République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Béjaia

Faculté des Sciences Exactes

Département d'Informatique

Mémoire de fin de cycle

En vue d'obtention du diplôme de Master Recherche en Informatique

Option : *R*éseaux et *S*ystèmes *D*istribués

Thème :

Le problème du consensus dans les réseaux mobiles ad hoc

Réalisé par :

Mlle Sahli Kahina

Mlle Kerbouche Lydia

Soutenu devant le jury composé de :

Président :	Mr Amad Mourad	**MCA U.A/Mira de Béjaia**
Rapporteur :	Mr Saadi Mustapha	**MAA U.A/Mira de Béjaia**
Examinatrice :	Mme Yaici Malika	**U.A/Mira de Béjaia**

Promotion : 2015/2016

Remerciements

Nous remercions Allah le tout puissant, qui nous a donné la force et la patience pourl'accomplissement de ce travail.

nous tenons à remercier chaleureusement notre promoteur, Mr Saadi Mustapha de nous avoir proposé ce projet, en nous faisant confiance, ainsi pour avoir dirigé ce travail avec ses orientations, ses précieux conseils et remarques. Nous vous remercions infiniment Monsieur pour le temps que vous avez consacré à ce modèste travail, votre gentillesse, votre patience et votre modestie.

Nos remerciements s'adressent également aux membres du jury pour l'immense honneur qu'ils nous ont fait en acceptant d'évaluer ce travail.

Enfin, que tous ceux qui ont contribué de prés ou de loin, par leurs encouragements et conseils à l'accomplissement de ce travail, trouvent ici l'expression de nos profondes reconnaissances.

Dédicaces

Je dédie ce modeste travail aux personnes les plus chères à mes yeux mes parents.

A ces deux grands cœurs qui m'entourent toujours par leur tendresse et leur affection.

A ceux qui m'ont toujours encouragée et soutenue dans mes études et m'ont éclairée et ouvert la vie de l'avenir.

Je vous dédie le fruit de mes efforts, comme un symbole de gratitude.Que Dieu vous me garde et que vous soyez toujours fières de moi.

A tous ceux qui me sont chers, qui m'ont aidée par leur soutien moral , en particulier

A mes trés chères soeurs et frères

qui occupent une place particulière dans mon cœur

A tous mes amis, pour leur amitié, leur soutien moral, et leur conseils en particulier

A mes trés chères amies : Lydia, Meriem et Salima.

En souvenir de nos éclats de rire et des bons moments

en souvenir de tout ce qu'on a vécu ensemble, j'éspère de tout mon cœur que notre amitié durera éternellement.

Kahina

Dédicaces

Je dédie ce modeste travail aux personnes les plus chères à mes yeux mes parents.

A ces deux grands cœurs qui m'entourent toujours par leur tendresse et leur affection.

A ceux qui m'ont toujours encouragée et soutenue dans mes études et m'ont éclairée et ouvert la vie de l'avenir.

Je vous dédie le fruit de mes efforts, comme un symbole de gratitude.Que Dieu vous me garde et que vous soyez toujours fières de moi.

A tous ceux qui me sont chers, qui m'ont aidée par leur soutien moral ,

en particulier

A mes trés chères soeurs et frères

qui occupent une place particulière dans mon cœur

A tous mes amis, pour leur amitié, leur soutien moral, et leur conseils

en particulier

A mes trés chères amies : Kahina, Meriem et Salima.

En souvenir de nos éclats de rire et des bons moments

en souvenir de tout ce qu'on a vécu ensemble, j'éspère de tout mon cœur que notre amitié durera éternellement.

Lydia

Table des matières

Remerciements **I**

Dédicaces **II**

Dédicaces **III**

Liste des figures **VII**

Liste des tableaux **VIII**

Liste des abréviations **IX**

Introduction générale **1**

1 Les systèmes distribués **3**

1.1 Introduction 4

1.2 Généralités sur les systèmes distribués 4

1.2.1 Système distribué 4

1.2.1.1 Avantages et inconvénients des systèmes distribués 4

1.2.1.2 Inconvénients des systèmes distribués 5

1.2.2 Modèles temporels d'un système distribué 5

1.2.2.1 Le modèle synchrone 5

1.2.2.2 Modèle asynchrone 6

1.3 Défaillances 6

1.4 Problèmes d'accord 7

1.5 Le consensus 8

1.5.1 Impossibilité de FLP 8

1.5.2 Approches utilisées pour résoudre le consensus 9
1.5.3 Méthodes utilisées pour contourner FLP 9
1.5.3.1 Randomisation 10
1.5.3.2 Les détecteurs de défaillances non fiables 10
1.6 Réseau mobile 11
1.6.1 Réseau mobile avec infrastructure 12
1.6.2 Réseau mobile sans infrastructure (Ad hoc) 13
1.6.2.1 Les caractéristiques des réseaux ad hoc (MANETs) 14
1.7 Conclusion 14

2 Le consensus dans les réseaux mobiles 15
2.1 Introduction 16
2.2 Protocoles pour les environnements filaires 16
2.2.1 Les protocoles basés sur le générateur de nombre aléatoire 16
2.2.2 Les protocoles basés sur les détecteurs de défaillances non fiables . 17
2.2.2.1 Protocole de Chandra et Toueg 17
2.2.2.2 Protocole HMR 19
2.2.3 Les protocoles basés sur le leader 20
2.3 Protocoles du consensus dans les réseaux mobiles 21
2.3.1 Protocoles du consensus dans les réseaux mobiles avec infrastructure 22
2.3.1.1 Le protocole BHM 22
2.3.2 Protocoles du consensus dans les réseaux mobiles sans infrastructure (MANETs) 23
2.3.2.1 Le protocole HCP 24
2.3.2.2 Le protocole HCD 26
2.4 Conclusion 35

3 Proposition 36
3.1 Introduction 37
3.2 Motivations et Contribution 37
3.3 Description de l'approche proposée 39
3.3.1 Modèle du système 39
3.3.2 Algorithme proposé 42
3.3.3 Exemple illustratif 44

3.3.4 Preuves 50
3.4 Conclusion 51

4 Simulation et étude des performances 52
4.1 Introduction 53
4.2 Environnement de simulation 53
4.3 Les paramètres de simulation 54
4.4 Les étapes de simulation 55
4.4.1 Initialisation des variabes de simulation 56
4.4.2 Déploiement du réseau 56
4.4.3 Détection des nœuds défaillants 57
4.4.4 Application de l'algorithme de clusterisation 58
4.4.5 Consensus 58
4.4.6 Les métriques d'évaluation de performances 58
4.5 Résultats et interprétations 58
4.6 Conclusion 62

Conclusion générale et perspectives 63

Bibliographie 65

Table des figures

1.1 Classification des défaillances de processus. 7
1.2 Les classes de détecteurs de défaillances. 11
1.3 Les modèles des réseaux mobiles. 12
1.4 Le modèle de réseaux mobile avec infrastructure. 13
1.5 Le modèle de réseaux mobile sans infrastructure. 13

2.1 Les phases du protocole de Chandra et Toueg [11]. 18

3.1 Le réseau modélisé. 44
3.2 Un coordinateur qui tombe en panne. 45
3.3 L'envoie du message de proposition. 46
3.4 Fausse suspicion du coordinateur. 47
3.5 Les clusterheads qui tombent en panne. 48
3.6 Les clusterheads dynamiques. 49
3.7 La transmission de la décision. 49
3.8 La transmission de la décision. 50

4.1 Les principales fonctions du simulateur. 55
4.2 Déploiement du réseau. 57
4.3 Le nombre de tours exécutés en fonction des nœuds. 59
4.4 Le nombre de tours exécutés en fonction de défaillances. 60
4.5 Le nombre de messages envoyés en fonction de nœuds. 61
4.6 Le nombre de messages envoyés en fonction de défaillances. 62

Liste des tableaux

4.1 Paramètres de simulation. 54
4.2 Table d'échéancier1. 54
4.3 Table d'échéancier2. 54
4.4 Structure d'un nœud. 56

Liste des abréviations

BHM : Badache, Hurfin and Macedo

BS : Base Station

CT : Chandra and Toueg

FLP : Fisher, Lynch and Peterson

HCD : Hierarchical Consensus with Dynamic clusterhead

HCP : Hierarchical Consensus Protocol

HMR : Hurfin, Mostefaoui and Raynal

MANET : Mobile Ad hoc Network

MH : Mobile Host

MSS : Mobile Support Station

NM : Nombre de Messages

NT : Nombre de Tours

PDA : Personal Digital Assistance

Introduction générale

1. Contexte

Les applications informatiques réparties sont omniprésentes dans divers domaines. Suite au développement récent des technologies réseaux, la conception de services nouveaux destinés à être exécutés dans des environnements distribués hétérogènes connaît un essor important dans de nombreux domaines industriels et plus particulièrement dans le domaine des télécommunications.

Cet avènement des systémes répartis, qui a été rendu possible grâce aux progrès technologiques réalisés, est également le fruit de l'intense activité de recherche qui au cours de ces dernières années a eu pour objectif de maîtriser les problèmes fondamentaux propres aux systèmes répartis. Au sein de la classe des problèmes d'accord, le problème du consensus fait figure d'exemple, il représente le plus grand dénominateur commun des différents problèmes d'accord tels que la cohérence des données, l'appartenance au groupe, la diffusion atomique... Malheureusement, il a été prouvé qu'il est impossible d'implémenter le consensus dans un système distribué totalement asynchrone (FLP) sujet aux défaillances. Parmis les approches proposées pour surmonter le FLP, la solution apportée par Chandra et Toueg est basée sur les détecteurs de défaillances non fiables.

Ces dernières années, de nouvelles architectures réparties ont été apparus comme les réseaux mobiles sans fil qui sont caractérisés par une forte mobilité des noeuds et une grande flexibilité d'emploi. Avec l'avance rapide des appareils portables, ainsi que des progrès des communications sans fil, le réseau mobile ad hoc (MANET) attire de plus en plus d'attention en raison de son déploiement rapide, son architecture flexible et son entretien facile. Les réseaux mobile ad hoc ouvrent de nouvelles opportunités dans la conception des applications réparties. Les problèmes de coordination distribués comme par exemple le consensus doit trouver de nouvelles spécifications et solutions adaptées à

la mobilité du système. Pour cela nous avons cité quelques approches qui ont été conçu pour contourner l'impossibilité FLP dans les réseaux ad hoc. Dans ce travail, nous nous intéressons au probléme du consensus dans les MANETs, où beaucoup de protocoles ont été proposés. Dans ce contexte, nous allons proposer un protocole de consensus qui permet de réduire le côut de messages échangés donc, réduire le nombre de tours.

2. Organisation du mémoire

L'organisation de ce mémoire refléte la démarche que nous avons adopté lors de la réalisation de ce travail. Pour cela notre projet est organisé comme suit :

Le premier chapitre présente les bases d'un système distribué en mettant en valeur les propriétés qui ont une influence sur sa sûreté de fonctionnement et notamment la tolérance aux fautes et cite quelques aspects lié au problème du consensus et il rappelle les différents concepts liés aux environnements mobiles tout en se basant sur les réseaux ad hoc.

Le deuxiéme chapitre est consacré à l'étude du problème du consensus dans un environnement mobile (réseau ad hoc) et aux différents protocoles hiérarchique tel que HCP,HCD proposés dans la littérature pour le résoudre.

Le troisième chapitre concerne le protocole que nous avons proposé qui est sujet du consensus dans les réseaux ad hoc, après avoir donné un aperçu sur le protocole (HCD) visé à être amélioré et les divers protocoles qui ont été proposés dans ce sens.

Le quatrième chapitre, quant à lui, est consacré à la comparaison des deux protocoles (HCD et le protocole proposé) en utilisant le simulateur MATLAB, dont les résultats de simulation ont été retournées afin d'être discutées.

Enfin, une conclusion qui, synthétise le travail et présente les perspectives envisagées.

1

Les systèmes distribués

1.1 Introduction

L'apparition des systèmes distribués répond à plusieurs limites des systèmes centralisés. Ils ont été conçus essentiellement dans le but de réaliser des systèmes à haute disponibilité et à grande capacité d'évolution permettant la résolution des problèmes complexes en un temps minime. Les systèmes distribués regroupent donc toutes les fonctionnalités nécessaires au partage des ressources, à la facilité d'extension du système et à la communication entre utilisateurs.

Un des avantages les plus importants de la distribution est la tolérance aux fautes qui est obtenue par la technique de réplication, cela peut entraîner un problème de coordination entre les données et les tâches répliquées. Ce dernier est généralement résolu par l'utilisation des protocoles d'accord qui peuvent être exprimés à partir du problème du consensus.

L'un des modèles de système les plus répandus dans les environnements actuels est le modèle sans fil. Les réseaux sans fil sont en plein développement du fait de leur flexibilité et de leur interface, qui offre à l'utilisateur la mobilité.

Dans ce chapitre, nous introduisons d'abord des généralités sur les systèmes distribués. Nous présenterons ensuite le problème de consensus et les différentes approches utilisées pour le résoudre. Nous terminerons par présenter les concepts des environnements mobiles et en particulier les réseaux ad hoc.

1.2 Généralités sur les systèmes distribués

1.2.1 Système distribué

Un système distribué est composé d'un ensemble fini de n sites reliés par un réseau de communication. Chaque site a une mémoire locale et exécute un ou plusieurs processus. Les processus synchronisent et communiquent entre eux en échangeant des messages à travers des canaux du réseau sous-jacent [21].

1.2.1.1 Avantages et inconvénients des systèmes distribués

- **Performance**

  l'amélioration du temps d'exécution d'une tâche par l'adjonction de plusieurs machines ou par la décomposition d'une tâche en sous tâches et leurs exécutions

en parallèle.

- **Coût**

  Un ensemble de nœuds reliés par un réseau d'interconnexion est moins chers qu'un gros ordinateur.

- **Tolérance aux pannes**

  C'est le fait qu'un système peut continuer à fournir un service malgré les défaillances d'un ou plusieurs processus du système.

- **Partage des données et des périphériques**

  Plusieurs utilisateurs peuvent accéder à une base de données partagée et des périphériques chers.

- **Scalabilité**

  Un système est dit scalable s'il est capable d'évoluer en puissance, le plus souvent par ajout ou remplacement de composants (processeurs, mémoire).

1.2.1.2 Inconvénients des systèmes distribués

- **Cohérence des données**

  Le partage des données dans un système distribué génère des difficultés pour assurer l'accès et la cohérence de ces objets partagés.

- **Sécurité**

  La sécurité pose un problème dans le système distribué du fait qu'ils permettent un accès facile à toutes les données du système y compris les données confidentielles.

1.2.2 Modèles temporels d'un système distribué

Les modèles de synchronisme établissent des propriétés temporelles pour la réalisation des tâches effectuées par les processus. Ils sont caractérisés par l'existence ou l'absence des hypothèses temporelles sur les délais de transmission des messages et les vitesses relatives des processus.

1.2.2.1 Le modèle synchrone

Dans un système synchrone on précise les bornes inférieures et supérieures de temps pour l'exécution des actions des processus. Il est caractérisé par l'existence des hypothèses temporelles sur les vitesses relatives des processus et les délais de transmission

des messages.

L'inconvénient du modèle synchrone est qu'il exige une maîtrise complète de la charge des entités impliquées (processus et canaux de communication). Cependant lorsque la moindre hypothèse temporelle du modèle est violée, les algorithmes basés sur ce modèle de système feront preuve de comportement incorrecte (perte de la sûreté) [7].

1.2.2.2 Modèle asynchrone

Un système est dit asynchrone s'il n'y a aucune borne sur les délais de transmissions des messages, les vitesses relatives des processus et les dérives des horloges (absence totale des hypothèses temporelles).

Les systèmes asynchrones sont simples et très proches des systèmes réels. L'avantage majeur d'utilisation du modèle asynchrone est que n'importe quel algorithme conçu sous ces hypothèses préserve sa correction (propriété de sûreté) dans n'importe quel autre modèle de système. Donc dans la conception des systèmes distribués actuel, l'utilisation d'un modèle n'ayant aucune synchronie est une approche tirée réaliste.

Due à sa faiblesse (l'absence d'hypothèses temporelles), il est difficile pour le modèle asynchrone de développer des solutions. Par conséquent, tout algorithme conçu pour ce modèle peut être utilisé dans tout autre [7].

1.3 Défaillances

Défaillance des processus

Un processus est défaillant lorsqu'il ne respecte pas ses spécifications durant l'exécution du système. On distingue plusieurs catégories de défaillances [23] :

- **Défaillance par arrêt définitif (crash fault)**

  Dans ce type de défaillance un processus s'arrête prématurément et ne fait rien à partir de ce point et il ne peut pas reprendre son exécution (à cause de la perte de tout ou d'une partie de son code par exemple).

- **Défaillance par omission**

  Ce cas un processus défaillant peut omettre certaines actions. Ces actions concernent par exemple l'envoi ou la réception des messages.

- **Défaillance de performance**

  Cette défaillance se caractérise par un non respect des contraintes temporelles comme par exemple les délais d'exécution des tâches.

– **Défaillance arbitraire (byzantine)**

Les défaillances byzantines sont définies en terme de déviation d'un processus de l'algorithme qu'il exécute. Par exemple, il diffuse un message M à un sous ensemble de processus. Pour d'autres processus, il envoie M' ou bien il envoie un message qui n'est pas prévu dans l'algorithme.

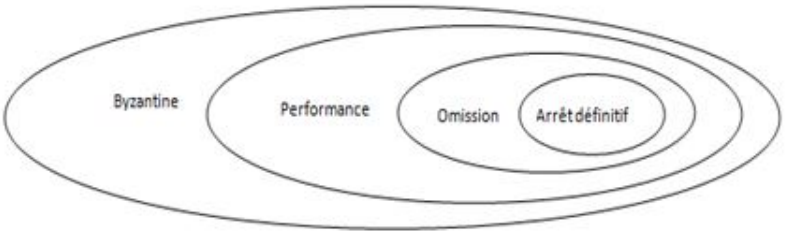

FIGURE 1.1 – Classification des défaillances de processus.

Remarque

La défaillance par arrêt définitif est un cas particulier de la défaillance par omission. Cette dernière est un cas particulier de la défaillance de performance alors que les défaillances arbitraires incluent tous les types de défaillances.

1.4 Problèmes d'accord

Les problèmes d'accord sont des problèmes liés à la mise en œuvre d'un service distribué qui doivent coopérer et synchroniser les tâches et les actions effectuées sur les données répliquées. Il faut de ce fait assurer que toutes les entités (les machines du système) soient cohérentes, autrement dit qu'elles se mettent d'accord.

Catégories de problèmes d'accord

– **Accord sur l'accès à une ressource partagée**

Exclusion mutuelle : Lorsque plusieurs processus veulent accéder à une même ressource, un processus désirant entrer en section critique doit demander un ensemble de permissions à d autres processus, Autrement dit, un processus demandeur ne pourra accéder à la ressource critique que s il a reçu un accord de tous les processus [12].

– **Accord sur un processus jouant un rôle particulier**

Election d'un maître : Un maître est un processus non fautif élu par les processus

du système et joue le rôle d'un coordinateur qui permet à un ensemble de processus de coopérer et contrôler le travail de plusieurs processus [17].

- **Accord sur une valeur commune**

  Consensus : Il s'agit de mettre d'accord toutes les entités du système sur l'une des valeurs proposées par ceux ci [12].

- **Accord sur l'ordre d'envoi de messages**

  Diffusion atomique : Elle implique que toutes les entités du système délivrent le même ensemble de messages dans le même ordre [10].

- **Accord sur une action à effectuer**

  Validation atomique : Action réalisée dans le cadre d'une transaction distribuée. C'est un accord entre tous les processus pour effectuer la requête de la transaction ou pas. L'accord se passe de manière atomique pour éviter toute interférence avec d'autres transactions [14].

1.5 Le consensus

Le consensus est une brique de base fondamentale dans les systèmes distribués pour la résolution des problèmes d'accord. Le problème du consensus s'énonce simplement : chaque processus correct propose une valeur et tous ces processus corrects doivent décider de manière irrévocable d'une valeur commune parmi toutes celles proposées. La solution au problème de consensus nécessite l'implémentation de deux primitives propose () et décide (). La proposition d'une valeur se fait grâce à la primitive propose (). La primitive décide () permet quant à elle de décider d'une valeur.

La spécification du problème du consensus s'exprime selon les propriétés suivantes :

- **Terminaison :** Chaque processus correct doit finalement décider d'une valeur.
- **Validité :**Si un processus décide d'une valeur v c'est que v a été proposée par un des processus.
- **Accord :**Si un processus correct décide d'une valeur v, de même si un autre processus correct décide une valeur v' alors v'= v.

1.5.1 Impossibilité de FLP

La spécification de problème du consensus peut être implémentée dans les deux modèles de systèmes. Cependant dans le système asynchrone, le consensus est difficile à

résoudre même en présence de défaillance. L'impossibilité de détecter avec certitude la défaillance d'un processus engendre l'impossibilité à résoudre des problèmes importants dans lequel il y'a une nécessité d'accord entre tous les processus.

Un résultat d'impossibilité de consensus dans un système réparti asynchrone est défini par Fisher, Lynch et Peterson [14]. En effet, ce résultat montre que le consensus n'as pas de solution déterministe surtout en présence d'une seule défaillance. De manière intuitive, ce résultat d'impossibilité est justifié par le fait qu'il est impossible de distinguer un processus lent d'un processus défaillant.

Afin de surmonter ce résultat d'impossibilité de FLP, il faut renforcer le modèle du système distribué asynchrone.

1.5.2 Approches utilisées pour résoudre le consensus

Il existe deux approches principales de contourner le résultat d'impossibilité FLP qui ont été proposées [5] :

1- D'imposer une certaine forme de contrainte temporelle sur le réseau.

2- L'allégement de la propriété de terminaison déterministe.

La première approche consiste à imposer des contraintes temporelles qui comprend l'introduction du modèle asynchrone temporisé [6], le modèle partiellement synchrone et les systèmes asynchrones avec un modèle de détecteurs de défaillance non fiable [23].

L'idée générale de ces approches consiste à supposer que le réseau passe par des périodes stables au cours de laquelle les progrès vers un consensus peut être fait. Parmi ces approches, l'approche du détecteur de défaillance non fiable est la plus générale. Nous allons donc en discuter d'avantage dans la section suivante.

Dans la deuxième approche, l'allégement de la propriété de terminaison déterministe, contourne le résultat d'impossibilité de FLP par l'affaiblissement de la propriété de terminaison avec une probabilité 1 (cela n'affect pas les propriétés de sûreté). Ce changement signifie qu'il existe encore des exécutions de consensus non-terminées mais elles se produisent avec une probabilité 0.

1.5.3 Méthodes utilisées pour contourner FLP

Modèle partiellement synchrone

Pour contourner cette impossibilité, il faut renforcer le modèle du système distribué asynchrone. Ce renforcement passe par un ajout suffisant de synchronie pour rendre le

problème soluble par l'ajout d'hypothèses temporelles (modèle partiellement synchrone) [4].

1.5.3.1 Randomisation

L'utilisation de la randomisation a été la première approche connue, proposée par Ben-Or [13], afin de contourner le résultat de l'impossibilité FLP. Cette approche suppose qu'il existe une source de caractère aléatoire dans le système de telle sorte que le choix des chemins d'exécution en certains points n'est pas déterministe, mais plutôt tirée d'une certaine distribution de probabilités.

L'ajout du caractère aléatoire au système de cette manière permet l'étude d'une distribution de probabilité de mauvaises exécutions. Si la propriété de vivacité du protocole est affaiblie pour exiger une résiliation éventuelle, les résultats d'impossibilité FLP ne tiennent plus. En exigeant seulement la terminaison éventuelle, les exécutions non-terminées continuent d'exister mais avec une probabilité 0 [5].

1.5.3.2 Les détecteurs de défaillances non fiables

Un détecteur de défaillance non fiable est une abstraction qui a été introduite par Chandra et Toueg dans le contexte de pannes franches [23]. Les détecteurs de défaillance fournissent des vues approximatives des processus défaillants. Les détecteurs de défaillance sont des modules associés à chaque processus. Ils fournissent une liste de processus suspectés. Dans un système asynchrone, les détecteurs de défaillance sont qualifiés de non fiables car ils peuvent commettre des erreurs. Autrement dit, un détecteur de défaillance peut soupçonner un processus correct, ou il peut ne pas suspecter un processus défaillant. Cependant, les détecteurs de défaillance doivent fournir des informations correctes. Les informations fournies sont définies en termes de complétude et de précision [5].

La complétude est une propriété de vivacité qui assure que les processus défaillants finiront par être suspectés. La précision est une propriété de sûreté qui restreint les suspicions erronées sur des processus corrects. Ces deux propriétés sont utilisées comme une base pour la conception des implémentations pour les détecteurs de défaillances.

La complétude

Le détecteur doit détecter les processus fautifs. Deux propriétés de complétude sont définies :

- **Complétude forte :** Tout processus fautif finira par être suspecté par tous les processus corrects.
- **Complétude faible :** Tout processus fautif finira par être suspecté par au moins un processus correct.

La précision

La précision exprime le degré de fiabilité dans la non suspicion des processus corrects. Idéalement, seuls les processus défaillants doivent être détectés comme tels. Il existe quatre propriétés de précision qui sont :

- **Précision forte :**Les processus corrects ne sont suspectés à aucun moment.
- **Précision faible :** Il existe un processus correct qui n'est jamais suspecté.
- **Précision ultime forte :** Il existe un instant à partir duquel aucun processus correct n'est plus jamais suspecté.
- **Précision ultime faible :** Il existe un instant à partir duquel il existe un processus correct qui n'est plus jamais suspecté par aucun processus correct.

Plusieurs classes de détecteurs de défaillance ont été définies par Chandra et Toueg à partir des propriétés de complétude et de précision. On regroupe en classes les détecteurs de défaillance ayant les mêmes propriétés. En combinant les deux propriétés complétude et précision, on obtient huit classes de détecteurs de défaillances [23].

complétude	précision			
	forte	faible	Ultime forte	Ultime faible
Forte	P	S	◊P	◊S
Faible	Q	W	◊Q	◊W

FIGURE 1.2 – Les classes de détecteurs de défaillances.

tableau comparatif entre les réseaux avec

1.6 Réseau mobile

Un réseau mobile est un réseau qui se compose des hôtes mobiles. Ils utilisent souvent une communication sans fil qui permet à ses utilisateurs d'accéder à l'information indépendamment de leurs positions physiques. Les environnements mobiles offrent aujourd'hui une bonne alternative de communication à moindre coût et à grande flexibilité d'emploi.

Les réseaux mobiles peuvent être classés en deux catégories en fonction de la manière de communication entre les hôtes mobiles : les réseaux avec infrastructure et les réseaux ad hoc [14].

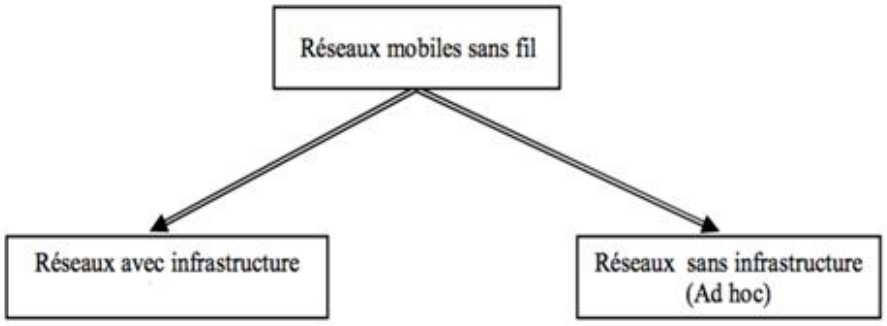

FIGURE 1.3 – Les modèles des réseaux mobiles.

1.6.1 Réseau mobile avec infrastructure

Un réseau avec infrastructure se compose de deux ensembles distincts d'entités : un grand nombre de hôtes mobiles (MHs) et des stations fixes relativement moins nombreuses mais plus puissantes que les stations support pour mobile (MSSs) ou station de base (BS). Les MSSs sont munies d'une interface de communication sans fil pour permettre la communication directe avec les MHs, localisées dans une zone géographique limitée appelé " cellule ". Dans ce réseau un MH ne peut être, à un instant donné, directement connecté qu'à son MSS locale.

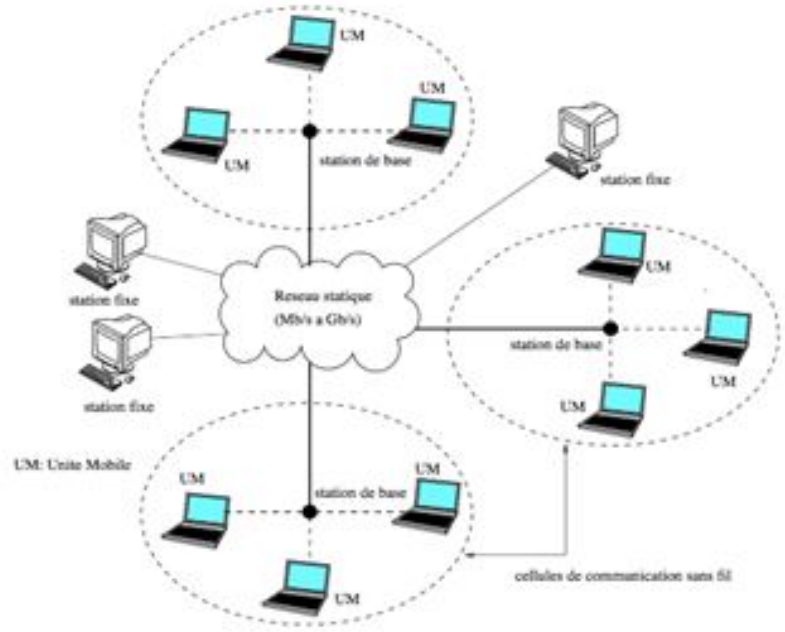

FIGURE 1.4 – Le modèle de réseaux mobile avec infrastructure.

1.6.2 Réseau mobile sans infrastructure (Ad hoc)

Contrairement aux réseaux avec infrastructure, un réseau mobile ad hoc (MANET) se compose d'une collection de MHs autonomes communiquant entre eux par des canaux sans fil. Les MANETs sont des réseaux multi-saut ce qui provoque un changement arbitraire de la topologie d'une façon dynamique en raison de la mobilité des MHs et des défaillances des hôtes [13].

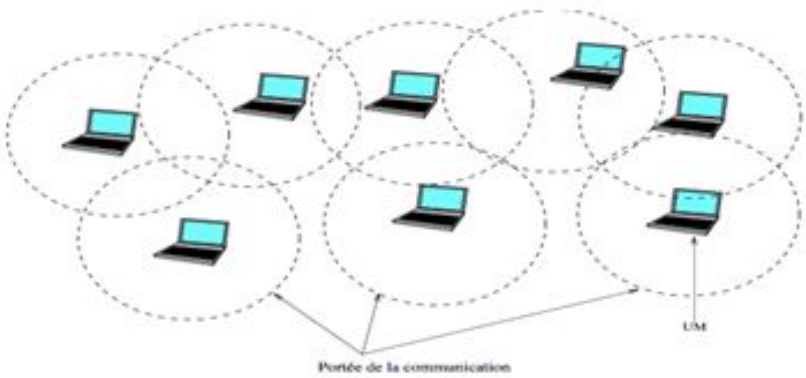

FIGURE 1.5 – Le modèle de réseaux mobile sans infrastructure.

1.6.2.1 Les caractéristiques des réseaux ad hoc (MANETs)

Les réseaux ad hoc ont plusieurs caractéristiques particulières :

- **L'absence d'infrastructure :** La différence principale entre les réseaux ad hoc et les autres réseaux mobiles est l'absence de toutes infrastructures ou administration centralisée préexistante. Pour cela, l'établissement et le maintient de la connectivité du réseau sont réalisés par des hôtes mobiles.
- **Ressources limitées :**Elles sont alimentées par des sources d'énergie limitées (batteries) tel que les Personal Digital Assistance(PDA) qui sont des équipements typiquement limités en énergie et par conséquent d'une durée de traitement réduite.
- **Topologie dynamique :** Les nœuds mobiles du réseau se déplacent d'une façon libre, ce qui fait que la topologie du réseau peut changer rapidement et de façon aléatoire à des instants imprévisibles.
- **Vulnérabilité des nœuds :** Les réseaux sans fil sont plus sensibles aux problèmes de sécurité que les réseaux filaires. Pour les réseaux ad hoc, le principale problème ne se situe pas tant au niveau du support physique mais principalement dans le fait que tous les nœuds sont équivalents et potentiellement nécessaire au fonctionnement du réseau.
- **Multihops :** Un réseau ad hoc est qualifié de "multihops" car plusieurs nœuds mobiles peuvent participer au routage et servent comme routeurs intermédiaires.
- **Bande passante limitée :** Une des caractéristiques primordiales des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé.

1.7 Conclusion

Dans ce chapitre nous avons présenté les différents modèles de défaillances, puis nous avons abordé quelques problème d'accord qui peuvent être réduits au consensus et l'impossibilité de résoudre ce dernier dans un un modéle asynchrone (résultat de FLP). Ensuite, nous avons cité les différentes approches pour le contourner. Enfin, nous avons décrit le domaine des réseaux mobiles qui est en plein essor. Il ouvre des perspectives de recherches importantes et une palette de nouvelles applications surtout très intéressantes en mobilité. Dans le chapitre suivant nous nous intéressons à l'étude du consensus dans les réseaux ad hoc(MANETs).

2

Le consensus dans les réseaux mobiles

2.1 Introduction

Dans le chapitre précédent, on a montré l'importance du consensus dans la construction d'un système distribué sûr. Le problème du consensus est un problème fondamental dans la conception des systèmes tolérants aux défaillances. L'étude de ce problème a permis de le résoudre dans plusieurs modèles de systèmes en particulier le modèle asynchrone avec pannes franches. Dans ce modèle, le problème du consensus était contraint par le résultat d'impossibilité de FLP.

Plusieurs solutions ont été proposées pour contourner cette impossibilité dont celle apportée par Chandra et Toueg (le protocole CT) basé sur les détecteurs de défaillances non fiables. Le résultat de CT a été utilisé pour résoudre le consensus dans un environnement mobile en tenant compte de la mobilité des MSSs et MHs.

Dans ce chapitre, nous donnerons un aperçu sur le protocole de Chandra et Toueg et le protocole HMR qui permettent de palier à l'impossibilité de FLP dans les réseaux filaires puis nous expliquerons le principe du protocole BHM qui est une extension du CT adapté aux environnements mobiles. Ensuite nous allons détailler deux approches hiérarchiques pour résoudre le problème du consensus dans les réseaux ad hoc appelé HCP et HCD qui sont fondés sur le protocole HMR et qui se distinguent par leur modularité et efficacité.

2.2 Protocoles pour les environnements filaires

Pour contourner le résultat d'impossibilité de FLP décrit auparavant, trois protocoles ont été proposés [23, 13] :

- Le protocole basé sur le générateur de nombre aléatoire.
- Le protocole basé sur les détecteurs de défaillance non fiable.
- Le protocole basé sur l'oracle d'élection.

2.2.1 Les protocoles basés sur le générateur de nombre aléatoire

Le premier protocole de consensus basé sur le générateur de nombres aléatoires est proposé dans [13]. Il est exécuté dans des tours asynchrones, mais aucun processus n'agit comme un coordinateur ou un leader dans un tour. Un processus met à jour son estimation à l'aide d'un générateur de nombres aléatoires. Fondamentalement, la primitive Random () délivre en sortie la valeur 0 (respectivement 1) avec une probabilité de 1/2, de sorte

que la solution permet de résoudre le consensus binaire, la valeur de décision est 0 ou 1.

Ezhilchelvan et al. [20] ont proposé un protocole de consensus multi-valeurs fondé sur un générateur de nombres aléatoires.

D'abord, chaque processus diffuse de manière fiable la valeur vi proposée par lui-même et rassemble les valeurs des autres. Ensuite, les processus exécutent des tours consécutifs asynchrones jusqu'à ce qu'une décision soit prise. Un tour est constitué de deux phases de communication.

- **La phase 1 :** Les processus échangent leurs propres estimations actuelles. Si un processus découvre que la majorité des estimations ont la même valeur v, il met à jour son estimation à v ; sinon, il met à jour son estimation à $\perp$ (une valeur qui ne peut pas être décidée).
- **La phase 2 :** Les processus entrent dans la deuxième phase au cours de laquelle ils échangent le nouveau contenu de leurs variables esti. Si un processus reçoit la même valeur v de la majorité des processus tel que v $\neq \perp$, il décide de v. Dans le cas contraire, si toutes les valeurs reçues sont $\perp$ alors le processus adopte une valeur d'estimation choisie arbitrairement parmi les valeurs reçues au début de l'exécution. Les processus qui n'ont pas décidé, ils passent au tour suivant.

L'inconvénient est que ces protocoles ne peuvent pas garantir la terminaison d'une manière déterminée.

2.2.2 Les protocoles basés sur les détecteurs de défaillances non fiables

La plupart des protocoles de consensus sont basés sur le détecteur de défaillance de la classe $\diamondsuit$ S ou S. Cependant, il a été rapporté que la mise en œuvre de S est aussi difficile que le problème du consensus lui-même [16], qui prétend que les solutions basées sur S ne sont pas rentables. Par conséquent, on s'intéresse aux protocoles basés sur $\diamondsuit$ S.

2.2.2.1 Protocole de Chandra et Toueg

Un certain nombre de protocoles de consensus ont été développés en utilisant des détecteurs de défaillances. Afin de résoudre le consensus, un protocole a été proposé par Chandra et Toueg qui est basé sur $\diamondsuit$ S [23].

Dans cette approche, tous les messages sont dirigés vers et envoyés par le coordinateur

courant. Une exécution du protocole CT se produit dans des tours asynchrones, où chaque tour est composé de quatre phases consécutives. Chaque processus pi maintient une estimation de la valeur de décision esti (initialement, la valeur d'estimation est égale à la valeur vi proposé par pi) [23].

La structure d'un tour r est décrite dans la figure 2.1 et expliquée dans ce qui suit[11] :

- **La phase 1 :** Chaque processus pi envoie sa propre valeur d'estimation est$i$ au coordinateur courant. Durant le premier tour, le processus pi propose sa valeur initiale vi.
- **La phase 2 :** Le coordinateur collecte la majorité des valeurs d'estimation des processus et il sélectionne l'estimation dont l'estampille de temps est la plus grande. Ensuite le coordinateur suggère cette estimation en l'envoyant à tous les processus.
- **La phase 3 :** Chaque processus attend une nouvelle valeur de la part du coordinateur. Il doit préparer une réponse à envoyer au coordinateur.Pour ce faire, pi interroge son détecteur de défaillances locales. Si les informations fournies par le détecteur de défaillances suspecte le coordinateur, alors pi envoie un accusé de réception négatif (un Nack) pour le coordinateur actuel. Sinon, pi reçoit le message du coordinateur, puis il met à jour son estimation à est$c$ et l'estampille au numéro du tour actuel r et il envoie un accusé de réception positif (un Ack) au coordinateur.
- **La phase 4 :** Si l'ensemble des réponses recueillies par le coordinateur ne contient que des (Ack), alors ce dernier diffuse de manière fiable un message de décision, sinon il procède au prochain tour.

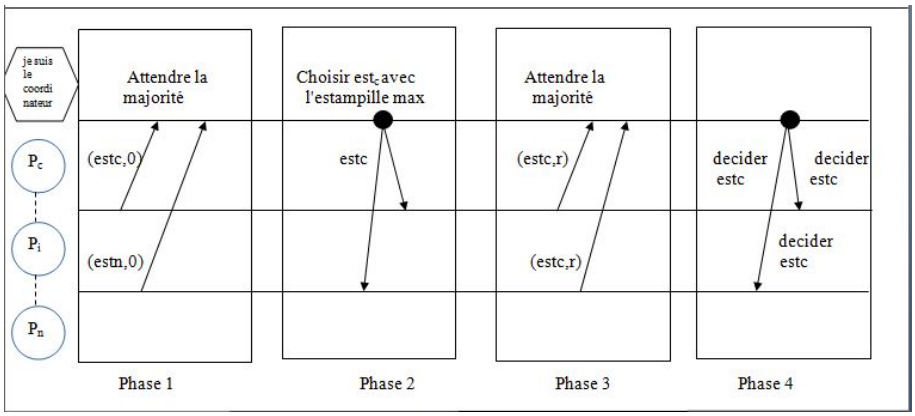

FIGURE 2.1 – Les phases du protocole de Chandra et Toueg [11].

2.2.2.2 Protocole HMR

Le protocole HMR [15] présente une approche d'unification pour parvenir à un consensus avec deux dimensions polyvalentes :

- La classe du détecteur de défaillance (Classe S ou $\diamondsuit$ S).
- Le modèle d'échange de message (d'un modèle centralisé à un modèle entièrement distribué) à chaque tour.

Comme tous les protocoles basés sur les détecteurs de défaillances, HMR utilise le paradigme de coordinateur altéré et il s'exécute dans des tours asynchrones. Pour la prise de décision et la maintenance de la propriété d'accord, le protocole HMR repose sur un ensemble de processus noté DA qui est l'union de deux ensembles decision_makers et les agreement_keepers.

– D (Decision_makers) est lié à la propriété de vivacité du protocole. D est l'ensemble des hôtes qui a besoin de vérifier l'état de la décision, à savoir s'ils peuvent décider dans le tour actuel. D est défini par les deux exigences suivantes :
  - D est déterministe, tous les processus ont le même D pour le même tour
  - D contient le coordinateur actuel .

– A(Agreement_keepers) est lié à la propriété de sûreté du protocole. A est utilisé pour assurer que, une fois qu'une valeur a été décidée dans un tour, aucune autre valeur ne peut être décidée dans les tours suivants.

  Chaque tour est exécuté en deux phases :

  Dans la première phase

  Le coordinateur est défini par une fonction déterministe (un coordinateur pour chaque tour), il envoie son estimation à chaque processus (y compris lui-même). Les processus attendent la valeur d'estimation du coordinateur sauf s'il est suspecté (il passe au tour suivant dans ce cas). Lorsque l'estimation est reçue, les processus mettent à jour leurs propres estimations à la valeur reçue.

  Dans la deuxième phase

  Le modèle d´échange de message ECHO est déterminé par deux ensembles de processus, D et A. Dans cette phase, chaque processus envoie un message ECHO à chaque processus dans DA. Chaque processus dans DA attend jusqu'à ce qu'il reçoit des messages ECHO d'au moins de n-f processus, puis il définit son estimation à la valeur portée par le message ECHO avec la plus haute estampille.

L'estampille indique le dernier tour au cours duquel son estimation actuelle a été mise à jour.

Un processus de l'ensemble D vérifie les estampilles des messages écho reçus. S'il a reçu f + 1 messages d'écho avec une estampille de temps identique au tour courant, alors il peut décider. Une fois qu'il a décide, il diffuse un message de décision portant sa valeur de décision à l'aide d'un mécanisme de diffusion fiable, et ensuite il arrête sa participation au protocole.

Par rapport aux autres protocoles existants, HMR est simple mais il est polyvalent et inefficace en terme de coût de message.

Un inconvénient commun des protocoles basés sur les détecteurs de défaillances est causé par le paradigme de coordinateur tournant. Un protocole de consensus ne peut pas parvenir à un consensus jusqu'à ce qu'il y 'ait un tour coordonné par un hôte non suspecté. Vu que les hôtes prennent leurs tours pour agir en tant que coordinateur de quelque tour dans un ordre prédéfini, les hôtes doivent encore exécuter un tour qui est coordonné par un hôte suspect ; ce qui retardera la prise de décision.

2.2.3 Les protocoles basés sur le leader

Ω est un détecteur de défaillance 'leader', il offre à chaque processus une primitive qui retourne l'identité d'un processus à chaque fois qu'elle est appelée. Il a été introduit par Neiger dans [8].Il a été montré dans [3] que Ω est le détecteur minimal pour résoudre le problème du consensus dans un système asynchrone avec une majorité de processus corrects.

Les protocoles [19] [1] peuvent garantir les trois propriétés d'exactitude. Comme tous les protocoles basés sur les détecteurs de défaillances, ils sont exécutés dans des tours asynchrones, mais il n'y a pas de coordinateur prédéfini pour un tour.

Chaque tour du protocole dans [1] se compose de trois phases.

- **La phase 1 :** Chaque processus diffuse sa valeur d'estimation pour les autres et puis attend les valeurs du processus leader (sélectionné par l'oracle de leader). Si une telle valeur est reçue, le processus met à jour sa valeur d'estimation à la valeur reçue et diffuse la nouvelle valeur reçue de son leader (le cas échéant) ou la valeur "$\perp$" (une valeur qui ne peut jamais être acceptée, ce qui signifie que le processus échoue à obtenir une valeur de son leader).

- **La phase 2 :** Au cours de la deuxiéme phase si un processus a reçu la même valeur v tel que($v \neq \perp$). Il diffuse la valeur v ; Sinon, il diffuse la valeur "$\perp$".
- **La phase 3 :** Enfin, sur la base des messages reçus dans la troisième phase, un processus met à jour sa propre estimation et prend une décision s'il reçoit la même valeur ($v \neq \perp$) de la majorité des processus.

Le protocole dans [19] utilise une procédure similaire, mais les deux premières phases sont fusionnées en une seule.

Un inconvénient causé par ce changement est que, dans le même tour, les leaders choisis par les divers hôtes peuvent être différents. Pour faire face à cela, les informations de leader doivent être échangées au même temps que les messages de la proposition, qui n'est pas si flexible que les protocoles basés sur les détecteurs de défaillances en termes de modèle d'échange de message.

2.3 Protocoles du consensus dans les réseaux mobiles

Les réseaux mobiles sans fil ont des propriétés fondamentalement différentes des réseaux filaires traditionnels dans les aspects de la communication, la mobilité et les contraintes de ressources, qui rendent la conception des protocoles distribués beaucoup plus difficile que dans des systèmes distribués traditionnels.

Dans ce modèle de réseaux, les hôtes mobiles (MHs) ont besoin d'un traitement spécial, pour plusieurs raisons :

- Premièrement, les MHs sont généralement moins robustes que les hôtes fixes (MSSs), en plus ils consomment beaucoup d'énergie.
- Deuxièmement, les liens sans fil sont plus lents et moins fiables par rapport aux liens câblés, ils subissent la perte de signal dû au bruit.
- Troisièmement, les MHs peuvent subir des défaillances soudaines. La connexion est souvent limitée par la bande passante, ce qui implique des périodes de déconnexion du réseau fixe, même lorsque l'hôte est connecté.

Lors de la conception des protocoles distribués pour un environnement mobile, les facteurs suivants doivent être pris en considération [3] :

- Réduire la consommation d'énergie des MHs ;
- Réduire le coût de communication totale dans le médium sans fil ;
- La gestion de connexions et de déconnexions fréquentes des MHs.

L'environnement mobile est sujet à de nouveaux défis pour la conception des

protocoles afin de résoudre le consensus. La plupart des approches conçues pour résoudre le problème de consensus dans les réseaux mobiles sont des extensions des protocoles principaux cités ci-dessus qui ont été proposés et développés dans les réseaux filaires.

2.3.1 Protocoles du consensus dans les réseaux mobiles avec infrastructure

Quelques protocoles ont été proposés pour résoudre le problème de consensus dans les réseaux mobiles avec infrastructure. Le principe de la conception de ces protocoles est de transférer les opérations des MHs aux MSSs pour parvenir à un consensus [17, 9]. Parmi les protocoles de consensus basés sur ce principe, on cite le protocole BHM.

2.3.1.1 Le protocole BHM

Pour répondre aux contraintes en énergie des MHs et à la faible bande passante des médiums sans fil, une solution simple a été proposée par Badache et al dans [17] qui consiste à exécuter le protocole du consensus au niveau des MSSs [10].

Le protocole BHM est une adaptation du protocole Chandra et Toueg [23] dans les environnement mobiles.

Dans ce protocole, les MSSs collectent les valeurs initiales proposées par les MHs et exécutent un protocole de consensus pour leur compte.
L'activité d'une MSS est divisée en trois sous-tâches principales :

- Interagir avec ses MHS locales pour collecter leurs valeurs initiales ;
- Interagir avec d'autres MSSs pour se mettre d'accord sur un sous-ensemble des valeurs proposées ;
- Interagir avec ses MHS locales pour diffuser le résultat final.

Une simple procédure handoff est utilisée pour gérer les mouvements des MHs.

La solution répond aux caractéristiques des réseaux mobiles mais elle a des inconvénients :

- La composition du groupe ne change pas, cette hypothèse ne reflète pas l'aspect dynamique des groupes.
- Le protocole se termine lorsqu'un nombre prédéfini de sites mobiles participent. Cette condition viole la propriété de terminaison liée au consensus si le nombre de sites mobiles qui participent est inférieur au nombre considéré. Il est plus intéressant que ce nombre soit variable et reflète les conditions de transmissions

et l'état des processus dans le système (correct ou défaillant).

- Le protocole utilise la station de base pour se charger du calcul de différents tours et communication relatifs au protocole de consensus. Cela n'est pas toujours valable. Les stations de base peuvent ne pas être conçues comme des unités de calcul.

2.3.2 Protocoles du consensus dans les réseaux mobiles sans infrastructure (MANETs)

L'efficacité en nombre de messages échangés est essentielle pour concevoir des algorithmes efficaces pour les environnements mobiles. A la différence des réseaux sans fil avec infrastructure, les MANETs, n'ont pas de MSS et tout doit être fait par les MHs. Le principe utilisé dans la conception des protocoles de consensus des réseaux avec infrastructure n'est pas applicable dans les MANETs.

Les caractéristiques du protocole HMR en terme de simplicité, d'utilisation de deux types de messages uniquement ainsi que la simplicité de traitement dans chaque phase rendent le protocole approprié et facile à intégrer dans les MANETs. En outre, en raison de la polyvalence, HMR peut donner lieu à une famille nombreuse et bien identifié des protocoles basés sur les détecteurs de défaillances[17]. Cependant, HMR n'est pas efficace en termes de coût du message. A chaque tour, le coordinateur doit envoyer la même proposition à chaque MH et chaque MH doit envoyer le même écho à chaque processus dans DA. Par conséquent, la hiérarchie à deux couches a été proposée pour améliorer l'efficacité du message.

L'idée de base est d'imposer une hiérarchie à deux couches sur les hôtes mobiles en les regroupant en clusters. L'approche de clusterisation a été adoptée et largement utilisée dans les MANETs pour réduire le coût du message et de parvenir à un consensus. Tous les messages envoyés par les hôtes qui se trouvent dans le même cluster doivent être fusionnés par le clusterhead correspondant puis délivrés à d'autres hôtes. De cette façon, le coût du message peut être considérablement réduit.

Sur la base de différentes approches, deux protocoles hiérarchiques ont été proposés, ils sont basés sur une variante du protocole polyvalent (HMR) [15]. Le premier protocole est le HCP (Hierarchical Clusterhead Protocol),il est basé sur un ensemble statique de clusterheads. Tandis que le second protocole est le HCD (Hierarchical Consensus with Dynamic Clusterhead) dont les clusterheads sont sélectionnés dynamiquement. Les deux

protocoles seront détaillés dans ce qui suit.

2.3.2.1 Le protocole HCP

Un protocole HCP basé sur les détecteurs de défaillances non fiables de la classe $\diamondsuit$ P, est un protocole de consensus conçu pour les MANETs. La hiérarchie à deux couches dans le protocole HCP est similaire à celle du protocole BHM, Les clusterheads sont équivalents aux MSSs dans BHM, ils agissent comme des hôtes privilégiés. Cette hiérarchie est définie comme suit :

- La couche clusterhead : Elle se compose d'un ensemble prédéfini de MHs qui agissent comme des clusterheads. Ils permettent de fusionner et échanger des messages en provenance des MHs. Seulement les hôtes clusterheads qui peuvent agir comme des coordinateurs, decision_makers, ou agreement_keepers.
- La couche hôte : Elle se compose d'un ensemble de MHs, y compris ceux de l'ensemble de clusterheads.

Cependant, il y a beaucoup de différence entre BHM et HCP. Dans BHM, les MHs fournissent des valeurs initiales et ils ne participent pas au protocole pour parvenir à un consensus, mais dans HCP, tous les hôtes doivent exécuter des tours d'échange et de traitement de message.

Le protocole se compose de quatres tâches :

- Tâche 1 est le corps principal du protocole (tâche réalisant le consensus).
- Tâche 2 est un simple algorithme de diffusion de la valeur décidée.
- Tâche 3 est utilisée comme moyen de gérer les messages en retard ECHOL.
- Tâche 4 concerne le mécanisme handoff.

Tâche1

Cette tâche se compose de deux phases :

La phase1

Au début d'un tour, le coordinateur courant envoie un message de proposition noté PROPG contenant sa valeur d'estimation aux clusterheads. A la réception de PROPG, chaque clusterhead transmet le message à tous ses hôtes locaux. Si un clusterhead suspecte le coordinateur avant la réception de PROP, alors il leur envoie un message proposition contenant la valeur $\perp$. un hôte attend le message de son clusterhead local et met à jour sa valeur d'estimation et son estampille, dans le cas où son clusterhead est suspecté, il fait appel à la procédure «handoff» qui sera présenté dans la tâche 4.

La phase 2

Chaque hôte envoie d'abord un message ECHOL à son clusterhead local. Ensuite il entre dans le tour suivant. Chaque clusterhead attend donc un message ECHOL de chacun de ses hôtes locaux qui ne sont pas suspectés, puis il construit un message ECHOG par la fusion des messages collectés contenant : la valeur d'estimation avec la plus haute estampille de temps portée par le message ECHOL, l'ensemble des hôtes qui a envoyé les messages ECHOL avec cette estampille et l'ensemble des hôtes qui a envoyé les messages ECHOL avec d'autres estampilles. Le clusterhead envoie ECHOG à l'ensemble DA.

Chaque hôte dans DA attend les messages ECHOG , on distingue deux cas

1. S'il reçoit les messages ECHOG comportant n-f hôtes (les hôtes qui ont envoyé les messages ECHOL à leurs clusterheads avec la plus haute estampille ainsi que les hôtes qui ont envoyé les messages ECHOL avec d'autres estampilles). Il met à jour sa valeur d'estimation à la valeur portée par le message ECHOG avec la plus haute estampille de temps. Il vérifie s'il peut décider dans le tour courant. S'il y a (f+1) hôtes ou plus dans les ensembles avec estampille égal à r, alors il prend la décision sur cette valeur et diffuse la valeur finale.
2. S'il reçoit un message ECHOG avec une estampille >r , il entre dans le tour suivant.

Tâche2

Elle consiste à diffuser simplement la valeur de décision. Lorsqu'un hôte qui n'a pas encore décidé reçoit un message DECISION, il transmet le message DECISION à tous les autres hôtes sauf l'expéditeur et décide de la valeur DECISION.

Tâche3

Elle consiste à gérer les messages ECHOL en retard. Un message ECHOL est "en retard" s'il arrive à un clusterhead qui a envoyé un message ECHOG pour le tour correspondant. Si un message ECHOL en retard est ignoré, alors les clusterheads de l'ensemble DA peuvent se bloquer définitivement. Pour éviter ceci, quand un clusterhead p reçoit un ECHOL avec une estampille inferieure à rp (rp le tour courant de p) ou ils sont égaux mais p a envoyé le message ECHOG pour ce tour alors p construit un message ECHOG pour lui et l'envoie à tous les clusterheads.

Tâche4

Cette tâche est appelée lorsqu'un hôte suspecte son clusterhead ou il n'est pas le plus proche clusterhead. Lorsqu'un hôte change de clusterhead, il se peut que ces deux hôtes seront dans des tours et des phases differentes, le but de la tâche 4 est de gèrer la

synchronisation des hôtes locaux avec leurs nouveaux clusterheads.

Malgré la performance et l'efficacité du protocole HCP en terme de coût de messages mais il présente plusieurs inconvénients :

1) La fonction de parvenir à un consensus est étroitement couplé avec la fonction de clusterisation. Quand un MH se déplace à un nouveau cluster, son exécution doit être changée en respectant l'état du nouveau clusterhead. Une telle conception rend le protocole compliqué.
2) L'ensemble des clusterheads est prédéfini, alors il ne peut pas adapter au crash qui se produit durant l'exécution ; ce qui retarde la prise de décision.
3) Le protocole nécessite un détecteur de défaillance de classe $\diamondsuit$ P, qui est plus puissant que le détecteur de défaillance le plus utilisé $\diamondsuit$ S [23].

2.3.2.2 Le protocole HCD

Pour répondre aux problèmes cités ci-dessus, un deuxième protocole hiérarchique a été proposé appelé HCD, il est constitué d'un ensemble de clusterheads dynamique. Il permet d'obtenir le consensus en utilisant les clusters construit par l'éventual clusterer $\diamondsuit$ C. La fonction de clustering des hôtes appelé $\diamondsuit$ C et la fonction de parvenir à un consensus sont séparées en utilisant une approche modulaire.

- **Eventual clusterer $\diamondsuit$ C**

  L'Eventual clusterer $\diamondsuit$ C est un outil qui fournit certain type d'information sur le système, il est basé sur le détecteur de défaillances $\diamondsuit$ S. $\diamondsuit$ C établit deux couches hiérarchiques en regroupant les hôtes de MANET en cluster, chacun de ces clusters est géré par un clusterhead et chaque hôte est associé à un module d'oracle Eventuel clusterer. Le module d'oracle retourne trois sorties [25] :

  1. $\diamondsuit$ C.CH : L'ensemble des MHs qui agissent comme des clusterheads.
  2. $\diamondsuit$ C.trusted : L'ensemble des hôtes corrects.
  3. $\diamondsuit$ C.clusterhead : Le clusterhead local de l'hôte mi.

1) **Les propriétés de $\diamondsuit$ C**

- **La complétude :** Il y a un instant après lequel certain hôte correct est inclus d 'une façon permanente dans l'ensemble $\diamondsuit$ C.CH et l'ensemble $\diamondsuit$ C.trusted.

- **La précision :** Chaque hôte qui tombe en panne est exclu d'une façon permanente de l'ensemble ♢ C.CH et l'ensemble ♢ C.trusted.
- **L'uniformité :** Tous les hôtes corrects maintiennent le même ensemble des clusterheads ♢ C.CH.
- **La stabilité :** Il y a un instant après lequel chaque hôte correct est toujours associé à certain clusterhead correct. Un hôte peut se déplacer de son clusterhead[25].

2) **L'implémentation de ♢ C**

La tâche 1 est consacrée à la construction de l'ensemble des clusterheads.

COBEGIN

the code executed by each host mi

/*———**Task 1 : Construction of Clusterhead Set CH** ———*/

CH ←M ; seq$i$ ← 0 ;

/*———**Task c1.1 : Send CH** ———————————*/

while(true){

CH ← CH ∩ ♢S.trusted ;

send (CH, seq$i$) to M ;

/*———**Task c1.2 : Receive CH**———————————-*/

upon reception of (CH', seq$q$) from host mq) ;

if (seq$q$ = seq$i$) CH ← CH ∩ CH ;

if (seq$q$ > seq$i$) CH ← CH' ; seq$i$ ← seq$q$;

if (CH = ∅){

CH ← M ;

seq$i$ ← seq$q$+1 ;

}

send (CH, seq$i$)to M ;

La tâche 2 consiste à construire les clusters. Chaque hôte dans ♢C.CH joue le rôle d'un clusterhead. Un hôte mi utlise la fonction NEAR ♢ C.CH pour sélectionner l'hôte mn le plus proche dans l'ensemble ♢ C.CH puis il lui envoie un mesage JOIN pour rejoindre le cluster.

Lors de la réception du message JOIN, si mn accepte d'être un clusterhead, il

envoie un ACK positif à mi ; sinon,il rejette la demande et il lui envoie un ACK négatif.

En raison des défaillances ou des fausses suspicions, un clusterhead peut être supprimé de l'ensemble ♢ C.CH implique que les membres du cluster correspondant nécessitent de se déplacer à un nouveau cluster. Lorsqu'un hôte détecte que son clusterhead courant est supprimé de ♢ C.CH ou il reçoit un message RELEASE de la part de son clusterhead courant, mi se déplace à un autre cluster en exécutant les instructions suivantes :

1. Si mi est le clusterhead courant qui a été supprimé, il envoie un message RELEASE à ses hôtes locaux, sinon il envoie un message LEAVE à son clusterhead courant.
2. Puis, mi sélectionne l'hôte le plus proche dans ♢ C.CH comme nouveau clusterhead et il lui envoie un message JOIN.
3. Si un ACK positif est reçu de l'hôte sélectionné, le déplacement est réussi ; sinon mi sélectionne un nouveau hôte et il lui envoie un message JOIN.

/*———**Task c2 : Clustering Host**———————————**-***/

clusterhead ← i ;rejected ← ∅ ;sn ←0 ;

/*———**Task c2.1 : Action of Cluster Member**———————*/

while (true){

if (clusterhead ∉CH , or clusterhead is no longer the nearest ;

or a RELEASE (sn') from mk with clusterhead = k) ;

if (clusterhead = i) send RELEASE (sn) to all local hosts execpt mi ;

else send LEAVE (sn) to clusterhead ;

sn ← sn + 1 ; rejected ← ∅ ;

if (CH\ rejected) = ∅ {

rejected← ∅ ;

sn ← sn+ 1 ;

clusterhead ← NEAR(CH\rejected) ;

send JOIN(sn) to clusterhead ;

wait until ACK(type, sn) received from clusterhead ;

if (ACK.type = false){

rejected ← rejected ∪ { clusterhead} ;

sn ← + 1 ; rejected ← ∅ ;

}
}
}
/*———**Task c2.2 : Action of Clusterhead**———————*/
while(true){
upon receiving JOIN (sn) from a host mk :
if (clusterhead ≠i) send ACK (false, sn) to mk ;
else {
add mk to local host list ;
Send ACK (true, sn) to mk ; }
upon receiving LEAVE(sn) from a host mk ;
if (clusterhead=i) delete mk from local host list ;
}

COEND

1. **Modéle du système**

   On considère un MANET qui se compose d'un ensemble fini de n hôtes mobiles M=*p1, p2,.... pn*, où n>1.

   L'ensemble des hôtes est statique et connu par tous les hôtes.

   Tous les hôtes sont distribués dans des clusters.

   Certains MHs sont sélectionnés comme des clusterheads, chacun se charge d'un cluster.

   Chaque paire de MHs est relié par un canal fiable (il n y a pas de perte de messages).

   Le nombre maximal de hôtes défaillants f est délimité par n/2.

2. **Structures de données et types de messages**

   Lors de l'exécution du protocole HCD, chaque hôte a besoin de maintenir certaines informations nécessaires sur son état, ces informations sont stockées dans les variables suivantes [25] :

   fl$i$: La valeur booléenne indiquant si mi a pris la décision.

   ri : Le numéro de séquence du tour actuel auquel mi participe.

   est$i$: L'estimation actuelle de la valeur de décision.

   ts$i$: L'estampille de est$i$.

   Lors de l'exécution du protocole HCD, les MHs ont besoin de communiquer les uns

avec les autres en échangeant des messages. Les types de messages impliqués dans un tour ri sont énumérés comme suit :

PROPG (ri, estcc) : Le message de proposition envoyé de l'hôte coordinateur à tous les clusterheads.

PROP (ri, v) : Le message de proposition envoyé par un clusterhead à ses hôtes locaux.

ECHO (ri, est$i$, ts$i$) : Le message écho envoyé de mi à son clusterhead.

ECHOG(ri, estcc, W, Z) : Le message écho envoyé par un clusterhead au Decision_makers et Agreement_keepers. Un message ECHOG rassemble les messages ECHO envoyé par les hôtes. v est l'estimation portée par le message ECHO avec l'estampille la plus élevée et tsv est l'estampille de v. W est l'ensemble de MHs qui envoie le message ECHO avec l'estampille tsv, ou Z est l'ensemble de MHs qui envoie les autres messages ECHO.

DECISION (est) : Le message envoyé par un MH pour diffuser la valeur de décision est.

NEW R (r) : Le message utilisé pour mener les MHs à un nouveau tour.

3. **Le fonctionnement du protocole HCD**

Le protocole HCD se compose de quatre tâches qui sont exécutées indépendamment et simultanément à chaque hôte. La tâche 1 est le corps principal pour la prise d'une décision en échangeant les messages dans des tours. La tâche 2 décrit un algorithme simple de diffusion pour propager la valeur décidée. La tâche 3 est utilisée pour envoyer des messages de rédemption pour gérer les messages ECHO retardés tandis que la tâche 4 gère les messages délivrés à un hôte. Le code des différentes tâches est représenté dans la figure 2.4.

Tâche1

Chaque tour se compose de deux phases :

La phase 1

Si mi est un hôte coordinateur, il envoie le message PROPG(ri,est) à tous les clusterheads, sinon il envoie un message NEWR(ri) au coordinateur mcc. (Le message NEWR(ri) est utilisé pour réveiller le coordinateur dans le cas où il est bloqué dans un tour précédent en raison de pertes de messages). Chaque clusterhead attend le message PROPG de mcc. Lorsque le message PROPG de mcc est reçu, un clusterhead transmet le message à tous ses hôtes locaux. Dans le cas ou il suspecte le coordinateur, il envoie le message PROP(ri,$\perp$) à ses hôtes locaux. Chaque hôte attend un message PROP de son clusterhead. A la réception de ce message, l'hôte met à jour son estimation et son estampille.

La phase 2

Chaque hôte envoie un message ECHO(ri, est, ts$i$) à son clusterhead. Chaque clusterhead attend les messages ECHO de tous ses hôtes locaux corrects. A la réception du message, le clusterhead fusionne tous les messages ECHO reçus dans un message ECHOG (ri, estm, tsm, W, Z). Ensuite,il transmet ECHOG à tous les hôtes de l'ensemble DA.

Les hôtes dans DA attendent les messages ECHOG(ri,*,*,*, W, Z) jusqu'a ce que :

1) Les messages ECHOG reçus incluent au moins n-f hôtes.
2) Un message ECHOG (-, -, ts$v$, -, -) avec tsv<ri est reçu. A la réception du message, chaque hôte de l'ensemble DA met à jour son estimation à la valeur portée par le message ECHOG avec la plus haute estampille et prends une décision, puis, il l'envoie à tous les autres hôtes en utilisant le message DECISION.

```
 the code executed by each host, mi ;
/*———Task 1 : Consensus———————————-*/
 ri ← 0 ;esti ← v ; fli ← false ;
 while (fli≠ true){
  ri ← ri+1 ;
 cc= coord(ri) ; ch= ♢ C.clusterhead ;
/*———Phase1 : Collecting Proposal————————*/
  if (i=cc) send PROPG(ri), esti) to ♢ C.CH ;
  else send NEWR(ri)) to mcc) ;
  if (i=cc) {
 mi is a clusterhead ;
```

wait until (received a PROPG(ri, vcc) from mcc, or

mcc $\notin \diamondsuit$ C.trusted, or m$i \notin \diamondsuit$ C.CH) ;

if (PROP(ri), vcc) is received)

send PROP(ri, vcc) to all local hosts

else send PROP (ri,$\perp$) to all local hosts ;

}

wait until ((received PROP(ri,*) from mch or ch $\neq \diamondsuit$ C.clusterhead) ;

if(PROP(ri), v) with v$\neq \perp$ is received) }

est$i \leftarrow$ v ; ts$i$ = ri ;

}

/*———**Phase2 : Collecting Echo**———*/

send ECHO (ri, est$i$, ts$i$) message to mch ;

if (i=ch) {

wait until (received ECHOG(ri, * , *) from each local host m$j \in \diamondsuit$ C.truted) :

merge the ECHO messages received into an ECHOG (ri, estm, tsm, W, Z) ;

send ECHOG(ri, estm, tsm, W, Z) to DA ; }

if (m$i \in$ DA {

wait until receive ((ECHOG(ri, *, *, W, Z) with | $\cup$ W$\cup$ Z | — $\geq$ n-f or

some ECHOG (-, -, $\geq$ r, -, -)message ;

est$i \leftarrow$ vm ;

if tsvm=ri and $\geq$f+1 hosts are in the W sets with tsvm {

$\forall$ j$\neq$ i, send DECISION(est i) to mj ;

fl$i \leftarrow$ true ;

} }

} endwhile

while (true){

if (clusterhead $\notin$CH , or clusterhead is no longer the nearest ;

or a RELEASE (sn') from mk with clusterhead = k) ;

if (clusterhead = i) send RELEASE (sn) to all local hosts execpt mi ;

else send LEAVE (sn) to clusterhead ;

sn $\leftarrow$ + ; rejected $\leftarrow \emptyset$;

if (CH rejected) = $\emptyset$ {

rejected$\leftarrow \emptyset$;

sn ← sn+ 1 ;
clusterhead ← NEAR(CH\rejected) ;
send JOIN(sn) to clusterhead ;
wait until ACK(type, sn) received from clusterhead ;
if (ACK.type = false){
rejected ← rejected ∪ { clusterhead} ;
sn ← + 1 ;
rejected ← ∅ ;
}
}
}
/*———**Task2 : Reliable Broadcast**————*/
upon reception of DECISION (est) from host mi ;
if (fl$i$≠ true){
∀ j≠i, k ; send DECISION (est)to mj ;
fl$i$ ← true ; }

COEND

Tâche2

Cette tâche diffuse tout simplement la valeur de décision. Quand un hôte reçoit un message DECISION, il décide de la valeur portée par le message DECISION et transmet le message à tous les autres hôtes sauf l'émetteur.

Tâche3

Cette tâche envoie des messages résdemption à cause des messages ECHO retardés. Les Decision_makers ou Agreement_keepers peuvent être bloqué si un message ECHO est ignoré. Pour éviter cela, quand un hôte mj reçoit un message ECHO à partir d'un hôte pour le tour ri où (ri <rj) ou (ri = rj et mj a envoyé un message ECHOG pour le tour ri), mj construit un message de rédemption ECHOG et l'envoie aux Decision_makers ou Agreement_keepers du tour ri .

Un hôte mj peut se déplacer de son cluster en raison de défaillance de son clusterhead actuel. Lors du déplacement de son clusterhead, mj a besoin d'envoyer certain message de rédemption. Il retransmet les messages ECHO au nouveau clusterhead.

Tâche4

Cette tâche manipule les messages futurs. Un message est dit un message futur s'il arrive à un MH avant que le MH entre dans le tour correspondant du msg. Quand un hôte mi reçoit un message futur, dans ce cas il va sauter à un tour futur après l'exécution des opérations suivantes :

. Si mi est un clusterhead et le msg n'est pas un message NEWR, alors il envoie un message NEWR à ses hôtes locaux de sorte que les hôtes locaux peuvent également passer au tour futur.

. Si mi n'est pas dans l'ensemble DA du tour r-1, il envoie des messages ECHO pour les tours sautés puis il passe au tour r.

. Dans le cas contraire, si le tour r-1 est aussi un futur tour et mi est dans l'ensemble DA du tour r-1, mi envoie des messages ECHO pour les tours sautés puis il passe au tour r-1.

the code executed by each host, mj ;

/*———**Task3 : Sending Redeeming Messages**———*/

upon reception of ECHO(ri,*, *) from host mi, with ((ri¡ rj) or (ri=rj but mj has sent an ECHOG(ri,*,*,*))){

construct an ECHOG for the ECHO and sent it to DA of round ri ;

}

upon the change of local clusterhead {

if (mj has not entered Phase 2)

for (ts$j$≤ rr <rj) send ECHO(rr,est$j$,ts$j$) to the new clusterhead ;

else for (ts$j$≤ rr ≤ rj) send ECHO(rr,est$j$,ts$j$) to the new clusterhead ;

}

the code executed by each MH mi ;

/*———**Task4 : Handling Future Messages**———*/

upon reception of a message msg with r>ri : {

if (i=ch and msg is not a NEWR message)

send NEWR(ri) to local hosts ;

if (mi is not the DA set of round r-1) {

for (ri< rr <r) send ECHO (rr, est$i$, ts$i$) to mch ;

ri ←r ; ri ← ri+1 ;

}

else if (r-1)> ri{

for (ri< rr <r-1) send ECHO (rr, est$i$, ts$i$) to mch ;

ri ← r-1 ; ri ← ri+1 ;

}

Le protocole HCD est l'un des protocoles le mieux adapté aux réseaux mobiles ad hoc, il a pour but l'amélioration de l'efficacité des messages pour parvenir à un consensus.

2.4 Conclusion

Nous avons abordé dans ce chapitre certains protocoles pour résoudre le consensus dans l'environnement mobile. Nous nous intéressons aux protocoles HCP et HCD qui sont dédiés aux réseaux ad hoc. Ils sont basés sur deux couches hiérarchiques. Cette hiérarchie nous permet de réduire le coût dû au nombre de messages transférés. Le chapitre suivant sera destiné à la description de notre proposition pour résoudre le problème du consensus dans le réseau ad hoc.

3

Proposition

3.1 Introduction

Dans ce qui a précédé, nous avons vu plusieurs approches adaptées ou reconçues pour répondre aux exigences des environnements mobiles. La conception des protocoles de consensus hiérarchiques basés sur des détecteurs de défaillances avait efficacement simplifié la résolution du problème de consensus.

Nous avons consacré ce chapitre à la description de notre proposition à propos d'une solution au problème de consensus dans les réseaux ad hoc tout en prenant en considération les deux concepts de base sur lesquels reposent les différents protocoles proposés pour résoudre ce problème : le coût lié au temps d'exécution, ainsi que le coût en communication de messages. Nous présentons tout d'abord les motifs qui nous ont conduits à proposer cette solution. Ensuite, nous donnons une description générale du protocole proposé, puis le protocole lui-même dont nous présentons les structures de données et les messages utilisés. Enfin, nous terminons par les preuves de validité du protocole proposé.

3.2 Motivations et Contribution

Le consensus dans le réseau mobile est considéré primordial vu son développement étendu et la nécessité d'assurer la coordination au sein du réseau. Le problème du consensus a été largement étudié, essentiellement dans ce modèle de réseaux qui détient des propriétés différentes des réseaux filaires (communication, mobilité et contraintes de ressources). Ceci rend la conception d'algorithmes distribués beaucoup plus difficile. La majorité des approches proposées adoptent des principes similaires à celles développées pour les réseaux traditionnels. Cependant, afin de concevoir des protocoles qui résolvent le problème du consensus dans ces réseaux, on doit faire face à de nouveaux défis. D'abord, le protocole doit garantir la fonctionnalité de consensus en dépit du nombre de défaillances. Cela est vrai pour tout protocole fiable tolérant aux fautes. Ensuite, le protocole doit fournir sa fonctionnalité garantie avec les côûts généraux raisonnables pour maitriser les contraintes de ressources limitées et la mobilité.

La plupart des protocoles proposés sont basés sur le paradigme du coordinateur tournant et ils procèdent par des tours asynchrones ; chaque tour est coordonné par un processus prédéterminé qui tente d'imposer son estimation. L'approche hiérarchique a été très utilisée dans la conception de protocoles pour les MANETs pour réduire les coûts de

communication de manière à faire face aux contraintes de ressources des environnements mobiles (moins de messages consomment moins de ressources de bande passante et moins de calcul) et de parvenir à la stabilité et la scalabilité. Rappelons que HCD est un protocole de consensus hiérarchique qui utilise les messages futurs de manière à réduire le temps d'attente et accélérer l'exécution du protocole en réduisant le nombre de tours requis pour parvenir à une décision. Le protocole HCD est l'un des protocoles les plus récents et les plus adaptés aux réseaux mobiles. Cependant il présente plusieurs inconvénients :

Premierement, un coordinateur ne se permet pas de prendre une décision jusqu'à ce qu'il reçoit des messages de la majorité des hôtes du réseau et que ces hôtes décident durant le tour en cours, c'est-à-dire si la majorité des hôtes du réseau ne suspectent pas le coordinateur d'être défaillant. Ce qui fait qu'une fausse suspicion au niveau d'un hôte affecte rigoureusement la prise de décision dans le tour courant.

Deuxièmement, dans chaque tour le coordinateur envoie son estimation aux clusterheads qui sera transmise et imposée aux hôtes locaux avant que celle-ci ne sera définitivement prise comme une décision finale. Ce qui fait que lorsque la décision ne pouvait pas être prise au cours de ce tour, le coordinateur du prochain tour suivra la même procédure avant de s'assurer que son estimation sera la décision finale, ce qui induit à des traitements et des communications supplémentaires et inutiles.

Notre but dans ces recherches est d'étudier les principes et les performances des approches développées pour la résolution de consensus dans les environnements mobiles. Pour cela nous avons essayé d'apporter quelques modifications au protocole HCD visant à améliorer la résolution du problème de consensus et obtenir une solution simple et efficace en terme d'augmentation de la vitesse d'exécution du protocole.
Pour remédier à ces contraintes, nous avons proposé ces modifications :

Au lieu que le coordinateur attende les messages de la plupart des hôtes du réseau, le coordinateur attend seulement les messages de la part des clusterheads c'est-à-dire les fausses suspicions et les défaillances des hôtes locaux ne seront pas prises en considération par le coordinateur et les clusterheads prennent en charge la décision de ses hôtes locaux. De cette manière le nombre de tours sera considérablement minimisé.
Les clusterheads ne transmettent pas l'estimation du coordinateur jusqu'à ce qu'ils s'assurent qu'elle a été prise en décision finale. Autrement dit les clusterheads ne transmettent pas les estimations des coordinateurs, mais ils transmettent directement la décision finale.

Notre solution a donc pour objectif fondamental de réduire le nombre de communications et de tours qu'il faut pour que les hôtes atteignent le consensus, ce qui facilite d'acquérir la propriété d'accord et d'accélérer l'exécution du protocole.

3.3 Description de l'approche proposée

3.3.1 Modèle du système

Le modèle du système pour le protocole proposé est le même que le protocole HCD défini dans le chapitre précédant :

- **Le système**

  On considère un MANET qui se compose d'un ensemble fini de n hôtes mobiles M=*p1, p2,.... pn*, où n>1.

  L'ensemble des hôtes est statique et connu par tous les hôtes.

  Tous les hôtes sont distribués dans des clusters.

  Certains MHs sont sélectionnés comme des clusterheads, chacun se charge d'un cluster.

  Il n y a pas de limite de délais de message ou de la vitesse des hôtes.

- **Les Communications**

  Les processus se communiquent par l'envoi et la réception des messages.

  Chaque paire de MHs est relié par un canal fiable (il n y a pas de perte de messages).

- **Les défaillances**

  Les processus ne peuvent tomber en panne que des pannes franches.

  Le processus s'exécute correctement jusqu'à la panne.

  Le nombre maximal de processus clusterheads défaillants f est délimité par n/2, où n est le nombre de noeuds dans le réseau. Chaque hôte est associé à un module d'oracle eventual clusterer $\diamondsuit$C qui est basé sur les détecteurs de défaillances non fiables de classe $\diamondsuit$S.

- **Structures de données**

  fl$i$:La variable booléenne indiquant si mi a pris la décision. La valeur initiale est mise à faux.

  ri :Le numéro de séquence du tour actuel auquel mi participe

  est$i$:Chaque processus dispose d'une estimation de la valeur de décision finale.

est$i$ L'estimation actuelle de la valeur de décision. Dans un premier temps, il est proposé par la valeur mi.

– **Type des messages**

PROPG (ri, est$i$) : Le message de proposition envoyée par le coordinateur aux clusterheads.

DECISION (v) : Le message envoyé par un coordinateur pour propager la valeur de décision v aux clusterheads et qui est transmise par les clusterheads aux hôtes locaux.

ECHOG (ri, v) : Le message écho envoyé par les clusterheads aux hôtes decision_makers et agreement_keepers.

NEWR (ri) : Le message utilisé pour réveiller le coordinateur en cas où il est bloqué dans le tour précédent.

ROUND(ri) : Le message destiné pour échanger le tour courant d'un clusterhead.

– **Les opérations du protocole**

Nous avons vu dans le chapitre précédent que le protocole HCD se compose de quatre tâches. Nous avons enlevé les tâches trois et quatre. La tâche 3 est utilisée pour manipuler les messages retardés envoyés par les hôtes locaux à leurs clusterheads, alors que dans notre solution les hôtes locaux seront pris en charge par leur clusterheads associés. Donc cette tâche n'est pas nécessaire. Quant à la tâche 4, elle est utilisée pour manipuler les messages futurs. Dans notre solution le seul message futur est envoyé par un clusterhead à l'ensemble DA et celui-ci sera manipulé dans la phase 2 de la tâche 1. Cependant on a ajouté une tâche exécutée par les hôtes locaux afin de prendre la décision transmise par son clusterhead et une quatriéme tâche qui permet de gérer la synchronisation des nouveaux clusterheads. Les quatres tâches proposées sont définies comme suit :

– **Tâche 1**

C'est la partie principale du protocole permettant la prise de décision en échangeant des messages dans des tours. Comme il est défini dans le protocole HCD après avoir déterminé l'hôte coordinateur. Cette tâche est divisée en deux phases :

- Dans la phase 1, l'hôte coordinateur envoie le message PROPG (ri, est$i$) à tous les clusterheads et chaque clusterhead attend le message PROPG du coordinateur, une fois reçu il met à jour sa valeur d'estimation à la valeur du coordinateur apportée par ce message.

- La phase 2

  Elle commence par l'envoi de messages d'ECHO, dés qu'un clusterhead reçoit le message PROPG, il envoie un message ECHO (ri , v) à tous les hôtes de l'ensemble DA.

  Les hôtes de DA attendent les messages ECHO (ri,*) jusqu'a ce qu'ils recoivent les messages d'ECHO de la part de tous les clusterheads qui se présente dans le réseau ou ils reçoivent un message ECHO avec r> ri. Notons que grâce à ♢C, l'ensemble ♢C.CH est dynamique et contient uniquement les hôtes corrects non défaillant. Ensuite, si tous les messages ECHO reçus comporte le tour ri, les hôtes DA prennent la décision sur v et envoient v à tous les autres hôtes en utilisant le message DECISION (v) sinon, ils passent au tour suivant.

– **Tâche 2**

Il s'agit de diffuser la valeur de décision aux clusterheads. Quand un clusterhead reçoit un message de décision, il décide de la valeur portée par le message de décision et transmet le message à tous ses hôtes locaux.

– **Tâche 3**

Cette tâche est exécutée par les hôtes mobiles. A la réception du message de décision, ils décident de la valeur portée par celui-ci.

– **Tâche 4**

Cette tâche permet aux nouveaux clusterheads de rejoindre le tour courant associé aux clusterheads participants à la réalisation du consensus.

Cette tâche ne se présente pas dans le protocole HCD car les hôtes qui deviennent des clusterheads participent déjà au consensus, donc ils prennent le même tour et continuent leurs exécutions, en revanche, dans le protocole proposé les hôtes locaux ne participent pas à l'exécution du protocole de consensus donc quand ils deviennent des clusterheads, ils doivent commencer l'exécution du protocole par le premier tour ce qui tarde la réalisation du consensus. Cette tâche a donc pour but de synchroniser les nouveaux clusterhead avec les autres clusterheads du système et surtout d'accélérer l'exécution du protocole.

3.3.2 Algorithme proposé

La fonction de clustering est presque la même que la fonction "eventual clustering" définie dans le chapitre 2 afin d'assurer la prise en charge de la décision au niveau des hôtes locaux. Des modifications sont introduites dans la tâche action de clusterhead qui est définie de la manière suivante :

/*——————————**Tâche c2.2 : Action de clusterhead** ——————————-*

Tanque(vrai){

Lors de la réception du message **JOIN** (sn) de l'hôte mk :

si (clusterhead <>i) envoyer **ACK** (faux, sn) à mk ;

sinon { ajouter mk à la liste de hôtes locaux ;

Envoyer **ACK** (vrai, sn) à mk ;}

si (fl$i$ =vrai)

Envoyer **DECISION** (vi) à mk ;

Lors de la réception du message **LEAVE**(sn) de mk ;

si (clusterhead=i)

Supprimer mk de la liste de hôtes locaux ;

}

Concernant le protocole de consensus son algorithme est définit comme suit :

Algorithme

debut

Le code est exécuté par chaque clusterhead ch$i$

/*——————————**Tâche 1**——————————-*/

ri← 1 ;est$i$ ← vi ; fl$i$← faux

envoyer **ROUND**(ri) aux ♢ C.CH ;

attendre jusqu'à (recevoir **ROUND** (ri)depuis tous les clusterheads ch$i$∉ ♢C.trusted)

ri←r ;//r est le plus grand tour reçu

Tanque(fl$i$ <> vrai){

cc= coord (ri) ; Ch = ♢ C.clusterhead ;

/*——————————**Phase 1**——————————-*/

si (i = cc) envoyer **PROPG** (ri , est$i$) aux ♢C.CH ;

sinon envoyer **NEWR**(ri) au mcc ;

attendre jusqu'à (recevoir **PROPG** (ri, vcc)

de mcc ou mcc∉ ♢C.trusted, ou mi∉ ♢C.CH) ;

si (recevoir **PROPG**(ri,vcc) de mcc { est$i$=vcc ; }

/*————————————**Phase 2**————————————-*/

envoyer**ECHO**(ri,vcc) à DA ;

si (mi∈ DA) {

Attendre jusqu'à(recevoir **ECHO**(ri,v) depuis tous les clusterheads ou recevoir **ECHO**

si (au moins tous les messages avec r=ri) {

est$i$=v

Envoyer **DECISION**(est$i$)à ♢C.CH

fl$i$=v

}

}

ri ← ri+1 ;

}// Fin tant que

Exécuter par les clusterheads

/*————————————**Tâche 2**————————————-*/

Lors de la réception de message**DECISION**(v)

si fl$i$<> vrai){

Envoyer **DECISION**(v) aux hôtes locaux

fl$i$ ← vrai ;

}

Exécuter par les hôtes locaux

/*————————————**Tâche 3**————————————-*/

Attendre **DECISION**(v) ou (ch ∉ ♢C.Clusterhead)

si recevoir **DECISION**(v) {

est$i$←v

fl$i$ ← vrai ;

}

Exécuter par chaque clusterhead ch$i$

/*————————————**Tâche 4**————————————-*/

Lors de la réception du message **ROUND**(r) depuis ch$j${

envoyer **ROUND**(ri) à ch$j$;

}

– **Remarque :** Si un hôte local change de cluster courant, la décision sera envoyée

par le nouveau clusterhead à partir de la fonction de clustersation si la décision a été déjà prise, ou à partir de la tâche 2 dans le cas contraire.

3.3.3 Exemple illustratif

Dans ce qui suit, nous allons donner un exemple illustrant notre approche, en exploitant le réseau modélisé dans la figure suivante :

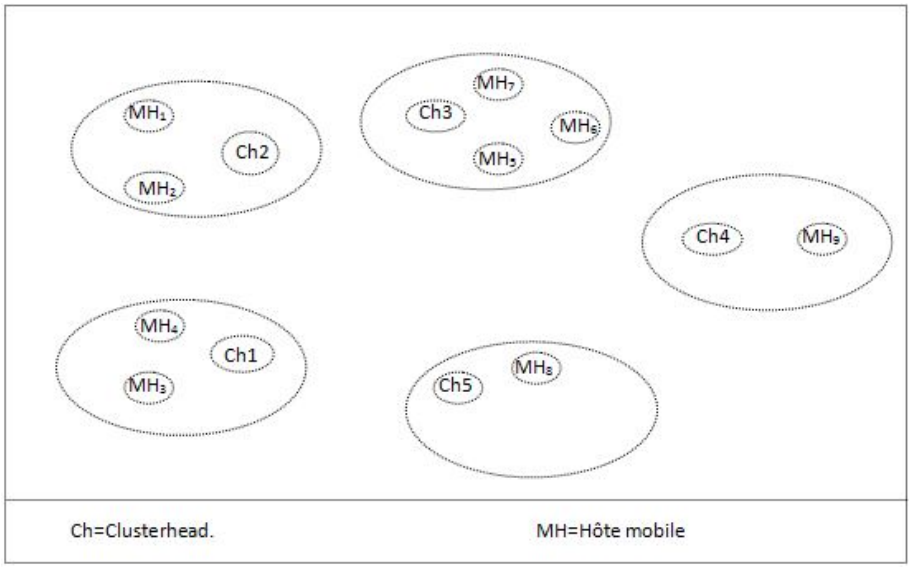

FIGURE 3.1 – Le réseau modélisé.

Dans cet exemple nous avons 14 hôtes, seules 6 défaillances sont tolérées au niveau des clusterheads.

Initialement :

r=1 pour tous les clusterheads.

Chaque clusterhead, aprés l'échange des numéros de tours entame le premier tour et séléctionne le coordinateur Ch1.

Initialement, l'ensemble DA comporte 2 nœuds Ch1 et Ch2(le coordinateur du prochain tour).

– **Premier cas de figure :** Supposons que le coordinateur tombe en panne, comme le montre la figure ci-dessous :

Les clusterheads suspectent le coordinateur.

1. Ils passent au tour suivant r2. Ils envoient à ch2 NEWR (r2).

2. Les hôtes locaux associés au clusterhead défaillant (Ch1) choisissent un nouveau clusterhead et rejoignent le nouveau cluster.

– **Remarque :** Puisque aucun des clusterheads de réseau n'a encore décidé alors lors du déplacement des hôtes locaux du clusterhead défaillant vers les autres clusterheads, seul l'ACK d'acceptation sera envoyé aux hôtes (fl$i$ = faux).

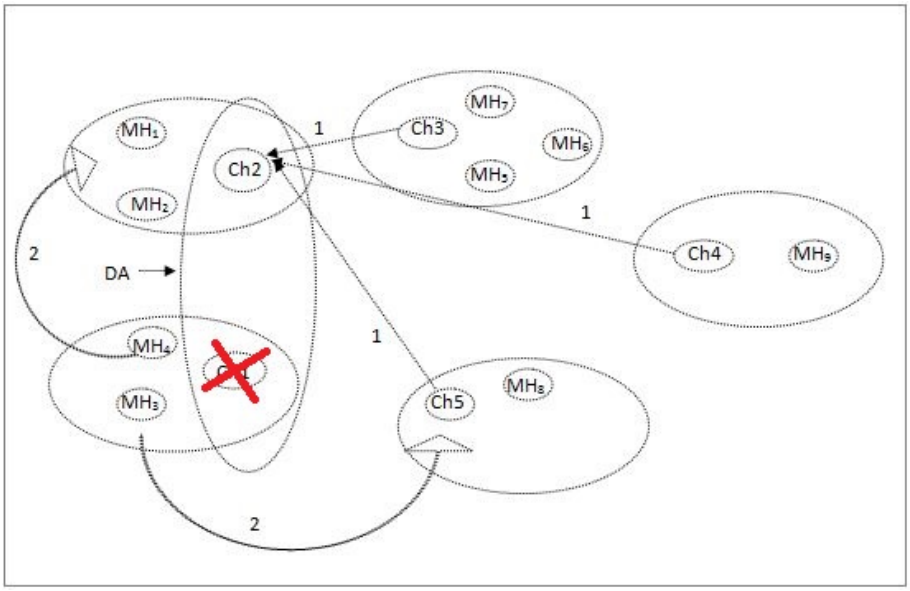

FIGURE 3.2 – Un coordinateur qui tombe en panne.

Les clusterheads passent au tour 2 et séléctionne le clusterhead Ch2 comme coordinateur.

3. Le coordinateur envoie le message de proposition PROP (r2, estcc) aux clustehreads figure 3.3.

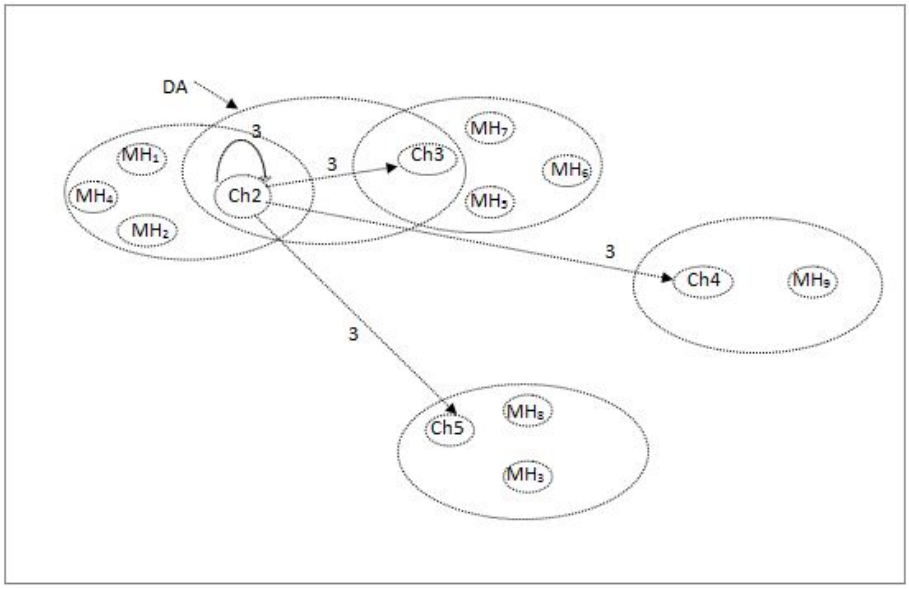

FIGURE 3.3 – L'envoie du message de proposition.

4. Les clusterheads Ch2 et Ch3 envoient les messages ECHO (voir Figure 3.4).
La décision aurait pu être prise si les clusterhead Ch4 et Ch5 ne suspectent pas Ch2 mais pour illustrer le deuxième cas de figure, on suppose que Ch4 et Ch5 suspectent Ch2.

– **Deuxième cas de figure :** Supposons maintenant que lors de l'envoie du message PROP, les clusterheads Ch4 et Ch5 ont suspecté par erreur le coordinateur Ch2. Ils vont donc passer au prochain tour et envoient des messages d'ECHO avec ri =3. D'autre part l'ensemble de DA ne peut pas prendre une décision puisque il n'as pas encore reçu tous les messages d'ECHO, alors il continue d'attendre (voi Figure 3.4).

4'. Les clusterheads Ch4 et Ch5 envoient un message ECHO(3,v)(voir Figure 3.4).

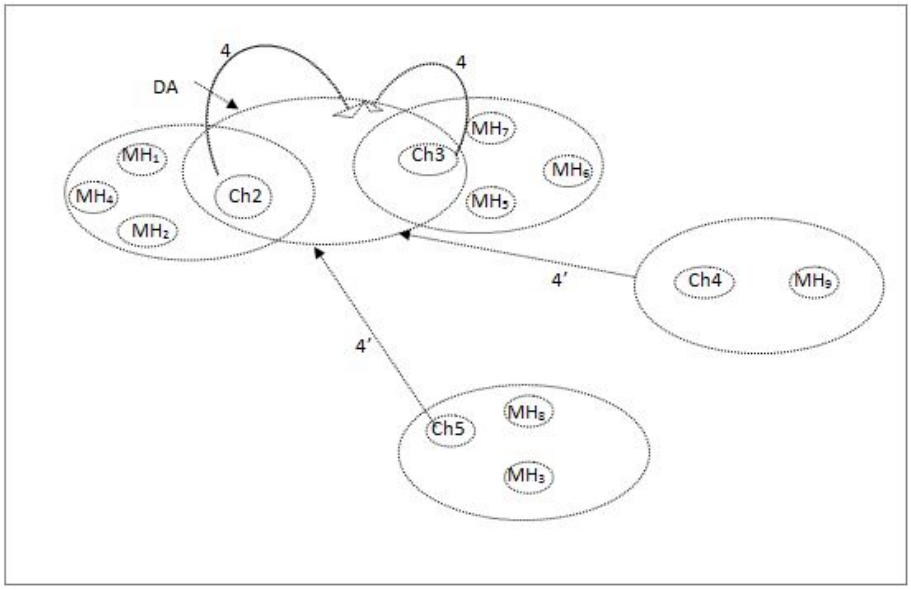

FIGURE 3.4 – Fausse suspicion du coordinateur.

L'ensemble de DA teste les messages ECHO qu'il vient de recevoir et il vérifie si les r apportés par ces messages égale à ri. Si ce n'est pas le cas, alors il passe au tour suivant.

5. Tous les clusterheads envoient un message d'ECHO contenant le tour courant r$i = 3$ (voir Figure 3.5).

pour illustrer le troisiéme cas de figure on suppose que Ch2 tombe en panne.

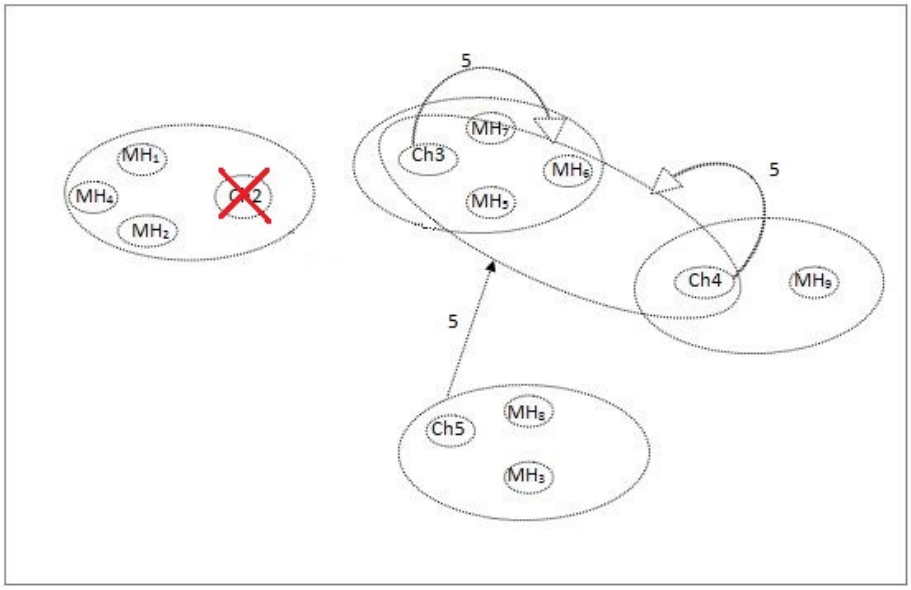

FIGURE 3.5 – Les clusterheads qui tombent en panne.

– **Troisième cas de figure :** Le cas des clusterheads dynamiques

  Dans ce cas supposons que le clusterhead Ch2 tombe en panne, en raison des défaillances de ce clusterhead et la mobilité de l'hôte MH*7*, la fonction d'eventuel clusterer détermine des nouveaux clusterheads. De ce fait les hôtes locaux MH*2* et MH*4* deviennent des clusterheads notés Ch2', Ch4' respectivement, MH*1* rejoint le cluster de Ch2' et le MH*7* rejoint Ch4'.

  Suite à l'ajout des deux clusterheads, les hôtes de l'ensemble DA ne peuvent pas décider(ils n'ont pas reçu les messages d'ECHO de tous les clusterheads)

– **Quatrième cas de figure :** La prise de décision

  5'. Les deux clusterheads ajoutés participent au consensus en rejoignant le tour des autres clusterheads(r=3 obtenu en échangeant les messages ROUND)et ils envoient un message ECHO(voir Figure 3.6)

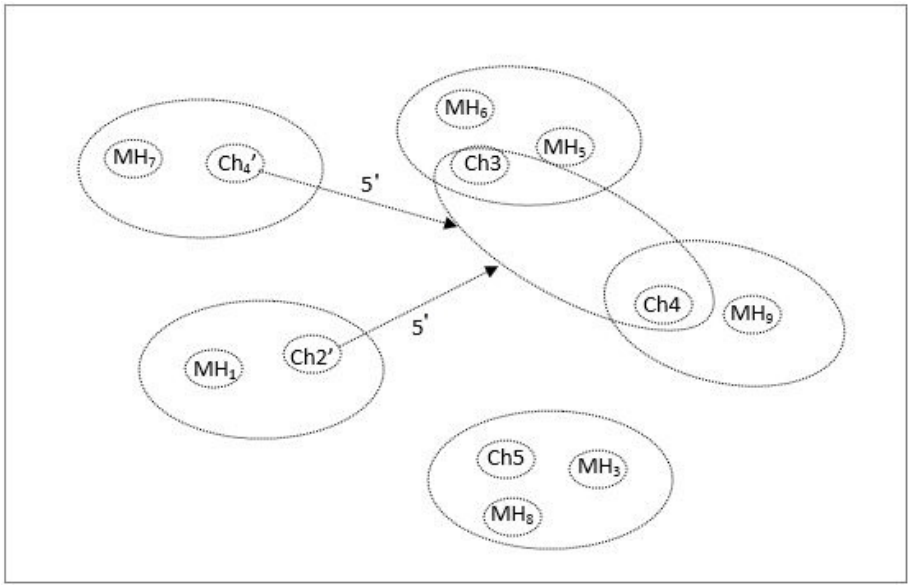

FIGURE 3.6 – Les clusterheads dynamiques.

Maintenant que tous les clusterheads ont reçu les messages ECHO de tous les clusterheads du réseau et que tous les r apportés par ce messages égale à r i(r=3), la décision sera prise.

6. L'ensemble des hôtes DA met à jour leurs estimations à la valeur apportée par les messages d'ECHO et ils envoient les messages de décision aux autres clusterheads du réseau. Figure 3.7.

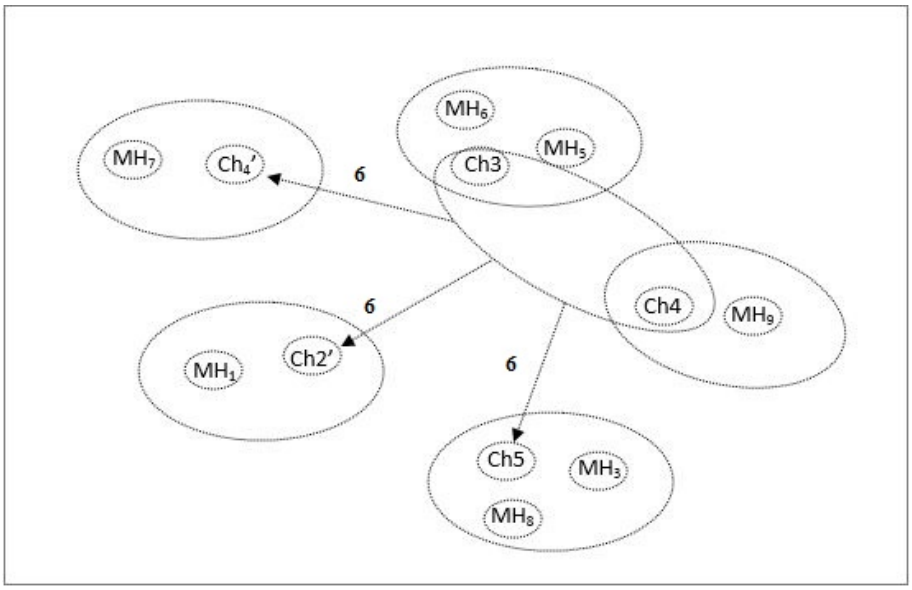

FIGURE 3.7 – La transmission de la décision.

A la réception de message de décision.

7. Les clusterheads mettent à jour leurs estimations et ils transmettent la décision à leurs hôtes locaux figure 3.8.

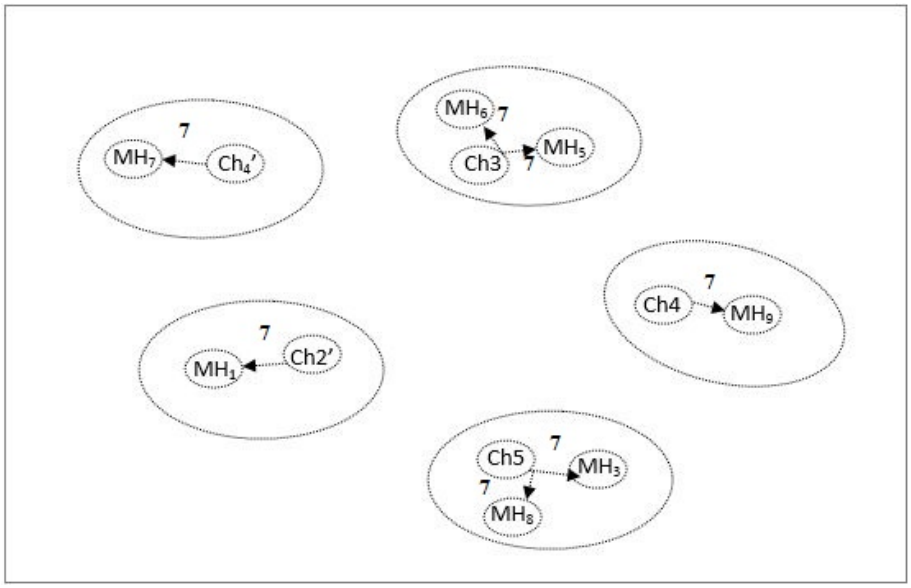

FIGURE 3.8 – La transmission de la décision.

Enfin, les hôtes reçoivent la décision et décident sur la valeur apportée par le message de décision. De cette manière tous les hôtes du réseau décident d'une même valeur et atteignent un consensus.

3.3.4 Preuves

Dans cette section, nous allons donner les éléments de preuve relative au protocole proposé. Cela revient à montrer que les propriétés de terminaison, accord et validité présentées dans le premier chapitre, sont satisfaites :

Propriété 1 (Terminaison) : Un hôte correct doit finalement décider d'une valeur.

Preuve : Un site mobile décide si le protocole de consensus se termine et la décision atteint le site mobile. Le protocole se termine si chacun de ces hôtes se termine :

- Le coordinateur :Puisque les canaux de communication sont supposés fiables et selon les propriétés du détecteurs de défaillances, le coordinateur finira par recevoir les messages d'ECHO et il se termine par prendre une décision.
- Les clusterheads : Ne se bloquent pas parce que l'ensemble DA leurs envoient la décision.
- Les hôtes locaux : Ne se bloquent pas car le clusterhead qui leurs sont associé leurs transmet la décision une fois qu'elle était prise.

Propriété 2 (validité) : Si un processus décide d'une valeur v, c'est que v a été proposée par un des processus.

Preuve : Puisque le coordinateur impose sa valeur, donc c'est la valeur du coordinateur du dernier tour qui sera prise en décision. Aucune autre valeur ne sera choisie.

Propriété 3 (Accord) : Si un processus correct décide d'une valeur v, et qu' un autre processus correct décide d'une valeur v', alors v'= v.

Preuve : Un hôte mobile ne décide que si son clusterhead reçoit la décision. Dans ce cas, l'hôte adopte la décision de son clusterhead. Ainsi, chaque clusterhead reçoit la décision de DA et puisque l'ensemble de DA décide sur une seule même valeur, donc tous les clusterheads reçoivent la même décision de la part de DA. Par conséquent, tous les hôtes ne peuvent pas décider différemment.

3.4 Conclusion

Dans ce chapitre nous avons présenté une solution plus performante pour un consensus dans les réseaux ad hoc, basée sur l'eventual clusterer $\diamondsuit$C, puis nous avons donné un exemple illustratif pour mieux expliquer le fonctionnement de notre approche, suivi des preuves des propriétés qui doivent être vérifiees pour tout protocole de consensus.

4

Simulation et étude des performances

4.1 Introduction

Dans le chapitre précédent, nous avons présenté un nouveau protocole de consensus adapté pour les réseaux mobiles ad hoc, basé sur l'approche hiérarchique HCD et sur l'eventual clusterer $\diamondsuit$C. Afin d'évaluer ses performances, nous l'avons comparé avec le protocole HCD. Nous présentons en premier lieu l'environnement de simulation utilisé avec les métriques de performances mesurées et les scénarios de simulations adoptés, puis nous donnons l'interprétation des résultats obtenus à l'issue de ces simulations.

4.2 Environnement de simulation

Pour tester les performances d'une solution apportée à un problème de communication dans un réseau, il n'est pas toujours possible d'accéder aux infrastructures nécessaires en raison de leurs coûts élevés. De plus, les expérimentations réelles n'offrent souvent pas une grande souplesse. Rappelons que les réseaux mobiles ad hoc sont des réseaux qui englobent plusieurs unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces radio. En effet, il serait très coûteux voir impossible de mettre en place un réseau à des fins de tests de certains critères. Pour remédier à ce problème et afin de tester les performances d'un nouveau protocole, on a recours à la simulation qui met à la disposition de l'utilisateur un environnement d'expérimentation. Parmi les simulateurs les plus utilisés dans la communauté des réseaux mobile ad hoc, nous citons le simulateur MATLAB.

Le choix de MATLAB

MATLAB est un logiciel de calcul numérique produit par MathWorks. Il est disponible sur plusieurs plateformes. MATLAB est un langage simple et très efficace, optimisé pour le traitement des matrices, d'où son nom. Pour le calcul numérique, MATLAB est beaucoup plus concis que les anciens langages (C, Pascal, Fortran, Basic) et pour la programmation, il optimise le code des programmes en utilisant des fonctions prédéfinies. On peut traiter la matrice comme une simple variable.

MATLAB contient une interface graphique puissante et on peut l'enrichir en ajoutant des "boites à outils" (toolbox) qui sont des ensembles de fonctions supplémentaires,profilées pour des applications particulières (traitement de signaux, analyses statistiques, optimisation, etc.). Il permet aussi la création de fonctions et distingue les données locales des données globales. Ces avantages ont rendu de MATLAB, un langage de programmation

Paramètres	**Valeur initiale**
Nombre de nœuds	50
Nombre de nœuds défaillants	10%-50%
Portée des nœuds	20 mètres
Temps de simulation	6000 *secondes*
Surface du réseau	$100 * 100\ m^2$

TABLE 4.1 – Paramètres de simulation.

et de simulation très sollicité.

4.3 Les paramètres de simulation

Le tableau suivant contient les paramètres du réseau sur lequel les simulations ont été effectuées.

– **L'échéancier**

Les évènements détectés dans le réseau sont rangés dans deux échéanciers

Echéancier 1 pour calculer et prendre des mesures en fonction de nombre de défaillances (voir le tableau 4.2).

Numéro d'évènement	1	2	3	4
Nombre de nœud dans le réseau	50	50	50	50
Nombre de défaillances	2	4	11	24

TABLE 4.2 – Table d'échéancier1.

Echéancier 2 pour calculer et prendre des mesures en fonction de nombre de nœuds dans le réseau (voir le tableau 4.3).

Numéro d'évènement	1	2	3	4
Nombre de nœud dans le réseau	16	25	35	50
Nombre de défaillances (le nombre maximum de défaillance toléré)	7	11	17	25

TABLE 4.3 – Table d'échéancier2.

4.4 Les étapes de simulation

Les étapes décrivants la réalisation de nos simulations sont illustrées par l'organigrame ci-dessous :

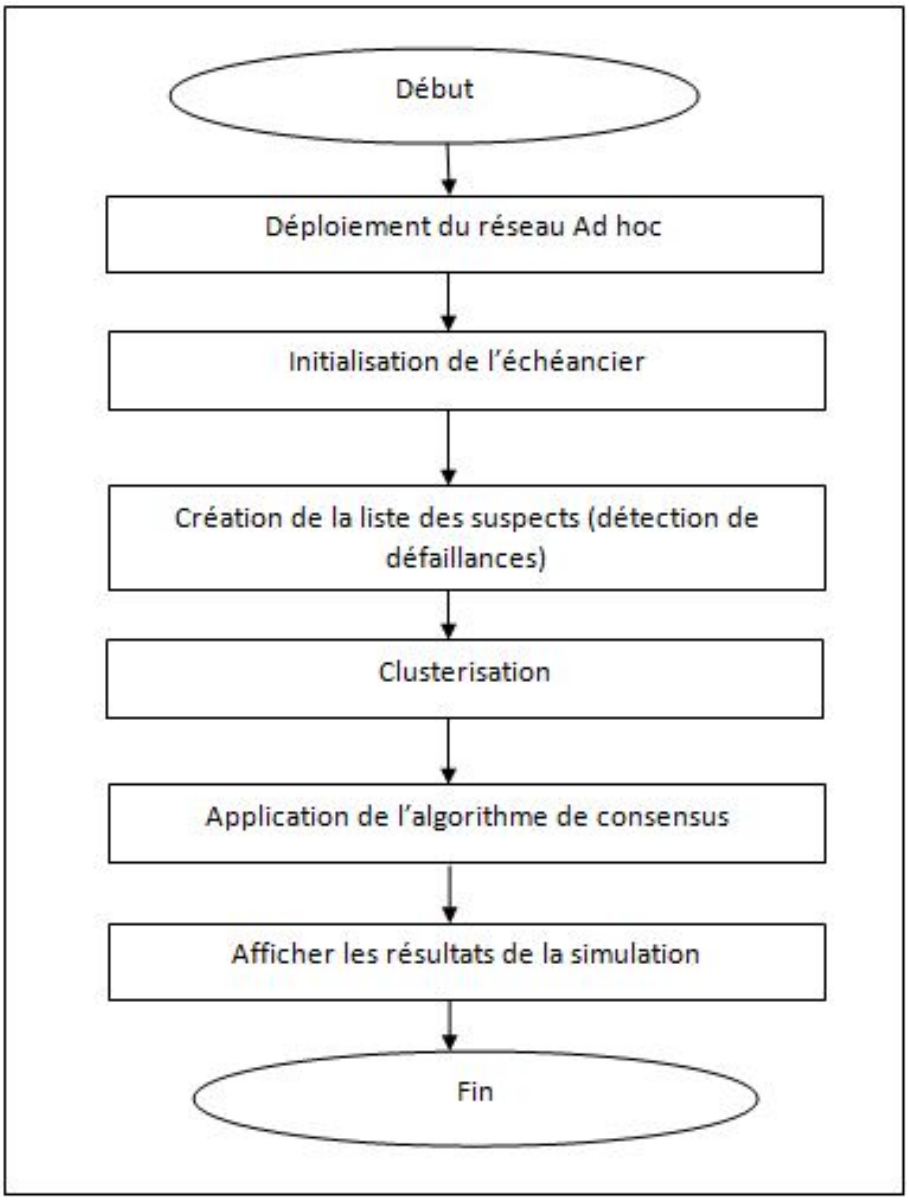

FIGURE 4.1 – Les principales fonctions du simulateur.

4.4.1 Initialisation des variabes de simulation

Cette phase est exécutée automatiquement au début du programme de simulation. Elle inclut la déclaration des variables globales (nombre de nœud, zone de déploiement simulée, temps max de simulation, etc.) et leur initialisations, ainsi que la création des nœuds sous forme d'une structure qui comporte (l'identité du nœud, leur coordonnées).

Structure d'un nœud
Identité du nœud : id
Coordonnées du nœud (x,y) : cord
Nombre de voisins : nb
Valeur estimée : est
Booléen indiquant si la décision est prise ou non : fl
Id de clusterhead : ch
Liste des nœuds suspectés : susp
Tour courant : r
Nombre de messages reçus : NbreMR.
Nombre de messages envoyés : NbreME

TABLE 4.4 – Structure d'un nœud.

4.4.2 Déploiement du réseau

Les nœud constituants notre réseaux sont déployés d'une manière aléatoire sur une surface de (100*100) m^2. Chaque nœud dans le réseau est représenté par ses coordonnées (x,y).

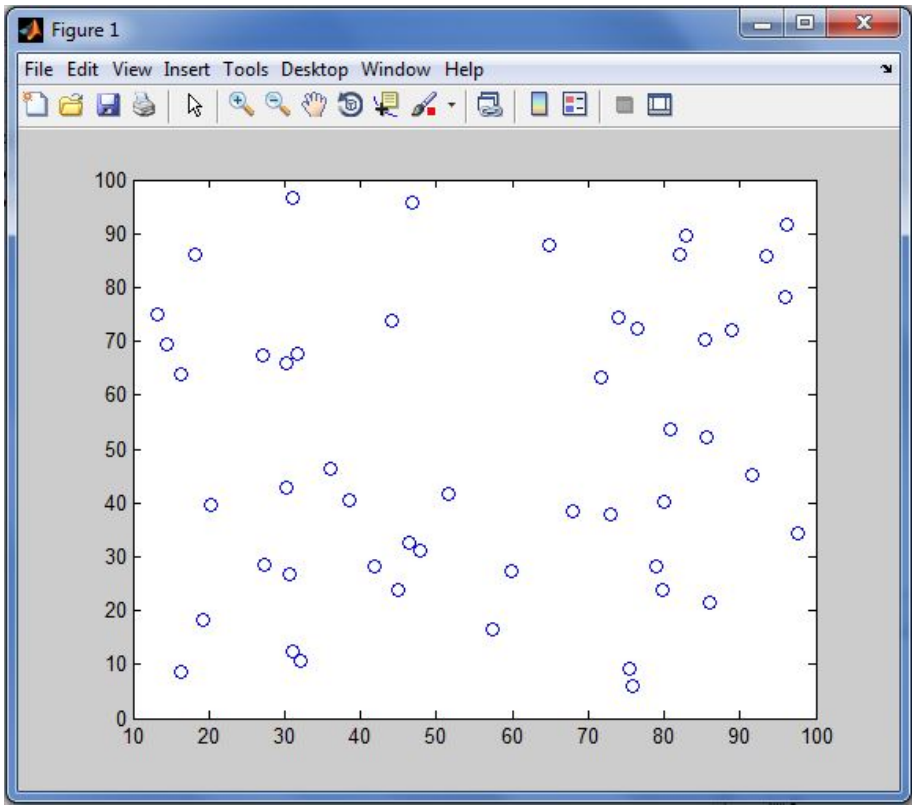

FIGURE 4.2 – Déploiement du réseau.

4.4.3 Détection des nœuds défaillants

L'introduction des nœuds défaillants dans le réseau se fait de manière aléatoire et chaque nœud dispose d'une liste de nœuds suspectés. La construction de la liste respecte les propriétés de détecteur de défaillances ♢ S sur lesquels ♢ C se base pour retourner les trois ensembles ♢C.trusted, ♢C.clusterhead et ♢C.CH.

Nous avons respecté la propriété de complétude forte en introduisant pour chaque nœud les nœuds défaillants dans sa liste des suspects, de cette façon tout nœud défaillant est détecté par tout autres nœuds corrects.

Au départ, chaque nœud suspecte tous les nœuds du réseau et en cours de l'exécution du protocole de consensus, les nœuds corrects seront retirés de la liste des suspects au fur et à mesure, ainsi la propriété de précision ultime faible sera respectée (il existe un instant à partir duquel il existe un nœud correct qui n'est plus jamais suspecté par aucun nœud correct).

4.4.4 Application de l'algorithme de clusterisation

Dans notre approche c'est ♢C qui se charge de la clusterisation qui consiste à grouper les hôtes du réseau en cluster et sur chaque cluster un clusterhead non défaillant est choisi. Dans notre simulation le clusterhead élu représente un nœud correct choisi aléatoirement parmi ses voisins à un saut.

4.4.5 Consensus

Le consensus est exécuté en des tours, dans chaque tour un coordinateur tente d'imposer son estimation. La décision final est prise si le coordinateur reçoit la majorité des messages avec le tour courant ce qui veut dire que la décision sera prise si la majorité des clusterheads (dans notre approche et la majorité des nœuds dans HCD) ne suspectent pas le coordinateur courant.

4.4.6 Les métriques d'évaluation de performances

Nous avons également adopté plusieurs mesures pour évaluer notre solution. Les métriques d'évaluation de performances utilisées dans les simulations sont définies comme suit :

NT (Nombre de Tours) : Le nombre de tours exécutés par les hôtes pour parvenir à un consensus.

NM (nombre de Messages) : Le nombre de messages envoyés par les hôtes pour obtenir la décision globale.

4.5 Résultats et interprétations

Pour étudier l'efficacité de notre solution, nous allons la comparer avec l'approche HCD. En se basant sur les paramètres d'évaluation cités précédemment, nous analysons les résultats de simulation obtenus dans les figures ci-dessous comme suit :

1.Le nombre de tours exécutés

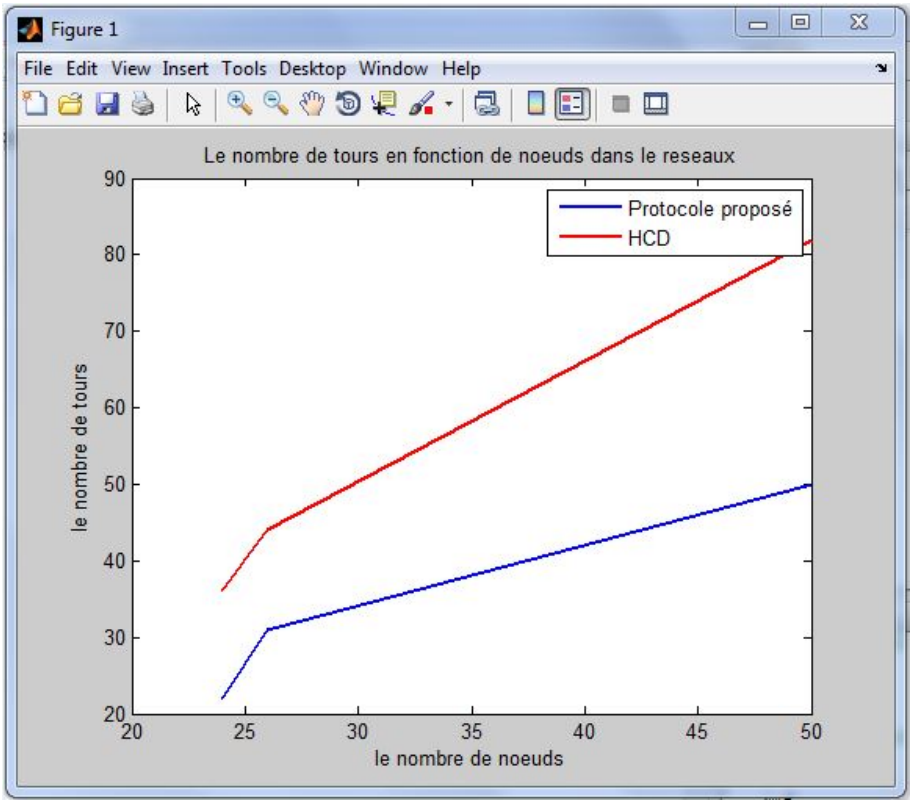

FIGURE 4.3 – Le nombre de tours exécutés en fonction des nœuds.

Cette figure représente deux graphes comparatifs qui montrent l'évolution du nombre de tours exécutés en fonction du nombre de nœuds du réseau. Dans les deux protocoles (HCD et le protocole proposé), le nombre de tours augmente en fonction du nombre de nœuds.

Le nombre de tours exécutés avec le protocole HCD a été réduit avec notre solution, ceci revient à ce que le coordinateur attend les messages de la part des clusterheads et seule la défaillance du coordinateur et les fausses suspicions des clusterheads retardent la prise de décision.

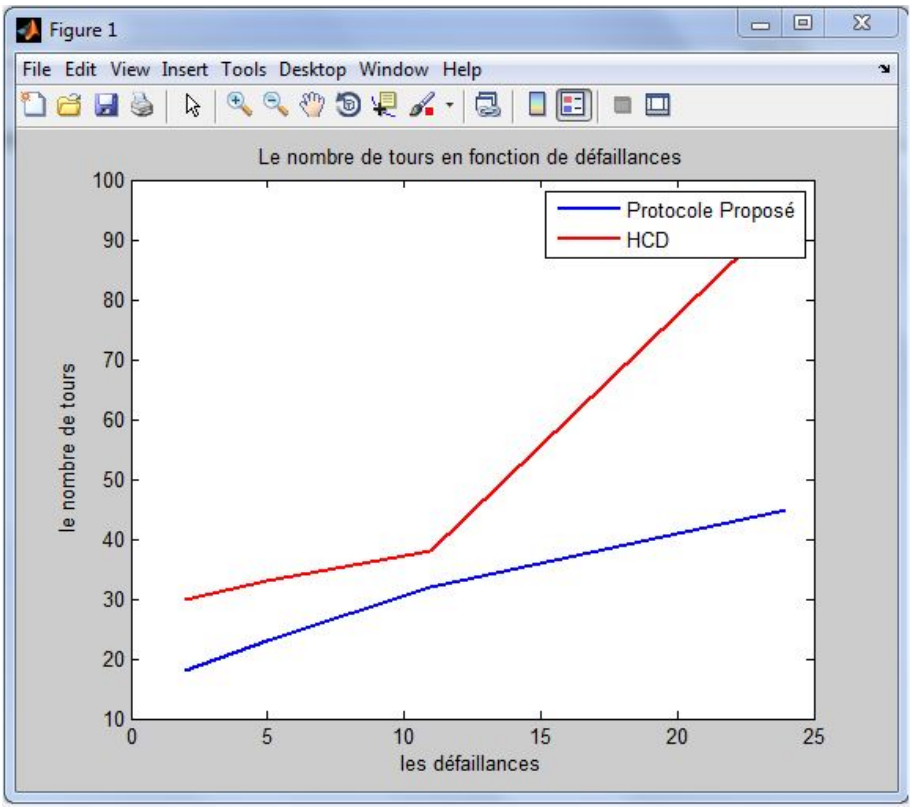

FIGURE 4.4 – Le nombre de tours exécutés en fonction de défaillances.

La figure ci-dessus représente le nombre de tours exécutés en fonction de défaillances. On remarque aussi que notre solution a permis de réduire de façon considérable la croissance du nombre de tours par rapport aux nombres de défaillances.

2.Le nombre de messages envoyés

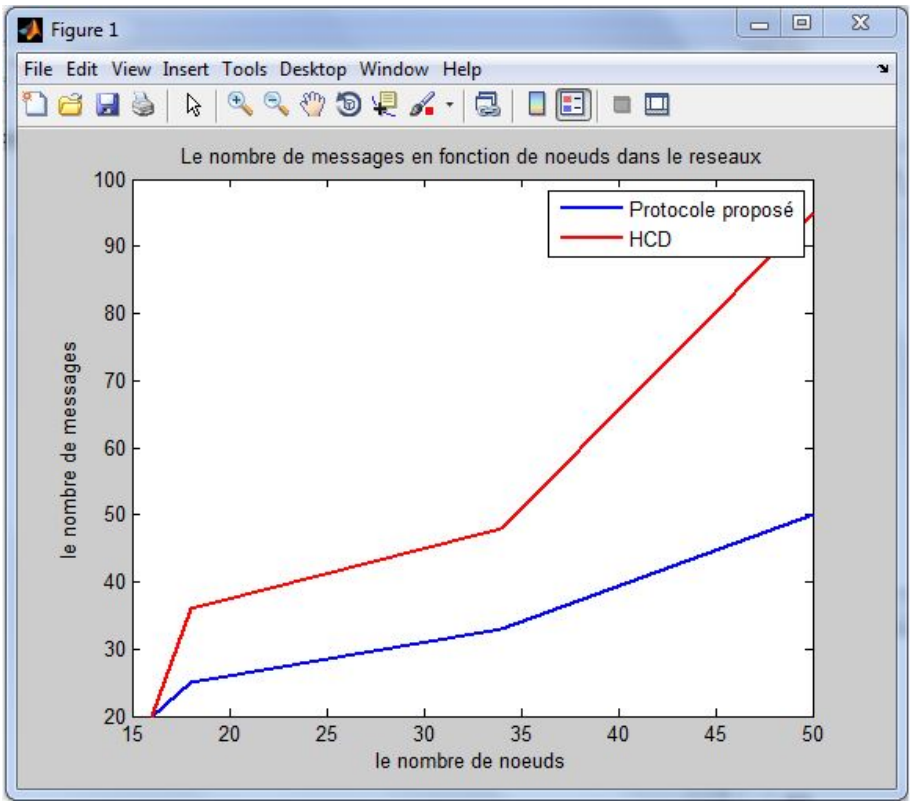

FIGURE 4.5 – Le nombre de messages envoyés en fonction de nœuds.

Cette figure illustre le nombre de messages envoyés en fonction de nœuds. D'après la figure, nous constatons la croissance de nombre de messages par rapport à la croissance de nombre de nœuds dans le réseau et que la solution proposée réduit ce nombre. Ceci est interprété par le fait que la croissance du nombre de tours induit à la croissance de nombre de messages.

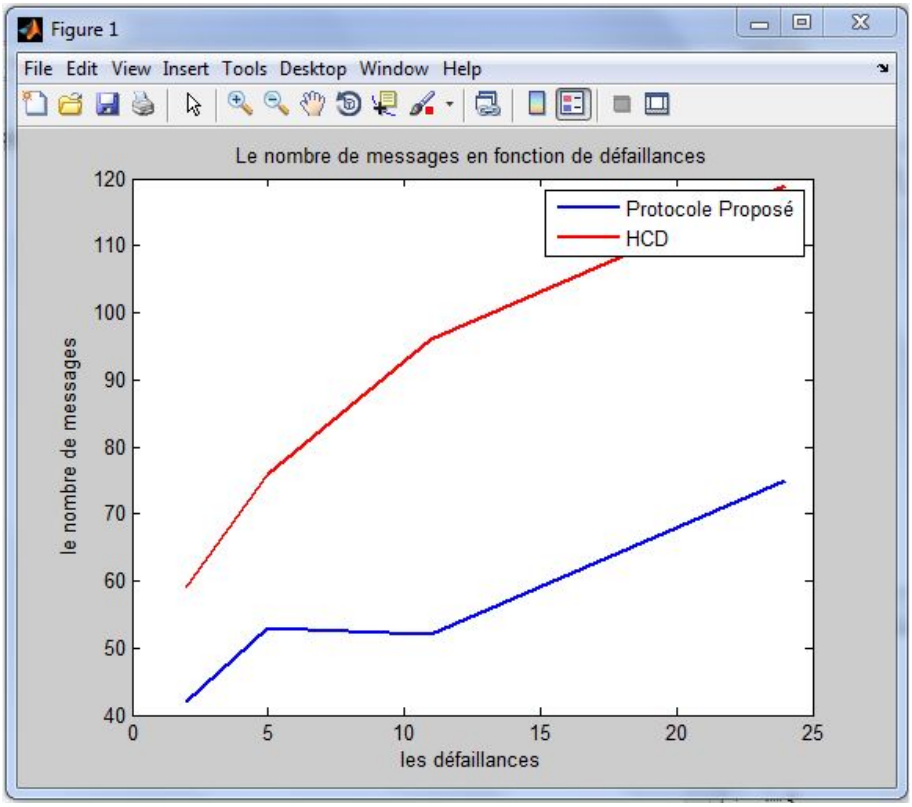

FIGURE 4.6 – Le nombre de messages envoyés en fonction de défaillances.

Pareillement, le nombre de messages envoyés est en progression par rapport à la croissance de nombre de défaillances dans le réseau.

4.6 Conclusion

Dans cette partie nous avons évalué les performances de notre approche en la comparant à l'approche HCD. A travers les figures obtenues, nous remarquons bien l'efficacité d'engager la prise en charge des hôtes locaux aux clusterheads.

La solution proposée semble offrir une possibilité réelle d'améliorer le protocole HCD et de mettre en œuvre un protocole de consensus rapide et moins coûteux.

Conclusion générale et perspectives

Ce mémoire traite le problème du consensus dans les réseaux mobiles ad hoc, il a comme objectif de proposer une nouvelle approche améliorant une approche existante (le protocole HCD). En premier lieu, nous avons présenté la notion de consensus et le problème de consensus dans les systèmes distribués en général puis nous avons défini les réseaux mobiles qui sont une branche de systèmes distribués. En deuxième lieu, nous avons étudié le problème de consensus dans ce type de réseaux, nous avons cité son utilité et ses difficultés. Par la suite et à travers les recherches que nous avons effectuées nous avons énuméré les principales approches existantes dans la littérature et les travaux qui ont été amenés et qui sont destinés à la réalisation de consensus de manière efficace. A base de cette étude, nous avons proposé une amélioration pour le protocole de consensus HCD qui répond aux contraintes en énergie des sites mobiles et à la faible bande passante des liaisons sans fil, en réduisant le nombre de tours et le nombre de messages échangés. Ce qui permet la réalisation rapide de consensus. Nos perspectives de recherche sont :

- Il serait intéressant d'étendre la simulation des deux protocoles et d'implémenter $\diamondsuit$ C afin d'évaluer les suspicions réelles et de calculer le temps d'exécution des deux protocoles.
- Il parait important de comparer notre approche avec d'autres approches de consensus en considérant d'autres critères d'évaluation et de prendre en considération d'autres paramètres comme le temps d'attente par exemple.
- Dans notre recherche on s'est limité à la notion de l'efficacité du temps des protocoles de consensus, de manière à faire face aux contraintes de ressources des environnements mobiles. Mais la conception de protocoles de consensus pour les systèmes mobiles dynamiques semble intéressant à explorer, où le nombre d'ordinateurs participant peut changer arbitrairement et un nombre illimité d'ordinateurs peut rejoindre ou quitter le système à tout moment.

Bibliographie

[1] A. Mostefaoui, M. Raynal, Leader-based Consensus, Parallel Processing Letters, vol. 11 no. 1, pp. 95-107, 2001.

[2] A. Mostefaoui, S. Rajsbaum, M. Raynal, C. Travers, Irreducibility and Additivity of Set Agreement-Oriented Failure Fetector Classes, Journal of the ACM, USA, pp. 162-153 , 2006.

[3] C. Delporte-Gallet et al., The weakest failure detectors to solve certain fundamental problems in distributed computing, in PODC '04 : Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, pp. 338-346, NewYork, NY, USA, 2004.

[4] C.Dwork, N.Lynch, L.Stockmeyer, Consensus in the Presence of Partial Synchrony, Journal of the ACM, 35(2), pp. 288-323, 1988.

[5] E.Vollset, Design and Evaluation of Crash Tolerant Protocols for Mobile Ad hoc Network, Thése de doctorat, Université de Newcastle upon tyne, Septembre 2005.

[6] F.Cristian, C.Fetzer, The Timed Asynchronous Distributed System Model, IEEE Trans. Parallel Distrib. Syst., 10(6), pp. 642-657, 1999.

[7] F.Greve, réponses efficaces au besoin d'accord dans un groupe, Thèse de doctorat , Université de renne 1, septembre 2002.

[8] G. Neiger, Failure detectors and the wait-free hierarchy, in PODC '95 : Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing, pp. 100-109, New York, NY, USA, 1995.

[9] H. Seba, N. Badache, A. Bouabdallah, Solving the Consensus Problem in a Dynamic Group : An Approach Suitable for a Mobile Environment, Proc. Int'l Symp. Computers and Comm. (ISCC), 2002.

[10] H.Seba, Sécurité et Tolérance aux Fautes dans les Environnements Mobiles, Thèse de doctorat, Université de Compiègne, Décembre 2003.

[11] I.Moise, Efficient Agreement Protocols in Asynchrones Distributed System, Thèse de doctorat, Université de Rennes 1, décembre 2011.

[12] J.Supina, Algorithme d'exclusion mutuelles : tolérances aux fautes et adaptation aux grilles, Thèse de doctorat, université pierre et marie curie- Paris 6, 2008.

[13] M. Ben-Or, Another Advantage of Free Choice : Completely Asynchronous Agreement Protocols, Proc. of the 2nd ACM Symposium on Principles of Distributed Computing PODC'83), pp. 27-30, 1983.

[14] M. Fischer, A. Lynch, M. S. Paterson, Impossibility of distributed consensus with one fault process, Journal of the ACM, 1985.

[15] M. Hurfin, A. Mostefaoui, M. Raynal, A Versatile Family of Consensus Protocols Based on Chandra-Toueg's Unreliable Failure Detectors, IEEE Trans. Computers, vol. 51, no. 4, pp. 395-408, April 2002.

[16] M. Larrea, A. Fernandez, S. Arevalo, On the Implementation of Unreliable Failure Detectors in Partially Synchronous Systems, IEEE Trans. on Computers, vol. 53, no. 7, pp. 815-828, 2004.

[17] N. Badache, M. Hurfin, R. Macedo, "Solving the Consensus Problem in a Mobile Environment." In Proc. of the 1999 IEEE Intl. Performance, Computing and Communication Conference (IPCCC'99), Phoenix, USA, pp.29-35, Février, 1999.

[18] O.Fouail, Découverte et fourniture de services adaptatifs dans le deuxième chapitre dans les environnements mobiles , Thèse doctorat, l'Ecole nationale supérieure des télécommunications, paris, 2004.

[19] P. Dutta and R. Guerraoui, Fast Indulgent Consensus with Zero Degradation, Proc. of the 4th European Dependable Computing Conference (EDCC'02), France, LNCS

2485, pp. 191-208, 2002.

[20] P. Ezhilchelvan, A. Mostefaoui, M. Raynal, Randomized Multivalued Consensus, Proc. of the 4th IEEE Int'l Symp. on Object-Oriented Real-Time Computing, pp. 195- 200, 2001.

[21] R.Guerraoui , M.Hurfin, A.Mostefaoui, R.Oliveira, M.Raynal, A.Schiper Consensus in Asynchronous Distributed Systems : A Concise Guided Tour, EPFL, Département d'Informatique, 1015 Lausanne, Suisse 2 IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France.

[22] T.Chandra, V. Hadzilacos, S.Toueg, The weakest failure detector for solving problems, Journal of the ACM, Vol. 43, No. 4, pp. 685-722, July 1996.

[23] T.Chandra, V. Hadzilacos, S.Toueg, Unreliable failure detectors for reliable distributed systems, Journal of the ACM, 43(2), pp. 225-267, March 1996.

[24] W.Weigang, Distributeur Coordination in Mobile Networks Environnements, Thèse de doctorat, Université de hong kong, Novembre 2006.

[25] W.Wu and J.Cao, M.Raynal, Eventual Clusterer : A Modular Approach to Designing Hierarchical Consensus Protocols in MANETs, IEEE Transactions On Parallel And Distributed , VOL. 20, No. 6, pp. 753-765, June 2009.

[26] W.Wu and J.Cao, J.Yang, M.Raynal ,Design and Performance Evaluation of Efficient Consensus Protocols for Mobile Ad Hoc Networks, IEEE Transactions On Computers, vol. 56, No. 8, pp. 1055-1070, August 2007.