Algorithmes de communication par bips dans les réseaux sans fil

par

Kokouvi Hounkanli

Thèse présentée au

Département d'informatique et d'ingénierie

Pour l'obtention du grade de

Philosophiae Doctor (Ph.D.)

en Sciences et technologies de l'information

Membres du jury :

Président	Kamel Adi, Ph.D., U.Q.O.
Examinateur interne	Jurek Czyzowicz, Ph.D., U.Q.O.
Examinateurs externes	Paola Flocchini, Ph.D., University of Ottawa
	Evangelos Kranakis, Ph.D., Carleton University
Directeur de recherche	Andrzej Pelc, Ph.D., U.Q.O.

Mars 2018

Resumé

Le sujet de cette thèse est la communication dans les réseaux sans fil. Nous proposons la conception et l'analyse d'algorithmes de communication dans un modèle très faible : les seuls messages envoyés dans une ronde sont des bips. Le modèle de communication par les bips est largement applicable, car il impose de petites exigences sur les appareils communicants, en s'appuyant uniquement sur la détection du signal ou de son absence.

Premièrement, nous étudions la tâche fondamentale de la diffusion de message dans un réseau. Notre objectif est d'étudier comment la combinaison de deux faiblesses du modèle de communication, des messages simples et courts d'une part, et la manière de livraison asynchrone de bips (le temps entre l'envoi et la réception est fini mais imprévisible) d'autre part, influence l'efficacité de la communication. Deuxièmement, nous étudions les algorithmes de communication dans le modèle de la communication par les bips dans un canal à accès multiple (MAC) pour les tâches de la synchronisation globale et du consensus d'une part et la tâche d'élection du chef d'autre part. Nous abordons le scénario où les pannes de communication se produisent dans le canal d'une façon aléatoire.

Abstract

The subject of this thesis is the communication in wireless networks. We propose the design and analysis of communication algorithms in a very weak model : the only messages sent in a round are beeps. The model of communication by beeps is widely applicable because it imposes small requirements on communicating devices, relying solely on the detection of the signal or its absence.

First, we study the fundamental task of broadcasting a message in a network. Our goal is to study how the combination of two weaknesses of the communication model, simple and short messages on the one hand, and the way of asynchronous delivery of beeps (the time between sending and receiving is finite but unpredictable) on the other hand, influences the efficiency of communication. Second, we study communication algorithms in the model of communication by beeps in a multi-access channel (MAC) for the tasks of global synchronization and consensus on the one hand and the task of leader election on the other hand. We address the scenario where communication failures occur in the channel in a random way.

Table des matières

Liste des figures

Chapitre 1

Introduction

Un réseau informatique est un ensemble de machines ou appareils connectés les uns aux autres. La connexion entre ces appareils peut se faire de manière physique (avec fil) ou via des ondes radio ou infrarouges (sans fil). Un réseau informatique permet, dépendamment des applications, d'échanger des informations ou de partager des ressources. L'intérêt pour la communication dans les réseaux informatiques a augmenté au cours des dernières décennies. Certes une raison importante est le rôle grandissant que les réseaux informatiques et la télécommunication jouent dans la vie de la société moderne. Le calcul distribué constitue un fondement de ces deux domaines d'application de l'informatique. Plusieurs processeurs avec un certain dégré d'autonomie, situés à des endroits différents doivent coopérer pour accomplir ensemble une tâche de traitement des informations. Chaque processeur exécute localement des parties de la tâche, indépendamment des autres processeurs. Toutefois, pour accomplir la tâche entière, les processeurs doivent partager certaines ressources, ce qui exige une communication entre eux pendant l'exécution de la tâche.

1.1 Les réseaux informatiques

Les opérateurs dans diverses industries recherchent de nouvelles façons de maximiser leurs investissements dans les réseaux de communication. Les réseaux avec fil et sans fil peuvent être comparés en termes de performance, de sécurité, de fiabilité, de coûts d'installation et de maintenance, ainsi que de facilité de déploiement. Dans le réseau sans fil, l'utilisateur est libre de circuler dans la zone de couverture du réseau pour y connecter. Avec une connexion sans fil, les utilisateurs peuvent également partager des informations et des ressources avec d'autres ports sans l'aide de câbles. Le réseau sans fil permet un transfert instantané d'informations tel que dans les médias sociaux qui l'ont rendu plus facile et plus rapide pour ses utilisateurs. Par exemple, les utilisateurs peuvent prendre des photos et les télécharger sur leurs pages de médias sociaux tels que Facebook, Twitter, Instagram, WhatsApp et bien d'autres. L'espace est une autre caractéristique des réseaux sans fil. Ces réseaux offrent de nombreux avantages sur le plan de la communication pour les lieux difficiles à connecter : de part et d'autre d'une rue, d'une rive à une autre, un entrepôt éloigné des locaux principaux ou encore des immeubles physiquement séparés mais fonctionnant ensemble. Le réseau sans fil permet ainsi aux utilisateurs de tracer un périmètre au sein duquel un appareil sera en mesure de communiquer avec les autres appareils dans les alentours. Le réseau sans fil apporte donc une alternative au réseau physique tel que les lignes coaxiales ou encore la fibre optique généralement plus coûteuses.

1.2 Les communications dans les réseaux sans fil

Les communications électroniques sans fil représentent aujourd'hui la plus grande part de l'industrie des télécommunications. En effet, les utilisateurs exigent en tout temps la mobilité, le haut débit et le multimédia. La mobilité qui permet d'accéder au service n'importe où sur un territoire donné est cet élément qui révolutionne le marché des télécommunications, des technologies de l'information et de la communication. Il est impossible d'accéder aux services n'importe où dans un environnement câblé. La mobilité s'appuie sur la propagation des ondes électromagnétiques ou des ondes hertziennes ou encore des ondes radioélectriques. Le concept radio indique toute communication assurée sans support matériel, ainsi le terme radiodiffusion englobe toutes les communications sans fil. La radiodiffusion désigne le service fourni à l'aide d'ondes électromagnétiques, tout comme la radiotéléphonie indique que le service téléphonique n'utilise pas le support câblé. Dans ce système de communication sans fil, les émetteurs et les récepteurs sont reliés par des ondes électromagnétiques. Il n'existe aucune liaison physique entre l'émetteur de son ou d'images et le récepteur radioélectrique qui peut être un poste de radio ou un téléviseur. C'est l'un des services sans fil les plus utilisés. La communication sans fil à micro-ondes est un type de communication efficace. Elle est utilisée dans les liaisons point-à-point et point-à-multipoints. Le principal inconvénient des signaux micro-ondes est leur vulnérabilité par rapport aux mauvais temps, en particulier la pluie. Les satellites de télécommunications sont utilisées pour toutes sortes de communications : radiodiffusion, télévision, téléphonie, transmission de données, etc. Parmi les inconvénients de la communication sans fil, on peut citer, entre autres : l'accès facile d'une personne non autorisée aux signaux sans fil qui se propagent dans l'air. Aujourd'hui, les communications

sans fil sont indispensables dans le monde. Il existe plus de 8 milliards de terminaux mobiles (téléphones cellulaires, tablettes électroniques) connectés à des réseaux de différents types grâce aux ondes électromagnétiques. Il suffit d'imaginer le monde un instant sans les services mobiles pour se faire une idée de l'importance d'une liaison sans fil. En plus d'exploits technologiques, les services et applications rendues possibles par ce moyen de communication deviennent la base de l'économie mondiale. Cela a conduit à de nombreuses menaces de sécurité. Il est très facile pour les pirates de saisir les signaux sans fil qui se propagent dans l'air. Il est très important de sécuriser le réseau sans fil afin que l'information ne soit pas exploitée par les utilisateurs non autorisés. Cela augmente également le risque de perte d'informations. De solides protocoles de sécurité doivent être créés pour sécuriser le réseau sans fil. Une autre façon de sécuriser le réseau sans fil est d'avoir un système de prévention des intrusions sans fil.

1.3 La communication sans fil par les bips

Le modèle de communication par les bips impose de petites exigences sur les appareils communicants, en s'appuyant uniquement sur la détection du signal ou de son absence. Les bips sont un moyen de communication encore plus facile à déployer que l'utilisation des messages radio à un bit, car ceux-ci demandent de différencier entre trois états (0,1 et aucun message), tandis que la communication par les bips demande de différencier uniquement entre un signal et son absence. La communication par les bips est largement appliquée dans différentes implémentations de réseaux de communication sans fil. Le modèle des bips a reçu une attention considérable, en partie à cause de sa relation avec d'autres modèles de communication tels que celui des réseaux radio ad-hoc.

Le modèle de communication par les bips suppose que les nœuds du réseau ont une connaissance minimale de leur environnement et que leurs capacités de communication sont limitées. Il peut donc être appliqué aux appareils de très faible puissance. Généralement, les nœuds n'ont pas d'informations sur la structure locale ou globale du réseau, ils n'ont pas non plus accès à des horloges synchronisées, et ils sont réveillés à des moments arbitraires. Ce modèle de communication peut être implémenté même dans des environnements de réseaux radio extrêmement restreints. Le réseau est modélisé comme un graphe connexe non-orienté, où les nœuds du graphe représentent les périphériques du réseau et les arêtes représentent une accessibilité directe entre deux nœuds. Le modèle de la communication par les bips est apparu comme une alternative au modèle radio traditionnel. Dans ce modèle, un nœud peut choisir, dans chaque ronde de communication, de biper ou de ne pas biper (rester silencieux). Si un nœud bipe, il n'obtient pas de signaux de ses voisins. Si un nœud est silencieux, il apprendra si tous ses voisins sont également silencieux ou si au moins un de ses voisins a bipé.

Le modèle de la communication par les bips est intéressant comme un modèle largement applicable, qui nécessite des dispositifs de communication très faibles, et peut être déployé là où les circonstances restrictives tendent à freiner les communications dans des modèles plus complexes.

Chapitre 2

Présentation des modèles et des résultats

Dans ce chapitre nous présentons les modèles utilisés dans cette thèse et les résultats obtenus.

Le fil unificateur de tous nos modèles est la communication par bips. Un bip est un signal qu'un processeur envoie à tous ses voisins dans un réseau sans fil. Il faut souligner une différence importante entre les modèles de communication par bips et le modèle classique de communication radio. Dans ce dernier, si deux messages arrivent simultanément dans un nœud du réseau, celui-ci n'entend que le bruit qui peut être ou ne pas être possible à distinguer du silence ayant lieu si aucun voisin ne transmet. Par contre dans la communication par bips le voisin d'un nœud qui bipe entend toujours un bip. L'autre différence c'est que les bips sont des messages plus courts et plus simples que les messages employés en général dans la communication radio.

Nonobstant ce fil unificateur, les modèles de communication utilisés dans les chapitres suivants diffèrent dans plusieurs aspects. Avant de décrire ces

modèles en détails nous soulignons ces différences.

Dans le chapitre 4 où nous considérons la tâche de diffusion dans les réseaux asynchrones, nous utilisons des bips de deux types, les bips forts et les bips faibles, car nous avons prouvé que la diffusion asynchrone en utilisant des bips uniformes est impossible. Par contre, dans les chapitres 5 et 6, nous utilisons des bips uniformes.

Une autre différence est le temps de livraison des bips aux voisins. Dans les chapitres 5 et 6 nous considérons la livraison synchrone où le bip est livré aux voisins dans la ronde où il a été envoyé,
pendant que dans le chapitre 4 la livraison est asynchrone : un bip envoyé dans une ronde est livré aux voisins dans une ronde ultérieure décidée par l'adversaire.

La troisième différence est le type de réseau considéré. Dans le chapitre 4 nous considérons les réseaux de topologie arbitraire, par contre dans les chapitres 5 et 6 nous considérons le canal à accès multiple qui est équivalent à un réseau complet.

Finalement, la quatrième différence concerne les pannes. Dans le chapitre 4 nous supposons que tous les bips sont livrés correctement et dans les chapitres 5 et 6 nous considérons des pannes aléatoires qui peuvent avoir lieu dans le canal à accès multiple.

Nous passons maintenant à la description détaillée des modèles et des résultats obtenus.

Dans le chapitre 4, nous étudions des algorithmes déterministes pour la tâche de diffusion de message à partir d'une source, qui est une tâche de communication fondamentale, dans un modèle très faible de communication

sans fil. Les seuls signaux envoyés par les nœuds sont des bips. En outre, ils sont livrés aux voisins du nœud bipant d'une manière asynchrone : le temps entre l'envoi et la réception est fini mais imprévisible. Nous observons d'abord que dans ce scénario, aucune communication n'est possible, si les bips sont tous de la même force. Par conséquent, nous étudions la diffusion de message dans le modèle de communication avec les bips bivalents, où chaque bip peut être soit faible soit fort. À la réception, si exactement un bip faible est reçu par un nœud dans une ronde, il est entendu comme faible. Toute autre combinaison de bips reçus dans une ronde est entendue comme un bip fort.
Notre objectif est d'étudier comment la combinaison de deux faiblesses du modèle de communication, des messages très simples et courts d'une part, et la manière de livraison asynchrone des bips d'autre part, influence l'efficacité de la communication. Antérieurement, chacune de ces deux faiblesses du modèle de communication a été étudiée séparément. La diffusion de message et les commérages synchrones avec les bips ont été étudiés dans [16]. La diffusion asynchrone dans le modèle radio, où de longs messages peuvent être envoyés en une ronde, a été étudiée dans [5, 11, 49]. À notre connaissance, la combinaison des deux faiblesses du modèle, à savoir des messages très courts et une livraison asynchrone, n'a jamais été étudiée auparavant.

Nous étudions ensuite les algorithmes de communication dans le modèle de la communication par les bips dans un canal à accès multiple (MAC) pour les tâches de la synchronisation globale et du consensus dans le chapitre 5 et la tâche d'élection du chef dans le chapitre 6. Nous abordons le scénario où les pannes de communication se produisent dans le canal d'une façon aléatoire. Certains processeurs se réveillent spontanément, dans des rondes possiblement différentes décidées par un adversaire. Dans chaque ronde, un processeur réveillé peut soit écouter, c'est-à-dire rester silencieux, soit parler, c'est-à-dire émettre

un signal. Dans chaque ronde, une panne peut se produire dans le canal indépendamment avec une probabilité constante $0 < p < 1$. Dans une ronde sans panne, un processeur réveillé entend un bip s'il écoute dans cette ronde et si un ou plusieurs autres processeurs bipent dans cette ronde. Un processeur encore dormant dans une ronde sans panne dans laquelle un autre processeur bipe est réveillé par ce bip qu'il entend. Dans une ronde défectueuse (c'est-à-dire avec panne), rien n'est entendu, quel que soit le comportement des processeurs.

La synchronisation globale est une condition préalable importante pour de nombreuses tâches distribuées. La communication entre les processeurs se déroule en rondes synchrones. Les processeurs sont réveillés dans des rondes possiblement différentes. L'horloge de chaque processeur commence dans sa ronde de réveil montrant la ronde locale 0, et ensuite tique une fois par ronde en incrémentant la valeur de l'horloge locale par un. La ronde globale 0, inconnue des processeurs, est la ronde de réveil du premier processeur. La synchronisation globale (ou l'établissement d'une horloge globale) signifie que chaque processeur choisit une ronde d'horloge locale telle que les rondes choisies correspondent toutes à la même ronde globale t. La réalisation de la synchronisation globale permet de supposer que dans une tâche ultérieure, tous les processeurs commencent dans la même ronde. Cette hypothèse est souvent utilisée dans la résolution de divers problèmes distribués.

Pour la tâche du consensus, les processeurs ont des valeurs d'entrée dans un ensemble V d'entiers non-négatifs. L'objectif pour tous les processeurs est de satisfaire les exigences suivantes.

Terminaison : tous les processeurs doivent afficher une valeur de l'ensemble V.

Accord : toutes les valeurs affichées doivent être identiques.

Validité : si toutes les valeurs d'entrée sont égales à v, alors toutes les valeurs affichées doivent être égales à v.

Pour la tâche d'élection du chef, les processeurs ont des étiquettes uniques qui sont des entiers non-négatifs. Un seul processeur doit décider qu'il est le chef, et tout autre processeur doit décider qu'il n'est pas le chef.

2.1 Résultats concernant la diffusion de message

Les résultats obtenus ont été publiés dans l'article :

K. Hounkanli et A. Pelc

Asynchronous Broadcasting with Bivalents Beeps

Proc. 23^{rd} International Colloquium on Structural Information and Communication Complexity (SIROCCO 2016), 291-306.

Nous avons considéré quatre niveaux de connaissance que les nœuds peuvent avoir sur le réseau : l'anonymat (aucune connaissance), l'ad-hoc (tous les nœuds ont des étiquettes distinctes et chaque nœud connaît seulement sa propre étiquette), la connaissance du voisinage (chaque nœud connaît son étiquette et les étiquettes de tous ses voisins) et une connaissance totale (chaque nœud connaît toute la carte étiquetée du réseau et l'identité de la source). Le coût d'un algorithme de diffusion est le nombre total de bips envoyés par tous les nœuds. Cela mesure (l'ordre de grandeur de) la consommation d'énergie par le réseau, car l'énergie utilisée pour envoyer un bip fort peut être considérée comme un multiple constant de celle utilisée pour envoyer un bip faible.

Nous avons d'abord montré que, dans le cas anonyme, la diffusion est impossible même pour les réseaux très simples. Pour chacun des trois autres niveaux de connaissance, nous avons fourni des bornes supérieures et inférieures sur le coût minimal d'un algorithme de diffusion. Nos résultats montrent les séparations entre tous ces scénarios. L'écart du coût de la diffusion entre les niveaux de connaissances ad-hoc et du voisinage est beaucoup plus important qu'entre la

connaissance du voisinage et la connaissance totale, bien que dans les deux premiers niveaux, la connaissance des nœuds soit locale et, dans le dernier niveau, elle soit globale.

2.2 Résultats concernant la synchronisation globale et le consensus

Les résultats obtenus ont été publiés dans l'article :

K. Hounkanli, A. Miller, A. Pelc

Global Synchronization and Consensus Using Beeps in a Fault-Prone MAC

Proc. 12th International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS 2016), 16-28.

Nous avons conçu et nous avons analysé, pour toute constante $\epsilon > 0$, un algorithme de synchronisation globale $\epsilon-$certain déterministe qui fonctionne en temps constant dans n'importe quel MAC où les rondes sont sujet aux pannes indépendantes avec probabilité constante $0 < p < 1$, qui utilise la communication par les bips. Un algorithme qui fonctionne avec probabilité d'erreur au plus ϵ, pour un $\epsilon > 0$ donné, s'appelle $\epsilon-$certain.

En tant qu'application, nous avons résolu le problème du consensus dans un MAC avec pannes aléatoires qui utilise la communication par les bips. À l'aide de la synchronisation globale, nous avons donné un algorithme du consensus $\epsilon-$certain déterministe qui fonctionne en temps $O(\log w)$ dans un MAC avec pannes aléatoires, où w est la plus petite valeur d'entrée de tous les processeurs participants. Nous avons montré que ce temps ne peut pas être amélioré, même si le MAC est sans panne.

2.3 Résultats concernant l'élection du chef

Comme ci-dessus, nous avons considéré un MAC de N processeurs, où les rondes sont sujet aux pannes indépendantes avec probabilité constante $0 < p < 1$. Chaque processeur a une étiquette unique dans l'ensemble $\{1, \cdots, L\}$, où L est un entier non-négatif polynomial en n, appelé l'espace des étiquettes. Chaque processeur connaît son étiquette et connaît L. Nous avons conçu et nous avons analysé un algorithme d'élection du chef presque certain (c'est-à-dire qui fonctionne avec probabilité au moins $1 - \frac{1}{N}$) travaillant en temps $O(\log^2 L) = O(\log^2 N)$.

Chapitre 3

Revue de littérature

Dans cette revue de littérature, nous considérons la communication dans divers réseaux et les tâches distribuées les plus importantes à y accomplir. Cet aperçu est organisé par modèles de communication et ensuite par tâches dans les sections 3.1 et 3.2. Pour la section 3.3, nous présentons la littérature concernant les tâches distribuées dans les autres modèles de communication. Nous considérons premièrement les réseaux sans fil, qui sont des réseaux de communication où différentes stations communiquent entre elles par des ondes radio.

La section 3.1 est consacrée aux réseaux radio dans lesquels nous distinguons le modèle de graphe (un modèle dans lequel les nœuds sont représentés par des points et les liens par des arêtes) et le modèle "Signal-to-Interference-and-Noise Radio" (SINR) qui est un modèle physique. Ce dernier a été introduit plus récemment. Dans cette section, nous étudions les problèmes de diffusion de message, de commérage, du réveil, de l'élection du chef et du consensus. Particulièrement, nous classons les tâches de diffusion de message et de commérage selon leur dégré de synchronie.

Dans la section 3.2, nous discutons des réseaux de la communication par les bips,

en considérant les tâches de diffusion de message, de commérage, de l'élection du chef et autres.
Dans la section 3.3, nous considérons diverses tâches distribuées, telles que la diffusion de message, le commérage, le réveil, l'élection du chef, le consensus et la sélection dans les modèles de communication avec fil.

3.1 Réseaux radio

Un réseau radio est un ensemble de stations qui communiquent entre elles par les envois et réceptions de signaux radio. Chaque station a une portée de transmission qui lui permet d'envoyer directement (dans un seul envoi) un message à toutes les stations à sa portée. Une station u qui n'est pas à portée de transmission d'une station v ne peut être atteinte que par une transmission à plusieurs envois [20].
Un réseau radio classique peut être modélisé par un graphe direct [13, 17, 18, 20] ou indirect [3, 31, 43] $G = (V, E)$, où les nœuds du graphe représentent les stations du réseau et un arc entre les nœuds v et u indique que v peut envoyer des messages à u. La communication se déroule généralement en tranches de temps (rondes) synchones. Une station reçoit un message durant un certaine ronde si exactement un de ses voisins envoit un message pendant cette ronde. Un des aspects importants dans ce processus de transmission est la détection ou non de collisions par un noœud v quand au moins deux des voisins de ce noœud lui envoient leurs messages simultanément. Dans le modèle avec détection de collisions, le nœud recevant peut entendre la différence entre cette situation (bruit) et la situation où aucun voisin ne transmet (silence). Dans le modèle sans détection de collisions, il n'y a pas de différence entre le bruit et le silence. Pour capturer le problème des interférences dans le réseau radio, le modèle physique

SINR (*Signal-to-interference-and Noises*) a été introduit récemment dans la littérature. Il faut noter que ce modèle était déjà utilisé en ingénierie et que [27] qui est apparu en 2007, est un des ouvrages en technologie de l'information à en discuter. Ce modèle est basé sur trois axiomes : (i) La diminution relative d'un signal est un polynôme de la distance parcourue. (ii) L'interférence s'ajoute. (iii) Le succès d'une réception de signal est une fonction du seuil du rapport signal-interférence [34].

3.1.1 Le modèle de graphe

Dans le modèle de graphe, le réseau est représenté comme un graphe dont les nœuds sont des stations. Un arc entre les points v et u indique que la station v peut envoyer des messages à la station u. Ces modèles diffèrent sous deux aspects : l'architecture du réseau et le niveau de synchronie. Dans beaucoup d'applications, les stations du réseau sans fil sont mobiles, et alors la topologie du réseau n'est pas stable. Pour cette raison, il est préférable de s'abstenir de faire des hypothèses sur la topologie du réseau ou sur les informations que les stations détiennent concernant la topologie [44]. Dans le niveau de synchronie, nous considérons la communication synchrone et la communication asynchrone. Dans la première, on suppose qu'une horloge globale est accessible à toutes les stations et que leurs actions sont synchronisées en rondes. Dans la deuxième, la charactérisation essentielle est que la durée de toutes les actions du système est finie mais indéterminée. C'est-à-dire qu'on ne peut pas prédire la durée des transmissions par les stations. En particulier, différentes transmissions peuvent avoir des durées différentes. Cette incertitude augmente de façon considérable la difficulté de l'analyse de la communication asynchrone et la construction des algorithmes pour ce modèle. Cependant, ce modèle est important car la majorité

des systèmes existants comporte au moins un certain dégré d'asynchronie.

A. Diffusion de message et commérage

La diffusion de message et le commérage sont des tâches fondamentales de la communication dans les réseaux. Dans la diffusion de message, ou la communication un-à-tous, l'information détenue initialement dans une station (appelée source) du réseau doit être transmise à toutes les autres stations. Dans le commérage, ou la communication tous-à-tous, chaque station détient initialement un message qu'elle doit transmettre à toutes les autres stations.

A1. Communication synchrone

Dans la communication synchrone, on suppose qu'une horloge globale est accessible à toutes les stations et que leurs transmissions sont synchronisées en rondes. Pendant chaque ronde, chaque station est soit en mode de transmission de message soit en mode de réception de message. Une station v qui transmet envoie un message qui atteint potentiellement tous ses voisins. Une station u, en mode de réception pendant une ronde r, entend un message pendant cette ronde r si et seulement si exactement un de ses voisins transmet dans cette ronde r.

Dans le modèle synchrone où le réseau est modélisé par un graphe non-orienté, et sans détection de collisions, les auteurs dans [3] ont présenté un algorithme aléatoire qui travaille en temps $O((D + \log \frac{n}{\varepsilon}) \cdot \log n)$, avec probabilité $1 - \varepsilon$, où n est le nombre de stations, où D est le diamètre du réseau, et où ε est un paramètre donné, positif et inférieur à 1. Pour la diffusion déterministe, les auteurs dans [18] ont montré un algorithme de diffusion de message en temps sublinéaire pour tous les graphes de diamètre $o(\log \log n)$. Ils ont aussi construit une classe de graphes de diamètre 4, telle que chaque algorithme de diffusion de message nécessite $\Omega(\sqrt[4]{n})$ rondes sur l'un de ces graphes. La comparaison de

ce résultat avec celui de [3] démontre un écart exponentiel en temps entre l'algorithme aléatoire et l'algorithme déterministe. Dans [18], le temps de diffusion de message dans le modèle aléatoire a été amélioré à $O(D \log n + \log^2 n)$ rondes pour tout graphe avec n nœuds, où D est l'excentricité spécifiée. L'excentricité est la distance maximale entre le nœud source et tout autre nœud du graphe. Ce temps est valide avec forte probabilité. Nous utilisons le terme de forte probabilité pour indiquer une probabilité d'au moins $1 - \frac{1}{n^c}$, pour une constante $c \geq 1$, et où n est la grandeur du réseau. Pour les algorithmes aléatoires, une borne inférieure de $\Omega(D \log(N/D))$ rondes, où D est le diamètre du réseau et où N est le nombre de nœuds, est montrée dans [44]. Ce résultat implique une borne inférieure de $\Omega(D \log N)$ rondes pour tout $D \leq N^{1-\varepsilon}$, où $\varepsilon > 0$ est toute constante. En conséquence, ce résultat prouve que la borne supérieure en temps de $O(D \log N + \log^2 N)$ dans [18], est serrée.

Dans le modèle synchrone où le réseau est modélisé par un graphe orienté, et sans détection de collisions, Gianluca De Marco a montré dans [20] un algorithme déterministe qui travaille en temps $O(n \log n \log \log n)$, pour un réseau de topologie inconnue, où chaque nœud connaît seulement sa propre étiquette mais ne sait rien sur les autres paramètres du système, y compris son voisinage ou une borne supérieure sur la grandeur n du réseau. Ceci améliore le meilleur résultat précédent de $O(n \log^2 n)$ obtenu dans [13], et donc réduit exponentiellement l'écart entre la borne inférieure $\Omega(n \log n)$ et la borne supérieure, de $O(\log n)$ à $O(\log \log n)$. Ce résultat en temps de $O(n \log n \log \log n)$ pour un algorithme déterministe de De Marco (2010) peut être comparé au résultat de $O(n \log D \log \log \frac{D\Delta}{n})$ de Arthur Czumaj et Peter Davies dans [17] (2016), pour un algorithme déterministe dans un réseau radio inconnu, modélisé par un graphe orienté ou non-orienté, dans lequel les nœuds ne connaissent pas la topologie du réseau, mais connaissent seulement la taille n du réseau, le

maximum dégré Δ de tout nœud, et l'excentricité D du réseau. Ce résultat est également le premier à venir plus proche d'un facteur log-logarithmique de la borne inférieure $\Omega(n \log D)$ due à Clementi et al. (2003).

Dans le modèle synchrone où le réseau est modélisé par un graphe non-orienté, et avec détection de collisions, les auteurs dans [31] ont présenté un algorithme aléatoire distribué qui diffuse un seul message dans $O(D + \log^6 n)$ rondes, avec une très forte probabilité. Il faut comparer ce résultat à la borne supérieure de $O(D \log n + \log^2 n)$ rondes dans [18], qui est optimale dans ce modèle sans détection de collisions. Les auteurs dans [31] ont également étudié les algorithmes distribués pour la diffusion de k messages à partir d'une seule station. Ils ont montré les résultats suivants. Si la topologie est connue de toutes les stations, alors une diffusion de k messages peut se faire en $O(D+k \log n+\log^2 n)$ rondes avec une très forte probabilité. Si la topologie est inconnue, mais la détection de collisions est possible, alors la diffusion de k messages peut se faire en $O(D + k \log n + \log^6 n)$ rondes avec une très forte probabilité.

Il faut noter que les auteurs dans [13] ont utilisé leur algorithme déterministe de diffusion de message pour construire un algorithme déterministe de commérage qui travaille en temps $O(n^{3/2} \log^2 n)$ dans le même modèle.

A2. Communication asynchrone

Dans la communication asynchrone, la durée de toutes les actions du système est finie mais indéterminée, c'est-à-dire qu'on ne peut pas prédire la durée des actions par les stations. En particulier, différentes transmissions peuvent avoir différentes durées.

Dans [11], les auteurs ont étudié l'asynchronie dans les réseaux radio avec les principaux ingrédients de leur approche comprenant (1) l'abstraction d'une exécution en tant que séquence de transmissions simultanées appelées

événements, et (2) l'imposition des restrictions sur les événements dans la même exécution pour capturer des propriétés supplémentaires du modèle.
Les exécutions possibles d'un protocole de diffusion de message sont définies à l'aide du type d'adversaire. Ils considèrent trois adversaires spécifiques, appelés arc, nœud et arrêt. L'adversaire arc est le plus puissant par ses délais de livraison de messages qui peuvent être indépendants pour chaque arc. L'adversaire arrêt est défini comme l'adversaire arc augmenté avec le pouvoir de bloquer certains des nœuds avant le début d'une exécution. L'adversaire nœud est sujet à la contrainte que toutes les livraisons de copies d'un message généré par une seule transmission d'un nœud se produisent simultanément.
Les auteurs montrent comment trouver un protocole de diffusion asynchrone pour un réseau donné. La complexité en travail du protocole obtenu, mesurée comme le nombre total de transmissions, peut être exponentielle. Ils montrent ensuite que c'est une propriété inhérente du modèle en prouvant des bornes inférieures exponentielles pour chacun des adversaires considérés. Le problème de vérifier si un protocole donné est correct contre soit l'adversaire arc soit l'adversaire arrêt se révèle résolu polynomialement. Le problème de décider s'il existe un protocole (avec une complexité de travail donnée) qui est correct contre l'adversaire arc est montré être NP-complet. Une différence notable entre les adversaires arc et nœud est que, bien que la solution contre l'adversaire arc pour un réseau et un protocole donnés soit polynomialement vérifiable, ce problème pour l'adversaire nœud est montré être complet en co-NP.

Les auteurs dans [5] ont considéré deux types de protocoles pour la diffusion asynchrone de message dans les réseaux radio : non-adaptatif et adaptatif. Dans un protocole non-adaptatif, chaque nœud doit envoyer tous ses messages dès qu'il est réveillé par le message source. En revanche, un protocole adaptatif est plus puissant, car un nœud peut décider du nombre et du contenu des

messages qu'il envoie, selon la séquence de messages reçus jusqu'à présent. Dans ce modèle, les auteurs ont utilisé le travail pour mesurer la complexité de leurs algorithmes. Le travail est défini comme le nombre total de transmissions de messages dans l'exécution d'un protocole dans le pire des cas. Ils ont obtenu les résultats suivants. Pour les réseaux *Graphe de Disque Unitaire* (UDG) (où les stations sont représentées par des points dans le plan euclidien et deux nœuds sont voisins si la distance euclidienne entre eux est au plus 1) avec une topologie connue, un résultat serré : le travail optimal est $\Theta(\tau)$, où τ est le nombre de blocs contenant au moins un nœud, les blocs formant une partition du plan en carrés disjoints de côté $1/\sqrt{2}$. Ce résultat tient à la fois pour les algorithmes non-adaptatifs et adaptatifs. Pour les réseaux UDG avec une topologie inconnue, les résultats changent considérablement et dépendent de la densité d qui est la distance euclidienne la plus petite entre deux nœuds quelconques. Si d est connue, alors le travail optimal dépend de τ et de la granularité $g = \frac{1}{d}$. Les auteurs ont montré d'une part, un algorithme de diffusion non-adaptatif avec le travail $O(\tau\alpha^{g^2})$, pour une certaine constante $\alpha > 1$. D'autre part, tout algorithme de diffusion, même adaptatif, doit utiliser le travail $\Omega(\tau\beta^{g^2})$, pour une certaine constante $\beta > 1$. Si d est inconnue, ils ont montré que la diffusion de message contre l'adversaire fort (advesaire nœud) est impossible dans les réseaux UDG.

Les auteurs ont ensuite présenté leurs résultats pour les réseaux modélisés par des graphes qui ne proviennent pas nécessairement de configurations de points dans le plan. Pour de tels graphes, ils supposent que tous les nœuds ont des étiquettes distinctes qui sont des entiers positifs et que chaque nœud connaît son étiquette. Dans les réseaux symétriques avec topologie connue, ils ont obtenu pour les algorithmes adaptatifs, le travail optimal de $\Theta(n)$, où n est le nombre total de nœuds, et pour les algorithmes non-adaptatifs une borne

supérieure de $O(2^n)$ et une borne inférieure de $\Omega(c^n)$ sur le travail, pour une certaine constante $c > 1$. Pour les réseaux arbitraires avec topologie connue, ils obiennent les mêmes résultats comme dans le cas des réseaux symétriques avec topologie connue pour les algorithmes non-adaptatifs. Finalement, pour les réseaux avec la topologie inconnue, le travail optimal est $\Theta(2^N)$, où N est l'étiquette maximale parmi les étiquettes des nœuds, et ce résultat ne dépend pas de la symétrie du graphe, et du type d'algorithme, adaptatif ou non-adaptatif.

B. Réveil

Le but de la tâche du réveil est d'activer (de réveiller) toutes les stations du réseau. Certaines stations se réveillent spontanément, peut-être dans différentes rondes, tandis que les autres stations doivent être réveillées par un message. Seules les stations déjà réveillées (actives) peuvent participer au processus. Une station en sommeil (non-active) se réveille automatiquement en entendant un message. Cela se produira lors de la première ronde réussie, à savoir la première ronde quand exactement une station voisine envoie un message.

Dans [9], pour un réseau radio modélisé par un graphe orienté, fortement connexe (c'est-à-dire qu'il existe un chemin orienté entre chaque paire de nœuds) $G = (V, E)$, avec $n = |V|$ nœuds, les auteurs ont montré l'existence d'un protocole déterministe qui accomplit le réveil en $O(n \log^2 n)$ rondes. Le meilleur précédent protocole déterministe pour le réveil a un temps de fonctionnement de $O(n^{3/2} \log n)$. Ce dernier a été développé dans [10]. Toutefois, ce résultat de $O(n \log^2 n)$ obtenu dans [9] a été amélioré en 2016 par Arthur Czumaj et Peter Davies dans [17]. Dans ce dernier article, les auteurs ont considéré un réseau radio inconnu dans lequel tous les nœuds ne connaissent pas la topologie du réseau mais connaissent seulement la grandeur n du réseau, le dégré maximum Δ des nœuds, et l'excentricité D du réseau. Pour un tel réseau, modélisé par

un graphe orienté ou non-orienté, ils ont donné un algorithme déterministe qui s'exécute dans le temps $O(\frac{\min(n,D\Delta)\log n \log \Delta}{\log\log\Delta})$, améliorant le meilleur résultat précédent de $O(n\log^2 n)$ dans tous les paramètres, mais surtout lorsque Δ est petit.

L'auteur dans [49] a étudié la tâche d'activation à partir d'un seul nœud (appelé source) dans un réseau radio anonyme à topologie inconnue, par un algorithme déterministe. Au début, seule la source est active et doit activer d'autres nœuds en diffusant des messages dans tout le réseau. Les nœuds du réseau ne connaissent pas sa topologie et ils n'ont pas d'étiquettes distinctes. Dans un tel réseau, certains nœuds sont impossibles à atteindre. Un nœud dans un réseau est accessible s'il peut être activé par un algorithme déterministe (éventuellement dépendant du réseau). L'auteur a tout d'abord montré que le problème de savoir si un nœud donné d'un réseau radio anonyme est accessible, peut être résolu en temps polynomial pour la communication synchrone. Ensuite pour la communication synchrone, l'auteur a conçu un algorithme déterministe universel d'activation (c'est-à -dire un algorithme qui active tous les nœuds accessibles dans tous les réseaux), et pour la communication asynchrone, il a montré qu'un tel algorithme n'existe pas.

Pour un réseau radio à plusieurs canaux, avec n stations connectées à b canaux, sans détection de collisions, où chaque station peut transmettre directement à toutes les autres stations, les auteurs dans [7] ont étudié la tâche du réveil dépendamment du paramètre b. Chaque station utilise tous les canaux simultanément et indépendamment. Certaines k stations peuvent devenir actives spontanément pendant des rondes arbitraires, et joignent le processus de réveiller tout le réseau. L'objectif de réveiller le réseau se produit lorsque toutes les stations entendent pendant une ronde r une transmission réussie sur

un certain canal. La durée d'un algorithme de réveil est mesurée à partir de la ronde r_0 de la première activation spontanée jusqu'à la ronde r.Toutes les stations connaissent b. Le paramètre k est utilisé en même temps que b pour charactériser la performance de l'algorithme de réveil. Les auteurs ont construit des algorithmes non-adaptatifs dans le sens que les transmissions des stations sont programmées en avance. Les algorithmes déterministes non-adaptatifs sont déterminés par les décisions données à chaque station de transmettre ou non sur un canal donné pendant une ronde donnée, tandis que les algorithmes aléatoires non-adaptifs sont déterminés par les probabilités pour chaque station et chaque canal, si une station doit transmettre sur un canal pendant une ronde donnée. Si k est inconnu, les auteurs ont donné deux algorithmes déterministes non-adaptatifs : un algorithme déterministe général qui réveille le réseau en temps $O(k \log^{1/b} k \log n)$, et un algorithme déterministe qui travaille en temps $O(\frac{k}{b} \log n \log(b \log n))$ quand $b > \log(128b \log n)$. La performance de ce dernier algorithme est écartée du temps optimal par un facteur d'au plus $O(\frac{k}{b} \log n \log(b \log n))$, puisque $\Omega(\frac{k}{b} \log \frac{n}{k})$ rondes sont nécessaires pour tout algorithme déterministe. Néanmoins, cet algorithme est le meilleur parmi tous les algorithmes développés par les auteurs par les paramètres k et b.
Si k est connu, les auteurs ont donné un algorithme aléatoire qui réveille le réseau en $O(k^{1/b} \ln \frac{1}{\epsilon})$ rondes avec la probabilité qui est au moins $1 - \epsilon$, pour tout $0 < \epsilon < 1$.

C. Élection du chef

L'élection du chef est l'une des tâches fondamentales dans le calcul distribué. Le but est qu'à la fin de l'algorithme, exactement un nœud prenne la décision qu'il est le chef et que tous les autres nœuds décident qu'ils ne sont pas chefs. L'élection du chef est la façon ultime de briser les symétries dans un

réseau initialement inconnu. C'est l'opération primitive naturelle qui est utilisée comme première étape dans la résolution de beaucoup d'autres tâches de haut niveau nécessitant ou bénéficiant de la présence d'un «organisateur» désigné. En raison de son importance, l'élection du chef a été étudiée dans de nombreux réseaux et en utilisant différents modèles.

Dans [30], les auteurs ont étudié la tâche de l'élection du chef dans un réseau radio, où les nœuds fonctionnent en rondes synchronisées. Dans une ronde, un nœud peut soit transmettre un message de taille logarithmique à ses voisins soit écouter en demeurant silencieux. Seulement un nœud écoutant dans une ronde r reçoit un message dans cette ronde r si exactement un seul voisin transmet dans cette ronde r. Les nœuds qui écoutent dans une ronde r dans laquelle plus d'un voisin transmettent, reçoivent seulement une collision. Une telle collision peut-être détectée ou non à la réception, dépendamment du système. Les auteurs ont construit des protocoles qui élisent le chef dans presque le même temps T_{BC} que la diffusion d'un seul message. Pour l'élection du chef dans le réseau sans détection de collisions, leur algorithme aléatoire travaille en $O(D \log \frac{n}{D} + \log^3 n) \cdot \min\{\log\log n, \log \frac{n}{D}\}$ rondes, avec une très forte probabilité, sur tout réseau avec n nœuds, où D est le diamètre. Comme $T_{BC} = \Theta(D \log \frac{n}{D} + \log^2 n)$ rondes est une borne inférieure, leur borne supérieure est presque optimale jusqu'à un facteur au plus $\log\log n$ et un terme additionnel de $\log n$. Cet algorithme est en outre le premier algorithme de temps $O(n)$ dans ce modèle. Pour l'élection du chef dans le réseau avec détection de collisions (même avec des messages d'un seul bit), les auteurs ont donné un algorithme aléatoire avec une très forte probabilité, qui travaille en temps $O(D + \log n \log\log n) \cdot \min\{\log\log n, \log \frac{n}{D}\} = O(D + \log n) \cdot O(\log\log n)^2$. Cet algorithme est optimal à un facteur $\log\log n$ près et se compare bien à l'algorithme déterministe qui nécessite $\Theta(n)$ rondes et est indépendant de D.

D. Consensus

Les problèmes du consensus distribué sont fondamentaux en algorithmique répartie et proviennent de nombreuses applications pratiques. Les processeurs du réseau commencent chacun par des entrées dans un ensemble donné. Les processeurs sont sujets aux pannes. Les bons processeurs doivent atteindre le consensus en décidant tous la même valeur comme sortie, tout en satisfaisant les conditions suivantes :

1. Accord. Toutes les valeurs décidées par les bons processeurs doivent être identiques ;

2. Validité. Si tous les processeurs ont la valeur d'entrée v, alors la décision de tous les bons processeurs est v ;

3. Terminaison. Tous les bons processeurs participants prennent la décision.

Les auteurs dans [12] ont étudié le problème du consensus tolérant aux pannes dans les réseaux radio avec pannes de processeurs. Ce problème a été étudié dans les réseaux radio anonymes (sans identificateurs) ou non, à un seul saut, c'est-à-dire où chaque nœud est situé à portée de diffusion de tous les autres nœuds. Ils ont présenté une communication tolérante aux collisions, une communication dans laquelle chaque nœud peut perdre une partie arbitraire des messages envoyés par ses voisins au cours de chaque ronde. Pour gérer le manque de fiabilité dans les transmissions, ils ont augmenté les nœuds avec un détecteur de collisions. Ils ont montré entre autres que les identificateurs uniques ne facilitent pas le consensus à moins que l'espace des identificateurs possibles est plus petit que l'ensemble des valeurs étant décidées, que le consensus ne peut pas être résolu dans un réseau sans fil sans une certaine capacité de détection de collisions, et que le consensus peut être résolu efficacement (c'est-à-dire dans un nombre constant de rondes) si les stations sont équipées de détecteurs de collisions du côté récepteur.

3.1.2 Le modèle SINR

Dans le modèle SNIR, les stations sont modélisées par des points dans le plan euclidien. Il existe trois constantes dans le modèle SINR qui jouent un rôle : α est la perte relative d'un signal généralement considérée comme étant dans l'interval (2,6) ; β, un paramètre du réseau et des fonctions de codage ; et N, le terme du bruit ambiant. Le modèle est basé sur trois axiomes : (i) La diminution relative d'un signal est un polynôme de la distance parcourue, c'est-à-dire si v transmet à la puissance P_v, alors il est entendu à u avec une puissance $P_{uv} = \frac{P_v}{d(u,v)^\alpha}$, où $d(u, v)$ est la distance entre u et v. (ii) L'interférence s'ajoute. L'interférence totale au nœud u d'un ensemble S d'émetteurs est $\sum_{v \in S} P_{uv}$. (iii) Le succès d'une réception de signal est une fonction du seuil du rapport signal-interférence. À savoir, la transmission de v à w réussit si $P_{vw} \geq \beta \cdot (N + \sum_{u \in S} P_{uw})$, où S est l'ensemble des autres émetteurs qui transmettent simultanément avec v [34]. Généralement, on assume que tous les nœuds transmettent avec la même puissance. Nous présentons maintenant quelques tâches étudiées dans ce modèle.

La tâche de planification a été étudiée dans [36]. Plus précisément, étant donné un ensemble de n liens, chaque lien constitué d'une paire émetteur-récepteur, nous souhaitons partitionner les liens en un nombre minimum de rondes, chaque ronde satisfaisant des contraintes d'interférence permettant une transmission simultanée. Dans cet article, les auteurs donnent un algorithme distribué qui travaille en temps $O(\log n)$ pour la tâche de planification. Cet algorithme est basé sur un autre algorithme dans [41] pour la tâche de planification (dans un espace métrique arbitraire pour toute distance et une puissance de transmission monotone et sublinéaire) dont le résultat a été amélioré par un facteur logarithmique. Ce résultat correspondant à la meilleure borne supérieure

connue pour les algorithmes centralisés.

Pour la tâche de la multi-diffusion de messages, il y a k messages initiaux, potentiellement appartenant à différents nœuds, qui doivent être transmis à tous les n nœuds du réseau. Dans [52], les auteurs ont présenté une approche assez complète et rigoureuse de la tâche de la multi-diffusion dans le contexte du modèle SINR. Le diamètre du réseau est dénoté par D, le degré maximal par Δ et la granularité (la portée maximale de transmission multipliée par l'inverse de la distance minimale entre deux stations quelconques) par g. Les auteurs ont obtenu : (i) pour une connaissance totale de la topologie du réseau, un algorithme déterministe qui s'exécute dans $O(D + k \log \Delta)$ rondes et un algorithme sensible à la granularité qui effectue la diffusion multiple dans $O(D + k + \log g)$ rondes ; (ii) pour une connaissance seulement de leurs propres coordonnées et des coordonnées de leurs voisins par tous les nœuds, un algorithme déterministe avec une complexité de $O(D \log^2 n + k \log \Delta)$ rondes ; (iii) quand on donne aux nœuds, seulement leur propres coordonnées, un algorithme déterministe avec une complexité de $O((n + k) \log n)$ rondes ; (iv) si les nœuds connaissent seulement leurs propres étiquettes et les étiquettes de leurs voisins, à part la connaissance standard des paramètres n, k, D, Δ, aucun résultat déterministe n'est connu dans la littérature.

Dans [39], la diffusion de message est faite d'une part avec la connaissance de la densité locale, c'est-tà-dire chaque station connaît une borne supérieure sur le nombre de voisins immédiats, et d'autre part sans aucune connaissance du voisinage. Dans le premier cas, les auteurs ont construit un algorithme aléatoire qui s'exécute en temps $O(D + \log(1/\gamma))$ avec une probabilité d'au moins $1 - \gamma$, sur tout réseau de n nœuds avec pour diamètre D, et avec une puissance de transmission uniforme. Dans le deuxième cas, ils ont donné

un algorithme aléatoire en complexité de temps $O(D + \log(1/\gamma) \log n)$ avec une probabilité d'au moins $1 - \gamma$. Les auteurs dans [40] ont considéré deux autres aspects du réseau pour la diffusion de message : avec ou sans réveil spontané des stations. Ils donnent un algorithme aléatoire qui travaille en temps $O(D \log n + \log^2 n)$ avec le réveil spontané et un algorithme aléatoire qui travaille en temps $O(D \log^2 n)$ rondes sans le réveil spontané, tous les deux avec une forte probabilité, c'est-à-dire avec une probabilité au moins $1 - \frac{1}{n}$, où n est le nombre de stations.

Dans [35], les auteurs ont combiné les modèles absMAC et SINR pour marquer le début d'une étude systématique qui simplifiera le développement d'algorithmes pour le modèle SINR. Dans le modèle absMAC (*Abstract Medium Access Control*), il y a la garantie d'accomplissement (c'est-à-dire il existe une borne supérieure sur le temps pour un message de l'envoyeur d'atteindre tous ses voisins) et la garantie de progression (c'est-à-dire il existe une borne supérieure sur le temps pour un receveur de recevoir un certain message quand au moins un voisin transmet un message). En particulier, les auteurs montrent que la garantie de progression dans le modèle absMAC n'est pas efficace dans le modèle SINR pour les graphes de forte connectivityé $G_{1-\epsilon}$ qui contient des arcs entre les nœuds de distance au plus $(1 - \epsilon)$ fois la portée de transmission des nœuds, où $\epsilon > 0$ est une petite constante qui peut être choisie par l'utilisateur. La garantie de progression limite le temps jusqu'à ce qu'un nœud soit garanti de recevoir un message lorsqu'au moins un de ses voisins transmet. Pour surmonter cette limitation, ils ont introduit la notion légèrement plus faible de la progression approximative dans la spécification du modèle absMAC. Ils ont donné une implémentation rapide de la spécification modifiée, en fonction de la décomposition de l'algorithme en couches locales et globales. Ils analysent l'algorithme en termes de paramètres locaux tels que les degrés des nœuds,

plutôt que des paramètres globaux tels que le nombre total de nœuds. Une contribution majeure est leur démonstration qu'une telle analyse locale est possible même en présence d'interférences globales.

3.2 Communication par les bips

La communication par les bips utilise des messages très économiques. Les nœuds communiquent seulement en émettant un bip ou en n'émettant pas de bip (le silence). Un nœud qui émet un bip pendant une ronde r n'apprend rien des autres nœuds pendant cette ronde r. D'autre part, si un nœud est silencieux pendant une ronde t, il entend un bip pendant cette ronde t si au moins un voisin émet un bip pendant cette ronde t. La communication par les bips a été introduite dans le calcul distribué par Alejandro Cornejo et Fabian Kuhn dans [15]. Ce modèle est plus faible que la détection des collisions puisque les nœuds ne peuvent pas distinguer entre un seul bip et une collision de deux ou plusieurs bips. Dans ce modèle, nous allons étudier les tâches de la diffusion de message, du commérage, de l'élection du chef, de la coloration d'intervalles, du calcul de l'ensemble indépendant maximal (MIS), du rendez-vous, de nommer les canaux, de l'appartenance, de la résolution de conflit, de la détection de collisions, du calcul du degré d'un nœud, et de la 2-coloration.

3.2.1 Diffusion de message et commérage

Dans [16], le réseau est modélisé par un graphe non-orienté $G = (V, E)$, où $|V| = n$, D est son diamètre, L est la borne supérieure sur les étiquettes des nœuds, M est tel que tous les messages n'ont pas plus de $\log M$ bits, et où k est le nombre de sources dans le cas de la diffusion de plusieurs messages.

La communication est synchrone et les algorithmes suivants obtenus sont déterministes : (i) Un algorithme optimal pour la diffusion de message qui travaille en temps $O(D + \log M)$; (ii) Un algorithme pour le commérage qui travaille en temps $O(n \log LM)$; (iii) un algorithme pour la diffusion de plusieurs messages qui travaille en temps $O(k \log \frac{LM}{k} + D \log L)$ avec provenance (c'est-à-dire chaque nœud doit apprendre toutes les paires (identité de la source s_i, message de la source s_i), où $1 \leq i \leq k$), et une borne inférieure $\Omega(k \log \frac{LM}{k} + D)$ sur le temps de résolution de ce problème ; (iv) un algorithme pour la diffusion de plusieurs messages sans provenance (c'est-à-dire chaque nœud doit apprendre tous les messages sources) qui travaille en temps $O(k \log \frac{M}{k} + D \log L)$ si $M > k$ ou $O(M + D \log L)$ si $M \leq k$, et des bornes inférieures correspondantes de $\Omega(k \log \frac{M}{k} + D)$ si $M > k$ ou $\Omega(M + D)$ si $M \leq k$; (v) ces derniers algorithmes pour la diffusion de plusieurs messages impliquent des algorithmes de commérage qui travaillent en temps $O(n \log \frac{LM}{n} + D \log L)$ ou $O(n \log \frac{M}{n} + D \log L)$ avec ou sans provenance respectivement.

3.2.2 Élection du chef

Quand les nœuds n'ont aucune connaissance initiale du réseau à part leurs propres étiquettes qui sont toutes différentes et quand la communication est synchrone, l'élection du chef est effectué par un algorithme déterministe en $O(D \log n)$ rondes, où D est le diamètre du réseau avec n nœuds. Ce résultat a été obtenu dans [25].

Dans [32], les auteurs ont étudié les quantités de ressources informatiques (états des machines à états et/ou la probabilité de transition dans le cas probabiliste) nécessaires pour résoudre l'élection aléatoire du chef. Le réseau est connexe avec n nœuds et la communication avec les bips est synchrone. Dans ce modèle,

ils ont montré pour une erreur $\epsilon \in [0, 1/2]$ et une probabilité de transition $q \geq 2$, que tout algorithme qui garantit l'élection du chef avec la probabilité $1 - \epsilon$ nécessite $s = \Omega(\log_q(1/\epsilon))$ états. Si une borne inférieure $\tilde{N}$ sur la taille du réseau est donné, alors la borne inférieure se réduit à $s = \Omega(\log_q(1/\epsilon)/\tilde{N})$ états. Les auteurs ont ensuite montré que ces résultats sont serrés.

3.2.3 Autres problèmes dans la communication par les bips

La tâche de la coloration des intervalles se définit comme suit : étant donné un ensemble de ressources, l'objective de la coloration des intervalles est d'assigner à chaque nœud une grande fraction contiguë de ressources telle que les nœuds voisins ont des ressources disjointes. Une coloration de k intervalles est une coloration où chaque nœud reçoit une fraction d'au moins $1/k$ des ressources.

Dans [15], le réseau est modélisé par un graphe $G = (V, E)$ non-orienté, et pour un nœud $u \in V$, $d(u) = |N(u)|$ est son dégré, où $N(u) := \{v \in V | \{u, v\} \in E\}$ est l'ensemble des voisins de u. Les auteurs ont supposé que les nœuds se réveillent de manière asynchrone dans un ordre déterminé par un adversaire. Au réveil, un nœud ne connaît rien sur la topologie et n'a aucune estimation de la grandeur n ou du dégré maximum Δ du graphe G. En plus, les nœuds n'ont pas d'étiquettes uniques, ne connaissent pas leurs voisins et n'ont pas d'estimation de leur nombre. En bref, le graphe de communication G n'est pas limité de quelque manière que ce soit. Les auteurs ont comparé leurs résultats avec une variante continue de ce modèle dans [47]. Dans le modèle continu, quand un nœud u bipe pendant une ronde t, cette action met tout nœud $v \in N(u)$ en mode écoute pour une periode de temps infiniment courte, c'est-à-dire pour les prochaines $\delta \geq 1$ rondes. Les auteurs dans [15] ont montré un algorithme

qui travaille en temps $O(1)$ avec probabilité 1, qui produit une coloration de $O(\Delta)$ intervalles. Ceci est une diminution en temps de $O(\log n)$ avec les mêmes garanties présentées en [47], et met l'accent sur les hypothèses irréalistes du modèle continu. Sous le modèle discret plus réaliste (où le temps est divisé en periodes de longueur μ, où μ dépend des caractéristiques physiques des dispositifs et du support de la communication sans fil), ils ont présenté un algorithme Las Vegas (c'est un algorithme aléatoire qui produit toujours un résultat correct) qui résout la coloration de $O(\Delta)$ intervalles en temps $O(\log n)$ avec une forte probabilité. Enfin, pour des graphes à dégrés constants, ils ont établi une borne inférieure de $\Omega(\log n)$ sur le temps nécessaire pour résoudre la coloration de $O(\Delta)$ intervalles dans ce modèle pour les algorithmes aléatoires. Cette borne inférieure implique que leur algorithme est asymptotiquement optimal pour des graphes à dégrés constants.

Soit un graphe $G = (V, E)$. Un ensemble indépendant de G est un sous-ensemble I de V tel qu'aucune paire de nœuds de l'ensemble I ne sont voisins. Un ensemble indépendant I est maximal, denoté (MIS), si tout nœud de G est dans I ou est voisin d'un nœud de I. La tâche de la sélection distributive d'un MIS dans un réseau modélisé par un graphe non-orienté, où la communication par les bips se déroule en rondes synchrones, a été étudié en [1]. Dans ce modèle, les nœuds n'ont aucune connaissance de la topologie du réseau ou même d'une borne supérieure sur sa grandeur. En plus, il est supposé que les nœuds se réveillent de manière asynchrone. Les auteurs ont commencé par prouver une borne inférieure qui montre que dans ce modèle, il n'est pas possible de converger localement vers un MIS en temps sous-polynomial. Donc, ils ont étudié quatre relaxations du modèle : Si une limite supérieure polynomiale sur la taille du réseau est connue, il est possible de trouver un MIS en temps $O(\log^3 n)$ avec une très forte probabilité ; Si les nœuds dormants sont réveillés par les bips des

voisins, nous pouvons également trouver un MIS en temps $O(\log^3 n)$ avec une très forte probabilité ; Si, en plus de cette hypothèse de réveil, la détection de collisions est assumée, on peut trouver un MIS en temps $O(\log^2 n)$ avec une très forte probabilité ; Si enfin, les nœuds sont dotés d'horloges synchrones, il est possible de trouver un MIS en temps $O(\log^2 n)$ avec une très forte probabilité.

Un rendez-vous, c'est quand deux agents mobiles, qui partent de nœuds quelconques d'un réseau inconnu, doivent se rencontrer à un certain nœud de ce réseau. Dans [22], le réseau est modélisé par un graphe non-orienté, et les agents se déplacent en rondes synchrones, c'est-à-dire à chaque ronde, un agent peut soit rester au nœud actuel soit se déplacer vers l'un des nœuds voisins. Par conséquent, dans chaque ronde, un agent se trouve à un nœud spécifique. Les agents sont des entités mobiles avec une mémoire illimitée. Les agents ont des étiquettes différentes qui sont des nombres entiers positifs. Chaque agent connaît sa propre étiquette, mais pas l'étiquette de l'autre agent. Les agents ne connaissent pas la topologie du réseau. Ils ne connaissent pas le nœud de départ ou la ronde d'activation de l'autre agent. Ils ne peuvent pas marquer les nœuds visités de quelque manière que ce soit. Chaque agent apparaît à son nœud de départ au moment de son activation par l'adversaire. Les auteurs ont supposé que les arêtes incidentes pour un nœud v ont des étiquettes distinctes dans $\{0, ..., d-1\}$ où d est le degré de v. Ainsi, chaque arête non-orientée $\{u, v\}$ a deux étiquettes appelées ses numéros de ports en u et v. La numérotation de ports est locale, c'est-à-dire qu'il n'y a pas de relation entre les numéros de ports à u et v. Un agent entrant dans un nœud apprend le port d'entrée et le dégré du nœud . Les auteurs ont noté que, en l'absence de numéros de ports, le rendez-vous est généralement impossible, car tous les ports d'un nœud sont identiques pour un agent et l'adversaire peut empêcher l'agent de prendre une arête incidente sur le nœud actuel. Dans les formulations traditionnelles du

problème de rendez-vous, celui-ci s'effectue lorsque les agents arrivent au même nœud dans la même ronde. Les auteurs ont atteint un objectif plus exigeant, appelé rendez-vous avec détection : les agents doivent prendre conscience que la rencontre est accomplie, la déclarer et s'arrêter simultanément. Pour se faire, les auteurs ont étudié deux variantes du modèle de communication par les bips. Dans le modèle local de communication par les bips, un agent, à un nœud v, entend un bip dans la ronde r s'il écoute dans la ronde r et si l'autre agent se trouve au même nœud , c'est-à-dire au nœud v . Dans le modèle global de communication par les bips, un agent, à un nœud v, entend un bip fort dans une ronde r s'il écoute dans cette ronde r et si l'autre agent est au même nœud v et bipe, et il entend un bip léger dans une ronde t s'il écoute dans cette ronde t et si l'autre agent se trouve à un autre nœud , c'est-à-dire au nœud $u \neq v$, et bipe. En effet, la même force de réception du bip rendrait impossible pour un agent A d'informer l'autre agent B de la présence de A au même nœud , et donc le rendez-vous avec détection serait impossible. Les auteurs ont d'abord présenté un algorithme déterministe de rendez-vous avec détection, qui travaille, même pour le modèle local, en temps polynomial dans la taille du réseau et dans la longueur de l'étiquette plus petite (c'est-à-dire dans le logarithme de cette étiquette d'agent). Cependant, dans cet algorithme, les agents dépensent beaucoup d'énergie : le nombre de déplacements qu'un agent doit accomplir est proportionnel à la durée du protocole du rendez-vous. Il est donc naturel de demander si des agents avec énergie bornée, c'est-à-dire des agents qui peuvent faire au plus c déplacements, pour un nombre entier c, peuvent toujours avoir un rendez-vous avec détection aussi, dans des réseaux à grandeur bornée. Les auteurs ont montré que la réponse à cette question est positive, même dans le modèle local, mais cette capacité est à un coût très élevé : le temps du rendez-vous des agents avec énergie bornée est exponentiellement supérieur à

celui des agents sans restrictions. Toutefois, ils ont montré un algorithme pour le rendez-vous avec détection dans le modèle global qui fonctionne pour les agents avec énergie bornée (dans les réseaux de taille bornée) aussi vite que pour les agents sans restrictions.

La tâche de nommer un réseau est d'attribuer des noms aux nœuds du graphe associé de telle manière que les noms forment un ensemble contigu d'entiers positifs. Cette tâche a été étudiée dans [8] pour un réseau anonyme de n stations dans le modèle de la communication par les bips. Les nœuds n'ont pas d' étiquettes uniques, la communication est synchrone et quand une station bipe, alors toutes les autres stations détectent un bip. Dans ce modèle, les auteurs ont donné un algorithme Las Vegas pour nommer le réseau pour le cas où le nombre de nœuds n est connu et un algorithme Monte Carlo pour le cas où le nombre de nœuds n est inconnu. Un algorithme Monte Carlo est un algorighme aléatoire qui termine toujours, cependant le résultat peut être incorrect avec une certaine probabilité. Les algorithmes obtenus ont le temps $O(n \log n)$ rondes, le nombre de bits aléatoires utilisés $O(n \log n)$, et la probabilité d'erreur est plus petite que $\frac{1}{2}$.

Dans l'article [38], les auteurs ont étudié les complexités distribuées de la tâche d'appartenance et de la tâche de la résolution de conflits qui sont deux tâches fondamentales dans le modèle de la communication par les bips. Dans les deux tâches, il existe un ensemble de n nœuds, chaque nœud a une étiquette unique $i \in \{1, 2, ..., n\}$. Les nœuds actifs (participants) constituent un sous-ensemble A de nœuds avec $|A| \leq k \leq n$, k un nombre entier. Dans la tâche d'appartenance, chaque nœud doit trouver les étiquettes de tous les nœuds actifs. Dans la tâche de résolution de conflits, l'objectif est de laisser chaque nœud actif utiliser seul le canal de communication (sans collisions) au moins une

fois. La communication se déroule en rondes synchrones et tous les nœuds ont accès à une horloge globale. Les auteurs ont comparé les résultats de ce modèle de communication par les bips avec ceux du modèle de détection de collisions, où les nœuds reçoivent des retours d'information ternaires sur l'état du canal de communication : aucune station n'émet (au repos), exactement un nœud émet (succès), ou deux ou plusieurs stations émettent (collision). En revanche, dans le modèle de la communication par les bips, les stations n'ont que des retours d'information binaires : aucune station ne bipe (au repos), ou une ou plusieurs stations bipent (occupé). Les auteurs ont montré que les deux problèmes ci-dessus sont également difficiles (avec la complexité en temps $\Theta(k \log \frac{n}{k})$) dans le modèle de la communication par les bips. Cela contraste fortement avec le modèle traditionnel ternaire de détection de collisions, dans lesquel le problème d'appartenance est strictement plus difficile que la résolution des conflits (avec la complexité en temps $\Omega(k \log_k n), O(k \log \frac{n}{k})$ pour l'algorithme déterministe, et $\Theta(k)$ pour l'algorithme aléatoire). Les algorithmes aléatoires ont tous une probablité de succès constante λ, où $0 < \lambda < 1$, dans le pire des cas.

Dans [45], les auteurs ont étudié les problèmes de détection de collisions, du calcul du dégré d'un nœud, de la coloration et de la 2-coloration, dans un réseau sans fil avec n processeurs, modélisé par un graphe connexe $G = (V, E)$, où les nœuds de V représentent les processeurs et les arêtes de E représentent les paires de processeurs qui communiquent entre eux avec des bips. Le réseau est anonyme, les transmissions sont divisées en rondes synchrones et tous les processeurs commencent en même temps. Ces problèmes ont été étudiés dans plusieurs variations du modèle de la communication par les bips. Deux types de collisions peuvent survenir du point de vue d'un nœud v de G :

- v bipe et simultanément au moins un voisin de v bipe (collision interne) ;
- au moins deux différents voisins de v bipent (collision périphérique).

Si un processeur bipe, il y a deux cas :

- le processeur ne sait pas si un autre processeur a bipé simultanément (cas utilisé dans [15]). Ce cas est dénoté B ;
- il peut distinguer s'il a bipé seul ou si au moins un voisin a bipé simultanément, c'est la collision interne. Ce cas est appelé la détection de collisions du côté expéditeur dans [1] et est dénoté B_{cd} dans cet article.

Si un processeur écoute, il y a aussi deux situations :

- il peut distinguer entre le silence et la présence d'au moins un bip (situation utilisée dans [15]). Ce cas est dénoté L ;
- il peut distinguer entre le silence, la présence d'exactement un bip, et la présence d'au moins deux bips ; dans ce cas, c'est la collision périphérique (qui a été étudiée dans [1]), ceci est dénoté par L_{cd} dans ce papier.

Les auteurs ont obtenu les résultats suivants :

Pour la détection locale de collisions (soit dans l'ensemble $N(v)$ des voisins du nœud v) dans le modèle BL, ils ont montré deux algorithmes Monte Carlo, un qui travaille en temps $O(\log(\frac{1}{\epsilon}))$ avec une erreur de probabilité au plus ϵ, où $0 < \epsilon < 1$, et avec ϵ donné à chaque nœud, l'autre qui travaille en temps $O(\log n)$ avec une erreur de probabilité $o(\frac{1}{n^2})$, avec n donné à chaque nœud . Pour la détection globale de collisions (soit dans tout G) dans le modèle BL, ils ont aussi montré deux algorithmes Monte Carlo, un qui travaille en temps $O(\log(\frac{n}{\epsilon}))$ avec une erreur de probabilité au plus ϵ, et avec ϵ et n donnés à chaque nœud, l'autre qui travaille en temps $O(\log n)$ avec une erreur de probabilité $o(\frac{1}{n})$, avec n donné à chaque nœud .

Pour le problème de coloration, dans le modèle $B_{cd}L$, ils ont montré deux algorthmes Las Vegas, tous les deux avec une très forte probabilité. Si aucune information n'est donnée aux nœuds, l'algorithme s'exécute en temps $O(\log n + \Delta)$, où Δ est le dégré maximum des nœuds dans G. Si une borne supérieure

K sur le dégré maximum des nœuds dans G, est donnée à chaque nœud, l'algorithme travaille en temps $O(K(\log n + \log^2 K))$. Ces bornes supérieures pour le problème de coloration peuvent être comparées avec la borne supérieure dans [15], de $O(\Delta \log n)$ rondes pour un algorithme Monte Carlo avec une très forte probabilité, dans le modèle BL , où chaque nœud connaît son propre dégré et une borne supérieure sur Δ.

Pour le problème de la 2-coloration (une 2-coloration du graphe G est une coloration du carré de G, c'est-à-dire, le graphe avec l'ensemble des arêtes de G, dans lequel il y a une arête entre deux différents nœuds u et v si la distance entre u et v dans G est au plus 2), les auteurs ont montré un algorithme Las Vegas avec une très forte probabilté qui travaille en temps $O(\log n + \Delta^2)$ dans le modèle $B_{cd}L_{cd}$, dans lequel aucune information n'est donnée aux nœuds.

Pour le problème du calcul des dégrés (dont l'objectif est pour un algorithme d'afficher le dégré pour chaque nœud), les auteurs ont montré un algorithme Las Vegas avec une très forte probabilité, qui travaille en temps $O(\log n + \Delta^2)$ dans le modèle $B_{cd}L_{cd}$, dans lequel aucune information n'est donnée aux nœuds. Dans le modèle BL, ils ont montré deux algorithmes Monte Carlo pour le calcul des dégrés, un qui travaille en temps $O((\log n + \Delta^2)(\frac{n}{\epsilon}))$ avec une erreur de probabilité au plus ϵ, où ϵ et la grandeur n du graphe sont donnés à chaque nœud, et un autre qui travaille en temps $O((\log n + \Delta^2) \log n)$ avec une erreur de probabilité $o(\frac{1}{n})$, avec n donné à chaque nœud .

3.3 Tâches distribuées dans les autres modèles de communication

Dans cette section, nous considérons diverses tâches distribuées, telles que la diffusion de message, le commérage, le réveil, l'élection du chef, le consensus,

l'exclusion mutuelle et la sélection dans les modèles de communication avec fil.

3.3.1 Diffusion de message et commérage

Un réseau de communication point-à-point est généralement modélisé par un graphe non-orienté $G(V, E)$, où les nœuds représentent des processeurs, et où les arêtes représentent des liens bidirectionnels de communication. La communication elle-même est synchrone ou asynchrone. Un message élémentaire peut contenir seulement un nombre constant de bits et un nombre constant d'identités de nœuds. Les messages plus longs doivent être coupés en messages élémentaires avant la transmission. La complexité de communication d'un algorithme est le nombre total de messages (élémentaires) envoyés dans une exécution au pire des cas. Dans [2], les auteurs ont supposé que les processeurs ont des étiquettes uniques dans $V = \{1, 2, \cdots, |V|\}$, et que chaque processeur connaît initialement son étiquette et ceux de ses voisins, mais ne connaît pas toute la topologie du réseau. Les auteurs ont montré des résultats suivants qui sont valides pour les modèles synchrone et asynchrone. Ils ont montré que la diffusion de message nécessite $\Theta(|V|)$ messages si les messages de grandeurs non bornées sont comptés à coût unitaire, et si les messages de grandeurs bornées sont comptés, la diffusion de message nécessite $\Theta(|E|)$ messages. Les auteurs ont montré une borne serrée de $\Theta(\min\{|E|, |V|^{1+\frac{\Theta(1)}{\rho}}\})$ messages pour les messages de grandeurs bornées, dans un modèle intermédiaire dans lequel chaque nœud connaît la topologie du réseau dans un rayon de $\rho \geq 1$ à partir de lui-même.

La tâche du commérage a été étudiée dans [24] dans les réseaux avec n nœuds pour lesquels n est impair. Les auteurs ont utilisé le modèle de communication à coût linéaire, dans lequel le coût de la communication est proportionnel à la quantité d'information transmise. Les auteurs ont étudié

deux variantes du problème. Dans le modèle synchrone, les communications par paires de nœuds sont organisées en rondes et toutes les transmissions dans une ronde commencent en même temps. Dans le modèle asynchrone, une paire de nœuds peut commencer à communiquer alors que les transmissions entre d'autres paires sont en cours. Les auteurs ont montré les bornes inférieures sur la durée totale du commérage dans les deux modèles. La borne inférieure en temps est strictement plus grande que $(k+1)\beta + n\tau$, pour tout $\beta > 0$, tout $\tau > 0$, et tout $n = 2^k - 1, k \geq 3$, dans le modèle asynchrone. Les auteurs ont montré que cette borne inférieure est réalisable pour certaines valeurs impaires de n, mais qu'aucun algorithme de commérage ne peut atteindre cette borne pour $n = 2^k - 1$ nœuds, $k \geq 3$. Pour le modèle synchrone, les auteurs ont montré une borne inférieure en temps de $(\lceil \log_2 n \rceil + 1)\beta + n\tau$, pour tout n impair, tout $\beta \geq 0$ et tout $\tau \geq 0$. Ils ont montré que cette borne inférieure est serrée pour $n = 2^k, k \geq 3$, $\beta > 0$ et $\tau > 0$.

3.3.2 Réveil

La tâche distribuée du réveil de tous les n processeurs d'un système connexe de diffusion de message, a été étudiée dans [29]. Dans cet article, les auteurs ont analysé cette tâche dans les modèles synchrones global et local, avec l'hypothèse que n est connu ou inconnu des processeurs. Dans le modèle synchrone global, tous les processeurs ont accès à une horloge globale. Dans le modèle synchrone local, chaque processeur a une horloge locale qui commence quand le processeur se réveille. Les auteurs ont supposé que les processeurs se réveillent spontanément ou après avoir entendu un message, et seuls les processeurs déjà réveillés peuvent envoyer des messages sur le canal de communication à plusieurs accès. Des collisions de messages peuvent se produire

parce que le canal de communication est partagé par tous les processeurs. Par conséquent, les auteurs ont supposé que les processeurs (en mode écoute) entendent un message dans une ronde r si et seulement si exactement un processeur envoie un message dans cette ronde r. Si aucun processeur ou au moins deux processeurs envoient des messages dans une ronde t, alors les processeurs (en mode écoute) pendant cette ronde t n'entendent rien. Donc, ce modèle de communication est équivalent au modèle radio sans détection de collisions dans un graphe complet. Les auteurs ont obtenu les résultats suivants. Dans le modèle synchrone global, si n est connu de tous les processeurs, et dans le pire des cas (c'est-à-dire en supposant qu'un adversaire contrôle les temps de réveil des processeurs), les auteurs ont montré un algorithme déterministe de réveil qui travaille en temps exactement n, et un algorithme aléatoire qui réveille le système en temps $O(\log n \cdot \log(1/\epsilon))$, avec probabilité de succès d'au moins $1 - \epsilon$, pour une constante $\epsilon > 0$. Dans le modèle synchrone global, les auteurs ont construit un algorithme de réveil qui travaille en temps $2n$ dans le pire des cas, si n est inconnu. Dans le modèle synchrone local, si n est connu de tous les processeurs, les auteurs ont montré un algorithme déterministe de réveil qui travaille en temps $O(n^2 \log n)$, et un algorithme aléatoire qui réveille le système en temps $O(n \log(1/\epsilon))$, avec probabilité de succès d'au moins $1 - \epsilon$. Les auteurs ont montré que, même si n est connu, tout algorithme déterministe de réveil nécessite un temps d'au moins $(1 + \epsilon)n$. Ceci a établi un écart en efficacité entre les modèles synchrones local et global. Avec les hypothèses les plus faibles, à savoir dans le modèle synchrone local, sans connaissance de n, les auteurs ont présenté deux algorithmes de réveil. Le premier est aléatoire et réveille le système en temps $O(n^2 \log(1/\epsilon))$, avec probabilité de succès d'au moins $1 - \epsilon$. Le second est déterministe et travaille en temps $O(n^4 \log^5 n)$.

3.3.3 Élection du chef

L'élection du chef est l'un des problèmes fondamentaux du calcul distribué. Un seul nœud du réseau se déclare chef et chacun des autres nœuds du réseau doit déclarer qu'il n'est pas chef. Si les nœuds du réseau possèdent des étiquettes distinctes, l'élection d'un seul nœud signifie que tous les nœuds doivent afficher l'étiquette du chef élu. Si les nœuds du réseau sont anonymes, la tâche de l'élection du chef est formulée comme suit : chaque nœud v du réseau doit générer un chemin simple, qui est codé comme une séquence de nombres de ports, de telle sorte que tous ces chemins se terminent à un nœud commun qui est le chef.

Les auteurs dans [6] ont étudié la tâche de l'élection déterministe du chef dans le modèle de transmission sans échec de messages dans un réseau de communication point-à-point. Dans cet article, le réseau est modélisé par un graphe connexe $G = (V, E)$ où les nœuds dans $V(|V| = n)$ représentent les processeurs et les arêtes dans E représentent les canaux non-orientés de communication entre les nœuds. Les auteurs ont supposé que chaque nœud a une étiquette unique qui est un entier positif de grandeur $O(\log n)$ bits. Les auteurs n'ont supposé aucune connaissance globale du réseau, même pas la grandeur du réseau, sa taille ou une borne supérieure sur sa taille. Cependant, les auteurs ont supposé que chaque nœud est équipé d'une fonction de numérotation des ports des arêtes incidentes, qui lui permet d'identifier le canal sur lequel un message est reçu, ou le canal sur lequel il doit envoyer un message. Les auteurs ont aussi supposé que le système est complètement synchrone, c'est-à-dire, tous les processeurs commencent au même moment et que le temps est divisé en rondes synchrones. Pendant une ronde chaque processeur envoie des messages à (certains de) ses voisins, reçoit des messages de (certains de) ses voisins, et

effectue des calculs locaux. Dans ce modèle, les auteurs ont montré un algorithme déterministe qui effectue l'élection du chef en $O(D + \log n)$ rondes, où D est le diamètre du réseau, avec des messages de grandeur $O(1)$. Cet algorithme a donc une complexité de $O(D + \log n)$ bit-rondes (une bit-ronde est une ronde synchrone où l'on envoie des messages avec un seul bit). Ce résultat améliore en termes de complexité en bit-rondes, le meilleur résultat précédent qui a une complexité de $O(D \log n)$ bit-rondes, obtenu dans [51]. Dans [51], l'auteur, en supposant des messages de grandeur $O(\log n)$, a montré que $O(D)$ rondes sont suffisantes pour élire un chef dans les réseaux arbitraires.

La tâche de l'élection déterministe du chef a été aussi étudiée dans [28] dans un modèle légèrement différent de celui de [6]. Il s'agit dans [28] de l'élection déterministe du chef dans le modèle $LOCAL$, où dans chaque ronde, un nœud peut échanger tous ses messages avec tous ses voisins et peut effectuer tous ses calculs locaux. Le réseau est modélisé par un graphe connexe non-orienté. La topologie du réseau est inconnue et les nœuds n'ont pas d'étiquettes, mais les ports de chaque nœud sont arbitrairement et fixement étiquetés. les numéros de ports conjointement avec la topologie du réseau, peuvent créer des asymétries à exploiter pour l'élection du chef. Les auteurs ont considéré deux versions de la tâche de l'élection du chef (LE) : la *LE forte* dans laquelle exactement un chef doit être élu si cela est possible, autrement tous les nœuds doivent terminer en déclarant que l'élection du chef est impossible ; et la *LE faible* qui diffère de la *LE forte* en ce sens qu'aucune exigence sur le comportement des nœuds n'est imposée si l'élection du chef est impossible. Les auteurs dans [28] ont montré que le temps de l'élection du chef dépend de trois paramètres du réseau : son diamètre D, sa grandeur n et son niveau de symétrie λ qui, quand l'élection du chef est faisable, est la plus petite profondeur à laquelle un nœud a une vue unique du réseau. La vue à la profondeur t à partir d'un nœud v du graphe,

est l'arbre constitué de tous les chemins de longueur t qui ont pour origine v. Le temps de l'élection du chef dépend aussi de la connaissance ou non, par les nœuds, des paramètres D et n. Les auteurs ont montré le temps optimal de la *LE faible* qui est $\Theta(D + \lambda)$ rondes, si D ou n est connu des nœuds. Ce temps optimal ne change pas si D et n sont connus des nœuds. Si ni D ni n ne sont connus, alors même la *LE faible* est impossible. Pour la *LE forte*, les auteurs ont montré que le fait de connaître uniquement D est insuffisant pour élire un chef. Toutefois, si seulement n est connu, alors le temps optimal est $\Theta(n)$.

La tâche de l'élection déterministe du chef dans le modèle $LOCAL$ de communication a été aussi étudiée dans [33]. Les réseaux dans [33] sont des arbres anonymes et l'objectif des auteurs a été d'établir des compromis entre le temps alloué τ et la quantité d'information qui doit être donnée a priori aux nœuds pour permettre l'élection du chef en temps τ dans tous les arbres pour lesquels l'élection du chef est possible en ce temps τ. Cette information (une seule séquence) est fournie à tous les nœuds au début par un oracle connaissant l'arbre entier. La longueur de cette séquence est appelée la grandeur des conseils. Pour un temps donné τ alloué à l'élection du chef, les auteurs ont donné des bornes supérieure et inférieure sur la grandeur minimale des conseils suffisants pour effectuer l'élection du chef en temps τ. Pour la plupart des valeurs de τ, leurs bornes supérieure et inférieure sont soit serrées à des constantes multiplicatives près, soit elles ne diffèrent que par un facteur logarithmique. Soit T un arbre avec n nœuds, de diamètre $diam \leq D$, où D est la borne supérieure sur les diamètres. Alors que l'élection du chef dans le temps $diam$ peut être effectuée sans aucun conseil, pour le temps $diam - 1$, les auteurs ont montré des bornes supérieure et inférieure serrées de $\Theta(\log D)$. Pour le temps $diam - 2$, ils ont montré des bornes supérieure et inférieure de $\Theta(\log D)$ pour des valeurs paires de $diam$, et des bornes supérieure et inférieure serrées de $\Theta(\log n)$ pour des

valeurs impaires de *diam*. En passant à un temps plus court, dans l'intervalle $[\beta \cdot diam, diam - 3]$, pour une constante $\beta > 1/2$, les auteurs ont montré une borne supérieure de $O(\frac{n \log n}{D})$ et une borne inférieure de $\Omega(\frac{n}{D})$. Cette borne inférieure est valide pour tout *diam* de valeur impaire ou pour tout temps τ qui est au plus $diam - 4$. Par conséquent, avec l'exception du cas spécial lorsque la valeur de *diam* est paire et le temps τ est exactement $diam - 3$, leurs limites ne laissent qu'un écart logarithmique dans cet intervalle de temps. Enfin, pour le temps $\alpha \cdot diam$, pour n'importe quelle constante $\alpha < 1/2$ (sauf pour le cas de très petits diamètres), les auteurs ont montré des bornes supérieure et inférieure serrées de $\Theta(n)$.

3.3.4 Consensus et exclusion mutuelle

La tâche du consensus a été définie dans la sous-section 3.1.1.D. Le problème d'exclusion mutuelle concerne une situation où n utilisateurs ont besoin d'accéder à une resource qui ne peut être ni partagée ni divisée. Chaque utilisateur a un processeur qui agit comme son agent. L'utilisateur avec l'accès à la resource est dans la *région critique*. L'utilisateur qui n'a aucun rapport avec la resource est dans la *région reste*. Les processeurs communiquent entre eux par des variables partagées. Un algorithme d'exclusion mutuelle s'exécute de façon répartie par tous les processeurs réveillés. Chaque processeur exécute l'algorithme qui est divisé en actions suivantes : (i) *entrée (essai)* : la partie de l'algorithme exécutée en préparation pour entrer dans la région critique ; (ii) *critique* : la partie de l'algorithme qui protège contre l'exécution simultanée ; (iii) *sortie* : la partie de l'algorithme exécutée à la sortie de la région critique ; (iv) *reste* : le reste de l'algorithme. Chaque processeur exécute ces actions de façon cyclique dans l'ordre suivant : *reste*, *entrée*, *critique*, et *sortie*. Tout algorithme

d'exclusion mutuelle doit satisfaire les deux propriétés suivantes : (i) *exclusion* : dans chaque cycle de toute exécution, au plus un processeur se trouve dans la *région critique* (entre les actions *critique* et *sortie*) ; (ii) *pas de blocage* : dans toute ronde r de toute exécution, s'il y a un processeur dans la *région entrée* (entre les actions *entrée* et *critique*), alors un processeur entrera éventuellement dans la *région critique* après la ronde r. La complexité en temps d'un algorithme d'exclusion mutuelle est le nombre maximal de rondes à partir de la ronde du début de l'exécution de l'algorithme jusqu'à la ronde où il y a un certain processeur dans la *région entrée* (entre les actions *entrée* et *critique*) et il n'y a aucun processeur dans la *région critique*.

Les systèmes de communication avec tolérance aux pannes nécessitent souvent un moyen par lequel les bons processeurs peuvent arriver à un consensus. Les auteurs dans [48] ont étudié le problème de la faisabilité du consensus par un ensemble de processeurs (dont certains sont byzantins) qui ne communiquent que par des messages bilatéraux. Les auteurs ont montré que la tâche n'est faisable que si $n \geq 3m+1$, où m est le nombre de mauvais processeurs et n est le nombre total de processeurs. Ils ont montré que si les mauvais processeurs peuvent refuser de transmettre des informations, mais ne peuvent pas transmettre des informations fausses, la tâche est faisable pour $n \geq m \geq 0$ arbitraires. Cette hypothèse plus faible peut être approximée en pratique en utilisant des méthodes cryptographiques [48].

Dans [19], les auteurs ont étudié la faisabilité déterministe et la complexité en temps pour les tâches du consensus et d'exclusion mutuelle sans pannes, dans les réseaux où les processeurs ont des étiquettes différentes et communiquent à travers un canal à plusieurs accès, et où l'adversaire réveille certains processeurs dans des rondes possiblement différentes. Dans chaque ronde, chaque processeur

réveillé écoute ou transmet. Dans une ronde r, le message d'un processeur v est entendu par tous les autres processeurs réveillés si v est le seul processeur à transmettre dans cette ronde r. Si plus d'un processeurs transmettent simultanément, il y a une collision et aucun message n'est entendu. Les auteurs ont considéré trois caractéristiques qui peuvent exister ou ne pas exister dans le canal : la détection de collisions, la disponibilité d'une horloge globale, et la connaissance du nombre n de tous les processeurs. Les auteurs ont montré que le consensus et l'exclusion mutuelle sont impossibles si aucune des trois caractéristiques n'est disponible dans le canal, et si au moins l'une des trois charactéristiques est disponible, les deux tâches sont réalisables. Si la détection de collisions est disponible, les auteurs ont montré que les deux tâches peuvent être effectuées en temps optimaux $\Theta(\min(\log n, \log \alpha))$ pour le consensus avec les valeurs d'entrée dans $\{1, \cdots, \alpha\}$, $\alpha \geq 2$, et $\Theta(\log n)$ pour l'exclusion mutuelle. Pour les deux tâches, en l'absence de la détection de collisions, les auteurs ont montré d'une part, une même borne supérieure en temps $O(\min(n+t, n \log^2 n))$, où t est la plus grande ronde de réveil, et une même borne inférieure en temps $\Omega(n)$, si seulement l'horloge globale est disponible ; d'autre part, ils ont montré une même borne supérieure en temps $O(n \log^2 n)$, et une même borne inférieure en temps $\Omega(n)$, si seulement les processeurs connaissent n.

Le problème du *consensus avec décision simultanée* dans un système de communication synchrone avec pannes de processeurs a été étudié dans [46]. Ce problème nécessite que tous les bons processeurs décident de la même valeur (consensus) et que toutes les décisions soient prises au cours de la même ronde (simultanéité). Ainsi, il y a un accord double, un sur la valeur décidée (accord de données) et un sur la ronde de décision (accord de temps). Ce problème a d'abord été défini par les auteurs dans [21], qui l'ont analysé et l'ont résolu en utilisant une analyse de l'évolution des états de connaissances dans un système

sans pannes. Les auteurs dans [46] ont présenté un algorithme simple qui résout le consensus simultané dans le modèle avec au plus t pannes de processeurs. Le réseau est composé d'un ensemble fini de n processeurs $\Pi = \{p_1, p_2, \cdots, p_n\}$. Chaque paire de processeurs est reliée par un canal bi-directionnel fiable. Soit $f_i[r]$ l'ensemble des processeurs de qui p_i n'a pas reçu de message pendant la ronde r, soit $S[r]$ l'ensemble des processeurs qui ont survecu (c'est-à-dire complété) la ronde r, soit $C[r] = \bigcup_{p_i \in S[r]} f_i[r]$ l'ensemble des processeurs qui sont vus comme étant en panne par au moins un processeur qui a survecu la ronde r, et soit $D = \max_{r \geq 0}(d_r)$ où $d_r = \max(0, |C[r]| - r)$ est le nombre de rondes que l'adversaire a utilisé pour retarder la décision le plus longtemps possible. L'algorithme présenté dans [46] travaille pour chaque exécution en temps $(t+1) - D$ si $t < n-1$, et $t - D$ si $t = n-1$. Les auteurs ont montré que cet algorithme est optimal si $t < n-1$ et qu'il peut être facilement modifié pour être aussi optimal si $t = n-1$.

3.3.5 Sélection

La tâche de la sélection a été étudiée dans [54], dans le modèle de la communication avec un canal à plusieurs accès. L'auteur a défini le problème comme suit : N nœuds sont connectés à un canal de communication qui génère des signaux ternaires 0, 1 et e, dans les cas respectifs où zéro, un et au moins deux nœuds veulent diffuser sur le canal simultanément. Si Q des ces N nœuds veulent diffuser sur le canal simultanément, alors l'objectif du protocole de sélection est d'émettre le signal 1 le plus rapidement possible. Selon l'auteur, ce problème est précédemment résolu en temps $O(1)$ si Q est connu d'avance par tous les nœuds participants. Il a donc étudié ce problème pour le cas où Q n'est pas initialement connu des processeurs participants. Il a montré un protocole

de sélection qui travaille en temps $\log\log N + O(1)$, et un second protocole qui travaille en temps $\log\log Q + o(\log\log Q)$ pour le cas légèrement différent où N n'est pas connu des processeurs participants. En plus, il a montré une borne inférieure en temps $\log\log N - O(1)$ qui démontre que le premier protocole est optimal à une constante additive près, et que le second protocole ne s'écarte de l'optimalité que par une quantité additive $o(\log\log Q)$.

Chapitre 4

Diffusion asynchrone avec bips bivalents

4.1 Introduction

4.1.1 Le contexte

La diffusion est une tâche de communication fondamentale dans les réseaux. Un nœud d'un réseau, appelé la source, a un message que tous les autres nœuds du réseau doivent apprendre. Nous étudions des algorithmes déterministes pour cette tâche, qui a fait l'object de plusieurs recherches, dans un modèle très faible de communication sans fil. Les seuls signaux envoyés par les nœuds sont des bips. En outre, ils sont livrés aux voisins du nœud bipant d'une manière asynchrone : le temps entre l'envoi et la réception est fini mais imprévisible. Nous observons d'abord que dans ce scénario, aucune communication n'est possible, si les bips sont tous de la même force (voir section 4.2). Par conséquent, nous étudions la diffusion de message dans le modèle de communication avec les bips bivalents, où chaque bip peut être soit faible soit fort. À la réception, si exactement un bip

faible est reçu par un nœud dans une ronde, il est entendu comme faible. Toute autre combinaison de bips reçus (un seul bip fort ou plus d'un bip de toute sorte) dans une ronde est entendue comme un bip fort. Cette façon de modéliser la réception correspond à un seuil dans le dispositif d'écoute : la puissance d'un seul bip faible est inférieure au seuil, et la puissance d'un bip fort ou la puissance combinée de plus d'un bip est supérieure au seuil. Notre objectif est d'étudier comment la combinaison de deux faiblesses du modèle de communication, des messages très simples et courts d'une part, et la manière de livraison asynchrone de bips d'autre part, influence l'efficacité de la communication.

4.1.2 Le modèle

La communication se déroule en rondes. Dans chaque ronde, un nœud peut écouter (c'est-à-dire rester silencieux), envoyer un bip faible, ou envoyer un bip fort. Pour tout bip envoyé par n'importe quel nœud, un adversaire omniscient asynchrone choisit un entier non négatif t et le livre le bip à tous les voisins du nœud émetteur t rondes plus tard. Le délai de livraison à tous les voisins est le même pour un bip donné, mais peut être différent pour les bips différents du même nœud et pour les bips de nœuds différents. La seule règle à laquelle l'adversaire doit obéir pour la livraison de bips différents envoyés par le même nœud, est que les bips doivent être livrés dans le même ordre qu'ils ont été envoyés et ne peuvent pas être combinés lors de la livraison, c'est-à-dire que deux bips ne peuvent être livrès comme un bip. Ce type d'adversaire asynchrone a été appelé l'adversaire nœud dans [11] et l'adversaire fort dans [5]. La motivation est semblable à celle de [11, 5]. Les nœuds exécutent simultanément le protocole de diffusion avec d'autres tâches. Les bips à envoyer par un nœud sont préparés pour la transmission (stockés), puis chaque bip (faible ou fort) est transmis

dans l'ordre. L'écart (inconnu) entre ces actions est décidé par l'adversaire. Dans notre terminologie, le stockage pour la transmission correspond à l'envoi et la transmission réelle correspond à la livraison simultanée à tous les voisins. Nous supposons que pour de courtes distances entre les nœuds, le temps de déplacement du bip est négligeable. Le délai entre le stockage et la transmission (dans notre terminologie, entre l'envoi et la livraison) dépend de l'occupation du nœud avec d'autres tâches de calcul simultanément exécutées.

Le réseau est modélisé par un graphe connexe simple non-orienté avec n nœuds, appelé simplement graphe. Nous utilisons les termes ≪réseau≫ et ≪graphe≫ de façon interchangeable. Nous considérons quatre niveaux de connaissances que les nœuds peuvent avoir sur le réseau :

1. réseaux anonymes : les nœuds n'ont pas d'étiquettes et ne connaissent rien sur le réseau ;
2. réseaux ad-hoc : tous les nœuds possèdent des étiquettes distinctes et chaque nœud connaît seulement sa propre étiquette ;
3. Réseaux avec connaissance du voisinage : tous les nœuds ont des étiquettes distinctes, et chaque nœud connaît son étiquette et les étiquettes de tous ses voisins ;
4. Réseaux avec connaissance totale : tous les nœuds possèdent des étiquettes distinctes, chaque nœud connaît toute la carte étiquetée du réseau et l'étiquette de la source.

Les messages à diffuser appartiennent à un ensemble de taille M, appelé espace de messages. Sans perte de généralité, soit l'espace de messages l'ensemble des entiers $0, \cdots, M-1$. Sauf pour les réseaux anonymes, tous les nœuds ont des étiquettes différentes appartenant à l'ensemble des entiers $0, \cdots, L-1$, appelé espace d'étiquettes.

Le coût d'un algorithme de diffusion est le nombre total de bips envoyés par tous les nœuds. Cela mesure (l'ordre de grandeur de) la consommation d'énergie par le réseau, car l'énergie utilisée pour envoyer un bip fort peut être considérée comme un multiple constant de celle utilisée pour envoyer un bip faible.

4.1.3 Les résultats

Les résultats de ce chapitre ont été publiés dans l'article :
K. Hounkanli et A. Pelc,
Asynchronous Broadcasting with Bivalent Beeps,
Proc. 23rd International Colloquium on Structural Information and Communication Complexity (SIROCCO 2016), 291-306.

Le tableau suivant montre les bornes supérieures et inférieures sur le coût de la diffusion dans chacun des quatre scénarios considérés :

Niveau de connaissance des processeurs	Coût	
	Borne inférieure	Borne supérieure
Réseaux anonymes	Diffusion impossible	
Réseaux Ad-hoc	$\Omega(2^L)$	$2^{O(L+M)^2)}$
Réseaux de connaissance du voisinage	$\Omega(n \log M + n \log\log L)$	$O(n \log M + e \log\log L)$
Réseaux de connaissance totale	$\Omega(n \log M)$	$O(n \log M)$

FIGURE 4.1 – Le coût de la diffusion de message

4.2 Préliminaires

L'observation suivante montre que la diffusion asynchrone avec les bips de force uniforme est impossible même dans des graphes très simples. C'est la

raison pour laquelle nous utilisons le modèle de communication avec les bips bivalents.

Proposition 4.2.1. *La diffusion asynchrone utilisant des bips de force uniforme est impossible même dans le graphe avec deux nœuds.*

Démonstration. Considérons deux messages sources, m_1 et m_2, qui doivent être transmis d'un nœud à l'autre dans le graphe avec deux nœuds. Supposons que la source envoie k_1 bips pour le message m_1 et k_2 bips pour le message m_2, où $k_1 \leq k_2$, sans perte de généralité. L'adversaire livre les bips pour le message m_1 dans les rondes consécutives $r, r+1, \ldots, r+k_1-1$. Supposons que s est la ronde dans laquelle le nœud récepteur décode correctement le message m_1. Ensuite, pour le message m_2, l'adversaire livre k_1 bips dans les rondes $r, r+1, \ldots, r+k_1-1$, et les k_2-k_1 bips restants dans les rondes $t+1, \ldots, t+k_2-k_1$, où $t = \max(s, r+k_1-1)$. Cependant, dans la ronde s, le nœud récepteur a exactement la même information que pour le message m_1, et par conséquent, il affiche incorrectement le message comme m_1. □

Dans le reste de ce chapitre, nous utilisons le modèle de communication asynchrone par les bips bivalents, décrit plus haut.

4.3 Réseaux anonymes

Dans cette section, nous montrons que, si les nœuds n'ont pas d'étiquettes, la diffusion est impossible, même pour des graphes très simples, et même lorsque les nœuds connaissent la topologie du réseau.

Proposition 4.3.1. *La diffusion dans les réseaux anonymes est impossible même dans le cycle de grandeur 4.*

Démonstration. Considérons le cycle anonyme de grandeur 4 et considérons un algorithme hypothétique de diffusion A. Pour convenance, nous étiquetons les nœuds a, b, c, d, dans l'ordre des aiguilles d'une montre. Ceci est uniquement pour l'argumentation négative : les nœuds n'ont pas accès à ces étiquettes. Supposons que le nœud a est la source. Notons que, pour toute exécution de l'algorithme A, les nœuds b et d envoient exactement les mêmes bips dans les mêmes rondes, car dans chaque ronde ils ont la même histoire : en effet, ils reçoivent les mêmes bips dans les mêmes rondes, ils sont identiques et exécutent le même algorithme déterministe. Soient m_1 et m_2 deux messages différents qui doivent être diffusés par la source. Considérons deux exécutions de l'algorithme A : l'exécution E_1 dans laquelle la source diffuse le message m_1 et l'exécution E_2 dans laquelle la source diffuse le message m_2. Soit s_1 la séquence des bips (faible ou fort) envoyés par b et d dans l'exécution E_1 et soit s_2 la séquence des bips envoyés par b et d dans l'exécution E_2. Soit k_1 la longueur de s_1 et soit k_2 la longueur de s_2, où $k_1 \leq k_2$ sans perte de généralité. Dans les deux exécutions, l'adversaire livre les bips consécutifs de b et de d dans les mêmes rondes. En conséquence, le nœud c n'entend que des bips forts : k_1 bips forts en exécution E_1, et k_2 bips forts en exécution E_2. Le choix des rondes de livraison des bits de b et d est le suivant. Dans l'exécution E_1, ce sont des rondes successives $r, r+1, \ldots, r+k_1-1$, à partir d'une certaine ronde r. Supposons que s est la ronde dans laquelle le nœud c affiche correctement le message m_1. Ensuite, en exécution E_2, l'adversaire livre les premiers k_1 bips de b et d dans les rondes $r, r+1, \ldots, r+k_1-1$, et les $k_2 - k_1$ bips restants dans les rondes $t+1, \ldots, t+k_2-k_1$, où $t = \max(s, r+k_1-1)$. Dans la ronde s, le nœud c a la même histoire dans les exécutions E_1 et E_2 : il a entendu un bip fort dans les mêmes rondes, dans ces deux exécutions. Par conséquent, dans l'exécution E_2, il affiche incorrectement le message m_1 dans la ronde s. □

4.4 Réseaux ad-hoc

Dans cette section, nous montrons que fournir des étiquettes distinctes aux nœuds rend la diffusion possible dans les graphes arbitraires, même si les nœuds n'ont aucune connaissance initiale du réseau, à l'exception de leurs propres étiquettes. Soit $\mathcal{N}$ dénote l'ensemble des entiers non-négatifs. Considérons la fonction $\varphi : \mathcal{N} \times \mathcal{N} \longrightarrow \mathcal{N}$ donnée par la formule $\varphi(x, y) = x+(x+y)(x+y+1)/2$. Ceci est une bijection avec la propriété $\varphi(x, y) \in O((x + y)^2)$. Intuitivement, c'est la "fonction serpent" qui arrange tous les couples d'entiers non-négatifs dans une séquence infinie.
L'idée de l'algorithme est la suivante. Après avoir décodé le message source y, un nœud v avec étiquette ℓ calcule un très grand nombre entier $\Psi(\ell, y)$ et envoie $\Psi(\ell, y)$ bips forts suivis de $\Psi(\ell, y)$ bips faibles. $\Psi(\ell, y)$ doit être un entier qui permet de décoder y de façon non-ambiguë. Il doit être si grand que tous les bips faibles reçus par un voisin w du nœud v (qui peuvent être combinés avec des bips faibles envoyés par d'autres voisins de w et reçus par w comme des bips forts) ne puissent changer le nombre de bips forts reçus par w de façon significative. De cette façon le nœud w pourra calculer $\Psi(\ell, y)$ et ensuite décoder y. Nous allons prouver que la fonction $\Psi(\ell, y) = 8^{\varphi(\ell,y)}$ répond à ces besoins.

L'algorithme suivant est exécuté par un nœud actif qui a pour étiquette ℓ. Au début, tous les nœuds sont actifs. La partie *Recevoir* est exécutée par n'importe quel nœud autre que la source. Son résultat est d'afficher le message source. Cette partie est ignorée par la source, car elle connaît le message. La partie *Envoyer* est exécutée par la source au début de l'algorithme, et elle est exécutée par tous les autres nœuds immédiatement après l'affichage du message source dans la partie *Recevoir*. Après l'exécution de la partie *Envoyer*, le nœud

devient non-actif.

Algorithme `Ad-hoc`

Partie 1. *Recevoir*

Attendre jusqu'à ce que le nombre de bips faibles reçus soit au moins $1/2$ du nombre de bips forts reçus.

Soit t le nombre de bips forts reçus, et soit z le plus grand entier tel que $8^z \leq t$.

Calculer l'unique couple d'entiers non-négatifs (x, y), tel que $\varphi(x, y) = z$.

Afficher y pour message source.

Partie 2. *Envoyer*

Calculer $\varphi(\ell, y)$, où y est le message source.

Envoyer $8^{\varphi(\ell,y)}$ bips forts, suivis par $8^{\varphi(\ell,y)}$ bips faibles. ◇

Le résultat suivant montre l'exactitude l'algorithme `Ad-hoc` estime son coût.

Théorème 4.4.1. *À la fin de l'exécution de l'algorithme* `Ad-hoc` *dans un graphe arbitraire, chaque nœud affiche correctement le message source. Le coût de l'algorithme est* $2^{O((L+M)^2)}$.

Démonstration. La preuve de l'exactitude est divisée en deux parties. Nous montrons d'abord qu'aucun nœud n'affiche le message source de manière incorrecte, puis nous montrons que chaque nœud affiche le message source en temps fini. Soit m le message source. La première partie de la démonstration est faite par contradiction. Supposons qu'un nœud affiche le message source de manière incorrecte. Soit r la première ronde dans laquelle cela se produit, et soit u le nœud avec étiquette ℓ, qui affiche incorrectement le message source dans la ronde r. Soient $u_1, \ldots, u_k$ les nœuds adjacents à u pour lesquels au moins un bip a été livré avant la ronde r, ordonnés selon l'ordre croissant de

leurs étiquettes $\ell_1, \ldots, \ell_k$. Comme u affiche le message source dans la ronde r, l'ensemble de nœuds $\{u_1, \ldots, u_k\}$ n'est pas vide. En plus, tous les nœuds $u_1, \ldots, u_k$ devraient avoir affiché le message source avant la ronde r (parce qu'ils ont déjà envoyé quelques bips avant la ronde r), et donc l'afficher correctement. Soit $1 \leq i \leq k$ le plus grand entier j, tel qu'au moins un bip faible du nœud u_j était livré avant la ronde r. Supposons que t était le nombre de bips forts entendus par u avant la ronde r. Comme u a affiché le message source incorrectement, le plus grand entier z', tel que $8^{z'} \leq t$, ne peut pas être égal à $z = \varphi(\ell_i, m)$. (Si c'était le cas, le nœud u devrait correctement calculer le message source m puisque φ est une bijection.) L'entier z' ne peut être plus petit que z parce que le nœud u a entendu au moins 8^z bips forts envoyés par le nœud u_i. Donc $z' \geq z + 1$. Ceci indique que le nœud u devrait avoir entendu au moins 8^{z+1} bips forts avant la ronde r. Combien de bips faibles devrait-il avoir entendu avant la ronde r ? Tous ces bips pourraient venir seulement des nœuds $u_1, \ldots, u_i$. Le nombre total de bips faibles envoyés par ces nœuds est $\sum_{j=1}^{i} 8^{\varphi(\ell_j, m)}$. Comme $\varphi(\ell_1, m) < \varphi(\ell_2, m) < \cdots < \varphi(\ell_i, m)$, nous avons $\sum_{j=1}^{i} 8^{\varphi(\ell_j, m)} < \frac{8}{7} \cdot 8^{\varphi(\ell_i, m)} = \frac{8}{7} \cdot 8^z$. D'autre part, le nombre total de bips faibles entendus par le nœud u avant la ronde r devrait être au moins $1/2$ du nombre total de bips forts qu'il a entendus avant la ronde r. Ceci implique $\frac{8}{7} \cdot 8^z \geq \frac{1}{2} \cdot 8^{z+1}$, qui est une contradiction, et complète la première partie de la démonstration.

Nous montrons maintenant que chaque nœud affiche le message source en temps fini. Cette partie de la démonstration est aussi faite par contradiction. Supposons qu'un node n'a jamais affiché le message source. Dès lors que la source elle-même connaît le message source, et que le graphe est connexe, il doit exister des nœuds adjacents u et v, tels que u affiche le message source en un temps fini et v ne l'affiche pas. Soient $v_1, \ldots, v_s$ les nœuds adjacents de

v qui ont envoyé au moins un bip, ordonnés dans l'ordre croissant de leurs étiquettes $\lambda_1, \ldots, \lambda_s$. L'ensemble des nœuds $\{v_1, \ldots, v_s\}$ n'est pas vide. Nous montrons qu'à un moment donné, le nombre de bips faibles entendus par v est au moins $1/2$ du nombre de bips forts entendus par v. En effet, supposons que cela ne s'est pas produit avant que tous les bips de tous les nœuds $v_1, \ldots, v_s$ sont livrés. Le nombre de tous les bips émis par les nœuds $v_1, \ldots, v_{s-1}$ est $2 \cdot \sum_{j=1}^{s-1} 8^{\varphi(\lambda_j, m)}$. Comme $\varphi(\lambda_1, m) < \varphi(\lambda_2, m) < \cdots < \varphi(\lambda_s, m)$, nous avons $2 \cdot \sum_{j=1}^{s-1} 8^{\varphi(\lambda_j, m)} < \frac{2}{7} \cdot 8^{\varphi(\lambda_s, m)}$. Dans le pire des cas, ces bips peuvent être livrés par l'adversaire simultanément avec le même nombre (moins que $\frac{2}{7} \cdot 8^{\varphi(\lambda_s, m)}$) de bips faibles envoyés par le nœud v_s, alors produisant des bips forts entendus par le nœud v. Ceci diminuerait le nombre de bips faibles entendus par v et augmenterait le nombre de bips forts entendus par ce nœud, mais le changement ne peut pas être très grand. Effectivement, cela donne moins de $\frac{9}{7} \cdot 8^{\varphi(\lambda_s, m)}$ bips forts entendus par v. D'autre part, le nœud v entend au moins $\frac{5}{7} \cdot 8^{\varphi(\lambda_s, m)}$ bips faibles envoyés par v_s et laissés intacts (non livrés simultanément avec d'autres bips) par l'adversaire. Donc le nombre de bips faibles entendus par le nœud v est au moins $1/2$ du nombre de bips forts qu'il a entendus. Il s'en suit que le nœud v affiche le message source contrairement à notre supposition. Ceci complète la démonstration de l'exactitude de l'algorithme `Ad-hoc`. Nous estimons maintenant son coût.

Un nœud avec étiquette ℓ émet $2 \cdot 8^{\varphi(\ell, m)}$ bips, où m est le message source. Par conséquent le coût de l'algorithme `Ad-hoc` dans un réseau avec n nœuds est au plus $2n \cdot 8^{\varphi(L, M)}$. Puisque $\varphi(L, M) \in O((L + M)^2)$, et $\varphi(L, M) \geq L \geq n$, ceci donne le coût de $2^{O((L+M)^2)}$. □

Remarque. Notez que, si les nœuds connaissent l'un des paramètres, soit L soit M, alors la bijection φ peut être remplacée par une fonction un-à-un

plus efficace de $\{0, \ldots, L-1\} \times \{0, \ldots, M-1\}$ à des entiers non-négatifs. Par exemple, si L est connu, cette fonction peut être $\psi(\ell, m) = mL + \ell$, et si M est connue, cette fonction peut être $\psi'(\ell, m) = \ell M + m$. Les valeurs de ces fonctions sont de grandeur $O(LM)$, et par conséquent, si nous remplaçons l'une d'entre elles par φ, le coût de l'algorithme devient $2^{O(LM)}$.

Comme nous l'avons vu plus haut, le coût de l'algorithme `Ad-hoc` est très grand : même avec une connaissance de L ou M, il est exponentiel dans le produit de ces paramètres. Par conséquent, il est naturel de demander s'il existe, pour les réseaux ad-hoc, des algorithmes de diffusion qui ont des coûts polynomiaux en L et M. Notre prochain résultat montre que la réponse est négative. Avant de le montrer, nous rappelons une notion et un fait de [5].

Un ensemble S d'entiers positifs est *dominé* si, pour tout sous-ensemble fini T de S, il existe $t \in T$ tel que t est plus grand que la somme de tous les $t' \neq t$ dans T.

Lemme 4.4.1. *Soit S un ensemble fini dominé et soit k sa grandeur. Alors, il existe $x \in S$ tel que $x \geq 2^{k-1}$.*

Théorème 4.4.2. *Pour des entiers arbitraires $L \geq 4$, il existe des réseaux ad-hoc avec L nœuds, pour lesquels le coût de chaque algorithme de diffusion est $\Omega(2^L)$.*

Démonstration. Soit A un algorithme de diffusion. Pour tout ensemble $S \subseteq \{1, \ldots, L-2\}$, de grandeur au moins 2, le graphe G_S est défini comme suit. G_S a $|S| + 2$ nœuds avec des étiquettes dans l'ensemble $S \cup \{0, L-1\}$. Chacun des nœuds avec étiquette dans S est adjacent à chacun des nœuds avec étiquettes 0 et $L-1$, et il n'y a pas d'autres arêtes dans le graphe. Le nœud avec l'étiquette

0 est la source et le nœud avec l'étiquette $L-1$ est appelé l'*évier*. La figure ci-dessous est un exemple de graphe G_S pour $L = 7$ et $S = \{2, 3, 5\}$.

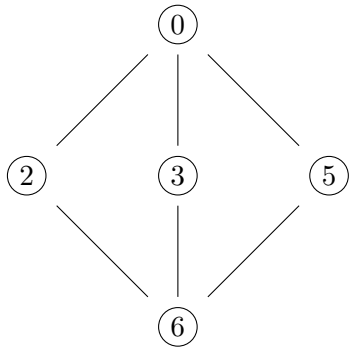

FIGURE 4.2 – Un exemple de graphe G_S pour $L = 7$ et $S = \{2, 3, 5\}$

Nous considérerons les exécutions de l'algorithme A dans les graphes G_S, dans lesquels l'adversaire obéit aux règles suivantes concernant la livraison des bips envoyés par la source et l'évier :

1. Tous les bips envoyés par la source après avoir entendu un bip, sont livrés après la ronde dans laquelle l'évier affiche le message source.
2. Tous les bips envoyés par l'évier sont livrés après la ronde dans laquelle l'évier affiche le message source.

Étant donné que nous considérons les réseaux ad-hoc, c'est-à-dire initialement chaque nœud connaît seulement sa propre étiquette, et l'adversaire obéit aux règles ci-dessus, le nombre de bips envoyés par un nœud avec une étiquette donnée $\ell \in \{1, \ldots, L-2\}$ avant la ronde dans laquelle l'évier affiche le message source, dépend uniquement de cette étiquette et du message source, et non du graphe G_S dans lequel l'algorithme est exécuté. En effet, l'histoire d'un nœud avec étiquette $\ell \in \{1, \ldots, L-2\}$, avant la ronde dans laquelle l'évier affiche le message source, est la même dans tous les graphes G_S, pour un message source donné m.

Considérons l'exécution de l'algorithme A dans le graphe $G_{\{1,\ldots,L-2\}}$, pour un message source fixé m. Soit $B(\ell)$, pour $\ell \in \{1,\ldots,L-2\}$, le nombre de bips des deux types que le nœud avec étiquette ℓ envoie avant la ronde dans laquelle l'évier affiche le message source. Si l'ensemble d'entiers $I = \{B(\ell) : \ell \in \{1,\ldots,L-2\}\}$ est dominé, alors par le Lemme 4.4.1, un entier dans cet ensemble est plus petit que 2^{L-3}, et nous avons terminé. Sinon, il existe un sous-ensemble $T \subseteq \{1,\ldots,L-2\}$, qui a la propriété suivante. Si $t \in T$ est tel que $B(t) \geq B(t')$, pour tout $t' \in T \setminus \{t\}$, alors $B(t) \leq \sum_{t' \in T \setminus \{t\}} B(t')$. Considérons l'exécution E de l'algorithme A dans le graphe G_T, pour le même message source m. Comme indiqué ci-dessus, le nombre de bips des deux types, que le nœud avec étiquette ℓ envoie dans cette exécution avant la ronde dans laquelle l'évier affiche le message source, est $B(\ell)$. L'adversaire livre les bips envoyés par les nœuds avec étiquettes dans T, en rondes consécutives, livrant simultanément un bip envoyé par le nœud avec étiquette t avec un ou plusieurs bips envoyés par les nœuds avec des étiquettes $t' \in T \setminus \{t\}$, de telle sorte qu'un seul bip n'est livré dans aucune ronde. Ceci est possible en raison de l'inégalité $B(t) \leq \sum_{t' \in T \setminus \{t\}} B(t')$. Par conséquent, l'évier entend seulement des bips forts. Considérons maintenant un message source différent m'. La même argumentation que ci-dessus montre que, si le coût de l'algorithme A pour le graphe $G_{\{1,\ldots,L-2\}}$ est plus petit que 2^{L-3}, alors il existe un ensemble $T' \subseteq \{1,\ldots,L-2\}$, tel que, pour l'exécution E' de l'algorithme A dans le graphe $G_{T'}$, avec le message source m', l'évier entend seulement des bips forts.

Supposons qu'au moment où il affiche le message source, l'évier entend k bips forts dans l'exécution E et entend k' bips forts dans l'exécution E'. Sans perte de généralité, supposons que $k \leq k'$. Le choix des rondes de livraison de ces bips par l'adversaire est le suivant. Dans l'exécution E, il s'agit de rondes consécutives $r, r+1, \ldots, r+k-1$, à partir d'une ronde r. Supposons que s est la ronde dans

laquelle l'évier affiche correctement le message m. Ensuite, dans l'exécution E', l'adversaire livre d'abord les bips dans les rondes $r, r+1, \ldots, r+k-1$, et les $k'-k$ rondes restantes de livraison de bips sont $z+1, \ldots, z+k'-k$, où $z = \max(s, r+k-1)$. Dans la ronde s, l'évier a la même histoire dans les exécutions E et E' : il a entendu seulement des bips forts, et cela s'est produit dans les mêmes rondes, dans ces deux exécutions. Par conséquent, dans l'exécution E', il affiche incorrectement le message m dans la ronde s.
La contradiction obtenue vient de l'hypothèse que le coût de l'algorithme A sur le graphe $G_{\{1,\ldots,L-2\}}$ est plus petit que 2^{L-3}, pour tous les messages source. Cela complète la démonstration. □

4.5 Réseaux avec la connaissance du voisinage

Dans cette section, nous supposons que tous les nœuds ont des étiquettes distinctes et que chacun d'eux connaît sa propre étiquette et les étiquettes de tous ses voisins. Cette apparente petite augmentation de la connaissance, par rapport aux réseaux ad-hoc (la connaissance de chaque nœud est toujours locale) diminue considérablement le coût de la diffusion. Afin de garantir un faible coût de la diffusion, nous devons coder les messages par des séquences de bips de manière très efficace. L'algorithme utilise des messages de deux types : des entiers non-négatifs et des triples d'entiers non-négatifs. Ces messages doivent être codés par des séquences de bips de longueur logarithmique en valeurs de ces entiers, de telle sorte que le destinataire sache quand la séquence commence et se termine, et puisse décoder sans ambiguïté le message de la séquence. Cependant, contrairement à l'algorithm `Ad-hoc` dans lequel les nœuds envoient des bips en nombres exponentiels, ce codage efficace est très vulnérable aux actions de l'adversaire qui peut intervertir arbitrairement les livraisons des bips

provenant de différents voisins d'un nœud. Pour éviter cela, nous concevons notre algorithme de telle sorte que les bips codant un message envoyé par un nœud sont livrés avant qu'un autre nœud commence à envoyer ses propres bips. De cette façon, le danger d'intervertir les bips est évité.

Avant de présenter l'algorithme, nous définissons le codage des entiers et de leurs triples, annoncé ci-dessus. Nous dénotons un bip fort par l, un bip faible par s, et nous utilisons le symbole $\cdot$ pour la concaténation des séquences de bips. Soit k un entier non-négatif, et soit $(c_1, \ldots, c_r)$ sa représentation binaire. Dénotons par $S(k)$ la séquence de $2r$ bips résultant de $(c_1, \ldots, c_r)$ en remplaçant chaque bit $c_i = 0$ par (ls) et en remplaçant chaque bit $c_j = 1$ par (sl). Le code d'un entier k, dénoté par $[k]$, est la séquence $(ll) \cdot S(k) \cdot (ll)$. Le code d'un triple d'entiers (a, b, c), dénoté $[a, b, c]$, est la séquence $(ll) \cdot S(a) \cdot (ss) \cdot S(b) \cdot (ss) \cdot S(c) \cdot (ll)$. Notons qu'une séquence de 2 bips forts marque le début et la fin d'un message, et que tous les messages contiennent un nombre pair de bips, logarithmique en entiers transmis. Un nœud à la réception peut déterminer le début du message comme la séquence σ de 2 bips forts consécutifs, et la fin du message comme la première séquence σ' de 2 bips forts consécutifs commençant après la fin de σ à une position impaire, où le premier bit de la séquence σ est à la position 1. Pour décoder le contenu du message $(ll) \cdot \alpha \cdot (ll)$, avec le début et la fin déjà identifiés correctement, un nœud recherche les séparateurs (ss) à partir de positions impaires de α. Il y a 0 ou 2 séparateurs de ce genre. Dans le premier cas, le message transmis était un entier et le nœud décode sa représentation binaire en remplaçant chaque couple (ls) par 0 et chaque couple (sl) par 1. Dans le second cas, le nœud peut représenter sans ambiguïté α comme $\alpha_1 \cdot (ss) \cdot \alpha_2 \cdot (ss) \cdot \alpha_3$, où chaque α_i a une longueur paire, et décode $\alpha_1, \alpha_2, \alpha_3$ comme ci-dessus.

En utilisant le codage ci-dessus, nous pouvons maintenant décrire notre

algorithme de diffusion. À un haut niveau d'abstraction, il est organisé comme un parcours en profondeur DFS ("Depth-first search") du graphe à partir de la source. Nous utiliserons les instructions "Envoyer $[a]$" et "Envoyer $[a, b, c]$" qui sont des procédures qui envoient les séquences de bips décrites ci-dessus, en rondes consécutives. Un message $[a]$, où $a \in \{0, 1, \ldots M-1\}$, est toujours le message source à diffuser. Il existe deux sortes de messages du type "triple d'entiers" : Pour $a, b \in \{0, 1, \ldots L-1\}$, un message de la forme $[a, b, 0]$ correspond à un message de parcours DFS en avant à partir du nœud avec étiquette a vers le nœud avec étiquette b, et un message de la forme $[a, b, 1]$ correspond à un message de parcours DFS en arrière, à partir du nœud avec étiquette a vers le nœud avec étiquette b.

L'algorithme est exécuté par un nœud avec étiquette ℓ. Les actions du nœud alternent entre l'exécution des instructions de la procédure "Envoyer" et l'écoute. L'algorithme est organisé de telle manière que la *propriété de disjonction* suivante soit satisfaite. Considérons un nœud u qui exécute certaines instructions $I(u)$ de "Envoyer". Soit $\sigma(u)$ le segment des rondes consécutives entre l'envoi du premier bit des instructions $I(u)$ et la livraison du dernier bit de ces instructions. Pour deux nœuds u et v, qui exécutent des instructions $I(u)$ et $I(v)$ de "Envoyer", les segments de rondes $\sigma(u)$ et $\sigma(v)$ sont disjoints. Cette propriété permet d'identifier les messages en circulation comme "paquets" distincts, et les utiliser pour implémenter un parcours DFS.

Quand un nœud écoute, il surveille le début et la fin d'un message formé par les bips reçus. Lorsqu'il détecte un message complet, il réagit de deux façons : soit il continue d'écouter et surveille un autre message complet soit il exécute des instructions de "Envoyer". Plus précisément, les actions du nœud avec étiquette ℓ, autre que la source, sont les suivantes. Après avoir reçu le

message source et le premier message DFS en avant $[a, \ell, 0]$ adressé à lui et qui provient du nœud avec étiquette a, le nœud avec étiquette ℓ commence par propager le message à tous ses voisins avec étiquettes a_i, sauf au voisin avec l'étiquette a. Il envoie le message source décodé $[m]$ et il envoie des messages DFS avant $[\ell, a_i, 0]$ à tous ses voisins avec étiquettes a_i, sauf au voisin avec l'étiquette a, dans l'ordre croissant de leurs étiquettes. Pour passer d'un voisin à l'autre, le nœud ℓ attend un message DFS en arrière $[a_i, \ell, 1]$ qui lui est adressé. En attendant, le nœud ℓ refuse tous les messages DFS en avant $[b, \ell, 0]$, pour $b \neq a$, qui lui sont adressés, en répondant par un message DFS en arrière $[\ell, b, 1]$. Les actions de la source sont similaires.

Le pseudocode de l'algorithme est le suivant.

Algorithme `Connaissance du voisinage`

si le nœud exécutant est la source, et le message source est m **alors**

$message \leftarrow m$

Soient $(a_1, a_2, \ldots, a_s)$ les étiquettes de tous les voisins du nœud,

dans l'ordre croissant

`Propager`$(a_1, \ldots, a_s)$

quand le message $[b, \ell, 0]$, pour un entier b, est décodé, alors

envoyer $[\ell, b, 1]$

sinon

quand le message $[m]$ est décodé pour la première fois, alors

$message \leftarrow m$

afficher $message$ comme message source

quand le message $[a, \ell, 0]$ est décodé pour la première fois, alors

soient $(a_1, a_2, \ldots, a_s)$ les étiquettes de tous les voisins du nœud,

à l'exception de a, dans l'ordre croissant

`Propager`$(a_1, \ldots, a_s)$

envoyer $[\ell, a, 1]$

quand le message $[b, \ell, 0]$, pour tout $b \neq a$, est décodé alors

envoyer $[\ell, b, 1]$ $\diamond$

La procédure `Propager`, utilisée par l'algorithme et exécutée par le nœud avec étiquette ℓ, est décrite comme suit.

Procédure `Propager`$(a_1, \ldots, a_s)$

envoyer $[message]$

$i \leftarrow 1$

tant que $i \leq s$ **faire**

envoyer$[\ell, a_i, 0]$

quand le message $[a_i, \ell, 1]$ est décodé alors

$i \leftarrow i + 1$ $\diamond$

Théorème 4.5.1. *À l'achèvement de l'algorithme* `Connaissance du voisinage` *dans un graphe arbitraire avec n nœuds et e arêtes, chaque nœud décode correctement le message source. Le coût de l'algorithme est $O(n \log M + e \log L)$.*

Démonstration. Compte tenu de la propriété de disjonction, tous les messages sont correctement décodés par leurs destinataires. Puisque les messages de contrôle $[a, b, 0]$ et $[a, b, 1]$ se déplacent dans un mode DFS, et chaque message $[a, b, 0]$ précède le message source $[m]$, tous les nœuds obtiennent le message source et le décodent correctement. Cela prouve l'exactitude de l'algorithme. Pour estimer son coût, notons que chaque nœud envoie le message source $[m]$ une fois, et, pour toute paire de nœuds adjacents a et b, deux messages de contrôle parmi $[a, b, 0]$, $[a, b, 1]$, $[b, a, 0]$ et $[b, a, 1]$ sont envoyés. Étant donné que le message source $[m]$ est constitué de $O(\log M)$ bips, et chaque message

de contrôle est constitué de $O(\log L)$ bips, le coût total de l'algorithme est $O(n \log M + e \log L)$. □

Avant de montrer la borne inférieure sur le coût des algorithmes de diffusion dans les réseaux avec la connaissance du voisinage, nous montrons les deux lemmes suivants.

Lemme 4.5.1. *Chaque algorithme de diffusion a un coût de* $\Omega(\log M)$ *dans un graphe avec deux nœuds.*

Démonstration. Supposons qu'il existe un algorithme de diffusion qui a un coût d'au plus $\frac{1}{2} \log M$ dans un graphe avec deux nœuds, où le nœud u est la source, et le nœud v est l'autre nœud du graphe.
Pour tout message source m, l'adversaire livre tous les bips envoyés par u en rondes consécutives. Étant donné qu'il y a moins de M différentes séquences binaires de longueur d'au plus $\frac{1}{2} \log M$, pour deux messages source différents, m_1 et m_2, les séquences de bips reçus par v doivent être identiques. Donc le message affiché par v doit être identique pour m_1 et m_2, et par conséquent ce message doit être incorrect dans l'un des deux cas. □

Lemme 4.5.2. *Chaque algorithme de diffusion a un coût de* $\Omega(\log \log L)$ *dans un cycle de grandeur 4.*

Démonstration. Considérons un algorithme A de diffusion qui fonctionne pour tous les cycles de grandeur 4, avec connaissance du voisinage. Supposons que le coût de l'algorithme A pour tous ces cycles est d'au plus $\frac{1}{2} \log \log L$. Considérons un cycle de grandeur 4, et dénotons ses nœuds a, b, c, d, dans le sens des aiguilles d'une montre. Supposons que le nœud a est la source. Soit 0 l'étiquette du nœud a, et soit $L - 1$ l'étiquette du nœud c. L'adversaire livre tous les bips

possiblement envoyés par le nœud c, seulement après que ce nœud affiche le message source. En conséquence, avant l'affichage du message source par le nœud c, les nœuds b et d entendent seulement les bips de la source a. L'adversaire livre tous les bips envoyés par le nœud a en rondes consécutives. Comme le nœud a peut envoyer au plus $\frac{1}{2}\log\log L$ bips, l'ensemble X de séquences possibles de bips entendus par nœuds b et d a la grandeur au plus $\sqrt{\log L}$. Soit $N = \{0, 1, \ldots, \lfloor \frac{1}{2}\log\log L \rfloor\}$. Comme chacun des nœuds b et d peut envoyer au plus $\frac{1}{2}\log\log L$ bips, le nombre de bips envoyés par chacun de ces nœuds doit être un entier de l'ensemble N. Pour toute étiquette $\ell \in \{1, \ldots, L-2\}$, soit $\Phi_\ell : X \longrightarrow N$ la fonction définie comme suit : $\Phi_\ell(x)$ est le nombre de bips envoyés par le nœud b ou d, s'il a pour étiquette ℓ, et s'il a reçu la séquence x de bips. Il existe $|N|^{|X|} < L-2$ de telles fonctions, pour L suffisamment grand. Donc il existe des étiquettes $\ell_1 \neq \ell_2$ de l'ensemble $\{1, \ldots, L-2\}$, pour lesquelles $\Phi_{\ell_1} = \Phi_{\ell_2}$. Assignons ces étiquettes aux nœuds b et d. Dans le cycle obtenu C, les nœuds b et d envoient le même nombre de bips, indépendamment de la séquence de bips obtenus de a. En particulier, cela se produira dans deux exécutions, E_1 et E_2, de l'algorithme A dans le cycle C, où l'exécution E_1 correspond au message source m_1, et l'exécution E_2 correspond au message source m_2, pour $m_1 \neq m_2$. Dans les deux exécutions l'adversaire livre les bips consécutifs de b et de d dans les mêmes rondes. Par conséquent, le nœud c entend seulement des bips forts : k_1 bips forts dans l'exécution E_1, et k_2 bips forts dans l'exécution E_2. Sans perte de généralité, supposons que $k_1 \leq k_2$. Le choix des rondes de livraison des bips de b et d est le suivant. Dans l'exécution E_1, elles sont des rondes consécutives $r, r+1, \ldots, r+k_1-1$, commençant à la ronde r. Supposons que s est la ronde dans laquelle le nœud c affiche correctement le message m_1. Dans l'exécution E_2, l'adversaire livre k_1 premiers bips de b et d dans les rondes $r, r+1, \ldots, r+k_1-1$, et les $k_2 - k_1$ bips restants dans les

rondes $t+1, \ldots, t+k_2-k_1$, où $t = \max(s, , r+k_1-1)$. Dans la ronde s, le nœud c a la même histoire dans les exécutions E_1 et E_2 : il a entendu un bip fort dans les mêmes rondes, dans les deux exécutions. En conséquence, dans l'exécution E_2, il affiche incorrectement le message m_1. □

Le résultat suivant donne une borne inférieure sur le coût de tout algorithme de diffusion dans les réseaux avec la connaissance du voisinage

Théorème 4.5.2. *Pour les entiers n arbitrairement grands, il existe des réseaux avec la connaissance du voisinage, avec n nœuds, dans lesquels chaque algorithme de diffusion a un coût de $\Omega(n \log M + n \log\log L)$.*

Démonstration. Pour tout entier k, considérons le graphe G_k défini comme suit. Soit P_k un simple chemin de longueur k, avec extrémités a et b. Considérons des copies disjointes $C_1, \ldots, C_k$ du cycle de grandeur 4 dont tous les nœuds sont différents des nœuds du chemin P_k. Soit a_i, b_i, c_i, d_i les nœuds de la $i^{\texttt{ème}}$ copie, ordonnés dans le sens des aiguilles d'une montre. Joindre le nœud a_1 au nœud b par une arête, et pour chaque $1 \leq i < k$, joindre le nœud c_i au nœud a_{i+1} par une arête, voir Figure 3.1 ci-dessous.

Le graphe obtenu a $n \in \Theta(k)$ nœuds.
Nous assignons les étiquettes aux nœuds du graphe G_k comme suit. Nous attribuons aux nœuds b_i et d_i dans les cycles C_i, pour $i = 1, \ldots, k$, des étiquettes distinctes par induction. Pour tout i, nous considérons toutes les étiquettes qui n'ont pas été utilisées précédemment et nous trouvons parmi elles deux étiquettes $\ell_1 \neq \ell_2$ pour lesquelles $\Phi_{\ell_1} = \Phi_{\ell_2}$, où Φ_ℓ, pour toute étiquette ℓ, est définie dans la démonstration du Lemme 4.5.2. Cela peut se faire de la même manière que dans la démonstration citée, car le nombre d'autres étiquettes encore disponibles est $\Theta(L)$. Finalement, nous attribuons aux nœuds du chemin

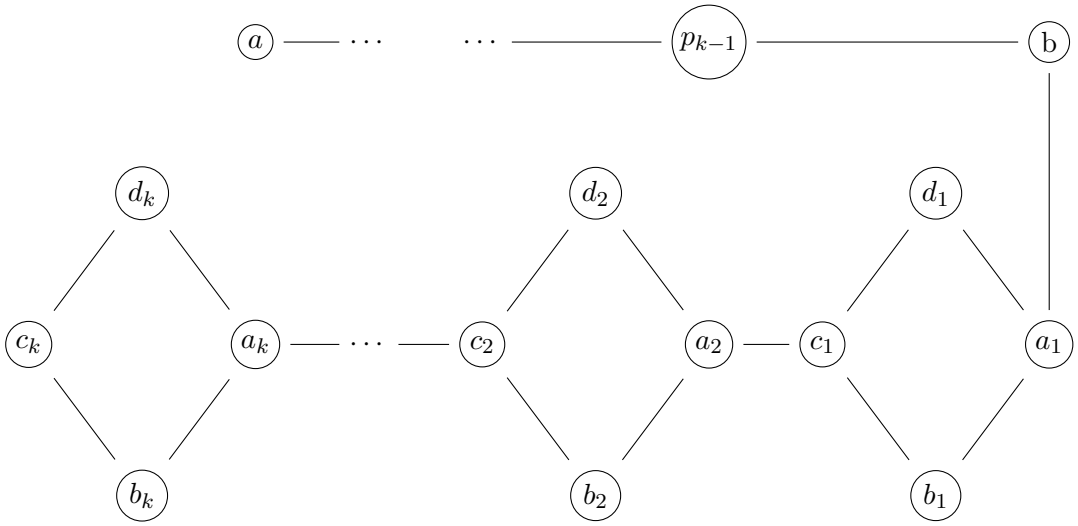

FIGURE 4.3 – Le graphe G_k où p_{k-1} est le $(k-1)^{\text{ème}}$ point de P_k

et aux autres nœuds a_i et c_i des étiquettes distinctes consécutives du bassin des étiquettes restantes.

Soit le nœud a la source, et considérons tout algorithme de diffusion dans le graphe G_k.
Par le Lemme 4.5.1, chaque nœud du chemin, autre que b, doit envoyer $\Omega(\log M)$ bips, sinon le nœud suivant ne peut pas recevoir le message.
Par le Lemme 4.5.2, le coût total de l'algorithme dans chaque sous-graphe C_i, pour $i < k$, doit être $\Omega(\log \log L)$, sinon les nœuds de la copie suivante ne peuvent pas recevoir le message. (Notons que les arêtes du chemin P_k et les arêtes qui joignent les copies consécutives du cycle, sont des ponts dans le graphe G_k.)
En conséquence, le coût total de l'algorithme est $\Omega(k \log M + k \log \log L) = \Omega(n \log M + n \log \log L)$. □

4.6 Réseaux avec connaissance totale

Dans cette section, nous considérons les réseaux dans lesquels les nœuds possèdent des étiquettes distinctes, chaque nœud connaît toute la carte étiquetée du réseau et l'étiquette de la source. Avec cette connaissance totale, tous les nœuds peuvent utiliser le même arbre couvrant T du réseau, enraciné à la source. (Tous les arbres enracinés à la source peuvent être canoniquement codés par des séquences binaires, et l'arbre T peut être choisi comme celui ayant le code le plus petit lexicographiquement.) Soit S le parcours DFS de l'arbre T dans lequel les enfants de chaque nœud sont explorés dans l'ordre croissant de leurs étiquettes. Le parcours Eulerien de l'arbre T correspondant à ce parcours DFS peut être représenté comme une séquence $(a_1, \ldots, a_{2(n-1)})$ de longueur $2(n-1)$ d'étiquettes de nœuds avec répétitions, où a_i correspond à la $i^{\text{ème}}$ arête traversée dans le parcours, à partir du nœud avec étiquette a_i au nœud avec étiquette a_{i+1}.

Le seul message qui circule dans le réseau est le message $[m]$, où m est le message source, et $[m]$ est le codage de cet entier, décrit dans la Section 4.5.

L' instruction envoyer $[m]$ est la procédure d'envoyer les bips du codage $[m]$ dans les rondes consécutives. De même que dans l'algorithme `Connaissance du voisinage`, la propriété de disjonction est satisfaite, et alors chaque message est correctement décodé par les nœuds adjacents.
L'idée de l'algorithme est la suivante. Chaque nœud connaît les termes de la séquence $(a_1, \ldots, a_{2(n-1)})$ qui correspondent à son étiquette. Il envoie le message $[m]$ quand le tour de l'un de tels termes arrive. (Plusieurs nœuds envoient le message plusieurs fois). Pour savoir quand cela se produit, le nœud calcule combien de messages précédents il devrait recevoir avant, de tous les nœuds

adjacents, et lorsque ce nombre de messages reçus est atteint, il procède à l'exécution de l'instruction envoyer $[m]$ correspondante au terme donné de la séquence.

Cet algorithme est exécuté par un nœud avec étiquette ℓ, quand le message source est m. Le pseudocode de l'algorithme suit.

Algorithme `Connaissance totale`

si le nœud exécutant n'est pas la source **alors**

quand un message $[m]$ est décodé pour la première fois, alors

afficher *message* comme message source

identifier toutes les positions de l'étiquette ℓ dans la séquence $(a_1, \ldots, a_{2(n-1)})$

soit $i_1, \ldots, i_r$ ces positions

soit x_1 le nombre d'indices $1 \leq j < a_1$, correspondant aux étiquettes a_j de nœuds adjacents au nœud avec étiquette ℓ

pour $1 < i \leq r$, soit x_i le nombre d'indices $a_{i-1} < j < a_i$, correspondant aux étiquettes a_j de nœuds adjacents au nœud avec étiquette ℓ

pour $1 < i \leq r$, soit $y_i = \sum_{t=1}^{i} x_t$

pour $k = 1$ **à** r **faire**

quand un total de y_k messages $[m]$ est reçu, alors envoyer $[m]$ $\diamond$

Théorème 4.6.1. *À l'achèvement de l'algorithme* `Connaissance totale` *dans un graphe arbitraire avec n nœuds, chaque nœud affiche correctement le message source. Le coût de l'algorithme est* $O(n \log M)$.

Démonstration. L'exactitude de l'algorithme résulte du fait que les nœuds envoient des messages chaque fois que leur tour arrive dans la séquence $(a_1, \ldots, a_{2(n-1)})$ qui correspond à un parcours Eulerien de l'arbre couvrant T, et de la propriété de disjonction qui garantit que le message source est toujours correctement

décodé. Le nombre total de messages envoyés est $2(n-1)$. Étant donné que chaque message correspond à $O(\log M)$ bits, le coût total de l'algorithme est $O(n \log M)$. □

La proposition suivante montre que le coût de l'algorithme `Connaissance totale` est optimal dans les réseaux avec connaissance totale.

Proposition 4.6.1. *Pour des entiers arbitraires $n \geq 2$, il existe des graphes avec n nœuds dans lesquels le coût de tout algorithme de diffusion est $\Omega(n \log M)$.*

Démonstration. Considérons un chemin simple P_n avec n nœuds, dont l'une des extrémités est la source. Notons que le Lemme 4.5.1 est aussi valide pour les réseaux avec connaissance totale. Par le Lemme 4.5.1, chaque nœud du chemin, autre que le dernier nœud, doit transmettre $\Omega(\log M)$ bips, sinon le nœud suivant ne peut pas recevoir le message correctement. En conséquence, le coût de tout algorithme de diffusion est $\Omega(n \log M)$. □

Chapitre 5

Synchronisation globale et consensus utilisant les bips dans un MAC sujet aux pannes

5.1 Introduction

Dans ce chapitre nous étudions les algorithmes de communication dans le modèle de la communication par les bips dans un canal à accès multiple (MAC) pour les tâches de la synchronisation globale et du consensus. Nous allons aborder le scénario où les pannes de communication se produisent dans le canal d'une façon aléatoire.

5.1.1 Le problème

La synchronisation globale est une condition préalable importante pour de nombreuses tâches distribuées. La communication entre les processeurs se déroule en rondes synchrones. Les processeurs sont réveillés dans des rondes

possiblement différentes. L'horloge de chaque processeur commence dans sa ronde de réveil montrant la ronde locale 0, et ensuite tique une fois par ronde en incrémentant la valeur de l'horloge locale par un. La ronde globale 0, inconnue des processeurs, est la ronde de réveil du premier processeur. La synchronisation globale (ou l'établissement d'une horloge globale) signifie que chaque processeur choisit une ronde d'horloge locale telle que les rondes choisies correspondent toutes à la même ronde globale t. La réalisation de la synchronisation globale permet de supposer que dans une tâche ultérieure, tous les processeurs commencent dans la même ronde. Cette hypothèse est souvent utilisée dans la résolution de divers problèmes distribués.

Pour la tâche du consensus, les processeurs ont des valeurs d'entrée dans un ensemble V d'entiers non-négatifs. L'objectif pour tous les processeurs est de satisfaire les exigences suivantes.
Terminaison : tous les processeurs doivent afficher une valeur de l'ensemble V.
Accord : toutes les valeurs affichées doivent être identiques.
Validité : si toutes les valeurs d'entrée sont égales à v, alors toutes les valeurs affichées doivent être égales à v.

5.1.2 La description du modèle

Nous étudions les tâches de la synchronisation globale et du consensus dans un MAC sujet aux pannes. Dans un MAC, tous les processeurs peuvent communiquer directement, c'est-à-dire que le graphe de communication sous-jacent est complet. Certains processeurs se réveillent spontanément, dans des rondes possiblement différentes décidées par un adversaire. Chaque processeur a une horloge locale qui commence à son réveil, affichant le nombre 0. Toutes les horloges tiquent au même rythme, une tique par ronde. C'est une version faible

de la synchronie, qui devrait être contrastée avec l'hypothèse d'une horloge globale où l'horloge de chaque processeur montre un nombre entier global qui est égal pour tous. Nous supposons que les processeurs sont sans pannes, tandis que le MAC est sujet à des pannes aléatoires. Les pannes dans le canal peuvent être dues à un bruit aléatoire se produisant dans l'arrière-plan. Dans chaque ronde, un processeur réveillé peut soit écouter, c'est-à-dire rester silencieux, soit parler, c'est-à-dire émettre un signal. Dans chaque ronde, une panne aléatoire peut se produire dans le canal indépendamment avec une probabilité constante $0 < p < 1$. Dans une ronde sans panne, un processeur réveillé entend un bip s'il écoute dans cette ronde et si un ou plusieurs autres processeurs bipent dans cette ronde. Un processeur encore dormant dans une ronde sans pannes dans laquelle un autre processeur bipe est réveillé par ce bip qu'il entend. Dans une ronde défectueuse (c'est-à-dire avec panne), rien n'est entendu, quel que soit le comportement des processeurs.

5.1.3 Les résultats

Nous concevons et analysons, pour toute constante $\epsilon > 0$, un algorithme de synchronisation globale $\epsilon-$certain déterministe qui fonctionne en temps constant dans n'importe quel MAC avec pannes aléatoires qui utilise la communication par les bips. Un algorithme qui fonctionne avec probabilité d'erreur au plus ϵ, pour un $\epsilon > 0$ donné, s'appelle $\epsilon-$certain. Les processeurs connaissent tous la valeur de ϵ.

En tant qu'application, nous résolvons le problème du consensus dans un MAC avec pannes aléatoires qui utilise la communication par les bips. Les processeurs ont des valeurs d'entrée dans un ensemble V et ils doivent décider de la même valeur à partir de cet ensemble. Si tous les processeurs ont la

même valeur d'entrée, ils doivent tous décider de cette valeur. À l'aide de la synchronisation globale, nous donnons un algorithme du consensus $\epsilon-$certain déterministe qui fonctionne en temps $O(\log w)$ dans un MAC avec pannes aléatoires, où w est la plus petite valeur d'entrée de tous les processeurs participants. Nous montrons que ce temps ne peut pas être amélioré, même si le MAC est sans pannes.

Les résultats de ce chapitre ont été publiés dans l'article :
K. Hounkanli, A. Miller, A. Pelc,
Global Synchronization and Consensus Using Beeps in a Fault-Prone MAC,
Proc. 12th International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS 2016), 16-28.

5.2 La synchronisation globale

Dans cette section, nous proposons un algorithme `GlobalSync` qui établit une horloge globale. À son réveil, chaque processeur dans le réseau exécute `GlobalSync` avec son horloge locale initialisée à 0. La ronde dans laquelle le premier réveil se produit est définie comme la ronde globale 0. Les processeurs ne connaissent pas la relation entre leurs valeurs d'horloges locales et cette ronde globale. L'établissement d'une horloge globale signifie que tous les processeurs du réseau quittent `GlobalSync` dans la même ronde globale.

Soit $0 < p < 1$ la probabilité qu'une ronde soit défectueuse. Soit $\epsilon > 0$ une constante fixée. Soit γ une constante telle que $p^\gamma < \frac{\epsilon}{4}$. Alors, dans une séquence de γ rondes consécutives de bips, au moins un de ces bips se produit dans une ronde sans panne avec probabilité d'au moins $1 - \frac{\epsilon}{4}$.

Nous décrivons l'algorithme `GlobalSync` dont l'objectif est de s'assurer

que tous les processeurs s'accordent sur une ronde globale commune, c'est-à-dire qu'ils établissent une horloge globale. À un haut niveau d'abstraction, l'algorithme se déroule comme suit. Un processeur qui se réveille spontanément bipe périodiquement en essayant de réveiller tous les autres processeurs qui sont encore en sommeil. Ces bips seront appelés des *bips d'alarme*. Ils sont séparés par des intervalles de temps de taille croissante, ce qui empêche un adversaire de régler les rondes de réveil afin que tous les bips d'alarme soient alignés. Dans les intervalles entre les bips d'alarme, le processeur attend une réponse des autres processeurs pour indiquer qu'ils ont entendu un bip d'alarme. Si un nombre suffisant de ces intervalles se produit sans réponse, le processeur suppose que l'ensemble du réseau a été réveillé en même temps, et une ronde globale est choisie comme la ronde dans laquelle le bip d'alarme suivant est programmé. Sinon, si un bip a été entendu dans l'un de ces intervalles, le processeur écoute 2γ rondes consécutives et ensuite il bipe 2γ rondes consécutives. De même, un processeur réveillé par un bip écoute 2γ rondes consécutives et ensuite il bipe 2γ rondes consécutives. La ronde globale choisie par l'algorithme est la ronde $r + 4\gamma + 1$, où r est la première ronde pendant laquelle un bip d'alarme a été entendu par un processeur. La difficulté est pour chaque processeur de déterminer la ronde r. C'est parce que, lorsqu'un bip est entendu, il y a deux cas possibles : un tel bip peut être un bip d'alarme provenant d'un autre processeur, ou peut-être un bip en réponse à un bip d'alarme. Nous surmontons cette difficulté comme suit. Le temps est divisé en blocs de 2γ rondes consécutives. Si un seul bip est entendu dans un bloc, le processeur conclut qu'il s'agissait d'un bip d'alarme ; si plus d'un bip est entendu dans un bloc, le processeur conclut que ces bips étaient en réponse à un bip d'alarme. Nous montrerons que de telles conclusions sont correctes avec une probabilité suffisamment grande. Enfin, chaque processeur considère la première ronde s dans laquelle il a entendu

un bip. Si ce bip était un bip d'alarme, le processeur définit $r = s$. Si ce bip a été émis en réponse à un bip d'alarme, le processeur définit r comme étant la ronde la plus récente avant s dans laquelle il a émis un bip.

Nous donnons maintenant les détails de l'algorithme `GlobalSync`. La procédure suivante fournit un compte global des bips récemment entendus par un processeur. Plus précisément, pour une ronde t' donnée, les 4γ prochaines rondes sont traitées comme deux blocs de 2γ rondes chacun, et pour chaque bloc on distingue les cas de 0, 1 ou plusieurs bips.

Procédure `ListenVector(`t'`)`

1 : $h_1 \leftarrow 0$

2 : $h_2 \leftarrow 0$

3 : $num_1 \leftarrow$ nombre de bips entendus dans les rondes $t', t'+1, \cdots, t'+2\gamma-1$

4 : $num_2 \leftarrow$ nombre de bips entendus dans les rondes $t'+2\gamma, \cdots, t'+4\gamma-1$

5 : **si** $num_1 = 1$, **alors** $h_1 \leftarrow 1$

6 : **si** $num_1 > 1$, **alors** $h_1 \leftarrow *$

7 : **si** $num_2 = 1$, **alors** $h_2 \leftarrow 1$

8 : **si** $num_2 > 1$, **alors** $h_2 \leftarrow *$

9 : **retourne** $[h_1 h_2]$

Nous donnons ci-dessous le pseudocode de l'algorithme `GlobalSync` en utilisant la procédure ci-dessus.

Algorithme `GlobalSync`

1 : **si** réveillé par un bip dans une ronde $heard$: ▷ reveillé par un bip

2 : bipe 2γ rondes consécutives à partir de la ronde $heard + 2\gamma + 1$

3 : $syncRound \leftarrow heard + 4\gamma + 1$

4 : **sinon :** ▷ reveillé spontanément

5 : $i \leftarrow 0$

6 : $myNextBeep \leftarrow 0$

7 : **répète** :

8 : $myCurrentBeep \leftarrow myNextBeep$

9 : bipe dans la ronde $myCurrentBeep$

10 : $i \leftarrow i + 1$

11 : $myNextBeep \leftarrow myCurrentBeep + 4\gamma + i$

12 : **jusqu'à** $(i = 3\gamma)$ ou (un bip est entendu dans l'une des rondes $\{myCurrentBeep + 1, \cdots, myNextBeep - 1\}$)

13 : **si** $i = 3\gamma$:

14 : $syncRound \leftarrow myNextBeep$

15 : **sinon** :

16 : $heard \leftarrow$ première ronde après $myNextBeep$ dans laquelle un bip est entendu

17 : $[h_1 h_2] = ListenVector(myCurrentBeep + 1)$

18 : **si** $[h_1 h_2] \in \{[00], [01], [10], [11], [1*]\}$:

19 : bipe 2γ rondes consécutives à partir de la ronde $heard+2\gamma+1$

20 : $syncRound \leftarrow heard + 4\gamma + 1$

21 : **si** $[h_1 h_2] \in \{[0*], [*0], [*1], [**]\}$:

22 : $syncRound \leftarrow myCurrentBeep + 4\gamma + 1$

23 : attend jusqu'à la ronde $syncRound$ et *quitte* ◇

Dans l'analyse de l'algorithme `GlobalSync`, nous faisons référence aux rondes globales, mais rappelons que les processeurs du MAC n'ont pas accès aux valeurs globales de l'horloge : tout ce qu'un processeur voit est son horloge locale. Le fait suivant découle de la description de l'algorithme par induction sur i.

Fait 5.1.1. *À la fin de chaque itération i de la boucle, la variable $myNextBeep$ est égale à $4\gamma i + \sum_{k=1}^{i} k$. De plus, si un processeur est réveillé à l'instant t,*

alors, à la fin de chaque itération de boucle, myNextBeep est égal à la valeur d'horloge locale correspondant à la ronde globale $t + 4\gamma i + \sum_{k=1}^{i} k$.

Nous disons qu'un processeur est *solitaire* dans la ronde t s'il n'a pas entendu un bip dans n'importe quelle ronde jusqu'à la ronde t inclusivement. En utilisant le Fait 5.1.1, nous pouvons déterminer le nombre de rondes qui s'écoulent avant qu'un processeur solitaire bipe un nombre donné de fois.

Fait 5.1.2. *Supposons qu'un processeur* v *se réveille spontanément dans la ronde* t_1*. Si* v *est solitaire dans la ronde* $t_1 + 4\gamma i + i(i+2)/2$*, alors* v *a bipé exactement* i *fois avant cette ronde.*

Le lemme suivant montre que, pendant un certain intervalle de temps après le premier réveil, aucun processeur ne termine son exécution de `GlobalSync` sans avoir d'abord entendu un bip.

Lemme 5.2.1. *Supposons que le premier réveil spontané se produise à la ronde* t_1*. Alors, aucun processeur ne termine son exécution de* `GlobalSync` *dans l'intervalle de temps* $[t_1, \cdots, t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2 - 1]$ *sans entendre un bip.*

Démonstration. Si un processeur v termine son exécution de `GlobalSync` sans entendre un bip, alors sa boucle **répète** s'achève avec $i = 3\gamma$. Par la ligne 14, le processeur terminera dans la ronde *myNextBeep*, qui d'après Fait 5.1.1, correspond à la ronde globale $t_v + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$, où $t_v \geq t_1$ est la ronde de réveil du processeur v. □

Afin de prouver l'exactitude de l'algorithme, nous considérons d'abord le cas où tous les processeurs se réveillent spontanément dans la même ronde.

Lemme 5.2.2. *Supposons que tous les processeurs se réveillent spontanément dans la même ronde globale t_1. Avec la probabilité 1, tous les processeurs terminent leur exécution de* `GlobalSync` *dans la ronde globale* $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$.

Démonstration. Puisque chaque processeur est réveillé spontanément dans la ronde globale t_1, la condition **si** à la ligne 1 est évaluée à fausse à chaque processeur. Par conséquent, tous les processeurs exécutent la boucle à la ligne 7. Cela signifie en particulier que tous les processeurs émettent un bip dans leur ronde locale 0. Par le Fait 5.1.1, à la fin de chaque itération de la boucle, la variable $myNextBeep$ à chaque processeur est égale à la valeur de l'horloge locale correspondant à la ronde globale $t_1 + 4\gamma i + \sum_{k=1}^{i} k$. Il s'ensuit que tous les processeurs bipent dans les mêmes rondes. En particulier, cela signifie qu'aucun processeur n'entend jamais un bip. Ainsi, à chaque processeur, la boucle sort avec $i = 3\gamma$. Donc, la condition **si** à la ligne 13 est évaluée à vraie. Par la ligne 14, chaque processeur met $syncRound$ à la valeur $4\gamma(3\gamma) + \sum_{k=1}^{3\gamma} k = 12\gamma^2 + (3\gamma)(3\gamma+1)/2$, qui est leur valeur d'horloge locale qui correspond à la ronde globale $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$. Par conséquent, tous les processeurs terminent leur exécution de `GlobalSync` dans la ronde globale $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$. □

Notons que, lorsque notre algorithme est exécuté dans le cas où tous les processeurs se réveillent spontanément dans le même ronde, aucun processeur n'entend jamais un bip, et, après une durée de silence fixe, tous les processeurs terminent leur exécution de `GlobalSync`. Dans le cas où tous les processeurs ne se réveillent pas spontanément dans la même ronde, si la même durée de silence fixe est observée par tous les processeurs, tous les processeurs termineront leur exécution de `GlobalSync`, mais cette fois dans des rondes différentes. Ce serait un mauvais cas pour notre algorithme. Nous montrons maintenant que, avec

une probabilité suffisamment grande, un tel cas ne se produit pas, c'est-à-dire qu'il existe une ronde t^* dans laquelle un bip est entendu par un processeur.

Lemme 5.2.3. *Supposons que tous les processeurs ne se réveillent pas spontanément dans la même ronde et supposons que le premier réveil spontané se produise dans une certaine ronde t_1. Avec la probabilité d'au moins $1 - \frac{\epsilon}{2}$, il existe une ronde globale $t^* \leq t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2$ dans laquelle tous les points suivants sont vérifiés : aucun processeur n'a terminé son exécution de `GlobalSync`, au moins un processeur bipe, au moins un processeur écoute et aucune panne ne se produit.*

Démonstration. Par Lemme 5.2.1, si au moins un processeur termine son exécution de `GlobalSync` dans l'intervalle $[t_1, \cdots, t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1) - 1]$, alors il existe une ronde t^* dans cet intervalle avant la première terminaison, avec la propriété qu'au moins un processeur bipe, au moins un processeur écoute, et aucune faute ne se produit, comme revendiqué.

Ainsi, nous partons de l'hypothèse qu'aucun processeur ne termine son exécution de `GlobalSync` dans l'intervalle $[t_1, \cdots, t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2]$. Soit t_2 la première ronde après t_1 telle qu'un processeur se réveille dans la ronde t_2. Soit v_1 un processeur qui se réveille dans la ronde t_1, et soit v_2 un processeur qui se réveille dans la ronde t_2.

Si le processeur v_1 entend un bip dans une ronde t^* dans l'intervalle $[t_1, \cdots, t_1 + 4\gamma^2 + (\gamma)(\gamma + 1)/2]$, nous avons terminé. Donc, dans le reste de la démonstration, nous supposons que v_1 est solitaire dans la ronde $t_1 + 4\gamma^2 + (\gamma)(\gamma + 1)/2$. Par le Fait 5.1.2, v_1 bipe exactement γ fois dans l'intervalle $[t_1, \cdots, t_1 + 4\gamma^2 + (\gamma)(\gamma + 1)/2 - 1]$.

Si le processeur v_2 entend un bip dans une ronde t^* dans l'intervalle

$[t_1, \cdots, t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2 - 1]$, nous avons terminé. Par conséquent, dans le reste de la démonstration, nous supposons que v_2 est solitaire dans la ronde $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2 - 1$. Cette hypothèse implique que, si $t_2 \leq t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2 - 1$, le réveil de v_2 est spontané.

Premièrement, nous montrons qu'avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, nous avons $t_2 \leq t_1 + 4\gamma^2 + (\gamma)(\gamma+1)/2$. Pour voir pourquoi, rappelons que v_1 bipe γ fois dans l'intervalle $[t_1, \cdots, t_1 + 4\gamma^2 + (\gamma)(\gamma+1)/2 - 1]$. Avec la probabilité d'au moins $1 - \frac{\epsilon}{4}$, un desγ premiers bips de v_1 est émis dans une ronde sans panne. Chaque fois que l'un des γ premiers bips de v_1 est émis dans une ronde s sans panne, nous avons $t_2 < s$, puisque, autrement, v_2 serait réveillé par un bip dans la ronde $s \leq t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2 - 1$, ce qui contredit notre hypothèse ci-dessus à propos du réveil de v_2.

Ensuite, nous notons qu'avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, les 2γ premières rondes dans lesquelles v_2 bipe se produisent avant la ronde $t_1 + 12\gamma^2 + (3\gamma)(3\gamma+1)/2$ (c'est-à-dire, avant que tout processeur ait terminé son exécution de `GlobalSync`). C'est parce que nous avons déjà montré que, avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, nous avons $t_2 \leq t_1 + 4\gamma^2 + (\gamma)(\gamma+1)/2$, et, par le Fait 5.1.2, les 2γ premiers bips de v_2 se produisent avant la ronde $t_2 + 8\gamma^2 + (2\gamma)(2\gamma+1)/2$.

Ensuite, nous montrons qu'avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, dans l'une des 2γ premières rondes dans lesquelles v_2 bipe, le processeur v_1 écoute et aucune panne ne se produit. Nous notons premièrement que, dans au moins γ des 2γ premières rondes dans lesquelles v_2 bipe, le processeur v_1 écoute. Pour voir pourquoi, nous montrons qu'il n'y a pas deux bips consécutifs de v_2 qui se produisent dans les mêmes rondes que les bips de v_1. Si v_2 bipe dans une ronde globale $t_2 + 4\gamma i_2 + \sum_{k=1}^{i_2} k$, et ceci est égale à la ronde globale

$t_1 + 4\gamma i_1 + \sum_{k=1}^{i_1} k$ dans laquelle v_1 bipe, alors, puisque $t_2 > t_1$, nous devons avoir $i_1 > i_2$. Il s'ensuit que le prochain bip de v_2 se produirait dans la ronde $[t_2 + 4\gamma(i_2) + \sum_{k=1}^{i_2} k] + 4\gamma + (i_2 + 1) = [t_1 + 4\gamma(i_1) + \sum_{k=1}^{i_1} k] + 4\gamma + (i_2 + 1) < [t_1 + 4\gamma(i_1) + \sum_{k=1}^{i_1} k] + 4\gamma + (i_1 + 1)$, c'est-à-dire avant le prochain bip de v_1. Ceci montre que, dans au moins γ des 2γ premières rondes dans lesquelles v_2 bipe, le processeur v_1 écoute. Avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, au moins un de ces γ bips se produit dans une ronde sans panne. Alors, avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, dans l'une de ces 2γ premières rondes dans lesquelles v_2 bipe, le processeur v_1 écoute et aucune panne ne se produit.

En somme, nous avons montré que, avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, les 2γ premières rondes dans lesquelles v_2 bipe surviennent avant la ronde $t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2$, et que, avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$, l'un des 2γ premiers bips de v_2 est entendu par v_1. Il s'ensuit que, avec une probabilité d'au moins $1 - \frac{\epsilon}{2}$, il existe une ronde $t^* \leq t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2$ dans laquelle v_2 bipe, v_1 écoute, aucune panne ne se produit, et aucun processeur n'a terminé son exécution de `GlobalSync`. □

Nous montrons maintenant l'exactitude de notre algorithme dans le cas où tous les processeurs ne se réveillent pas spontanément dans la même ronde. Nous serons en mesure de le faire lorsqu'il existe une ronde globale t^* satisfaisant les conditions spécifiées au Lemme 5.2.3. Nous utiliserons le lemme suivant qui établit les périodes d'écoute des processeurs.

Lemme 5.2.4. *Supposons que tous les processeurs ne se réveillent pas spontanément dans la même ronde. Soit t^* la première ronde globale dans laquelle tous les points suivants sont vérifiés : aucun processeur n'a terminé son exécution de `GlobalSync`, au moins un processeur bipe, au moins un processeur écoute et*

aucune panne ne se produit. Alors, aucun processeur ne bipe dans les rondes $t^* + 1, \cdots, t^* + 2\gamma$.

Démonstration. Il y a plusieurs cas à considérer. Premièrement, nous considérons chaque processeur v qui ne bipe pas dans la ronde t^*, et nous montrons que v ne bipe pas avant la ronde $t^* + 2\gamma + 1$. Supposons que v soit réveillé par un bip dans la ronde t^*. Dans ce cas, par la ligne 2, v attend 2γ rondes avant son prochain bip, comme revendiqué. Ensuite, supposons que v entende un bip dans la ronde t^*, et que v ait été réveillé avant la ronde t^*. Ce cas correspond à la clause **sinon** à la ligne 15. Notons que la valeur d'horloge locale correspondant à t^* est stockée dans la variable *heard* de v. Puisque le prochain bip de v se produit à la ligne 19, il s'ensuit que v attend 2γ rondes après la ronde t^*, avant son prochain bip, comme revendiqué.

Enfin, nous considérons le cas où v bipe dans la ronde t^*. Cela se produit à la ligne 9 et notons que la valeur d'horloge locale correspondant à t^* est stockée dans la variable $myCurrentBeep$ de v. Si v n'entend pas de bip entre les rondes $myCurrentBeep$ et $myNextBeep$, alors v ne bipera pas encore avant la ronde $myNextBeep = myCurrentBeep + 4\gamma + i > t^* + 2\gamma$, comme revendiqué. Si v entend un bip entre les rondes $myCurrentBeep$ et $myNextBeep$, soit dans une ronde *heard*, alors le prochain bip de v se produit à la ligne 19. Ce bip est dans la ronde $heard + 2\gamma + 1 > myCurrentBeep + 2\gamma + 1 = t^* + 2\gamma + 1$, comme revendiqué. □

Le prochain lemme montre que tous les processeurs terminent leur exécution de `GlobalSync` dans la même ronde globale peu après t^*.

Lemme 5.2.5. *Supposons que tous les processeurs ne se réveillent pas spontanément dans la même ronde. Soit* t^* *la première ronde globale dans laquelle tous*

les points suivants sont vérifiés : aucun processeur n'a terminé son exécution de `GlobalSync`*, au moins un processeur bipe, au moins un processeur écoute et aucune panne ne se produit. Avec une probabilité au moins* $1 - \frac{\epsilon}{2}$*, tous les processeurs terminent leur exécution de* `GlobalSync` *dans la ronde* $t^* + 4\gamma + 1$.

Démonstration. Premièrement, nous considérons un processeur v qui ne bipe pas dans la ronde t^*. Nous montrons que v bipe dans les rondes $t^*+2\gamma+1, \cdots, t^*+4\gamma$ et que v met *syncRound* à la valeur de l'horloge locale qui correspond à la ronde globale $t^* + 4\gamma + 1$. Il y a deux cas à considérer :
1. Supposons que v se réveille dans la ronde globale t^*. Ceci se produit à la ligne 1. Notons que dans la ronde t^*, la valeur de l'horloge locale correspondant à t^* est stockée dans la variable *heard* de v. Alors, par la ligne 2, le processeur v bipe dans les rondes $t^* + 2\gamma + 1, \cdots, t^* + 4\gamma$, et par la ligne 3, v met *syncRound* à $heard + 4\gamma + 1$, qui est la valeur de l'horloge locale qui correspond à la ronde globale $t^* + 4\gamma + 1$, comme revendiqué.
2. Supposons que v est réveillé avant la ronde globale t^*. Soit s la dernière ronde avant t^* durant laquelle v a bipé. Notons que, dans la ronde t^*, la valeur de l'horloge locale correspondant à s est stockée dans la variable $myCurrentBeep$ de v. Avec le choix de s et t^*, le bip entendu par v durant t^* est le premier beep que v entend entre les rondes $myCurrentBeep$ et $myNextBeep$. Alors, la valeur de l'horloge locale correspondant à t^* est stockée dans la variable *heard* de v. Par Lemme 5.2.4. il n'y a pas de bips émis dans les 2γ prochaines rondes immédiatement après la ronde *heard*. En particulier, cela veut dire que la première entrée non-zéro de $[h_1 h_2]$ ne peut être $*$, donc la condition **si** à la ligne 21 est évaluée à fausse. Puisque les conditions **si** aux lignes 18 et 21 épuisent toutes les 9 possibilités pour le vecteur $[h_1 h_2]$, il s'ensuit que la condition **si** à la ligne 18 est évaluée à vraie. Par conséquent, avant la ligne 19, v bipe dans

les rondes $t^* + 2\gamma + 1, \cdots, t^* + 4\gamma$, et avant la ligne 20, v met *syncRound* à $heard + 4\gamma + 1$, qui est la valeur de l'horloge locale qui correspond à la ronde globale $t^* + 4\gamma + 1$, comme revendiqué.

Par le choix de t^*, il y a au moins un processeur qui ne bipe pas dans la ronde globale t^*. De ce que nous venons de montrer, il s'ensuit qu'au moins un processeur v bipe dans les rondes $t^*+2\gamma+1, \cdots, t^*+4\gamma$. Ensuite, nous montrons que, avec la probabilité d'au moins $1 - \frac{\epsilon}{2}$, plus d'un de ces bips sont émis dans des rondes sans pannes. Considérons les intervalles $[t^* + 2\gamma + 1, \cdots, t^* + 3\gamma]$ et $[t^* + 3\gamma + 1, \cdots, t^* + 4\gamma]$. Le processeur v bipe γ fois dans chacun de ces intervalles. Il s'ensuit que dans chacun de ces intervalles, v bipe dans une ronde sans panne avec une probabilité d'au moins $1 - \frac{\epsilon}{4}$. Par conséquent, avec une probabilité d'au moins $1 - \frac{\epsilon}{2}$, dans l'intervalle $[t^* + 2\gamma + 1, \cdots, t^* + 4\gamma]$, plus d'un bip de v sont émis dans des rondes sans pannes.

Finalement, nous montrons que si plus d'un bip dans les rondes $t^* + 2\gamma + 1, \cdots, t^* + 4\gamma$ sont émis dans des rondes sans pannes, alors chaque processeur v qui bipe dans la ronde t^* met *syncRound* à une valeur d'horloge locale qui correspond à la ronde globale $t^* + 4\gamma + 1$. Considérons tout processeur v qui bipe dans la ronde t^* et supposons que plus d'un de ses bips sont émis dans des rondes sans pannes parmi $t^* + 2\gamma + 1, \cdots, t^* + 4\gamma$. Le bip émis par v dans la ronde t^*, a lieu à la ligne 9, et notons que dans la ronde t^*, la valeur de l'horloge locale correspondant à t^* est stockée dans la variable $myCurrentBeep$ de v. Par Lemme 5.2.4, il n'y a pas de bips qui sont produits dans les rondes $t^*+1, \cdots, t^*+2\gamma$. Puisque plus d'un bip dans des rondes $t^*+2\gamma+1, \cdots, t^*+4\gamma$ sont émis dans des rondes sans pannes, il s'ensuit que, au processeur v, le vecteur $[h_1 h_2]$ est égal à $[0*]$. Donc la condition **si** à la ligne 21 est évaluée à vraie. Par conséquent le processeur v met *syncRound* à $myCurrentBeep + 4\gamma + 1$, qui

est la valeur de l'horloge locale qui correspond à la ronde globale $t^* + 4\gamma + 1$, comme revendiqué. □

Enfin, nous montrons que l'algorithme `GlobalSync` travaille en temps constant et échoue avec une probabilité au plus ϵ pour toute constante donnée $\epsilon > 0$.

Théorème 5.2.1. *Fixons une constante $\epsilon > 0$. Avec une probabilité au moins $1 - \epsilon$, tous les processeurs terminent Algorithme `GlobalSync` dans la même ronde globale `sync`, qui survient $O(1)$ rondes après le premier réveil.*

Démonstration. Soit t_1 la première ronde dans laquelle se produit un réveil. Dans le cas où tous les processeurs se réveillent spontanément dans la même ronde, Lemme 5.2.2 indique que, avec probabilité 1, tous les processeurs terminent Algorithme `GlobalSync` dans la ronde globale `sync`$= t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2$. Dans le cas où tous les processeurs ne se réveillent pas spontanément dans la même ronde, Lemme 5.2.3 et Lemme 5.2.5 indiquent que tous les processeurs terminent Algorithme `GlobalSync` dans la ronde globale `sync`$= t^* + 4\gamma + 1$, où $t^* \leq t_1 + 12\gamma^2 + (3\gamma)(3\gamma + 1)/2$, avec probabilité d'erreur au plus $\frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$. □

5.3 Le consensus

Dans cette section, nous proposons un algorithme de décision déterministe qui aboutit à un consensus dans le MAC avec pannes aléatoires en supposant que la synchronisation globale ait été effectuée précédemment. L'algorithme est exécuté après Algorithme `GlobalSync` et a la propriété suivante. Soit `sync` la ronde globale dans laquelle tous les processeurs dans le réseau terminent leur exécution de Algorithme `GlobalSync`. Algorithme `Décision` accomplit

un consensus avec une probabilité d'erreur au plus ϵ dans la ronde globale $s = \texttt{sync} + O(\log w)$, où w est la plus petite de toutes les valeurs d'entrée des processeurs dans le MAC.

Considérons la valeur d'entrée *val* d'un processeur v et soit $\mu = (a_1, \cdots, a_m)$ sa représentation binaire. Nous transformons la suite μ en remplaçant chaque bit 1 par (10), chaque bit 0 par (01) et en ajoutant (11) à la fin. Donc la séquence transformée est $(c_1, \cdots, c_{2m+2})$, où
$c_i = 1$, pour $i \in \{2m+1, 2m+2\}$, et,
pour $j = 1, \cdots, m$:
$c_{2j-1} = 1$ et $c_{2j} = 0$, si $a_j = 1$,
$c_{2j-1} = 0$ et $c_{2j} = 1$, si $a_j = 0$.

La séquence $(c_1, \cdots, c_{2m+2})$ est appelée la *valeur d'entrée transformée* du processeur v et est dénotée val^*. Remarquons que si les valeurs d'entrée de deux processeurs sont différentes, alors il existe un index pour lequel les valeurs correspondantes des bits de leurs valeurs d'entrée transformées diffèrent (ce n'est pas nécessairement le cas pour les valeurs d'entrée d'origine, puisque l'une des représentations binaires pourrait être un préfixe de l'autre).

L'idée, à un niveau élevé d'abstraction, de Algorithme `Décision` est la suivante. Un processeur émet un bip et écoute pendant des intervalles de temps de longueur prescrite, en commençant dans la ronde globale `sync`+1, en fonction de sa valeur d'entrée transformée. S'il n'entend aucun bip, il conclut que toutes les valeurs d'entrée sont identiques et affiche sa valeur d'entrée. Sinon, il conclut qu'il existe différentes valeurs d'entrée et affiche ensuite une valeur par défaut. Nous montrons que ces conclusions sont correctes avec une probabilité au moins $1 - \epsilon$, et que tous les processeurs prennent la décision dans une ronde globale commune $s = \texttt{sync} + O(\log w)$.

Nous donnons maintenant le pseudocode de l'algorithme exécuté par un processeur dont la valeur d'entrée est val. Nous supposons que l'algorithme est démarré dans la ronde globale `sync`+1, et soit r la valeur d'horloge locale du processeur correspondant à la ronde globale `sync`. Soit x le plus petit entier positif tel que $p^x < \epsilon/2$. Soit val_0, le plus petit entier dans V, que nous utiliserons comme valeur de décision par défaut.

Algorithme `Décision`

1 : $(c_1, \cdots, c_k) \leftarrow val^*$

2 : $i \leftarrow 1$

3 : $heard \leftarrow fausse$

4 : **tant que** ($heard = fausse$ **et** $i \leq k$) **faire**

5 : **si** $c_i = 1$ **alors** biper dans x rondes et ensuite écouter dans x rondes

6 : **si** $c_i = 0$ **alors** écouter dans x rondes et ensuite biper dans x rondes

7 : **si** un bip est entendu **alors** $heard \leftarrow vraie$

8 : $i \leftarrow i + 1$

9 : **si** $heard = fausse$ **alors** afficher val dans la ronde $r + 2(i-1)x + 1$

10 : **sinon** affiche val_0 dans la ronde $r + 2(i-1)x + 1$ $\diamond$

Le résultat suivant montre qu'avec une probabilité d'erreur d'au plus ϵ, à la fin de Algorithme `Décision`, tous les processeurs dans le MAC résolvent correctement le consensus dans la même ronde, et cette ronde survient $O(\log w)$ rondes après la ronde globale `sync`, où w est la plus petite de toutes les valeurs d'entrée des processeurs dans le MAC.

Théorème 5.3.1. *Soit* `sync` *la ronde globale commune dans laquelle tous les processeurs terminent leur exécution de Algorithme* `GlobalSync`*, et soit w la plus petite de toutes les valeurs d'entrée des processeurs dans le réseau. Il existe*

une ronde globale $s = \texttt{sync} + O(\log w)$ *telle que, avec la probabilité au plus* $1 - \epsilon$*, à la fin de Algorithme* `Décision`*, tous les processeurs du réseau affichent la même valeur dans la ronde globale* s*, et cette valeur est leur valeur d'entrée commune si toutes les valeurs d'entrée étaient identiques.*

Démonstration. Supposons d'abord que les valeurs d'entrée de tous les processeurs du réseau sont identiques. Soit $k \in O(\log w)$ la longueur de leur valeur d'entrée transformée commune. Alors chaque processeur quitte la boucle **tant que** avec la valeur de la variable *heard* égale à $fausse$, et par conséquent, à la ligne 9, il affiche la valeur d'entrée commune à la ronde $r + 2xk + 1$, qui est sa valeur d'horloge locale correspondant à la ronde globale $\texttt{sync} + 2xk + 1$. Puisque x est une constante, nous avons $2xk + 1 \in O(\log w)$, ce qui conclut la démonstration dans ce cas.

Dans le reste de la démonstration, nous supposons qu'il existe au moins deux valeurs d'entrée distinctes. Soit k_1 la longueur de la valeur d'entrée transformée w^* correspondant à la valeur d'entrée w. Considérons toutes les valeurs d'entrée transformées des processeurs dans le réseau, et soit $j \leq k_1$ le premier index dans lequel deux de ces valeurs d'entrée transformées diffèrent.

Pour tout $t > 0$, soit A_t l'intervalle de temps global $\{\texttt{sync} + 2x(t-1) + 1, \cdots, \texttt{sync} + 2x(t-1) + x\}$, soit B_t l'intervalle de temps global $\{\texttt{sync} + 2x(t-1) + x + 1, \cdots, \texttt{sync} + 2x(t-1) + 2x\}$. Soit E l'événement qu'au moins une ronde dans l'intervalle de temps A_j soit sans panne et qu'au moins une ronde dans l'intervalle de temps B_j soit sans panne. Par la définition de x, la probabilité de l'événement E est au moins $1 - \epsilon$. Supposons que l'événement E est valide. Par le choix de j, aucun bip n'a été entendu dans le canal dans les rondes globales $\{\texttt{sync} + 1, \cdots, \texttt{sync} + 2x(j-1)\}$, donc tous les processeurs du réseau participent à la $j^{\texttt{ième}}$ itération de la boucle. Considérons tout processeur v pour

lequel le $j^{\texttt{ième}}$ bit de sa valeur d'entrée transformée est 0 et tout processeur v' pour lequel le $j^{\texttt{ième}}$ bit de sa valeur d'entrée transformée est 1. Le processeur v écoute dans toutes les rondes de A_j et bipe dans toutes les rondes de B_j, le processeur v' bipe dans toutes les rondes de A_j et écoute dans toutes les rondes de B_j. Par conséquent, v entend au moins un bip dans l'intervalle de temps A_j, et v' entend au moins un bip dans l'intervalle de temps B_j. Donc les deux processeurs v et v' évaluent la valeur de *heard* à *vraie* dans l'itération j de la boucle **tant que**. Par conséquent, chaque processeur affiche la valeur par défaut val_0 à la ligne 10 dans la ronde $r + 2xj + 1$, qui est sa valeur d'horloge locale correspondant à la ronde globale `sync`$+2xj + 1$. Puisque x est une constante et $j \leq k_1 \in O(\log w)$, on a $2xj + 1 \in O(\log w)$, ce qui conclut la démonstration dans le cas où il y a au moins deux valeurs d'entrée distinctes. □

Finalement, étant donnée une borne $\epsilon > 0$ sur l'erreur de probabilité du consensus, nous exécutons d'abord Algorithme `GlobalSync` et ensuite Algorithme `Décision`, chacun avec une erreur de probabilité $\frac{\epsilon}{2}$, pour obtenir le corollaire suivant.

Corollaire 5.3.1. *Soit $\epsilon > 0$ une constante fixée. Considérons un MAC sujet aux pannes aléatoires avec probabilité $0 < p < 1$, avec une communication par les bips, où w est la plus petite de toutes les valeurs d'entrée des processeurs dans le canal. Avec une probabilité d'erreur au plus ϵ, le consensus peut-être résolu de façon déterministe dans la même ronde, $O(\log w)$ rondes après le premier réveil.*

Nous concluons cette section en montrant que même dans un modèle où chaque ronde du MAC est sans panne et que tous les processeurs sont réveillés spontanément dans la même ronde, le consensus déterministe avec des entrées

de longueur m bits nécessite $\Omega(m)$ rondes, ce qui indique que notre algorithme de consensus a une complexité optimale en temps.

Théorème 5.3.2. *Le consensus avec des entrées de longueur m bits dans un MAC sans pannes avec la communication par bips nécessite $\Omega(m)$ rondes.*

Démonstration. Considérons un algorithme de consensus A. Supposons que, pour chaque valeur d'entrée s de m bits, l'exécution de A avec l'entrée s par un processeur unique dans le canal utilise $o(m)$ rondes. Pour toute entrée, soit $Pattern(s)$ la séquence de bips d'un processeur qui est seul dans le canal et exécute A avec l'entrée s. D'après le *Principe des tiroirs*, il existe des entrées de de m bits, a et b distinctes, telles que $Pattern(a) = Pattern(b)$.

Pour chaque $s \in \{a, b\}$, soit α_s l'exécution de A dans le cas où un processeur v_s est seul dans le canal et est donné la valeur d'entrée s. Par la *validité*, pour tout $s \in \{a, b\}$, à la fin de l'exécution α_s, le processeur v_s doit afficher s. Ensuite, considérons l'exécution $\alpha_{a,b}$ de A dans le cas où les processeurs v_a et v_b sont dans le canal et sont donnés les entrées a et b respectivement. Puisque $Pattern(a) = Pattern(b)$, il s'ensuit que les exécutions α_a et $\alpha_{a,b}$ sont indiscernables au processeur v_a et les exécutions α_b et $\alpha_{a,b}$ sont indiscernables au processeur v_b. Donc dans l'exécution $\alpha_{a,b}$, le processeur v_a affiche a et le processeur v_b affiche b, ce qui contredit l'*accord*. Par conséquent, nous avons supposé à tort que, pour chaque entrée de m bits, l'exécution de A avec entrée s par un seul processeur dans le canal utilise $o(m)$ rondes. Il s'ensuit qu'il existe une exécution de A qui utilise $\Omega(m)$ rondes, comme revendiqué. □

Chapitre 6

Élection du chef avec bips dans un canal à accès multiple sujet aux pannes

6.1 Introduction

6.1.1 Le contexte

Le but de la tâche d'élection du chef est qu'à la fin de l'algorithme, exactement un nœud prenne la décision qu'il est le chef et que tous les autres nœuds décident qu'ils ne sont pas chefs. L'élection du chef est la façon ultime de briser les symétries dans un réseau. C'est l'opération primitive naturelle qui est utilisée comme première étape dans la résolution de beaucoup d'autres tâches de haut niveau nécessitant ou bénéficiant de la présence d'un «organisateur» désigné. Nous étudions les algorithmes de communication dans le modèle de la communication par les bips dans un canal à accès multiple (MAC) sujet aux pannes, pour la tâche d'élection du chef. Nous allons aborder le scénario où les

pannes de communication se produisent dans le canal d'une façon aléatoire.

6.1.2 Le modèle

Nous considérons un MAC de N processeurs, où les rondes sont sujettes aux pannes indépendantes avec probabilité constante $0 < p < 1$. Les pannes sont de la même nature que celles considérées dans le chapitre 5. Dans une ronde sans panne, un processeur réveillé entend un bip s'il écoute dans cette ronde et si un ou plusieurs autres processeurs bipent dans cette ronde. Un processeur encore dormant dans une ronde sans panne dans laquelle un autre processeur bipe est réveillé par ce bip qu'il entend. Dans une ronde défectueuse (avec panne), rien n'est entendu, quel que soit le comportement des processeurs. La communication se déroule en rondes. Dans chaque ronde, un nœud peut soit écouter (c'est-à-dire rester silencieux) soit envoyer un bip. Certains processeurs se réveillent spontanément, dans des rondes possiblement différentes décidées par un adversaire. Chaque processeur a une horloge locale qui commence à son réveil, affichant le nombre 0. Toutes les horloges tiquent au même rythme, un tic par ronde. Chaque processeur a une étiquette unique dans l'ensemble $\{1, \cdots, L\}$, où $L \geq N$ est un entier non-négatif polynomial en N, appelé l'espace des étiquettes. Chaque processeur connaît son étiquette et connaît L. Notre objectif est de concevoir et d'analyser un algorithme d'élection du chef presque certain (c'est-à-dire qui fonctionne avec probabilité au moins $1 - \frac{1}{N}$) travaillant en temps $O(\log^2 L) = O(\log^2 N)$.

6.2 Élection du chef

Dans cette section, nous proposons l'algorithme `Élection du chef` qui élit le chef dans n'importe quel MAC sujet aux pannes de transmission aléatoires et qui est presque certain. Puisque les processeurs n'ont pas accès à une horloge globale à leur réveil (possiblement dans des rondes différentes), nous établissons préalablement la synchronisation globale du réseau. Cette condition primitive est réalisée par l'exécution par chaque processeur à son réveil, de l'algorithme `GlobalSync` décrit dans le chapitre précédent. Après cette exécution les processeurs établissent une ronde globale commune à laquelle ils commencent l'exécution de la procédure `Élection synchronisée`.

La procédure `Élection synchronisée` élit le chef dans n'importe quel MAC avec pannes, en supposant que la synchronisation globale ait été effectuée précédemment et que les processeurs commencent tous en même temps l'élection du chef. Avant de présenter la procédure, nous définissons le codage des étiquettes (entiers non-négatifs). Soit l l'étiquette du nœud v et soit $(a_1 \cdots a_k)$ la représentation binaire de l. Soit $(a_1 \cdots a_K)$ la représentation binaire de L. Donc $K = \lfloor \log L \rfloor + 1$. Nous appelons représentation binaire augmentée de l la représentation binaire de l augmentée à gauche de $(K - k)$ zéros, c'est-à-dire $(c_1 \cdots c_K) = (\underbrace{0 \cdots 0}_{(K-k) fois}\ a_1 \cdots a_k)$.

En utilisant le codage ci-dessus, nous pouvons décrire notre procédure d'élection du chef. À un haut niveau d'abstraction, les nœuds transmettent leurs représentations binaires augmentées par des segments d'une certaine longueur (logarithmique en L) : si le bit de leur représentation binaire augmentée est 1, le nœud bipe pendant toutes les rondes du segment, sinon, il écoute. Au fur et à mesure que nous avançons dans la transmission des séquences binaires représen-

tatives des étiquettes, les nœuds d'étiquettes augmentées lexicographiquement moins grandes s'éliminent (deviennent non-actifs) jusqu'à ce que le nœud avec l'étiquette la plus grande lexicographiquement soit le seul actif et s'élit chef. Au début, tous les nœuds sont actifs. Un nœud actif dans un segment s reste actif dans le segment $s+1$ s'il n'entend pas de bip dans ce segment. Un nœud actif dans un segment s devient non-actif à partir du début du segment $s+1$ jusqu'à la fin du processus, s'il entend un bip dans le segment s. Nous formulons maintenant la procédure `Élection synchronisée` qui travaille dans tous les MAC avec pannes de transmission aléatoires. Cette procédure est exécutée par chaque nœud. Soit le nœud v avec représentation binaire augmentée $(c_1 \cdots c_K)$.

Procédure `Élection synchronisée`

Soit t la ronde 0 de synchronization globale, dans laquelle le processus de l'élection du chef commence. Chaque nœud divise toutes les rondes à partir de t en K segments $\{s_1, \cdots, s_K\}$, chacun de longueur $a = \lceil d \log L \rceil$ rondes, où $d = \lceil -\frac{3}{\log p} \rceil$. Au début tous les nœuds sont actifs. Pendant toutes les rondes du segment s_i, si le nœud v est resté actif il bipe si $c_i = 1$ et il écoute si $c_i = 0$. À la fin du segment s_i un nœud actif qui a entendu au moins un bip pendant ce segment devient non-actif. Le nœud qui est resté actif jusqu'à la fin du processus s'élit chef et tous les autres nœuds (non-actifs) décident qu'ils ne sont pas chefs. ♢

Le théorème suivant montre que la procédure `Élection synchronisée` élit correctement le chef et est presque certaine dans un MAC avec pannes de transmission aléatoires.

Théorème 6.2.1. *Considérons un MAC sujet aux pannes aléatoires indépendantes avec probabilité constante $0 < p < 1$. Si tous les processeurs commencent la procédure `Élection synchronisée` à la même ronde globale t, alors, avec*

la probabilité au moins $1 - \frac{1}{2N}$*, le processeur avec l'étiquette augmentée lexicographiquement la plus grande est le seul processeur actif à la fin de cette procédure.*

Démonstration. Soit E l'évènement qu'il existe un segment s_i, pour $i \leq K$ dans lequel toutes les rondes sont en panne. Nous avons :
$Prob(E) \leq K \cdot p^a = (\lfloor \log L \rfloor + 1) \cdot p^{d \log L} \leq L \cdot L^{d \log L} \leq L \cdot L^{-3} = \frac{1}{L^2} \leq \frac{1}{2N}$.
Soit F le complément de l'évènement E. Donc il suffit de montrer que si l'évènement F a lieu, alors le processeur v est le seul processeur actif à la fin de la procédure `Élection synchronisée`. Supposons que l'évènement F a lieu. Nous prouvons d'abord que v reste actif à la fin de la procédure. Supposons que non et soit j l'indexe du segment à la fin duquel v devient non-actif. Dans ce segment v a dû entendre un bip d'un processeur w. Donc le $j^{\text{ème}}$ bit de v est 0 et le $j^{\text{ème}}$ bit de w est 1. Dans tous les segments précédents aussi bien v que w sont actifs. Donc pour tout $i < j$, si le $i^{\text{ème}}$ bit de v est 1 alors le $i^{\text{ème}}$ bit de w est 1, sinon w deviendrait non-actif avant le $j^{\text{ème}}$ segment. Il s'ensuit que l'étiquette augmentée de w est lexicographiquement plus grande que l'étiquette augmentée de v, ce qui contredit le choix de v.

Il reste à montrer que chaque processeur $v' \neq v$ devient non-actif pendant la procédure. Soit m l'indexe du premier bit où les étiquettes augmentées de v et v' diffèrent. Donc le $m^{\text{ème}}$ bit de v est 1 et le $m^{\text{ème}}$ bit de v' est 0. Si v' est resté actif jusqu'au $m^{\text{ème}}$ segment, alors il entend le bip de v pendant ce segment et devient non-actif. □

Nous formulons maintenant l'algorithme `Élection du chef` qui élit le chef dans un MAC avec réveil des processeurs possiblement pendant des rondes différentes, sans horloge globale et avec pannes de transmission aléatoires. Les

processeurs exécutent à leur réveil, premièrement l'algorithme `GlobalSync` pour $\epsilon = \frac{1}{2L}$. Soit `sync` la ronde globale dans laquelle tous les processeurs dans le réseau terminent leur exécution de l'algorithme `GlobalSync` avec probabilité au moins $1 - \frac{1}{2L}$. Les processeurs exécutent ensuite la procédure `Élection synchronisée` avec la ronde globale de commencement $t =$`sync`. Chaque processeur exécute l'algorithme suivant.

Algorithme `Élection du chef`

1. **Appeler** Algorithme `GlobalSync` à la ronde du réveil avec $\epsilon = \frac{1}{2L}$
2. **Appeler** Procédure `Élection synchronisée` avec $t =$`sync`.

Théorème 6.2.2. *L'algorithme `Élection du chef` est un algorithme d'élection du chef presque certain dans un MAC avec pannes de transmission aléatoires. Il fonctionne en temps* $O(\log^2 L) = O(\log^2 N)$.

Démonstration. Par le théorème 5.2.1 et le choix de $\epsilon = \frac{1}{2L} \leq \frac{1}{2N}$, la probabilité que l'algorithme `GlobalSync` soit incorrect est au plus $\frac{1}{2N}$. Par le théorème 6.2.1, la probabilité pour que la procédure `Élection synchronisée` soit incorrecte est au plus $\frac{1}{2N}$. Par conséquent l'algorithme `Élection du chef` est correct avec une probabilité au moins $1 - \frac{1}{N}$.
Il reste à montrer que l'algorithme `Élection du chef` fonctionne en temps $O(\log^2 L) = O(\log^2 N)$. L'algorithme `GlobalSync` pour $\epsilon > 0$ fonctionne en temps $O(\gamma^2)$, où γ est le nombre satisfaisant $p^\gamma < \frac{\epsilon}{4}$. Donc pour $\epsilon = \frac{1}{2L}$ cet algorithme fonctionne en temps $O(\log^2 L) = O(\log^2 N)$. La procédure `Élection synchronisée` fonctionne en temps $Ka = (\lfloor \log L \rfloor + 1) \cdot \lceil d \log L \rceil$, où d est une constante, donc elle fonctionne aussi en temps $O(\log^2 L) = O(\log^2 N)$. Il s'ensuit que l'algorithme `Élection du chef` fonctionne en temps $O(\log^2 N)$. □

Nous concluons ce chapitre en montrant la borne inférieure en temps

$\Omega(\log N)$ de la tâche d'élection du chef dans tout MAC, même sans pannes de transmission.

Théorème 6.2.3. *Tout algorithme d'élection du chef dans un MAC fonctionne en temps* $\Omega(\log N)$.

Démonstration. Supposons qu'un algorithme A accomplit l'élection du chef dans un MAC sans pannes de transmission, en un temps $< \frac{1}{2}\log L$, où L est la grandeur de l'espace des étiquettes. Alors, le nombre de comportements possibles (un comportement est une suite de bips et de silences quand un nœud est seul) est inférieur à $2^{\frac{1}{2}\log L} = L^{\frac{1}{2}}$. C'est-à-dire, il y a moins de comportements que d'étiquettes. Alors deux nœuds distincts avec étiquettes $l_1 \neq l_2$ ont le même comportement. Chacun de ces nœuds s'il est seul doit s'élire chef. Considérons l'exécution E de l'algorithme A en présence exclusivement de ces deux nœuds dans le MAC. Aucun de ces nœuds ne peut distinguer l'exécution E de l'exécution où il est seul dans le MAC. Dans dans l'exécution E chacun de ces nœuds doit s'élire chef, ce qui est une contradiction. Il s'ensuit que l'élection du chef dans un MAC prend un temps au moins $\Omega(\log L)$. Puis que $L \geq N$, ça termine la preuve. □

Chapitre 7

Conclusion

Nous avons étudié la communication dans les réseaux sans fil. Premièrement, nous avons étudié la tâche fondamentale de diffusion dans le modèle sans fil asynchrone avec communication par bips bivalents où chaque bip est soit fort soit faible. Deuxièmement, nous avons étudié les algorithmes de communication dans le modèle de la communication par les bips dans un canal à accès multiple (MAC) pour les tâches de la synchronisation globale et du consensus d'une part et la tâche d'élection du chef d'autre part. Nous avons abordé le scénario où les pannes de communication se produisent dans le canal d'une façon aléatoire.

Dans le chapitre 4, nous avons considéré le coût de la diffusion asynchrone dans les réseaux avec quatre niveaux de connaissance différents : anonyme, ad-hoc, connaissance du voisinage et connaissance totale. Nous avons prouvé que la diffusion dans les réseaux anonymes est impossible, et nous avons montré des bornes supérieures et inférieures sur le coût de la diffusion pour les trois autres niveaux de connaissance. Nos résultats montrent des séparations de coûts entre tous les niveaaux. Alors que les bornes pour les réseaux de connaissance totale sont asymptotiquement serrées, les autres bornes ne le sont pas, et la

conception d'algorithmes de diffusion à coût optimal pour les réseaux ad-hoc et pour les réseaux de voisinage connu est un problème naturel ouvert. Un autre problème ouvert est la conception et l'analyse d'algorithmes d'échange d'information dans ce modèle.

Dans le chapitre 5, nous avons conçu et nous avons analysé, pour toute constante $\epsilon > 0$, un algorithme de synchronisation globale $\epsilon-$certain déterministe qui fonctionne en temps constant dans n'importe quel MAC avec pannes aléatoires qui utilise la communication par les bips. En tant qu'application, nous avons résolu le problème du consensus dans ce modèle. À l'aide de la synchronisation globale, nous avons donné un algorithme du consensus $\epsilon-$certain déterministe qui fonctionne en temps $O(\log w)$ dans un MAC avec pannes aléatoires, où w est la plus petite valeur d'entrée de tous les processeurs du canal. Nous avons montré que ce temps ne peut pas être amélioré, même si le MAC est sans pannes. Nos travaux laissent ouvertes plusieurs directions intéressantes pour de futures recherches. Une question est de savoir comment parvenir à une synchronisation globale dans les réseaux multi-sauts. On pourrait se demander comment concevoir un algorithme de synchronisation efficace si la probabilité p des pannes de transmission dans le canal n'est pas connue par les processeurs. On pourrait aussi penser à différents types de pannes. Plutôt que des pannes de brouillage affectant tout le canal, nous pouvons penser aux pannes qui surviennent sur des processeurs individuels, par exemple si un processeur qui tombe en panne ne transmet pas sur le canal ou si un processeur qui tombe en panne n'entend pas un bip transmis par le canal. On pourrait aussi élargir le MAC sujet aux pannes à des réseaux de topologie arbitraire avec pannes.

Dans le chapitre 6, nous avons conçu et nous avons analysé un algorithme d'élection du chef presque certain travaillant en temps $O(\log^2 N)$ dans n'importe

quel MAC avec pannes aléatoires, contenant N processeurs, qui utilise la communication par les bips. Nous avons montré la borne inférieure $\Omega(\log N)$, sur le temps d'élection du chef, même pour les MAC sans pannes. Un problème naturel ouvert est si l'on peut effectuer l'élection du chef presque certaine en temps $O(\log N)$ dans un MAC en présence des pannes aléatoires.

Bibliographie

[1] Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, F. Kuhn, Beeping a maximal independent set, Distributed Computing 26 (2013), 195-208.

[2] B. Awerbuch, O. Goldreich, D. Peleg, R. Vainish, A trade-off between information and communication in broadcast protocols, J. ACM 37 (1990), 238-256.

[3] R. Bar-Yehuda, O. Goldreich, A. Itai, On the time complexity of broadcast in radio network : an exponential gap between determinism and randomization, Journal of Computer and System Sciences 45 (1992), 104-126.

[4] B. Baker, R. Shostak, Gossip and telephones, Discr. Math. 2 (1972), 191-193

[5] T. Calamoneri, E.G. Fusco, A. Pelc, Impact of information on the complexity of asynchronous radio broadcasting, Proc. 12th International Conference on Principles of Distributed Systems (OPODIS 2008), 311-330.

[6] A. Casteigts, Y. Métivier, J. Robson, A. Zemmari, Deterministic Leader Election in O(D + log n) Time with Messages of Size O(1), Proc. 30th International Symposium on Distributed Computing (DISC 2016), 16-28.

[7] B. Chlebus, G. De Marco, D. Kowalski, Scalable Wake-up of Multi-Channel Single-Hop Radio Networks, Theoretical Computer Science 615 (2016), 23-44.

[8] B. Chlebus, G. De Marco, M. Talo, Naming a channel with beeps, arXiv :1608.04174v1 [cs.DC] (2016).

[9] B. Chlebus, L. Gasieniec, D. Kowalski, T. Radzik, On the wake-up problem in radio networks, Proc. 32nd Colluquium on Automata, Languages and Programming (ICALP 2005), 347-359.

[10] B. Chlebus, D. Kowalski, A better wake-up in radio networks, 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004), 266-274.

[11] B. Chlebus, M. Rokicki, Centralized asynchronous broadcast in radio networks. Theor. Comput. Sci. 383 (2007), 5-22.

[12] G. Chockler, M. Demirbas, S. Gilbert, N. Lynch, C. Newport, T. Nolte, Consensus and collision detectors in radio networks, Distributed Computing 21 (2008), 55-84.

[13] M. Chrobak, L. Gasieniec, W. Rytter, Fast broadcasting and gossiping in radio networks, J. Algorithms 43 (2002), 177-189.

[14] M. Chrobak, L. Gasieniec, W. Rytter, A randomized algorithm for gossiping in radio networks, Networks 43 (2004), 119-124.

[15] A. Cornejo, F. Kuhn, Deploying wireless networks with beeps, Proc. 24th International Symposium on Distributed Computing (DISC 2010), 148-162.

[16] A. Czumaj, P. Davis, Communicating with beeps, Proc. 19th International Conference on Principles of Distributed Systems (OPODIS 2015), 30-46.

[17] A. Czumaj, P. Davies, Faster Deterministic Communication in Radio Networks, arXiv : 1506.00853v5 [cs.DC] (2016).

[18] A. Czumaj, W. Rytter, Broadcasting algorithms in radio networks with unknown topology, Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003), 492-501.

[19] J. Czyzowicz, L. Gasieniec, D. Kowalski, A. Pelc, Consensus and mutual exclusion in a multiple access channel, IEEE Transactions on Parallel and Distributed Systems 22 (2011), 1092-1104.

[20] G. De Marco, Distributed broadcast in unknown radio networks, SIAM J. Comput. 39 (2010), 2162-2175.

[21] C. Dwork, Y. Moses, Knowledge and common knowledge in a byzantine environment : crash failures, Information and Computation 88(1990), 156-186.

[22] S. Elouasbi, A. Pelc, Deterministic rendezvous with detection using beeps, International Journal on Foundations of Computer Science 28 (2017), 77-97.

[23] U. Feige, D. Peleg, P. Raghavan, E. Upfal, Randomized broadcast in networks, Random Structures and Algorithmes 1 (1990), 447-460.

[24] G. Fertin, J. Peters, L. Raabe, C. Xu, Odd gossiping, Discrete Applied Mathematics 216 (2017), 550-561.

[25] K. Foerster, J. Seidel, R. Wattenhofer, Deterministic leader election in multhop beeping networks, Proc. 28th International Symposium on Distributed Computing (DISC 2014), 212-226.

[26] P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks, Disc. Appl. Math. 53 (1994), 79-133.

[27] M. Franceschetti, R. Meester, Random networks for communication, From statistical physics to Information systems, Vol. 24, Cambridge University Press (2007).

[28] E. Fusco, A. Pelc, Knowledge, level of symmetry, and time of leader election, Distributed Computing 28 (2015), 221-232.

[29] L. Gasieniec, A. Pelc, D. Peleg, The wakeup problem in synchronous broadcast systems, SIAM Journal on Discrete Mathematics 14 (2001), 207-222.

[30] M. Ghaffari, B. Haeupler, Near optimal leader election in multi-hop radio networks, Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013), 748-766.

[31] M. Ghaffari, B. Haeupler, M. Khabbazian, Randomized broadcast in radio networks with collision detection, Distrib. Comput. 28 (2015), 407-422. S

[32] S. Gilbert, C. Newport, The computaional power of beeps, Proc. 29th International Symposium on Distributed Computing (DISC 2015), 31-46.

[33] C. Glacet, A. Miller, A. Pelc, Time vs. information tradeoffs for leader election in anonymous trees, Proc. 27th annual ACM-SIAM symposium on Discrete Algorithms (SODA 2016), 600-609.

[34] M. Halldórsson, Invited paper : Models for wireless algorithms, IEEE 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2016), 377-381.

[35] M. Halldórsson, S. Holzer, N. Lynch, A local broadcast layer for the SINR network model, Proc. of the 2015 ACM Symposium on Principles of Distributed Computing (PODC 2015), 129-138.

[36] M. Halldórsson, P. Mitra, Nearly optimal bounds for distributed wireless scheduling in the SINR model, Distributed Computing 29 (2016), 77-88.

[37] S.M. Hedetniemi, S.T. Hedetniemi, A.L. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1988), 319-349.

[38] B. Huang, T. Moscibroda, Conflict resolution and membership problem in beeping channels, Proc. 27th International Symposium on Distributed Computing (DISC 2013), 314-328.

[39] T. Jurdzinski, D. Kowalski, T. Maciejewski, G. Stachowiak, Distributed broadcasting in wireless networks under the SINR Model, arXiv :1207.6732v2 [cs.DC] (2013).

[40] T. Jurdzinski, D. Kowalski, M. Rozanski, G. Stachowiak, On the impact of geometry on ad hoc communication in wireless networks, Proc. of the 2014 ACM Symposium on Principles of Distributed Computing (PODC 2014), 357-366.

[41] T. Kesselheim, B. Vöcking, Distributed contention resolution in wireless networks, 24th International Symposium on Distributed Computing (DISC 2010), 163-178.

[42] D. Kowalski, Algorithmic Foundations of Communication in Radio Networks, Morgan et Claypool Publishers, 2011.

[43] D. Kowalski, A. Pelc, Time of deterministic broadcasting in radio networks with local kownledge, SIAM Journal on Computing 33 (2004), 870-891.

[44] E. Kushilevitz, Y. Mansour, An Omega(D log (N/D)) lower bound for broadcast in radio networks, SIAM Journal on Computing 27 (1998), 702-712.

[45] Y. Métivier, J. Robson, A. Zemmari, On distributed computing with beeps, arXiv :1507.02721 [cs.DC] (2015).

[46] Y. Moses, M. Raynal, Revisiting simultaneous consensus with crash failures, Journal of Parallel and Distributed Computing 69 (2009), 400-409.

[47] A. Motskin, T. Roughgarden, P. Skraba, L. Guibas, Lightweight coloring and desynchronization for networks, Proc. of the 28th IEEE International Conference on Computer Communications (INFOCOM 2009), 2383–2391.

[48] M. Pease, R. Shostak, L. Lamport, Reaching agreement in the presence of faults, *J. ACM* 27 (1980), 228-234.

[49] A. Pelc, Activating anonymous ad hoc radio networks, Distributed Computing 19 (2007), 361-371.

[50] A. Pelc, Fault-tolerant broadcasting and gossiping in communication networks, Networks 28 (1996), 143-156.

[51] D. Peleg, Time-optimal leader election in general networks, Journal of Parallel and Distributed Computing 8 (1990), 96-99.

[52] S. Reddy, D. Kowalski, S. Vaya, Multi-Broadcasting under the SINR model, Proc. of the 2016 ACM Symposium on Principles of Distributed Computing (PODC 2016), 479-481.

[53] P.J. Slater, E.J. Cockayne, S.T. Hedetniemi, Information dissemination in trees, SIAM Journal on Computing 10 (1981), 692-701.

[54] D. Willard, Log-logarithmic Selection Resolution Protocols in a Multiple Access Channel, SIAM J. on Computing 15 (1986), 468-477.