SOLIST, correspondant à un nombre restreint de cellules, les nœuds de contact sont situés assez loin de l'émetteur, ceux-ci étant les plus proches des points d'entrée et non de l'émetteur. Mais, plus la taille des cellules diminue, plus les nœuds de contact sont proches. Un effet de bord peut être observé pour un très grand nombre de cellules, la courbe d'anycast dans SOLIST remontant légèrement. En effet, lorsqu'un nœud rejoint un LIGH-t-LAYER, celui-ci informe le point d'entrée le plus proche de son arrivée. À partir d'une forte diminution de la taille des cellules, d'autres points d'entrées peuvent être plus proche de ce nouveau nœud que de tous les autres de ce LIGH-t-LAYER, mais ils ne seront pas informés de cette arrivée. Contrairement à la création et à l'élimination d'un LIGH-t-LAYER dont les occurrences sont éparses, nous avons préféré ne pas inonder les points d'entrées à chaque arrivée, afin de limiter la consommation d'énergie globale du système.

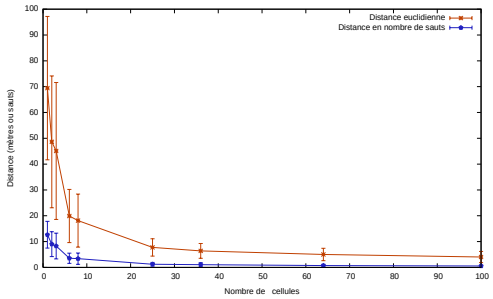

FIG. 3.7 – Distance moyenne entre l'initiateur et le point d'entrée correspondant.

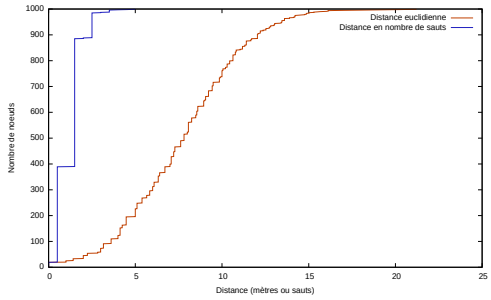

FIG. 3.8 – FR de la distance moyenne entre l'initiateur et le point d'entrée pour une topologie à 5×5 cellules.

La figure 3.6 présente la fonction de répartition (*FR* ou *CDF* pour *Cumulative Distribution Function* chez les anglo-saxons) de cette distance sus-citée, considérant SOLIST avec une topologie à 5×5 cellules et une approche par marches aléatoires. Ainsi, pour ces simulations, plus de 95 % des nœuds dans le contexte de SOLIST se situent à une distance inférieure à 13,5 mètres, contre seulement 80 % pour l'approche comparative. De surcroît, le queue de la FR pour SOLIST révèle une distance maximale de 62,5 mètres, tandis que celle relative aux marches aléatoires se situe autour de 89 mètres, correspondant approximativement à la largeur de l'espace de simulation.

D'autre part, afin de comparer un second critère de l'anycast dans SOLIST, la figure 3.7 présente la distance moyenne, et l'écart type de cette dernière, entre tout nœud du système et le point d'entrée le plus proche, en fonction du nombre de cellules dans SOLIST. La distance euclidienne n'est pas seulement représentée comme en figure 3.5 et 3.6, mais accompagnée du nombre de sauts nécessaires pour contacter un point d'entrée. À l'instar des conjectures préalables, plus les cellules sont petites, plus proches sont les points d'entrées. Cette figure montre que pour les topologies données, à partir de 6 cellules dans SOLIST, le nombre moyen de sauts est inférieur à 3, illustrant la basse consommation d'énergie nécessaire à l'obtention d'un nœud de contact. Comme précédement, la figure 3.8 présente la FR de ces deux distances pour une topologie à 5×5 cellules. 1,5 sauts[8] seulement sont nécessaires pour aboutir sur 90 % des points d'entrées, et au plus 2,5 sauts pour 98.5 %. Pour cette expérimentation, chaque cellule à une taille de $19,2 \times 19,2$ mètres carrés. Aucun nœud n'est donc situé à une distance supérieure à $\sqrt{2} \times 19,2$ mètres d'un point d'entrée (la diagonale d'une cellule), mais certains se répartissent entre dans l'intervalle $[\frac{\sqrt{2} \times 19,2}{2}; \sqrt{2} \times 19,2]$: ces nœuds correspondent à ceux situés sur la bordure du réseau. Ceux-ci ont effectivement un choix moindre de points d'entrées que les nœuds centraux, lesquels ont autour au moins quatre points d'entrées du même type.

Évaluation du broadcast et du k-cast Concernant les primitives de broadcast et k-cast, la fiabilité des résultats est simple à analyser, correspondant à un comportement binaire : si une requête entraîne l'obtention d'une réponse, alors respectivement tous ou k nœuds ont été contac-

[8] Les valeurs sont calculées pour un message aller-retour, puis divisé par 2 afin de prendre en compte les chemins non-symétriques entre deux nœuds, dus à GPSR.

Opération	nAh
Transmission d'un paquet	20,000
Reception d'un paquet	8,000
Écoute sur le médium radio durant 1 milliseconde	1,250
Lecture de données sur mémoire Flash	1,111
Écriture/Suppression de données sur mémoire Flash	83,333

TAB. 3.1 – Coût énergétique typique de diverses opérations sur un nœud Mica© [MCP+02].

tés. Sinon, aucune réponse ne sera renvoyée. Ce dernier cas n'apparaît que rarement : uniquement si le LIGH-t-LAYER est en cours de maintenance (départ ou défaillance d'un nœud). Considérant le système en régime stable, le taux de succès d'une requête de k-cast ou de broadcast est de 100 % au vu des caractéristiques du système présentées au paragraphe 3.2. En cas de maintenance d'un LIGH-t-LAYER, l'avancement de la primitive sera retardé en attendant un retour en régime stable. Le seul cas d'échec d'une requête advient donc uniquement lorsqu'un nœud est défaillant et que son nœud surveillant ne l'a pas encore détecté. Dans ce cas, la requête sera transférée et la réponse attendue indéfiniment. Un délai de garde, ou des accusés de réception de requêtes, peuvent être instaurés pour éviter ce cas de figure. Ces optimisations n'ont pas été incluses dans les simulations présentées dans ce chapitre.

3.6.4 Consommation d'énérgie

Nous nous intéressons maintenant à la consommation d'énergie nécessaire au fonctionnement de SOLIST, comparée avec celle nécessaire à l'utilisation de marches aléatoires et d'inondation. Afin de comparer équitablement ces approches, nous avons effectué les simulations en imitant une consommation réelle d'énergie, proposé dans [MCP+02], dont les chiffres, issus de mesures sur des capteurs MICA réels, sont fournis dans le tableau 3.1. Chaque capteur possède au début de l'expérimentation une batterie chargée de capacité 2 200 mAh.

Afin de comparer équitablement les approches considérées, chaque simulation est lancée à deux reprises : (*i*) avec SOLIST et (*ii*) avec les algorithmes comparatifs, et ce, avec le même comportement des nœuds (*i.e.* égalité des temps d'arrivées, de départs, de défaillances, du nombre d'évènements, *etc.*) sur une période de 10 000 temps discrets. Comme chaque introduction de nœud, ou requête de **-cast*, nécessite un anycast, ces simulations, présentées plus en détail au paragraphe 3.6.1, génèrent 3 000 requêtes d'anycast, 1 000 de broadcast et 1 000 de k-cast, sans compter l'énergie comptabilisée pour la maintenance de la structure.

En premier lieu, intéressons nous à la consommation totale du système au cours d'une simulation. La figure 3.9 présente, pour différents nombres de cellules dans SOLIST, la consommation d'énergie moyenne en fin de simulation. Cette figure illustre l'efficacité de SOLIST en terme d'économie d'énergie, comparativement aux marches aléatoires combinées avec l'inondation. Néanmoins, la courbe de consommation de SOLIST remonte sensiblement pour un grand nombre de cellules. Ceci est dû à la croissance de l'énergie nécessaire à la maintenance du système, comme nous le verrons ci-après. À titre d'exemple, les figures 3.15 et 3.16 présentent l'état du réseau en terme d'énergie consommée à la fin d'une simulation, sur la topologie présentée en figure 3.13, respectivement pour une utilisation de SOLIST avec une

configuration à 2×4 cellules et pour une utilisation de marches aléatoires et inondations. Elles permettent de se rendre compte de l'équilibrage de charge sur le réseau. Toutes les figures de ce type présentent les courbes de niveaux d'énergie en fin de simulation, exprimé en pourcentage consommé. Plus les couleurs sont sombres, dans le spectre des couleurs froides (bleu à violet), moins ces nœuds ont été sollicités, et plus leur quantité d'énergie restante est importante. À l'inverse, plus les couleurs sont claires et dans le spectre des couleurs chaudes (rouge à jaune), plus la consommation d'énergie a été importante durant la simulation. Ainsi, il est visible que la majorité des nœuds souffre d'une pénurie d'énergie sur la figure 3.16. *A contrario*, l'utilisation de SOLIST sur le même scénario, avec 1 000 requêtes de k-cast supplémentaires, illustre une consommation d'énergie maximale inférieure à 33 %. Nous pouvons toutefois observer des zones de forte densité. Les nœuds de ces zones ont en effet été plus sollicités, en raison d'un plus grand nombre de messages ayant transités – et été générés – par ceux-ci au cours de la simulation.

La figure 3.10 présente, pour les mêmes configurations que précédemment, la moyenne de l'énergie spécifiquement consommée par l'utilisation de la primitive d'anycast. Celle-ci montre l'intérêt de SOLIST à partir d'une configuration à 3 cellules, en comparaison avec les marches aléatoires. Néanmoins, avec une configuration à très petites cellules, le nœud de contact reçu ne correspond pas forcement au plus proche (*cf.* paragraphe 3.6.3), rehaussant légèrement la courbe énergétique en fonction du nombre de cellules. Cependant, l'énergie nécessaire pour l'anycast dans SOLIST reste inférieure à 35 % de celle utilisée en moyenne avec des marches aléatoires. Utilisant l'exemple de la figure 3.13, les figures 3.17 et 3.18 présentent la consommation spécifique en fin de simulation de l'anycast issu de SOLIST *versus* des marches aléatoires. Sur celles-ci, plusieurs zones de forte consommation apparaissent. Pour SOLIST, ces points correspondent aux 80 points d'entrées du système, pour lesquels les huit cellules se découpent clairement. À l'opposé, les points de forte consommation des marches aléatoires se rapportent aux zones de haute concentration de nœuds dans le système. Cette observation s'explique par la forte probabilité pour la requête de se perpétuer dans une zone à forte densité[9]. Au demeurant, la consommation de celles-ci reste proportionnellement prépondérante à celle de SOLIST.

Concernant la primitive de broadcast, la figure 3.11 présente l'énergie consommée dans les mêmes configurations de SOLIST, comparativement à l'utilisation de l'inondation. Etant donné que la structure des LIGH-t-LAYERS n'est pas corrélée avec le nombre de cellules dans SOLIST, la consommation d'énergie moyenne est constante pour chaque courbe. Comme attendu, cette figure illustre l'intérêt de ne contacter que les nœuds nécessaires à la diffusion sur un type donné plutôt que d'impliquer tous les nœuds du réseau. Plus spécifiquement, les figures 3.19 et 3.20 présentent les cartes de consommation d'énergie pour le broadcast dans SOLIST et l'inondation. Pour SOLIST, seulement certains nœuds de types spécifiques, correspondant à ceux les plus contactés, montrent une consommation supérieure aux autres (*cf.* les points clairs sur la figure 3.19). En revanche, une utilisation de l'inondation implique la totalité du réseau. Dans le cas où le broadcast serait appliqué sur différents types, l'efficacité énergétique de l'émission de plusieurs requêtes dans SOLIST persiste par rapport à une seule procédure d'inondation. Toutefois, si un broadcast intégral est requis, pour *tous* les nœuds du système, une dissémination

[9]Les marches aléatoires ont une faible probabilité de sillonner les voisinages à faible densité.

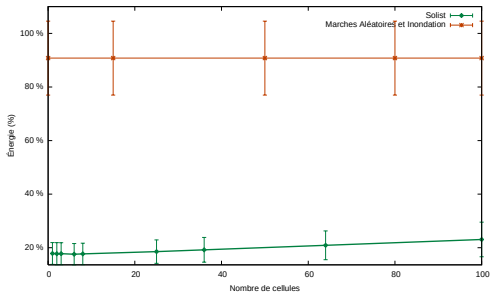

FIG. 3.9 – Consommation d'énergie moyenne intégrale.

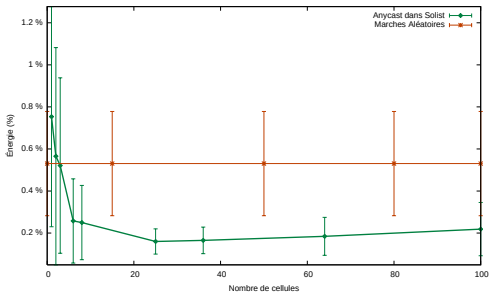

FIG. 3.10 – Consommation d'énergie moyenne pour l'anycast exclusivement.

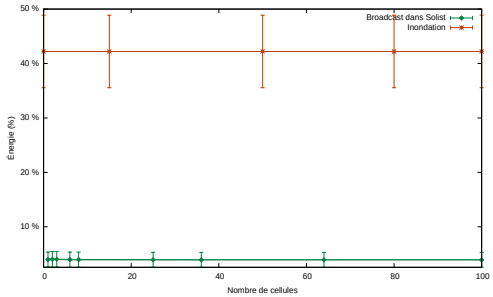

FIG. 3.11 – Consommation d'énergie moyenne pour le broadcast exclusivement.

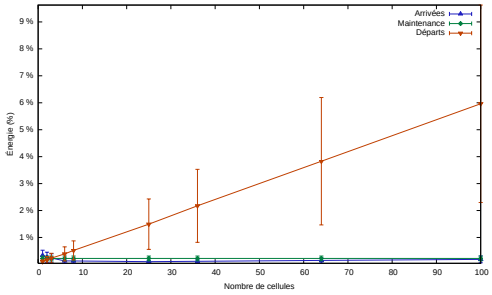

FIG. 3.12 – Consommation d'énergie moyenne pour l'entretien de la structure.

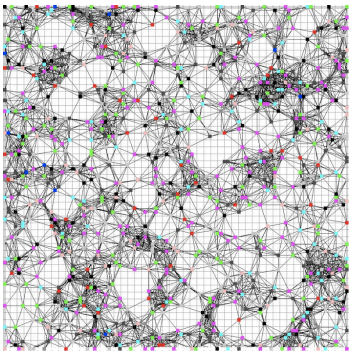

FIG. 3.13 – Exemple de référence d'un réseau à 1000 nœuds et 10 types.

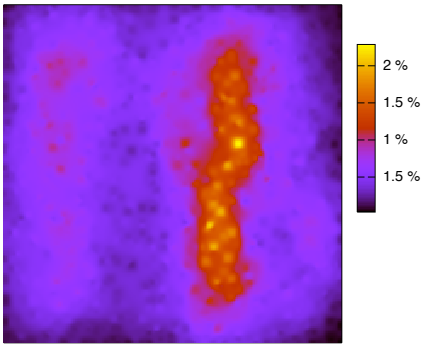

FIG. 3.14 – Consommation d'énergie totale pour le k-cast dans SOLIST.
La topologie se réfère à celle présentée en figure 3.13 avec 2×4 *cellules.*

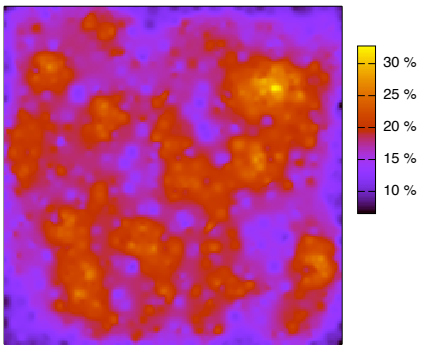

FIG. 3.15 – Consommation d'énergie totale intégrale avec SOLIST.
La topologie se réfère à celle présentée en figure 3.13 avec 2×4 cellules.

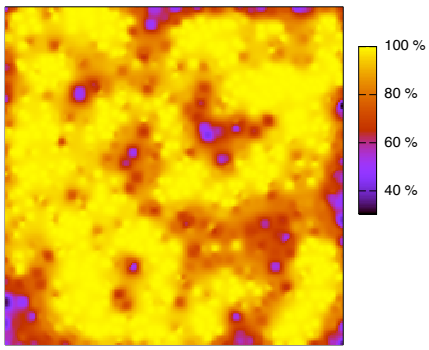

FIG. 3.16 – Consommation d'énergie totale intégrale sans SOLIST.
La topologie se réfère à la figure 3.13 avec marche aléatoire et inondation.

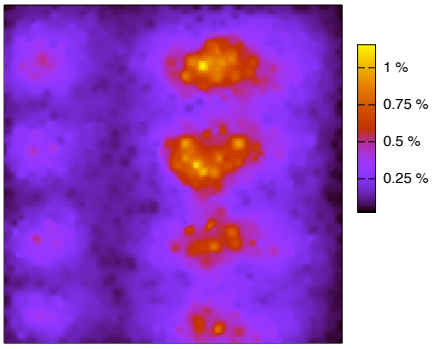

FIG. 3.17 – Consommation d'énergie totale pour l'anycast dans SOLIST.
La topologie se réfère à celle présentée en figure 3.13 avec 2×4 cellules.

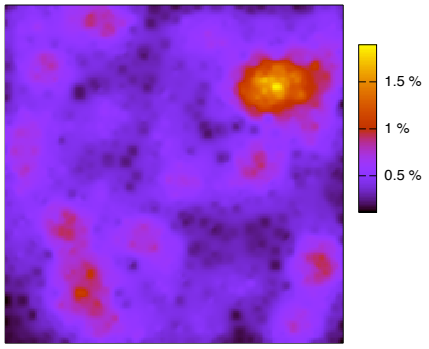

FIG. 3.18 – Consommation d'énergie totale pour l'anycast par marche aléatoire.
La topologie se réfère à celle présentée en figure 3.13.

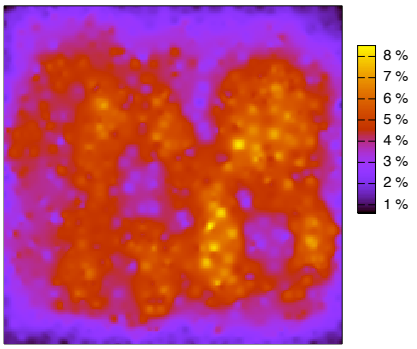

FIG. 3.19 – Consommation d'énergie totale pour le broadcast dans SOLIST.
La topologie se réfère à celle présentée en figure 3.13 avec 2×4 cellules.

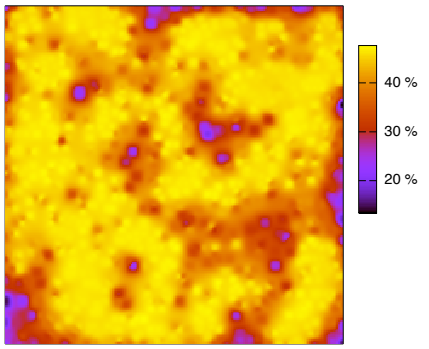

FIG. 3.20 – Consommation d'énergie totale pour le broadcast par inondation.
La topologie se réfère à celle présentée en figure 3.13.

par inondation est alors plus intéressante qu'un broadcast de chaque type du réseau (d'autant qu'il implique de connaître la liste exhaustive des types présents dans la système).

La figure 3.14 dénote quant à elle la consommation d'énergie finale instantanée nécessaire pour le mécanisme de k-cast dans SOLIST. À l'inverse de la consommation du broadcast, celle du k-cast dépend fortement du nœud de contact retourné par la primitive d'anycast. De fait, étant donné que le mécanisme de k-cast ne concerne qu'un sous-ensemble de nœuds d'un type spécifique et que le nœud de contact est situé à proximité du point d'entrée interrogé, des zones de forte consommation se dégagent autour des points d'entrées, mais plus vastes que celles apparaissant en figure 3.17.

Enfin, la consommation d'énergie moyenne dédiée à la maintenance de la cohésion de la structure est présentée en figure 3.12 en fonction du nombre de cellules dans SOLIST. Sachant que les primitives d'arrivée et de réorganisation en cas de départ ou de défaillance sont gérées localement, l'énergie nécessaire reste logiquement réduite quelle que soit la configuration (moins de 0,25 % de l'énergie totale disponible sur un nœud). En contrepartie, la primitive de départ demande une part de plus en plus importante d'énergie à mesure que le nombre de cellules augmente. Ceci est une conséquence de l'accroissement linéaire du nombre de messages de départ envoyés aux points d'entrées, pour la mise à jour sur ces derniers de l'information concernant les nœuds de contact. Par exemple, avec une configuration à 10×10 cellules, durant la simulation, chaque nœud devra quitter le réseau ou subir une défaillance. 100 000 messages de départ sont alors générés à travers le réseau uniquement pour la maintenance de la structure.

Ces résultats montrent que SOLIST permet de devancer les approches classiques utilisées ici, en terme de consommation d'énergie. Pour la topologie de réseau utilisée dans ces évaluations (1 000 nœuds répartis en 10 types), la configuration à 8 cellules est idéale pour obtenir le meilleur compromis entre efficacité et consommation d'énergie. Trouver la meilleure division pour une topologie de réseau donnée reste un problème ouvert.

3.7 Comparatif de travaux relatifs

À notre connaissance, il n'existe pas de proposition antérieure d'une interface générique aux RCsF statiques, fondée sur des primitives de communications. Aussi, nous comparons dans ce paragraphe chaque élément de la collection **-cast* avec leurs travaux antécédents relatifs.

En premier lieu, considérons les travaux d'anycast dans les RCsF. Entre autres, les auteurs de [TTS05] proposent un protocole de routage des requêtes d'anycast construit sur des arbres hiérarchiques. La grande majorité de ces travaux supposent l'existence d'une station de base (*cf.* chapitre 1) de manière à construire un arbre de routage. Bien que ces méthodes peuvent se révéler pertinentes dans certains contextes, les avantages de SOLIST reposent sur une présence facultative de station de base et donc une généricité plus grande des applications potentielles.

En second lieu, considérons les travaux relatifs au *multicast*[10] dans les RCsF. Ces derniers sont trivialement comparables à la définition du broadcast par type dans SOLIST. De nombreuses études ont été menées récemment, notamment le résultat présenté par [WNE02]. Cet article présente un mécanisme de broadcast et de multicast pour les RCsF, fondé également sur une construction d'arbre recouvrant pour réseaux statiques. Les arbres de multicast sont établis

[10] Mécanisme permettant de transmettre la même information à un ensemble prédéfinie de nœuds dans le système.

par suppression des arcs superflus pour chaque groupe de multicast. Désavantageusement, ce système a été évalué sur un réseau restreint. De surcroît, ce système ne prend pas en compte les départs ou défaillances de nœuds. Deux approches de multicast intéressantes pour les RCsF sont présentées respectivement dans [CRB01] et [HLR03]. La première [CRB01] compense la fiabilité de tout protocole de multicast pertinent par récupération de message en cas de perte. La seconde [HLR03] considère le multicast sur des RCsF mobiles, ce qui sort du cadre d'étude de ce chapitre.

La plus proche contribution a été proposée récemment dans [FW07]. Celle-ci présente une analyse théorique de leur approche, laquelle consiste en la construction d'ensemble dominant (*cf.* [FW07] pour de plus amples détails). Cet article introduit une proposition intéressante, mais ne repose pas sur les mêmes caractéristiques de système. De même, la gestion dynamique des groupes de multicast nécessite la reconstruction d'un arbre recouvrant minimum pour chaque modification du groupe considéré. Enfin, aucune simulation ou expérimentation réelle n'a été menée permettant d'illustrer en pratique leurs résultats fondamentaux. Par conséquent et pour l'ensemble de ces considérations, une comparaison est complexe à mener, ne présentant pas nous-même d'étude théorique de notre système.

D'autre part, différents travaux fondés sur de la diffusion dans les RCsF doivent être cités. Vollset *et al.* [VE03] proposent une classification des protocoles de broadcast fiables, selon deux grands axes : déterministe et probabiliste. Les protocoles déterministes cherchent à renforcer les garanties de forte fiabilité – à l'instar de SOLIST – contrairement aux approches probabilistes, délivrant une garantie dépendant d'une probabilité connue *a priori*. Alors que les protocoles déterministes induisent le plus souvent un compromis de qualité moindre relatif au délai d'acheminement du message, les approches probabilistes infèrent une fiabilité médiocre dans certains cas de figure. Dans la continuité, Wang and al. [WLJC05] proposent un protocole de broadcast fiable utilisant des techniques de groupement et une méthodologie épidémique (*cf.* chapitre 4) pour les MANET. Ils soutiennent un taux de délivrance de message fortement élevé pour un faible délai de transmission de bout en bout, mais uniquement pour des réseaux mobiles. Aucun résultat concernant des RCsF statique n'est présenté.

Enfin, il est indispensable de citer une approche pertinente de traitement de l'information dans les RCsF. TinyDB [MFHH05] repose sur une acquisition de l'information directement du réseau, à l'instar de notre système, mais en considérant le réseau comme une base de donnée physique. En conséquence, ils proposent une extension du langage de requête SQL. À l'instar de SOLIST, TinyDB est générique et représente une solution *tout-en-un* pour les applications sur des RCsF. Cependant, celui-ci est dépendant de l'application considérée en terme d'optimisation des requêtes et de leur exécution.

3.8 Conclusion

Dans ce chapitre, nous avons proposé SOLIST, une solution générique *tout-en-un* pour fournir l'implémentation de la collection *-cast (*i.e.* anycast, k-cast et broadcast) dans les RCsF statiques. Celle-ci est une architecture de système, faible consommation, pour les RCsF large-échelle. SOLIST est composé d'un ensemble fini de sur-couches (les LIGH-t-LAYERS) permettant de mettre en œuvre cette interface commune, fondée sur le groupement de nœuds de même

type. Une mise en œuvre efficace de la collection *-cast en terme de fiabilité des résultats et de consommation d'énergie est fournie.

SOLIST a été évalué par simulation pour chacune de ces fonctionnalités. Nous avons comparé SOLIST avec d'autres algorithmes classiques : les marches aléatoires pour l'anycast et l'inondation par arbre de diffusion pour le broadcast. Les résultats de cette étude illustrent la forte économie d'énergie due à l'utilisation de SOLIST par rapport à ces algorithmes standards.

Cependant, depuis quelques années, nombreuses sont les applications qui fleurissent autour des RCsF dits mobiles. Cette mobilité ne se rapporte pas aux événements à surveiller comme dans le chapitre 2, mais aux nœuds du réseau même. L'étude fondamentale de ces systèmes est l'optique de la seconde grande partie de ce manuscrit.

Deuxième partie

Réseaux de capteurs mobiles

CHAPITRE 4

Modèles de calcul pour réseaux de capteurs mobiles

Les applications et résultats de recherches concernant les réseaux mobiles sont légion depuis ces dernières années [AAD+06, BBC+05, BIM07, BBFK07b, CDT00, DQA04, Fee01, JBRAS03, PBBvR06, SYH04, TNCS02, VE03, WLJC05, XW04]. Cependant, ceux-ci se rapportent le plus souvent à des aspects pratiques spécifiques des systèmes mobiles, ne permettant de valider le plus souvent qu'empiriquement les solutions proposées.

L'objectif de la deuxième partie de ce manuscrit est de présenter et d'étudier en profondeur des modèles fondamentaux de réseaux mobiles existants, de les enrichir et, ainsi, de permettre de tisser des liens directs avec les aspects pratiques sus-cités. Pour cela, nous présentons en premier lieu dans ce chapitre les modèles théoriques existants en terme de réseaux mobiles.

Ensuite, dans le chapitre suivant, nous étendons les modèles présentés ci-après afin de pouvoir analyser théoriquement le comportement des algorithmes et protocoles proposés pour les réseaux de capteurs mobiles. Dans le chapitre 6, nous établissons une corrélation entre les réseaux mobiles et les réseaux épidémiques dynamiques, et dressons un inventaire des voies qu'ouvrent ce résultat.

4.1 Contexte général

Réseaux de capteurs mobiles La mobilité est une caractéristique usuelle des applications sur des réseaux de capteurs embarqués sur des objets ou des êtres vivants mobiles. En effet, qu'ils soient installés sur des populations de manchots empereurs en Antarctique ou sur des personnes atteintes de diabète, les capteurs ne peuvent généralement pas agir sur la mobilité de leur support[1]. Pourtant, la mobilité a un impact fort sur l'évolution des algorithmes. Par exemple, un algorithme conçu pour avoir un fonctionnement optimal avec des interactions identiques multiples (comme dans un groupe d'amis par exemple) aura immanquablement une fiabilité et une efficacité réduite sur un groupe d'unités mobiles au comportement

[1] À noter qu'il existe cependant une autre gamme de réseaux dont la mobilité est contrôlée (*e.g.* les robots), non traitée directement dans ce document.

aléatoire [BRS03, CHC+07, CDT00, JBRAS03].

C'est dans cette optique que nous proposons, au chapitre suivant, un modèle théorique de réseaux de capteurs mobiles permettant de prendre en compte l'impact du modèle de mobilité envisagé sur les différentes caractéristiques comportementales des algorithmes. Ce modèle est une extension des modèles dits *protocoles de population*, et plus récemment, *de communauté*.

Protocoles de population et de communauté Les *protocoles de population*, introduits initialement dans [AAD+06], ont été conçus originellement pour représenter les réseaux de capteurs composés d'agents mobiles à très faibles ressources, n'ayant aucun contrôle sur leur propre mouvement. Ainsi, ceux-ci modélisent les interactions dans un ensemble donné d'agents. Ce modèle fournit une base théorique commune pour les systèmes répartis, sur lesquels un comportement global émerge d'un ensemble d'interactions locales entre les nœuds. Un tel modèle peut être utilisé dans de nombreux contextes tels que MANET ou les RCsF.

L'abstraction des protocoles de population consiste en un ensemble fini d'états, un ensemble fini d'entrées et de sorties ainsi qu'une fonction de transition. L'ensemble des interactions possibles est représenté sous la forme d'un graphe d'interactions. Si deux agents mobiles se trouvent être suffisamment proches l'un de l'autre durant un laps de temps suffisamment long, ils interagissent en échangeant leurs informations locales. Par exemple, en considérant de petites entités embarquées sur des animaux, une interaction a lieu chaque fois qu'un animal équipé se trouve à proximité du rayon de transception d'un autre.

La puissance des protocoles de population repose sur la simplicité du modèle. Les agents sont considérés comme anonymes et aucune hypothèse spécifique n'est faite sur le synchronisme, l'ordre des interactions, l'infrastructure émergente ou même la taille du système. La suite d'interactions est uniquement guidée par un *ordonnanceur* sur lequel une unique hypothèse est imposée : celle d'être *équitable* (*i.e.* de s'assurer que toute évolution possible du système ne pourra être évitée indéfiniment). Une présentation formelle de ce modèle est donnée au paragraphe 4.2.

De nombreuses extensions ont été proposées parmi lesquelles se trouve les *protocoles de communauté* [GR07]. En effet, l'anonymat des agents dans les protocoles de population contraste fortement avec la grande majorité des modèles de systèmes répartis, au sein desquels la possession d'un identifiant unique par processus est souvent une propriété cruciale des algorithmes. Ainsi, récemment, Guerraoui et Ruppert ont proposé ce modèle étendu qui, tout en conservant l'essence des protocoles de population par la nature minimaliste des agents, permet à ces derniers de se voir attribuer des identifiants uniques issus d'un ensemble déterminé. Néanmoins, afin d'éviter une extension du modèle dans laquelle les espaces mémoires dédiés au stockage d'identifiants soient utilisés pour conserver une masse d'information arbitrairement grande, l'utilisation des identifiants est limité à l'*identification*. Ceci permet de généraliser les protocoles de population, tout en gardant l'esprit de ceux-ci, en interdisant aux agents d'utiliser l'espace réservé au stockage des identifiants pour accroître leur puissance de calcul. Une présentation plus complète est proposée au paragraphe 4.3.

4.2 Protocoles de population

Le modèle des protocoles de population décrit simplement et formellement un ensemble d'agents mobiles miniatures, lesquels interagissent localement les uns avec les autres afin de mener à bien un calcul global. L'inspiration et son nom proviennent de l'étude des RCsF équipant une nuée d'oiseaux. Ce modèle est analogue à celui des molécules interactives en chimie moléculaire, où les molécules interagissent deux à deux, modifiant l'état global de la structure.

4.2.1 Fonctionnalités de bases

Les caractéristiques définissant le modèle de base [AAD+06] sont les suivantes :

Agents anonymes à états finis Le système consiste en une grande *population* d'agents représentés par des machines à états finis indiscernables.

Calcul par interaction simple Dans le modèle original, les agents n'émettent pas de messages ou ne disposent pas d'une mémoire partagée. Au lieu de cela, une *interaction* entre deux agents permet de mettre à jour leur état respectif, selon une fonction de transition donnée (*cf.* paragraphe 4.2.2).

Séquence d'interactions imprévisible Le choix des agents prenant part à une interaction est effectué par une entité tierce : l'*ordonnanceur*. Un agent n'a pas de contrôle sur la nomination des autres agents avec lesquels il interagit, de même que l'ordonnanceur est limité à l'association d'agents adjacents dans le *graphe d'interaction*, représentant typiquement les contraintes de distance des agents. Un *condition d'équité* globale forte est imposée à l'ordonnanceur, permettant de garantir la progression de tout protocole (*cf.* paragraphe 4.2.2).

Entrées et sorties réparties La donnée d'entrée d'un protocole de population est répartie sur l'état initial de la population entière. Identiquement, les données de sortie sont réparties sur la totalité des agents.

La convergence plutôt que la terminaison Les protocoles de population ne peuvent généralement pas déceler s'ils ont convergé ou non. Cependant, la donnée de sortie des agents doit nécessairement converger, après un temps fini, vers une valeur commune et correcte.

Présentons dès à présent le modèle formalisé, puis quelques exemples de protocoles de population utilisés dans le reste de ce manuscrit.

4.2.2 Formalisation du modèle

L'évolution des états des machines à états finis, au cours d'une interaction, est définie par une fonction de transition permettant, à terme, de calculer une fonction f. Continuellement, les agents calculent leur valeur de sortie, corrélée à leur état courant, et convergent *irrémédiablement*[2] vers la valeur de sortie correspondante aux entrées initialement réparties sur les agents.

Plus formellement, un protocole de population est composé de :

[2]Dans tout ce manuscrit, nous utiliserons l'adverbe *irrémédiablement* en lieu et place de son équivalent anglo-saxon *eventually*.

- $\Lambda(\Upsilon, \Theta)$: un graphe d'interaction (le plus souvent complet) où Υ représente l'ensemble des $n \geq 2$ agents anonymes et Θ l'ensemble de tous les couples d'agents possibles pouvant interagir sur ce système. Dans le modèle de base, $\Theta = \{(\upsilon, \upsilon') \in \Upsilon^2 | \upsilon \neq \upsilon'\}$;
- Σ : un alphabet d'entrée fini ;
- Y : un alphabet de sortie fini ;
- Q : un ensemble fini de tous les états possibles des agents ;
- $\iota : \Sigma \to Q$: une fonction d'association permettant de relier l'état inital $\iota(\sigma)$ d'un agent ayant une valeur d'entrée σ ;
- $\omega : Q \to Y$: une fonction d'association permettant de relier la valeur de sortie $\omega(q)$ d'un agent dans un état q ;
- $\delta : Q \times Q \to Q \times Q$ une relation de transition sur tous les couples d'états.

Dès lors, nous notons indistinctement $(p, q) \mapsto (p', q')$, ou (p, q, p', q'), une transition si il existe $\delta(p, q) = (p', q')$. Une transition ne peut survenir entre deux agents que si ceux-ci ont une interaction. Un protocole est dit *déterministe* si la relation δ est une application (*i.e.* au plus une transition possible pour chaque couple de Q^2).

L'exécution d'un protocole consiste à appliquer une séquence d'interactions à la configuration initiale. Une *configuration* du système est représentée par un vecteur contenant l'état de tous les agents du système à un moment spécifique. Étant donné que les agents sont anonymes, deux agents avec le même état sont donc indiscernables. Ainsi, chaque configuration C peut être considérée comme un multi-ensemble non-ordonné d'états. Nous notons $C \to C'$ le fait que C' peut être obtenu à partir de C en une seule étape (*i.e.* avec uniquement une transition pour une interaction $\theta \in \Theta$). Plus formellement, si C contient deux états q_1 et q_2, C' est obtenu à partir de C en remplaçant q_1 et q_2 par respectivement q'_1 et q'_2, avec $(q_1, q_2, q'_1, q'_2) \in \delta$. Une exécution d'un protocole Ξ est donc une suite, finie ou non, de configurations de la population : $\langle C_0, C_1, C_2, \ldots \rangle$ telle que $\forall i, C_i \to C_{i+1}$.

Comme introduit préalablement, l'ordre des interactions est imprévisible. L'ordonnancement de celles-ci est choisi par une entité tierce, et tout protocole de population doit fonctionner selon n'importe quel ordonnancement choisi par cette entité. Néanmoins, quelques restrictions doivent être imposées à cet *ordonnanceur* afin de garantir des exécutions concluantes (*e.g.* il est nécessaire d'empêcher l'ordonnanceur de diviser la population en deux parties en supprimant toute communication entre l'une et l'autre). La *condition d'équité* permet d'interdire à l'ordonnanceur d'exclure indéfiniment une étape potentiellement réalisable. D'un point de vue plus formel, si C est une configuration apparaissant infiniment souvent au cours d'une exécution et qu'il existe une étape $C \to C'$, alors, C' doit également apparaître infiniment souvent au cours de cette même exécution. En d'autre termes, cette condition assure que tout ce qui est possible advient irrémédiablement (*i.e.* toute configuration qui est toujours atteignable est irrémédiablement atteinte).

En résumé, un protocole de population calcule *de façon stationnaire* une fonction $f : \Sigma^+ \to Y$ si $\forall n \in \mathbb{N}, \forall \sigma \in \Sigma^n$, chaque exécution équitable sur une population de n agents, respectivement initialisés avec les éléments de σ, se stabilise irrémédiablement $f(\sigma)$. Par conséquent, la valeur de sortie de tout agent se stabilise irrémédiablement sur $f(\sigma)$.

4.2.3 Exemples de protocoles de population

Dans les chapitres suivants, nous référençons différents exemples de protocoles de population pour illustrer nos contributions. Nous présentons donc, dans ce paragraphe, trois protocoles classiques dans les différentes contributions gravitant autour de ce modèle, ainsi que l'intuition de la correction de ceux-ci. Comme précisé au préalable, l'essence des protocoles de population repose avant tout sur la convergence plutôt que la terminaison.

La primitive « *ou* »

Cette opération peut également être considérée comme un sorte de diffusion. Pour cette raison, nous la dénommons le plus souvent *inondation* dans le reste de ce manuscrit. Cet algorithme très simple permet d'illustrer simplement le principe des protocoles de population. Tout agent ayant une valeur de sortie 0 retournera simplement 1 dès qu'il découvrira qu'un autre agent a 1 pour valeur de sortie.

Plus formellement, nous avons $\Sigma = Y = Q = \{0, 1\}$. ι et ω correspondent à la fonction identité et δ est le singleton $\{(0, 1) \mapsto (1, 1)\}$. Toutes les autres transitions sont définies de sorte qu'elles laissent inchangés les couples d'états des agents interagissants.

Estimation de la majorité

Considérons deux types d'entités dans le système (*e.g.* mâle et femelle, meneur et suiveur dans un couple de danseurs, *etc.*). Chaque agent de la population représente exclusivement l'un des deux types d'entités présentes. Ce protocole doit retourner la valeur 1 si la configuration initiale contient une *stricte majorité* du second type d'entité, et 0 sinon.

$$\forall x \in \{0, 1\}^+, f(x) = \begin{cases} 1 & \text{if } \sum_{x_i \in x} x_i > \frac{n}{2} \\ 0 & \text{sinon.} \end{cases}$$

Étant donné qu'il est impossible de compter un nombre d'objet anonymes de manière répartie, l'idée de ce protocole est de grouper une entité de chaque type par couple, et d'observer finalement le type des entités restantes. Pour cela, posons $\Sigma = \{\perp_0, \perp_1\}$ où le premier type d'entités est représenté par $\perp_0$ et le second par $\perp_1$. Considérons également $Q = \{\perp_0, \perp_1, 0, 1\}$ et $Y = \{0, 1\}$. ι correspond à la fonction identité et ω associe $\{\perp_0, 0\}$ à 0 ainsi que $\{\perp_1, 1\}$ à 1. Enfin, δ contient les quatre transitions suivantes : $(\perp_0, \perp_1) \mapsto (0, 0)$, $(\perp_0, 1) \mapsto (\perp_0, 0)$, $(\perp_1, 0) \mapsto (\perp_1, 1)$, $(0, 1) \mapsto (0, 0)$. Toutes les autres transitions sont définies de sorte qu'elle laissent inchangés les couples d'états des agents interagissants.

Calcul de la somme modulo

Il est possible de calculer de manière répartie la *somme modulo* x. Pour donner une idée de la conception d'un tel algorithme, nous présentons ce dernier protocole qui calcule la somme modulo 4 de toutes les valeurs d'entrée prises dans $\Sigma = \{0, 1, 2, 3\}$ (ici encore, l'ensemble Y est identique à Σ). Toutes les valeurs peuvent être collectées par un même unique agent, lequel se stabilisera irrémédiablement vers la valeur de la somme modulo 4. Dans l'optique d'ôter du

calcul les valeurs déjà prises en compte, l'état de chaque agent devient $\perp$, étiqueté de la valeur de l'irrémédiable agent unique sans valeur $\perp_i$.

Nous avons donc $Q = \{0, 1, 2, 3, \perp_0, \perp_1, \perp_2, \perp_3\}$. Soit ι la fonction identité et $\omega(v) = \omega(\perp_v) = v$. Le seul ensemble de règles de transition de δ qui ne laissent pas inchangé le couple d'états des agents interragissants, est le suivant : $\forall v, w \in \{0, 1, 2, 3\}, (v, w) \mapsto (v + w, \perp_{v+w})$ et $(v, \perp_w) \mapsto (v, \perp_v)$ pour lesquels l'addition est effectuée modulo 4.

Correction irrémédiable

Pour chacun de ces protocoles, la condition d'équité garantit que tous les agents vont irrémédiablement se stabiliser vers la valeur de sortie correcte. Par exemple, pour l'inondation, l'idée de la preuve de correction repose sur le fait que le nombre d'agents ayant pour état 1 ne peut décroître. Ainsi, si aucun agent ne possède initialement la valeur d'entrée 1, le système est d'ores et déjà stabilisé. Sinon, si au moins un agent possède cette valeur d'entrée, la condition d'équité garantie que le nombre d'agents à 1 tendra vers n irrémédiablement.

De même, pour le protocole d'estimation de la majorité, la première règle de transition permet d'éliminer l'intégralité des états à $\perp_0$, ou bien celle des états à $\perp_1$. Les deuxième et troisième règles garantissent qu'irrémédiablement, tous les agents retourneront la valeur de sortie correspondant au type de l'ensemble des entités non encore groupées. Enfin, la dernière règle assure que le système se stabilisera à 0 en cas d'égalité parfaite du nombre d'entités de part et d'autre. La condition d'équité garantie donc que le système se stabilisera à la valeur correcte avec une probabilité de 1.

4.2.4 Puissance du modèle

Afin de caractériser les fonctions calculables par les protocoles de population, nous pouvons, sans perdre en généralité, nous restreindre à l'étude des prédicats[3]. En effet, Pour toute fonction f ayant pour ensemble image Y, soit $P_{f,y}$ le prédicat défini par $P_{f,y} = 1$ si et seulement si $f(x) = y$. D'où, f est calculable par un protocole de population *si et seulement si* $P_{f,y}$ l'est aussi pour tout $y \in Y$. La preuve de la première implication de cette équivalence repose sur la possibilité de calculer en parallèle tous les prédicats $P_{f,y}$, en utilisant un composant séparé de l'état d'un agent pour chaque y. Or, Y est fini, puisque les valeurs de sortie correspondent au moins à un état de Q et que l'ensemble des états est lui-même fini par définition. L'autre implication de cette équivalence est triviale.

Considérons à présent une caractérisation des prédicats décidables par les protocoles de population. Introduisons pour cela les notions suivantes. Un multi-ensemble généré sur un alphabet Σ peut être représenté par un vecteur de taille $d = |\Sigma|$, où chaque membre du vecteur représente la multiplicité d'un caractère d'entrée donné. Par exemple, le multi-ensemble $\{a, a, c, c, c\}$ peut être représenté par le vecteur $(2, 0, 3) \in \mathbb{N}^3$ sur un alphabet d'entrée $\Sigma = \{a, b, c\}$. Un *ensemble semi-linéraire* est un sous-ensemble de $\mathbb{N}^d$ composé de l'union finie d'*ensembles linéaires* de la forme $\{\overrightarrow{b} + k_1\overrightarrow{a_1} + k_2\overrightarrow{a_2} + \cdots + k_m\overrightarrow{a_m}\}$, où $\overrightarrow{b}$ est le vecteur de référence à d dimensions, $\overrightarrow{a_1}$ à $\overrightarrow{a_m}$ sont des vecteurs de bases, et k_1 à k_m sont des coefficients non-négatifs. Un *prédicat semi-linéaire* sur un ensemble d'entrées correspond à un

[3]Les prédicats sont les fonctions dont l'ensemble image est restreint à $\{0, 1\}$.

prédicat exactement valide sur un ensemble semi-linéaire. Concernant le modèle de base des protocoles de population, il existe une caractérisation des prédicats calculables, correspondant précisément à l'ensemble des *prédicats semi-linéaires* [AAD$^+$06, AAE06b].

En effet, dans [AAD$^+$06], Angluin *et al.* montrent que les protocoles de population peuvent calculer tous les prédicats définissables à partir de l'*arithmétique de Presburger*, lesquels correspondent précisément à l'ensemble de tous les prédicats semi-linéaire. En outre, tout prédicat semi-linéaire peut être calculé par un protocole de population [AAE06b].

4.2.5 Travaux connexes

Récemment, un grand nombre de travaux ont été conduits sur la puissance de ce modèle, ou sur diverses extensions de celui-ci [AR07]. Par exemple, contraindre l'ordonnancement des interactions possibles (tout en conservant évidemment la condition d'équité) [AAE06a] permet d'étudier l'évolution en temps-réel de l'état du système. Nous revenons plus précisément sur cette notion et l'étendons dans le chapitre 5 suivant.

L'auto-stabilisation des protocoles de population a également été étudiée par suppression de la connaissance de l'état initial du système, et en se basant uniquement sur l'ensemble des valeurs d'entrée [AAFJ05]. De même, Delporte-Gallet *et al.* présentent comment les défaillances d'agents peuvent affecter la puissance des protocoles de population [DGFGR06]. Ils montrent notamment qu'à partir de n'importe quel protocole calculant une fonction dans le modèle de base sans défaillance, il existe une transformation générique permettant d'extrapoler un protocole tolérant $O(1)$ défaillances. Toutefois, cette méthode nécessite un affaiblissement des caractéristiques du problème. Dans [AAER07], Angluin *et al.* affirme en outre que le modèle de communication unidirectionnel (lequel peut être considéré comme l'utilisation d'un graphe d'interactions orienté) implique une puissance restreinte en comparaison au modèle bidirectionnel classique.

Enfin, l'extension du modèle par suppression de l'anonymat [GR07] fait l'objet d'une présentation détaillée au sein du paragraphe suivant. Nous présenterons alors les résultats des études concernant les défaillances byzantines en 4.3.4.

4.3 Protocoles de communauté

Récemment, Guerraoui et Ruppert [GR07] ont proposé une extension du modèle de base présenté ci-avant, appellée *protocoles de communauté*.

4.3.1 Motivation

La motivation de ce modèle est de relâcher l'hypothèse d'anonymat des agents tout en préservant la nature minimaliste de ceux-ci, afin de préserver l'esprit du modèle de base. En résumé, le modèle original est augmenté en attribuant un identifiant unique à chacun des agents et les autorisant à mémoriser un ensemble constant d'identifiants issus d'autres agents. Afin d'éviter de sombrer dans un modèle surpuissant, dans lequel l'espace mémoire dédié au stockage d'identifiants peut permettre la conservation d'une grande quantité d'information, l'usage des identifiants est limité à l'identification uniquement. Cette restriction garantit indirectement que

lorsqu'un algorithme est déployé en pratique, la taille de l'espace mémoire dédié au stockage d'identifiant peut être bornée, en fonction de l'ensemble des identifiants utilisés dans le système considéré.

4.3.2 Formalisation du modèle

Dans [GR07], l'extension des protocoles de population proposée, dénommée *protocoles de communauté*, consiste à attribuer un identifiant unique aux agents. Tous les identifiants possibles ainsi qu'un symbole spécial $\perp$ sont regroupés dans un ensemble infini U. La différence entre les protocoles de population et de communauté repose sur la définition de l'ensemble des états : $Q = B \times U^d$ où B représente la définition de l'ensemble des états dans le modèle de base, adjoint d'une mémoire de d identifiants. Comme dans les protocoles de population, les algorithmes sont uniformes : ils ne peuvent utiliser aucune borne sur le nombre des agents et U est infini. Afin de maintenir l'esprit des protocoles de population dans ce modèle étendu, certaines contraintes doivent être ajoutées : seul des identifiants d'agents existant déjà peuvent être mémorisés dans les d espaces dédiés de l'état d'un agent *et* aucune information structurelle concernant les identifiants ne peut être utilisée par les algorithmes. Ainsi, les protocoles de communauté doivent vérifier la formalisation de ces deux contraintes :

- $\forall (q_1, q_2) \mapsto (q'_1, q'_2) \in \delta, id \in q'_1 \vee id \in q'_2 \Rightarrow id \in q_1 \vee id \in q_2$; où $id \in q$ signifie que id apparait dans l'un des d derniers membres du $d+1$-uplet q, pour $q \in Q$ et $id \in U$.
- Soit π une permutation de U avec $\pi(\perp) = \perp$. Pour $q = \langle b, u_1, u_2, \ldots, u_d \rangle \in Q$, soit $\hat{\pi}(q) = \langle b, \pi(u_1), \pi(u_2), \ldots, \pi(u_d) \rangle$. Il est nécessaire que $\forall (q_1, q_2) \mapsto (q'_1, q'_2) \in \delta$: $(\hat{\pi}(q_1), \hat{\pi}(q_2)) \mapsto (\hat{\pi}(q'_1), \hat{\pi}(q'_2)) \in \delta$.

En résumé, la première condition garantie qu'aucune transition ne peut introduire de nouveaux identifiants et la seconde que les identifiants peuvent uniquement être stockés ou comparés par égalité, mais ne peuvent être manipulés en aucune autre manière. Finalement, tout protocole de population peut être considéré comme un protocole de communauté avec $d = 0$.

4.3.3 Puissance du modèle

Guerraoui and Ruppert ont montré que la puissance de leur modèle étendu est grandement augmenté [GR07]. En effet, les protocoles de communauté sont équivalents à la classe des fonctions symétriques dans $NSPACE(n \log n)$; *i.e.* tout langage décidé par une machine de Turing non-déterministe utilisant $O(S \log S)$ cases, avec la condition présentée ci-après. Considérons un fonction $f : \Sigma^+ \to Y$. f est *symétrique* si pour toute chaîne y obtenue par permutation des caractères d'une chaîne x, nous avons $f(x) = f(y)$. En d'autres termes :

$$\forall x \in \Sigma^+, \forall \pi \text{ une permutation de } x, f(x) = f(\pi(x)) \Leftrightarrow f \text{ est symétrique.}$$

4.3.4 Travaux connexes

Ce modèle ayant été proposé très récemment, il n'existe encore que très peu de travaux connexes à l'heure actuelle. Cependant, les auteurs de [GR07] traitent de la présence de défaillances dans ce modèle. Ils montrent que l'émulateur de machines de Turing utilisé pour caractériser la puissance du modèle peut être étendu pour tolérer un nombre donné de défaillances

bénignes. Cette caractérisation modifiée est fondée sur l'approche conditionnelle proposée par Mostefaoui *et al.* [MRR03] et inspirée de la preuve de Delporte-Gallet *et al.* [DGFGR06], permettant ainsi de rendre l'émulation tolérante aux défaillances.

Pour conclure, considérons la présence de défaillances Byzantines dans ce modèle. Un agent est dit Byzantin s'il possède un comportement arbitrairement défaillant, *i.e.* il peut interagir avec tous les autres agents, en prétendant être dans n'importe quel état à chaque interaction. Ainsi, il est connu qu'aucun prédicat non-trivial ne peut être calculable par un protocole de population en présence d'un seul agent Byzantin [GR07]. En effet, aucun mécanisme classique des systèmes répartis permettant d'identifier et de contraindre l'action de Byzantins ne peut être utilisé sans identification des agents. Ce même résultat d'impossibilité persiste dans les protocoles de communauté si un agent malicieux peut créer de nouveaux identifiants. Par contre, si aucun identifiant ne peut être créé, alors il est possible de conduire des calculs non-triviaux en présence d'un nombre constant de défaillances arbitraires [GR07]. Cependant, la détermination précise de la puissance d'un ordonnancement aléatoire et la caractérisation de la calculabilité des protocoles de communauté en présence d'agents Byzantins sont toujours des questions ouvertes [AAE07, AR07, GR07].

4.4 Conclusion

Dans ce chapitre, nous avons présenté deux modèles des RCsF mobiles : les protocoles de population et de communauté. Ces modèles épurés décrivent le comportement d'agents mobiles interagissants ensemble pour mener à bien un calcul réparti. Dans le premier modèle, les agents sont programmés identiquement avec une machine à états finis. Le second modèle est une extension du premier, permettant d'associer à chaque agent un identifiant unique, augmentant ainsi la puissance des protocoles de population.

Ce chapitre introduit la seconde partie de ce manuscrit en établissant un tour d'horizon des possibilités et des contributions gravitant autour de ces deux modèles. Ces derniers sont le fondement des différentes contributions présentées dans les chapitres suivants.

CHAPITRE 5

Prise en compte de la mobilité dans les protocoles de population et de communauté

Dans ce chapitre, notre étude se focalise sur les protocoles de population et de communauté. Nous présentons ici l'impact de la mobilité des agents sur la vitesse de convergence de ces protocoles. Après un court développement de nos motivations, nous étendons les modèles de protocoles de population et de communauté présentés respectivement dans les paragraphes 4.2 et 4.3 afin de formaliser le comportement de l'ordonnanceur. De ce modèle, nous pouvons ainsi analyser formellement la vitesse de convergence de n'importe quel protocole. Cependant, l'obtention d'équations s'avère être une tâche laborieuse, voire inaccessible dans la plupart des cas. Nous présentons alors un ensemble de résultats empiriques sur les trois protocoles introduits en section 4.2.3. Forts de ces observations, nous démontrons l'existence d'une borne inférieure de la vitesse de convergence pour tout protocole des deux classes sus-citées, ainsi que la configuration de l'ordonnanceur pour l'atteindre. Nous achèverons ce chapitre par une considération pratique de ces résultats et notamment par une réflexion sur la pertinence du modèle de mobilité classique dit de « points de navigation aléatoires » (ou *Random Way-Point*).

5.1 Motivation

La plupart des travaux concernant les protocoles de population (et de communauté) ont porté sur la puissance de calcul des dits modèles. Le plus souvent, aucune hypothèse n'est imposée sur le modèle d'interaction entre les agents, excepté celle d'équité. Le *modèle d'interaction* correspond à la modélisation de l'ordonnanceur des interactions, *i.e.* la manière dont celui-ci choisit l'ordre des interactions au cours de l'exécution du protocole. Les seules contributions ayant étudié la convergence des protocoles de population selon un modèle d'interaction spécifique, n'ont considéré que la version uniforme de celui-ci [AAE06a], *i.e.* chaque couple d'agent possède la même probabilité que tous les autres d'être sélectionné pour l'interaction suivante.

Cependant, dans la réalité, les modèles de mobilité d'un ensemble d'entités mobiles est rarement uniforme. Effectivement, dans la société, nous avons tendance à rencontrer très souvent un sous-ensemble restreint de la population existante, et à l'inverse, de ne jamais rencontrer une grande majorité de celle-ci. De même, dans les populations de manchots empereurs sur la banquise, les couples formés restent fidèles durant toute la période de reproduction, et les poussins sont regroupés géographiquement en « crèches ». Ces comportements organisés impliquent un modèle d'interaction trivialement non-uniforme.

Bien que ces modèles d'interaction n'aient aucune influence ni sur la puissance de calcul du modèle original, ni sur la convergence d'un protocole, ils peuvent néanmoins avoir un impact sur la vitesse de convergence. Des études récentes sur les réseaux mobiles ont effectivement confirmé que les modèles de mobilité ont des conséquences sur la vitesse d'un protocole [FMP06]. Ainsi, de nombreux travaux se sont efforcés de caractériser des modèles de mobilité réalistes [CHC+07, CLF07]. Nous les présentons plus en détails au paragraphe 5.3.2.

À partir de ces analyses, et par l'utilisation de notre modèle étendu présenté au paragraphe suivant, nous présentons dans la suite de ce chapitre, trois contributions majeures :

1. Une analyse empirique de l'impact des modèles de mobilité, par rapport à un ensemble de caractéristiques définissant la population observée (*cf.* paragraphe 5.3) ;
2. Une preuve de l'existence et la caractérisation d'une borne inférieure de la vitesse de convergence moyenne pour tout protocole de population et de communauté (*cf.* paragraphe 5.4) ;
3. Une analyse formelle du modèle de mobilité à « points de navigation aléatoires », largement répandu dans le domaine des réseaux mobiles (*cf.* paragraphe 5.5).

5.2 Mobilité avec les protocoles de population et de communauté

Afin d'étudier formellement l'impact des modèles de mobilité sur la rapidité de convergence de protocoles, nous proposons un modèle étendu des deux présentés dans le chapitre précédent (*cf.* paragraphes 4.2 et 4.3).

5.2.1 Les modèles MAPP et MAPC

L'objectif de l'extension de ces modèles est de modéliser l'ordonnanceur et de définir ses heuristiques, tout en conservant la condition d'équité nécessaire au bon fonctionnement des protocoles. Nos proposons ainsi deux modèles, nommés respectivement *MAPP* (*Mobilité Appliquée aux Protocoles de Population*) et *MAPC* (*Mobilité Appliquée aux Protocoles de Communauté*).

Ces extensions consistent à pondérer le graphe d'interaction des modèles initiaux. Chaque arc reliant deux agents d'une population se verra donc attribuer une étiquette. Celle-ci représente la probabilité que les deux agents, encadrant un arc donné, soient choisis pour participer à la prochaine interaction au cours d'une exécution. En d'autres termes, elle représente la probabilité de rencontre de deux agents donnés, dans une vision globale du système. Nous enrichissons donc le graphe d'interaction $\Lambda(\Upsilon, \Theta)$ tel que

$$\forall \theta \in \Theta, \exists \upsilon, \upsilon' \in \Upsilon, \upsilon \neq \upsilon' \wedge \theta = \upsilon \xrightarrow{p_{\upsilon,\upsilon'}} \upsilon'.$$

Par la suite, nous utiliserons indistinctement, en fonction du contexte, les notations $p_{v,v'}$ ou p_θ. Enfin, sans perdre en généralité, nous supposons que la condition suivante est vérifiée pour tous les graphes d'interaction considérés :

$$\sum_{\theta \in \Theta} p_\theta = 1.$$

En pratique, ce modèle permet de modéliser les choix de l'ordonnanceur au cours de l'exécution d'un protocole donné. En effet, étant donnée une population[1] dans une configuration C, l'ordonnanceur choisit donc les protagonistes de la prochaine interaction selon la probabilité de rencontre de ceux-ci.

5.2.2 Pertinence du modèle

Condition d'équité

Dans un premier temps, nous devons vérifier que notre modèle respecte la condition d'équité, imposée par définition à l'ordonnanceur (*cf.* paragraphe 4.2). C'est l'objectif du lemme suivant.

Lemme 5.1 *Tout ordonnanceur généré par la distribution des probabilités d'un MAPP respecte la condition d'équité.*

Preuve – $\forall \theta \in \Theta$, si $p_\theta > 0$ alors la probabilité que cette interaction se produise après n'importe quelle autre est non nulle. Ainsi, étant données deux configurations C et C' telles que $C \rightarrow C'$ par l'interaction θ, la probabilité que cette transition soit choisie par l'ordonnanceur est également non nulle. Donc, si C apparaît infiniment souvent durant l'exécution, alors C' apparaît également infiniment souvent.

À l'inverse, si $p_\theta = 0$, alors, sans perdre en généralité, nous pouvons considérer que le système est modélisé par un graphe d'interaction restreint, tel que $\theta \notin \Theta$. Dans ce cas, il ne peut exister $C \rightarrow C'$ par l'interaction θ.

D'où, quelle que soit la distribution d'un MAPP donné, l'ordonnanceur utilisant celle-ci ne peut rompre la condition d'équité. ■

Intégrations dans les travaux existants

L'un des avantages de notre extension est son intégration aisée dans tous les modèles dérivés des protocoles de population. En effet, notre proposition ne réduit pas la puissance des modèles initiaux.

Par exemple, les puissances présentées au chapitre précédent reste inchangées, que ce soit pour les protocoles de population (*i.e.* les prédicats semilinéaires [AAE06b, AAER07]) ou de communauté (*i.e.* les fonctions symétriques de $NSPACE(n \log n)$ [GR07]). Ainsi, tous les résultats issus de l'auto-organisation de ces protocoles [AAFJ05] et de la prise en compte de défaillances [DGFGR06] restent valides.

[1] Dans la suite de ce chapitre, nous utiliserons le terme de population pour représenter à la fois les populations et les communautés, sauf indication contraire.

D'où, par l'utilisation de MAPP, nous pouvons modéliser la grande majorité des modèles issus des protocoles de population. Par conséquent, considérer un *graphe d'interaction restreint* [AAD+06] peut être vu comme un protocole sous MAPP avec une partie des arcs du graphe d'interaction pondérés par 0. De même, tous les résultats proposés dans le contexte d'*interactions aléatoires* [AAD+06, AAE06a] sont valides pour tout MAPP affecté d'une distribution uniforme des probabilités sur les arcs du graphe. Enfin, les différents modèles introduits dans [AAER07] sur la modélisation des communications unidirectionnelles restent valables dans un MAPP avec un graphe orienté.

5.2.3 Analyse fondamentale de la convergence

De nombreux articles [AAD+06, AAE06a, AAE06b, AAER07, AR07, GR07] ont traité de la puissance des protocoles de population et de ses extensions, mais très peu concernant le temps de convergence de ceux-ci. C'est l'objectif principal de l'utilisation de notre modèle MAPP, lequel fourni un cadre simplifié pour analyser l'évolution de ces protocoles.

Estimation du temps de stabilisation par des chaînes de Markov

Grâce à MAPP, il est possible de modéliser le comportement de l'ordonnanceur, et ainsi étudier formellement l'évolution de l'état d'une population. À chaque étape, l'ordonnanceur choisit le prochain couple d'agent uniquement en fonction des probabilités présentes sur le graphe d'interaction Λ. Trivialement, pour une configuration donnée, les choix futurs des agents à interagir sont indépendants des choix effectués dans le passé. Ainsi, étant donné une situation présente, le futur est conditionnellement indépendant du passé. D'où, un MAPP donné peut être vu comme une *chaîne de Markov à espace d'état discret*, puisque le nombre d'agents dans la population et l'ensemble Q sont tout deux finis. Cette chaîne de Markov est composée d'une part de l'ensemble des configurations possibles d'un MAPP donné (*i.e.* son espace des états), et d'autre part de l'ensemble des transitions d'un pas du processus stochastique, qui est directement extrapolé de la distribution des probabilités disponibles sur le graphe d'interaction. La *distribution des probabilités de transition* dans la chaîne de Markov peut donc être représentée par une *matrice stochastique*, à partir de laquelle l'évolution du système peut être extraite *a priori*.

Cependant, le nombre de configurations d'un MAPP croît exponentiellement en fonction de la taille des ensembles Q et Υ. Le paragraphe suivant présente une analyse de l'évolution d'un MAPP sur un exemple simple, puis démontre que le nombre d'états de la chaîne de Markov associée est $|Q|^{|\Upsilon|}$ $(= |Q|^n)$.

Au préalable, il est nécessaire d'introduire les outils mathématiques utilisés dans l'étude de ces chaînes de Markov. Dans la suite, considérons T un ensemble ordonné d'index, tel que l'ensemble des entiers naturels $\mathbb{N}$, des réels positifs $[0, +\infty[= \mathbb{R}^+$, ou un sous-ensemble de ceux-ci. Les éléments $t \in T$ peuvent être interprétés somme des « *temps* ». À partir de cette notion de temps, il est possible, étant donné un processus stochastique, de connaître le temps auquel le système atteindra un état spécifique. Par conséquent, nous pouvons également définir l'espérance de ce temps, correspondant à un temps d'atteinte moyen :

Définition 5.1 (Premier temps d'atteinte) *Étant donnés* (Ω, Σ, Pr) *un espace de probabilité et* S *un espace des états mesurable, soient* $X : \Omega \times T \to S$ *un processus stochastique et* $\mathcal{A}$ *un sous-ensemble mesurable de* S*. Le* premier temps d'atteinte $\tau_{\mathcal{A}} : \Omega \to [0, +\infty]$ *est la variable aléatoire définie par*

$$\tau_{\mathcal{A}}(\omega) = \inf\{t \in T | X_t(w) \in \mathcal{A}\}. \qquad \diamond$$

Définition 5.2 (Espérance et écart-type du temps d'atteinte) *Étant donnés* $\mathcal{I}$ *un état et* $\{\tau^i_{\mathcal{A}}\}_{i \in \{n \in \mathbb{N} | n < N\}}$ *un ensemble de premier temps d'atteinte, l'*espérance du temps d'atteinte *(ou* temps d'atteinte moyen*) correspond à la valeur présumée de* $\tau_{\mathcal{A}}$ *pour une distribution initiale* $\mathcal{I}$*,* i.e. *:*

$$\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{A}}) = \lim_{N \to \infty} \frac{1}{N} \cdot \sum_{i=1}^{N} \tau^i_{\mathcal{A}}$$

*et l'*écart-type du temps d'atteinte *est défini par*

$$\sigma^2_{\mathcal{I}}(\tau_{\mathcal{A}}) = \lim_{N \to \infty} \frac{1}{N-1} \cdot \sum_{i=1}^{N} (\tau^i_{\mathcal{A}} - \mathbb{E}(\tau_{\mathcal{A}}))^2. \qquad \diamond$$

Dès lors, nous dénommons *convergence*, l'atteinte de l'état stationnaire dans une chaîne de Markov associée à un MAPP (*i.e.* l'atteinte de la configuration stable pour une population donnée). Ainsi, le terme de *vitesse de convergence* représente l'espérance du temps d'atteinte de la distribution stationnaire.

Difficulté d'analyse sur un exemple simple

À présent, il est possible d'illustrer notre méthode d'analyse à partir d'un exemple très simple : la primitive *ou* présentée au paragraphe 4.2.3 (page 85). Dans ce protocole, l'ensemble des états est de taille minimum[2] ($|Q| = 2$). De plus, la chaîne de Markov est *apériodique* étant donné qu'un agent « *infecté* » ne peut pas être guéri. Ci-dessous, nous montrons qu'une analyse de la vitesse de convergence pour un nombre d'agents quelconque n'est pas calculable, aussi simple que soit ce protocole.

Considérons une population de taille très restreinte, à savoir composée de 4 agents. La partie gauche de la figure 5.1 présente le graphe d'interaction de cette population. Pour simplifier le cadre d'étude, nous considérons ici que chaque agent possède la même distribution sortante des probabilités (*i.e.* chacun des agents à exactement un lien pondéré par p, un autre par p' et un dernier par p''). Ainsi, les agents sont non seulement indiscernables par leur état, mais également par leur distribution sortante. Nous formalisons cette hypothèse par l'équation 5.2 du paragraphe 5.3.1.

Ainsi, la partie droite de la figure 5.1 représente la chaîne de Markov $\mathcal{M}$ associée à cette population pour la primitive *ou*. Les agents dans l'état 0 sont représentés par des cercles blancs contrairement aux disques noirs représentant ceux dans l'état 1. Pour identifier les différentes configurations, chaque état de la chaîne de Markov est étiqueté par un entier de $[\![1, 8]\!]$. Enfin, afin de ne pas surcharger la figure, les transitions ne changeant pas l'état de la population (*i.e.*

[2] Dans le cas d'un ensemble d'état des agents singleton, l'espace des états de la chaîne de Markov associée est également un singleton et le système est donc toujours stabilisé dans l'état unique.

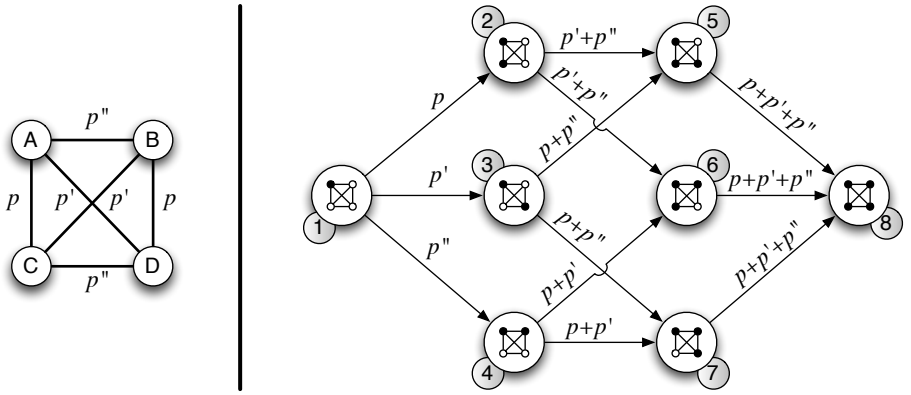

FIG. 5.1 – Graphe d'interaction avec 4 agents partageant la même distribution sortante des probabilités, et la chaîne de Markov associée pour la primitive *ou*.

les boucles) ne sont pas représentées. Chacune de ces boucles est pondérée dans la chaîne par une probabilité de 1 à laquelle est retranché l'ensemble des probabilités sortantes :

$$p_{i,i} = 1 - \left(\sum_{j \in [\![1,8]\!] \setminus \{i\}} p_{i,j} \right)$$

Considérons la configuration initiale dans l'état 1. Par l'utilisation d'une analyse stochastique, il est possible d'exprimer formellement l'expression du temps d'atteinte moyen dans ce cas[3] :

$$\begin{aligned}
\mathbb{E}_1(\tau_8) &= \sum_{k \in adj_{\mathcal{M}}(1)} p_{1,k} \cdot \mathbb{E}_1[t_{1\to k} + t_{k\to 8} | 1 \to k] \\
&= \left(\sum_{k \in adj_{\mathcal{M}}(1)} p_{1,k} \cdot \mathbb{E}_1[t_{1\to k} | 1 \to k] \right) + \left(\sum_{k \in adj_{\mathcal{M}}(1)} p_{1,k} \cdot \mathbb{E}_1[t_{k\to 8} | 1 \to k] \right) \\
&= \left(\sum_{k \in adj_{\mathcal{M}}(1)} p_{1,k} \right) + \left(\sum_{k \in adj_{\mathcal{M}}(1)} p_{1,k} \cdot \mathbb{E}_k(\tau_8) \right) \\
&= 1 + \sum_{k \in adj_{\mathcal{M}}(1)} p_{1,k} \cdot \mathbb{E}_k(\tau_8)
\end{aligned}$$

Par récurrence, nous obtenons ainsi l'équation suivante :

$$\mathbb{E}_1(\tau_8) = 2 + \sum_{q \in \{p, p', p"\}} 2 \cdot q \cdot \frac{3 - 4 \cdot q}{1 - 2 \cdot q} \tag{5.1}$$

Considérons à présent l'extension de cette chaîne de Markov pour n agents, dont la structure est représentée en figure 5.2. Sur cette figure, l'état de la population est représenté sous

[3] la définition de la fonciton $adj.(\cdot)$ est identique à celle donnée au paragraphe 2.3.1, page 32.

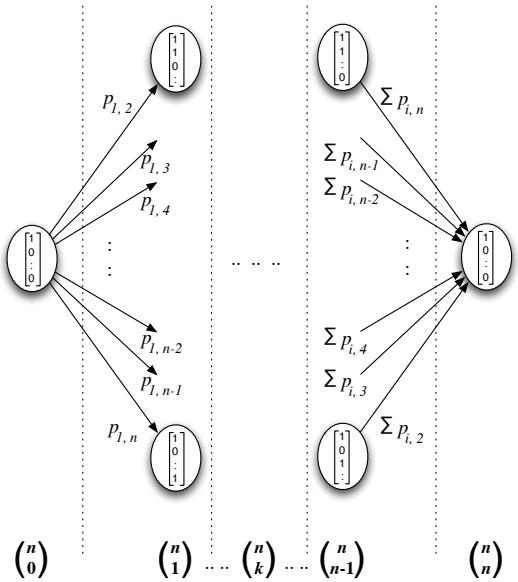

FIG. 5.2 – Chaîne de Markov associée à la primitive *ou* pour n agents partageant la même distribution sortante des probabilités.

forme de vecteur, contenant l'état de chaque agent. La combinaison présentée au-dessous de chaque colonne d'état de cette chaîne représente le nombre d'états contenus dans chacune d'elle.

Dans le cas d'une distribution uniforme des interactions (*i.e.* $\forall \theta, \theta' \in \Theta, p_\theta = p_{\theta'}$), tous les états peuvent être regroupés ensemble, et la chaîne de Markov est une simple ligne. Ainsi, l'équation du temps d'atteinte moyen est assez simple à obtenir. Cependant, dans le cas général, il s'avère difficile d'établir une équation formelle de l'espérance du temps d'atteinte, en raison du grand nombre d'états dans la chaîne :

Lemme 5.2 *Étant donnés n agents et une distribution des probabilités d'interaction quelconque, le nombre d'états dans la chaîne de Markov associé à un protocole $\mathcal{P}$ est $|Q|^{|\Upsilon|}$.*

Preuve – Chaque agent possède un état appartenant à Q. Ainsi, si nous considérons l'ensemble des configurations de la population comme des vecteurs, il existe au plus $|Q| \times \cdots \times |Q|$ états dans la chaîne. En effet, il n'est plus possible de considérer les configurations comme des multi-ensembles dans MAPP, étant donné la possibilité pour les agents d'avoir une distribution de sortie différente de celle des autres. Ainsi, la position d'un agent dans le graphe d'interaction n'est plus indépendante. Alors, nous avons $|Q|^{|\Upsilon|}$ différents états, lesquels pouvant tous correspondre à l'état initial du système. Alors, le nombre d'état de la chaîne de Markov est $|Q|^{|\Upsilon|}$. ■

S'il est simple et utile de comprendre précisément l'évolution d'un système comportant un petit nombre d'agents, il devient impraticable de l'étudier formellement lorsque n croît significativement. Dans ce cas, il est toujours possible d'étudier empiriquement ces systèmes.

Estimation par la méthode de Monte-Carlo pour des populations de grande taille

Dans le cas d'un système avec un grand nombre d'agents, le calcul du temps moyen d'atteinte de la stabilité sur la chaîne de Markov associée à un MAPP donné se révèle donc fastidieux, voire impossible. Cependant, par des méthodes probabilistes, il est possible d'obtenir des résultats empiriques sur le comportement de ce protocole. En dépit de la taille grandissante de la chaîne de Markov, il est toujours faisable de calculer les transitions de sortie possibles d'une configuration donnée, et leur probabilité de se produire. Ainsi, il est concevable d'estimer le chemin suivi au sein d'une chaîne de Markov via sa distribution de probabilité. Néanmoins, il persiste toujours le problème de déterminer combien d'étapes sont nécessaires pour converger vers la distribution stationnaire, avec un taux d'erreur acceptable. Pour cela, sur un très grand ensemble d'estimations s'arrêtant dès que la distribution stationnaire est atteinte, et par l'utilisation de la *loi des grands nombres*[4], le nombre moyen d'étapes peut représenter une estimation correcte de l'espérance du temps d'atteinte de la convergence.

Afin d'estimer précisément le temps d'atteinte moyen ainsi que les erreurs d'estimation, nous utilisons la méthode de Monte-Carlo pour les chaîne de Markov (MCMC) [Ber04]. Cette méthode est un algorithme de calcul reposant sur une estimation aléatoire répétée permettant de chiffrer le résultat [MU49]. Ce type de méthode est couramment utilisé pour extraire des résultats empiriques, permettant ensuite de conjecturer une résultante formelle [BJM06]. En effet, par l'observation de l'évolution d'un système sur un grand nombre de simulations, il est possible d'extrapoler des caractéristiques communes à toutes ces exécutions. Elle permettent ainsi d'orienter l'étude formelle qui s'en suit.

Dans la suite, nous utilisons indistinctement la moyenne du temps d'atteinte théorique et sa valeur estimée par la méthode de Monte-Carlo (de même pour sa variance théorique et estimée). Ainsi, dans la méthode de Monte-Carlo, nous considérons les mêmes définitions 5.2, mais avec un N très grand plutôt qu'avec $N \to \infty$. L'erreur potentielle de cette estimation est exprimée en intervalle de confiance. Celui-ci représente, avec un pourcentage d'erreur connu, un intervalle qui est supposé contenir la valeur théorique à estimer. Plus formellement, nous avons :

Définition 5.3 (Intervalle de confiance) *Étant donné $\mathcal{I}$ un état initial et $\{\tau_{\mathcal{A}}\}_{i\in\{n\in\mathbb{N}:n<N\}}$ un ensemble fini de premiers états d'atteinte, l'*intervalle de confiance *de l'estimation est défini comme suit :*

$$\begin{cases} \left[\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{A}}) - 1.96 \cdot \dfrac{\sigma_{\mathcal{I}}(\tau_{\mathcal{A}})}{\sqrt{N}} \quad ; \quad \mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{A}}) + 1.96 \cdot \dfrac{\sigma_{\mathcal{I}}(\tau_{\mathcal{A}})}{\sqrt{N}}\right] & \textit{avec 5\% d'erreur} \\ \left[\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{A}}) - 2.5758 \cdot \dfrac{\sigma_{\mathcal{I}}(\tau_{\mathcal{A}})}{\sqrt{N}} \quad ; \quad \mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{A}}) + 2.5758 \cdot \dfrac{\sigma_{\mathcal{I}}(\tau_{\mathcal{A}})}{\sqrt{N}}\right] & \textit{avec 1\% d'erreur} \end{cases}$$

◇

[4]Étant donné un échantillon de variables aléatoires indépendantes, définies sur le même espace probabilisé, ayant les mêmes espérance et variance finies, la moyenne des observations approchera irrémédiablement et restera proche de la moyenne théorique.

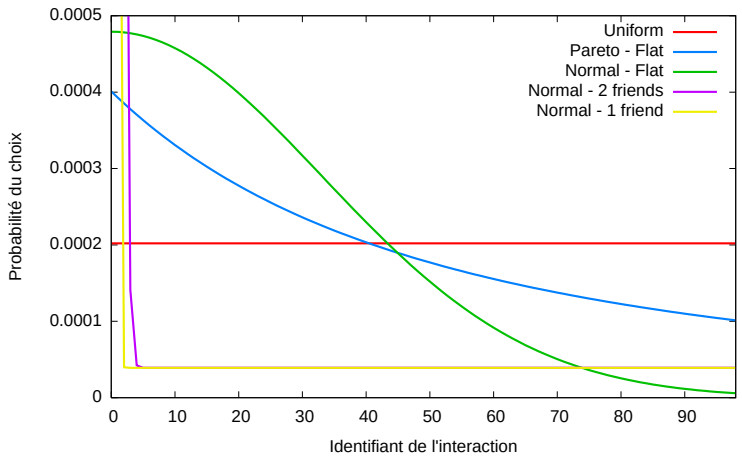

FIG. 5.3 – Comparaison des différentes DSP pour un agent donné.

5.3 Impact des modèles de mobilité sur la vitesse de convergence

Dans ce paragraphe, nous cherchons à mettre en évidence l'influence des modèles de mobilité sur la convergence des algorithmes, via différentes distributions des probabilités sur MAPP. Il est évident que les mêmes remarques sont également applicables à MAPC, et donc à la classe des protocoles de communauté. Nous illustrons nos conjectures via les exemples introduits au paragraphe 4.2.3.

5.3.1 Echantillon des distributions de probabilité sur Λ

Afin de simplifier le cadre d'étude, et ainsi préserver l'esprit des protocoles de population, dans tout le paragraphe 5.3, nous prendrons pour hypothèse l'unicité de la distribution sortante des probabilités (DSP). Plus formellement, étant donné un agent $\upsilon \in \Upsilon$, la DSP est définie par le multi-ensemble suivant : $\mathcal{P}_\upsilon = \{p_{\upsilon,\psi} | \psi \in \Upsilon - \{\upsilon\}\}$ regroupant l'ensemble des probabilités d'interaction dans lesquelles υ est impliqué. Pour une population à n agents, il est trivial que $|\mathcal{P}_\upsilon| = n - 1$. Ainsi, tous les agents d'une population possède la même DSP pour un MAPP donné. Soit, formellement, nous avons :

$$\forall \upsilon, \psi \in \Upsilon, \mathcal{P}_\upsilon = \mathcal{P}_\psi \tag{5.2}$$

Par exemple, considérons une population de 100 agents. La figure 5.3 présente, pour un $\upsilon \in \Upsilon$, cinq DSP différentes utilisées dans les simulations stochastiques de la méthode MCMC ci-après. Chacune de ces DSP est équivalente à une fonction de densité. Ainsi, étant donné que $\sum_{\theta \in \Theta} p_\theta = 1$, l'aire située sous la courbe de chaque DSP est la même et égale à $\frac{2}{n}$ (le graphe d'interaction étant supposé non-orienté). Pour chaque DSP de cette figure, nous présentons le comportement réel correspondant à cette modélisation. Le nom de chaque DSP est donné en fonction de son apparence similaire aux fonctions de densité des distributions de probabilité classique.

Uniforme *Chaque agent de la population a la même probabilité d'interagir avec tous les autres agents.* Chaque agent de la population côtoie tous les autres agents équitablement, et ne partage pas d'affection particulière ;

Pareto - Plat *Cette distribution traduit que chaque agent a une préférence envers quelques autres agents.* La distribution Pareto est également connue sous le nom de *loi de puissance*. Cette relation décrit une connaissance globale de la population, mais avec une préférence pour un sous-ensemble de celle-ci (cette DSP peut être comparée aux relations dans un petit village) ;

Normal - Plat *Cette distribution montre deux opposés : un penchant vers quelques autres agents et une antipathie vers un autre ensemble.* Cette relation décrit une connaissance de groupe, et peut être vue comme la DSP d'un petit groupe de personnes ;

Normal - 2 amis *Dans cette distribution, chaque agent a une probabilité plus forte d'interagir avec deux agents spécifiques* ;

Normal - 1 ami *Dans cette distribution, chaque agent a une probabilité plus forte d'intéracgir avec un unique agent.* Ces deux dernières DSP peuvent être interprétées comme la connaissance proportionnelle d'un agent parmi un grand et un très grand groupe de personnes.

À partir de ces DSP, nos observations empiriques se déclinent en deux familles. La première étudie en temps discret l'impact de la topologie du graphe d'interaction. En second lieu, nous observons l'impact du modèle de distribution du temps d'inter-rencontre, afin de modéliser le temps continu.

5.3.2 Estimation en temps discret : les modèles de rencontre

Observons l'impact de ces différentes DSP et des caractéristiques du graphe sur l'espérance du temps d'atteinte et donc, sur la vitesse de convergence des protocoles.

Graphe d'interaction complet

Considérons en premier lieu un graphe d'intéraction complet, comme dans le modèle de base. Chaque agent a donc une probabilité non nulle d'interagir avec n'importe quel autre agent de la population. Plus formellement, nous avons : $\forall \upsilon \in \Upsilon, \forall p \in \mathcal{P}_\upsilon, p > 0$.

Les figures 5.4, 5.5 et 5.6 présentent respectivement la vitesse de convergence des primitives *ou*, de *majorité stricte* et de *somme modulo 4* pour chacune des distributions présentées en figure 5.3. Sur ces courbes en échelle double logarithmique, la vitesse de convergence est exprimée en nombre d'étapes (*i.e.* d'interactions) nécessaires dans la chaîne de Markov pour atteindre la distribution stationnaire. À chacune de ces espérances est associé l'intervalle de confiance à 1% d'erreur correspondant.

Sur ces trois figures, il est évident de constater que la DSP possède un impact non négligeable sur la vitesse de convergence d'un protocole. De plus, plusieurs remarques peuvent être extrapolées plus précisément.

D'une part, sur chacune de ces courbes, plus la distribution des probabilités est équilibrée, plus le temps de convergence est court. C'est d'autant plus conséquent pour les grands graphes

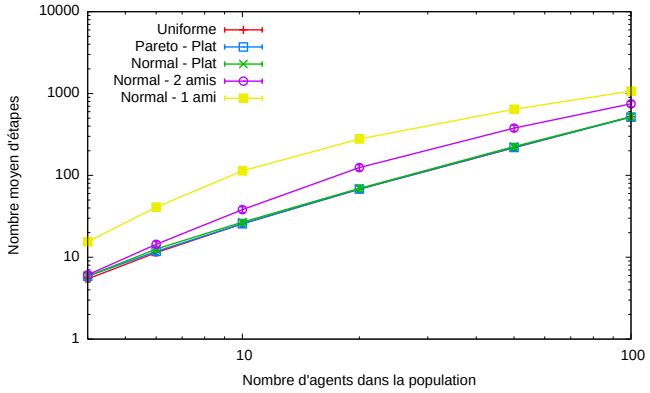

FIG. 5.4 – Vitesse de convergence de la primitive *ou* en fonction du nombre d'agents dans un graphe complet.

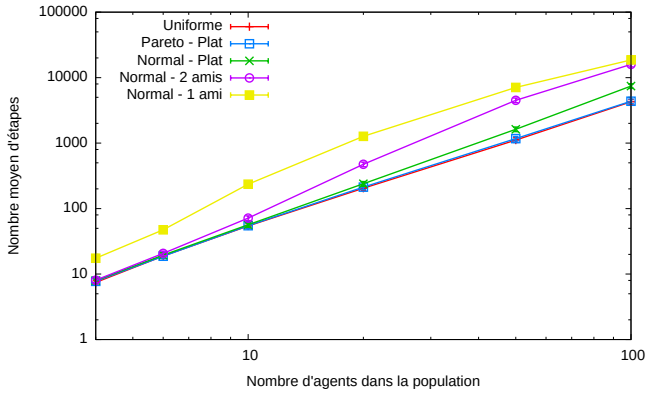

FIG. 5.5 – Vitesse de convergence de la primitive *majorité* en fonction du nombre d'agents dans un graphe complet.

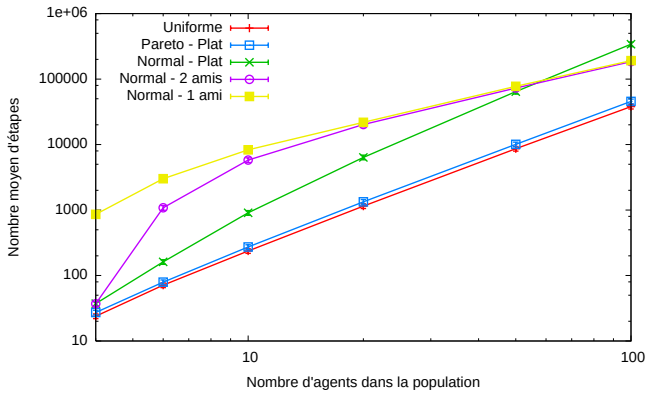

FIG. 5.6 – Vitesse de convergence de la primitive *somme modulo 4* en fonction du nombre d'agents dans un graphe complet.

pour lesquels les trois DSP dit « plates » (Uniforme, Pareto et Normal) sont souvent bien meilleures que les DSP à un nombre très restreint d'amis. Par ailleurs, la DSP uniforme infère une convergence plus rapide que toutes les autres, et ce, quels que soient le protocole et le nombre d'agents. Nous revenons en détail sur cette remarque au paragraphe 5.4.

D'autre part, observons plus précisément certaines allures de courbes. Celles étiquetées « Normal - 2 ami » possède une caractéristique singulière. Pour un nombre restreint d'agents, la distribution est équivalente à « Normal Plat ». À l'inverse, pour un nombre plus conséquent, ce sont « Normal - 1 et 2 amis » qui deviennent équivalents. Ce comportement s'explique naturellement par la fonction de densité de « Normal - 2 ami » qui se rapproche plus de « Normal - Plat » pour un population de 4 agents et approxime l'allure de « Normal - 1 ami » pour un nombre d'agents croissant. Cette évolution est commune à toutes les protocoles observés, mais de façon plus représentative pour la somme modulo 4. En effet, la croissance rapide entre 4 et 10 noeuds peut être expliquée par le fait que pour ce protocole, atteindre l'état stationnaire nécessite plusieurs rencontres pour chaque couple d'agents.

Enfin, pour ce même protocole, la courbe « Normal - Plat » vient croiser celle des « Normal - 1 et 2 amis » puis croître plus rapidement. Dans le cadre de ce protocole spécifique, pour une population de taille importante, avec les DSP à 1 ou 2 amis, dès qu'il existe un unique agent possédant la valeur de sortie correcte, il la partage avec son ou ses deux amis et ces probabilités de rencontres étant forte, elle se transmettra rapidement à toute la population. Dans le cas de la DSP « plate », il pourra y avoir de nombreuses interactions sans modification de valeur, étant donné que l'unique agent doit rencontrer chaque autre agent individuellement. Nous pouvons supposer que pour des plus grandes tailles de population, les DSP « Uniforme » et « Pareto » viendront affleurer les courbes à DSP « plates ».

Pour conclure nos observations sur les graphes d'interaction complets, il est intéressant d'étudier le *cœfficient de brassage*, défini ci-après, pour les probabilités les plus grandes. En effet, par défaut, les graphes générés sont réguliers, *i.e.* les grandes probabilités sont réparties uniformément et régulièrement parmi l'ensemble de la population. Cependant, tout en veillant à conserver l'hypothèse d'unicité des DSP, il est possible de brasser le graphe afin de supprimer les zones denses du graphes en terme de grandes probabilités (*i.e.* sorte de groupes d'interaction simulant les réseaux sociaux). Le cœfficient de brassage représente la probabilité d'inverser deux arcs dans le graphe régulier initial, et ainsi, de rendre le graphe d'interaction plus ou moins aléatoire. La figure 5.7 présente pour les distributions non-uniformes le nombre d'interactions moyen nécessaire pour atteindre l'état stationnaire de la primitive *or*, en fonction du cœfficient de brassage du graphe. Il est évident que le cœfficient de clique (*cf.* paragraphe suivant) pour les arêtes de poids fort du graphe d'interaction complet n'a aucun impact sur la vitesse de convergence.

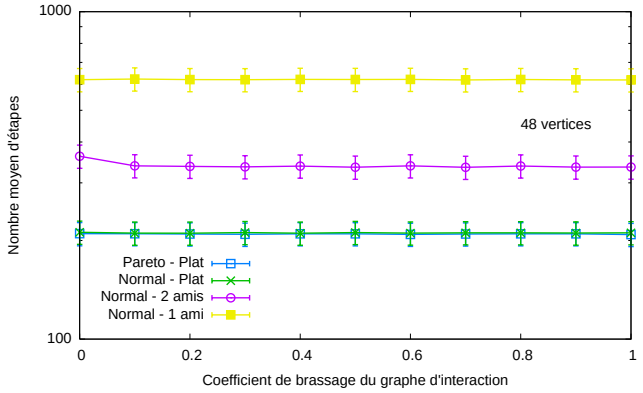

FIG. 5.7 – Vitesse de convergence de la primitive *ou* en fonction du cœfficient de brassage dans un graphe complet.

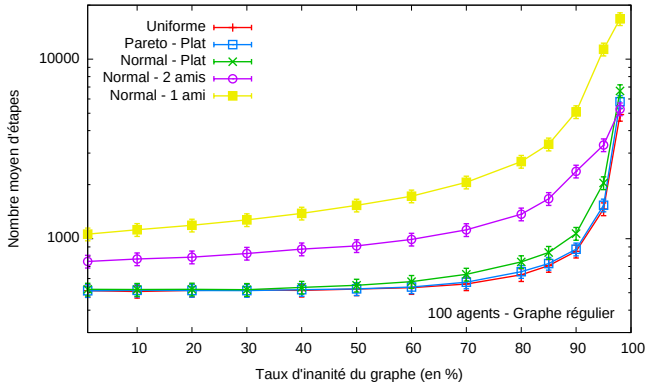

FIG. 5.8 – Vitesse de convergence de la primitive *ou* en fonction de l'inanité du graphe sur une population de 100 agents.

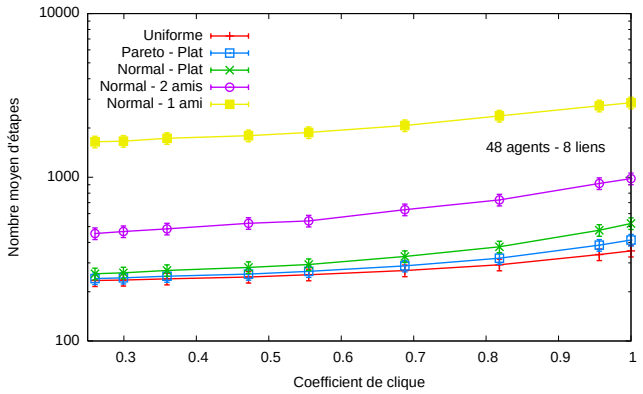

FIG. 5.9 – Vitesse de convergence de la primitive *ou* en fonction du cœfficient de clique sur un graphe restreint à 48 agents.

Graphe d'interaction restreint

Considérons à présent l'impact des caractéristiques d'un graphe d'interaction restreint sur la vitesse de convergence. Comme présenté au paragraphe 5.2.2, un graphe d'interaction non complet dans MAPP revient à relâcher l'hypothèse de stricte positivité des probabilités. Plus formellement, dans ce contexte, nous avons : $\forall \upsilon \in \Upsilon, \forall p \in \mathcal{P}_\upsilon, p \geq 0$. De plus, pour les mêmes raisons que précédemment, nous supposons également l'hypothèse d'unicité des DSP. Ainsi, il est possible d'établir la condition suivante : $\forall \upsilon \in \Upsilon, \exists p \in \mathcal{P}, p = 0$.

Nous ne considérons dans cette sous-partie que la primitive *or*. En effet, les primitives de majorité et de somme modulo 4 nécessitent des topologies particulières pour converger. La somme modulo 4, par exemple, implique que l'agent collecteur unique rencontre tous les autres agents afin de leur transmettre la valeur finale de la somme. Il est donc trivial que ce protocole ne peut converger sur un graphe restreint.

Les observations révélées au paragraphe précédent restent évidemment valides dans le contexte des graphes d'interaction restreints. Nous focalisons dans ce paragraphe sur deux autres caractéristiques : l'*inanité du graphe* et le *cœfficient de clique*.

L'inanité du graphe est l'inverse de la densité en terme de nombre d'arcs dans un graphe quelconque (*i.e.* représente la proportion de *vide* d'un graphe). Dans notre contexte, le taux d'inanité correspond au nombre d'arcs avec une probabilité nulle.

La figure 5.8 présente pour chacune des distributions présentées en figure 5.3, et en fonction du taux d'inanité, le nombre moyen d'interactions nécessaires pour atteindre l'état stationnaire de la primitive *or*. Ces résultats sont issus d'une population de 100 agents pour un graphe régulier. Malgré un faible impact pour un taux faible d'inanité (du graphe quasi-complet jusqu'à un taux de 40%), les conséquences sur la vitesse de convergence pour un taux très fort sont considérables. Bien que ce résultat soit intuitif de premier abord, certains auraient penser que pour un graphe régulier avec un taux d'inanité de 98% (*i.e.* graphe avec une topologie en anneau), la vitesse de convergence en serait accrue. Au contraire, pour toutes les DSP considérées (excepté la « Normal - 1 ami » pour laquelle on ne parle pratiquement qu'à un agent sur les deux connus), le nombre d'interactions nécessaires augmente significativement et les différences se confondent. Ceci s'explique naturellement puisque les DSP n'ont que deux probabilités non-nulles, celles-ci sont alors toutes équivalentes à l'uniforme.

D'autre part, le cœfficient de clique d'un graphe (ou *clustering cœfficient* pour les anglo-saxons) représente le taux de sommets adjacents à un autre sommet et liés les uns avec autres. Initialement introduit dans [WS98], il est souvent corrélé à la longueur caractéristique des chemins dans le graphe. Plus formellement, ces deux notions sont définies par :

Définition 5.4 (cœfficient de clique) *Soit un sommet $v \in V$ d'un graphe $G(V, E)$. Étant donné $K = \frac{k_v \cdot (k_v - 1)}{2}$ où $k_v = |adj_G(v)|$, le* cœfficient de clique *du sommet v est :*

$$C_v = \frac{|\{(x, y) \in E | x, y \in adj_G(v)\}|}{K}.$$

De manière globale, le cœfficient de clique *de G est $C_G = \frac{\sum_{v \in V} C_v}{|V|}$.* ◇

Définition 5.5 (Longueur caractéristique des chemins) *Soit $G(V, E)$ un graphe donné. La* longueur du plus court chemin $L_{v,v'}$ *entre v et v' représente le nombre d'arcs minimum séparant $(v, v)' \in V^2$. De manière globale, la* longueur caractéristique des chemins *du graphe G est :*

$$L_G = \frac{\sum_{(v,v)' \in V^2} L_{v,v'}}{|V^2|}. \qquad \diamond$$

Comme la longueur caractéristique des chemins est fortement corrélée au cœfficient de clique [WS98], leur impact sur la vitesse de convergence l'est également. Ainsi, nous présentons uniquement les observations issues de l'impact du cœfficient de clique dans ce paragraphe.

La figure 5.9 présente pour chacune des distributions présentées en figure 5.3 la vitesse de convergence en fonction du cœfficient de clique. La population utilisée dans ce cas est composée de 48 agents avec un taux d'inanité de 83%. Il est observable ici que le cœfficient de clique d'un graphe a un impact sur le nombre d'itérations nécessaires à la convergence, mais contrairement à ce que chacun pourrait penser, celui-ci est somme toute assez faible.

Toutes les mesures des caractéristiques du graphe ont été données en temps discret. Le temps de convergence est exprimé en nombre d'interactions nécessaires pour atteindre la distribution stationnaire de la chaîne de Markov associée. Deux interactions s'exécutant en parallèle sur deux couples disjoints d'agents sont considérées comme séquentielles. Cette hypothèse n'est plus valide dans le cas où la vitesse de convergence est exprimée en temps continu. C'est l'objectif du paragraphe suivant.

5.3.3 Estimation en temps continu : modèles d'inter-rencontre

Il n'est pas possible de déduire la durée moyenne de convergence d'un protocole à partir de son étude en temps discret. En effet, en considérant un environnement en temps continu, une DSP donnée ne reflète pas la fréquence des interactions, uniquement la probabilité d'advenir à un état de la population donné. Cette fréquence est dénommé *fréquence d'inter-rencontre* dans la suite.

Ainsi, pour une DSP donnée, le nombre d'étapes nécessaires pour atteindre la distribution stationnaire reste identique, quelle que soit la distribution des durées de rencontre et d'inter-rencontre considérée. Cependant, étant donnée une distribution de ces dernières, à partir de la DSP, il est possible d'échantillonner les dates des rencontres et la durée de celles-ci pour chaque couple d'agents de la population. En utilisant cet échantillonnage, il est possible d'estimer une date de convergence du protocole dans le domaine continu par simulation stochastique.

Dans ces simulations, nous considérons un graphe d'interaction complet et une unicité des DSP sur la population. Pour générer les échantillonnages de temps d'inter-rencontre, trois des distributions les plus communes dans la littérature [CHC+07, CLF07] sont utilisées : *Exponentiel*, *Pareto* et *LogNormal*. Ces distributions possèdent respectivement les fonctions de repartition

$$\lambda \cdot e^{-\lambda x} \quad , \quad \frac{k \cdot x_m^k}{x^{k+1}} \quad \text{et} \quad \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}},$$

de paramètres respectifs $\lambda > 0, x_m > 0, k > 0, \sigma > 0$ et $-\infty < \mu < \infty$.

Afin de mesurer séparément les durées de rencontre et d'inter-rencontre, chaque simulation a été effectuée d'une part avec des rencontres simples (*i.e.* une unique interaction à chaque rencontre d'un couple donné) et d'autre part, avec des rencontres diffuses (*i.e.* une interaction par seconde pendant toute la durée de la rencontre). Pour chacune des simulations, les paramètres des distributions ont été fixés de manière à pouvoir les comparer les unes avec les autres : la durée moyenne d'inter-rencontre est de 15 minutes et dans le cas de rencontres diffuses, la durée de celles-ci est tirée uniformément entre 1 et 100 secondes.

Deux jeux de simulation ont été effectués pour différentes tailles de population : un pour la primitive *ou* et un autre pour *majorité*. Les figures 5.10 et 5.11 présentent les données obtenues sur ces deux jeux dans le cas de rencontres simples, tandis que les figures 5.12 et 5.13 présentent celles dans le cas de rencontres diffuses. Chaque jeu de simulation contient une estimation par MCMC du temps de convergence dans le domaine continu, pour les trois distributions d'inter-rencontre sus-citées sur les DSP « Uniforme », « Normal - Plat » et « Normal - 2 amis ».

L'observation de ces diagrammes permet de conclure trivialement sur le fort impact des distributions d'inter-rencontres sur le temps de convergence d'un protocole pour un nombre d'agents spécifique. De même, il parait évident que sur ces exemples, la durée des rencontres influe très peu sur la vitesse de convergence (*cf.* l'allure des diagrammes 5.10 et 5.12, de même que celle des diagrammes 5.11 et 5.13).

Nous pouvons cependant relever deux singularités sur ces quatre diagrammes. Alors que la taille de la population influe peu sur le temps de convergence pour les distributions d'inter-rencontre *Pareto* et *LogNormal*, celui-ci a une forte influence sur la distribution *exponentielle*. Le temps d'exécution quasi-constant avec les deux premières distributions peut s'expliquer par la multiplication des interactions parallèles lorsque la population grandit. D'un autre côté, la troisième distribution influe, en gain et en perte, sur la vitesse de convergence pour des tailles importantes de population. Cette remarque révèle la seconde particularité observable de ces diagrammes. Celle-ci porte également sur la distribution *exponentielle*. Tandis que le temps de convergence diminue avec l'augmentation de la population pour des DSP « Uniforme » et « Normal - Plat », celui-ci croît fortement dans le cas d'une DSP « Normal - 2 amis » sur un grand graphe. Cela provient du déséquilibre des probabilités de rencontres induit par la DSP considérée. En revanche, la raison pour laquelle cette disproportion n'influe pas dans le cadre d'une distribution *Pareto* d'inter-rencontre reste une question ouverte.

À la suite de cet ensemble d'observations représentatives, que la simulation se déroule dans le domaine discret ou continue, il persiste néanmoins que, pour une configuration de graphe donnée, une DSP uniforme permet toujours d'obtenir le meilleur temps de convergence. Inspirée de cette conjecture, nous proposons de formaliser et de démontrer ce résultat quelle que soit la configuration considérée et pour tous les protocoles, qu'ils soient de population ou de communauté.

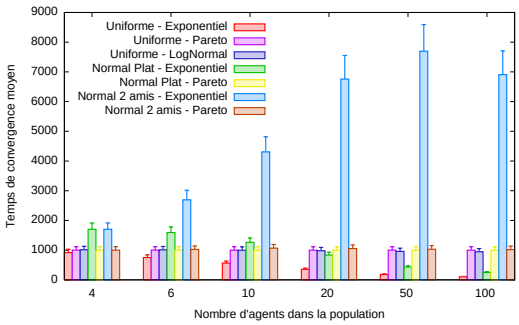

FIG. 5.10 – Temps de convergence de la primitive *ou* en fonction du nombre d'agents avec des rencontres simples.

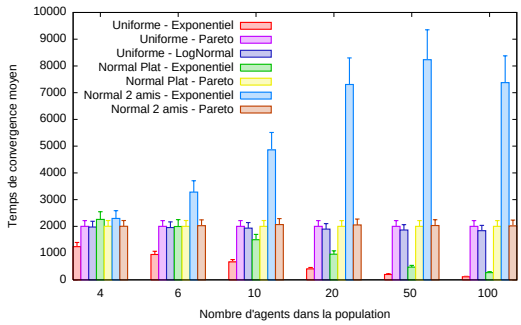

FIG. 5.11 – Temps de convergence de la primitive *majorité* en fonction du nombre d'agents avec des rencontres simples.

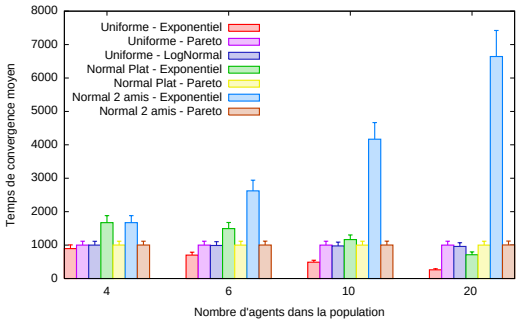

FIG. 5.12 – Temps de convergence de la primitive *ou* en fonction du nombre d'agents avec des rencontres diffuses uniformes.

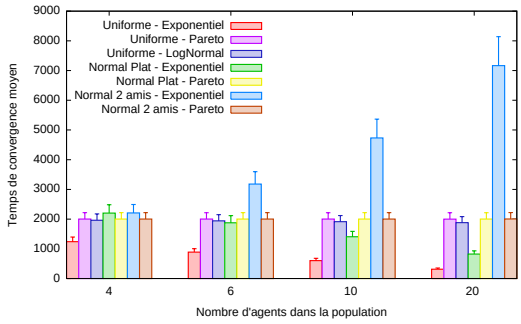

FIG. 5.13 – Temps de convergence de la *majorité* en fonction du nombre d'agents avec des rencontres diffuses uniformes.

5.4 Une borne inférieure optimale de la vitesse de convergence

La DSP uniforme semble en effet obtenir les meilleurs temps de convergence sur l'intégralité des simulations présentées dans la section précédente. Si ce résultat paraît naturel pour certains protocoles tels que les calculs de somme ou de majorité [AR07], il est bien moins intuitif pour d'autre tel que l'inondation par exemple. En effet, pour le protocole de *somme modulo 4*, tous les agents du système doivent interagir les uns avec les autres à plusieurs reprises pour converger. Une distribution uniforme est alors intuitivement recommandée. Néanmoins, pour la primitive *ou*, en supprimant l'hypothèse d'unicité des DSP supposée jusqu'à lors, certains pourraient penser qu'une DSP où l'agent source de l'inondation est impliqué dans quasiment toutes les interactions, converge de manière plus rapide qu'une DSP uniforme.

L'objectif du théorème suivant est de démentir cette intuition. En effet, celui-ci révèle que, quel que soit le protocole de population considéré, une DSP uniforme est l'optimal en terme de vitesse moyenne de convergence. En outre, ce résultat reste valide sans l'hypothèse d'unicité des DSP.

Theorème 5.1 *Pour toute fonction f calculable par un protocole de population, la borne inférieure du temps de convergence est atteinte avec une DSP uniforme dans MAPP.*

Preuve – Préalablement, nous présentons une esquisse de cette preuve. En premier lieu, (1) nous caractérisons la classe d'équivalence des protocoles de population. Puis, (2) nous déduisons de cette classe d'équivalence que tout protocole de population possède une équation de l'espérance du temps d'atteinte polynomiale. Enfin, (3) nous montrons que tout temps d'atteinte moyen polynomial accepte une borne inférieure par l'application d'une DSP uniforme.

Caractérisation des fonctions calculables Dans [AAD+06], le théorème 5 expose que chaque prédicat définissable dans l'arithmétique de Presburger est calculable de façon stationnaire par un protocole de population standard. De même, cette arithmétique caractérise l'intégralité des fonctions calculables par un protocole de population [AAE06b]. Ainsi, ces deux domaines de fonctions partagent la même classe d'équivalence.

En premier lieu, observons une décomposition de tout prédicat de l'arithmétique de Presburger. Soient $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ un alphabet d'entrée arbitraire, et A_i, c, m des constantes entières avec que $m \geq 2$. Il est montré que toute l'arithmétique de Presburger peut être calculée de façon stationnaire en utilisant une composition des prédicats sur des entiers naturels $x_1, \ldots, x_k$ suivants :

- $\sum_i a_i x_i < c$;
- $\sum_i a_i x_i \equiv_m c$ (*i.e.* $\sum_i a_i x_i \equiv c$ modulo m) ;
- Toute fonction booléenne ξ à deux variables.

Nous présentons ci-après les protocoles de population permettant de calculer chacun de ces trois prédicats générateurs. Nous allons montrer que pour toute composition de ces prédicats, une DSP uniforme dans MAPP correspond au temps de convergence optimal de cette composition. Ainsi, tout prédicat de l'arithmétique de Presburger calculé par composition de ces protocoles de population aura un temps de convergence optimal avec la DSP uniforme. Cette classe de composés ayant la même classe d'équivalence que les protocoles de population [AAD+06], il est trivial de conclure que si toute composée de protocoles de population générateurs atteint son temps de convergence optimal pour une DSP donnée, cette DSP sera également optimale pour tous les protocoles de population existants.

Définitions des protocoles de population générateurs Voici les deux protocoles de population proposés dans [AAD+06] permettant de calculer les deux premiers prédicats présentés ci-dessus. Soit

$s = \max(|c|+1, m, \max_i |a_i|)$. Dans chacun des deux protocoles, l'espace des états Q est l'ensemble $\{0,1\} \times \{0,1\} \times \{u \in \mathbb{Z} | -s \leq u \leq s\}$ et la fonction ι est définie par $\sigma_i \mapsto (1, 0, a_i)$. Le premier *bit* de chaque état est appelé le *bit de meneur* et permet d'élire un leader unique qui agrégera les valeurs de la combinaison linéaire. Le second bit est le *bit de sortie* qui enregistre pour chaque agent la valeur de sortie calculée par le dernier leader rencontré. Le dernier terme du triplet d'état est le *compteur* utilisé pour agréger la combinaison linéaire des x_i (membre gauche du prédicat donné). La fonction de sortie ω est définie simplement par $(\cdot, b, \cdot) \mapsto b$.

Nous décrivons maintenant les règles de transition pour chacun des deux protocoles (leur correction est prouvée dans [AAD+06]).

1. Soit, pour tout couple d'entiers u, u' avec $-s \leq u, u' \leq s$, les deux fonctions suivantes :

$$q(u,u') = \max(-s, \min(s, u+u')) \text{ et } r(u,u') = u + u' - q(u,u')$$

   Il en découle que $q(u,u') \in [-s,s]$, $r(u,u') \in [-s,s]$ et $q(u,u') + r(u,u') = u + u'$. Soit $b(u,u') = 1$ si $q(u,u') < c$ et 0 sinon. Les règles de transition sont données par la formule suivante si au moins ℓ ou ℓ' est égal à 1 :

$$(\ell, \cdot, u), (\ell', \cdot, u') \to (1, b(u,u'), q(u,u')), (0, b(u,u'), r(u,u')).$$

   Dans le cas où ℓ et ℓ' valent 0, l'interaction n'a aucun effet.

2. Soit $b(u,u') = 1$ si $u+u' \equiv_m c$ et 0 sinon. Le second protocole possède la famille de transitions suivante, si au moins ℓ ou ℓ' est égal à 1 :

$$(\ell, \cdot, u), (\ell', \cdot, u') \to (1, b(u,u'), (u+u') \mod m), (0, b(u,u'), 0).$$

   Sinon, dans le cas où ℓ et ℓ' valent 0, l'interaction n'a aucun effet.

Inspiré par la preuve du lemme 3 de [AAD+06], nous présentons à présent le calcul d'une fonction booléenne ξ sur deux prédicats F et G calculables de façon stationnaire. Soit $\mathcal{A}$ (repectivement $\mathcal{B}$) un protocole de population qui calcule de façon stationnaire F (respectivement G), en supposant que $\mathcal{A}$ et $\mathcal{B}$ ont le même ensemble d'entrée Σ. Le protocole $\mathcal{C}$ calcule de façon stationnaire $\xi(F,G)$ par composition parallèle de $\mathcal{A}$ et $\mathcal{B}$ (la population exécute les protocoles $\mathcal{A}$ et $\mathcal{B}$ en parallèle et renvoie les résultats des prédicats F et G ainsi calculés, appliqués à la fonction ξ) :

Soit $Q_{\mathcal{A}}$ et $Q_{\mathcal{B}}$ l'espace des états de $\mathcal{A}$ et $\mathcal{B}$ respectivement. L'espace des états de $\mathcal{C}$ est $Q_{\mathcal{C}} = Q_{\mathcal{A}} \times Q_{\mathcal{B}}$. La fonction d'entrée $\iota_{\mathcal{C}}$ associe $\sigma \in \Sigma \mapsto (\iota_{\mathcal{A}}(\sigma), \iota_{\mathcal{B}}(\sigma))$ et la fonction de transition est définie par $\delta_{\mathcal{C}}((p_1,p_2),(q_1,q_2)) = ((p'_1,p'_2),(q'_1,q'_2))$ avec $\delta_{\mathcal{A}}(p_1,q_1) = (p'_1,q'_1)$ et $\delta_{\mathcal{B}}(p_2,q_2) = (p'_2,q'_2)$. La fonction de sortie applique ξ aux sorties des deux protocoles composés :

$$\omega_{\mathcal{C}}((q_1,q_2)) = \xi(\omega_{\mathcal{A}}(q_1), \omega_{\mathcal{B}}(q_2)).$$

La vitesse de convergence de $\mathcal{C}$ dépend donc directement de celles de $\mathcal{A}$ et $\mathcal{B}$. Plus précisément, la vitesse de convergence de $\mathcal{C}$ est égale à la vitesse de convergence du plus lent de $\mathcal{A}$ et $\mathcal{B}$. Sans perdre en généralité, supposons que $\mathcal{A}$ a une espérance de temps d'atteinte plus faible que $\mathcal{B}$. Donc, l'espérance de $\mathcal{C}$ est la même que $\mathcal{B}$. La DSP optimale pour $\mathcal{B}$ sera donc également optimale pour $\mathcal{C}$.

Résumé intermédiaire Considérons une vision générale du reste de la preuve. Nous allons à présent montrer que la distribution optimale pour chacun des deux premiers protocoles sus-cités est atteinte par la DSP uniforme. Ainsi, par compositions parallèles, tout prédicat issu de l'arithmétique de Presburger sera calculable de façon stationnaire en un temps optimal par la DSP uniforme. Ce résultat est produit par le fait que l'espérance du temps d'atteinte de ces protocoles est caractérisée par une fonction polynomiale en les p_i (*i.e.* les probabilités des arcs du graphe d'interaction).

Le temps d'atteinte moyen est polynomial Soit un protocole $\mathcal{P}$ et sa chaine de Markov correspondante $\mathcal{M}_{\mathcal{P}}$ donnés. On note l'état stationnaire $\mathcal{S}$. Considérant un état de départ $\mathcal{I}$, on pose le système à $m = \frac{n(n-1)}{2}$ inconnues suivant :

$$\begin{cases} f(p_1, \dots p_m) = \mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{S}}) \\ g(p_1, \dots p_m) = \sum_{p_i} p_i = 1 \end{cases}$$

On définit un chemin dans $\mathcal{M}_{\mathcal{P}}$ par une liste d'état de $\mathcal{M}_{\mathcal{P}}$: $\langle k_1, k_2, \dots, k_s \rangle$ pour $s \in \mathbb{N} \backslash \{0\}$. Soit $\mathcal{C}_s(\mathcal{I}, \mathcal{S})$ l'ensemble des chemins de longueur s menant de $\mathcal{I}$ à $\mathcal{S}$ sans passer par $\mathcal{S}$:

$$\mathcal{C}_s(\mathcal{I}, \mathcal{S}) = \{k_1, k_2, \dots, k_s | k_1 = \mathcal{I}, k_s = \mathcal{S}, \forall i \in \{1, \dots, s-1\}, k_i \neq \mathcal{S}\}$$

Il est donc possible d'en déduire une expression de l'espérance du temps d'atteinte (*i.e.* la somme, sur toutes les longueurs et tous les chemins de ces longueurs, de l'espérance de tous ces chemins pondérée par la probabilité de ce chemin de se produire) :

$$\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{S}}) = \sum_{s \in \mathbb{N} \backslash \{0\}} \sum_{c \in \mathcal{C}_s(\mathcal{I}, \mathcal{S})} \mathbb{E}_{\mathcal{I}}[\tau_{\mathcal{S}} | c] \cdot \mathbb{P}[c]$$

Comme le chemin dans $\mathcal{M}_{\mathcal{P}}$ est déterminé pour un $c \in C_s(\mathcal{I}, \mathcal{S})$, on a $\mathbb{E}_{\mathcal{I}}[\tau_{\mathcal{S}} | c] = s$. D'où :

$$\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{S}}) = \sum_{s \in \mathbb{N} \backslash \{0\}} s \cdot \mathbb{P}[C_s(\mathcal{I}, \mathcal{S})]$$

Soit $q_{A,B}$ la probabilité de passer de l'état A à l'état B dans la chaine de Markov $\mathcal{M}_{\mathcal{P}}$, on a :

$$\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{S}}) = \sum_{s \in \mathbb{N} \backslash \{0\}} s \cdot \sum_{c \in \mathcal{C}_s(\mathcal{I}, \mathcal{S})} q_{\mathcal{I}, k_2} \cdot q_{k_2, k_3} \cdot \dots \cdot q_{k_{s-1}, \mathcal{S}}$$

Dans la chaine de Markov $\mathcal{M}_{\mathcal{P}}$, toute transition dépend uniquement des interactions potentielles permettant de passer d'un état k à un état k', et donc de la somme des p_i correspondant à ces interactions. Formellement, par groupement d'état dans la chaîne $\mathcal{M}_{\mathcal{P}}$, où $\mathcal{P}$ correspond respectivement à chacun des deux protocoles présentés précédemment, $\mathcal{M}_{\mathcal{P}}$ peut être simplifiée pour ne prendre en compte que le premier terme du triplet (le bit de meneur) définissant l'état d'avancement du système. Ainsi, soit E l'ensemble des états de $\mathcal{M}_{\mathcal{P}}$ avec $E = \{0, 1\}^n$ pour n agents dans la population (Tous les états avec la même distribution des bits de meneur sont groupés au sein d'un même méta-état dans la chaîne de Markov apériodique résultante). Pour chaque transition de δ possible, le nombre de 1 peut soit être réduit (dans le cas où les deux agents ont leur bit de meneur à 1), soit rester constant (dans le cas où un agent possède un bit de meneur à 1, et l'autre agent à 0). Il existe donc deux sortes de transitions

dans $\mathcal{M}_{\mathcal{P}}$:

$$\forall e, e' \in E, q_{e,e'} = \begin{cases} \mathbb{P}[i_1 \to i_2] = p_{i_1,i_2} & \text{si} \begin{cases} \forall j \notin \{i_1, i_2\}, e_j = e'_j \\ e'_{i_1} = e_{i_2} = 1 \\ e_{i_1} = e'_{i_2} = 0 \end{cases} \\ \sum_{j \neq i_0 \wedge e_j = 1} \mathbb{P}[j \to i_0] = \sum_{j \neq i_0 \wedge e_j = 1} p_{j,i_0} & \text{si} \begin{cases} \forall j \neq i_0, e_j = e'_j \\ e_{i_0} = 1 \\ e'_{i_0} = 0 \end{cases} \\ 0 & \text{sinon} \end{cases}$$

Donc, quel que soit la transition q dans $\mathcal{M}_{\mathcal{P}}$, q est une application linénaire des p_i. Ceci implique que $\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{S}})$ (et par conséquent la fonction f) est un polynome en les p_i.

Caractérisation de la borne inférieure On cherche à minimiser la fonction $f(p_1, \ldots, p_m)$, en fonction des paramètres $(p_1, \ldots, p_m)$, avec la contrainte $\sum_{i=1}^{m} p_i = 1$, pour des $p_i \in]0, 1[$ (En réalité, par cette contrainte, il est même possible de déduire p_m des autres paramètres $(p_1, \ldots, p_{m-1})$).

Considérons une distribution des p_i dans le graphe d'interaction complet (telle que $\forall p_i, p_i > 0$). On note $D = \{(p_i)_{1 \leq i \leq n} \in]0, 1[^n, \sum_{i=1}^{n} p_i = 1\}$. Il s'agit d'un ouvert de $\mathbb{R}^n$. Or, le lemme suivant montre que la dérivée s'annule pour un minimum d'une fonction de classe C^1 sur un ouvert.

Lemme 5.3 *Soit $f_c : \mathbb{R}^n \to \mathbb{R}$, de classe C^1 sur U, un ouvert de $\mathbb{R}^n$. Si f_c admet un minimum ou maximum local en a, alors $f'(a) = 0$.*

Preuve – Étant donné que f_c est de classe C^1, $f(a + h) = f(a) + f'(a) \cdot h + o(\|h\|)$.

Raisonnons par l'absurde : si $f'_c(a) \neq 0$, alors il existe h (suffisamment petit) tel que pour tout $t \in [0, 1]$, $f'(a) \cdot t \cdot h > 0$ et $f'(a) \cdot (-t \cdot h) < 0$, donc $f'_c(a)$ ne peut être un minimum local, car ne l'est pas sur le segment (vectoriel) $[-h, h]$. ■

Il est évident que la fonction f est de classe C^1, étant un polynôme. Donc, le minimum de f sur l'adhérence de D est soit atteint au bord, soit en un vecteur $p*$ de l'intérieur pour lequel $\nabla(f)(p*) = 0$ (∇ représente le gradient d'une fonction). Dans le modèle des protocoles de population de base, le graphe d'interaction est complet. Nous avons donc supposé préalablement que $\forall i \in \{1, \ldots, m\}, p_i \neq 0$. Donc, le minimum de f n'est atteint que si $\nabla(f)(p*) = 0$ pour $p* \in D$. Ce minimum étant réalisé en un tel point $p*$, d'après le théorème des extrémas liés, nous avons également $\nabla(g)(p*) = 0$. Hors, par définition, g est une fonction constante. Nous pouvons donc en conclure donc que chaque dérivée partielle en un p_i est identique sur g. D'où, $p*$ est le point équidistribué (*i.e.* $\forall i, j \in [\![1, m]\!], p_i = p_j$).

D'où, quel que soit le protocole de population donné, le minimum de $\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{S}})$ est réalisé pour une distribution uniforme des $p_i, i \in [\![1, m]\!]$, soit la DSP uniforme. ■

En outre, ce résultat n'est pas uniquement valide pour les protocoles de population. Effectivement, le théorème suivant montre qu'une DSP uniforme implique également le meilleur temps de convergence pour tout protocole de communauté.

Theorème 5.2 *Pour toute fonction f calculable par un protocole de communauté, la borne inférieure du temps de convergence est atteinte avec une DSP uniforme dans MAPC.*

Preuve – Toute fonction symétrique et appartenant à $NSPACE(n \log n)$ est calculable de façon stationnaire par un protocole de communauté. Dans [GR07], il est rappelé que tout langage décidable par une machine de Turing non-déterministe utilisant $O(S \log S)$ cases sur le ruban peut être décidé par une extension non-déterministe des machines à modification de stockage (NSMM). Cette preuve inspirée de celle de [vEB89] dans le contexte déterministe. Un NSMM est spécifié par un alphabet d'entrée $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$, un ensemble fini de directions Δ et un programme, composé d'une liste consécutive d'instructions numérotées. Les entrées d'un NSMM sont les mots de Σ^* de longueur finie. Un NSMM représente un unique ordinateur (*i.e.* pas un système réparti) stockant un graphe orienté fini à degré sortant constant, possédant un sommet particulier appelé *centre*. Les arcs du graphe, appelés *pointeurs*, sont étiquetés par des *directions* distinctes de l'ensemble Δ. Toute mot $x \in \Delta^*$ peut être utilisé pour référencer un sommet – dénoté $p^*(x)$ – atteint à partir du centre en suivant les pointeurs correspondants à la séquence de caractères de x.

Ce modèle de machines possède la même classe d'équivalence que les protocoles de communautés [GR07]. Ainsi, il suffit de considérer les instructions de base utilisées par ces machines pour obtenir le comportement limite du protocole de communauté correspondant à un programme donné sur une NSMM. Ces instructions permettent à la machine de changer la structure du graphe en insérant des sommets, modifiant des pointeurs ou testant si deux chemins de pointeurs (*i.e.* deux mots de Δ^*) conduisent au même sommet. De surcroît, ces programmes sont séquentiels. Donc, l'enchaînement d'instructions exécutées en temps optimal entraîne logiquement un temps optimal d'exécution total du programme considéré.

La preuve de ce théorème repose sur le même squelette que celle du théorème 5.1, en considérant un ensemble de briques génératrices différent. Dans le cas des protocoles de communauté, il suffit de montrer que chaque instruction utilisable dans un programme de NSMM se simule en un temps optimal avec la DSP uniforme pour le protocole MAPC correspondant. Pour chacun de ces derniers, nous montrons que toutes les transitions de la chaîne de Markov associée sont des applications linéaires des p_i du DSP dans MAPC, et donc que le temps d'atteinte moyen de l'état stationnaire ($\mathbb{E}_{\mathcal{I}}(\mathcal{S})$) est un polynôme des p_i. Ainsi, le théorème des extremas liés associé à la contrainte $\sum_{i=1}^{m} p_i = 1$ permet d'affirmer que le temps d'atteinte moyen optimal est atteint pour une DSP equidistribué (autrement dit, uniforme).

La liste exhaustive des protocoles nécessaires pour simuler un programme de NSMM est présentée ci-dessous. Chaque protocole est associé à l'instruction correspondante dans les NSMM si elle existe, et les transitions de la chaîne de Markov correspondante sont analysées. Une spécification plus précise des différents protocoles nécessaires à la simulation d'une NSMM est donnée dans [GR07].

Fonctionnement général Les agents s'auto-organisent au sein de *bandes* disjointes. Chaque bande est une liste chaînée d'agents, où chacun d'entre eux possède un pointeur vers l'agent situé juste à droite et un autre vers celui le plus à gauche de la bande. Initialement, chaque agent est situé dans une bande propre de longueur 1. Chaque bande exécute une simulation de l'algorithme NSMM. Lorsqu'un agent situé le plus à droite d'une bande rencontre un second agent le plus à gauche d'une autre bande, ils assemblent les deux bandes en une plus grande qui reprend la simulation au début (tout calcul effectué précédemment dans les bandes plus petites est supprimé). Ainsi, *irrémédiablement*, il existera une bande unique contenant tous les agents et celle-ci simulera le NSMM aboutissant irrémédiablement à un calcul stationnaire de la valeur de sortie.

Ainsi, pour obtenir la valeur de sortie correcte de l'algorithme, il est nécessaire d'attendre la fusion de toutes les bandes du système. Le nombre de bandes ne pouvant que réduire, la probabilité de réduction est la probabilité qu'un agent situé à droite rencontre un agent situé le plus à gauche, soit l'application linéaire suivante : $\sum_{\substack{i \neq j \\ i \text{ à gauche d'une bande} \\ j \text{ à droite d'une bande}}} p_{i,j}$. Puis, afin de redémarrer la simulation, l'agent le plus à gauche est défini comme le centre du graphe. La probabilité de redémarrer la simulation sur l'agent c à partir de l'agent i le plus à droite de la première bande est $p_{i,c}$, qui est également

un application linéaire.

Nous décrivons maintenant succintement comment une bande simule un NSMM. Chaque sommet du graphe de NSMM est représenté par un agent dans la bande. Un agent, représentant le sommet v, stocke l'identifiant des agents représentant chacun des $|\Delta|$ sommets accessibles par un pointeur issu de v. Initialement le graphe consiste en un unique sommet, qui est le centre du graphe, stocké sur l'agent situé le plus à gauche de la bande. Cet agent est responsable de l'exécution de la simulation du programme du NSMM. Un champ $control$ fonctionnant comme un compteur d'exécution représente la progression de la simulation. À chaque fois que p achève une instruction, le champ $control$ de l'agent p est mis à jour avec la prochaine ligne à exécuter dans le programme du NSMM. La bande est également utilisée pour représenter le mot d'entrée du NSMM. Le NSMM simulé lit les caractères en entrée des agents dans l'ordre de la bande. Le centre possède donc également un champ $next$ représentant l'identifiant de l'agent le plus à gauche dont l'entrée n'a pas encore été consommée. Au démarrage de la simulation sur la bande, le champ $control$ de l'agent central est initialisé à la ligne 1 du programme et le champ $next$ à son propre identifiant (en tant qu'agent le plus à gauche de la bande).

new Cette instruction permet de créer un nouveau sommet dans le graphe, de le déclarer comme nouveau centre de celui-ci, et d'initialiser tous ces arcs vers l'ancien centre du graphe. L'agent p responsable de la simulation doit localiser un agent libre (*i.e.* un agent ne représentant pas de sommet, ou représentant un sommet inaccessible). Pour cela, p parcourt toute la bande et marque tous les agents comme *non-visité*. Puis, il lance un parcours en profondeur du graphe du NSMM, et marque tous les agents atteints comme *visité*. Il attend ensuite de rencontrer un agent q *non-visité* et recentre le graphe sur q (tous les pointeurs de q sont fixés sur p, et les champs $control$ et $next$ de q sont initialisés aux valeurs courantes de ceux de p). Pour le parcours de la bande comme pour le parcours en profondeur du graphe, l'ordre des interactions dans le MAPC correspondant est imposé. Supposons que les agents de la bande sont numérotés de i_0 à i_b. La probabilité pour l'agent central i_c d'interagir avec le noeud i_0 est p_{i_c,i_0}, puis celui d'interagir avec $i1$ est p_{i_c,i_1}, *etc*. Ainsi, l'espérance moyenne du temps d'atteinte de parcours complet de la bande et celle du parcours en profondeur est un polynôme en les p_i du MAPC. Quant à l'attente de rencontrer un q *non-visité*, la probabilité de rencontre correspond à l'application linéaire suivante : $\sum_{j \text{ non-visité}} p_{c,j}$. Le recentrage du graphe est effectué de manière locale et en temps constant (échange de paramètres durant l'interaction entre c et q).

recentre x où $x \in \Delta^+$. Cette instruction définit comme nouveau centre du graphe le noeud $p^*(x)$. Pour cela, le graphe est parcouru selon l'ordre déterminé par le mot x. Comme précédemment, le temps de parcours moyen est un polynôme en les p_i du MAPC et le recentrage est effectué en temps constant.

set $x\delta$ **to** y où $x, y \in \Delta^*$ et $\delta \in \Delta$. Cette instruction modifie le pointeur δ du sommet $p^*(x)$ afin qu'il pointe à présent vers le sommet $p^*(y)$. Il s'agit donc successivement d'un parcours du graphe selon l'ordre imposé par le mot y, puis un autre parcours selon celui du mot x et d'une modification du pointeur étiqueté par δ sur l'agent hébergeant le sommet $p^*(x)$. Comme précédemment, ces parcours sont effectués en temps polynomial des p_i du MAPC.

if $x = y$ **then goto** ℓ où $x, y \in \Delta^*$ et ℓ est un numéro de ligne du programme simulé. Cette instruction vérifie que les mots x et y permettent d'atteindre le même sommet du graphe. Si c'est le cas, le champ $control$ du noeud central est mis à jour à ℓ, sinon, il est incrémenté pour pointer vers la ligne suivante. De la même manière que l'instruction précédente, celle-ci est réalisée par deux parcours consécutifs, l'un menant à $p^*(x)$ et l'autre à $p^*(y)$. Le reste de l'instruction est réalisé en temps constant localement sur l'agent représentant le centre du graphe.

input $\ell_1, \ldots, \ell_r$ où $\ell_1, \ldots, \ell_r$ sont des numéros de lignes du programme simulé. Cette instruction lit le premier caractère non-lu du mot d'entrée et met à jour le champ *control* avec la ligne l_i si le caractère lu est σ_i. L'agent représentant le centre du graphe attend de rencontrer l'agent pointé par le champ $next$, correspondant à la probabilité de rencontre $p_{c,next}$, puis récupère le caractère d'entrée de $next$

et met à jour son champ $control$ en temps constant.
output o où $o \in \{0, 1\}$. Cette instruction arrête la machine et répond o. Si $o = 1$, alors la simulation de la machine est dans l'état d'acceptation, et le mot d'entrée appartient au langage décidé par le NSMM considéré. Sinon, dans le cas où $o = 0$, cette valeur est considérée comme provisoire et la bande est réinitialisée pour commencer une nouvelle simulation, afin de simuler potentiellement toutes les exécutions, dû au non-déterminisme du NSMM. L'exécution de cette instruction sur le MAPC correspondant ne dépend pas de l'ordre des interactions potentielles étant donné que c'est une fonction locale à un agent. Elle s'exécute donc en temps constant.
choose ℓ_0, ℓ_1 où ℓ_0, ℓ_1 sont des numéros de lignes du programme simulé. Cette instruction modifie le champ $control$ le faisant pointer vers la ligne ℓ_0 ou la ligne ℓ_1, choisie de façon non-déterministe. Comme pour l'instruction précédente, cette instruction est locale à l'agent représentant le centre du graphe, et ne dépend pas des p_i du MAPC.
Cas d'une exécution infinie Enfin, certaines simulations d'un NSMM peuvent ne jamais terminer. L'agent central du graphe est autorisé à redémarrer de manière non-déterministe la simulation en cours (de la même façon qu'en cas de sortie nulle). Comme les deux dernières instructions, cette décision est locale et la condition d'équité du modèle MAPC assure que si une exécution acceptante existe, elle sera simulée, et le centre renverra alors indéfiniment la valeur 1 en sortie. Ce choix étant non-déterministe, quelle que soit la DSP considérée pour le MAPC donné, la modification de l'espérance du temps d'atteinte de l'état stationnaire issue de ces choix de reinitialisation est donc indépendante du choix de cette DSP.

Donc, chaque protocole de communauté nécessaire à l'exécution d'un programme sur NSMM possède une chaîne de Markov associée composée uniquement de transitions qui sont des applications linéaires des p_i. Ainsi, par équivalence de classe, tous les protocoles de communauté ont un temps d'atteinte moyen de l'état stationnaire polynomial en les p_i (somme ou produits d'application linéaires, de polynômes en les p_i et de constantes). Donc, via le théorème des extrema liés, le temps d'atteinte moyen optimal pour n'importe quel MAPC est atteint avec la DSP uniforme. ■

Ces deux précédents résultats permettent de conclure qu'il est impossible d'obtenir une vitesse de convergence moyenne meilleure que celle obtenue par une distribution uniforme des interactions pour les protocoles de population et de communauté. Le dernier paragraphe de ce chapitre se penche sur une implication directe et pratique de ce résultat.

5.5 À propos de la pertinence du modèle de RWP

Le modèle de *points de navigation aléatoires* [BRS03] (*RWP* pour *Random WayPoint*) est excessivement utilisé pour évaluer les différentes contributions dans le domaine des réseaux mobiles, malgré le manque de réalisme évident de celui-ci. Dans ce paragraphe, nous montrons qu'une DSP uniforme dans MAPP[5] modélise une exécution du modèle de RWP. Au-delà de l'intérêt théorique de cette preuve, notre objectif est de souligner le fait que l'utilisation du RWP, pourtant généralement justifiée comme fournissant un modèle de mobilité moyen et représentatif (bien que non-réaliste), correspond en fait au meilleur des cas par rapport au temps de convergence des protocoles.

Theorème 5.3 *Le modèle des points de navigation aléatoires (RWP) est modélisé par une DSP uniforme dans MAPP.*

[5]Nous appelons dès lors MAPP l'ensemble des deux modèles MAPP et MAPC.

Preuve – Le modèle Random Way Point peut se définir formellement comme suit : Soit n agents mobiles de positions initiales $p_1^{(0)}, \ldots, p_n^{(0)}$. Au début de l'expérimentation, chaque agent tire aléatoirement une destination et une vitesse selon deux lois P_{pos} et P_{vit}, communes à tous les agents. Arrivé à sa destination, l'agent considéré tire un nouveau couple de destination et vitesse. De manière générale, la distribution des points destinations est uniforme sur l'espace considéré et la distribution des vitesses est tirée uniformément sur un intervalle donné [BRS03]. Ici, nous montrons le résultat dans un contexte plus générique : nous supposons uniquement que chaque tirage correspond à un système ergodique indépendant.

Pour tout agent x, nous pouvons définir $\gamma_x(t)$ la trajectoire de cet objet x dans l'espace. Cette trajectoire ne dépend que de la suite de couples $\langle (p_x^{(i)}, v_x^{(i)}) \rangle_{i \in \mathbb{N}^*}$. Bien que la distribution spatiale des noeuds dans le RWP réfléchissant (espace de mobilité borné – pas un tore – et rebond sur les bords) ne soit pas uniforme [BRS03], nous allons montrer que la probabilité de rencontre de deux agents quelconques est uniforme (les déplacement des agents étant i.i.d).

Soit x, y deux objets. Soit l'équation suivante pour une période T fixée donnant une expression de la moyenne temporelle :

$$\frac{1}{T} \int_0^T \mathbf{1}_{|\gamma_x(t) - \gamma_y(t)| \leq \varepsilon}(t) \cdot dt. \tag{5.3}$$

Dans cette équation, $\mathbf{1}_{|\gamma_x(t) - \gamma_y(t)| \leq \varepsilon}(t)$ représente la fonction indicatrice qui vaut 1 si x et y sont a une distance inférieure à ε, et 0 sinon. Si l'équation 5.3 tend vers une valeur constante ne dépendant pas du choix de (x, y), alors cette constante reste la même pour tout couple d'agents.

Considérons la loi de probabilité des trajectoires $\gamma_x(t)$ pour un agent x donné :

$$P_x = \left(\bigotimes_{i=1}^{\infty} p_x^{(i)} \right) \otimes \left(\bigotimes_{i=1}^{\infty} v_x^{(i)} \right).$$

Considérons le couple de trajectoire (γ_x, γ_y). C'est une variable aléatoire de loi $P_x \otimes P_y$. Il est donc possible de définir le système suivant :

$$\left(\left\langle (p_x^{(i)}, v_x^{(i)}, p_y^{(i)}, v_y^{(i)}) \right\rangle_{i \in \mathbb{N}^*}, P_x \otimes P_y, \lambda_\otimes \right)$$

où $\lambda_\otimes$ est la mesure produit invariante de Lebesgue. Ce système est un produit de systèmes ergodiques indépendants, c'est donc un système ergodique lui-même. Nous pouvons donc lui appliquer le théorème ergodique (ou loi des grand nombres, *i.e. la moyenne temporelle converge vers la moyenne spatiale*). Soit la moyenne spatiale définie comme suit :

$$\mathbb{E}[\mathbf{1}_{|\gamma_x - \gamma_y| \leq \varepsilon}] = \iint \mathbf{1}_{|\gamma_x - \gamma_y| \leq \varepsilon} \cdot dP_x(\gamma_x) \cdot dP_y(\gamma_y) \tag{5.4}$$

Pour presque tout chemin γ_x et γ_y, quand T tend vers ∞, l'équation 5.3 tend vers l'équation 5.4. Dans cette dernière équation, l'intègrale se fait sur l'ensemble de toutes les trajectoires possibles des agents x et y. Cette expression ne dépend donc plus de x et y mais uniquement de ε et des lois de probabilités P des trajectoires. Or, ces dernières sont identiques pour tous les agents du système.

En résumé, quel que soit le couple d'agents considéré, le nombre moyen de rencontre est le même que pour tout autre couple d'agents. Donc toutes les paires d'agents ont la même probabilité de se rencontrer au temps t. Ceci infère une distribution des rencontres uniforme, qui est par définition la DSP uniforme dans MAPP. ■

5.6 Conclusion

Dans ce chapitre, nous avons étudié l'impact des modèles de mobilité sur la vitesse de convergence des protocoles de population. Premièrement, nous avons introduit MAPP, une extension du modèle de base dans laquelle le graphe d'interaction est pondéré par chaque probabilité d'interaction de deux agents.

Nous avons ensuite étudié empiriquement l'impact significatif de divers modèles de mobilité sur la vitesse de convergence de trois protocoles de population classique (*i.e.* les primitives *ou*, *majorité* et *somme modulo 4*). Cette étude stochastique nous a permis de mettre en exergue des biais tant pour les modèles de rencontre que d'inter-rencontre.

De ces observations est apparu que la distribution uniforme des probabilités d'interaction induisait toujours la meilleure vitesse de convergence moyenne. Ainsi, nous avons prouvé formellement qu'une DSP uniforme permet d'atteindre la borne inférieure du temps moyen de convergence de tout protocole de population et de communauté.

Enfin, nous avons mis en avant que le modèle de RWP, largement utilisé dans les simulations de RCsF mobile, correspond en réalité à une DSP uniforme. Bien qu'il soit communément admis que ce modèle n'est pas réaliste, ce résultat montre que le RWP correspond en sus au meilleur des cas en terme de temps de convergence. Ceci remet potentiellement en cause la pertinence de ce modèle, utilisé le plus souvent comme un modèle représentatif des résultats escomptés sur un RCsF mobile réel.

Le modèle MAPP ouvre ainsi de nouveaux horizons dans le domaine de l'analyse fondamentale des RCsF mobiles. Le chapitre suivant apporte une vision plus pratique de ces réseaux. Par un rapprochement de ces modèles avec le paradigme épidémique, nous présentons une corrélation captivante entre la vision théorique des RCsF mobiles présentée précédemment et la vision pratique des systèmes répartis épidémiques introduite dans le chapitre suivant.

CHAPITRE 6

Équivalence des protocoles épidémiques et de population

Les contributions de ce chapitre proviennent de multiples observations. Principalement, nous avons constaté que les protocoles de population en général et les protocoles épidémiques, présentés ci-après, possèdent de nombreuses ressemblances, bien que les uns soient issus d'une approche théorique et les autres aient une origine plus pratique. Ceci nous mène à proposer une première classification de ces protocoles épidémiques.

Inspirés de cette remarque, nous présentons ensuite les motivations qui nous ont menés aux résultats de ce chapitre. Puis, après avoir étendu en une seconde classification, à granularité plus fine, celle de la section 6.1, nous montrons les relations d'équivalence existant entre tous ces modèles. Nous concluons ce chapitre par divers exemples de transfert de contributions issues d'un domaine vers l'autre.

6.1 Les protocoles épidémiques

Par analogie avec le cheminement d'une rumeur ou la propagation d'un virus, les *protocoles épidémiques* sont conçus autour d'un échange périodique d'informations entre les participants d'un système réparti, lequel peut tout aussi bien être filaire ou non. Dans le contexte des systèmes filaires, ces protocoles fournissent une trame fiable et robuste, même à grande échelle, pouvant être utilisée dans une large gamme d'application [KvS07], telles que la cohérence de bases de données, la diffusion d'informations, la construction de réseaux logiques, *etc.* [BBFK07a, BK07, DQA04, DGH+87, EHG+03, EGKM04, EGKM06, GVvS05, JB04, JVG+07, vR02, SCS06, VGvS05, WMI+07]. La robustesse des protocoles épidémiques reposent sur leur caractère aléatoire : dans cette classe de protocoles, chaque nœud du système échange périodiquement des informations avec un autre nœud choisi au hasard parmi un sous-ensemble connu du système. La taille de cet ensemble est classiquement d'un ordre de grandeur bien plus faible que celui de la taille du système. Le choix de ce sous-ensemble, la plupart du temps dénoté *vue de voisinage*, est cruciale pour l'efficacité de la dissémination épidémique [JVG+07, VGvS05].

Ces protocoles ont été également étudiés théoriquement afin d'extraire notamment certaines propriétés par analogie avec des modèles épidémiologiques et de la théorie de la percolation. À titre d'exemple, dans un réseau à topologie aléatoire, la vitesse de propagation épidémique d'une information est exponentiellement rapide [EGKM06] (*i.e.* la vitesse moyenne de dissémination est de l'ordre de $e^{-e^{-c}}$ si chaque nœud connaît $c \sim \log(N)$ autres nœuds, où N est la taille du système). Seulement, à notre connaissance, aucun modèle formel de protocoles épidémiques n'a encore été proposé. La majorité des résultats de cette classe de protocoles est observée empiriquement. De plus, aucune classification des différents protocoles existants n'a encore été proposée à ce jour. Proposer une telle classification des protocoles épidémiques est l'objectif de ce chapitre, inspirée des classes de protocoles induites par les modèles des protocoles de population et de communauté et fondée sur la puissance de l'échantillonnage des nœuds sous-jacent (ou *PS* pour *Peer Sampling*).

6.1.1 Historique des protocoles épidémiques

Initialement introduit pour disséminer de l'information, les protocoles épidémiques reposent sur un paradigme très simple mais très fiable et efficace. À l'origine, ce paradigme a été proposé dans le cadre de maintenance de base de données [DGH+87]. Dès lors, de nombreux travaux ont étendu ce modèle et le terme de *protocole épidémique* (ou *gossip-based protocol*) est aujourd'hui utilisé également pour tout protocole probabiliste, fondé sur des échanges périodiques d'informations entre participants [EGKM04].

Le principe de base de ce type de protocole est épuré : chaque participant possède un ensemble d'informations qu'il met à la disposition des autres, et a la possibilité d'en publier de nouvelles. L'objectif de chacun des participants est de collaborer afin d'obtenir des informations actualisées sur le réseau (par exemple, dans [DGH+87], connaître toutes les mises à jour de la base de données). Chaque nœud sélectionne périodiquement un autre nœud du réseau, avec lequel il échange des informations.

Ainsi, la propagation des informations possède le même comportement que la propagation d'*épidémie* dans un réseau d'interaction humain [RG05] (*e.g.* virus, rumeur, *etc.*).

6.1.2 Protocole épidémique générique

Dans ce modèle, chaque nœud est représenté par le même type de machine à états finis introduit au paragraphe 4.2.2, et ne peut mettre à jour son état que par l'appel d'une fonction *miseAJour*. Cette dernière primitive définit en quelque sorte une fonction de transition de l'états des nœuds lors de la découverte de nouvelles informations. Afin de garantir que tous les nœuds mettent à jour leur état, le principe fondamental de ces protocoles est la sélection périodique d'un nœud parmi un ensemble de nœuds connus, puis l'échange d'informations *locales* issues de chacun de ces deux participants. Ainsi, l'information s'affine à mesure que les échanges se font.

Ces derniers s'effectuent selon l'algorithme 6.1. Périodiquement, chaque nœud initie un *échange épidémique* avec un autre nœud du système. Cette communication s'effectue selon deux tâches, effectuées indéfiniment par tous les nœuds du système : un *processus actif* initiant la communication et un *processus passif* attendant une requête d'un nœud distant pour effectuer

Algorithme 6.1 : Protocole épidémique générique

Processus actif	***Processus passif***
pour chaque T *unités de temps* **faire**	
attendre t_r temps ($t_r \in [\![0; T-1]\!]$)	
p = *sélectionPair*()	**pour** *toujours* **faire**
Envoyer *sélectionInfo*$_A$(état) **à** p	**Recevoir** info$_q$ **de** q
Recevoir info$_p$ **de** p	**Envoyer** *sélectionInfo*$_P$(état) **à** q
$état$ = *miseAJour*(info$_p$)	$état$ = *miseAJour*(info$_q$)

l'échange.

En considérant une discrétisation du temps en unités, pour chaque *cycle* de T unités de temps, chaque nœud exécute le processus actif de l'algorithme 6.1 à un temps t_r choisi aléatoirement. Tout nœud du système connait via le PS[1] un sous-ensemble de ce dernier, celui-ci conservant une liste de *nœuds connus*, dits *voisins*. Dans le reste de ce manuscrit, l'ensemble de ceux-ci est appelé *vue* du nœud. Celle-ci est le plus souvent limitée à un nombre borné c de voisins, et/ou de cardinal d'ordre de grandeur faible par rapport à la taille du système. Parmi ceux-ci, le nœud initiateur, dénoté p_{actif} par la suite, obtient du PS un nœud p en utilisant la fonction *sélectionPair*, lui envoie un sous-ensemble des informations locales dont il dispose, et se met en attente de la réponse de p. En symétrie, tout nœud, recevant une requête d'échange d'un autre nœud q, transfère également ses informations locales à q. Ainsi, chacun des deux protagonistes de l'échange peut mettre à jour son état à l'aide de la fonction *miseAJour*.

En pratique, le sous-ensemble des informations à envoyer est désigné par une fonction *sélectionInfo*, qui peut être différente si le nœud se trouve en position d'initiateur (*sélectionInfo*$_A$), ou d'échangeur uniquement (*sélectionInfo*$_P$).

Ce modèle de protocole épidémique est dit de *push-pull*, étant donné que les deux nœuds participant à l'échange envoient de l'information, et pas uniquement l'initiateur comme dans la version *push*, ou uniquement le nœud échangeur dans la version *pull*.

Le squelette de ce protocole peut ainsi être adapté dans un grand nombre d'applications, en personnalisant le modèle d'échange (*push*, *pull* et *push-pull*) et en définissant les fonctions et primitives *sélectionInfo*, *sélectionPair* et *miseAJour*.

6.1.3 Deux grandes classes de protocoles épidémiques

Parmi les nombreuses fonctions que peuvent mettre en œuvre les protocoles épidémiques, nous avons extrait deux grandes classes, inférant une sorte de hiérarchisation de ces protocoles. Ainsi, nous proposons dans ce paragraphe une première classification de ces protocoles.

Cette hiérarchie à deux niveaux s'appuie sur deux caractérisations des fonctions *selectionPair*, *selectionInfo* et *miseAJour* de l'algorithme 6.1, présentées ci-avant. Plus spécifiquement, les protocoles épidémiques diffèrent par leur prérequis concernant l'anonymat des nœuds, lequel est pourvu par le PS sous-jacent. Le service d'échantillonnage PS des nœuds (*i.e.* la « boîte

[1] Le service d'échantillonnage permet d'obtenir un nœud du réseau avec lequel échanger de l'information. Il se comporte comme une sorte de « boîte noire » selon des heuristiques spécifiques.

noire » fournissant un nœud spécifique à partir d'un échantillon du réseau) peut aussi bien retourner n'importe quel échantillonnage nécessaire à l'exécution d'un protocole épidémique. Nous proposons donc deux classes de protocoles épidémiques, définies en fonction de la puissance du PS sous-jacent et donc de l'anonymat des nœuds.

De même que pour les protocoles de population et de communauté, il est possible de séparer en deux groupes les protocoles utilisant, ou non, des identifiants de nœuds. Nous revenons sur la pertinence de cette hiérarchisation au paragraphe 6.3. Cependant, il est évident que la classe ne nécessitant pas une mise en œuvre sur des nœuds identifiables est strictement incluse dans la seconde. Ci-après, nous décrivons un ensemble de travaux connexes aux protocoles épidémiques, lesquels sont catalogués selon leur appartenance à l'une des deux classes.

Les protocoles épidémiques sur des nœuds anonymes (PENA)

La première classe de protocoles épidémiques, présentée ici, ne considère que les protocoles permettant un anonymat complet des nœuds. Ainsi, à l'instar des protocoles de populations présentés ci-avant, les protocoles de la classe PENA mettent en œuvre des nœuds indiscernables les uns des autres, et doivent donc mener une auto-organisation ne reposant que sur l'état de ceux-ci. Dans ce cadre, les fonctions *selectionInfo* et *miseAJour* prennent en compte uniquement l'état local du nœud considéré. En revanche, la fonction *selectionPair*, n'ayant pas d'identifiants de nœuds à sa disposition dans la vue locale, est considérée dans ces cas comme une sorte de *boîte noire* permettant d'entrer en contact avec un autre nœud du système afin de procéder périodiquement à l'échange épidémique.

La plupart des premiers protocoles épidémiques entrent dans cette catégorie. C'est le cas de [DGH+87] précédemment cité. Plus récemment, toujours dans un contexte d'agrégation, Kempe *et al.* proposent une évaluation de protocoles épidémiques algébriques (tels que le calcul de la somme, moyenne, échantillonnage aléatoire, quantiles, *etc.*) [KDG03]. Le calcul de moyenne sur une épidémie uniforme a permis de développer des protocoles d'estimation de la taille du système à l'instar de [MMKG06]. Nous terminerons notre parcours non-exhaustif des PENA en citant les travaux relatifs à la diffusion. Dans les réseaux filaires, Eugster *et al.* proposent LPBcast, un protocole épidémique de dissémination probabiliste d'information [EHG+03]. Enfin, dans le cadre des RCsF mobiles et des MANET, plusieurs propositions de diffusion, fondées sur une propagation épidémique de l'information, ont été suggérées [BBFK07b, CRB01], créant un pont entre ces deux communautés.

Les protocoles épidémiques sur des nœuds identifiables (PENI)

À l'inverse, ont éclos de nombreuses contributions dans le cadre des protocoles épidémique, mises en œuvre par l'utilisation de données identifiant les nœuds. L'ensemble de ces propositions sera dénommé dès à présent la classe PENI. Ces données peuvent être de différentes natures en fonction de l'objectif considéré. Le plus souvent représenté sous la forme d'un entier naturel utilisé comme identifiant unique [JVG+07], ce dernier peut prendre l'aspect d'une adresse internet (IP) dans le cadre du routage [HHL02], de coordonnées géographique (physique ou logique) dans le cadre de construction de réseaux logiques [JB04, Riv07] ou même d'un ensemble de ressources disponibles dans un système de partage de ressources [SCS06]

ou de fichiers (profil sémantique) [BK07].

Enfin, l'identification dans le cadre filaire ou dans celui des RCsF peut permettre d'améliorer les performances des PENA classiques dans une optique de diffusion d'information sur le système [DQA04, WLJC05, WK06].

6.2 Motivation

Les protocoles de population et les protocoles épidémiques possèdent de nombreuses caractéristiques communes. Premièrement, les deux reposent sur une séquence d'interactions menée par un *ordonnanceur*. Dans le contexte des protocoles de population, ce dernier spécifie les interactions des agents de manière équitable dans un environnement mobile. Au sein des protocoles épidémiques, l'ordonnanceur fournit un service d'échantillonnage des nœuds, lequel approvisionne les nœuds participants aux échanges. De plus, l'objectif de chacun de ces protocoles est de faire émerger un comportement global à partir d'un ensemble d'interactions locales, de manière totalement décentralisée.

Bien qu'ils aient été étudiés par deux communautés différentes, d'autres nombreuses ressemblances entre ces deux modèles existent. Chacune de ces classes de protocoles reposent sur les propriétés suivantes :

- Un modèle complètement décentralisé ;
- Un ensemble d'agents possédant une capacité de stockage finie et interagissant deux-à-deux. Les agents sont mobiles à communication sans fil dans les protocoles de population. Ils sont statiques mais communiquent via un réseau dynamique sur une infrastructure fixe dans les protocoles épidémiques
- Une séquence d'interactions dirigée par un ordonnanceur équitable modélisant la mobilité des agents dans les protocoles de population, ou par un service d'échantillonnage de nœuds (PS) dans les protocoles épidémiques ;
- Une fonction spécifiant le procédé suivant lequel les données sont traitées au cours d'une interaction : ceci correspond à la fonction δ des protocoles de population et à la fonction *miseAJour* des protocoles épidémiques ;
- Un échange d'états durant l'interaction, lesquels correspondent à une valeur dans Q pour les protocoles de population et au retour de la fonction fonction *sélectionInfo* dans les protocoles épidémiques.

Dans la suite de ce chapitre, nous développons les contributions suivantes :

- Nous établissons une corrélation entre les protocoles de population en général et les protocoles épidémiques. Pour cela, nous enrichissons notre *classification* proposée au paragraphe 6.1 en considérant, non plus uniquement le service de PS, mais également le synchronisme des communications. Nous identifions ainsi quatre classes de protocoles épidémiques (*cf.* paragraphe 6.3) ;
- Nous montrons que les protocoles épidémiques asynchrones sur des nœuds anonymes sont *équivalents* au modèle initial des protocoles de population ;
- Nous montrons que les protocoles épidémiques sur des nœuds identifiables sont *équivalents* aux protocoles de communauté ;
- Grâce à ces équivalences, nous pouvons tirer profit du cadre théorique des protocoles

de population pour comprendre et analyser la puissance et les limitations des protocoles épidémiques. De même, nous pouvons exploiter les résultats obtenus dans le domaine des protocoles épidémiques pour tirer des conclusions quant à l'utilisation pratique des protocoles de population. Ceci permet de fournir des considérations à la fois théoriques et pratiques pour ces systèmes large-échelle : ce parallèle entre protocoles de population et épidémique peut ainsi être exploité pour des résultats existants et à venir.

– Dans cette optique, les résultats obtenus au chapitre précédent peuvent être étendus dans le contexte des protocoles épidémiques. Ainsi, nous pouvons conclure que le service d'échantillonage aléatoire des nœuds (*RPS* pour *Random Peer Sampling* [JVG+07]) est optimal en terme de vitesse de convergence des protocoles épidémiques. Cet exemple illustre clairement la façon dont la corrélation présentée ici peut être exploitée.

6.3 Une classification des protocoles épidémiques

En considérant les protocoles épidémiques, il est possible de faire un parallèle entre *(i)* les différences de caractéristiques des protocoles de population et de communauté, et *(ii)* la nécessité d'identifier les nœuds dans les protocoles épidémiques. Cette observation nous a permis de proposer une première classification : PENA et PENI (représentant les protocoles épidémiques sur des nœuds *anonymes* et *identifiables* – *cf.* paragraphe 6.1.3, page 119).

Afin d'établir un parallèle précis entre ces protocoles et les modèles de population et de communauté, il est nécessaire d'étendre cette classification à une granularité plus fine.

6.3.1 Entre synchronisme et asynchronisme

Pour chacune des deux classes sus-citées, nous considérons deux sous-classes de protocoles, caractérisées par le canal de communication. Nous considérons deux types complémentaires que nous dénotons *synchrone* et *asynchrone* par la suite.

Dans notre modélisation, le canal *synchrone* est modélisé par des délais de transmission de messages bornés. Ainsi, nous pouvons considérer les communications comme atomiques *i.e.* le temps de transmission d'un message entre l'émetteur et le récepteur est négligeable. Dans ce cas, il est possible de considérer que l'exécution de l'échange d'information (*i.e.* exécution parallèle des processus passif et actif lors d'un échange épidémique) est atomique également. Dans ce type de communication, il est donc possible de tirer profit de la périodicité des échanges (*cf.* description de l'algorithme épidémique, paragraphe 6.1, page 117).

À l'inverse, un canal de communication *asynchrone* ne conditionne pas le délai de transmission d'un message. Ainsi, un message émis par un nœud peut être reçu par le nœud destinataire après un temps excessivement long. L'unique hypothèse imposée à ce modèle consiste en un délai de transmission fini. Tout message envoyé ne peux être différé indéfiniment.

Dans le cadre asynchrone, un nœud engageant un processus actif ne peut déterminer une borne de temps d'exécution de ce processus. Ainsi, il est nécessaire de décrire le comportement d'un nœud recevant une nouvelle requête d'échange alors qu'il est déjà engagé dans un autre échange. Étant donné que les nœuds sont représentés par des machines à états finis, nous ne pouvons considérer une temporisation de délivrance des messages dans ce contexte. En effet, la mémoire tampon permettant de stocker les messages en attente est également de taille finie.

Algorithme 6.2 : Gestion des communications asynchrones

```
1  échangeCourant = faux;
2  tant que vrai faire
3      À la réception de la requête d'échange d'un nœud faire
4          si échangeCourant alors
5              Envoyer (Refus);
6          sinon
7              échangeCourant = vrai ;
8              Envoyer (Accept);
9      À la réception de la fin d'un échange faire
10         échangeCourant = faux ;
```

Ainsi, les délais de transmission pouvant être potentiellement élevés, cette mémoire tampon représente une source de débordement possible, et *a fortiori* de perte de message.

Afin de simplifier le cadre d'étude, et en présence d'une perte potentielle de message, nous modélisons les échanges des protocoles épidémiques comme exclusifs. Ce comportement est formalisé par l'algorithme 6.2. Ainsi, lorsqu'un nœud est en cours d'échange (*i.e.* échangeCourant = *vrai*), toute tentative de communication sera ignorée jusqu'à l'aboutissement de la procédure d'échange. Le refus ou l'acceptation de la procédure d'échange est envoyé au nœud initiateur afin que celui-ci ne reste pas en état d'attente indéfiniment. Comme nous ne considérons pas les défaillances dans notre modèle, tout message envoyé sera irrémédiablement reçu dans un temps fini par le nœud destinataire, et ainsi, aucun nœud ne restera indéfiniment en attente.

6.3.2 Puissances des classes de protocoles épidémiques

Cette différence de canal de communication entre les nœuds induit une inégalité des puissances de calcul des protocoles épidémiques, en fonction de la présence de synchronisme ou non. Ainsi, nous obtenons donc une nouvelle classification, plus fine que celle présentée au paragraphe 6.1, composée de quatre classes de protocoles distinctes :

syncPENA Communication synchrones et nœuds anonymes ;
asyncPENA Communication asynchrones et nœuds anonymes ;
syncPENI Communication synchrones et nœuds identifiables ;
asyncPENI Communication asynchrones et nœuds identifiables.

La figure 6.1 représente les ordres de puissances de ces quatre modèles, en fonction de l'anonymat des nœuds et du modèle de communication considéré.

D'une part, il est évident que la puissance de la classe des protocoles épidémiques avec identifiants est plus forte que celle sans identifiants. En effet, l'utilisation d'identifiants de nœuds permet de résoudre des calculs répartis impossibles dans un contexte d'anonymat (*e.g.* calcul d'exponentielle, construction de voisinage logique, *etc.*). Ainsi, nous avons :

$$\text{asyncPENA} \prec \text{asyncPENI} \quad \text{et} \quad \text{syncPENA} \prec \text{syncPENI}$$

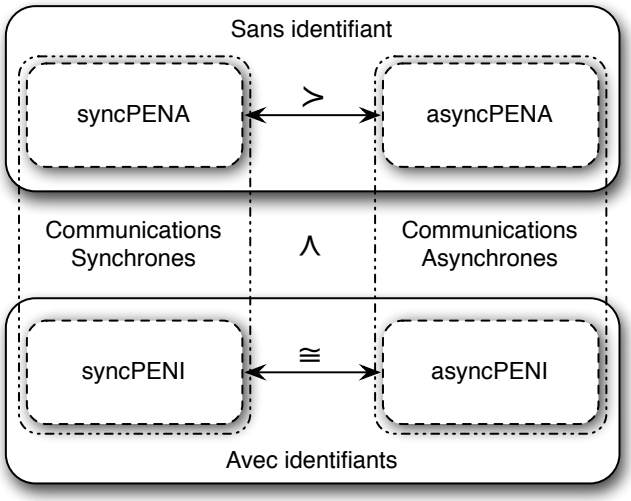

FIG. 6.1 – Schématisation des relations de puissance des protocoles épidémiques.

D'autre part, au sein de la classe des protocoles épidémiques sans identifiant, il est possible de tirer profit de la périodicité des échanges afin d'augmenter la puissance de calcul de ces protocoles. Par exemple, il est possible avec syncPENA d'instaurer une notion de temps global du système, grâce à la structure en cycle, mais pas avec asyncPENA. Ainsi, il est trivial de conclure que :

$$\text{asyncPENA} \prec \text{syncPENA}$$

Enfin, dans le cadre de PENI, nous montrons dans la preuve du lemme 6.4 (page 128) que l'identification des nœuds permet de simuler le synchronisme du canal de communication. D'où, nous avons :

$$\text{asyncPENI} \cong \text{syncPENI}$$

Cette classification et les relations de puissance en résultant sont ainsi résumées en figure 6.1.

Nous allons à présent montrer les liens des classes présentées ci-avant avec les protocoles de population et de communauté.

6.4 Combler le fossé entre les protocoles épidémiques et les protocoles de population

Pour résumer, dans PENA, l'échantillonnage de nœuds (PS) fournit un nœud destinataire, avec qui communiquer, à tout nœud initiant un processus actif d'échange, sans tenir compte des identifiants. Si le résultat du PS garantit que tout couple d'interaction peut indéfiniment advenir, alors un protocole de PENA ressemble à un protocole de population issu du modèle de base.

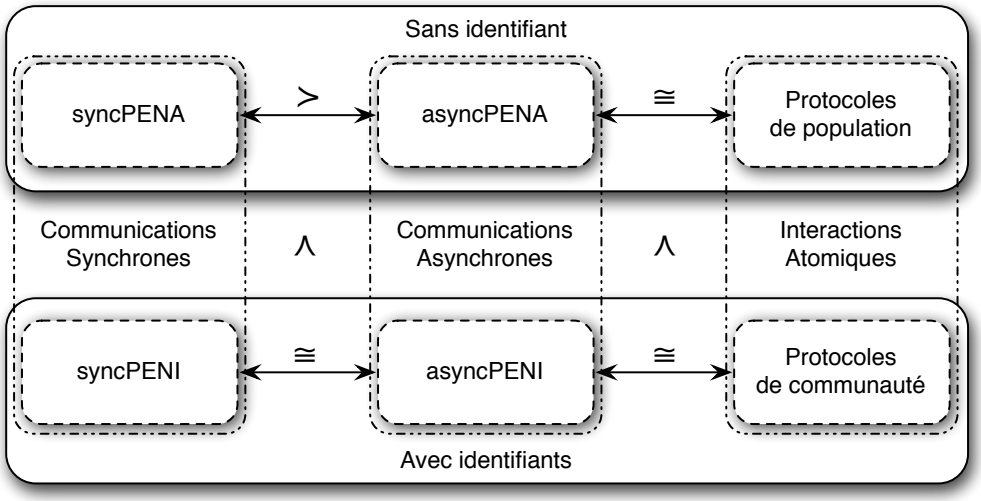

FIG. 6.2 – Schématisation des relations de puissance de tous les modèles.

À l'inverse, un protocole de PENI nécessite un PS fournissant un nœud destinataire à partir d'un sous-ensemble de pairs clairement identifiés. Ceci implique que la procédure de sélection du nœud échangeur ou de mise à jour des données nécessitent des identifiants, ou d'autres informations complémentaires, afin d'obtenir la fonctionnalité donnée. Les protocoles de communauté, décrits auparavant, correspondent parfaitement à cette description.

Dans ce paragraphe, nous démontrons la véracité de ces équivalences, en fonction de la classification présentée ci-avant. Nous pouvons ainsi étendre le domaine de la figure 6.1 aux protocoles de population et de communauté. Cette nouvelle classification est présentée en figure 6.2. Elle permet de rendre compte que les protocoles de population (et de communauté), qui apporte un point de vue théorique, et les protocoles épidémiques, reposant sur un fondement pratique, sont équivalents et décomposables de la même manière.

6.4.1 Equivalence entre les protocoles de population et de asyncPENA

Dans cette sous-partie, nous prouvons que le modèle de base des protocoles de population est équivalent aux protocoles de la classe asyncPENA.

Theorème 6.1 *Une fonction est calculable par un protocole de population si, et seulement si, elle peut être calculée par un protocole épidémique à nœuds anonymes via un canal de communication asynchrone (asyncPENA).*

Preuve – Afin de prouver cette équivalence, nous considérons les fonctions calculables par les protocoles de population et celles calculables par les protocoles de asyncPENA. Ainsi, nous prouvons ci-après dans les lemmes 6.1 et 6.2 qu'ils partagent la même classe d'équivalence. En réalité, nous montrons que la classe des fonctions calculables par les protocoles de population est un sous-ensemble de celles calculables par asyncPENA, et *vice-versa*. ■

En premier lieu, considérons l'implication PP $\prec$ asyncPENA, par le lemme suivant :

Lemme 6.1 *Si f est calculable par un protocole de population, alors il existe un protocole de asyncPENA qui peut calculer f.*

Preuve – Soit $\mathcal{P}$ le protocole de population calculant f et défini par le 7-uplets $(\Lambda, \Sigma, Y, Q, \iota, \omega, \delta)$. Considérons le protocole épidémique $\mathcal{G}$ décrit ci-dessous. Nous devons montrer que $\mathcal{G}$ simule $\mathcal{P}$.

Un sous-ensemble de définitions communes : Chaque agent dans Λ est hébergé par un nœud spécifique de $\mathcal{G}$. Les ensembles d'entrées, de sorties et d'états sont les mêmes dans $\mathcal{G}$ que dans $\mathcal{P}$. *A fortiori*, les fonctions d'association ι et ω restent identiques dans $\mathcal{G}$.

Traitement de la fonction de transition : la fonction *miseAJour* de $\mathcal{G}$ est définie à partir de δ : soient l et r deux nœuds du système, participants à un échange commun au temps t. Supposons que l initialise l'échange épidémique avec r. Soit p_l (respectivement p_r) l'information sélectionnée par la fonction *sélectionInfo* à partir de l'état de l (respectivement de r). Ainsi, étant donné que l exécute le processus actif, la fonction *miseAJour* retourne localement la troisième entrée du 4-uplet $(p_l, p_r, p'_l, p'_r) \in \delta$, et l'état de l devient alors p'_l. Sur le nœud distant r, un appel à la fonction *miseAJour* durant le processus passif retourne la dernière entrée du même 4-uplet (p_l, p_r, p'_l, p'_r).

Ainsi, la suite de configurations des populations est valide et représente une évolution correcte du protocole de population $\mathcal{P}$, puisque qu'elle simule la fonction de transition δ, en utilisant uniquement des interactions par paire.

À propos de la condition d'équité : La dernière hypothèse à vérifier est la condition d'équité de l'ordonnanceur. Dans asyncPENA, l'ordonnancement des interactions est uniquement conditionné par l'ordre des échanges épidémiques, lequel est défini *(i)* par la fonction *sélectionPair*, *(ii)* par la génération aléatoire des temps d'attente t_r, mais surtout *(iii)* par l'environnement asynchrone entraînant la perte de certains échanges épidémiques. Dans ce contexte, toutes les possibilités d'ordonnancements finis d'interaction ont une probabilité non nulle d'advenir. Ainsi, toute transition possible entre deux configurations du système $C \rightarrow C'$ a une probabilité non nulle de se produire. Donc, si la configuration C apparaît infiniment souvent durant l'exécution de ce protocole dans le contexte de simulation sus-cité, alors C' apparaîtra également infiniment souvent dans cette exécution. La condition d'équité est donc respectée.

Pour conclure, considérons le protocole épidémique $\mathcal{G}$ présenté ci-dessus, associé à un PS dans un environnement asynchrone. Alors, $\mathcal{G}$ simule le protocole de population $\mathcal{P}$, lequel calcule la fonction f. Ainsi, pour toute fonction calculable à l'aide d'un protocole de population, il existe un protocole épidémique de syncPENA qui calcule de façon stationnaire cette fonction. ■

En second lieu, montrons à présent l'inverse du lemme 6.1, correspondant à la seconde implication du theorème 6.1 : asyncPENA $\prec$ PP.

Lemme 6.2 *Si f est calculable par un protocole de asyncPENA, alors il existe un protocole de population qui peut calculer f.*

Preuve – Soit $\mathcal{G}$ un protocole épidémique de asyncPENA qui calcule une fonction f spécifique par l'utilisation des primitives *sélectionInfo* et *miseAJour*. Comme nous l'avons introduit précédemment, dans notre modèle épidémique, les nœuds sont modélisés par des machines à états finis.

Définition des domaines de la fonction de transition : Le domaine de définition de *misAJour* est fini (et correspond exactement au produit cartésien de $\mathcal{D}_S$, l'ensemble des états des nœuds, avec $\mathcal{D}_E$, l'ensemble image de *sélectionInfo*). De plus, étant donné que la relation *miseAJour* est une application,

son ensemble image est également fini par définition. À partir de ces ensembles, nous définissons $\mathcal{D}_{\mathcal{G}}$, un sous-ensemble spécifique du produit cartésien entre le domaine et le co-domaine de la fonction *miseAJour* (*i.e.* $\mathcal{D}_{\mathcal{G}}$ contient tous les couples ordonnés tels que le premier membre de chaque couple est un élément du domaine de *miseAJour* et le second membre est l'image du premier membre par la même fonction *miseAJour*). Plus formellement, $\mathcal{D}_{\mathcal{G}} \subseteq (\mathcal{D}_S \times \mathcal{D}_E) \times \mathcal{D}_S$. Ainsi, $\mathcal{D}_{\mathcal{G}}$ est fini et contient toutes les évolutions possibles de l'état d'un nœud, à partir de la connaissance de l'état du nœud distant participant à l'échange ($\mathcal{D}_E$).

Simulation par un protocole de population : Nous allons dès lors montrer comment concevoir un protocole de population $\mathcal{P}$ permettant de simuler $\mathcal{G}$. $\mathcal{P}$ est représenté par le 7-uplet $(\Lambda, \Sigma, Y, Q, \iota, \omega, \delta)$, défini ci-après. Considérons un graphe d'interaction complet Λ. L'ensemble des états des agents dans $\mathcal{P}$ est identique à l'ensemble des états des nœuds, *i.e.* $Q = \mathcal{D}_S$. De même, Σ et Y sont les mêmes que les ensembles d'entrées et de sorties de $\mathcal{G}$, si ceux-ci existent. Dans ce cas, ι et ω restent identiques aux fonctions associant respectivement l'ensemble des entrées de $\mathcal{G}$ à $\mathcal{D}_S$, et $\mathcal{D}_S$ à l'ensemble des sorties de $\mathcal{G}$. À l'inverse, si aucun ensemble d'entrées et sorties n'est défini dans $\mathcal{G}$, alors $\Sigma = Y = \mathcal{D}_S$ et $\iota \equiv \omega$ correspondent à la fonction identité. Enfin, la fonction de transition δ est définie par :

$$\forall (s_l, s_r, s'_l) \in \mathcal{D}_{\mathcal{G}}, \exists (s_r, s_l, s'_r) \in \mathcal{D}_{\mathcal{G}} \quad \text{tels que} \quad (s_l, s_r, s'_l, s'_r) \in \delta.$$

À propos de la périodicité de l'ordonnanceur équitable : La périodicité des échanges est une caractéristique intrinsèque aux protocoles épidémiques. Cependant, la principale différence entre asyncPENA et syncPENA repose sur la potentielle perte temporaire de cette périodicité, en raison de délais de transmission trop grands. Ainsi, comme il a été explicité dans la preuve du lemme 6.1, étant donné qu'aucune hypothèse de périodicité ne peut être considérée dans un environnement asynchrone, l'ordonnanceur équitable du protocole de population considéré est suffisant pour aboutir à une exécution correcte de $\mathcal{G}$, via $\mathcal{P}$ défini ci-dessus.

Ainsi, il existe un protocole de population $\mathcal{P}$, simulant le protocole $\mathcal{G}$ de asyncPENA, lequel calcule la fonction f. D'où, pour toute fonction calculable par un protocole de asyncPENA, il existe un protocole de population qui calcule de façon stationnaire cette même fonction. ■

Si on désire établir l'équivalence entre les protocoles de population et syncPENA, il est nécessaire d'imposer une restriction supplémentaire à l'ordonnanceur des protocoles de population. En effet, ce dernier doit être contraint de générer un ordre respectant la périodicité des protocoles de syncPENA (un échange épidémique par nœud et par cycle de T unités de temps). Il est possible d'exprimer cette *condition de périodicité* de la manière suivante. Soit Γ l'ensemble des agents du système et $\mathcal{I}$ la séquence d'interactions générée par l'ordonnanceur (*i.e.* $\mathcal{I}$ est une suite infinie de couple d'agents représentant l'ordre dans lequel les interactions ont lieu pour une exécution donnée).

$$\forall p \in \Gamma, \forall j \in \mathbb{N}, \exists i \in [\![1; |\Gamma|]\!], \exists q \in \Gamma, \mathcal{I}[i \cdot j] = (p, q).$$

Intuitivement, l'équation précédente signifie que chaque agent sera l'initiateur d'un échange une unique fois durant un cycle. En réalité, étant donné qu'aucune notion de temps n'existe dans le modèle des protocoles de population, nous divisons la séquence d'interactions en blocs de taille $|\Gamma|$, au sein desquels chaque agent n'apparaît qu'une unique fois en tant qu'initiateur (*i.e.* en tant que premier élément du couple).

6.4.2 Equivalence entre les protocoles de communauté et de PENI

En suivant la même démarche, dans ce paragraphe, nous prouvons le théorème suivant permettant de conclure l'équivalence entre les protocoles de communauté et les protocoles épidémiques de PENI. En effet, nous montrons dans la preuve du lemme 6.4 qu'il est possible de simuler la périodicité des protocoles de syncPENI avec un protocole de asyncPENI (ces deux classes sont alors équivalentes).

Theorème 6.2 *Une fonction est calculable par un protocole de communauté si et seulement elle peut être calculée par un protocole épidémique à nœuds identifiables (PENI).*

Preuve – Comme pour le théorème 6.1, la preuve du théorème 6.2 est immédiate par les assertions des lemmes 6.3 et 6.4, lesquels montrent respectivement les deux implications de cette équivalence. ■

Considérons de prime abord la première implication de ce théorème. Inspiré du lemme 6.1, la preuve du résultat suivant est relativement triviale.

Lemme 6.3 *Si f est calculable par un protocole de communauté, alors, il existe un protocole de PENI qui peut calculer f.*

Preuve – La seule différence entre les protocoles de population et les protocoles de communauté consiste en la définition de l'ensemble des états (*i.e.* $Q = B \times U^d$) et les deux contraintes imposées à l'utilisation des identifiants dans ceux-ci (*i.e.* la partie appartenant à U^d ne peut être utilisée librement). Ainsi, nous pouvons nous inspirer de l'esquisse de la preuve du lemme 6.1 pour montrer qu'un protocole de PENI peut être concu pour simuler un protocole de communauté $\mathcal{C}$ donné. Pour cela, nous considérons également la fonction *sélectionPair* comme une sorte de « boîte noire » fournissant un ordonnancement équitable. Par contre, les fonctions *selectionInfo* et *miseAJour* sont dorénavant définies respectivement sur les domaines $\mathcal{D}_S = B \times U^d$ (en lieu et place de B dans la version épidémique anonyme) et $\mathcal{D}_S \times \mathcal{D}_E$.

Ceci n'enfreint donc pas la définition des protocoles de PENI (synchrone ou asynchrone), étant donné que l'information additionnelle relève uniquement de la présence d'identifiants uniques des agents. Cette condition est requise dans la version non-anonyme des protocoles épidémiques. ■

Enfin, montrons à présent l'inverse du lemme 6.3, correspondant à la seconde implication du théorème 6.2.

Lemme 6.4 *Si f est calculable par un protocole de PENI, alors il existe un protocole de communauté qui peut calculer f.*

Preuve – Soit $\mathcal{G}$ un protocole épidémique de PENI donné. Donc, chaque nœud du système possède un identifiant unique. Considérons le protocole de communauté $\mathcal{C}$ suivant.

Hypothèses préliminaires sur $\mathcal{C}$: Nous supposons que, dans le protocole de communauté $\mathcal{C}$, il existe un unique agent possédant un identifiant spécifique $id_{\mathcal{L}}$. Cet agent est considéré par tous les autres comme le *meneur*. De plus, dans ce protocole de communauté spécifique, le meneur a la connaissance de la taille du système, noté n dans la suite (Cette hypothèse n'est nécessaire que pour la réalisation des barrières de synchronisation : elle peut être relâchée par l'utilisation du mécanisme

d'horloges probabilistes proposé pour les protocoles de population dans [AAE06a]). Enfin, la vue de voisinage d'un nœud est modélisée par un ensemble de $d-1$ identifiants dans le modèle des protocoles de communauté (une cellule du d-uplet dans U^d est réservée au stockage de son propre identifiant).

Résumé des prérequis de l'état d'un agent : En sus de l'état d'un nœud dans $\mathcal{D}_S$, chaque agent doit maintenir :

- une valeur binaire $gc_{\text{parité}}$ permettant de mémoriser la parité du cycle épidémique courant ;
- une valeur ternaire $gc_{\text{progression}}$ stockant l'état de l'échange épidémique d'un agent dans le cycle courant. $gc_{\text{progression}}$ peut prendre successivement les trois valeurs suivantes : *(i)* $\grave{a}Faire$ si le processus actif n'a pas encore été exécuté dans le cycle courant, *(ii)* $Fait$ si au contraire, il a été exécuté ou *(iii)* $Attente$ dans le cas où le nœud est entre deux cycles épidémiques (*i.e.* l'agent attend à la barrière de synchronisation que tous les autres nœuds aient exécuté leur processus actif) ;
- un identifiant d'agent id_{suivant} représentant l'identifiant du prochain agent avec qui l'échange doit avoir lieu ;
- et par ailleurs, l'agent meneur $\mathcal{L}$ conserve un compteur gc_{compteur} utilisé dans le processus de synchronisation des cycles.

Pour résumer formellement, l'ensemble des états des agents est défini par

$$Q = \mathcal{D}_S \times \{vrai, faux\} \times \{\grave{a}Faire, Fait, Attente\} \times U \times [\![1; n]\!] \times U^d$$

i.e. le produit cartésien entre *(i)* le domaine de la fonction *miseAJour* (représentant l'état d'un nœud), *(ii)* le domaine de $gc_{\text{parité}}$, *(iii)* le domaine de $gc_{\text{progression}}$, *(iv)* l'ensemble U des identifiants pour id_{suivant}, *(v)* le domaine de gc_{compteur} et finalement, *(vi)* U^d pour stocker la vue de voisinage d'un agent. À l'initialisation, chaque agent fixe $gc_{\text{parité}}$ à $faux$, $gc_{\text{progression}}$ à $\grave{a}Faire$ et id_{suivant} à $id_{\mathcal{L}}$. De plus, gc_{compteur} est initialisé à 0 pour l'agent meneur $\mathcal{L}$.

Simulation d'un cycle épidémique avec $\mathcal{C}$: Nous décrivons à présent le comportement d'un agent durant un cycle épidémique, en fonction de sa valeur de $gc_{\text{progression}}$. Considérons le cas où $gc_{\text{progression}} = \grave{a}Faire$. L'agent considéré n'a donc pas encore exécuté son processus actif durant ce cycle épidémique. Pour ce faire, il attend de rencontrer son prochain partenaire d'échange, dont l'identifiant est mémorisé dans id_{suivant}. Pour toute interaction (id_1, id_2), l'agent correspondant à id_1 verifie si $gc_{\text{progression}} = \grave{a}Faire$. Si c'est le cas, il contrôle si $id_2 = id_{\text{suivant}}$. Uniquement dans ce cas, id_1 et id_2 vont mettre à jour leur état par la fonction de transition δ, laquelle est définie par

$$\forall (q_1, q_2) \in \mathcal{D}_S^2, \quad (q_1, q_2, \textit{miseAJour}(q_1, \textit{sélectionInfo}(q_2)), \textit{miseAJour}(q_2, \textit{sélectionInfo}(q_1)) \in \delta.$$

(où q_1 et q_2 représentent respectivement les sous-états dans $\mathcal{D}_S$ de ces agents). Toutes les transitions possibles apparaissent dans $\mathcal{D}_{\mathcal{G}}$, défini dans la preuve du lemme 6.2. À la fin de l'interaction, id_1 et id_2 ont donc mis à jour leur état en fonction du précédent (q_i) et de celui du nœud distant (*sélectionInfo*(q_j)). Pour finir, id_1 place $gc_{\text{progression}}$ à $Fait$. Dans tous les autres cas ($id_2 \neq id_{\text{suivant}}$), rien ne sera exécuté. Pour résumer, chaque agent est en attente tant qu'il n'est pas en relation avec l'agent identifié par id_{suivant}, puis échange ses informations avec celui-ci, et enfin attend le passage au cycle suivant.

Comment simuler la périodicité de T : Le problème subsistant consiste en la simulation de cycles dans le contexte des protocoles de communauté. Dans ce modèle, les intervalles de temps de taille T peuvent être simulés par l'utilisation de barrières de synchronisation. Dans le reste de cette preuve, nous présentons comment établir un *rendez-vous* entre tous les agents, puis comment chacun d'entre eux passe au cycle épidémique suivant. Pour cela, nous introduisons le comportement des agents dans les cas où $gc_{\text{progression}} \neq \grave{a}Faire$.

Considérons un agent id. Après l'exécution de son processus actif, id passe dans l'état $Fait$. Dans cet état, il se met en attente jusqu'à la prochaine interaction avec l'agent meneur $id_{\mathcal{L}}$. Au cours

de celle-ci, id fixe sa valeur de $gc_{\text{progression}}$ à $Attente$ et $id_{\mathcal{L}}$ incrémente gc_{compteur} de 1. Ainsi, tous les agents vont irrémédiablement se stabiliser dans l'état $Attente$, et gc_{compteur} convergera vers n (le nombre d'agents dans la population). Une fois la convergence atteinte, tous les agents ont donc atteint la barrière de synchronisation.

À partir de ce rendez-vous, $id_{\mathcal{L}}$ permet aux agents de commencer leur prochain cycle épidémique de la façon suivante. Il intervertit sa propre valeur de $gc_{\text{parité}}$ à la valeur opposée, $gc_{\text{progression}}$ à la valeur $\grave{a}Faire$, id_{suivant} à la valeur retournée par *sélectionPair* et enfin gc_{compteur} à 0. À partir de ce point, à chaque fois qu'un agent dans l'état $Attente$ interagit avec l'agent meneur possédant une valeur opposée de $gc_{\text{parité}}$, ce premier agent passera dans cycle épidémique suivant par inversion de sa valeur de $gc_{\text{parité}}$ et en fixant $gc_{\text{progression}}$ et id_{suivant} respectivement à $\grave{a}Faire$ et à la valeur retournée par *sélectionPair*. Ainsi, tous les agents vont irrémédiablement quitter l'état $Attente$ du dernier cycle et seront prêt à effectuer leur prochain échange épidémique.

Pour conclure, si $\mathcal{G}$ calcule une fonction f, alors le protocole de communauté $\mathcal{C}$ simule le comportement de $\mathcal{G}$ et ainsi calcule également la fonction f. ■

La dernière étape de cette preuve nous permet de montrer également qu'un protocole épidémique de PENI dans un environnement asynchrone est capable de simuler un protocole de PENI dans un environnement synchrone. Or, il est évident que asyncPENI $\preceq$ syncPENI (pour les mêmes raisons que la classe PENA). Ainsi, nous pouvons effectivement conclure que asyncPENI $\cong$ syncPENI, et par conséquent, que les protocoles de communauté sont équivalents à tous les protocoles de PENI (asyncPENI $\cup$ syncPENI). Nous avons ainsi démontré l'intégralité des relations de la classification présentée en figure 6.2 (page 125).

6.4.3 Un impact potentiel pour les deux domaines

Les équivalences en amont sont de la plus grand importance dans le domaine des protocoles épidémiques. Elles permettent notamment de définir clairement ce qui peut être calculable par un protocole épidémique anonyme, *i.e.* toute fonction issue de l'arithmétique de Presburger peut ainsi être calculable par un membre de la classe PENA. Nous pouvons également conclure qu'aucune autre fonction ne peut être calculable par un protocole de asyncPENA [AAD$^+$06, AAE06b]. Par ailleurs, les résultats empiriques sur la vitesse de convergence et la mise en pratique des protocoles épidémiques peuvent être immédiatement appliqués afin d'évaluer l'efficacité des protocoles de population et de communauté.

De même, les protocoles épidémiques reposant sur un ensemble de nœuds identifiables sont équivalents aux protocoles de communauté. Ceci permet de conclure que l'ensemble des fonctions calculables par les protocoles de la classe PENI correspondent à l'ensemble des fonctions symétriques de $NSPACE(n \log n)$, et peuvent également mettre en œuvre des algorithmes tolérants à des défaillances bénignes ou non (*cf.* chapitre 4, paragraphe 4.3).

L'importance de ces résultats est de pouvoir tirer profit des résultats existants, et à venir, pour chacun des deux domaines. Nous illustrons cette assertion en portant un protocole issu de chaque domaine dans l'autre. Nous montrons également dans le paragraphe suivant qu'une de nos contributions du chapitre 5 permet de conduire à une propriété significative pour les protocoles épidémiques en général.

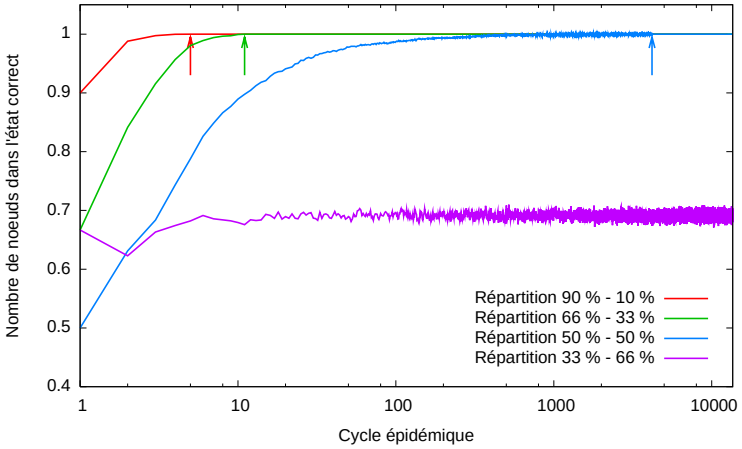

FIG. 6.3 – Évolution du système en fonction de l'avancement du protocole épidémique synchrone de calcul de la majorité.

Calcul épidémique de la majorité

Considérons la primitive de *majorité* introduite au paragraphe 4.2.3 (page 85). À notre connaissance, ce protocole n'a encore jamais été pris en considération dans le contexte épidémique. Nous montrons qu'il possible de concevoir un protocole de PENA permettant de calculer la majorité dans un environnement filaire de manière totalement répartie.

Chaque nœud de système est initialisé à une valeur de $\{\perp_0, \perp_1\}$, et exécute l'algorithme 6.1 paramétrisé comme suit :

- *sélectionPair* retourne un nœud échangeur via le service de PS ;
- *sélectionInfo* retourne l'état courant du nœud $q_i \in \{\perp_0, \perp_1, 0, 1\}$;
- *miseAJour* est construit selon les règles de δ. Étant donnés deux nœuds avec les états q_1 et q_2 et la transition $(q_1, q_2) \mapsto (q'_1, q'_2) \in \delta$: l'état mis à jour du nœud initiateur correspond alors à q'_1, et celui du nœud échangeur à q'_2.

Par la fonction *miseAJour*, la configuration du système évolue de manière identique qu'avec le protocole de population sus-cité. Cet exemple illustre qu'un calcul épidémique de majorité peut être mis en œuvre en pratique.

Ainsi, nous avons simulé une exécution de ce protocole épidémique sous diverses conditions initiales. Nous avons considéré le service de RPS proposé dans [VGvS05], associé à un canal de communication synchrone. La figure 6.3 présente l'évolution du système au cours de l'avancement de ce protocole. Pour chaque fin de cycle épidémique, le nombre de nœuds possédant la valeur correcte attendue est représenté. Quatre répartitions des valeurs initiales ont été utilisées : *(i)* 90 % des nœuds, *(ii)* 66 % des nœuds, *(iii)* 50 % des nœuds, et enfin, *(iv)* 33 % des nœuds avec une valeur d'entrée $\perp_0$ (et *a fortiori* les autres nœuds étant initialisés à $\perp_1$).

Il est possible d'observer qu'uniquement 5 cycles épidémiques sont nécessaires à la convergence de la répartition *i*. Puis, plus la proportion de nœuds initialisés à $\perp_0$ diminue, plus la convergence nécessite un grand nombre de cycles (11 pour la répartition ii et 4185 pour la répartition iii). Enfin, dans le cas où il y a une majorité stricte de nœuds initialisés à $\perp_1$, le

protocole requiert un temps de convergence très important. Cet effet est dû à la quatrième règle de δ (*i.e.* $(0,1) \rightarrow (0,0)$), qui aide à la convergence dans les trois premiers cas, mais handicape clairement dans le cas *iv*. Ainsi, la convergence sera atteinte pour un ordonnancement du PS entraînant une utilisation de cette règle aussi peu que possible. Comme nous utilisons un échantillonnage aléatoire, ce type d'ordonnancement se produira irrémédiablement, mais potentiellement après un nombre de cycles très grand.

À propos de la convergence d'un algorithme épidémique de découpage du réseau

À l'inverse, nous modélisons un algorithme de *découpage en tranches* du réseau (*slicing* pour les anglo-saxons) par un protocole de population. Ceci permet de prouver formellement la convergence de cet algorithme.

Une version épidémique de cet algorithme est définie dans [JK06] permettant de classer les nœuds dans un système large-échelle en fonction de la valeur d'un attribut donné. Cet algorithme fonctionne comme suit. Chaque nœud du système est caractérisé par une valeur x, tirée dans un ensemble ordonné $\mathcal{X}$ (*e.g.* cet attribut peut représenter la capacité de stockage ou la bande passante du nœud). Comme la taille de cet ensemble $\mathcal{X}$ est d'un ordre de grandeur plus grand que n, la taille du système, il est pratiquement impossible pour un nœud de déterminer sa position dans le classement des nœuds, ordonné par la valeur de cet attribut. Afin d'obtenir une estimation de cette position, chaque nœud génère un nombre aléatoire *uniforme* r à partir d'un intervalle préalablement fixé. Par la suite, l'ensemble de ces valeurs aléatoires est trié *selon l'ordre des valeurs de l'attribut spécifique* via un protocole épidémique présenté ci-après. Ainsi, ces valeurs aléatoires étant tirées uniformément dans un intervalle donné, les nœuds peuvent estimer leur position dans le système d'attributs à partir de cette valeur. Irrémédiablement, l'ordre des valeurs aléatoires correspond à celui des valeurs de l'attribut. Ainsi, un nœud, qui obtient une valeur aléatoire supérieure à $0,9$ pour l'intervalle $[0;1[$ donné, peut en déduire qu'il fait partie des 10 % des meilleurs nœuds du réseau, par rapport à l'attribut considéré. L'algorithme épidémique est le suivant : à chaque échange, s'il y a une disparité entre l'ordre des valeurs de l'attribut et l'ordre des valeurs aléatoires, les deux nœuds échangent leur valeur aléatoire. Les fonctions du protocole correspondant sont les suivantes :

- *sélectionPair* retourne un nœud échangeur via le service de PS ;
- *sélectionInfo* retourne la valeur d'attribut et la valeur aléatoire du nœud local : (x_p, r_p) ;
- *miseAJour* met à jour l'état du nœud local si $(x_p - x_q) \cdot (r_p - r_q) < 0$ (*i.e.* les valeurs aléatoires ne sont pas bien ordonnées par rapport aux valeurs d'attribut). Dans ce cas, l'état du nœud local devient (x_p, r_q) et celui du nœud distant q devient (x_q, r_p).

Ce protocole fait partie de la classe PENA et peut donc être modélisé par le protocole de population suivant. Soit la fonction de transition $\delta : (\mathcal{X} \times \mathcal{R})^2 \rightarrow (\mathcal{X} \times \mathcal{R})^2$ tel que :

$$((x_1, r_1), (x_2, r_2)) \mapsto \begin{cases} ((x_1, \min(r_1, r_2)), (x_2, \max(r_1, r_2))) & \text{si } x_1 < x_2 \\ ((x_1, \max(r_1, r_2)), (x_2, \min(r_1, r_2))) & \text{sinon.} \end{cases}$$

Cette modélisation permet d'établir aisément et formellement, d'une autre manière, la convergence irrémédiable de ce protocole épidémique de découpage, présenté initialement dans [JK06]. En outre, la version préliminaire de ce protocole épidémique exploite la vue de voisinage des nœuds afin de maximiser la probabilité d'interagir avec un nœud pertinent

à l'échange de valeur, et ainsi augmenter la vitesse de convergence par un protocole de PENI. Cette version pourrait être modélisée par un protocole de communauté.

Par cette observation, nous pouvons constater que l'utilisation d'identifiants dans la conception de protocoles (*i.e.* le passage d'un protocole de population à un protocole de communauté par exemple) peut avoir un impact substantiel. En effet, bien que la prise en compte des identifiants sur les protocoles de PENA n'accroît pas la puissance de ceux-ci, il permet cependant d'améliorer la vitesse de convergence de ceux-ci, à l'image des résultats présentés dans le chapitre 5.

6.5 Optimalité du RPS en terme de vitesse de convergence

Nous avons montré dans le paragraphe 5.4 du chapitre 5, que pour tout protocole de population et de communauté, il est impossible d'avoir une vitesse de convergence moyenne meilleure que celle obtenue par une DSP uniforme.

Afin de poursuivre dans l'esprit de ce chapitre, nous mettons à profit ce résultat afin de tirer des conclusions dans le contexte des protocoles épidémiques. Ainsi, il est montré qu'une distribution uniforme des interactions conduit au meilleur temps de convergence dans le cas moyen d'un protocole de population. Donc, afin d'obtenir le temps de convergence optimal en moyenne pour un protocole épidémique de PENA, le service de PS doit mettre en œuvre une distribution aléatoire uniforme.

Dans la littérature, il est communément accepté qu'il existe un service épidémique d'échantillonnage aléatoire de nœuds (*RPS* pour *Random Peer Sampling*) fournissant un ordonnancement aléatoire d'interactions [JVG+07, VGvS05]. Il a notamment été montré par des simulations et évaluations intensives dans [JVG+07] que la topologie de graphe aléatoire pouvait être obtenue par l'utilisation d'un protocole épidémique de PS. La conclusion de ces travaux affirme que ce protocole génère un échantillonnage aléatoire uniforme du système, de manière totalement décentralisée. Ainsi, ce RPS spécifique peut être utilisé pour simuler un ordonnanceur *uniforme*, et ainsi obtenir l'optimum en terme de temps de convergence moyen.

Quantifier théoriquement le gain obtenu sur les temps de convergence grâce à ce protocole de RPS, ou à une version biaisée (*e.g.* [BKR07]), reste une question ouverte. Cependant, les premiers résultats empiriques confirment ce résultat [BKR07].

6.6 Conclusion

La contribution majeure de ce chapitre a consisté à établir une corrélation entre les protocoles de population (et de communauté) et les protocoles épidémiques. Ce parallèle entre deux domaines, jusqu'à lors explorés indépendamment, ouvrent des perspectives particulièrement attractives.

Pour ce faire, nous avons introduit une classification plus fine des protocoles épidémiques que celle présentée au chapitre 4. Nous avons ainsi pu établir un cadre théorique pour cette classe de protocole, permettant de spécifier formellement ce qui peut, ou ne peut pas, être calculable par un protocole épidémique, en fonction de la nature du service de PS sous-jacent. Ainsi, un protocole de asyncPENA est équivalent à un protocole de population et un protocole

de PENI est équivalent à un protocole de communauté. Par exemple, il est maintenant clairement établit qu'il n'est pas possible de formaliser des concepts tels que la *divisibilité* ou les *nombres premiers* via les protocoles épidémiques à nœuds anonymes.

Alors que le domaine théorique des protocoles de population capture la puissance et la limitation des protocoles épidémiques et de population, l'équivalence de ceux-ci permet de tirer avantage des propriétés obtenues empiriquement sur les protocoles épidémiques, en terme de passage à l'échelle, de mise en pratique et de vitesse de convergence par exemple. Au-delà de l'exploitation de résultats connus, nos contributions permettent également de partager toute nouvelle proposition dans l'un ou l'autre des domaines. À titre d'illustration, nous avons intégré un des résultats majeurs du chapitre 5 dans le contexte des protocoles épidémiques. Celui-ci nous a permis de conclure que le RPS [JVG+07] correspond en réalité au service optimal d'échantillonnage de nœuds pour les calculs épidémiques.

Conclusion

Contexte d'étude

L'évolution des systèmes informatiques a connu ces dernières années de profondes mutations. Encore mono-poste et mono-utilisateur il y a un quart de siècle, les réseaux de machines ont embrassé un large succès, fortifié par l'apparition de l'Internet. La multiplication des liens entre machines, travaillant toutes à l'achèvement d'une même application, a permis aux systèmes dits répartis de voir le jour. Dès lors, ces derniers n'ont eu de cesse de s'accroître, tant par leur taille que par leur dynamisme.

En parallèle, les progrès dans la miniaturisation des éléments électroniques utilisés dans le contexte informatique ont également été considérables durant la dernière décennie. Ceux-ci ont donné lieu à l'existence de micro-ordinateurs de la taille d'une pièce de monnaie. Ces entités miniatures, appelés communément capteurs dans la littérature, ont l'avantage de leur taille réduite, de leur faible coût et de leur forte adaptabilité à des problèmes jusqu'à lors insolubles.

L'adjonction de ces deux évolutions, préalablement dissociées, a donné naissance à des systèmes répartis sur des capteurs miniatures, autrement nommés *réseaux de capteurs*. Le large spectre des applications rendues possibles par l'utilisation de ces réseaux en a fait un domaine en pleine expansion. Usuellement équipés d'une interface de communication *sans-fil*, les rendant plus modulables aux contraintes de conceptions de ces systèmes, ces capteurs restent cependant limités en terme de ressources disponibles (*e.g.* mémoire, processeur, batterie, *etc.*) en raison de leur petite taille. Ainsi, la mise en œuvre réelle d'un RCsF est souvent optimisée pour une application donnée, permettant de prendre en compte les paramètres matériels, et ainsi de trouver un compromis entre l'efficacité, la fiabilité de l'application et surtout la consommation d'énergie nécessaire à celle-ci. L'augmentation de la durée de vie d'un RCsF reste une caractéristique majeure de conception.

Parmi l'ensemble de ces applications, deux grandes classes de réseaux émergent, à la fois indépendantes mais d'importance similaire : celles des RCsF statiques et mobiles. Dans la

première classe, les capteurs sont fixés manuellement ou placés aléatoirement dans un environnement spécifique. Le statisme des nœuds permet de concevoir des algorithmes de routages efficaces ainsi que des systèmes aux structures logiques complexes. À l'inverse, dans la classe des RCsF mobiles, la topologie physique du réseau est en constante évolution. Les applications déployées dans ce contexte doivent donc prendre en compte le dynamisme du réseau, et avoir une forte réactivité et adaptabilité.

Ces deux classes divisent donc le domaine des RCsF en deux parties distinctes. Cependant, malgré une différence marquée, ces réseaux possèdent intrinsèquement des caractéristiques et des objectifs communs. Par exemple, tous les RCsF doivent pouvoir faire face à des conditions environnementales difficiles et ainsi tolérer les défaillances usuelles de nœuds et de communication. De même, le fonctionnement de ces systèmes nécessitent un déploiement à *large échelle* et sans surveillance humaine. Ainsi, à l'instar de la revendication de cette thèse, la capacité d'*auto-organisation* du système est un objectif essentiel à toute conception de RCsF.

Ainsi, au vu des faibles ressources des nœuds du système, du grand nombre d'entités dans celui-ci et de la capacité d'auto-organisation de ceux-ci, nous arguons dans ce manuscrit que tout système déployé sur un RCsF doit être conçu selon une approche *collaborative*. La philosophie de cette démarche consiste en l'émergence d'un comportement global du système uniquement à partir d'interactions locales des entités du système. À l'instar des réseaux pair-à-pair, désormais célèbres dans le contexte des réseaux filaires, nous défendons la nécessité du déploiement de *systèmes d'informations auto-organisants et collaboratifs pour les réseaux de capteurs à large échelle*.

Dans cette optique, nous avons présenté diverses contributions, scindées selon les deux grandes classes de RCsF. Au sein de chacune des parties, la première portant sur les RCsF statiques et la seconde sur ceux mobiles, nous avons suivi une démarche allant *de la théorie à la pratique* et *du particulier au général*. Nous résumons l'ensemble de ces contributions ci-après.

Contributions

Les contributions de cette thèse, malgré le découpage par famille de RCsF, partagent les mêmes choix de conception présentés ci-avant, à savoir l'auto-organisation selon une approche collaborative. Ainsi, dans le cadre des réseaux de capteurs sans-fil statiques, nous avons proposé :

- l'analyse théorique du problème usuel de suivi de trajectoire d'objets non identifiables dans un espace donné (*i.e.* MOTI). Après avoir montré que ce problème était insoluble dans le cadre général, nous avons proposé différentes restrictions du modèle, ou du système, permettant d'identifier les limites de solvabilité de MOTI. Ainsi, toujours dans une approche collaborative large échelle, nous avons proposé plusieurs algorithmes totalement décentralisés permettant de résoudre MOTI sous différentes contraintes.
- la structure logique multicouches SOLIST, laquelle permet de mettre en œuvre l'intégralité des primitives de la collection **-cast*, tout en limitant la consommation d'énergie du RCsF. Entièrement décentralisé, SOLIST a de plus l'avantage d'être générique en tant que support à une grande majorité d'applications sur les RCsF. Ce système est évalué par

de nombreuses simulations de RCsF de grande taille, hétérogènes et chaotiques, lesquels représentent le pire des cas en terme d'efficacité.

D'autre part, nous avons présenté nos travaux dans le contexte des RCsF mobiles, lesquels sont constitués de :

- la modélisation théorique des modèles de mobilités s'appliquant aux RCsF. Notre extension MAPP du modèle existant des protocoles de population fournit les fondements théoriques permettant d'analyser d'un point de vue stochastique l'évolution de l'état d'un système. Outre une exploration empirique de l'impact de la mobilité et des caractéristiques du système sur la vitesse de convergence d'un algorithme, nous avons montré que quels que soit le système et le protocole envisagés, une répartition uniforme des rencontres entre les différents agents d'une population obtiendra toujours la vitesse moyenne de convergence optimale. Par analogie avec célèbre modèle de mobilité de points de navigations aléatoires (RWP), nous avons controversé la pertinence de ce modèle en tant que référentiel représentatif du comportement d'un protocole sur tout RCsF mobile.
- l'analogie entre les protocoles de population (et ceux de communauté) avec les protocoles épidémiques. Nous avons proposé une première classification des protocoles épidémiques, déterminée par la puissance du service d'échantillonnage des nœuds (PS) et le synchronisme du canal de communication. Celle-ci nous a permis de rapprocher les travaux effectués dans le cadre théorique des RCsF mobiles avec ces protocoles. Ainsi, nous avons établi une relation d'équivalence entre les protocoles de population et les protocoles épidémiques à nœuds anonymes, sur un canal de communication asynchrone (asyncPENA). De même, nous avons montré l'équivalence entre les protocoles de communauté et les protocoles épidémiques sur des nœuds identifiables, quelque soit le synchronisme du canal de communication (PENI). L'ouverture de ces résultats permet d'ouvrir de nouveaux horizons dans chacun de ces domaines. Par exemple, de par les résultat de MAPP, nous pouvons établir que le PS aléatoire proposé dans [JVG+07] représente le service optimal pour les calculs épidémiques.

Pour conclure ce manuscrit, nous proposons diverses perspectives de recherche pour chacune des contributions développées précédemment, permettant ainsi de mettre en évidence les questions restées ouvertes au sein des domaines abordés dans cette thèse.

Perspectives

Dans cette ultime section, nous développons et complétons les perspectives introduites dans les conclusions intermédiaires, dédiées à chaque chapitre. Dans un souci de clarté, nous traitons les perspectives de chacun des domaines abordés dans leur ordre d'apparition au sein de ce document.

Le problème MOTI

Dans le chapitre 2, nous avons focalisé notre étude sur une résolution déterministe du problème. Ainsi, nous avons proposé des caractérisations permettant de résoudre le problème dans

tous les cas. Cependant, il peut advenir qu'un déploiement s'avère impossible sous ces conditions. Ainsi, l'évolution de nos résultats dans un contexte probabiliste est indéniablement une voie de recherche à prendre en considération.

Pour cela, étant données les caractéristiques décentralisées des protocoles que nous avons proposés, il est possible d'imaginer un algorithme probabiliste permettant de prendre une décision en cas de mouvement faillible. Celle-ci peut être prise en connaissance des observations effectuées par les nœuds voisins. À cet effet, un algorithme épidémique pourrait être envisagé pour propager régulièrement les informations à son voisinage direct, par adjonction dans les messages périodiques intrinsèques aux RCsF, à l'instar de la méthodologie utilisée dans [BBFK07b].

La structure SOLIST

Diverses perspectives peuvent être envisagées pour poursuivre les travaux dans le contexte de SOLIST. D'une part, sans revenir sur la conception de la contribution, il serait intéressant d'établir une analyse théorique complète de la structure et des algorithmes de diffusion utilisés pour la collection **-cast*. Par exemple, une preuve formelle de l'unicité de la délivrance d'un message dans le cadre de l'algorithme de *broadcast* pourrait être édifiée. De manière plus fonctionnelle, une analyse de la convergence en terme de correction et de vitesse pourrait être menée. Dans la même veine, nous pourrions évaluer l'impact du ratio entre le gain de messages non retransmis par l'utilisation d'algorithmes non redondants et la probabilité de défaillance d'un capteur.

D'autre part, un recul sur la conception de notre structure nous permettrait peut-être de mettre en œuvre un système équivalent mais plus tolérant aux défaillances massives, ou aux partitionnements du réseau (temporaires ou non). Par la même occasion, mieux que des simulations (aussi exhaustives qu'elles soient), des évaluations en environnement réel seraient à concevoir. Nous envisageons d'ors et déjà mettre en œuvre la version actuelle de SOLIST sur le réseau de capteurs large échelle national SensLAB [Sen08].

Le modèle MAPP

Concernant la modélisation de la mobilité des agents dans le contexte des protocoles de population, les questions ouvertes restent nombreuses, à l'instar de chaque proposition d'un nouveau modèle. Dans ce manuscrit, nous avons développé uniquement des analyses concernant le temps de stabilisation des protocoles de population (et de communauté). Cependant, ce modèle, additionné à la méthode de Monte-Carlo pour les chaînes de Markov, peut permettre de conjecturer bien plus de propriétés, au regard des différents modèles de mobilité. Entre autres, la théorie ergodique peut supporter des analyses approfondies de l'évolution d'un système au cours des exécutions.

Il est également possible d'envisager l'étude comportementale de protocoles plus complexes en présence de différents modèles de mobilité, et ainsi d'améliorer les performances de ceux-ci en présence d'une mobilité imposée (*e.g.* un réseau de capteurs déployé sur des personnes vivant en milieu urbain, ou sur des manchots empereurs en Arctique).

L'équivalence entre protocoles épidémiques et protocoles de population

Pour terminer, les perspectives envisagées à la suite des travaux présentés au chapitre 6 sont nombreuses. En effet, le rapprochement entre les deux communautés de recherche, disjointes à l'origine, ouvre de nouveaux horizons attrayants pour l'avenir.

En premier lieu, nous continuerons d'explorer les différentes classes de protocoles de population existantes, ou à venir, afin d'affiner encore notre classification des protocoles épidémiques.

Sur un autre aspect, la quantification formelle de la vitesse de convergence des algorithmes épidémiques reste un problème ouvert. Il est possible d'envisager l'utilisation de MAPP afin de tirer profit de l'équivalence, et ainsi, d'obtenir des résultats probants dans ce contexte.

Enfin, il existe plusieurs versions de RPS (échantillonnage de nœuds aléatoire). La grande majorité des analyses de ces RPS ont été conduites de manière empirique [BKR07, JVG+07, VGvS05]. Une des perspectives de ce travail est d'analyser formellement les séquences d'interactions issues de chacun de ces RPS, dans une optique de comparaison puis de recherche du RPS optimal en terme d'uniformité de la distribution des interactions.

Bibliographie

[AAD+06] Dana ANGLUIN, James ASPNES, Zoë DIAMADI, Michael J. FISCHER et René PERALTA, Computation in networks of passively mobile finite-state sensors. *Springer Distributed Computing (Special Issue : PODC'04)*, 18(4):235–253, mars 2006.

[AAE06a] Dana ANGLUIN, James ASPNES et David EISENSTAT, Fast computation by population protocols with a leader. Dans *20th International Symposium on Distributed Computing (DISC '06)*, pages 61–75, Stockholm, Suède, septembre 2006.

[AAE06b] Dana ANGLUIN, James ASPNES et David EISENSTAT, Stably computable predicates are semilinear. Dans *25th annual ACM symposium on Principles of distributed computing (PODC '06)*, pages 292–299, Denver, CO, USA, juillet 2006.

[AAE07] Dana ANGLUIN, James ASPNES et David EISENSTAT, A simple protocol for fast robust approximate majority. Dans *21th International Symposium on Distributed Computing (DISC '07)*, pages 20–32, Chypre, novembre 2007.

[AAER07] Dana ANGLUIN, James ASPNES, David EISENSTAT et Eric RUPPERT, The computational power of population protocols. Dans *21th International Symposium on Distributed Computing (DISC '07)*, pages 279–304, Chypre, novembre 2007.

[AAFJ05] Dana ANGLUIN, James ASPNES, Michael J. FISCHER et Hong JIANG, Self-stabilizing population protocols. Dans *9th International Conference Principles of Distributed Systems (OPODIS '05)*, pages 103–117, Pisa, Italy, décembre 2005.

[ABC+03] Javed ASLAM, Zack BUTLER, Florin CONSTANTIN, Valentino CRESPI, George CYBENKO et Daniela RUS, Tracking a moving object with a binary sensor network. Dans *1st International Conference on Embedded networked sensor systems (SenSys '03)*, pages 150–161, Los Angeles, CA, USA, novembre 2003.

[AGY04] James ASPNES, David GOLDENGERG et Yang Richard YANG, On the computational complexity of sensor network localization. Dans *1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensor '04)*, pages 32–44, Turku, Finlande, juillet 2004.

[AK04] Ian F. AKYILDIZ et Ismail H. KASIMOGLU, Wireless sensor and actor networks : Research challenges. *Elsevier Ad Hoc Network Journal*, 2(4):351–367, octobre 2004.

[AKK04] Jamal N. AL-KARAKI et Ahmed E. KAMAL, Routing techniques in wireless sensor networks : a survey. *IEEE Wireless Communications*, 11(6):6–28, décembre 2004.

[AR07] James ASPNES et Eric RUPPERT, An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science, Distributed Computing Column*, 93:98–117, octobre 2007.

[ARK+05] Filipe ARAÚJO, Luís RODRIGUES, Jörg KAISER, Changling LIU et Carlos MITIDIERI, Chr : A distributed hash table for wireless ad hoc networks. Dans *Workshops of the 25th International Conference on Distributed Computing Systems (ICDCS '05 Workshops)*, pages 407–413, Columbus, OH, USA, juin 2005.

[ASSC02a] Ian F. AKYILDIZ, Weilian SU, Yogesh SANKARASUBRAMANIAM et Erdal CAYIRCI, A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, août 2002.

[ASSC02b] Ian F. AKYILDIZ, Weilian SU, Yogesh SANKARASUBRAMANIAM et Erdal CAYIRCI, Wireless sensor networks : a survey. *Elsevier Computer Networks Journal*, 38(4):393–422, mars 2002.

[BA99] Albert-László BARABÁSI et Réka ALBERT, Emergence of scaling in random networks. *Science*, 286:509–512, décembre 1999.

[BBC+05] Roberto BALDONI, Roberto BERALDI, Gianpaolo CUGOLA, Matteo MIGLIAVACCA et Leonardo QUERZONI, Structure-less content-based routing in mobile ad hoc networks. Dans *2nd International Conference of Pervasive Services (ICPS '05)*, pages 37–46, Ile Santorin, Grèce, décembre 2005.

[BBFK07a] Yann BUSNEL, Marin BERTIER, Eric FLEURY et Anne-Marie KERMARREC, GCP : Gossip-based code propagation for large-scale mobile wireless sensor network. Dans *2nd Conference of the European Chapter of ACM SIGOPS (EuroSys '07)*, volume Poster Session, Lisbonne, Portugal, mars 2007.

[BBFK07b] Yann BUSNEL, Marin BERTIER, Eric FLEURY et Anne-Marie KERMARREC, GCP : Gossip-based code propagation for large-scale mobile wireless sensor networks. Dans *1st International Conference on Autonomic Computing and Communication Systems (Autonomics '07)*, page 11, Rome, Italie, octobre 2007.

[BBFK07c] Yann BUSNEL, Marin BERTIER, Eric FLEURY et Anne-Marie KERMARREC, GCP : Gossip-based code propagation for large-scale mobile wireless sensor networks. Rapport de Recherche RR-6251, Institut National de Recherche en Informatique et Application (INRIA), Rennes, France, juillet 2007.

[BBK07] Yann BUSNEL, Marin BERTIER et Anne-Marie KERMARREC, Solist : A lightweight multi-overlay structure for wireless sensor networks. Rapport de Recherche RR-6404, Institut National de Recherche en Informatique et Application (INRIA), Rennes, France, décembre 2007.

[BBK08a] Yann BUSNEL, Marin BERTIER et Anne-Marie KERMARREC, Bridging the gap between population and gossip-based protocols. Rapport de Recherche RR-6720, Institut National de Recherche en Informatique et Application (INRIA), Rennes, France, novembre 2008.

[BBK08b] Yann BUSNEL, Marin BERTIER et Anne-Marie KERMARREC, On the impact of the mobility on convergence speed of population protocols. Rapport de Recherche RR-6580, Institut National de Recherche en Informatique et Application (INRIA), Rennes, France, juillet 2008.

[BBK08c] Yann BUSNEL, Marin BERTIER et Anne-Marie KERMARREC, Solist : Structure multi-couche pair-a-pair à faible consommation pour les réseaux de capteurs sans-fil. Dans *6ème Conférence Française en Systèmes d'Exploitation (CFSE '6)*, Fribourg, Suisse, février 2008.

[BBK08d] Yann BUSNEL, Marin BERTIER et Anne-Marie KERMARREC, Solist or how to look for a needle in a haystack ? Dans *4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '08)*, Avignon, France, octobre 2008.

[BBT08] Marin BERTIER, Yann BUSNEL et Gilles TREDAN, SeNSim (sensor network simulator) : `http://sensim.gforge.inria.fr/`. par Équipe-Projet ASAP, INRIA Rennes – Bretagne Atlantique, France, 2005–2008.

[Ber04] Bernd A. BERG, *Markov Chain Monte Carlo Simulations and Their Statistical Analysis*. World Scientific, octobre 2004.

[BG83] Philip A. BERNSTEIN et Nathan GOODMAN, Multiversion concurrency control-theory and algorithms. *ACM Transactions on Database Systems*, 8(4):465–483, avril 1983.

[BGS01] Philippe BONNET, Johannes GEHRKE et Praveen SESHADRI, Towards sensor database systems. Dans *2nd International Conference on Mobile Data Management (MDM '01)*, pages 3–14, Hong Kong, janvier 2001.

[BIL+07] Domenico BLOISI, Luca IOCCHI, G. R. LEONE, R. PIGLIACAMPO, L. TOMBOLINI et Lorena NOVELLI, A distributed vision system for boat traffic monitoring in the venice grand canal. Dans *2nd International Conference on Computer Vision Theory and Applications (VISAPP '07)*, pages 549–556, Barcelone, Espagne, mars 2007.

[BIM07] Roberto BALDONI, Kleoni IOANNIDOU et Alessia MILANI, Mobility versus the cost of geocasting in mobile ad-hoc networks. Dans *21th International Symposium on Distributed Computing (DISC '07)*, pages 48–62, Chypre, novembre 2007.

[Bit08] BITTORRENT, `http://www.bittorrent.com/`. par BitTorrent, Inc., 2001–2008.

[BJM06] Joffroy BEAUQUIER, Colette JOHNEN et Stéphane MESSIKA, Brief announcement : Computing automatically the stabilization time against the worst and the best schedulers. Dans *20th International Symposium on Distributed Computing (DISC '06)*, pages 443–447, Stockholm, Suède, septembre 2006.

[BK05] Yann BUSNEL et Anne-Marie KERMARREC, Integrating file popularity and peer generosity in proximity measure for semantic-based overlays. Rapport de Recherche RR-5731, Institut National de Recherche en Informatique et Application (INRIA), Rennes, France, octobre 2005.

[BK06] Yann BUSNEL et Anne-Marie KERMARREC, ProxSem : Mesure de proximité sémantique pour les systèmes de partage de fichiers pair-à-pair. Dans *5ème Conférence Française en Systèmes d'Exploitation (CFSE '5)*, pages 37–48, Perpignan, France, octobre 2006.

[BK07] Yann BUSNEL et Anne-Marie KERMARREC, ProxSem : Interest-based proximity measure to improve search efficiency in p2p systems. Dans *4th European Conference on Universal Multiservice Networks (ECUMN '07)*, pages 62–71, Toulouse, France, février 2007.

[BKR07] François BONNET, Anne-Marie KERMARREC et Michel RAYNAL, Small-world networks : From theoretical bounds to practical systems. Dans *11th International Confe-*

rence Principles of Distributed Systems (OPODIS '07), pages 372–385, Guadeloupe Française, décembre 2007.

[BQB+08a] Yann BUSNEL, Leonardo QUERZONI, Roberto BALDONI, Marin BERTIER et Anne-Marie KERMARREC, On the deterministic tracking of moving objects with a binary sensor network. Dans *4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '08)*, pages 46–59, Ile Santorin, Grèce, juin 2008.

[BQB+08b] Yann BUSNEL, Leonardo QUERZONI, Roberto BALDONI, Marin BERTIER et Anne-Marie KERMARREC, Suivi et identification de trajectoires sur un réseau de capteurs binaires. Dans *10ème Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (AlgoTel '08)*, Saint-Malo, France, mai 2008.

[BR03] Charles BLAKE et Rodrigo RODRIGUES, High availability, scalable storage, dynamic peer networks : Pick two. Dans *9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX)*, pages 1–6, Lihue, Hawaii, mai 2003.

[Bre00] Eric A. BREWER, Towards robust distributed systems (invited talk). Dans *19th annual ACM symposium on Principles of distributed computing (PODC '00)*, Portland, OR, USA, juillet 2000.

[BRS03] Christian BETTSTETTER, Giovanni RESTA et Paolo SANTI, The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, mars 2003.

[BS06] Stephen BROWN et Cormac J. SREENAN, A new model for updating software in wireless sensor networks. *IEEE Network*, 20(6):42–47, novembre-décembre 2006.

[Bus05a] Yann BUSNEL, Prise en compte d'aspects sémantiques dans la contruction d'un réseau pair-à-pair. Dans *3ème MAnifestation des JEunes Chercheur francophones dans les domaines des STIC (MajecSTIC '05)*, pages 435–438, Rennes, France, novembre 2005.

[Bus05b] Yann BUSNEL, Prise en compte de la proximité sémantique dans la restructuration de réseau logique pair à pair. Mémoire de master, Ecole Normale Supérieure de Cachan - Antenne de Bretagne / Université de Rennes 1, Rennes, France, juin 2005.

[Bus06a] Yann BUSNEL, ProxSem : Interest-based proximity measure to improve search efficiency in p2p systems. Dans *1st Conference of the European Chapter of ACM SIGOPS (EuroSys '06)*, volume Poster Session, Leuven, Belgique, avril 2006.

[Bus06b] Yann BUSNEL, Système d'information pair-à-pair pour les réseaux de capteurs larges échelles. *La lettre de la Fondation Michel Métivier*, 5:3–5, octobre 2006.

[Bus07] Yann BUSNEL, Le modèle de pair à pair profite aux réseaux de capteurs très étendus. *Interstices : Découvrir la recherche en informatique*, page 22096, mai 2007.

[BV05] Jean-Yves Le BOUDEC et Milan VOJNOVI, Perfect simulation and stationarity of a class of mobility models. Dans *24th IEEE Conference on Computer Communications (InfoCom '05)*, pages 2743–2754, Miami, FL, USA, mars 2005.

[CDHR03a] Miguel CASTRO, Peter DRUSCHEL, Y. Charlie HU et Antony ROWSTRON, Proximity neighbor selection in tree-based structured peer-to-peer overlays. Rapport de Recherche MSR-TR-2003-52, Microsoft Research, Cambridge, Angleterre, septembre 2003.

[CDHR03b] Miguel CASTRO, Peter DRUSCHEL, Y. Charlie HU et Antony I. T. ROWSTRON, *Future Directions in Distributed Computing*, volume 2584, chapitre Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks., pages 103–107. Springer, 2003.

[CDT00] Mainak CHATTERJEE, Sajal K. DAS et Damla TURGUT, A weight based distributed clustering algorithm for mobile ad hoc networks. Dans *7th International Conference on High Performance Computing (HiPC '00)*, pages 511–521, Bangalore, Inde, décembre 2000.

[CES04] David CULLER, Deborah ESTRIN et Mani B. SRIVASTAVA, Guest editors' introduction : Overview of sensor networks. *IEEE Computer*, 37(8):41–49, août 2004.

[CGSR05] Jean CARLE, Antoine GALLAIS et D. SIMPLOT-RYL, Preserving area coverage in wireless sensor networks based on surface coverage relay dominating sets. Dans *10th IEEE Symposium on Computers and Communications (ISCC '05)*, pages 347–352, Cartage, Espagne, juin 2005.

[CHC+07] Augustin CHAINTREAU, Pan HUI, Jon CROWCROFT, Christophe DIOT, Richard GASS et James SCOTT, Impact of human mobility on the design of opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, juin 2007.

[CJK+03] Miguel CASTRO, Michael B. JONES, Anne-Marie KERMARREC, Antony ROWSTRON, Marvin THEIMER, Helen WANG et Alec WOLMAN, An evaluation of scalable application-level multicast built using peer-to-peer overlays. Dans *22nd IEEE Conference on Computer Communications (InfoCom '03)*, pages 1510–1520, San Francisco, CA, USA, avril 2003.

[CLF07] Vania CONAN, Jérémie LEGUAY et Timur FRIEDMAN, Characterizing pairwise intercontact patterns in delay tolerant networks. Dans *1st International Conference on Autonomic Computing and Communication Systems (Autonomics '07)*, page 19, Rome, Italy, octobre 2007.

[Cor08] CORNELL DATABASE GROUP, `http://www.cs.cornell.edu/database/cougar/`. par Équipe Cougar, 2000–2008.

[CRB01] Ranveer CHANDRA, Venugopalan RAMASUBRAMANIAN et Ken BIRMAN, Anonymous gossip : Improving multicast reliability in ad-hoc networks. Dans *21th International Conference on Distributed Computing Systems (ICDCS '01)*, pages 275–283, Phoenix, Arizona, avril 2001.

[CRZ04] Maurice CHU, James REICH et Feng ZHAO, Distributed attention for large video sensor networks. Dans *Intelligent Distributed Surveilliance Systems*, pages 61–65, Palo Alto, CA, USA, février 2004.

[DDC97] Christophe DIOT, Walid DABBOUS et Jon CROWCROFT, Multipoint communication : A survey of protocols, functions and mechanisms. *IEEE Journal on Selected Area in Communication*, 15(3):277–290, avril 1997.

[DFKS07] Vadim DRABKIN, Roy FRIEDMAN, Gabriel KLIOT et Marc SEGAL, RAPID : Reliable probabilistic dissemination in wireless ad-hoc networks. Dans *26th IEEE International Symposium on Reliable Distributed Systems (SRDS '07)*, pages 13–22, Los Alamitos, CA, USA, octobre 2007.

[DGFGR06] Carole DELPORTE-GALLET, Hugues FAUCONNIER, Rachid GUERRAOUI et Eric RUPPERT, When birds die : Making population protocols fault-tolerant. Dans *2nd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '06)*, pages 51–66, San Francisco, CA, USA, juin 2006.

[DGH+87] Alan DEMERS, Dan GREENE, Carl HAUSER, Wes IRISH, John LARSON, Scott SHENKER, Howard STURGIS, Dan SWINEHART et Doug TERRY, Epidemic algorithms for replicated database maintenance. Dans *6th annual ACM symposium on*

Principles of distributed computing (PODC '87), pages 1–12, Vancouver, BC, Canada, août 1987.

[DHZ+01] Chris H. Q. DING, Xiaofeng HE, Hongyuan ZHA, Ming GU et Horst D. SIMON, A min-max cut algorithm for graph partitioning and data clustering. Dans *1st IEEE International Conference on Data Mining (ICDM '01)*, pages 107–114, San Jose, CA, USA, décembre 2001.

[DQA04] Anwitaman DATTA, Silvia QUARTERONI et Karl ABERER, Autonomous gossiping : A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks. Dans *International Conference on Semantics of a Networked World (IC-SNW '04)*, pages 126–143, Paris, France, juin 2004.

[DW04] Fei DAI et Jie WU, Performance analysis of broadcast protocols in ad hoc networks based on self-pruning. *IEEE Transactions on Parallel and Distributed Systems*, 15(11):1027–1040, novembre 2004.

[EE01] Jeremy ELSON et Deborah ESTRIN, Time synchronization for wireless sensor networks. Dans *15th International Parallel and Distributed Processing Symposium (IPDPS '01)*, pages 1965–1970, San Francisco, CA, USA, avril 2001.

[EE04] Jeremy ELSON et Deborah ESTRIN, Sensor networks : a bridge to the physical world. *Kluwer Wireless Networks*, pages 3–20, janvier 2004.

[EGKM04] Patrick T. EUGSTER, Rachid GUERRAOUI, Anne-Marie KERMARREC et Laurent MASSOULIÉ, Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–67, mai 2004.

[EGKM06] Patrick T. EUGSTER, Rachid GUERRAOUI, Anne-Marie KERMARREC et Laurent MASSOULIÉ, From epidemics to distributed computing. *IEEE Computer*, 37(5):60–67, mai 2006.

[EHG+03] Patrick T. EUGSTER, Sidath HANDURUKANDE, Rachid GUERRAOUI, Anne-Marie KERMARREC et Petr KOUZNETSOV, Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, novembre 2003.

[EP05] Nathan EAGLE et Alex (Sandy) PENTLAND, CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http ://crawdad.cs.dartmouth.edu/mit/reality, juillet 2005.

[Fee01] Laura Marie FEENEY, An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. *Kluwer Mobile Networks and Applications*, 6(3):239–249, juin 2001.

[FGJ+07] Antonio FERNÁNDEZ, Vincent GRAMOLI, Ernesto JIMÉNEZ, Anne-Marie KERMARREC et Michel RAYNAL, Distributed slicing in dynamic systems. Dans *27th International Conference on Distributed Computing Systems (ICDCS '07)*, pages 66–75, Toronto, Canada, juin 2007.

[FHKM04] Fabrice Le FESSANT, Sidath HANDURUKANDE, Anne-Marie KERMARREC et Laurent MASSOULIÉ, Clustering in peer-to-peer file sharing workloads. Dans *3rd International Workshop on Peer-to-Peer Systems (IPTPS '04)*, San Diego, CA, USA, février 2004.

[Flo62] Robert W. FLOYD, Algorithm 97 : Shortest path. *Communications of the ACM*, 5(6):345, juin 1962.

[FMP06] Laurent FRIBOURG, Stéphane MESSIKA et Claudine PICARONNY, Coupling and self-stabilization. *Springer Distributed Computing (Special Issue : DISC'04)*, 18(3): 221–232, février 2006.

[FW07] Roland FLURY et Roger WATTENHOFER, Routing, anycast, and multicast for mesh and sensor networks. Dans *26th IEEE Conference on Computer Communications (InfoCom '07)*, pages 946–954, Anchorage, AL, USA, mai 2007.

[GGC03] Abhishek GHOSE, Jens GROSSKLAGS et John CHUANG, Resilient data-centric storage in wireless sensor networks. *IEEE Distributed Systems Online*, novembre 2003.

[GGE+05] Deepak GANESAN, Ben GREENSTEIN, Deborah ESTRIN, John HEIDEMANN et Ramesh GOVINDAN, Multiresolution storage and search in sensor networks. *ACM Transaction on Storage*, 1(3):277–315, août 2005.

[GHH+02] Ramesh GOVINDAN, Joseph M. HELLERSTEIN, Wei HONG, Samuel MADDEN, Michael FRANKLIN et Scott SHENKER, The sensor network as a database. Rapport de Recherche 02-771, Computer Science Department, University of Southern California, Berkeley et Los Angeles, CA, USA, 2002.

[GL02] Seth GILBERT et Nancy LYNCH, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGART News*, 33(2):51–59, juin 2002.

[GM04] Chao GUI et Prasant MOHAPATRA, Power conservation and quality of surveillance in target tracking sensor networks. Dans *10th ACM International Conference on Mobile Computing and Networking (MobiCom '04)*, pages 129–143, Philadelphia, PA, USA, septembre 2004.

[GMS05] Saikat GUHA, Rohan MURTY et Emin Gün SIRER, Sextant : a unified node and event localization framework using non-convex constraints. Dans *6th ACM International Symposium on Mobile ad hoc networking and computing (MobiHoc '05)*, pages 205–216, Urbana-Champaign, IL, USA, mai 2005.

[GPM03] Rajiv GANDHI, Srinivasan PARTHASARATHY et Arunesh MISHRA, Minimizing broadcast latency and redundancy in ad hoc networks. Dans *4th ACM International Symposium on Mobile ad hoc networking and computing (MobiHoc '03)*, pages 222–232, Annapolis, MD, USA, juin 2003.

[GR07] Rachid GUERRAOUI et Eric RUPPERT, Even small birds are unique : Population protocols with identifiers. Rapport de Recherche CSE-2007-04, Department of Computer Science and Engineering, York University, York, ON, Canada, septembre 2007.

[GSAA04] Abhishek GUPTA, Ozgur D. SAHIN, Divyakant AGRAWAL et Amr El ABBADI, Meghdoot : content-based publish/subscribe over P2P networks. Dans *5th ACM/IFIP/USENIX international Middleware conference (Middleware '04)*, pages 254–273, Toronto, Canada, octobre 2004.

[GVvS05] Daniela GAVIDIA, Spyros VOULGARIS et Maarten van STEEN, Epidemic-style monitoring in large-scale wireless sensor networks. Rapport de Recherche IR-CS-012.05, Vrije Universiteit Amsterdam, Amsterdam, Hollande, 2005.

[Har01] Takahiro HARA, Effective replica allocation in ad hoc networks for improving data accessibility. Dans *20th IEEE Conference on Computer Communications (InfoCom '01)*, pages 1568–1576, Anchorage, AL, USA, avril 2001.

[HAR07] HART COMMUNICATION FOUNDATION, WirelessHART – the new industry standard : `http://www.hartcomm2.org/hart_protocol/wireless_hart/wireless_hart_main.html`. *cf.* HART 7 Specification, juin 2007.

[HBWB06] Marc HEISSENBÜTTEL, Torsten BRAUN, Markus WÄLCHLI et Thomas BERNOULLI, Optimized stateless broadcasting in wireless multi-hop networks. Dans *25th IEEE Conference on Computer Communications (InfoCom '06)*, pages 1–12, Barcelone, Espagne, avril 2006.

[HC04] Jonathan W. HUI et David CULLER, The dynamic behavior of a data dissemination protocol for network programming at scale. Dans *2nd International Conference on Embedded networked sensor systems (SenSys '04)*, pages 81–94, Baltimore, MD, USA, novembre 2004.

[HCB00] Wendi Rabiner HEINZELMAN, Anantha CHANDRAKASAN et Hari BALAKRISHNAN, Energy-efficient communication protocol for wireless microsensor networks. Dans *33rd Annual Hawaii International Conference on System Science (HICSS '00)*, pages 1–10, Ile Maui, Hawaii, janvier 2000.

[HCS+05] Pan HUI, Augustin CHAINTREAU, James SCOTT, Richard GASS, Jon CROWCROFT et Christophe DIOT, Pocket switched networks and human mobility in conference environments. Dans *ACM SIGCOMM Workshop on Delay-tolerant networking (WDTN '05)*, pages 244–251, Philadelphia, Pennsylvania, USA, août 2005.

[Her03] Klaus HERRMANN, Modeling the sociological aspects of mobility in ad hoc networks. Dans *6th ACM International Workshop on Modeling analysis and simulation of wireless and mobile systems (MSWiM '03)*, pages 128–129, San Diego, CA, USA, septembre 2003.

[HHL02] Zygmunt J. HAAS, Joseph Y. HALPERN et Li LI, Gossip-based ad hoc routing. Dans *21th IEEE Conference on Computer Communications (InfoCom '02)*, pages 1707–1716, New York, NY, USA, juin 2002.

[HJL+03] Mark HEDLEY, Mark JOHNSON, Chris LEWIS, David CARPENTER, Howard LOVATT et Don PRICE, Smart sensor network for space vehicle monitoring. Dans *International Signal Processing Conference*, Dallas, TX, USA, mars 2003.

[HK92] Lars W. HAGEN et Andrew B. KAHNG, New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 11(9):1074–1085, septembre 1992.

[HKLF+06] Sidath HANDURUKANDE, Anne-Marie KERMARREC, Fabrice LE FESSANT, Laurent MASSOULIÉ et Simon PATARIN, Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. Dans *1st Conference of the European Chapter of ACM SIGOPS (EuroSys '06)*, pages 359–371, Leuven, Belgium, avril 2006.

[HLR03] Qingfeng HUANG, Chenyang LU et Gruia-Catalin ROMAN, Spatiotemporal multicast in sensor networks. Dans *1st International Conference on Embedded networked sensor systems (SenSys '03)*, pages 205–217, Los Angeles, California, USA, novembre 2003.

[HMN04] Takahiro HARA, Norishige MURAKAMI et Shojiro NISHIO, Replica allocation for correlated data items in ad hoc sensor networks. *ACM SIGMOD Record*, 33(1):38–43, mars 2004.

[HXTG04] Mei HAN, Wei XU, Hai TAO et Yihong GONG, An algorithm for multiple object trajectory tracking. Dans *4th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04)*, pages I–864–I–871, Washington, DC, USA, juillet 2004.

[IG99] Tomasz IMIELINSKI et Samir GOEL, Dataspace : querying and monitoring deeply networked collections in physical space. Dans *1st ACM international workshop on Data engineering for wireless and mobile access (MobiDe '99)*, pages 44–51, Seattle, WA, USA, octobre 1999.

[IGE+03] Chalermek INTANAGONWIWAT, Ramesh GOVINDAN, Deborah ESTRIN, John HEIDEMANN et Fabio SILVA, Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions Networks*, 11(1):2–16, février 2003.

[ISR05] Francois INGELREST et David SIMPLOT-RYL, Localized broadcast incremental power protocol for wireless ad hoc networks. Dans *10th IEEE Symposium on Computers and Communications (ISCC '05)*, pages 28–33, Cartage, Espagne, juin 2005.

[JB04] Márk JELASITY et Ozalp BABAOGLU, T-Man : Fast gossip-based construction of large-scale overlay topologies. Rapport de Recherche UBLCS-2004-7, University of Bologna, Department of Computer Science, Bologna, Italy, mai 2004.

[JBRAS03] Amit JARDOSH, Elizabeth M. BELDING-ROYER, Kevin C. ALMEROTH et Subhash SURI, Towards realistic mobility models for mobile ad hoc networks. Dans *9th ACM International Conference on Mobile Computing and Networking (MobiCom '03)*, pages 217–229, San Diego, CA, USA, septembre 2003.

[JGKvS04] Márk JELASITY, Rachid GUERRAOUI, Anne-Marie KERMARREC et Maarten van STEEN, The peer sampling service : experimental evaluation of unstructured gossip-based implementations. Dans *5th ACM/IFIP/USENIX international Middleware conference (Middleware '04)*, pages 79–98, Toronto, Canada, octobre 2004.

[JK06] Mark JELASITY et Anne-Marie KERMARREC, Ordered slicing of very large-scale overlay networks. Dans *6th IEEE International Conference on Peer-to-Peer Computing (P2P '06)*, pages 117–124, Cambridge, Angleterre, septembre 2006.

[JM96] David B. JOHNSON et David A. MALTZ, Dynamic source routing in ad hoc wireless networks. *Kluwer Mobile Computing*, 353:153–181, décembre 1996.

[JMB01] David B. JOHNSON, David A. MALTZ et Josh BROCH, *Ad Hoc Networking*, chapitre DSR : The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, pages 139–172. Addison-Wesley, 2001.

[JMB05] Márk JELASITY, Alberto MONTRESOR et Ozalp BABAOGLU, Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, août 2005.

[JVG+07] Márk JELASITY, Spyros VOULGARIS, Rachid GUERRAOUI, Anne-Marie KERMARREC et Maarten van STEEN, Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3):8, août 2007.

[KCG06] Pradeep KYASANUR, Romit Roy CHOUDHURY et Indranil GUPTA, Smart gossip : An adaptive gossip-based broadcasting service for sensor networks. Dans *3rd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS '06)*, Vancouver, Canada, octobre 2006.

[KDG03] David KEMPE, Alin DOBRA et Johannes GEHRKE, Gossip-based computation of aggregate information. Dans *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03)*, pages 482–491, Cambridge, MA, USA, octobre 2003.

[KGSL05] Purushottam KULKARNI, Deepak GANESAN, Prashant SHENOY et Qifeng LU, SensEye : a multi-tier camera sensor network. Dans *13th annual ACM international conference on Multimedia*, pages 229–238, Singapour, AU, novembre 2005.

[KK00] Brad KARP et H. T. KUNG, GPSR : Greedy perimeter stateless routing for wireless networks. Dans *6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 243–254, Boston, MA, USA, août 2000.

[Kle00] Jon KLEINBERG, The small-world phenomenon : An algorithmic perspective. Dans *32nd ACM Symposium on Theory of Computing (STOC '00)*, pages 163–170, Portland, OR, USA, mai 2000.

[KMG03] Anne-Marie KERMARREC, Laurent MASSOULIÉ et Ayalvadi J. GANESH, Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, mars 2003.

[KMS+05] YoungMin KWON, Kirill MECHITOV, Sameer SUNDRESH, Wooyoung KIM et Gul AGHA, Resilient localization for sensor networks in outdoor environments. Dans *25th International Conference on Distributed Computing Systems (ICDCS '05)*, pages 643–652, Columbus, OH, USA, juin 2005.

[KvS07] Anne-Marie KERMARREC et Maarten van STEEN, éditeurs. *ACM Operating Systems Review on Gossip Potocols*, volume 41(5). ACM-SIGOPS, ACM Press, octobre 2007.

[KW05] Sandeep S. KULKARNI et Limin WANG, MNP : multihop network reprogramming service for sensor networks. Dans *25th International Conference on Distributed Computing Systems (ICDCS '05)*, pages 7–16, Columbus, OH, USA, juin 2005.

[LEH04] Jun LUO, Patrick T. EUGSTER et Jean-Pierre HUBAUX, Pilot : Probabilistic lightweight group communication system for ad hoc networks. *IEEE Transactions on Mobile Computing*, 3(2):164–179, avril 2004.

[LH04] Ning LI et Jennifer C. HOU, Blmst : A scalable, power-efficient broadcast algorithm for wireless networks. Dans *1st International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE '04)*, pages 44–51, Dallas, TX, USA, octobre 2004.

[LLR+03] Juan LIU, Jie LIU, James REICH, Patrick CHEUNG et Feng ZHAO, Distributed group management for track initiation and maintenance in target localization applications. Dans *2nd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '03)*, pages 113–128, Palo Alto, CA, USA, avril 2003.

[LPSC04] Philip LEVIS, Neil PATEL, Scott SHENKER et David CULLER, Trickle : A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. Dans *1st USENIX Symposium on Network Systems Design and Implementation (NSDI '04)*, pages 1–14, San Francisco, CA, USA, mars 2004.

[LWF03] Xiang-Yang LI, Yu WANG et Ophir FRIEDER, Localized routing for wireless ad hoc networks. Dans *38th IEEE International Conference on Communications (ICC '03)*, pages 443–447, Anchorage, AL, USA, mai 2003.

[MBF05] Nathalie MITTON, Anthony BUSSON et Eric FLEURY, Broadcasting in self-organizing wireless multi-hop network. Rapport de Recherche RR-5487, Institut National de Recherche en Informatique et Application (INRIA), Lyon, France, février 2005.

[MCP+02] Alan MAINWARING, David CULLER, Joseph POLASTRE, Robert SZEWCZYK et John ANDERSON, Wireless Sensor Networks for Habitat Monitoring. Dans *1st ACM International Workshop on Wireless sensor networks and applications (WSNA '02)*, pages 88–97, Atlanta, GE, USA, septembre 2002.

[MDGS06] G. MATHUR, P. DESNOYERS, D. GANESAN et P. SHENOY, Ultra-low power data storage for sensor networks. Dans *5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '06)*, pages 374–381, Nashville, TE, USA, avril 2006.

[MFHH05] Samuel R. MADDEN, Michael J. FRANKLIN, Joseph M. HELLERSTEIN et Wei HONG, TinyDB : an acquisitional query processing system for sensor networks. *ACM Transactions on Database System*, 30(1):122–173, mars 2005.

[MJB04] Alberto MONTRESOR, Márk JELASITY et Ozalp BABAOGLU, Robust aggregation protocols for large-scale overlay networks. Dans *34th International Conference on Dependable Systems and Networks (DSN '04)*, pages 19–28, Florence, Italie, juin 2004.

[MLRR05] Hugo MIRANDA, Simone LEGGIO, Luís RODRIGUES et Kimmo RAATIKAINEN, A stateless neighbour-aware cooperative caching protocol for ad-hoc networks. DI/FCUL TR 05–23, Department of Informatics, Universidade di Lisboa, Lisbonne, Portugal, décembre 2005.

[MLRR06a] Hugo MIRANDA, Simone LEGGIO, Luís RODRIGUES et Kimmo RAATIKAINEN, An algorithm for distributing and retrieving information in sensor networks. Dans *10th International Conference Principles of Distributed Systems (OPODIS '06)*, pages 31–45, Bordeaux, France, décembre 2006.

[MLRR06b] Hugo MIRANDA, Simone LEGGIO, Luís RODRIGUES et Kimmo RAATIKAINEN, A power-aware broadcasting algorithm. Dans *17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '06)*, pages 1–5, Helsinki, Finlande, septembre 2006.

[MMKG06] Laurent MASSOULIÉ, Erwan Le MERRER, Anne-Marie KERMARREC et Ayalvadi GANESH, Peer counting and sampling in overlay networks : random walk methods. Dans *25th annual ACM symposium on Principles of distributed computing (PODC '06)*, pages 123–132, Denver, CO, USA, juillet 2006.

[MN01] Laura MARIE et Martin NILSSON, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. Dans *20th IEEE Conference on Computer Communications (InfoCom '01)*, pages 1548–1557, Anchorage, AL, USA, avril 2001.

[MRR03] Achour MOSTEFAOUI, S. RAJSBAUM et Michel RAYNAL, Conditions on input vectors for consensus solvability in asynchronous distributed systems. *Journal of the ACM*, 50(6):922–954, juin 2003.

[MU49] N. METROPOLIS et S. ULAM, The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, septembre 1949.

[MVG+06] Erwan Le MERRER, Aline Carneiro VIANA, Vincent GRAMOLI, Anne-Marie KERMARREC et Marin BERTIER, Adaptive topology management scheme for improving network lifetime in WSNs. Dans *International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking (MobiShare '06)*, Los Angeles, CA, USA, septembre 2006.

[Nap08] NAPSTER, `http://free.napster.com/`. par Napster, LLC, depuis 2001. – Version litigieuse disponible sur «Internet Archive» : `http://web.archive.org/web/20001220001300/http://www.napster.com/`, 1999–2008.

[NS03] James NEWSOME et Dawn SONG, GEM : Graph EMbedding for routing and data-centric storage in sensor networks without geographic information. Dans *1st International Conference on Embedded networked sensor systems (SenSys '03)*, pages 76–88, Los Angeles, California, USA, novembre 2003.

[OS05] Songhwai OH et Shankar SASTRY, Tracking on a graph. Dans *4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '05)*, pages 195–202, Los Angeles, CA, USA, avril 2005.

[OW05] Regina O'DELL et Roger WATTENHOFER, Theoretical aspects of connectivity-based multi-hop positioning. *Elsevier Theoretical Computer Science*, 344(1):47–68, novembre 2005.

[PB94] Charles E. PERKINS et Pravin BHAGWAT, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, octobre 1994.

[PBBvR06] Stefan PLEISCH, Mahesh BALAKRISHNAN, Ken BIRMAN et Robbert van RENESSE, MISTRAL : Efficient flooding in mobile ad-hoc networks. Dans *7th ACM International Symposium on Mobile ad hoc networking and computing (MobiHoc '06)*, pages 1–12, Florence, Italie, mai 2006.

[PK00] Gregory J. POTTIE et William J. KAISER, Wireless integrated network sensors. *Communications of the ACM*, 43(5):551–558, mai 2000.

[PR99] Charles E. PERKINS et Elizabeth M. ROYER, Ad-hoc on-demand distance vector routing. Dans *2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pages 90–100, New Orleans, LO, USA, février 1999.

[REG+02] Sylvia RATNASAMY, Deborah ESTRIN, Ramesh GOVINDAN, Brad KARP, Scott SHENKER, Li YIN et Fang YU, Data-centric storage in sensornets. *ACM SIGCOMM Computer Communication Review*, 33(1):137–142, janvier 2002.

[RFH+01] Sylvia RATNASAMY, Paul FRANCIS, Mark HANDLEY, Richard KARP et Scott SCHENKER, A scalable content-addressable network. Dans *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01)*, pages 161–172, San Diego, CA, USA, août 2001.

[RG05] Etienne RIVIÈRE et Philippe GAURON, Rechercher parmi ses pairs ou quand le hasard ne fait pas si bien les choses, tutoriel. Dans *3ème MAnifestation des JEunes Chercheur francophones dans les domaines des STIC (MajecSTIC '05)*, Rennes, France, novembre 2005.

[RHKS01] Sylvia RATNASAMY, Mark HANDLEY, Richard KARP et Scott SHENKER, Application-level multicast using content-addressable networks. Dans *3rd International Workshop on Networked Group Communication (NGC '01)*, pages 14–29, London, UK, novembre 2001.

[Riv07] Etienne RIVIÈRE, *Réseaux logiques collaboratifs pour la recherche décentralisée dans les systèmes à large échelle. Thèse de doctorat*, IFSIC – Université de Rennes 1, Rennes, France, novembre 2007.

[RKS+03] Sylvia RATNASAMY, Brad KARP, Scott SHENKER, Deborah ESTRIN, Ramesh GOVINDAN, Li YIN et Fang YU, Data-centric storage in sensornets with GHT, a geographic hash table. *Kluwer Mobile Networks and Applications*, 8(4):427–442, août 2003.

[RMGS01] Praveen RENTALA, Ravi MUSUNURI, Shashidhar GANDHAM et Udit SAXENA, Survey on sensor networks. Dans *7th ACM International Conference on Mobile Computing and Networking (MobiCom '01)*, Rome, Italie, juillet 2001.

[RS96] S. RAMANATHAN et Martha STEENSTRUP, A survey of routing techniques for mobile communications networks. *Kluwer Mobile Networks and Applications*, 1(2):89–104, juin 1996.

[SCS06] Eric SIMONTON, Byung Kyu CHOI et Steven SEIDEL, Using gossip for dynamic ressource discovery. Dans *35th International Conference on Parallel Processing (ICPP '06)*, pages 319–328, Columbus, OH, USA, août 2006.

[Sen08] SENSLAB, `http://www.senslab.info/`. par SensLAB – Very Large Scale Open Wireless Sensor Network Testbed, 2008.

[SGW01] Loren SCHWIEBERT, Sandeep K.S. GUPTA et Jennifer WEINMANN, Research challenges in wireless networks of biomedical sensors. Dans *7th ACM International Conference on Mobile Computing and Networking (MobiCom '01)*, pages 151–165, Rome, Italie, juillet 2001.

[Sky08] SKYPE, `http://www.skype.com/`. par Skype Technologies SA., 2003–2008.

[SMK+07] Jaspreet SINGH, Upamanyu MADHOW, Rajesh KUMAR, Subhash SURI et Richard CAGLEY, Tracking multiple targets using binary proximity sensors. Dans *6th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '07)*, pages 529–538, Cambridge, MA, USA, avril 2007.

[SMP01] Mani B. SRIVASTAVA, Richard R. MUNTZ et Miodrag POTKONJAK, Smart kindergarten : sensor-based wireless networks for smart developmental problem-solving enviroments. Dans *7th ACM International Conference on Mobile Computing and Networking (MobiCom '01)*, pages 132–138, Rome, Italy, juillet 2001.

[SS03] Philipp STEURER et Mani B. SRIVASTAVA, System design of smart table. Dans *1st IEEE International Conference on Pervasive Computing and Communications (PerCom '03)*, page 473, Dallas, TX, USA, mars 2003.

[SSA06] Sundar SUBRAMANIAN, Sanjay SHAKKOTTAI et Ari ARAPOSTATHIS, Broadcasting in sensor networks : The role of local information. Dans *25th IEEE Conference on Computer Communications (InfoCom '06)*, pages 1–12, Barcelone, Espagne, avril 2006.

[SSJ01] Chien-Chung SHEN, Chavalit SRISATHAPORNPHAT et Chaiporn JAIKAEO, Sensor information networking architecture and applications. *IEEE Personel Communication Magazine*, 8(4):52–59, août 2001.

[SYH04] Samba SESAY, Zongkai YANG et Jianhua HE, A survey on mobile ad hoc wireless network. *Information Technology Journal*, 3(2):168–175, 2004.

[TLLB04] Yuldi TIRTA, Zhiyuan LI, Yung-Hsiang LU et Saurabh BAGCHI, Efficient collection of sensor data in remote fields using mobile collectors. Dans *13th International Conference On Computer Communications and Networks (ICCCN '04)*, pages 515–520, Chicago, IL, USA, octobre 2004.

[TMH03] Sameer TILAK, Amy MURPHY et Wendi HEINZELMAN, Non-uniform information dissemination for sensor networks. Dans *11th IEEE International Conference on Network Protocols (ICNP '03)*, pages 295–304, Atlanta, GE, USA, novembre 2003.

[TNCS02] Yu-Chee TSENG, Sze-Yao NI, Yuh-Shyan CHEN et Jang-Ping SHEU, The broadcast storm problem in a mobile ad hoc network. *Kluwer Wireless Networks*, 8(2/3):153–167, mars–mai 2002.

[TTP+95] Douglas B. TERRY, Marvin M. THEIMER, Karin PETERSEN, Alan J. DEMERS, Mike J. SPREITZER et Carl H. HAUSER, Managing update conflicts in bayou, a weakly connected replicated storage system. Dans *15th ACM Symposium on Operating Systems Principles (SOSP '95)*, pages 172–183, Copper Mountain Resort, CO, USA, décembre 1995.

[TTS05] Niwat THEPVILOJANAPONG, Yoshito TOBE et Kaoru SEZAKI, Har : Hierarchy-based anycast routing protocol for wireless sensor networks. Dans *5th Symposium on Applications and the Internet (SAINT '05)*, pages 204–212, Trento, Italy, février 2005. IEEE Computer Society.

[TYD+04] Niki TRIGONI, Yong YAO, Alan DEMERS, Johannes GEHRKE et Rajmohan RAJARAMAN, WaveScheduling : energy-efficient data dissemination for sensor networks. Dans *1st International Workshop on Data management for sensor networks (DMSN '04)*, pages 48–57, Toronto, Canada, août 2004.

[VDSJCJDM03] Marcos Augusto M. VIEIRA, Diógenes Cecílio DA SILVA JUNIOR, Claudionor N. COEHLO JUNIOR et José M. DA MATA, Survey on wireless sensor network devices.

Dans *9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '03)*, pages 537–544, Lisbonne, Portugal, septembre 2003.

[VE03] Einar VOLLSET et Paul EZHILCHELVAN, A survey of reliable broadcast protocols for mobile ad-hoc networks. Rapport de Recherche CS-TR-792, University of Newcastle, Newcastle, Angeleterre, 2003.

[vEB89] Peter van EMDE BOAS, Space measures for storage modification machines. *Elsevier Information Processing Letters*, 30(2):103–110, janvier 1989.

[vGR03] Jana van GREUNEN et Jan RABAEY, Lightweight time synchronization for sensor networks. Dans *2nd ACM international conference on Wireless sensor networks and applications (WSNA '03)*, pages 11–19, San Diego, CA, USA, septembre 2003.

[VGvS05] Spyros VOULGARIS, Daniela GAVIDIA et Maarten van STEEN, Cyclon : Inexpensive membership management for unstructured p2p overlays. *Journal of Network System Management*, 13(2):197–217, juin 2005.

[vR02] Robbert van RENESSE, Power-aware epidemics. Dans *21th IEEE Symposium on Reliable Distributed Systems (SRDS '02)*, pages 358–361, Suita, Japon, octobre 2002.

[VvRB06] Einar VOLLSET, Robbert van RENESSE et Ken BIRMAN, ChickWeed : Group communication for embedded devices in opportunistic networking environments. Dans *3rd International Workshop on Dependable Embedded Systems (WDES '06)*, pages 49–52, Leeds, Angleterre, octobre 2006.

[War62] Stephen WARSHALL, A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, janvier 1962.

[Wat06] Thomas WATTEYNE, Using existing network simulators for power-aware self-organizing wireless sensor network protocols. Research Report RR-6022, Institut National de Recherche en Informatique et Application (INRIA), Grenoble, France, octobre 2006.

[WK04] Limin WANG et Sandeep S. KULKARNI, MNP : multihop network reprogramming service for sensor networks. Dans *2nd International Conference on Embedded networked sensor systems (SenSys '04)*, pages 285–286, Baltimore, MD, USA, novembre 2004.

[WK06] Limin WANG et Sandeep S. KULKARNI, Gappa : Gossip based multi-channel reprogramming for sensor networks. Dans *2nd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '06)*, pages 119–134, San Francisco, CA, USA, juin 2006.

[WLJC05] Guojun WANG, Dingzhu LU, Weijia JIA et Jiannong CAO, Reliable gossip-based broadcast protocol in mobile ad hoc networks. Dans *1st International Conference on Mobile Ad-hoc and Sensor Networks (MSN '05)*, pages 207–218, Wuhan, China, décembre 2005.

[WMI+07] Hakim WEATHERSPOON, Hugo MIRANDA, Konrad IWANICKI, Ali GHODSI et Yann BUSNEL, Gossiping over storage systems is practical. *ACM Operating System Review*, 41(5):75–81, octobre 2007.

[WNE02] Jeffrey E. WIESELTHIER, Gam D. NGUYEN et Anthony EPHREMIDES, Energy-efficient broadcast and multicast trees in wireless networks. *Kluwer Mobile Networks and Applications*, 7(6):481–492, décembre 2002.

[WP02] Jennifer L. WONG et Miodrag POTKONJAK, Search in sensor networks : Challenges, techniques and applications. Dans *4th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '02)*, pages IV–3752–IV–3755, Orlando, FL, USA, mai 2002.

[WS98] Duncan J. WATTS et Steven H. STROGATZ, Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, juin 1998.

[WSS05] Bernard WONG, Aleksandrs SLIVKINS et Emin Gun SIRER, Meridian : A lightweight network location service without virtual. *ACM SIGCOMM Computer Communication Review*, 35(4):85–96, octobre 2005.

[XHE01] Ya XU, John HEIDEMANN et Deborah ESTRIN, Geography-informed energy conservation for ad-hoc routing. Dans *7th ACM International Conference on Mobile Computing and Networking (MobiCom '01)*, pages 78–84, Rome, Italie, juillet 2001.

[XW04] Bo XU et Ouri WOLFSON, Data management in mobile peer-to-peer networks. Dans *2nd International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P '04)*, pages 1–15, Toronto, Canada, août 2004.

[YG02] Yong YAO et Johannes GEHRKE, The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 31(3):9–18, septembre 2002.

Table des figures

1.1 Echantillon de spécimens de capteurs existants. 18
1.2 Constitution standard d'un capteur. 19
1.3 Zone de transmission d'un capteur standard 20
1.4 Probabilité de réussite d'une transmission en fonction de la distance 20
1.5 Surface de couverture d'un RCsF. 24

2.1 Illustration d'une trajectoire réelle et observée d'un objet o 33
2.2 Concrétisation de l'impossibilité de MOTI 36
2.3 Graphe des états d'un système à 2 objets mouvants sur un GCC à 4 sommets. . 38
2.4 Faillibilité des états et des mouvements globaux. 38
2.5 Confusion possible entre les deux trajectoires. 42
2.6 Schématisation du GCC illustrant l'existence potentielle de chemins parallèles . 42
2.7 Illustration de la faillibilité des cycles dans le GCC. 44

3.1 Projection en sur-couches 59
3.2 Évolution de la structure d'un LIGH-t-LAYER suivant différents évenements . . 60
3.3 Exemple d'une division de l'espace en 3×2 cellules. 63
3.4 Instance de broadcast dans un LIGH-t-LAYER. 66
3.5 Distance moyenne entre l'initiateur et le nœud de contact. 70
3.6 FR de la distance moyenne entre l'initiateur et le nœud de contact. 70
3.7 Distance moyenne entre l'initiateur et le point d'entrée. 71
3.8 FR de la distance moyenne entre l'initiateur et le point d'entrée. 71
3.9 Consommation d'énergie moyenne intégrale. 74
3.10 Consommation d'énergie moyenne pour l'anycast exclusivement. 74
3.11 Consommation d'énergie moyenne pour le broadcast exclusivement. 74
3.12 Consommation d'énergie moyenne pour l'entretien de la structure. 74

3.13 Exemple de référence d'un réseau à 1000 nœuds et 10 types. 74
3.14 Consommation d'énergie totale pour le k-cast dans SOLIST. 74
3.15 Consommation d'énergie totale intégrale avec SOLIST. 75
3.16 Consommation d'énergie totale intégrale sans SOLIST. 75
3.17 Consommation d'énergie totale pour l'anycast dans SOLIST. 75
3.18 Consommation d'énergie totale pour l'anycast par marche aléatoire. 75
3.19 Consommation d'énergie totale pour le broadcast dans SOLIST. 75
3.20 Consommation d'énergie totale pour le broadcast par inondation. 75

5.1 Graphe d'interaction avec 4 agents partageant la même distribution sortante des probabilités, et la chaîne de Markov associée pour la primitive *ou*. 96
5.2 Chaîne de Markov associée à la primitive *ou* pour n agents partageant la même distribution sortante des probabilités. 97
5.3 Comparaison des différentes DSP pour un agent donné. 99
5.4 Vitesse de convergence de la primitive *ou* en fonction du nombre d'agents dans un graphe complet. 101
5.5 Vitesse de convergence de la primitive *majorité* en fonction du nombre d'agents dans un graphe complet. 101
5.6 Vitesse de convergence de la primitive *somme modulo 4* en fonction du nombre d'agents dans un graphe complet. 101
5.7 Vitesse de convergence de la primitive *ou* en fonction du cœfficient de brassage dans un graphe complet. 103
5.8 Vitesse de convergence de la primitive *ou* en fonction de l'inanité du graphe sur une population de 100 agents. 103
5.9 Vitesse de convergence de la primitive *ou* en fonction du cœfficient de clique sur un graphe restreint à 48 agents. 103
5.10 Temps de convergence de la primitive *ou* en fonction du nombre d'agents avec des rencontres simples. 107
5.11 Temps de convergence de la primitive *majorité* en fonction du nombre d'agents avec des rencontres simples. 107
5.12 Temps de convergence de la primitive *ou* en fonction du nombre d'agents avec des rencontres diffuses uniformes. 107
5.13 Temps de convergence de la *majorité* en fonction du nombre d'agents avec des rencontres diffuses uniformes. 107

6.1 Schématisation des relations de puissance des protocoles épidémiques. 124
6.2 Schématisation des relations de puissance de tous les modèles. 125
6.3 Évolution du système en fonction de l'avancement d'un protocole épidémique . 131

Liste des Algorithmes

2.1 Construction du graphe des états 39
2.2 Contrainte des mouvements des objets en temps réel 45
2.3 Prévention de mouvements infaillibles 50
2.4 Détection répartie d'états faillibles – Sans information topologique 52
2.5 Détection répartie d'états faillibles – Avec informations topologiques 53
3.1 ANYCAST(t) 65
3.2 BROADCAST(t) 67
3.3 KCAST(t) 68
6.1 Protocole épidémique générique 119
6.2 Gestion des communications asynchrones 123

Lexique

Acronymes

ADC . . . . . Convertisseur Analogique-vers-Numérique
asyncPENA . . . . . Sous-classe de PENA sur communications asynchrones
asyncPENI . . . . . Sous-classe de PENI sur communications asynchrones

CAN . . Content Addressable Network (table de hachage distribuée proposée dans [RFH+01])

DSP . . . . . Distribution Sortante des Probabilités d'interaction

FR . . . . . Fonction de répartition

GCC . . . . . Graphe de Connectivité des Chemins
GCP . . . . . Gossip-based Code Propagation
GPSR . . . . . Greedy Perimeter Stateless Routing (protocole de routage sur RCsF statique)

LIGH-t-LAYER . . . . . Sur-couche logique regroupant tous les nœuds de type t

MANET . . . . . Réseaux Mobiles Ad-hoc
MAPC . . . . . Mobilité Appliquée aux Protocoles de Communauté
MAPP . . . . . Mobilité Appliquée aux Protocoles de Population
MCMC . . . . . Méthode de Monte-Carlo pour les Chaînes de Markov
MEMS . . . . . Systèmes Micro-Électro-Mécaniques
MIPS/W . . . . . Millions d'Instructions Par Seconde et par Watt
MOTI . . . . . Multiples Objets Tracking and Identification

NSMM ..... Machine non-déterministe à modification de stockage (*cf.* [vEB89])

PàP ..... Pair-à-Pair
PC ..... Protocoles de Communauté
PENA ..... Protocoles épidémiques sur des nœuds anonymes
PENI ..... Protocoles épidémiques sur des nœuds identifiables
PP ..... Protocoles de Population
PS ..... Échantillonage de nœuds (Peer Sampling)
PTA ..... Points Tournants Aléatoires

QoS ..... Qualité de Service

RCsF ..... Réseau de Capteurs sans-Fil
RPS ..... Échantillonage aléatoire de nœuds (Random Peer Sampling)
RWP ..... Random WayPoint

SOLIST .. Self-Organized Large-scale and lightweight Information-based Sensor Technology
syncPENA ..... Sous-classe de PENA sur communications synchrones
syncPENI ..... Sous-classe de PENI sur communications synchrones

UCT ..... Unité Centrale de Traitement

Notations mathématiques

$[\![a, b]\!]$ ..... Ensemble des entiers compris entre a et b inclus
$\langle \ldots \rangle$ ..... Suite d'objets ordonnée
$\otimes$ ..... Produit de loi de probabilité

$\mathbf{1}_X$ ..... Fonction indicatrice du prédicat X

$d(a, b)$ ..... Distance euclidienne entre les points a et b

$\mathbb{E}_{\mathcal{I}}(\tau_{\mathcal{A}})$ ..... Espérance du temps d'atteinte de l'ensemble d'état $\mathcal{A}$

$\lambda_{\otimes}$ ..... Mesure produit invariante de Lebesgue

$\mathcal{M}$ ..... Chaîne de Markov

∇ ..... Gradient d'une fonction

$\mathbb{P}$ ..... Loi de probabilité
$\pi(A)$ ..... Permutation des éléments de l'ensemble A

$\mathcal{S}_C(x)$ ..... Sphère de centre C et de rayon x
ssi ..... si, et seulement si
$\sigma^2_{\mathcal{I}}(\tau_{\mathcal{A}})$ ..... Écart-type du temps d'atteinte de l'ensemble d'état $\mathcal{A}$

$\tau_{\mathcal{A}}(\omega)$ ..... Premier temps d'atteinte de l'ensemble d'état $\mathcal{A}$

Notations algorithmiques

**-cast* ..... Ensemble des primitives d'anycast, k-cast et broadcast

$adj_G(v)$ ..... Ensemble des nœuds adjacents au sommet v dans le graphe G

C_v ..... Cœfficient de clique du sommet v
C_G ..... Cœfficient de clique global du graphe G
$\mathcal{C}$ ..... Protocole de communauté

δ ..... Relation de transition sur les couples d'états

ep_t ..... Coordonnées du point d'entrée de type t le plus proche

f_x, f_y ..... Fonctions de hachage sur $[0, 1[$

$G(V, E)$ ..... Graphe G composé de l'ensemble des sommets V et des arcs E
$\mathbb{G}$ ..... Ensemble des graphes $G(V, E)$
$\mathcal{G}$ ..... Protocole épidémique

ι ..... Fonction d'association entre l'entrée et l'état d'un agent

$\kappa_c(S)$ ..... Ensemble des sommets inoccupés du cycle c dans l'état S

$l_p(v, v')$ ..... Longueur entre v et v' sur le chemin p en terme de nombres de sommets
ℓ_c ..... Longueur du cycle c en terme de nombres de sommets
L_G ..... Longueur caractéristique des chemins du graphe G
$loc_t(o), \overline{loc}_t(o)$ ..... Position réelle et observée d'un objet o au temps t
$\Lambda(\Upsilon, \Theta)$ ..... Graphe d'interaction Λ composé des agents de Υ et des interactions de Θ

m_x, m_y ..... Division de l'espace de coordonnées (en abscisses et ordonnées)
M_t ..... Mouvement unitaire global de tous les objets du système
$\mathbb{M}$ ..... Ensemble de tous les mouvements unitaires possibles du système
$mov_t(o), \overline{mov}_t(o)$ ..... Mouvement réel et observé d'un objet o entre t et $t + 1$
$\mu_{c,t}$ ..... Longueur de la chaîne minimale continue d'objets

n ..... Taille du système (Nombre de participants)

ω ..... Fonction d'association entre l'état d'un agent et sa valeur de sortie

P_{min} ..... Probabilité minimale de réception d'un message
$P_{t_i,t_j,o}, \overline{P}_{t_i,t_j,o}$ ..... Trajectoire réelle et observée d'un objet o entre t_i et t_j
$P_{t_i,t_j}, \overline{P}_{t_i,t_j}$ ..... Trajectoire globale réelle et observée de l'ensemble des objets entre t_i et t_j
$\mathcal{P}$ ..... Protocole de population
$\mathcal{P}_v$ ..... DSP de l'agent v
$\mathbb{P}_{t_i,t_j}$ ..... Ensemble des trajectoires possibles de tous les objets entre t_i et t_j
$\mathbb{P}^k$ ..... Ensemble des trajectoires possibles avec au plus k mouvements simultanés
$p_{v,v'}, p_\theta$ ..... Probabilité de rencontre de deux agents v et v' ou de l'interaction θ

Q ..... Ensemble fini de tous les états possibles des agents

r ..... Rayon de transmission uniforme
R ..... Rayon de transmission maximal

S_t ..... État du système au temps t
$\mathbb{S}$ ..... Ensemble de tous les états possibles du système
$SG(\mathbb{S}, \mathbb{M})$ ..... Graphe des états $\mathbb{S}$
Σ ..... Alphabet d'entrée fini

U ..... Ensemble infini d'identifiants

$view_t$ ..... Vue de voisinage dans le LIGH-t-LAYER

ws_x, ws_y ..... Dimensions du système (en abscisses et ordonnées)

ξ ..... Fonction booléenne à deux variables
Ξ ..... Exécution d'un protocole de population

Y ..... Alphabet de sortie fini

Résumé

Les systèmes informatiques ont connu récemment de grandes avancées dans leur conception. D'une part, la démocratisation des réseaux via la croissance exponentielle de l'Internet a permis d'envisager des systèmes à l'échelle mondiale, visant de mettre en commun une multitude de ressources à travers la planète entière. D'autre part, la réduction continue de la taille des équipements informatiques a permis l'apparition de matériels miniatures. Le jumelage de ces deux évolutions est à l'origine de l'apparition des réseaux de capteurs sans fil. Le spectre des applications potentielles de ces réseaux est extrêmement large, que cela soit dans le contexte d'une infrastructure fixe autant que dans l'informatique embarquée. Cette thèse propose un ensemble de contributions pour la gestion de l'information à la fois dans le contexte mobile et statique. Édifiées autour des mêmes propriétés de collaboration et d'auto-organisation, ces propositions sont conçues selon une méthodologie de la théorie vers la pratique. Cette thèse vise ainsi, en premier lieu, une analyse théorique *a priori* d'une application classique des réseaux de capteurs statiques, à savoir le suivi de trajectoires d'objets mobiles non identifiés. Par la suite, nous étendons le spectre des applications visées en proposant une structure générique à toute mise en œuvre réelle de réseaux de capteurs statiques. En second lieu, nous considérons une modélisation de la mobilité permettant d'analyser fondamentalement les impacts de celle-ci sur la convergence des protocoles dits de population. Enfin, nous établissons un parallèle entre les travaux menés théoriquement sur les réseaux de capteurs mobiles avec ceux plus pratiques et empiriques proposés dans le cadre des protocoles épidémiques sur réseaux filaires. En démontrant que ces deux domaines portent en réalité sur la même classe de protocoles – et donc de problèmes – nous ouvrons ainsi une voie captivante pour de futures recherches dans chacun de ces deux domaines, par l'utilisation de l'un dans l'autre.

Abstract

Self-organized collaborative information system for large-scale wireless sensor networks. "Moving theory into practice"

Computer systems have recently known an increasing interest. In first hand, the late-democratisation of network by the high expansion of Internet let us consider some worldwide systems, in order to share with everyone a mass of resources among the whole planet. In other hand, the continuous reduction of the computer size lets appearing several miniaturized equipments. The twinning of these both evolutions are at the origin of the wireless sensor network (WSN) emergence. The application spectrum of these networks is extremely large, both in fixed infrastructure than in embedded system. This thesis proposed a set of contribution for information management both in mobile and static networks. Build among the same properties as collaboration and self-organization, these propositions are design along a common methodology : "moving theory into practice". Then, in the first place, this thesis aims at *a priori* theoretical analysis of a very classical application in static WSN : the trajectory tracking of anonymous objects. After that, we enlarge the considered application spectrum by proposing a generic structure for real implementation of static WSN. In the second place, we considered an establishment of a model for mobility pattern, which can be used to formally analyze the convergence behaviour of population protocols. Last but not least, we parallel the works made in the theoretical context of mobile WSN with the practical and empirical ones proposed on gossip-based protocols for wired networks. By proving that the both domains are equivalent, we show that we can use any result of one in the other. Then, we open an attractive way for further research in the both communities.