RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE 20 Aout 1955-Skikda

Faculté des sciences

Département d'Informatique

THESE

En vue de l'obtention du Diplôme de :

Doctorat de 3° cycle (LMD) en Informatique

Option

Computation et Cognition des Systèmes Informatiques

Présentée par

SLIMANI Amira

Thème

Approches Formelles et Opérationnelles pour la Modélisation des Echanges et de la Coopération dans les Systèmes Distribués

Le 19/12/2018

Devant le jury composé de :

Boucheham Bachir	Professeur université 20 Aout 1955 Skikda	Président
Boudour Rachid	Professeur université Badji Mokhtar Annaba	Examinateur
Mazouzi smaine	MCA université 20 Aout 1955 Skikda	Examinateur
Redjimi Mohammed	Professeur université 20 Aout 1955 Skikda	Directeur de thèse

2018/2019

Remerciements

Je tiens tout d'abord à remercier ALLAH le tout puissant pour nous avoir donné la force de réaliser ce modeste travail, et aussi mes parents pour leurs soutiens et encouragements durant mes années d'études. Mes vifs remerciements s'étendent également à mon encadreur Pr.Redjimi Mohammed, professeur à l'université 20 Aout 1955 de Skikda, pour l'encouragement et ses valeureux conseils et orientations, la confiance et la patience, qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port. Permettez-moi de vous exprimer mon admiration pour vos qualités humaines et professionnelles. Je remercie également les membres du jury qui ont accepté de participer à la discussion de mon travail.

J'exprime toute ma gratitude à mon père le professeur Slimani Djamel pour son énorme aide scientifique durant toute ma formation.

J'exprime aussi toute ma gratitude aux enseignants du département d'informatique.

Je remercie toutes les personnes qui m'ont aidée durant la préparation de ce mémoire de prés ou de loin.

Dédicace

Grace à Dieu voilà notre travail terminé et il est temps pour moi de partager ma joie avec tous ceux qui mon soutenu et encourager.

Je dédié ce modeste travail :

A vous, ma mère et mon père qui avez consacré votre vie à notre éducation et a faire notre bonheur ;

Cher père je me rappellerai toujours tous les moments où tu m'as poussée à travailler et à réussir, Cher père j'avoue que si je suis devenue quelque chose actuellement c'est grâce à tes efforts, à tes conseilles et à ta surveillance. Merci et j'espère que vous trouverez dans ce travail l'expression de ma gratitude et mon respect.

Ma très chère mère : j'aimerai toujours te remercier pour tout ce que tu as fait jusqu'à aujourd'hui pour assurer mon éducation et ma formation. Je me rappelle vraiment de tous tes efforts avec moi dès mon enfance, chère mère j'avoue vraiment que tu été pour moi la lumière qui me guide mes routes et qui m'emmène aux chemins de la réussite, c'est grâce à toi que je dois toute ma réussite. J'espère que mon travail sera le témoignage de ma gratitude et mon respect le plus profond.

A la mémoire de ma chère grande mère ;

A mes grands-pères, et ma grand-mère;

A mon cher frère Oussama et A ma chère sœur Isra;

A mon adorable petite sœur Safa;

A toute ma famille ;

A mon mari Turki Mohammed Fares;

A tous ceux que j'aime et qui m'aiment.

Résumé

Au cours de ces dernières décennies, les développements spectaculaires dans le domaine de l'informatique, de la micro-électronique et des média de communication ont permis des évolutions considérables dans les systèmes répartis. Ces systèmes distribués ont reçu une attention considérable de la part des universitaires et de l'industrie et sont devenus un domaine de recherche très important. La motivation principale des systèmes distribués est le partage des ressources et l'accès distant aux informations et aux services indépendamment de leurs localisations. Plusieurs architecture des systèmes distribués se sont développées afin d'améliorer leurs performances et de répondre aux besoins des utilisateurs. Les réseaux sans fil sans ont vu naître, au cours de leur évolution, diverses architectures dérivées, telles que les réseaux cellulaires, les réseaux locaux sans fil, les réseaux WiMax, etc. Durant la dernière décennie, de nouveaux types de réseaux sans fil ont suscité un grand intérêt auprès de la communauté scientifique et des industriels, il s'agit des réseaux ad hoc et des réseaux de capteurs sans fil (WSN). Ces dernières années, les réseaux de capteurs sans fil se sont introduits à tous les niveaux de la société : les entreprises, les gares, les hôpitaux, les usines, les domiciles, les téléphones et autres équipements domestiques, militaire, de santé, environnement.

Dans cette thèse, nous nous sommes intéressés au problème de l'énergie dans les réseaux de capteurs sans fil. Nous avons abordé cette problématique en investiguant deux axes de recherche, à savoir les protocoles de routage hiérarchique et les stratégies de gestion de l'énergie. Notre première contribution consiste à proposer un protocole de clustering nommé WDC-LEACH-C pour établir une architecture hiérarchique clustérisée avec une meilleure sélection des clusters heads (CHs) afin de minimiser la consommation d'énergie dans les clusters.

La seconde contribution consiste dans la proposition d'un autre protocole de routage nommé Improved-ACORP qui intègre une stratégie de gestion d'énergie des CHs nommée CHMS afin de conserver le maximum l'énergie aux CHs et prolonger, ainsi, la durée de vie du réseau. Plusieurs simulations ont été effectuées pour démontrer la faisabilité et l'efficacité de ces solutions.

Mots clés : Systèmes distribués Réseaux de capteurs sans fil (WSN), Clustering, gestion d'énergie, cluster-Head (CH).

Abstract

Over the last decades, spectacular developments in the field of computing, microelectronics and communication media have led to significant changes in distributed systems.

These systems have received considerable attention from academia and industry and have become a very important area of research. The primary motivation for distributed systems is resource sharing and remote access to information and services regardless of location.

Several distributed systems architecture have been developed to improve their performance and meet the needs of users. Wireless networks have seen the emergence of various derived architectures, such as cellular networks, wireless local area networks, WiMax networks, and so on. During the last decade, new types of wireless networks have attracted a great deal of interest from the scientific and industrial communities, namely ad hoc networks and wireless sensor networks (WSN).

In this thesis, we are interested in the problem of energy in wireless sensor networks. We approached this issue by investigating two lines of research, namely hierarchical routing protocols and energy management strategies. Our first contribution is to propose a clustering protocol named WDC-LEACH-C to establish a hierarchical architecture. Clustered with better selection of cluster heads (CHs) to minimize power consumption in clusters.

In recent years, wireless sensor networks have been introduced at all levels of society: businesses, railway stations, hospitals, factories, homes, telephones and other household, military, health and environmental equipment.

In this thesis, we are interested in the problem of energy in wireless sensor networks. We approached this issue by investigating two lines of research, namely hierarchical routing protocols and energy management strategies. Our first contribution is to propose a clustering protocol named WDC-LEACH-C to establish a hierarchical architecture clustered with better CH selection to minimize energy consumption in clusters. While the second contribution is to propose another routing protocol named Improved-ACORP that integrates a CHs energy management strategy called CHMS in order to keep the CH energy to the maximum and thus extend the network lifetime. Several simulations have been performed to demonstrate the feasibility and the effectiveness of these solutions.

Keywords: Distributed Systems Wireless Sensor Networks (WSN), Clustering, Power Management, Cluster Head (CH).

ملخص

في العقود الأخيرة، مع انفجار الإنترنت، فإن التعقيد المتزايد لمشاكل التحسين في العالم الحقيقي يثير تحديات جديدة للحوسبة القابلة للتوسعة. واستجابة لهذه التحديات، حظيت النظم الموزعة باهتمام كبير من طرف الباحثين الأكاديميين والصناعيين وأصبحت مجالًا مهمًا للغاية في ميادين البحث والتطوير العلمي والتكنولوجي. الدافع الرئيسي للأنظمة الموزعة هو تطوير ومشاركة الموارد والوصول عن بعد إلى المعلومات والخدمات من قبل المستخدمين بغض النظر عن موقعهم والطرق المنتهجة لنتناولها.

تم تطوير العديد من بنيات الأنظمة الموزعة لتحسين هذا الأداء وتلبية احتياجات المستخدمين. من بينها، ظهر النموذج اللاسلكي حسب بنيات و هندسيات متنوعة مشتقة، مثل الشبكات الخلوية، وشبكات المناطق المحلية اللاسلكية، وشبكات WiMAX، إلخ.

خلال العقد الماضي، اجتذب نوع جديد من الشبكات اللاسلكية اهتماما كبيرا من المجتمع العلمي، وهي شبكات مخصصة وشبكات الاستشعار اللاسلكية WSN. في هذه السنوات الأخيرة، تم إدخال شبكات الاستشعار اللاسلكية على جميع مستويات المجتمع: الشركات ومحطات السكك الحديدية والمستشفيات والمصانع والمنازل والهواتف وغيرها من المعدات المنزلية والعسكرية والصحية والبيئية.

في هذه الأطروحة، نحن مهتمون بمشكلة الحفاظ على الطاقة في شبكات الاستشعار اللاسلكية. لقد تعاملنا مع هذه المسألة من خلال التحقيق في سطرين من الأبحاث، وهما بروتوكولات التوجيه الهرمي واستراتيجيات إدارة الطاقة، وتتمثل مساهمتنا الأولى في اقتراح بروتوكول تجميع يطلق عليه WDC-LEACH-C لإنشاء هيكل هرمي. مصفوف مع أفضل اختيار CH للحد من استهلاك الطاقة في المجموعات المنشأ.

بينما تتمثل مساهمتنا الثانية في اقتراح بروتوكول توجيه آخر باسم Improved-ACORP يدمج استراتيجية إدارة الطاقة CHsتسمى CHMS من أجل الحفاظ على طاقة CH إلى أقصى حد وبالتالي تمديد عمر الشبكة. كما أنه تم القيام بالعديد من عمليات المحاكات لإثبات الجدوى والفعالية للحلول المقدمة.

الكلمات المفتاحية: الأنظمة الموزعة ,أنظمة الاستشعار اللاسلكية (WSN) ، إدارة الطاقة ، رئيس المجموعة.(CH)

Table des matières

Introduction générale 1

Chapitre 1 : Introduction aux systèmes distribués 4

1.1 Introduction 4

1.2 Définitions 5

1.3 Aspects matériels des systèmes distribués 5

1.4 Aspects logiciels des systèmes distribués 6

1.5 Caractéristiques des systèmes distribués 7

1.6 Intérêts des systèmes distribués 8

1.7 Les défis et les propriétés requises des systèmes distribués 9

1.8 La communication dans les systèmes distribués 13

1.9 les modèles d'interaction 25

1.10 Conclusion 28

Chapitre 2 : La gestion du temps et la coopération dans les systèmes distribués 30

2.1 Introduction 30

2.2 La notion du temps dans les systèmes distribués 30

2.3 Horloge physique 31

2.4 Horloge logique 35

2.5 Accord et coordination 41

2.6 Exclusion mutuelle 42

2.7 Election d'un leader 50

2.8 Consensus 54

2.9 Les transactions distribuées 56

2.10 La terminaison d'une application distribuée 63
2.11 Les détecteurs de défaillances dans les systèmes distribués 64
2.12 Conclusion 66
Chapitre 3 : Les réseaux de capteurs sans fil 67
3.1 Introduction. 67
3.2 Architectures des réseaux de capteurs sans fil 67
3.3 Facteurs et contraintes liés aux réseaux de capteurs 69
3.4 Architecture matérielle d'un capteur 71
3.5 Consommation d'énergie dans les RCSF 73
3.6 Le routage dans les réseaux de capteurs 74
3.7 Le clustering 77
3.8 Les algorithmes de clustering 83
3.9 La taxonomie proposée pour les algorithmes de clustering 86
3.10 Conclusion 96
Chapitre 4 : Contributions 97
4.1 Introduction 97
4.2 Motivation 98
4.3 Spécifications générales 99
4.4 Protocole de clustering proposé WDC-LEACH-C 101
4.5 La modélisation de notre proposition WDC-LEACH-C 104
4.6 Simulation et évaluation des performances 113
4.7 Le deuxième protocole de routage hiérarchique proposé Improved-ACORP 118
4.8 Simulation et évaluation des performances 139
4.9 Conclusion 142
Conclusion générale et perspectives 143
Bibliographie 145

Liste des figures

Figure 1-1 Représentation d'architectures matérielles des systèmes distribués 6
Figure 1-2 Architecture logicielle d'un système distribué. 7
Figure 1-3 Acheminement simple du socket TCP..... 14
Figure 1-4 Les étapes d'un appel de procédure à distance par message..... 18
Figure 1-5 Les étapes d'un appel de méthode à distance..... 19
Figure 1-6 Le bus CORBA..... 22
Figure 1-7 Architecture d'un système distribué structuré avec un intergiciel..... 23
Figure 1-8 Mémoire repartie partagée Vs. Mémoire distribués 24
Figure 1-9 La migration d'un agent mobile vers un site distant 25
Figure 1-10 L'architecture Client/Serveur..... 26
Figure 1-11 L'architecture du cluster 27
Figure 1-12 L'architecture de grille..... 27
Figure 1-13 L'architecture P2P..... 28
Figure 1-14 L'architecture du cloud 28
Figure 2-1 La mise à jour périodique par l'algorithme de Berkeley..... 33
Figure 2-2 Envoi du temps serveur en tenant compte la valeur de décalage 35
Figure 2-3 Chronogramme..... 36
Figure 2-4 Chronogramme illustrant les horloges de Lamport 38
Figure 2-5 Horloge de Mattern..... 39
Figure 2-6 Chronogramme présente une coupure 41
Figure 2-7 Diagramme d'états de l'accès en exclusion mutuelle..... 42
Figure 2-8 Election sur un anneau virtuel unidirectionnel..... 52
Figure 2-9 Algorithme Bully d'élection 54
Figure 2-10 Diagramme de Validation à deux phases 58
Figure 2-11 Diagramme de Validation à trois phases 59
Figure 2-12 Architecture du protocole de diffusion..... 60
Figure 2-13 groupes de diffusion fermé et ouvert..... 60
Figure 2-14 La relation entre les différents diffusions..... 63
Figure 3-1 Fonctionnement des réseaux de capteur. 68
Figure 3-2 Architecture matérielle d'un capteur [46]. 73
Figure 3-3 Les techniques de conservation d'énergie..... 74
Figure 3-4 Classification des protocoles de routage pour les RCSFs 75
Figure 3-5 Les opérations de LEACH représenté dans le temps..... 77
Figure 3-6 Les caractéristiques de clustering..... 81
Figure 3-7 La taxonomie proposée pour les algorithmes de clustering..... 94
Figure 4-1 Le modèle réseau..... 100
Figure 4-2 Le modèle radio [71] 101
Figure 4-3 Le nombre de chemins dans la communication avec un seul saut et celui utilisant multi-saut avec la technique WDC 102
Figure 4-4 L'organigramme du protocole proposé. 103
Figure 4-5 Calcule de WDC et groupe G Figure 4-6 Sélection de CH 104
Figure 4-7 Les étapes suies pour construire une solution 105
Figure 4-8 Le diagramme de classe du protocole WDC-LEACH-C..... 106
Figure 4-9 Diagramme de cas d'utilisation globale du protocole 107
Figure 4-10 Diagramme de cas d'utilisation pour le clustering..... 107
Figure 4-11 Diagramme de cas d'utilisation pour le routage inter cluster des données 108
Figure 4-12 Diagramme de cas d'utilisation pour le routage intra-cluster des données..... 109

Figure 4-13 Diagramme de cas d'utilisation pour les opérations d'un nœud capteur 109
Figure 4-14 Diagramme de séquence des événements du WDC-LEACH-C 110
Figure 4-15 Automate représentant un nœud capteur 112
Figure 4-16 Automate représentant un cluster head 112
Figure 4-17 Automate représentant la station de base 113
Figure 4-18 La durée de vie du réseau 114
Figure 4-19 Le nombre des nœuds vivants 115
Figure 4-20 L'énergie consommée 116
Figure 4-21 Quantité de données reçus par la SB 116
Figure 4-22 Le nombre de paquets reçus par la SB 117
Figure 4-23 Les étapes de la phase de clustering de ACORP 119
Figure 4-24 Les étapes de la phase de routage de ACORP 119
Figure 4-25 L'organigramme de ACORP 120
Figure 4-26 L'organigramme de Improved-ACORP 124
Figure 4-27 Le diagramme de classe du protocole Improved-ACORP 127
Figure 4-28 Diagramme de cas d'utilisation globale du protocole 128
Figure 4-29 Diagramme de cas d'utilisation pour le clustering 128
Figure 4-30 Diagramme de cas d'utilisation pour le routage inter cluster des données 129
Figure 4-31 Diagramme de cas d'utilisation pour le routage intra-cluster des données. 130
Figure 4-32 Diagramme de cas d'utilisation pour le routage intra-cluster des données lors de l'application de CHMS 130
Figure 4-33 Diagramme de cas d'utilisation pour les opérations d'un nœud capteur 131
Figure 4-34 Diagramme de cas d'utilisation pour les opérations d'un nœud superviseur 132
Figure 4-35 Diagramme de cas d'utilisation pour les opérations d'un nœud assistant 132
Figure 4-36 Diagramme de séquence des événements du Improved ACORP 133
Figure 4-37 Diagramme de classe agents pour la stratégie CHMS 135
Figure 4-38 Les étapes de CHMS modéliser par les agents 135
Figure 4-39 La durée de vie du réseau 140
Figure 4-40 Le nombre de nœuds vivants 140
Figure 4-41 L'énergie consommée 141
Figure 4-42 La quantité de données reçues par la SB 141

Liste des tableaux

Tableau 1-1 Comparaison entre systèmes distribués, parallèles et centralisés..... 8
Tableau 1-2 Souches client et serveur..... 17
Tableau 3-1 Les derniers états de l'art sur le clustering dans les RCSFs 85
Tableau 3-2 Le principe et l'objectif des algorithmes de clustering..... 94
Tableau 4-1 Les paramètres de simulation..... 113
Tableau 4-2 Les résultats de simulation..... 117
Tableau 4-3 Les paramètres de simulation..... 139
Tableau 4-4 Les résultats de simulation..... 142

Liste des travaux

Travaux Publiés

1- Slimani, A., Redjimi, M., & Slimani, D. (2017). Weighted Density Center (WDC-LEACH-C) Clustering Protocol for Wireless Sensor Networks (WSN). *Informatica*, *42*(2).

2- A. Slimani, M. Redjimi, D, Slimani, A literature survey on Clustering techniques for diverse Wireless Sensor Networks, ICNAS'2017, Annaba, 119-125, ISBN: 978-9931-9142-0-4

Travaux soumis

1- A. Slimani, M. Redjimi et al. Ant Colony Optimization Protocol for Wireless Sensor Networks, Informatica Slovenia.

2- Slimani A., Redjimi M., D. Slimani, An Agent-based Cluster Head Management Strategy for Hierarchical Wireless Sensor Network, IJSCC, Inderscience.

WSN: Wireless Sensor Network

WDC-LEACH-C : Weighted Density Center-Low Energy Adaptive Clstering Hierarchy

CH : Cluster Head

ACORP : Ant Colony Optimization Routing Protocol

CHMS : Cluster Head Management Strategy

QoS: Quality of Service

IP: Internet Protocol

SD:Système Distribué

DNS: Doamin Name System

TCP: Transport Communication Protocol

UDP: User Datagram Protocol

RMI: Remote Method Invocation

RPC: Remote Procedure Call

API: Application Programming Interface

JVM: Java Virtual Machine

OSI: Open System Interconnexion

RRL : Remote Reference Layer

CORBA: Common Object Request Broker Architecture

MOM : Messaging-Oriented Middleware

LCMP : Loosely Coupled Message Passing

DCOM: Distributed Component Object Model

ORB : L'*Object Request Broker*

IDL : Interface Définition Langage

OMG: Object Management Group

WAN: Wide Area Network

UTC: Temps Universel Coordonné

GMT: Greenwich Mean Time (GMT).

FIFO: First In First Out

Ack: Aquittement

SC : section critique

RCSF : réseaux de capteurs sans fils

WSNs : Wireless sensor networks

SB : station de base

ADC : Analogique Digital Converter

GPS : Global Position System

TDMA : Time Division Multiple Access

CSMA/CD : Carrier Sense Multiple Access / Collision Detection

BS: Base Station

MANET: Mobile ad hoc networks

VANET: Vehicular Ad-Hoc Network

WDC: Weighted Density Center

FND : First Node Died

HLD : Half Node Died

LND : Last Node Died

MAS: Multi agent System

Introduction générale

Incontestablement, ce début de vingt et unième siècle est placé sous le signe de technologie de l'information et de la communication. Les fonctionnalités des systèmes distribués peuvent être considérablement améliorées en expliquant l'utilisation non traditionnelle de la technologie. Après le phénomène Internet, La démocratisation des technologies sans fil révolutionne les moyens de communication avec notamment l'apparition de réseaux spontanés ou réseaux ad hoc. En combinaison de la technologie sans fil et la miniaturisation, les micro-capteurs sont devenus des techniques possibles et populaires dans les applications militaires, de santé, de sécurités, commerciales et industrielles. Ces petits capteurs sont dispersés en grand nombre et de manière autonome, et capables de constituer des réseaux auto-organisés. Les réseaux de capteurs sans fil sont l'une des technologies visant à répondre aux besoins des utilisateurs pour lesquels ils ont apporté des bénéfices aux applications distribués tels que l'amélioration de la couverture, à la création de lignes virtuelles (par exemple, l'interférométrie) ou à l'augmentation de la taille.

Problématique :

De nombreux travaux de recherches ont été menés pour résoudre le problème de gestion de l'énergie liée aux réseaux de capteurs sans fil, tels que : l'établissement de protocoles efficaces de la couche mac pour éviter les collisions, l'augmentation de la qualité des composants électroniques des capteurs et, parmi les solutions les plus populaires, le développement de protocoles de routage efficaces afin d'améliorer la qualité de transmission.

Nous nous sommes intéressés dans nos travaux aux protocoles de routage entre nœuds capteurs permettant une gestion efficace de l'énergie du réseau de capteurs sans fil. Étant données les perspectives applicatives prometteuses des réseaux de capteurs ainsi que la problématique de gestion d'énergie soulevée et considérée comme un point critique dans de nombreux systèmes. L'objectif de cette thèse est d'étudier et proposer de nouveaux protocoles de routage permettant d'apporter des réponses à la problématique de

conservation de l'énergie. Nous résumons ci-dessous les contributions présentées dans cette thèse.

Contributions :

La première proposition concerne un protocole de routage dénommé WDC-LEACH-C (Weighted Density Center - Low Energy Adaptive Clustering Hierarchy) qui est un protocole de routage hiérarchique qui considère le mode de transmission à un seul saut dans lequel les messages sont directement transmis à partir des membres vers les CHs. La pierre angulaire de cette proposition est la sélection des CHs où elle est basée sur le critère d'énergie et le centre de densité du cluster. Le protocole WDC-LEACH-C choisit le point le plus proche des nœuds du cluster comme CH. cette idée réduit considérablement la distance de communication entre les nœuds ce qui évite la dissipation de grandes quantités d'énergie causée par des transmissions sur de longs chemins. La transmission à saut unique réduit le nombre de chemins établis et ainsi la consommation d'énergie et, par voie de conséquence, augmente la durée de vie du réseau. Notre deuxième proposition concerne un protocole de routage hiérarchique intelligent pour les réseaux de capteurs sans fils homogènes auto-organisés basé sur les colonies de fourmis nommé Improved-ACORP (Ant Colony Optimization Protocol). L'idée principale de Improved ACORP est basée sur ACORP qui est lui-même inspiré du comportement des colonies de fourmis réelles et où la distance entre les nœuds et le nombre de sauts pour établir les chemins sont les critères utilisés. Improved ACORP introduit le critère d'énergie dans la sélection des chemins vers le CH et ainsi ; il intègre une stratégie de gestion d'énergie des CHs nommée CHMS (Cluster Head Management Strategy). Cette stratégie intervient lorsque le niveau d'énergie d'un CH a atteint un seuil critique. La conservation d'énergie ainsi que la maximisation de la durée de vie du réseau ont été les buts principaux pour la conception de nos protocoles.
Ce manuscrit est organisé en quatre chapitres suivis d'une conclusion générale. Un état de l'art est présenté dans les deux premiers chapitres et nos contributions sont détaillées dans les deux derniers. Notre premier chapitre est consacré à introduire les notions préliminaires des systèmes distribués. Nous y présentons les définitions des différents concepts gravitant autour de cette thématique telles que les architectures matérielles et logicielles des systèmes distribués, leurs caractéristiques, les intérêts, objectifs et défis ainsi que les techniques de communication et les modèles d'interaction dans les systèmes distribués

Dans le deuxième chapitre, nous étudions les différents algorithmes qui modélisent les techniques des échanges et de la coopération dans les systèmes distribués. La coopération entre les différents composants d'un système distribués est réalisée dans différentes opérations telles que l'élection d'un leader, l'exclusion mutuelle, la terminaison des applications distribuées, la diffusion et la validation atomique et les transactions distribuées. Le troisième chapitre, concerne les réseaux de capteurs sans fils. Nous commençons par présenter les concepts de base de ces types de réseaux, leurs architectures et celles des capteurs. Ensuite, nous présentons les sources de dissipation d'énergie et les techniques de sa conservation. Nous poursuivons avec une étude bibliographique concernant l'axe de recherche, à savoir, les protocoles de routage et nous abordons des protocoles à architecture hiérarchique clustérisée. Nous présentons les caractéristiques et les objectifs de cette architecture. Nous survolons l'état des travaux existants recensés dans la littérature, en effectuant une analyse de ces travaux nous proposons une taxonomie basée sur les caractéristiques du réseau. Dans le quatrième chapitre, nous présentons nos contributions liées aux protocoles de routage Notre première contribution consiste dans la proposition d'un protocole de clustering nommé WDC-LEACH-C pour établir une architecture hiérarchique clustérisée avec une meilleure sélection de CH afin de minimiser la consommation d'énergie dans les clusters. Dans notre seconde contribution, nous proposons un second protocole de routage nommé Improved-ACORP qui intègre une stratégie de gestion d'énergie des CHs nommée CHMS afin de conserver le maximum d'énergie des CHs et ainsi prolonger la durée de vie du réseau. Une modélisation formelle et opérationnelle est effectuée. Plusieurs simulations ont été réalisées pour démontrer la faisabilité et l'efficacité de nos solutions. Nous finalisons ce manuscrit par une conclusion générale. Nous rappelons les différentes contributions réalisées tout au long de ce travail de recherche et nous présentons quelques perspectives de recherche à ce travail.

Chapitre 1 : Introduction aux systèmes distribués

1.1 Introduction

On assiste, ces dernières années, à une véritable révolution dans les nouvelles technologies de l'informatique, de la micro-électronique et des réseaux de communications. Ces nouvelles technologies ont un impact direct et important sur tous les domaines de notre vie aussi bien professionnelle que domestique. Les réseaux informatiques actuels permettent d'interconnecter des ensembles gigantesques de machines le plus souvent hétérogènes et géographiquement dispersées ensemble. L'usager utilise, ainsi, un ensemble de machines sur le réseau sans pour autant les connaitre ; c'est le principe de la transparence. Cette explosion spectaculaire dans le domaine des nouvelles technologies de la communication et de la communication alliant la micro-électronique, l'informatique et les télécommunications soulève de nouveaux défis concernant l'optimisation des traitements et du stockage distribués ainsi que des protocoles d'échange, de coopération, de synchronisation et de la gestion du temps qui devient une notion relative vu que ces systèmes ne sont pas censé être régis par une même horloge. Aussi, et pour répondre à ces nouveaux défis, les systèmes distribués sont au cœur d'attentions considérables de la part des chercheurs et des industriels. Ces systèmes ont été rapidement considérés comme une réponse à une forte demande dans différents secteurs de l'informatique : calcul réparti, simulation, modélisation, base de données, serveurs multimédia... Les systèmes distribués peuvent être composés de nombreuses machines différentes en interaction et, de ce fait, sont dynamiques et complexes à bien des points de vue, que ce soit de par leur structure ou leur comportement. Les systèmes distribués bénéficient de plusieurs fonctionnalités comme la redondance, la robustesse, la fiabilité et la capacité de croissance incrémentielle. Cependant, la complexité potentiellement très élevée de ces systèmes est un défi important et problématique pour les développeurs d'applications distribuées. De même, d'autres problèmes sous-jacents constituent un risque majeur notamment en ce qui concerne la sécurité, la confidentialité, l'intégrité, les pannes

des réseaux ainsi que la distribution des logiciels d'exploitation, de l'exactitude et de la qualité de service (QoS).

Dans ce chapitre, nous décrivons deux axes concernant les systèmes distribués ; le premier concerne l'architecture et le fonctionnement de ces systèmes : les composants matériels et logiciels, les caractéristiques, les intérêts, les objectifs et défis ainsi que les modèles d'interaction requis pour établir les communications dans les systèmes distribués. Le deuxième axe concerne les technologies middleware qui lient les composants des systèmes et facilitent l'interopérabilité.

1.2 **Définitions**

Dans la littérature plusieurs définitions sont données aux systèmes informatiques distribués. Selon Tanenbaum [1] *'Un système distribué est un système qui s'exécute sur une collection d'ordinateurs indépendants qui apparait aux utilisateurs comme un système unique intégré et cohérent'.* Selon Coulouris [2] *'Un système reparti est un ensemble de machines autonomes connectées par un réseau de communication et communiquent entre eux via ce réseau par le passage des messages. Ce système équipé d'un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ses ressources.*' Selon Lamport [3] '*un système reparti est un système qui nous empêche de travailler quand une machine dont nous n'avons jamais entendu parler tombe en panne'.*

A partir de ces définitions on peut dire qu'un système distribué peut être vu comme un ensemble de machines autonomes, au minimum deux, (ordinateurs, PDA, processeur, processus, processus légers, imprimantes, capteurs,…), interconnectées via un réseau de communication et qui peuvent communiquer et partager des ressources. Chaque machine exécute un ensemble de composantes logicielles, par exemple des séquences de calculs, issues du découpage d'un projet de calcul global, et utilise un intergiciel (middleware), qui s'occupe d'activer ces composantes et de coordonner leurs activités. La motivation principale d'un système distribué et de coopérer et partager plusieurs ressources pour effectuer un traitement parallèle de telle sorte qu'un utilisateur perçoive le système comme un unique système intégré.

1.3 **Aspects matériels des systèmes distribués**

Les systèmes distribués sont, dans leur définition la plus simpliste, des logiciels dont l'exécution se déroule sur différentes entités de calcul ou machines multiprocesseurs distribuées géographiquement et reliées entre elles par des interconnexions réseau [1] [2].

Ces entités ne possèdent ni horloge globale ni mémoire commune. Autrement dit, un système distribué est défini comme étant un ensemble de ressources physiques et logiques (ordinateurs, PDA, processeurs, processus, processus léger etc.) géographiquement dispersées et reliées par des réseaux de communication comme illustré figure 1-1.

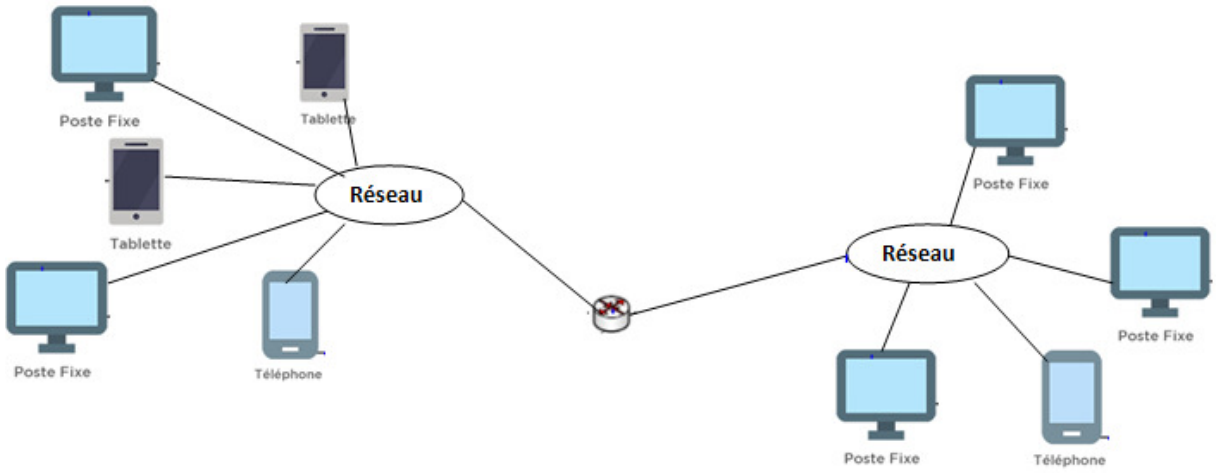

Figure 1-1 Représentation d'architecture matérielle des systèmes distribués

1.4 **Aspects logiciels des systèmes distribués**

Le matériel des systèmes distribués constitue l'infrastructure physique sur laquelle s'exécutent les différentes applications. Ses caractéristiques et performances sont autant de facteurs favorisant les vitesses de calcul et les capacités de stockages. L'aspect complémentaire est fourni par les différents logiciels, environnements, systèmes et plateformes de développements, d'exploitation et d'implémentation. Un système distribué est souvent vu comme une application qui coordonne les taches de plusieurs équipements informatiques (ordinateurs, téléphones mobiles, PDA, capteurs). Cette coordination se fait le plus souvent par envoi de messages et utilise un intergiciel (middleware) via un réseau de communication qui peut être de type LAN, WAN ou Internet. Les exécutions de taches réparties et concourantes sont organisées par l'utilisation de différents outils et mécanismes de synchronisation. La figure 1-2 présente l'architecture logicielle d'un système distribué [4].

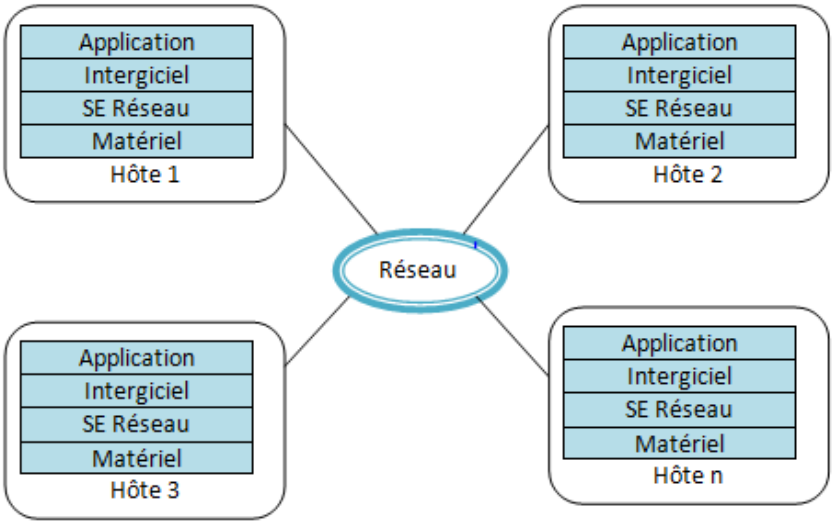

Figure 1-2 Architecture logicielle d'un système distribué.

1.5 Caractéristiques des systèmes distribués

Un système distribué est caractérisé par les propriétés suivantes

- la concurrence : le parallélisme est la principale caractéristique des systèmes distribués vu que des composants autonomes peuvent exécuter des tâches concurrentes.
- Les systèmes distribués disposent de mémoires locales non partagées
- Dans un système distribué, les processus n'ont pas de vision globale du système : Il n'y'a pas de notion d'état global.
- Il n' y'a pas d'horloge globale ce qui implique qu'il n'est pas possible de synchroniser les horloges de manière précise
- Chaque composant dispose de ses propres « unités de contrôle » et de ses propres mémoires ainsi que d'autres ressources physiques lui permettant d'être autonome.

À partir de ces caractéristiques, il apparait clairement qu'un système centralisé présente beaucoup de différences par rapport à un système distribué. Ces points de différence sont récapitulés dans le tableau 1-1 ci-dessous.

Tableau 1-1 Comparaison entre systèmes distribués, parallèles et centralisés.

Réseau Vs SD	SC. Vs SD.	Système parallèle Vs SD.
Le réseau : est un “media” pour interconnecter des ordinateurs locaux et distants s’échanger des messages en respectant des protocoles. Les composantes d’un réseau ont des adresses IP.	Système centralisé : Système logiciel s'exécutant sur une seule machine Accédant localement aux ressources nécessaires (données, code, périphériques, mémoire ...)	Calcul Parallèle : homogénéité, haut niveau de couplage des tâches. – **Un système est dit parallèle** lorsque les processus communiquent avec de la mémoire partagée (par exemple des threads).
Un système distribué : un réseau + une couche logicielle " Les systèmes distribués s’intéressent aux applications. " Ils reposent sur les services du réseau	Système distribué : Ensemble d'ordinateurs indépendants connectés en réseau et communiquant via ce réseau Cet ensemble apparaît du point de vue de l'utilisateur comme une unique entité	La programmation distribuée : – Chaque composant s'exécute sur un matériel interconnecté par un réseau local ou global. – Les processus envoient entre eux des messages. Distribué : hétérogénéité, indépendance logique et physique des tâches, client-serveur

1.6 **Intérêts des systèmes distribués**

Les systèmes distribués sont largement étudiés et ils fournissent des solutions efficaces, performantes et économiques [5] :

- la possibilité d'utiliser et de partager des ressources logicielles et matérielles couteuses distantes par plusieurs utilisateurs permet l’optimisation des coûts. Par exemple,

l'utilisation partagée de ressources matérielles telles qu'une imprimante ou des applications.

- Mise en commun d'un nombre théoriquement infini de ressources pour de faibles coûts. Ce qui conduit à de très bons rapports performances/prix.

- Accès à distance à des ressources indisponibles en local.

- Possibilités d'accès aux ressources depuis tous les endroits du système

- Puissance globale virtuellement illimitée supérieure à celle des gros calculateurs.

- Equilibrage de charge (load balancing) pour garantir la qualité de service (QoS)

- Système plus robuste et fiable par la réplication locale des ressources globales et duplication des ressources avec sauvegarde.

- Distribution de gros traitements tels que les calculs scientifiques pouvant être répartis sur un ensemble de machines.
- Délocalisation d'espaces de stockage (stockages dans des data centers).
- Accès au cloud computing.
- Facilité d'intercommunication entre utilisateurs.
- Flexibilité et facilité d'extension du système (matériels et logiciels).
- Grande tolérance aux pannes (fiabilité et disponibilité) : Un élément peut tomber en panne sans bloquer tout le système, même si le système devrait fonctionner en mode dégradé.

1.7 **Les défis et les propriétés requises des systèmes distribués**

Un système distribué doit assurer plusieurs propriétés pour être considéré comme performant. Nous citons dans cette section ceux qui nous semblent être les plus importants [5].

1.7.1 **La transparence**

La transparence est une propriété importante des systèmes distribués. Cette notion signifie que la distribution est généralement cachée pour l'utilisateur et les programmeurs de l'application. Un système transparent permet de cacher les répartitions des composants d'un SD ainsi que les détails techniques et organisationnels. L'objectif est de pouvoir faire bénéficier les applications d'une multitude de services sans avoir besoin de connaitre

exactement la localisation ou les détails techniques des ressources qui les fournissent. Le système est perçu par les utilisateurs comme une application unique cohérente et non comme une collection des composants indépendants répartis sur un ensemble de sites. Cette notion permet le développement des applications en donnant à l'utilisateur l'impression qu'elles sont développées sur un système centralisé. Ceci fait que la fourniture de transparence exerce une influence considérable sur la qualité des systèmes. Plusieurs types de transparences sont définis pour un SD.

- Transparence d'accès : ceci permet de manipuler des données et l'accès à des ressources distantes comme si c'était des données et ressources locales. L'accès aux données se faisant indépendamment de leur format de représentation
- Transparence de localisation : il s'agit de masquer l'emplacement des ressources et permet l'accès aux éléments indépendamment de leur localisation.
- Transparence de concurrence : il s'agit de cacher aux utilisateurs la gestion de la concurrence dans un SD c'est à dire la possibilité d'exécuter plusieurs processus en parallèle tout en utilisant une ressource partagée par plusieurs utilisateurs compétitifs.
- Transparence de réplication : il s'agit de cacher la possibilité de dupliquer certains éléments (ressources) pour augmenter la fiabilité et améliorer la performance du système.
- Transparence de la mobilité et migration : il s'agit de masquer la possibilité qu'une ressource puisse changer d'emplacement sans que celle-ci ne soit aperçue. Ce type de transparence permet la migration des ressources et des clients à l'intérieur du système sans influencer le déroulement des applications.
- Transparence de relocalisation : il s'agit de cacher le fait qu'une ressource puisse changer d'emplacement au moment où elle est utilisée (lors de son utilisation).
- Transparence de défaillance (panne) : permettant à un utilisateur de ne pas s'interrompre à cause d'une panne. Il s'agit de masquer les défaillances et cacher à l'utilisateur l'occurrence de pannes et le recouvrement possible des objets lorsqu'un ou plusieurs éléments tombent en panne.
- Transparence d'échelle : il s'agit de la possibilité d'augmenter la taille du système et d'ajouter de nouvelles ressources sans perturber son organisation.
- Transparence de persistance : il s'agit de masquer si une ressource logicielle est en mémoire interne ou sur un support externe.

- Transparence d'hétérogénéité : L'utilisateur n'a pas à se soucier des différences matérielles ou logicielles des ressources qu'il utilise ; c'est la notion d'interopérabilité.

Dans la pratique, un système distribué donné doit offrir un minimum de transparences telles que : la transparence de localisation, d'accès et de concurrence. Minimum qui doit être toléré, accepté et supporté par ces systèmes.

1.7.2 **L'ouverture (Openness)**

Un système ouvert est un système qui peut être étendu en nombre d'éléments matériels qui le constituent et il offre la possibilité d'ajouts de nouveaux services ou d'implémentation de services existants aux niveaux logiciels. Un système ouvert assure l'extensibilité, la flexibilité et la portabilité. Le fonctionnement se base sur des interfaces d'interaction clairement définies et supportées.

1.7.3 **La mise à l'échelle et le dimensionnement (Scalability)**

Un système distribué doit être relativement facile à étendre et à augmenter en dimension. Il doit fonctionner correctement et efficacement lorsque le nombre d'utilisateurs et de ressources augmente ; c'est-à-dire qu'il doit pouvoir offrir la possibilité de rajouter des ressources sans modifier la structure du système. La complexité des algorithmes distribués ne doit pas dépasser un certain seuil lorsque la taille des participants augmente.

1.7.4 **La concurrence**

L'autonomie des composants distribués fait que plusieurs processus répartis sur des sites différents puissent être exécutés simultanément (programmation concurrente). Un système distribué doit offrir et gérer les accès concurrents aux ressources partagées en assurant un ordonnancement juste et performant et en évitant les inter-blocages.

1.7.5 **L'hétérogénéité**

Par essence, un système distribué doit pouvoir être séparable en plusieurs entités entièrement autonomes. Il n'est, ainsi, pas envisageable de désigner une ou plusieurs composantes maître qui contrôle les autres. Chacune des composantes doit, donc, être capable d'assurer son propre fonctionnement et contrôle. L'usager n'a pas à se soucier des différences matérielles ou logicielles des ressources qu'il utilise. Cette notion d'interopérabilité permet, entre autres, de disposer et d'utiliser des technologies hétérogènes pour une ou plusieurs composantes. Ces dernières peuvent être écrites en utilisant des langages de programmation totalement différents (Java, Cobol, C++, etc.) et s'exécuter sous divers systèmes d'exploitation (Mac OS X, Linux, Windows, etc.). Ainsi,

cette hétérogénéité dans les systèmes distribués peut intervenir à différents stades et se présenter à des niveaux divers : au niveau des réseaux qui peuvent être de différents types, au niveau du matériel informatique, du codage des données, au niveau des systèmes d'exploitation, au niveau des langages de programmation et autres [6].

1.7.6 **La disponibilité**

On dira d'un système qu'il est disponible s'il peut délivrer correctement les services requis de manière conforme à leurs spécifications et cahiers de charges. Ainsi, afin de rendre un système disponible, il y'a lieu, donc, de lui donner la capacité de faire face à tout obstacle qui peut compromettre, nuire et empêcher son bon fonctionnement. En effet l'indisponibilité d'un système peut être causée par plusieurs sources parmi lesquelles nous pouvons citer : les pannes, les attaques et les surcharges du système. Il y'a lieu de prévoir, ainsi, des modules supplémentaires dans le but d'assurer la disponibilité (gestion de réplication, détection et résolution des pannes et politiques de sécurité)

1.7.7 **La sécurité**

Dans le cas des systèmes centralisés, leur sécurisation est relativement simple. Le déroulement des différentes applications peut être contrôlé, vérifié et supervisé de façon instantanée et assez facilement étant donnée que la vision complète de toutes les ressources octroyées au système à tout instant est possible. Par contre, dans un système distribué, la sécurisation reste une problématique cruciale dont la garantie est loin d'être assurée. Des protections draconiennes doivent être mises en œuvre pour la protection des applications, des ressources et des réseaux contre toutes tentatives d'intrusions et de manipulations malveillantes. Le système ne doit être utilisé que pour ses fonctions prévues. Les accès ne doivent être permis qu'aux utilisateurs autorisés Cependant et malgré toutes les précautions que l'on peut envisager et toutes les techniques imaginées et utilisées, le risque zéro n'existe pas. Ainsi et de par leur complexité et leur ouverture au monde extérieur, les systèmes répartis déployés sur l'Internet ne peuvent être totalement infaillibles. Ainsi, contre l'ingéniosité et la technicité des protections mises en œuvre, les attaquants potentiels déploient sans cesse de nouvelles menaces pour les détourner et les neutraliser. Un système distribué doit assurer : la confidentialité, l'intégrité, la disponibilité et la non-répudiation. On compte, à l'heure actuelle plusieurs mécanismes de sécurité, tels que le cryptage (Blowfish, RSA,…), l'authentification (password, public key authentication),… [7].

1.7.8 **La tolérance aux fautes**

Contrairement aux systèmes centralisés, les systèmes distribués peuvent révéler plusieurs points de défaillances (problème de composantes, réseau, trafics, etc.). La tolérance aux fautes dans un système distribué concerne la possibilité de continuer à fournir les services requis même en cas de pannes ou d'erreurs. Cette tolérance aux fautes peut être assurée par plusieurs mécanismes parmi lesquels, on peut citer : La détection des fautes, les sommes de contrôle, le *heartbeat*, le masquage de fautes (recommencer l'opération échouée), la réplication sur plusieurs routes IP, la réplication des données DNS et des bases de données [8].

1.8 **La communication dans les systèmes distribués**

Les systèmes distribués sont constitués d'un ensemble d'entités logicielles qui s'exécutent sur des machines reliées par des réseaux de communication et communiquent via ces réseaux. La communication interprocessus dans un système centralisé se fait via la mémoire partagée mais un système distribué est caractérisé, en général, par l'absence d'une telle mémoire partagée étant donnés que les différents éléments d'un système distribué sont placés sur des machines distinctes et distantes et ont chacun leur propre espace mémoire. Pour permettre l'échange de données entre ces espaces mémoire séparés, il faut des outils de communications communs. Nous allons en traiter trois : l'échange de messages, l'appel de procédures à distance et la mémoire virtuellement partagée. On doit pouvoir communiquer par l'utilisation de protocoles sachant qu'un protocole est un ensemble de règles qui gèrent les communications interprocessus dans un système distribué. Il détermine l'accord sur la façon dont les communications doivent s'effectuer.

1.8.1 **L'échange de messages**

La communication par échanges de messages est la méthode de communication la plus basique. C'est une méthode de communication directe interprocessus. Il s'agit de l'envoi d'un message d'un élément vers un autre. Pour qu'une entité Y puisse communiquer avec une entité X, elle doit lui envoyer un message contenant l'information. Cet envoi de message est explicite au niveau du paradigme de programmation et doit être pris en charge par le développeur. Il existe de nombreux mécanismes et de nombreux outils pour l'envoi de messages, les sockets sont probablement l'outil le plus utilisé pour développer l'envoi de messages. Il en existe d'autres, nous pouvons citer : Les bibliothèques de communication de groupes pour la programmation parallèle comme MPI ou PVM, ou encore les messages actifs.

1.8.2 **Les sockets**

Un socket est un outil de communication universel de bas niveau qui permet l'utilisation de l'interface de transport standard TCP, UDP introduit dans les années 80. Un socket est une unité communicante dans un système distribué et elle est considérée comme l'extrémité d'un canal de communication permettant de transmettre et de recevoir des données à travers le réseau. Les sockets peuvent être utilisés en mode connecté (fonctionnant généralement au-dessus du protocole TCP) ou en mode déconnecté (fonctionnant généralement sur UDP) [11].

1.8.2.1 Principe de fonctionnement

Le principe de fonctionnement des sockets se résume en trois phases pour un simple acheminement en mode connecté TCP

1- le serveur crée un "socket serveur" associé à un port et se met en attente
2- le client se connecte au socket serveur ; deux sockets sont alors créés ; un "socket client" du coté client et un "socket service client" du côté serveur. Ces sockets sont connectés entre elles.

3- le client et le serveur communiquent par l'intermédiaire des sockets. L'interface est celle des fichiers (lecture (), écriture()). Le socket serveur peut accepter de nouvelles connexions.

La figure 1-3 représente un simple acheminement utilisant les sockets TCP

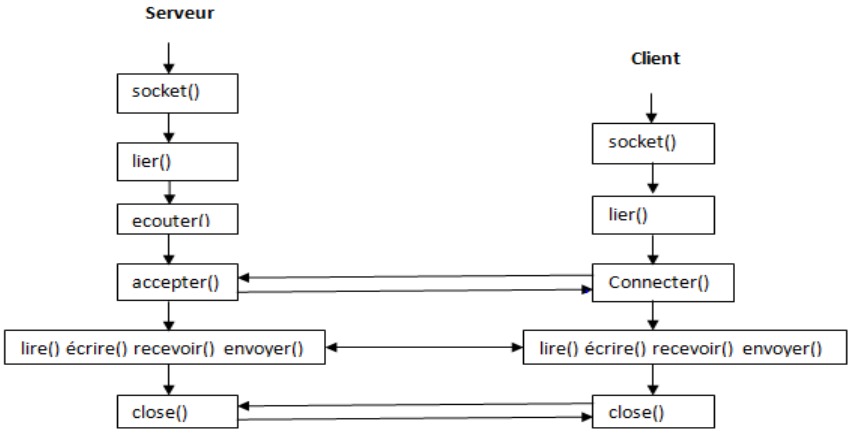

Figure 1-3 Acheminement simple du socket TCP.

1.8.2.2 Programmation des systèmes distribués avec les sockets

Les différentes opérations de gestion des sockets sont fournies comme primitives (appels système) dans UNIX et d'autres systèmes d'exploitation. Ces Operations peuvent être

utilisées directement mais les langages usuels fournissent des outils facilitant leur usage et couvrant les cas courants coté client et coté serveur.

Lors de la réalisation des sockets avec le modèle client/serveur, s'il y a de multiples clients, ils sont traités en séquence et les requêtes sont conservées dans une file d'attente associée au socket serveur. La longueur maximale de cette file d'attente peut être spécifiée lors de la création du socket serveur. Un autre schéma veilleur-exécutant peut être utilisé à l'aide des threads comme suit

- Le thread veilleur doit créer explicitement un nouveau thread s'exécutant à chaque nouvelle connexion d'un client,
- un socket service client différent est créé pour chaque client,
- le thread veilleur se remet ensuite en attente

Tant que (vrai)
Début

accepter la connexion d'un nouveau client

créer un socket service client

créer un thread pour interagir avec ce client sur le nouveau socket service client

Fin

1.8.2.3 Types de socket

On distingue les types de sockets suivants :

- Raw socket Communication via IP en gardant l'entête (capture de trafic).
- Datagram socket Communication au-dessus d'UDP (non-connecté).
- Datagram socket Communication au-dessus d'UDP (non-connecté
- Stream socket Communication au-dessus de TCP (ordre, délivrance, . . .)
- WebSockect Utilise le port 80 du protocole TCP.

L'envoi de messages n'est pas exprimé dans un paradigme de programmation local, la question de sa transparence n'a donc pas de sens. Cependant, il constitue l'élément de base pour les communications dans un système distribué. C'est sur ce mécanisme que repose l'implantation des deux autres outils de gestion de la mémoire répartie que nous présentons dans la suite : la mémoire virtuellement partagée et l'appel de routines à distance. Dans les

sections suivantes nous allons étudier les mécanismes des appels distants. Nous allons voir l'appel de procédures à distance (RPC) ainsi que l'invocation de méthodes à distance (RMI).

1.8.3 **Appel de procédures à distance**

L'appel de procédures permet le transfert du contrôle et des données entre deux procédures présentes sur une même machine. La communication entre machines peut s'effectuer par l'appel de procédures à distance. Dans ce cas, une machine A demande à une machine B d'effectuer un traitement et de lui retourner un résultat. L'appel de procédures à distance permet à une machine d'appeler une procédure dont le code réside sur une autre machine et étend la notion de procédure locale à celle de procédures situées sur des machines différentes et distantes. Lorsqu'un appel est effectué et fait référence à une procédure distante, l'appelant est suspendu, les arguments de la procédure appelante sont transférés, via le réseau, sur la machine où la procédure doit s'exécuter. Sur la machine de l'appelé, la procédure souhaitée est exécutée avec les arguments reçus par le réseau. Lorsque cette procédure se termine, le résultat de celle-ci est transféré, via le réseau, vers la machine de l'appelant. L'appelant redémarre en récupérant le résultat de la procédure appelée comme si celle-ci s'était exécutée localement. L'un des protocoles les plus répandus pour l'appel de procédures à distance est ONC RPC (Open Network Computing Remote Procedure Call) connu aussi sous le nom de *Sun RPC*.

1.8.3.1 Remote Procedure Call (RPC)

Remote Procedure Call (RPC) est un protocole permettant de lancer une procédure d'un programme sur un ordinateur distant. Cette méthode permet au programmeur de réaliser assez simplement des programmes Client/Serveur. Ainsi, la génération du code des procédures appelantes de même que leurs programmations et leurs adaptations sont facilitées grâce à l'utilisation du RPC. Dans le RPC, la communication est stockée dans une librairie de procédures. L'utilisation d'un service à distance se fait par l'appel d'un processus de la procédure correspondante qui forme un message rassemblant le code des opérations et les paramètres dans un message, puis envoie ce message et attend la réponse. C'est là un processus qui fonctionne bien, cependant il présente quelques limites dont celui de la difficulté de transférer des données graphiques ou des structures de données complexes qui contiennent des pointeurs ou autres adresses. C'est le cas aussi quand il s'agit de manipuler des variables globales. De même, le transfert de tableaux complexes est une tâche qui revient chère car il se fait par valeur et non pas par référence. En général,

le paradigme de programmation est muni de mécanismes préexistants pour effectuer l'appel de routines à distance [12]. La figure 1-4 illustre les étapes d'un appel de procédures à distance. Le tableau 1-2 présente le fonctionnement des souches clients et la souche serveur.

Tableau 1-2 Souches client et serveur

La souche client ("client stub")	La souche serveur ("server stub")
1 -> Procédure coté client qui reçoit l'appel en mode local 2 -> Le transforme en appel distant 3 -> En envoyant un message 8 -> Reçoit le message contenant les résultats après l'exécution 9 et 10 -> Retourne les résultats comme c'est le cas d'un retour de procédure	3-> procédure coté serveur qui reçoit le message d'appel distant 4-> transforme le message d'appel en mode local 5-> demande l'exécution sur le site serveur par la procédure serveur 6 et 7-> récupère les résultats 8-> retransmet les résultats par message

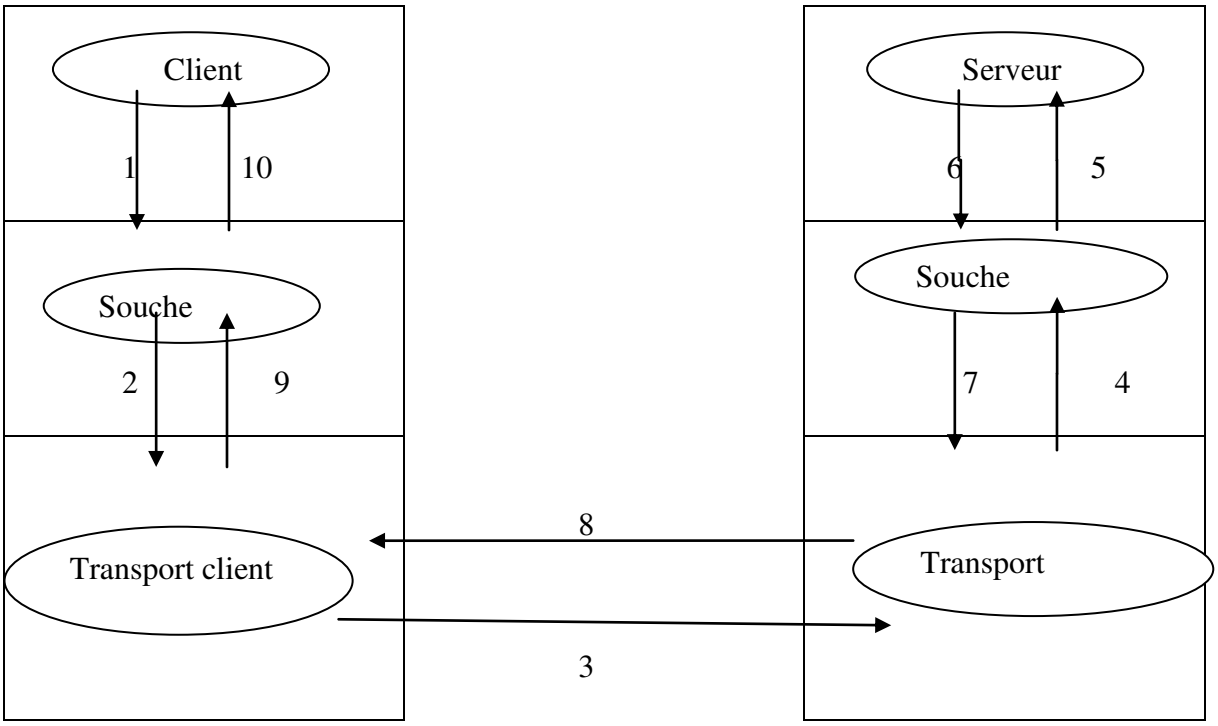

Figure 1-4 Les étapes d'un appel de procédure à distance par message

1.8.4 **Appel de méthodes à distance**

L'appel de méthodes à distance permet d'exploiter un objet distant et d'invoquer une méthode sur cet objet. Pour cela, la notion de référence sur un objet est étendue au cadre distant, c'est-à-dire qu'une référence peut représenter un objet situé en réalité sur une machine distante. L'appel d'une méthode sur cette référence provoquera l'appel de la méthode en question à distance. Parmi les plates-formes permettant l'appel de méthode à distance on trouve Java RMI. Le procédé généralement utilisé est celui du *stub*. C'est-à-dire qu'une référence sur un objet distant est en fait une référence sur un objet local, un *stub*, dont le rôle est de transférer tous les appels de méthode qu'il reçoit vers l'objet distant qu'il représente. Le fonctionnement, dans son principe, est finalement très proche de l'appel de procédure à distance [13].

1.8.4.1 Java RMI

RMI (Remote Method Invocation) est une API Java permettant de manipuler des objets distants c'est à dire des objets instanciés sur une autre machine virtuelle ou éventuellement sur une autre machine du réseau de manière transparente pour l'utilisateur, c'est à dire de la même façon que si l'objet était sur la machine virtuelle (JVM) de la machine locale. Ainsi un serveur permet à un client d'invoquer des méthodes à distances sur un objet qu'il instancie. Deux machines virtuelles sont donc nécessaires (une sur le serveur et une sur le

client) et l'ensemble des communications se fait en Java. La transmission des données se fait à travers un système de couches, basées sur le modèle OSI afin de garantir une interopérabilité entre les programmes et les versions de Java [14]. La figure 1-5 illustre les étapes d'un appel de méthode à distance

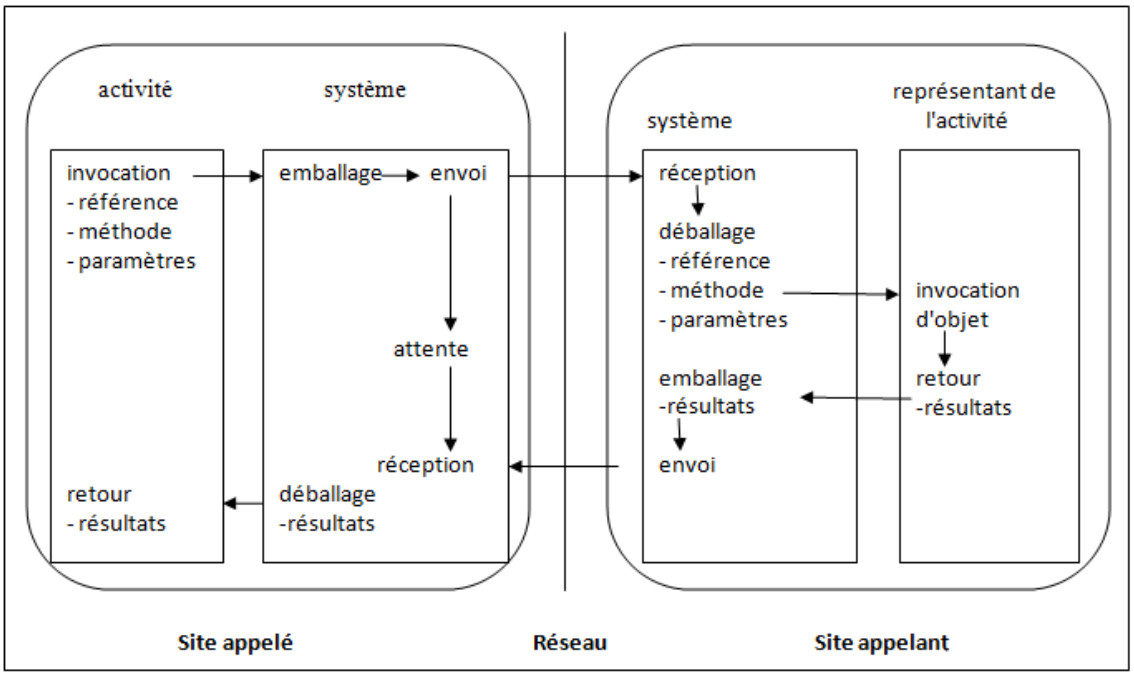

Figure 1-5 Les étapes d'un appel de méthode à distance

- La souche (stub) et le squelette (skeleton), respectivement sur le client et le serveur, assurent la conversation des communications avec l'objet distant

- La couche de référence (RRL, Remote Reference Layer) est chargée du système de localisation afin de fournir un moyen aux objets pour obtenir une référence à l'objet distant. Elle est assurée par le package java.rmi.Naming. On l'appelle généralement registre RMI car elle référence les objets.

- La couche de transport permet d'écouter les appels entrants ainsi que d'établir les connexions et le transport des données sur le réseau par l'intermédiaire du protocole TCP. Les packages java.net.Socket et java.net.SocketServer assurent implicitement cette fonction.

1.8.5 **Les intergiciels intégrés (middleware)**

Le mot intergiciel ou logiciel médiateur (middleware en anglais) désigne un ensemble de logiciels ou de technologies informatiques qui servent d'intermédiaire entre les applications. Ces logiciels peuvent, aussi, être définis comme étant des outils de

communication entre des clients et des serveurs. Ils fournissent un moyen aux clients pour trouver leurs serveurs et aux serveurs pour trouver leurs clients et, en général, de trouver n'importe quel objet atteignable. En ce sens qu'ils peuvent être vus comme des bus de communication auxquels les applications se connectent par l'intermédiaire d'interfaces. Les intérêts que présente un middleware sont multiples et peuvent être résumés en quatre points principaux [15] :

– Cacher la répartition, c'est-à-dire le fait qu'une application soit constituée de parties interconnectées s'exécutant à des emplacements géographiquement répartis ;

– Cacher l'hétérogénéité des composants matériels, des systèmes d'exploitation et des protocoles de communication utilisés par les différentes parties d'une application,

– Fournir des interfaces uniformes, normalisées, et de haut niveau aux équipes de développement et d'intégration pour faciliter la construction, la réutilisation, la portabilité et l'interopérabilité des applications,

– Fournir un ensemble de services communs réalisant des fonctions d'intérêt général pour éviter la duplication des efforts et faciliter la coopération entre applications.

Les processus communiquent par l'intermédiaire d'un intergiciel qui gère les messages. Il se présente comme une couche logicielle masquant l'hétérogénéité et proposant des services de haut niveau (e.g., RPC, échange de messages, notification). Exemples d'intergiciels :CORBA, J2EE, DCOM/DCOM+, Messaging-Oriented Middleware (MOM) ou plus récemment de Loosely Coupled Message Passing (LCMP)

1.8.5.1 Mise en oeuvre des intergiciels

De façon pratique, la mise en œuvre d'un intergiciel suit les points généraux suivants :

– Fournir une interface ou API de haut niveau aux applications,
– Masquer l'hétérogénéité des systèmes matériels et logiciels sous-jacents,
– Rendre la répartition aussi invisible (transparente) que possible,
– Faciliter la programmation répartie (développement, évolution, réutilisation, portabilité des applications),
- cacher l'hétérogénéité des plateformes mises en œuvre pour les applications.

1.8.5.2 CORBA

CORBA (Comon Object Request Broker Architecture) est un standard développé par l'OMG (Object Management Group). L'objectif de CORBA est de fournir un standard

pour la gestion d'objets distribués. CORBA permet l'appel de méthodes à distance. L' *Object Request Broker* (ORB) fournit les mécanismes par lesquels des objets font des requêtes et reçoivent des réponses, et ce de manière transparente. Il fournit également l'interopérabilité entre des applications sur différentes machines dans des environnements distribués hétérogènes et il interconnecte de multiples systèmes objets. Le cœur de CORBA est constitué par l'ORB qui est le conduit/bus par lequel les requêtes sur les objets transitent. L'*Object Request Broker* ou ORB n'est pas directement accessible au programmeur d'applications CORBA qui n'utilise qu'une référence. L'ORB gère les transferts entre les clients et les serveurs. L'adaptateur d'objets, est quant à lui, chargé de la gestion des références aux objets et de l'activation des implémentations : l'adaptateur d'objets est donc, comme son nom l'indique, le " canal " par lequel l'ORB est connectée à l'implémentation de l'objet. L'ORB est défini plutôt par ses interfaces que comme un unique composant. Parmi les caractéristiques de CORBA on peut citer le langage IDL (Interface Définition Langage) qui permet de spécifier l'interface d'un objet, indépendamment du langage dans lequel celui-ci est implanté ou utilisé (voir figure 1-6). Le langage IDL est en fait le pendant du langage RPC mais il offre, en outre, l'héritage d'interface. L'ORB permet la communication entre objets clients et serveurs, quel que soit le langage d'implémentation des objets, leurs environnements respectifs, leurs positions et leurs états dans le réseau. L'ORB masque l'ensemble de l'architecture sous-jacente (réseau, système, etc..). CORBA spécifie également de nombreux services : nommage, événements, persistance, etc. Cependant, pour être conforme à la spécification CORBA, une plate-forme doit simplement implanter la spécification CORBA-Core, c'est-à-dire le principe de l'appel de méthode à distance tel que défini dans CORBA ainsi qu'une correspondance du langage IDL vers au moins un langage de programmation. L'une des forces de CORBA est de permettre l'appel de méthodes à distance indépendamment des architectures et des langages utilisés pour implanter appelant et appelé. Nombre de langages bénéficient d'une traduction du langage IDL standardisé par l'OMG [16].

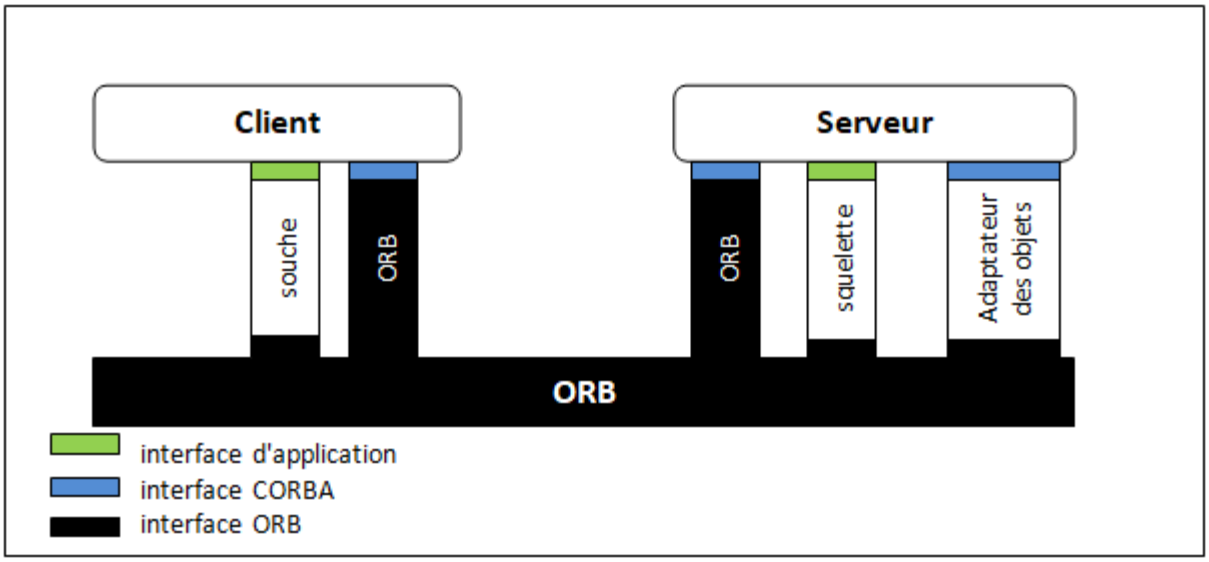

Figure 1-6 Le bus CORBA

1.8.5.3 Implémentation d'un système distribué avec un middleware

La plupart des systèmes distribués sont implémentés avec un middleware comme le montre la figure 1-7.

1.8.5.4 Architecture distribuée

Dans l'approche distribuée, les tâches du middleware sont réparties sur plusieurs sites. Ceci fait qu'il est possible d'accéder au middleware de façon parallèle. Ce type d'architecture supporte mieux les pics de charges en répartissant au mieux les demandes sur les différents composants répartis du middleware. Il reste, en contrepartie, que la maintenance et la gestion cohérente des entités des systèmes deviennent beaucoup plus complexes et plus difficiles. Les composants du middleware sont amenés à exécuter des taches collaboratives afin d'éviter des incohérences. Tout ceci implique l'envoi de messages ou le partage de structures de données et procure, ainsi et par voie de conséquence, des surcharges supplémentaires. Une autre approche de la distribution d'un middleware consiste à avoir plusieurs instances du middleware qui s'exécutent sur un ensemble de sites. Cette technique, en plus du fait qu'elle supporte nettement mieux la tolérance aux pics de charges, offre aussi une meilleure parade aux pannes. En fait, la duplication du middleware sur plusieurs sites permet de masquer la panne de l'une des instances du middleware ou du site qui l'héberge.

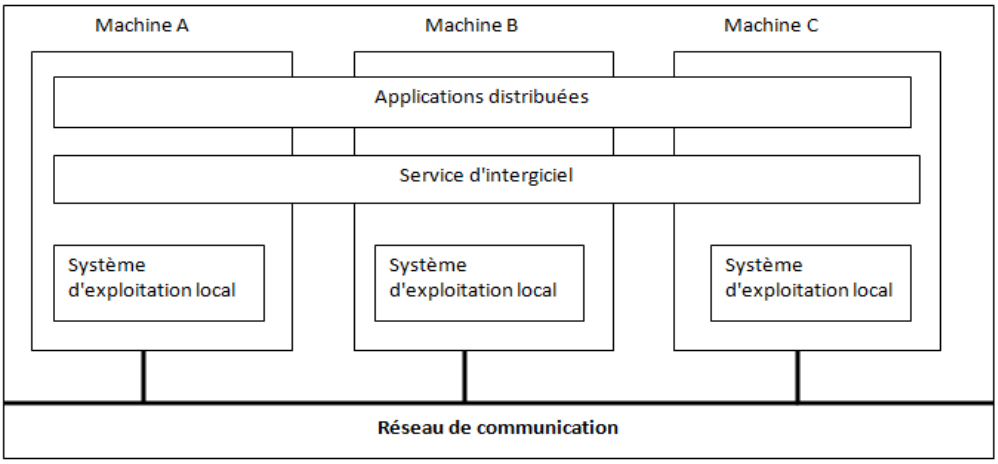

Figure 1-7 Architecture d'un système distribué structuré avec un intergiciel

1.8.6 **Mémoire virtuellement partagée**

Une mémoire virtuellement partagée permet de faire apparaître un ensemble d'espaces mémoires distincts situés sur des machines distinctes d'un réseau comme une même zone mémoire. Lors de la lecture d'une donnée en mémoire par un processus donné, le mécanisme de mémoire virtuellement partagée assure l'accès à cette donnée même si elle se situe dans la mémoire d'une machine distante. Lors d'un accès en écriture, le mécanisme assure que la donnée sera bien modifiée, quel que soit son emplacement. Une mémoire virtuellement partagée implémente, de façon distribuée, un modèle mémoire. Un modèle mémoire détermine les règles de cohérence qui doivent s'appliquer aux variables contenues en mémoire lorsqu'elles sont lues et écrites par différents processus. Un certain nombre de règles sont définies par le modèle mémoire afin de synchroniser le contenu de l'espace privé d'un processus léger avec la mémoire centrale. C'est en particulier le cas lors de l'acquisition ou de la libération d'un verrou. D'autres contraintes sont imposées, par exemple la lecture ou l'écriture de variables déclarées volatiles provoquent également une synchronisation avec la mémoire globale. De même les champs déclarés final reçoivent un traitement particulier. Cela signifie que l'accès à des données partagées entre plusieurs processus légers, en l'absence de synchronisation entre ceux-ci, peut aboutir à des comportements aléatoires [17]. La figure 1-8 illustre une comparaison entre une mémoire partagée et une mémoire distribuée

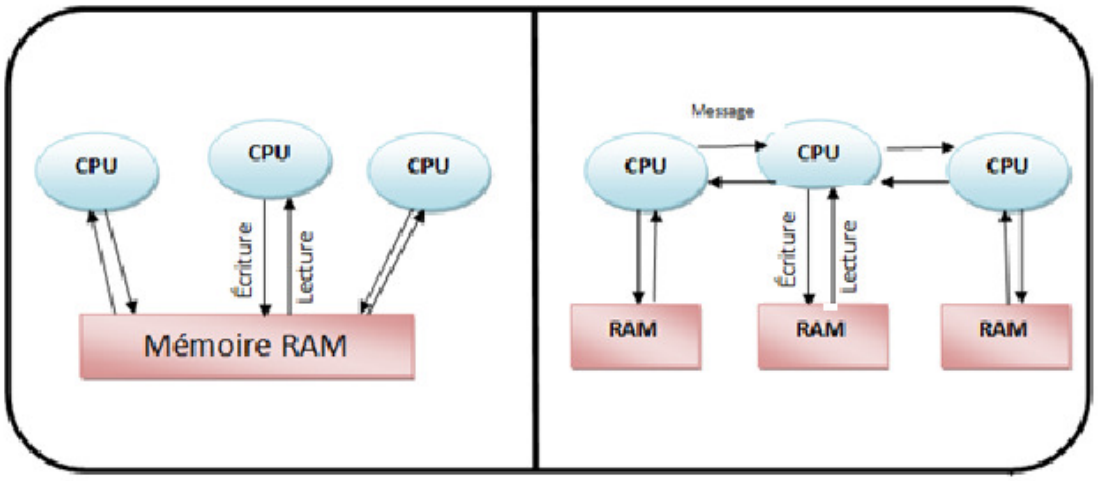

Figure 1-8 Mémoire repartie partagée Vs. Mémoire distribués

1.8.7 **Paradigme des agents mobiles**

L'agent mobile est un processus, incluant du code et des données, pouvant se déplacer entre des machines pour réaliser une tâche. Cette notion de migration détermine la possibilité qui permet à un code de naviguer en fonction de l'application. Elle supporte des opérations déconnectées. La migration des agents et des codes est utilisée dans plusieurs cas de figures tels que : la répartition de charges, la tolérance aux pannes par la survivance aux arrêts de machines et le partage d'information. C'est un mécanisme puissant mais qui reste complexe à mettre en œuvre de par la complexité des outils qu'il utilise suite aux problèmes du coût important de la migration, des problèmes de sécurité, des problèmes d'autonomie, de la gestion de l'état difficile, du problème de l'hétérogénéité hard / soft ainsi qu'en ce qui concerne la vérification des exécutions de code migrants [18].

1.8.7.1 Caractéristiques des agents mobiles

On distingue plusieurs caractéristiques des agents mobiles :

- opérations à gros grain, code défini par l'utilisateur
- partage de charge
- réduction du trafic des données et rapprochement de leur traitement.
- Asynchronisme : le site d'activation peut se déconnecter.
- localisation explicite : migration proactive dans le sens où l'agent décide où aller, et quand (autonomie), migration réactive : réponse à une requête externe (du propriétaire notamment)

- le code à exécuter et les données se déplacent d'une ressource à une autre sur le réseau
- réduction du volume de données échangées sur le réseau

1.8.7.2 Principe de migration de l'agent ou du code

La migration de l'agent et du code est déterminée par les actions suivantes et illustrée figure 1-9 :

• Le mouvement d'une entité en cours d'exécution sur une machine distante

• La connexion seulement pendant la phase de migration

• L'activation

• L'exécution de la mission

• La récupération des résultats

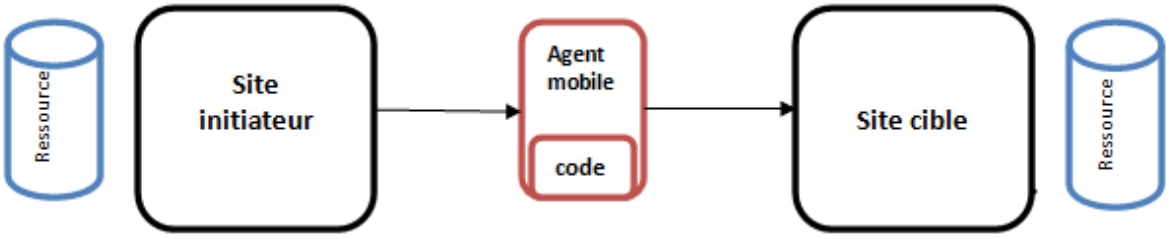

Figure 1-9 La migration d'un agent mobile vers un site distant

1.9 **Les modèles d'interaction**

A partir de la méthode de communication utilisée dans un système distribué on peut construire les protocoles de communication correspondant aux modèles d'interaction.

Nous discutons, dans cette section, des architectures applicatives en passant du modèle classique client-serveur aux architectures émergentes comme les grilles de calcul, les clusters et le pair à pair (p2p). Ces dernières présentent de fait le domaine applicatif privilégié des plateformes industrielles disponibles aujourd'hui. Le modèle d'interaction ou l'architecture logique d'un système repose en grande partie sur sa structure en termes de composants spécifiquement séparés, imagée et conçue dans le but d'assurer un fonctionnement optimal en réponse aux demandes présentes mais aussi celles futures. Le système doit présenter des critères élevés de fiabilité, de maniabilité et d'adaptabilité à un coût réaliste. Ces critères sont favorisés par un bon placement des entités sur le réseau (distribution des données, charge effective) et des relations existantes entre ces entités

(rôles fonctionnels, communications). L'organisation et les caractéristiques des ressources sont des éléments essentiels qui permettent de bien prendre en compte les besoins spécifiques des applications en termes de performances. Ainsi, un classement est fait qui s'appuie plus sur l'organisation et les caractéristiques des ressources [19].

- Le modèle client/serveur **:** Le modèle d'interaction client-serveur est un modèle de communication entre deux entités d'un groupe : Une entité à l'initiative du dialogue (le client) et l'autre qui est dans l'attente d'une requête (le serveur). C'est une extension naturelle de la notion de service dans un environnement centralisé. Le serveur est programmé pour répondre à un ensemble très précis de requêtes, la liste des requêtes autorisées doit être parfaitement définie. Le client émet un message contenant une requête à destination d'un serveur, ce dernier exécute le service associé à la requête émise par le client et retourne au client un message contenant le résultat du service effectué. L'interaction doit être synchrone entre le client et le serveur : le client est bloqué tant que le serveur n'a pas répondu. La figure 1-10 représente l'architecture Client/Serveur.

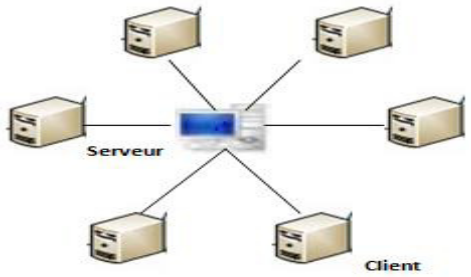

Figure 1-10 L'architecture Client/Serveur

Le problème du modèle Client/serveur réside essentiellement dans son principe de fonctionnement qui est basé sur l'échange explicite de données donc orienté Entrées/Sorties et qui ne peut pas être utilisé comme principe clé d'un système distribué

– La grappe d'ordinateurs (Cluster) : C'est un ensemble d'ordinateurs connectés par un réseau LAN rapide et fiable pour assurer la disponibilité. Les ressources quasi-homogènes (même système d'exploitation, logiciels quasi-similaires) sont sous le contrôle d'un seul nœud appelé nœud maître (cluster-Head). La figure 1-11 représente l'architecture Cluster.

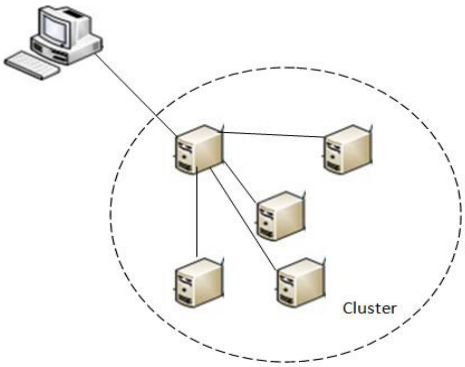

Figure 1-11 L'architecture du cluster

– Les grilles de calcul (Grid Computing) : Dans ce genre de topologies, une collection d'ordinateurs hétérogènes est répartie sur différents sites maintenus par plusieurs organisations. Les sites sont reliés par un réseau de type WAN et chacun d'entre eux contient plusieurs ressources informatiques administrées de manière autonome et uniforme impliquant un partage de ressources et de puissance de calcul dans de très larges organisations multi-institutionnelles. La figure 1-12 représente l'architecture de grille.

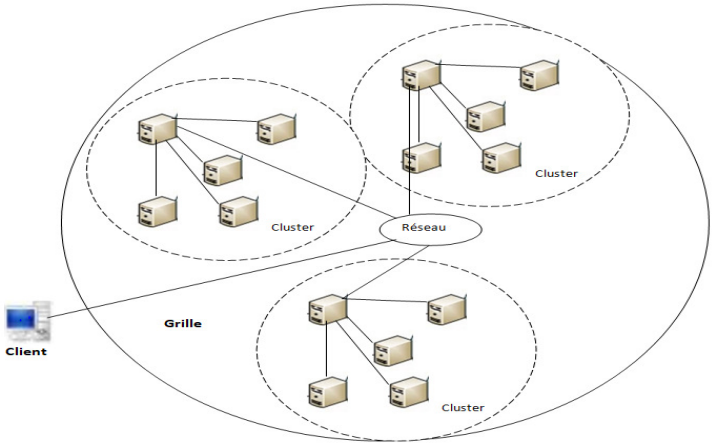

Figure 1-12 L'architecture de grille

– Les systèmes pair-à-pair (peer-to-peer (P2P)) : cette notion détermine un ensemble d'ordinateurs appelés pairs, qui s'accordent à exécuter une tâche particulière. Les ressources peuvent être des ordinateurs ou des assistants personnels interconnectés via Internet, ce qui rend le système beaucoup plus flexible mais aussi moins fiable et stable. La figure 1-13 représente l'architecture P2P.

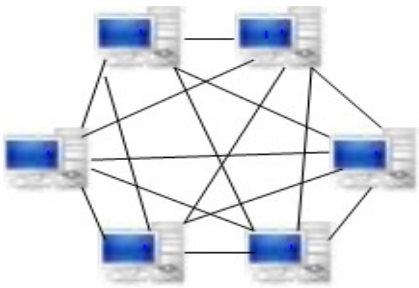

Figure 1-13 L'architecture P2P

–L'architecture du cloud : Du point de vue système, le cloud est un ensemble d'ordinateurs (ressources) stockés sur de vastes grilles de serveurs ou de data centers. Les ressources du Cloud sont mutualisées à travers une virtualisation qui favorise la montée en charge, la haute disponibilité et un plan de reprise à moindre coût. La figure 1-14 présente l'architecture Cloud.

Figure 1-14 L'architecture du cloud

– les réseaux de capteurs (Sensor Networks) : C'est une collection de micro-ressources informatiques (micro-capteur ou système embarqué) reliées le plus souvent par un réseau sans fil en vue de collecter, d'échanger et de transmettre des données (en général environnementales) vers un ou plusieurs points de collecte, d'une manière autonome.

1.10 **Conclusion**

Un système distribué (ou réparti) est composé d'éléments reliés par un système de communication. Les éléments possèdent des fonctions de traitement (processeurs), de stockage (mémoire) et de relation avec le monde extérieur (capteurs, actionneurs, communication). Le système de communication met en œuvre une méthode de communication (par exemple, par envoi de messages). Les différents éléments du système ne fonctionnent pas indépendamment mais collaborent à l'exécution d'une ou plusieurs

tâches communes. En conséquence, une partie au moins de l'état global du système est distribuée entre plusieurs éléments. Le chapitre suivant traite de la gestion du temps et de la coopération entre les différents composants d'un système distribué.

Chapitre 2 : La gestion du temps et de la coopération dans les systèmes distribués

2.1 Introduction

Un système distribué est composé d'un ensemble d'éléments (processeurs, mémoires, capteurs et actionneurs). Ces éléments possèdent des fonctions de traitement, de stockage, de collecte des données et de réaction avec le monde extérieur. Ces éléments sont reliés par un système de communication qui met en œuvre une ou plusieurs méthodes de communication (par exemple, par envoi de messages ou par appels de procédures et méthodes à distance). Cette communication a pour but de partager des données et des ressources entre les éléments du système ou de collaborer pour réaliser une ou plusieurs taches communes. Dans un système réparti les communications et les traitements sont asynchrones. Il n'y a pas de borne supérieure de temps de transition d'un message. Nous avons passé en revue, dans le chapitre précédent, les différentes méthodes de communication et les modèles d'interaction dans un système réparti. La communication inter processus est importante, mais il faut voir aussi les méthodes de coopération et de synchronisation des éléments du système distribué. Nous allons discuter dans ce chapitre de la notion de temps dans les systèmes distribués (les horloges physiques et logiques). Les différents contextes d'applications de ces horloges pour coordonner les éléments composant le système seront abordés. Nous discuterons de l'exclusion mutuelle lors de l'accès à une ressource partagée, de la cohérence du système et de la gestion des possibles inter blocages, des consensus entre éléments et de l'élection d'un leader, de la terminaison d'une application distribuée, des transactions atomiques en distribué, de la tolérance aux fautes et du recouvrement du système ainsi que des problèmes de sécurité.

2.2 La notion du temps dans les systèmes distribués

Les applications distribuées nécessitent la coopération de l'exécution de différents processus pour bien accomplir l'utilisation des fonctions et des services requis. Pour une coopération correcte et efficace, les processus doivent se mettre d'accord sur l'ordre dans

lequel les événements se produisent. Donc, il est nécessaire d'avoir une notion du temps qui soit cohérente pour déterminer les ordres d'exécution. Cependant, cela est très loin d'être trivial en raison de nombreux facteurs parmi eux : les délais inconnus dans la livraison des messages, les changements environnementaux qui peuvent survenir lors de l'exécution du système (panne d'un composant, ajouts ou migrations de processus). La notion du temps dans les systèmes distribués est assez complexe à cause de l'absence d'une horloge globale. En effet, chaque composant du système possède sa propre horloge et peut percevoir le temps de manière différente [20]. L'exécution d'un processus est une suite d'événements (local, émission, réception) appelée histoire (ou trace) du processus. Cette suite est ordonnée par une horloge locale au processus mais comment savoir si un évènement qui se produit sur un site x précède un autre évènement se produisant sur un site y différent ? Pour répondre à cette question il y'a lieu de considérer et de définir un temps global dans le système réparti qui ne dispose généralement pas d'une horloge globale. La définition d'un état global est possible seulement si on est capable de définir un temps global. Bien synchroniser les horloges locales des composants est une condition préalable et importante pour obtenir un système distribué cohérent. Deux solutions sont proposées pour la synchronisation des horloges et la détermination d'un temps global cohérent et identique ou presque à tous les processus du système : soit synchroniser au mieux les horloges physiques locales avec une horloge de référence ou entre elles. Soit créer un temps logique et définir un ordre entre évènements appartenant à plusieurs processus. Pour cela, il est possible, par exemple, de définir une relation globale de précédence, l'une des plus répandues étant la relation de causalité définie par Lamport [3]. Il existe divers mécanismes de codage de ces relations d'ordre (horloges logiques de Lamport, horloges vectorielles et matricielles).

2.3 **Horloge physique**

Chaque ordinateur dispose d'une horloge électronique locale qui garde la trace de l'exécution des événements (temps dans le monde réel). Ces horloges électroniques, appelées aussi physiques, peuvent dériver et présenter des différences dans les temps affichés de sorte que les horloges de différents ordinateurs du même système montrent des temps significativement différents, ce qui conduit à un état incohérent (même si ces différences sont de l'ordre de la nanoseconde ; elles peuvent influer sur l'ordre du temps global). Donc la dérive de l'horloge physique constitue une problématique pour toute

application distribuée dont le comportement et la coopération entre éléments dépendent du temps et nécessitent l'enregistrement de l'heure à laquelle les événements se produisent.

2.3.1 **Technique de synchronisation des horloges physiques**

La synchronisation des horloges physiques du système constitue une condition préalable pour obtenir un système cohérent. Les premiers protocoles étaient principalement basés sur la synchronisation par paire. Plus précisément, avant la synchronisation, une structure de réseau particulière (structure d'arbre ou cluster) doit être constituée puis la synchronisation par paire est exécutée à partir d'un nœud de référence. L'explosion des systèmes distribués (surtout vu le développement de l'internet) ces dernières années a favorisé un ensemble de recherches dans le domaine de la gestion du temps et de la synchronisation par horloge dans les systèmes distribués. Il s'en dégage deux types principaux : la synchronisation interne et la synchronisation externe. Ainsi, plusieurs algorithmes ont été proposés pour chacun de ces types [21].

2.3.1.1 La synchronisation interne

La synchronisation interne correspond à un groupe d'horloges qui ont, entre elles, une dérive bornée de sorte qu'à tout instant, la dérive entre deux horloges quelconques du groupe ne dépasse jamais la borne D en augmentation et en diminution, une mesure locale du temps sur un site donnera un temps identique à une mesure d'un autre site avec une différence d'au plus D comme c'est indiqué dans l'équation 1. Les dérives sont bornées entre les horloges mais pas nécessairement avec le temps réel.

$$V\ (i,j,t) \qquad t = |Ci(t) - Cj(t)| < D \tag{1}$$

Où

i, j : deux processus équipés des horloges

D: borne max de la dérive

Ci: horloge physique du processus i

t: temps réel

Ci(t): valeur de l'horloge Ci à l' instant t

L'algorithme de Berkeley [22] est un algorithme de synchronisation interne pour un système asynchrone. Son principe est le suivant : un des processus du système est élu maître et c'est lui qui gère le temps. Ce dernier désigne un serveur de temps qui fournit une

source de temps précise et stable aux clients sur le réseau. Pour ce faire, ce maitre doit synchroniser son horloge physique en accord avec une horloge réelle comme UTC/GMT. Les autres processus lui envoient leur valeur d'horloge en prenant en compte les temps d'aller-retour, à partir de toutes les horloges, le maitre détermine le temps moyen global puis il envoie à chaque processus la variation de leur temps local par rapport au temps global. L'algorithme de Berkeley permet d'éliminer les horloges défaillantes car le maître a une vision globale et ainsi si le maître se plante, un autre est élu à sa place. La figure 2-1 montre le principe de l'algorithme de Berkeley et la mise à jour périodique des horloges.

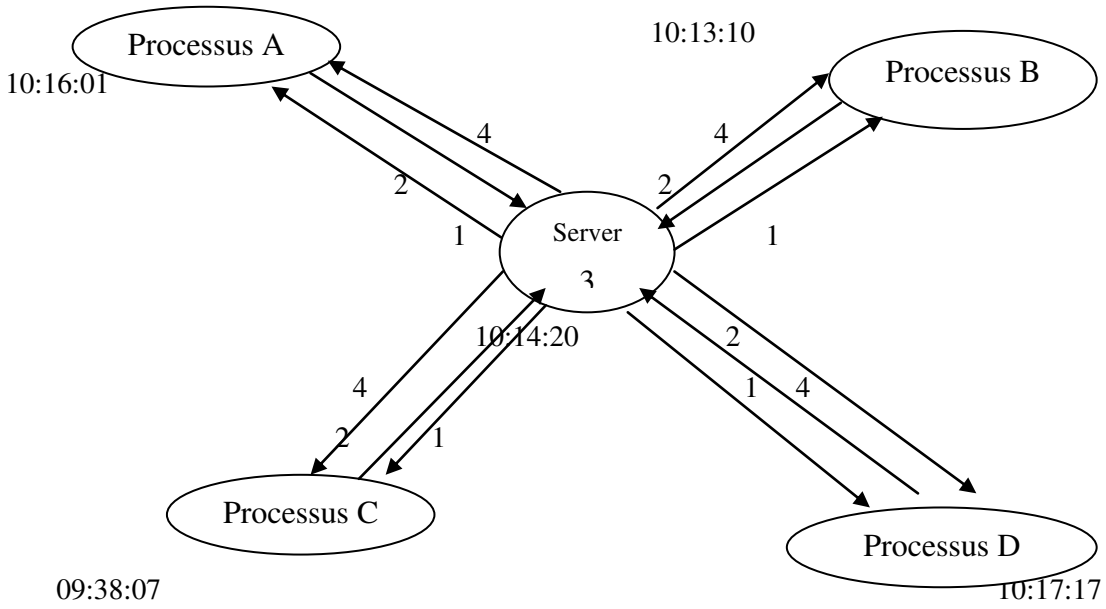

Figure 2-1 La mise à jour périodique par l'algorithme de Berkeley

La séquence des messages pour le fonctionnement de l'algorithme de Berkeley se déroule comme suit :

1. Le serveur de temps qui est élu maître interroge les autres ordinateurs pour leurs valeurs de temps actuels ;
2. Chacun des processus qui a reçu la demande du serveur, lui répond par la valeur de son horloge
3. Le maitre calcule la moyenne des horloges envoyées (moyenne = 10:15:12) en ignorant les valeurs aberrantes (09:38:07), puis calcule le temps d'aller-retour d'un message pour chacun des processus et le temps de décalage pour chacune des horloges des processus.
4. Il envoie la mise à jour des décalages individuels à chacun des processus.

2.3.1.2 La synchronisation externe

La synchronisation externe est basée sur une source externe au système pour fournir un temps de référence. Les horloges se resynchronisent régulièrement à partir de cette source en prenant en compte le temps de propagation des messages entre le processus courant et celui gérant le temps de référence, et aussi le temps de traitement de la requête du processus gérant le temps de référence s'il est connu. La même contrainte est imposée comme pour la synchronisation interne mais par rapport à une source de référence comme l'indique l'équation 2.

$$V\ (i,t) \qquad t = |\Theta i(t) - Cj(t)| < D \tag{2}$$

Où

i: processus équipés des horloges

Θ: le temps de référence donné par la source externe

D: la borne max de la dérive de temps

Ci: l'horloge physique du processus i

t: un instant réel

Ci(t): la valeur de l'horloge Ci à l' instant t

Il existe plusieurs sources dans le monde réel qui sont considérées comme des référentiels temporels tel que UTC/GMT, comme il existe des horloges atomiques dans chaque pays dont le décalage est très faible qui est estimé à deux secondes chaque année.

L'algorithme de Christian est un algorithme de synchronisation externe pour un système asynchrone. Il utilise un serveur de temps synchronisé sur le temps universel UTC. Les horloges des autres ordinateurs du réseau sont définies en interrogeant le serveur. La caractéristique principale de l'algorithme de Christian [23] est la prise en compte de la latence du réseau. La figure 2.2 montre le principe de l'estimation des délais du réseau utilisé dans l'algorithme de Christian. Le concept est simple ; le temps d'aller-retour pour qu'un message soit envoyé à un autre ordinateur et une réponse à renvoyer est mesuré par d. Cela correspond à une latence de d/2 dans chaque direction. Afin de synchroniser son horloge, le client envoie une requête au serveur à l'instant T_0 pour demander le temps. Le message de réponse est supposé arriver à l'instant T_1. On a, ainsi, le temps d'envoi est Tserveur + $(T_1-T_0)/2$, ensuite l'horloge du client est réglée avec précision (voir figure 2-2).

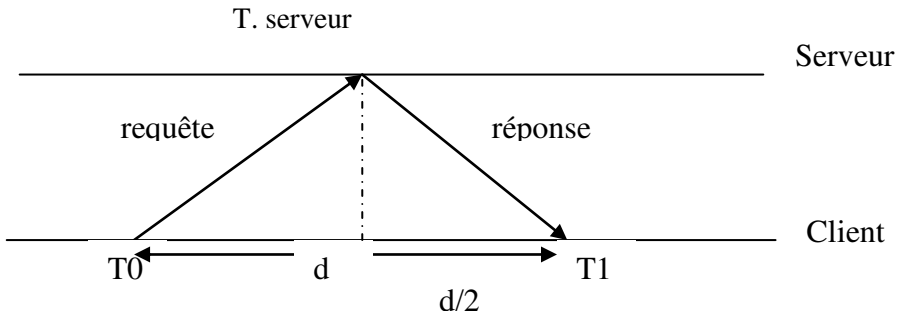

Figure 2-2 Envoi du temps serveur en tenant compte la valeur de décalage

2.4 **Horloge logique**

L'horloge logique est un temporisateur associé à chaque processeur permettant de générer un compteur de temps local (des impulsions d'horloge). Cette horloge associe à chaque événement un simple numéro appelé date ou estampille qui représente le temps d'occurrence de cet événement. A partir de cette date on peut préciser l'ordonnancement de l'exécution des processus et de leur communication. Cet ordonnancement logique des événements est créé en fonction des événements locaux des processus et des messages envoyés et reçus. Trois approches principales d'estampilles et horloges sont définies : horloge et estampille scalaire (horloge de Lamport), horloge et estampilles vectorielles (horloge de Mattern), horloge et estampille matricielles.

2.4.1 **Chronogramme**

C'est un graphisme qui donne une vision globale des interactions des processus du système et qui décrit l'ordonnancement temporel des événements des processus et des échanges de messages dans un système distribué. Dans un chronogramme ; chaque processus est représenté par une ligne, ces processus sont reliés entre eux par des canaux. Chaque ligne peut contenir trois types d'événements : événement interne dans l'évolution du processus, événement d'émission d'un message vers un autre processus ou événement de réception d'un message venant d'autres processus. Les messages échangés doivent respecter la topologie de liaison des processus vis-à-vis des canaux. Chaque événement possède un numéro sur la ligne à laquelle il appartient, ce numéro est représenté par la notation e_{xy} avec x le numéro du processus et y le numéro de l'événement pour le processus. La figure 2-3 présente le chronogramme d'un système distribué composé de 3 processus où chacun exécute des événements locaux d'émission et de réception.

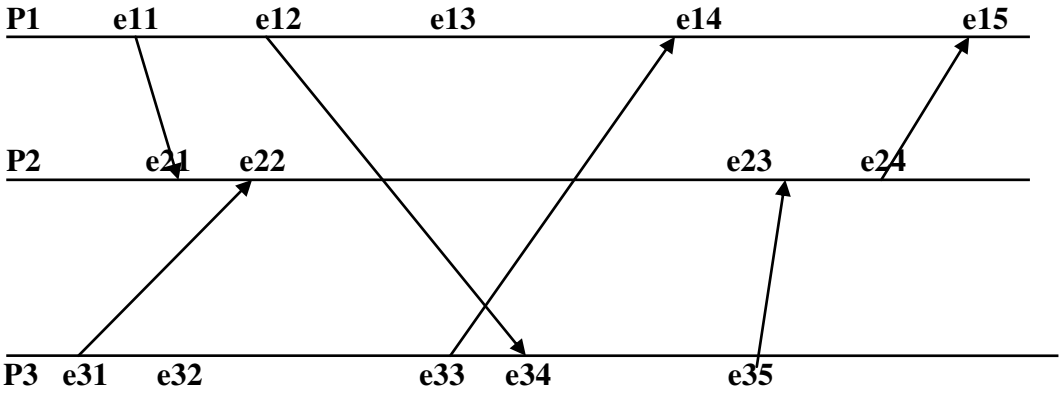

Figure 2-3 Chronogramme

2.4.2 **Relation de précédence causale**

Il est possible de définir un ordre entre évènements appartenant à plusieurs processus par la définition d'une relation globale de précédence. L'une des plus répandues étant la relation de causalité présentée comme suit : Notons e →e' (on lit e précède e') la relation de causalité entre deux événements e et e'. On peut dire que

- e→ e' si e et e' sont deux événements d'un même processus, et que e se produit avant e'.
- e → e' si e est une émission de message et e' est une réception.
- e → e' s'il existe e" tel que e → e" et e" → e'.

La relation de causalité est représentée en général par la formule logique suivante

$$e \rightarrow e' \Longrightarrow H(e) < H(e') \quad H : \text{Horloge logique}$$

2.4.3 **Relation de parallélisme ||**

On note e|| e' (qu'on lit « e concurrent avec e' ») lorsque e -/-> e' et e' -/-> e (e ne précède pas e' et e' ne précède pas e) la relation de parallélisme logique ne signifie pas que deux événements se déroulent simultanément mais qu'ils peuvent se dérouler dans n'importe quel ordre. Cette relation peut être exprimée par la formule logique suivante:

$$e \parallel e' \Longleftrightarrow \neg((e \rightarrow e') \vee (e \rightarrow e'))$$

Les dépendances causales définissent des ordres partiels pour des ensembles d'événements à cause du parallélisme logique de certains événements qui rendent la définition d'un ordre total impossible. Donc l'utilisation d'horloge logique a pour but de créer un ordre total global sur tous les événements de tous les processus

2.4.4 **Horloges et estampilles scalaires (Horloge de Lamport)**

L'horloge de Lamport [24] est une horloge logique respectant la relation de causalité. Localement chaque processus Pi est équipé d'une horloge locale logique Hi. Une estampille (date) est associée à chaque événement et se présente comme un couple (np,ne) où np est le numéro de processus et ne est un compteur dont la valeur est un entier qui représente le numéro d'événement. Ce compteur est initialisé à 0 au lancement du système. De telle sorte qu'il n'y ait pas deux événements locaux ayant le même numéro.

2.4.4.1 Algorithme

- A chaque occurrence d'un événement local sur Pi, on incrémente l'horloge locale (Hi = Hi +1)
- Pour chaque événement correspondant à l'émission d'un message par Pi, on incrémente Hi de 1 puis on envoie le message avec (i,Hi) comme estampille
- pour chaque événement correspondant à la réception d'un message m d'estampille (np,ne) qui se produit sur Pi, on aura Hi = max (Hi , ne) + 1 et on marque l'événement de réception avec Hi. La réception d'un message permet de synchroniser l'horloge du récepteur avec celle de l'émetteur du message (qui est transportée par le message).

Pour l'ordonnancement global des événements par l'horloge de Lamport, la valeur de Hi permet de savoir si un événement a lieu avant un autre ou pas, qu'il s'agisse d'événements du même processus ou de processus différents. L'ordre des événements n'est pas un ordre strict car plusieurs événements peuvent porter la même valeur. Pour rendre cet ordre strict on adjoint à la valeur de l'horloge logique d'un site l'identification du site. On note e<<e' c'est à dire que e s'est déroulé avant e'. Soit e événement de Pi et e' événement de Pj:

$$e << e' \iff (Hi(e) < Hj(e')) \vee (Hi(e) = Hj(e') \text{ avec } i<j)$$

Hi donne l'ordre des événements d'un même processus localement si (i=j), et sur différents processus (i différent de j). Sauf en cas d'égalité des valeurs d'horloge des événements, le numéro de processus est utilisé pour les ordonner. La figure 2-4 montre l'application d'horloges de Lamport pour un système à 3 processus P1, P2 et P3

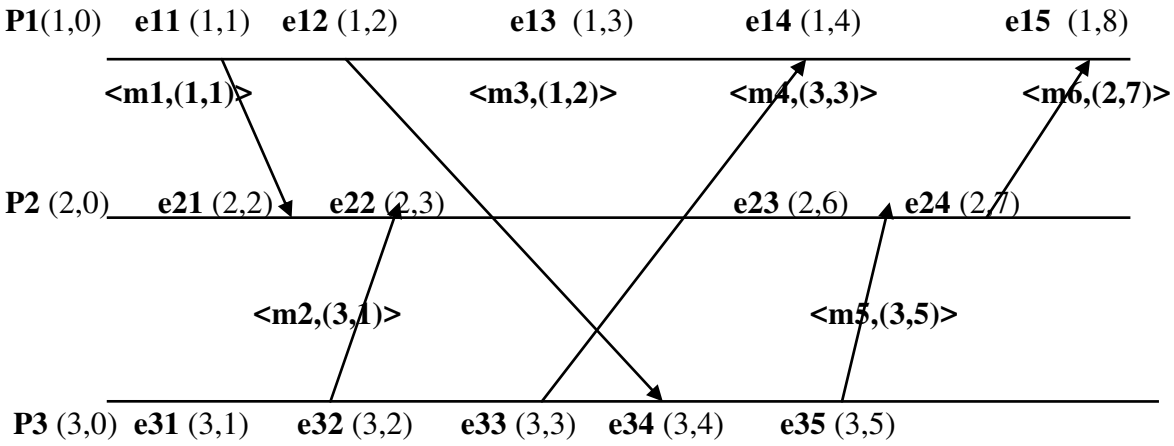

Figure 2-4 Chronogramme illustrant les horloges de Lamport

2.4.4.2 Limites de l'horloge de Lamport

Les limites des horloges de Lamport peuvent se résumer dans les points suivants :

- Ordonnancement artificiel des événements concurrents. L'ordonnancement global obtenu est arbitraire et n'est pas forcément celui obtenu en pratique.
- Respect des dépendances causales, mais avec e et e' tel que H(e) < H(e'). On ne peut rien dire sur la dépendance entre e et e'.
- Ne permettent pas de corriger la défaillance vis-à-vis de l'ordre FIFO (First In First Out)

2.4.5 **Horloges et estampilles vectorielles (Horloge de Mattern)**

L'horloge de Mattern [25] assure la réciproque de la dépendance causale : Hi(e) < Hj(e') ⇨ e→ e'. Elle permet de dater les événements d'un site et est mise à jour lors de l'occurrence des événements. Elle permet également de savoir si deux événements sont parallèles (non dépendants causalement). Son principe est comme suit : chaque site gère une horloge vectorielle constituée de n entiers (le système comporte n sites). Les estampilles sont utilisées sur les messages, elles sont des vecteurs de taille égale au nombre de processus. Localement, chaque processus Pi a un vecteur Vi et pour chaque processus Pi, chaque case Vi[j] du vecteur contiendra des valeurs de l'horloge du processus Pj.

2.4.5.1 Algorithme

- Initialisation : pour chaque processus Pi, Vi= (0,.....,0)
- Pour un processus Pi, à chacun de ses événements (local, émission, réception) : incrémentation du compteur local d'événement Vi[i]= Vi[i] +1

- Si émission d'un message, alors Vi est envoyée avec le message : Les messages envoyés par un site sont estampillés en utilisant la valeur courante de l'horloge vectorielle du site émetteur,
- Pour un processus Pi, à la réception d'un message m contenant un vecteur Vm, on met à jour les cases j≠i de son vecteur local Vi; $\forall j: Vi[j] = \max(Vm[j], Vi[j])$. La réception d'un message permet au site récepteur de synchroniser son horloge vectorielle avec celle du site émetteur du message.

  La figure 2-5 montre l'application des horloges de Mattern sur 3 processus P1, P2 et P3

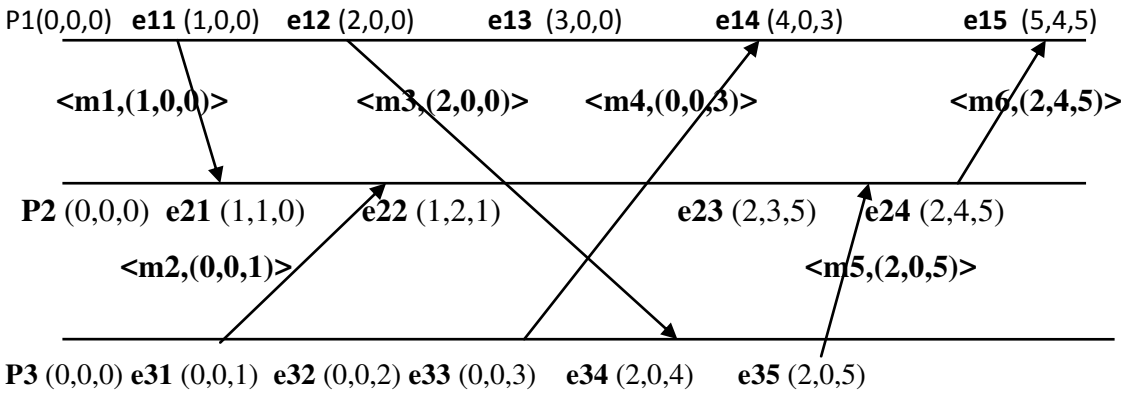

Figure 2-5 Horloge de Mattern

L'horloge de Mattern respecte les dépendances et les indépendances causales des événements en assurant les propriétés suivantes, avec e et e' deux événements et V(e) et V(e') leurs datations :

$$V(e) < V(e') \Longrightarrow e \rightarrow e'$$

- à partir de l'équation ci-dessus ; on peut dire que si deux dates sont ordonnées, on a forcément une dépendance causale entre les événements datés.

$$V(e) \parallel V(e') \Longrightarrow e \| e'$$

- on peut dire aussi que s'il n'y a aucun ordre entre les deux dates, les deux événements sont indépendants causalement.

2.4.5.2 Limites de l'horloge de Mattern

Les points suivants déterminent les limites des horloges de Mattern :

- Ne permettent pas de définir un ordre global total
- Ne permettent pas de garantir une délivrance causale des messages.

- Ne permettent pas de corriger la défaillance vis-à-vis de la délivrance causale des messages.
- En cas de nombreux processus, la taille du vecteur peut-être importante et donc des données à transmettre relativement importante.

2.4.6 **Définition des états globaux dans un système distribué**

L'état global du système représente les données et les valeurs des variables de tous les processus du système à un instant donné. La recherche des états globaux pour un système distribué a pour but de trouver des états cohérents à partir desquels on peut reprendre un calcul distribué en cas de plantage du système. Ainsi que de faciliter le debugging et la mise au point d'applications distribuées. Ces états globaux sont définis à partir des coupures. Une coupure est une observation à un instant donné de l'état du système, il existe des événements qui appartenant au passé et d'autres au futur par rapport à l'instant de la coupure. Une coupure C détermine, ainsi, un sous-ensemble fini de E tel que : soient a et b deux événements du même processus :

$$a \in C \text{ et } b \rightarrow a \Longrightarrow b \in C$$

- à partir de cette équation, on peut dire que si un événement d'un processus appartient à la coupure, alors tous les événements locaux le précédant y appartiennent également.

- une coupure est dite cohérente si elle respecte les dépendances causales des événements du système de telle sorte qu'il n'existe aucun événement qui vient du futur.

2.4.7 **Algorithme Chandy & Lamport pour détermination des états cohérents**

L'algorithme de Chandy & Lamport [26] est un algorithme permettant aux processus distribués d'enregistrer un état global cohérent. Son principe général est basé sur la diffusion d'un marqueur par le processus voulant déterminer un état global à tous les autres processus pour enregistrer leurs états. On suppose que les canaux de communication sont unidirectionnels, FIFO et fiables. Le processus du système doit être fortement connexe.

2.4.7.1 Algorithme

- Un processus Pk est initiateur du lancement de l'algorithme, il enregistre son état local et envoie un message marqueur à tous les autres processus.
- Un processus Pj, à la réception du marqueur sur son canal <k,j>, exécute ces trois étapes en séquence atomique
    - Il enregistre son état local (données et variables)

- Il positionne l'état de chacun de ses canaux entrants <i,j> à vide
- Il Envoie un marqueur sur tous ses canaux sortants : à tous ses voisins

- Pour un processus Pj, à la réception du marqueur venant de Pi, sur le canal entrant<i,j>, il enregistre l'état du canal <i,j> qui concerne tous les messages reçus sur <i,j> depuis la réception du premier marqueur venant de Pk.
- Pour un processus Pj, l'algorithme se termine quand il a reçu un marqueur sur chacun de ses canaux.
- Pour constituer l'état global, on collecte l'ensemble des états enregistrés par les processus une fois qu'ils sont l'ont enregistrés. La figure 2-6 représente l'application de l'algorithme Chandy&Lamport pour déterminer un état global (coupure)

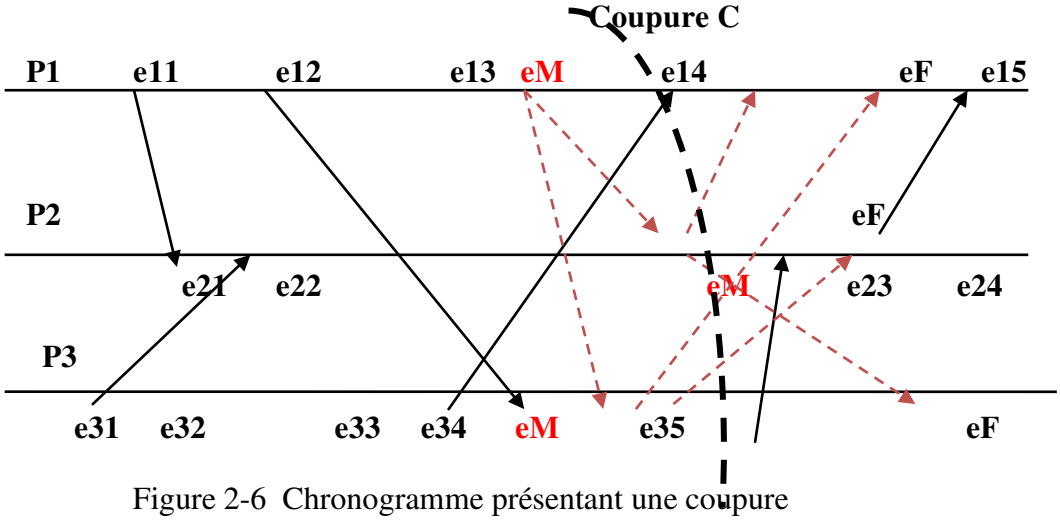

Figure 2-6 Chronogramme présentant une coupure

eM: événement d'enregistrement d'état et diffusion d'un marqueur
eF événement où l'état local complet est enregistré

2.5 Accord et coordination

Une grande famille de problèmes en algorithmique distribuée consiste dans la coordination et l'accord entre les processus du système, cette famille inclut les catégories des problèmes suivantes :

- Accord sur l'accès à une ressource partagée : exclusion mutuelle
- Accord sur un processus jouant un rôle particulier : élection d'un coordinateur
- Accord sur une valeur commune : consensus
- Accord sur une action à effectuer par tous ou personne : transaction distribuée
- Accord sur l'ordre d'envoi de messages à tous : diffusion atomique

2.6 **Exclusion mutuelle**

Plusieurs processus du système peuvent accéder à des ressources partagées. Les parties de programme qui permettent l'accès à ces ressources ne devant être exécutées que par un seul processus à la fois. Ces parties sont appelées sections critiques. Le maintien d'un état cohérent du système nécessite la gestion des accès aux ressources partagées et la protection des sections critiques. Cette fonction est très souvent effectuée en utilisant les techniques d'exclusion mutuelle. Cette dernière est l'un des paradigmes fondamentaux des systèmes distribués pour garantir des accès cohérents aux ressources partagées. Ce paradigme est considéré comme une application du temps logique. Dans les systèmes distribués, le mécanisme de l'exclusion mutuelle ne peut pas reposer sur des primitives ou des variables globales comme dans un système centralisé, il est nécessaire d'obtenir un consensus entre les différents processus par la mise en œuvre d'un algorithme distribué. Un processus peut se trouver dans trois états par rapport à l'accès à la ressource :

1. Demandeur : demande l'utilisation de la ressource partagée ; c'est à dire demande de rentrer dans la section critique.
2. Dedans : le processus est dans la section critique et utilise la ressource partagée.
3. Dehors : le processus est en dehors de la section critique.

La figure 2-7 présente le diagramme d'états de l'accès en exclusion mutuelle

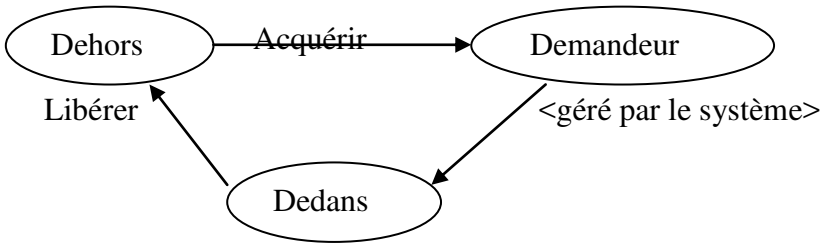

Figure 2-7 Diagramme d'états de l'accès en exclusion mutuelle

2.6.1 **Propriétés des algorithmes d'exclusion mutuelle**

Un algorithme d'exclusion mutuelle doit respecter et satisfaire les propriétés fondamentales suivantes [27] :

- Propriété de sureté (safety property) : à un instant donné t, un seul processus au plus peut exécuter une portion du code manipulant une ressource partagée appelée section critique c'est la propriété essentielle de l'exclusion mutuelle.

- Propriété de vivacité (liveness property) : toute demande d'accès à la section critique sera satisfaite dans un temps fini, et chaque site obtiendrait l'opportunité d'exécuter la section critique au bout d'un temps fini ; c'est à dire l'absence de l'inter blocage et de la privation (la famine).
- Propriété d'équité (fairness property) : tout processus aura la chance d'exécuter la section critique, c'est à dire que les demandes d'exécution de la section critique sont traitées dans l'ordre d'arrivé dans le système (temps déterminé par l'horloge logique).

2.6.2 **Les métriques de performance des algorithmes d'exclusion mutuelle**

Les performances des algorithmes d'exclusion mutuelle sont généralement mesurées selon les quatre métriques suivantes :

- Complexité en message (CM) : relative au nombre de messages générés par les processus pour pouvoir entrer en section critique.
- Délai de synchronisation (SD) : Intervalle de temps entre le moment ou la section critique devient libre (après que le processus qui détient la section critique la quitte) et avant le prochain accès (avant que le prochain processus y accède)
- Temps de réponse (TR) : Intervalle de temps entre l'émission du message de requête et la sortie de la section critique.
- Débit système (DS) : taux d'exécution (par le système) des requêtes d'entrées en section critique.

  DS= 1/(SD+E) avec E = temps moyen d'exécution de la section critique

2.6.3 **Algorithmes d'exclusion mutuelle**

Plusieurs algorithmes d'exclusion mutuelle existent dans la littérature, on peut distinguer principalement trois grandes familles : contrôle par serveur, algorithmes à base de permission et les algorithmes à base de jeton.

2.6.3.1 Contrôle par un serveur

Cette méthode utilise un serveur qui centralise et gère les demandes d'accès à la ressource partagée

Algorithme

- un processus voulant accéder à la ressource (quand il passe dans l'état demandeur) envoie une requête au serveur
- Quand le serveur lui envoie l'autorisation, il accède à la ressource (passe dans l'état dedans) : et il informe le serveur quand il relâche la ressource (passe à l'état dehors)
- Le serveur reçoit les demandes d'accès et envoie les autorisations aux processus demandeurs.

Avantages

- Très simple à mettre en œuvre
- Simplicité de gestion de la concurrence d'accès à la ressource

Inconvénients

- Nécessite un élément particulier pour gérer l'accès
- Potentiel point faible : problème de goulot d'étranglement

Il est nécessaire de supprimer le serveur centralisateur pour résoudre les problèmes précédents, via le consensus entre les différents processus et l'échange de messages.

2.6.3.2 Méthode avec permission

Dans les algorithmes d'exclusion mutuelle à base de permission, avant d'accéder à la section critique, un processus demande à tous les autres processus du système une permission d'accès et il attend que chacun ait répondu. Si le processus demandeur reçoit l'autorisation des autres processus, il va rentrer à la section et utiliser la ressource. Deux modes existent : Permission individuelle où un processus peut donner sa permission a plusieurs autres à la fois et permission par arbitre où un processus ne donne sa permission qu'à un seul processus à la fois.

a- Algorithme de Lamport(1978)

L'algorithme de Lamport d'exclusion mutuelle à permission individuelle est basé sur l'horloge logique de Lamport et nécessite un ordre total de tous les événements dans le système. Trois types de messages sont utilisés : REQUEST, REPLY et RELEASE. Lorsqu'un processus reçoit l'un des messages précédents, il met à jour son horloge en utilisant les valeurs des estampilles véhiculées dans ces messages. L'algorithme se déroule selon les étapes a, b et c.

a- Demande d'entrer en section critique

- Quand un processus Pi désire entrer en section critique, il diffuse une requête REQUEST (TPi,i) à tous les autres processus Pj (y compris lui-même), puis enfile la requête dans sa propre file Request-queue i.
- Quand un processus Pj reçoit une requête REQUEST (TPi,i) de Pi, il enfile cette requête dans la file Request-queue j à la bonne place (insertion par ordre croissant des estampilles) puis renvoie à Pi une réponse de type REPLY.

b- Exécution de la section critique

Le processus Pi entre en section critique si les deux conditions suivantes sont satisfaites :

- C1: Pi a reçu un message de réponse ou acquittement (Ack) de tous les autres processus.
- C2: La requête de Pi est en tête de la file

c- Libération de la section critique

- A la sortie de la section critique, Pi efface sa requête en tête de sa file et diffuse un message de libération estampillé RELEASE à tous les autres processus
- Lorsqu'un processus Pj reçoit un message RELEASE de Pi, il efface de sa file la requête de Pi.

- L'algorithme de Lamport satisfait la propriété de sureté, il assure bien l'exclusion mutuelle et il est équitable si les requêtes d'entrée en SC sont traitées dans l'ordre de leurs estampilles. L'algorithme de Lamport n'est pas tolérant aux fautes : si le processus p se bloque dans la SC alors il ne peut pas envoyer un RELEASE et donc les autres ne pourraient jamais entrer à la SC. La propriété de vivacité n'est pas assurée.

b- Algorithmes de Ricard Agrawala

C'est un algorithme d'exclusion mutuelle à permission individuelle [28] qui est une amélioration de l'algorithme de Lamport ; il se déroule selon les étapes a, b et c suivantes :

a- Demande d'entrer en section critique

- Quand un processus Pi désire entrer en section critique (SC), il diffuse une requête REQUEST (TPi,i) à tous les autres processus Pj (y compris lui-même), puis enfile la requête dans sa propre file Request-queue i à la bonne place.
- Quand un processus Pj reçoit un message REQUEST de Pi, il envoie un REPLY a Pi si Pj n'est pas demandeur pour entrer en SC, ni exécuteur de la SC, ou bien si Pj désire

entrer en SC mais son estampille est supérieure à celle de Pi. Autrement, la réponse sera différée et Pi met à jour RDj[i] =1. RDj est un tableau contenant les identités des processus pour lesquels la réponse sera différée.

b- Exécution de la section critique

Le processus Pi entre en section critique après avoir reçu un message REPLY de chaque processus à qui il a envoyé un REQUEST.

c- Libération de la section critique

Lorsque Pi quitte la section critique, il envoie tous les messages REPLY différés c'est à dire qu'il envoie un message REPLY à tout processus Pk tel que RDi[k]=1 et met à jour RDi[k]=0.

- L'algorithme de Ricard Agrawala respecte les propriétés de sureté et de vivacité mais son inconvénient principal est le nombre relativement important de messages échangés**.** Une amélioration de l'algorithme Ricart agrawala est proposée par Chandy& Misra comme suit : les processus ne voulant pas accéder à la ressource et qui ont déjà donné leur permission ne reçoivent pas de demande de permission.

Inconvénient de l'approche à base de permission individuelle

La panne d'un des processus empêche les autres de rentrer en section critique. Pour éviter cela des mécanismes à base de quorum peuvent être mise en place : l'autorisation d'un sous ensemble des processus peut suffire pour l'entrée en section critique

Les algorithmes d'exclusion mutuelle fondés sur un quorum suivent l'approche suivante :

- Un processus Pi n'a pas besoin de demander une permission à tous les processus pour pouvoir exécuter une SC, mais seulement à un sous ensemble de processus. Ce qui réduit substantiellement le nombre de messages requis pour l'accès à une section critique.

c- Algorithme de Maekawa

L'algorithme de Maekawa [29] est un algorithme d'exclusion mutuelle à base de permission par arbitre c'est à dire qu'un processus ne donne qu'une permission à la fois puis il redonnera sa permission à un autre processus quand le processus à qui il avait donné précédemment la permission lui a indiqué qu'il a fini d'accéder à la ressource.

- Toute paire de sous-ensembles de processus possède un médiateur pour résoudre les conflits

L'algorithme de Maekawa se déroule en sept étapes comme suit :

1. pour entrer en section critique, Pi envoie une demande REQUEST (Ti,i) à tous les processus Pj ∈ Ri, i≠j

2. Lorsque Pj reçoit un message REQUEST (Ti,i):

a- Si Pj n'a pas encore envoyé un message REPLY à tout processus depuis le dernier message RELEASE qu'il a reçu alors Pj envoie REPLY (Tj,j) à Pi.

b- Sinon, il existe un processus Pk tel que Pj a déjà envoyé un message REPLY (Tj, j) à Pk, et

- Si (Ti,i) < (Tk,k), Pj envoie un message INQUIRE(j) à Pk et place la demande REQUEST dans sa file
- Sinon, Pj envoie un message FAILED(j) à Pi

3. Lorsque Pk reçoit un message INQUIRE(j), il envoie un message YIELD(k) si une des conditions suivantes est satisfaite

a- Pk a reçu un message FAILED d'un processus de Rk

b- Pk a envoyé un message YIELD(m) à un processus Pm ∈ Rm et n'a pas reçu de message de réponse REPLY. Sinon Pk ne fait rien.

4. Lorsque Pj reçoit un message YIELD(k), Pj place REQUEST (Tk,k) dans sa file et envoie un REPLY(Tj,j) au processus pour lequel la demande est en tête de file.

5. Lorsque Pj reçoit un message RELEASE (Ti,i), il envoie un message REPLY(Tj,j) au prochain processus dans la file et efface l'entrée dans la file. Si la file est vide, Pj met à jour son état de sorte qu'aucun message REPLY n'ait été envoyé à aucun processus depuis le dernier message RELEASE reçu par Pj.

6. Pi entre en SC lorsqu'il a reçu un message REPLY de tous les processus de Ri

7. Lorsque Pi quitte la section critique, il envoie un message RELEASE (Ti,i) à tous les processus de Ri.

2.6.3.3 Méthode du jeton

Un jeton unique circule entre tous les processus et donne l'accès à la ressource. Le processus qui possède le jeton est le seul qui peut accéder à section critique. Il existe deux versions, soit un jeton qui circule dans un anneau en permanence, soit le jeton est affecté à la demande des processus.

a- Algorithme de Le Lann

- Un jeton unique circule en permanence entre les processus via une topologie en anneau

- Quand un processus reçoit le jeton : s'il est dans l'état demandeur, il passe dans l'état dedans et accède à la ressource. S'il est dans l'état dehors, il passe le jeton à son voisin
- Quand le processus quitte l'état dedans il passe le jeton à son voisin

L'algorithme de Le Lann respecte la propriété de sureté via le jeton unique qui autorise l'accès à la ressource. La propriété de vivacité est assurée si un processus lâche le jeton (la ressource) en un temps fini et que tous les processus appartiennent à l'anneau.

Avantages

- Très simple à mettre en œuvre
- Intéressant si de nombreux processus sont demandeurs de la ressource : le jeton arrivera rapidement à un processus demandeur.
- équitable en termes de nombre d'accès et de temps d'attente (aucun processus n'est privilégié).

Inconvénients

- Nécessite des échanges de messages (pour faire circuler le jeton) même si aucun site ne veut accéder à la ressource
- le temps d'accès à la ressource peut être potentiellement relativement long : si le processus i+1 a le jeton et que le processus i veut accéder à la ressource et est le seul à vouloir y accéder, il faut quand même attendre que le jeton fasse tout le tour de l'anneau.
- Dans le cas des algorithmes à base de jeton, une panne peut provoquer la perte du jeton et empêcher un processus d'entrer en section critique par la suite. Il est alors possible de mettre en œuvre un mécanisme de détection de perte de jeton et d'émission d'un nouveau jeton mais là encore d'autres problèmes doivent être réglés, notamment pour garantir l'unicité du jeton.

- Une variante de la méthode du jeton : au lieu d'attendre le jeton, un processus diffuse à tous les autres le fait qu'il veut obtenir le jeton. Le processus qui a le jeton sait alors à qui il peut l'envoyer et donc éviter les attentes et les circulations inutiles du jeton.

b- Algorithme de Ricart& Agrawala

Cet algorithme [30] fonctionne selon le schéma suivant :

- Soit N processus avec un canal bi-directionnel entre chaque processus : les canaux sont fiables mais pas forcément FIFO
- Localement, un processus Pi possède un tableau nbreq, de taille N

- Pour Pi, nbreq[j] est le nombre de requêtes d'accès que le processus Pj a fait et que Pi connait.

- Le jeton est un tableau de taille N
  - jeton[i] est le nombre de fois ou le processus Pi a accédé à la ressource
  - La case i de jeton n'est modifiée que par Pi quand celui-ci accède à la ressource
- Initialisation
  - Pour tous les sites Pi : $\forall j \in [1..N]$: nbreq[j]=0
  - Jeton : $\forall j \in [1..N]$: jeton[j]=0
  - un site donné possède le jeton au départ
- Quand un site veut accéder à la ressource et n'a pas le jeton, il envoie un message de requête à tous les processus
- Quand un processus Pj reçoit un message de requête venant de Pi, il mémorise que Pi a demandé la ressource et modifie son nbreq localement : nbreq [i] = nbreq[i]+1. Si Pj possède le jeton et est dans l'état dehors, il envoie le jeton à Pi.
- Quand un processus Pi récupère le jeton, il accède à la ressource et passe à l'état dedans.
- Quand Pi libère la ressource, il passe à l'état dehors et met à jour le jeton : jeton[i] = jeton[i]+1.
- Ensuite le parcours de nbreq est fait pour trouver un j tel que : nbreq[j] > jeton [j]: si une demande d'accès à la ressource de Pj n'a pas encore été satisfaite, Pi envoie le jeton à Pj. Si aucun processus n'attend le jeton, Pi le garde.

L'algorithme de Ricart & Agrawala qui est une variante de l'algorithme de Le Lann assure la propriété de sureté car un seul processus ayant le jeton accède à la ressource. La propriété de vivacité est assurée si les processus distribuent équitablement le jeton aux autres processus.

2.7 **Election d'un leader**

Il existe de nombreux scenarios dans les systèmes distribués où il est nécessaire de sélectionner un processus à partir du groupe, afin qu'il puisse effectuer un rôle de coordinateur de ce groupe. On a, ainsi, besoin d'un algorithme d'élection. Les scénarios où les algorithmes d'élection sont utilisés incluent: les mécanismes de communication de groupe dans lesquels il est nécessaire d'établir un groupe coordinateur, les services répliqués dans lesquels une coordination est nécessaire, les services fiables dans lesquels la continuité du service malgré les défaillances est importante et les services auto-organisés dans lesquels les processus peuvent avoir besoin de connaître leur statut par rapport aux autres. Les algorithmes d'élection sont donc une exigence fondamentale pour de nombreuses applications distribuées. Les algorithmes d'élection sont des algorithmes distribués qui fonctionnent dans des applications distribuées. Ils doivent satisfaire un certain nombre d'exigences garantissant l'élection d'un maitre dans un environnement distribué [31]. Les exigences non fonctionnelles sont généralement les mêmes que pour tous les algorithmes d'élection :

- Évolutivité : l'algorithme devrait fonctionner correctement quel que soit le nombre de processus impliqués.
- Efficacité : il ne devrait pas y avoir de communication excessive, en particulier dans la phase de négociation.
- Réactivité : il devrait y avoir une faible latence dans la détection de l'absence d'un processus maître et par la suite l'ouverture d'une élection.

Les différents algorithmes d'élection diffèrent dans leur conception et donc dans leur fonction interne. Ceci signifie que les exigences fonctionnelles peuvent différer d'un algorithme à l'autre, mais comprennent généralement :

- La proportion du temps où un maître est élu devrait être très élevée.
- Un remplacement doit être trouvé rapidement si le maître actuel est défaillant.
- Les scénarios multi maîtres doivent être détectés et résolus rapidement.
- Les élections frauduleuses provoquées par une fausse détection d'échec du maître doivent être évitées
- Les frais généraux de communication (la bande passante de communication totale requise par l'algorithme d'élection) doit être faible.

2.7.1 **Algorithme de Chang & Robert**

L'algorithme d'élection sur un anneau virtuel [32] est similaire à l'algorithme d'élection de Bully, mais les processus sont organisés en anneau logique. Les processus peuvent communiquer uniquement avec leurs voisins logiques.

- Un processus Pi demande une élection s'il vient de rejoindre le système ou bien s'il vient de détecter que le leader est en panne (défaillant).
- L'initiateur Pi envoie un message d'élection à son successeur Pj qui est non défaillant.
- Pj ayant reçu le message d'élection, ajoute son identité ID au message.
- Lorsque le message d'élection revient vers l'initiateur Pi, celui-ci détermine le nœud de plus haute identité et le proclame comme leader dans le message coordinateur.
- le message sera éliminé lorsqu'il revient vers son émetteur.

La figure 2-8 montre le fonctionnement de l'algorithme d'élection en anneau. Les nœuds sont connectés dans un anneau logique unidirectionnel ou chaque site i dispose d'un pointeur vers son successeur. Lorsque le coordinateur 7 se plante, ce qui brise l'anneau, les nœuds 1 et 5 remarquent l'absence du maître (par l'absence des messages attendus) et déclenchent une élection en envoyant un message spécial à leurs voisins logiques (nœuds 2 et 6). Le message d'élection se propage autour de l'anneau jusqu'à ce que le nœud restant le plus élevé soit identifié et que tous les nœuds connaissent son identité. A la fin et au bout d'un tour complet le nœud 6 est élu comme coordinateur et il envoie un message de coordination a tous les nœuds de l'anneau.

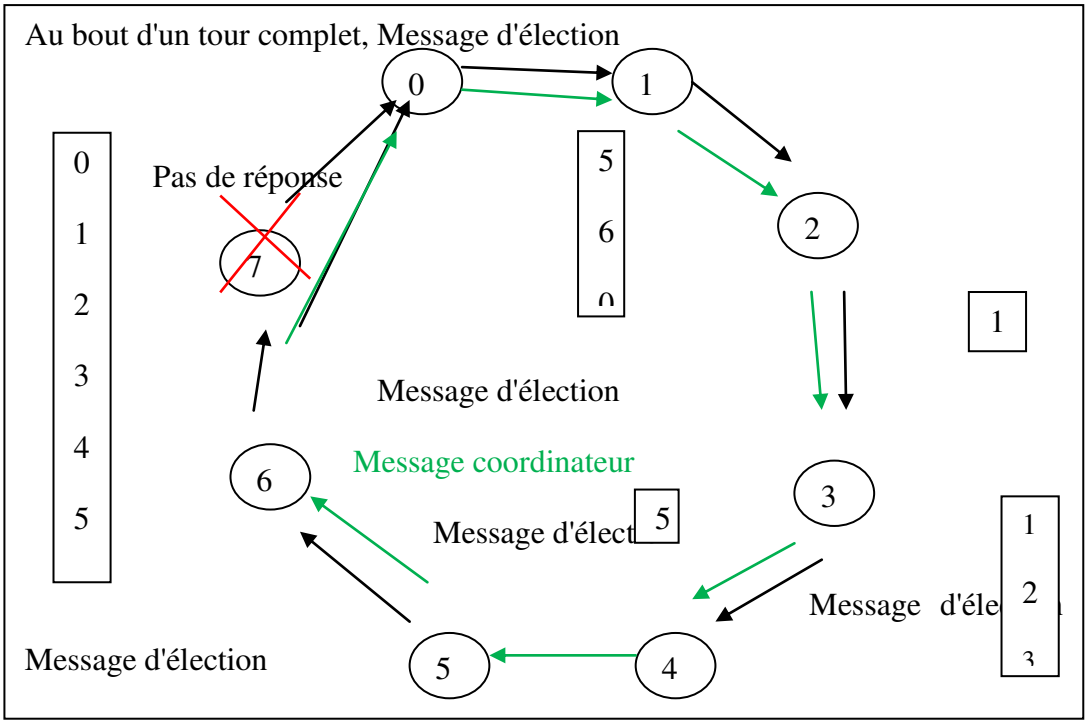

Figure 2-8 Election sur un anneau virtuel unidirectionnel

2.7.2 **Algorithme de Bully**

Un certain nombre d'algorithmes d'élection ont été conçus. L'algorithme de Bully est probablement l'algorithme d'élection le plus connu. L'algorithme d'Élection brute (Bully) se base sur une demande d'élection par inondation. Il se caractérise par une transmission fiable et un système synchrone (timeouts pour détecter les pannes). Il permet de déceler l'occurrence de pannes dans tous les processus durant l'élection. Dans cet algorithme, chaque processus connaît tous les autres processus qui ont un plus grand identificateur id que lui et peut communiquer avec eux. Il y a aussi trois types de messages :

- Élection : pour déclencher une élection.
- OK : pour répondre à un message Élection.
- Coordinateur : pour annoncer le nouveau coordinateur.

Lorsque le maître échoue, les nœuds communiquent, en tours, pour choisir un nouveau maître et l'algorithme se déroule comme suit :

- ∀ pi, pi peut lancer une élection, donc déclenchement d'une élection par pi : Envoie du message (Élection, pi) à pj , i.e., idj > idi

- Attendre l'arrivée des messages (OK, pj) depuis pj. Si aucun message (OK, pj) n'arrive durant la période T, Alors Pi devient le leader et envoie un message "coordinateur := pi" à tous les processus de plus faible id. Sinon attendre l'arrivée d'un message (coordinateur)
- Réception d'un message (Coordinateur, pj) par pi. pi considère pj comme coordinateur.
- Réception d'un message (Élection, pj) à pi : Envoi du message (OK, pi) à pj.

La figure 2-9 montre le fonctionnement de l'algorithme d'élection Bully. La partie (a) montre que le maître (nœud 7) s'est planté. Le premier nœud esclave (4) a remarqué l'absence du maître (par l'absence de ses signaux de battement), il déclenche une élection en envoyant un message d'élection (qui sert aussi à informer tous les autres esclaves de son identifiant unique). Dans la partie (b), l'esclave 4 reçoit un message OK de tous les processus ayant un identificateur supérieur de lui (5 et 6). Dans la partie (c), le processus qui envoie un message OK, déclenche une nouvelle élection. Dans la partie d, l'esclave 5 reçoit un message OK de la part du nœud 6. Si l'esclave 6 n'entend plus aucun message d'élection dans un court laps de temps donné, il suppose qu'il a gagné l'élection (du fait qu'il a l'id le plus élevé) et qu'il se hisse au statut de maître (partie e).

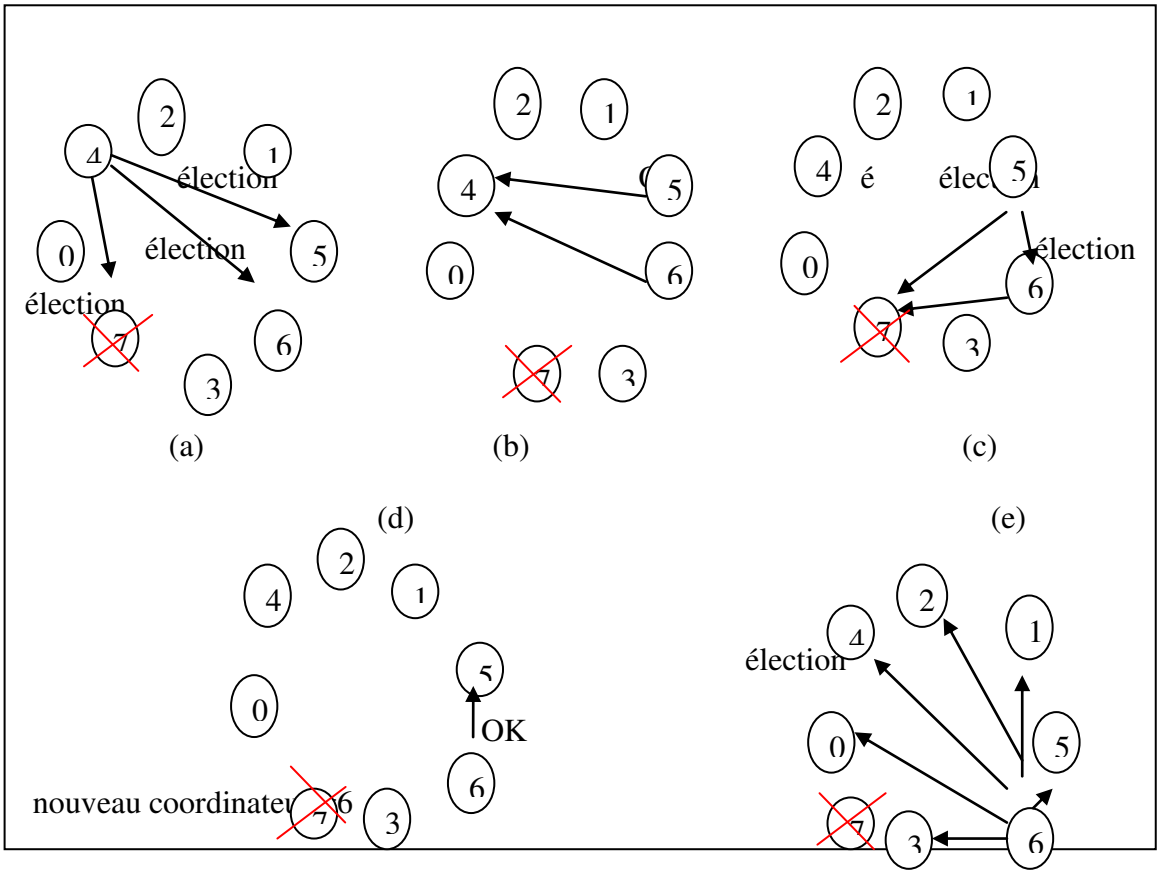

Figure 2-9 Algorithme Bully d'élection

2.8 **Consensus**

Le consensus consiste dans le dégagement d'accords de base en algorithmique distribuée. Il est considéré comme un outil générique pour la tolérance aux pannes. De façon générale, La réalisation de consensus est simple à réaliser entre les processus du système si l'environnement est sans faute, mais difficile voire impossible à réaliser dans un contexte avec faute. Ainsi, Soit un ensemble de processus P1, P2,....., Pn reliés par des canaux de communication. Chaque processus effectue une mesure ou un calcul de valeurs locales qui seront proposées à tous les autres processus. A partir de ces valeurs proposées, les processus doivent décider et se mettre d'accord sur une valeur unique commune. Un processus initie la phase d'accord. Cette phase est lancée à des instants prédéterminés. Il faut que ces quatre propriétés soient assurées par tout processus participant [33] :

- Accord : la valeur décidée est la même pour tous les processus corrects,
- Intégrité : Un processus décide au plus une fois c'est à dire qu'il n'y a pas de changement de choix de valeur,

- Validité : la valeur choisie par un processus est une des valeurs proposées par l'ensemble des processus (sa propre proposition ou celle d'un autre processus),
- Terminaison : la phase de décision se déroule en un temps fini, à la fin, tout processus correct prend une décision.

Dans un contexte où il n'y a pas de fautes, le consensus est facile à mettre en œuvre. Après la mesure de la valeur, un processus envoie sa valeur à tous les autres via une diffusion fiable. A la réception de toutes les valeurs, chaque processus applique une fonction donnée et connue par tous les autres et donc récupère la valeur commune qui sera la même pour tous. Une autre solution peut être appliquée. Cette deuxième solution est basée sur un processus coordinateur qui centralise la réception des valeurs proposées et prend la décision, puis la diffuse. Ces deux solutions fonctionnent pour les systèmes distribués qu'ils soient synchrone ou asynchrones. La différence réside dans le temps de terminaison. Mais dans un contexte avec fautes, si le modèle temporel du système est asynchrone, il y'a impossibilité de définir des algorithmes assurant à l'aboutissement d'un consensus dans tous les cas. Le problème principal est l'impossibilité de différencier un processus planté ou non. Dans le cas où le modèle temporel du système est synchrone et si les processus se plantent par des pannes franches, le consensus est atteignable systématiquement. Dans le cas des pannes byzantines, contrairement aux pannes franches, il n'est pas possible d'assurer que le consensus sera toujours atteint dans un contexte de fautes byzantines. Lamport, Shostak et Pease ont montré en 1982, via le problème des généraux byzantins que si f est le nombre de processus fautifs, il faut au minimum 3f+1 processus au total pour que le consensus soit assuré, en dessous de ce seuil, il n'est pas assuré.

2.8.1 **Algorithme des généraux byzantins**

L'algorithme des généraux byzantins [34] correspond à une réalisation de consensus dans un contexte de fautes byzantines. L'armée byzantine est formée de plusieurs généraux (processus) dont un chef et les autres sont des lieutenants. Certains généraux sont traitres (effectuent des fautes byzantines : envoi de valeurs erronées), les autres sont loyaux et doivent aboutir à prendre la bonne décision. Chaque général peut envoyer un message à n'importe quel autre général et savoir de qui vient un message. L'algorithme proposé doit assurer les conditions de "consistance interactive" (IC) :

- IC1: tous les lieutenants loyaux obéissent au même ordre.
- IC2: si le chef est loyal, les lieutenants loyaux doivent obéir à son ordre.

- Chaque lieutenant Li possède une valeur locale vi qui est l'ordre qu'il a décidé de suivre, cette valeur est déterminée en fonction de ce qu'il a reçu des autres généraux. Si le nombre de traitres est limité, il faut n >3m où n est le nombre de généraux et m le nombre de traîtres. L'algorithme P(m) s'appliquera de manière récursive : P(m-1), P(m-2) jusqu'à atteindre P(0). Pour un P(x), un processus joue le rôle de chef et diffuse sa valeur à tous les autres lieutenants.

- P(0) : pas de traitre

  1- le chef diffuse sa valeur à tous les lieutenants

  2- pour tout i, à la réception de la valeur du chef, chaque lieutenant positionne vi à la valeur reçue, s'il ne reçoit rien, vi = DEF (absence de valeur).
- P(m) : m traitres

  1- le chef diffuse sa valeur à tous les lieutenants Li

  2- pour chaque Li, à la réception de la valeur du chef, chaque lieutenant positionne vi à la valeur reçue, s'il ne reçoit rien, vi = DEF(absence de valeur).

  3- chaque lieutenant exécute P(m-1) en jouant le rôle du chef: en diffusant sa valeur aux n-2 autres lieutenants.

  4- une fois reçue la valeur vj de chacun des lieutenants (vj = DEF si rien reçu de Lj), détermine la valeur locale vi.

2.9 **Les transactions distribuées**

Une transaction distribuée est un ensemble d'opérations apparentées qui se déroulent sur un ensemble de processus distribués caractérisé par quatre propriétés ACID [35]

- Atomicité : implique que la transaction est entièrement réalisée ou pas du tout.
- Cohérence : implique que les opérations de la transaction doivent être effectuées correctement et donc l'exécution de la transaction fait passer le système d'un état cohérent à un autre
- Isolation : implique que seulement les opérations de la transaction considérée peuvent manipuler les données, les opérations des autres transactions ne peuvent pas accéder aux résultats partiels obtenus ; c'est à dire que les différentes transactions n'interférent pas entre elles.

- Durabilité : implique que les résultats d'une transaction terminée (complète) ne peuvent en aucune manière être altérés et que les effets de la transaction sont enregistrés de manière permanente

Pour maintenir les quatre propriétés d'une transaction distribuée ; une coordination est nécessaire entre les différents processus participants à son traitement. Ces processus interagissent par des communications point à point. Par le biais de cette communication ; les processus peuvent assurer :

- Le transfert des données
- La notification des anomalies
- L'initialisation, la validation ou l'annulation (avec remise à l'état précédent) de la transaction
- La terminaison de la transaction

2.9.1 **Validation atomique**

La validation atomique est au cœur de la problématique des transactions distribuées. Lors de la coordination entre processus pour se mettre d'accord sur la faisabilité de la transaction, l'ensemble des processus détermine une décision globale : soit valider la transaction et l'exécuter par tous les processus ou l'annuler et aucune action ne sera exécutée par aucun processus. Il existe de nombreux protocoles dont :

- Validation à 2 phases (2PC) : c'est la validation la plus utilisée en pratique mais n'assure pas la terminaison
- Validation à 3 phases (3PC) : c'est une extension de la validation à deux phases qui assure la terminaison

2.9.2 **Validation atomique à deux phases (2PC)**

La première phase consiste dans la demande des votes et la réception des résultats. La deuxième phase effectue la diffusion de la décision à tous et l'exécution de l'action aux niveaux des processus. Deux types de processus interagissent : un processus coordinateur qui lance et coordonne la réalisation de la validation atomique et l'ensemble des processus participants à la transaction. Une demande est adressée à chacun d'entre eux pour savoir s'il peut ou non réaliser l'action (voir figure 2-10)

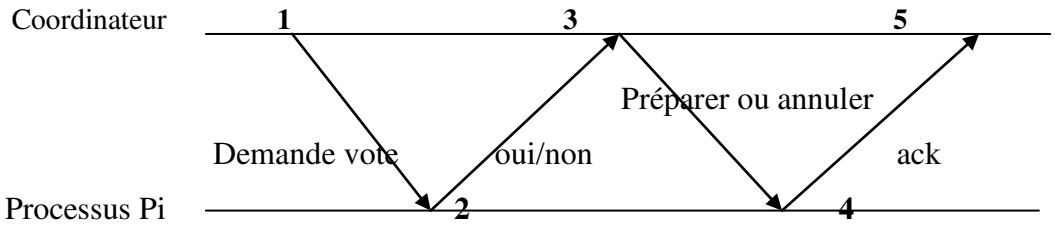

Figure 2-10 Diagramme de Validation à deux phases

1. Le coordinateur envoie une demande de vote à tous les processus
2. Chaque processus étudie la demande selon son contexte local et répond par oui ou non (selon qu'il peut ou pas exécuter la requête demandée)
3. Quand le coordinateur a reçu tous les votes, il envoie la décision finale : valider si tout le monde a répondu oui ou annuler.
4. Chaque processus exécute la requête s'il reçoit valider ou ne fait rien s'il reçoit annuler et envoie un acquittement au coordinateur pour lui préciser qu'il a reçu la décision globale.

On peut déduire les problèmes suivants dans le cas de la validation atomique à deux phases:

- La terminaison n'est pas assurée, le protocole est potentiellement bloquant : un processus peut se retrouver dans un état où il ne sait pas quoi faire,

- Le problème général est qu'entre les moments 2 et 4, les processus sont dans une zone d'incertitude

2.9.3 **Validation atomique à trois phases (3PC)**

La validation atomique à 3 phases (3PC) rajoute une étape dans laquelle, les processus sont informés de la décision globale avant de leur demander de la réaliser. La première phase dans cette validation consiste dans la demande de vote. La deuxième phase consiste dans l'envoi de la décision globale à tous les processus et attente des acquittements de la réception. Dans la troisième phase, la diffusion de la demande d'exécution de la requête est effectuée sauf s'il y'a une décision d'annulation.

1. Le coordinateur envoie une demande de vote à tous les processus
2. Chaque processus étudie la demande selon son contexte local et répond par oui ou non (selon qu'il peut ou non exécuter la requête demandée)

3. Quand le coordinateur a reçu tous les votes, il envoie soit ‘annuler’ si un processus avait voté non, la validation est alors terminée et il n’y aura pas d'autres échanges de messages. Sinon il envoie une demande aux processus afin qu’ils se préparent à exécuter la requête.
4. Chaque processus envoie un acquittement au coordinateur de la réception de cette demande.
5. Une fois tous les acquittements reçus, le coordinateur diffuse la validation finale.
6. Chaque processus exécute alors la requête.

La figure 2-11 illustre ce principe à 3CP.

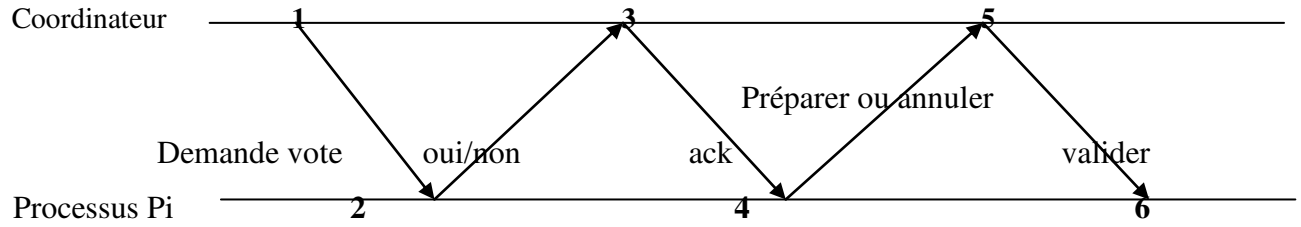

Figure 2-11 Diagramme de Validation à trois phases

2.9.4 **La diffusion atomique**

Dans plusieurs contextes distribués, les processus du système ou d'un sous ensemble doivent être adressés comme un tout. Il y’a, ainsi, un besoin de diffusion. La diffusion est un mode de communication par envoi de message pour de multiples récepteurs. Dans ce type de communication ; un processus émetteur envoie un message pour d'information à tous les processus du groupe (diffusion générale), ou à un sous ensemble (multicast). La diffusion est réalisée au-dessus d'un système de communication sous-jacent, qui peut être point à point. La diffusion est un outil préalable dans l'accord et le consensus entre processus pour prendre des décisions. Le protocole de diffusion basique utilise deux primitives de diffusion exécutées par un processus p :

- B_broadcast(m) : le processus p diffuse un message m à tous les autres processus.
- B_delivrer(m) : le message m sera délivré à un processus p.

Deux autres opérations sont associées aux canaux de communication

- recv(m,q): Réception du message m venant du processus q. En ce sens que les messages ne sont pas directement reçus par l'application : ils sont reçus par recv(m), mis en attente, traités puis délivrés à l'application par délivrer(m).
- send(m,q): envoi du message m par p au processus q. Les messages sont diffusés par broadcast(m) qui n'est pas atomique en pratique : elle s'appuie sur la primitive d'envoi point à point send(m, p). La figure 2-12 illustre l'architecture du protocole de diffusion

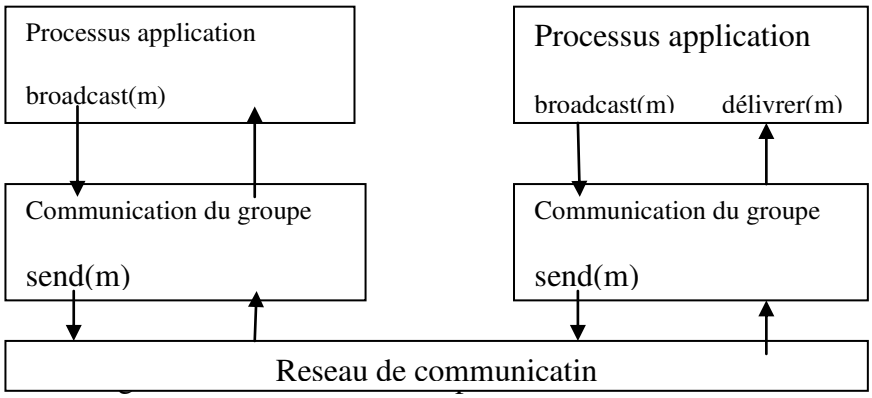

Figure 2-12 Architecture du protocole de diffusion

Un groupe peut être fermé ; le broadcast(m) ne peut être appelé que par un membre du groupe. Comme il peut être ouvert et dans ce cas, le broadcast(m) peut être appelé par un processus extérieur au groupe. La figure 2-13 représente la différence entre les groupes de diffusion fermé et ouvert.

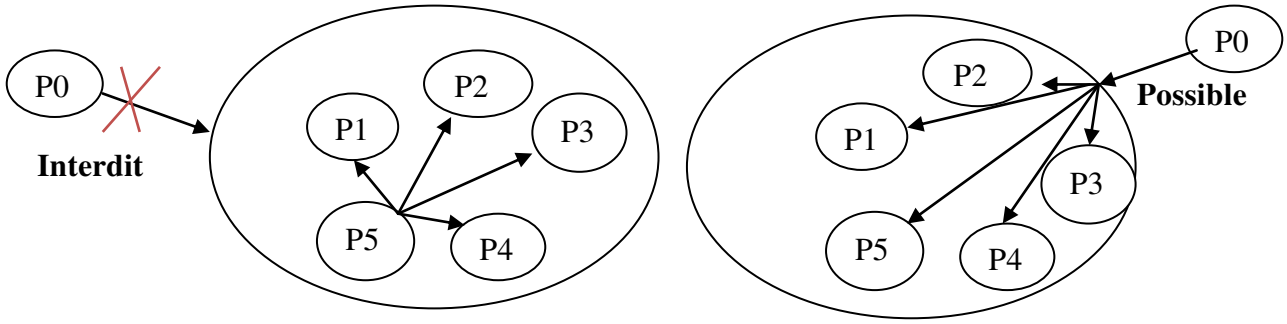

Figure 2-13 groupes de diffusion fermé et ouvert

Il existe de nombreuses formes de diffusion selon les propriétés requises. Il existe des propriétés indépendantes de l'ordre d'émission (concernent uniquement les récepteurs).

- Diffusion fiable : un message est délivré à tous ses destinataires ou à aucun.

- Diffusion totalement ordonnée (ou atomique) : la diffusion est fiable et les messages sont délivrés dans le même ordre à tous leurs destinataires.
- Diffusion FIFO : deux messages issus du même émetteur sont délivrés à tous les récepteurs dans leur ordre d'émission
- Diffusion causale : pour tout récepteur, l'ordre de réception de deux messages respecte leur ordre causal d'émission (implique FIFO)
- Diffusion fiable temporisée : c'est une diffusion fiable qui a une borne temporelle de délivrance (système de communication synchrone). Il existe une constante C telle que si un message m est diffusé à l'instant t, alors aucun processus correct ne délivre m après le temps t+C.

2.9.5 **La diffusion fiable (Reliable diffusion)**

C'est un type de diffusion qui garantit la propriété de remise. Tous les processus corrects délivrent un message diffusé en assurant les caractéristiques suivantes :

- Accord : si un processus correct p délivre un message m, tous les processus corrects délivrent m (au bout d'un temps fini).
- Validité : si un processus correct diffuse un message m, tous les processus corrects délivrent le message m (au bout d'un temps fini).
- Intégrité : pour chaque message de diffusion m, tout processus correct délivre m au plus une fois, et seulement si m a été antérieurement diffusé par l'émetteur de m.

Pendant la diffusion fiable, à la première réception d'un message, un processus le rediffuse à tous les autres ce qui permet d'assurer que tout le monde le recevra même si certains plantent pendant un B_Broadcast(m). La Mise en œuvre de la diffusion fiable se fait comme suit :

- R_broadcast(m) sur p : le processus p envoie le message m à tout le monde via la diffusion basique B-broadcast(m).
- Lors d'un receive(m,q) sur p: s'il n'y a pas de réception antérieure du message m, exécuter B-broadcast(m) pour diffuser le message à tout le monde si p$\neq$ q. Ensuite délivrer le message et exécuter R_delivrer(m)

2.9.5.1 Avantage

- La fiabilité ne repose pas sur la détection de la panne de l'émetteur, donc l'algorithme est valable dans tous les modèles temporels.

2.9.5.2 Inconvénients

- L'algorithme est très couteux en terme de messages : il génère n(n-1) envois par diffusion, ce qui le rend inutilisable en pratique.
- L'algorithme ne garantit aucun ordre de remise
- Aucune garantie de délivrance n'est offerte pour un processus fautif.

2.9.6 **La diffusion fiable uniforme (Reliable Uniform diffusion)**

Ce type de diffusion prend en compte tous les processus y compris ceux qui sont plantés. Si m est délivré par un processus fautif ou correct alors tout processus correct fini aussi par délivrer m. Les propriétés d'accord, validité et intégrité sont les même que pour la diffusion fiable sauf qu'elles ne sont pas restreintes aux processus corrects.

Trois grands types d'ordonnancement des messages

- Ordre FIFO : si un processus diffuse un message m puis un message m', alors un processus correct qui délivre m', délivrera m avant m'. Si un processus plante après avoir diffusé m, la contrainte reste correcte bien qu'il ne délivrera pas m' : pas d'obligation de délivrance de m' (ni de m) pour tous les processus, contexte non fiable.
- Ordre causal : si la diffusion d'un message m' dépendait causalement de l'émission d'un message m, alors un processus correct qui délivre m' délivrera m avant m'.
- Ordre total : si un processus correct délivre m avant de délivrer m', alors tous les autres processus corrects délivreront m avant m'. Même ordre de délivrance des messages pour tous les processus corrects.

La relation entre les diffusions est présentée par la figure 2-14

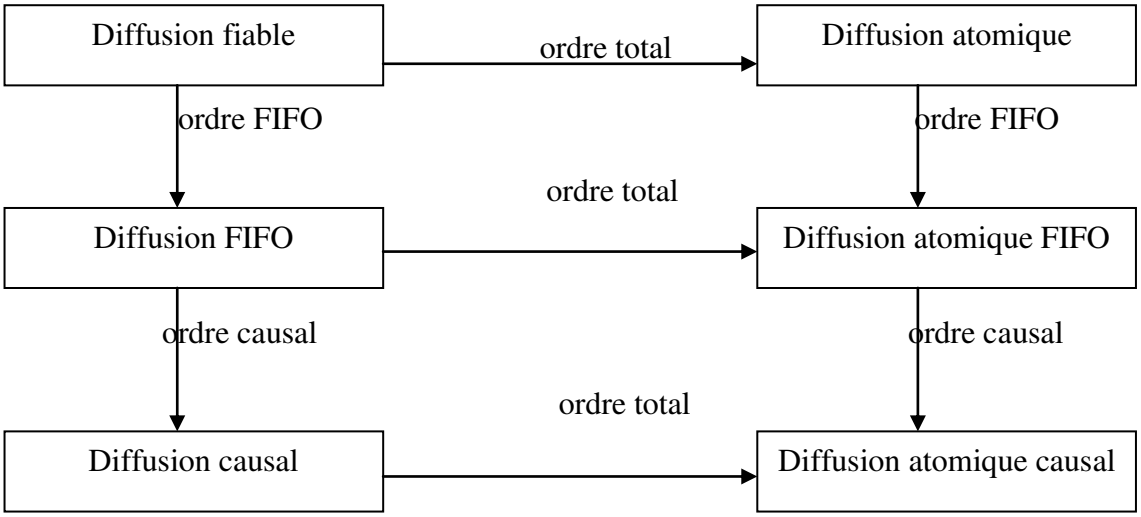

Figure 2-14 La relation entre les différentes diffusions

2.10 **La terminaison d'une application distribuée**

Dans une application distribuée ou un calcul reparti, chaque processus ou site i est soit dans l'état actif s'il est en train d'exécuter un code, soit dans l'état passif s'il est en attente ou s'il a terminé son exécution. La problématique de la terminaison d'une application distribuée consiste dans la détection de la fin d'un calcul distribué effectué par plusieurs processus. Détecter la terminaison revient à montrer que tous les processus sont dans l'état passif et qu'aucun message n'est en transit. La propriété de la terminaison est stable, une fois vraie, elle le restera toujours au cours de temps.

2.10.1 **Protocole de Hvang pour la détection de la terminaison**

Soit une application distribuée représentée par un ensemble de processus Pi $\in$ [1...N]. soit

- P0: agent contrôleur "initiateur du protocole"
- w_i: poids du processus Pi initialement w_0=1 et $\forall$ Pj i $\neq$ j w_j=0
- B(w): message d'application avec w (message d'exécution du poids w)
- C(w): message de contrôle émis par un processus au contrôleur du poids w.

Protocole

- Pi envoie un message d'exécution d'application B(w_j) et B(w_k) à Pj et Pk, en divisant w_i en w_i', w_j et w_k tel que w_i= w_i' + wj+wk avec wi' nouveau poids du pi.
- Quand Pj reçoit un message d'exécution d'application B(wj) de Pi: w_j= w+w_j, si Pj ne fait rien alors Pj devient actif

- Quand Pj devient actif, il envoie un message de contrôle à P0 (son père), met w_j à 0 et Pj devient oisif (ne fait rien)
- Quand Pi reçoit un message de controle C(w): $w_i=w_i+w$, si $w_i=1$ alors l'application est terminée.

2.10.2 **Terminaison sur un anneau : Algorithme de Misra**

Pour la détection de la terminaison, on structure l'ensemble des processus en anneau de contrôle [36]: sur cet anneau les processus sont rangés dans un certain ordre selon leur identité : P0,P 1,..., Pi,...Pn-1; les communications sur cet anneau sont unidirectionnelles: de Pi vers Pi-1. Une détection de terminaison est initialisée par un processus particulier, soit P0.

2.10.2.1 Principe de l'algorithme

L'algorithme de Misra consiste à faire circuler un jeton entre les processus pour les visiter. La constatation par le jeton que tous les processus sont passifs lorsqu'ils ont été visités ne permet pas de conclure à la terminaison car des processus peuvent être réactivés et des messages peuvent être en transit. Dans le cas particulier où la topologie de communication entre processus est en anneau, le jeton peut affirmer que le calcul est terminé si, après avoir effectué un tour sur l'anneau, il constate que chaque processus est resté passif depuis sa dernière visite à ce processus. En effet, les messages ne pouvant pas se doubler, entre deux visites du jeton un processus a nécessairement reçu les messages qui étaient en transit lors de la première visite du jeton. Si le jeton a effectué deux tours sur l'anneau et qu'il n'a observé que des processus restés en permanence passifs, on peut conclure à la passivité des processus et à l'absence de messages en transit, i.e à la terminaison.

• lorsqu'un processus devient actif il devient noir, et c'est le jeton qui le peint en blanc lorsqu'il quitte le processus (devient passif).

• ainsi si le jeton retrouve blanc un processus, c'est que ce dernier est resté passif en permanence depuis sa dernière visite.

• lorsque le jeton a visité les n processus et qu'il les a trouvés tous blancs, il peut conclure en la terminaison (initialement les processus sont noirs).

• tous les processus ont des comportements identiques.

2.11 **Les détecteurs de défaillances dans les systèmes distribués**

En général, un mécanisme de détection de panne dans un système distribué décrète un délai – *time out* – au-delà duquel, si aucune réponse n'a été reçue, le processus est

considéré comme étant en panne. Dans un système synchrone pour lequel le délai maximum de communication et de traitement d'une requête est connu, ce système détectera les pannes correctement. Cependant, dans un système totalement asynchrone et pour lequel les délais de réponse ne sont pas prévisibles, comme c'est le cas pour la plupart des systèmes distribués, cette méthode ne permettra pas de distinguer un processus en panne d'un processus très lent. Les détecteurs de défaillances : sont des éléments associés au processus qui fournissent des informations (pas nécessairement entièrement précises) concernant les processus ayant crashé (en panne/planté). Un détecteur de défaillance est considéré comme une boite noire donnant des suspicions relatives aux nœuds défaillants [37]. La précision et l'exactitude des suspicions dépendent de la force du modèle utilisé. Les détecteurs de défaillances ont deux propriétés basiques qui sont :

- Complétude : un processus défaillant doit être détecté comme défaillant
  – Complétude forte : Il existe un instant à partir duquel tout processus défaillant est suspecté par **tous** les processus corrects
  – Complétude Faible : Il existe un instant à partir duquel tout processus défaillant est suspecté par **un** processus correct.

- Précision : un processus correct ne doit pas être considéré comme défaillant
  – Forte : aucun processus correct n'est suspecté
  – Faible : il existe au moins un processus correct qui n'est jamais suspecté
  – Finalement forte : il existe un instant à partir duquel tout processus correct n'est plus suspecté par aucun processus correct
  – Finalement faible : il existe un instant à partir duquel au moins un processus correct n'est suspecté par aucun processus correct

2.12 **Conclusion**

L'absence d'une horloge globale dans un système distribué impose des difficultés dans la définition d'un état global du système ce qui rend la coopération entre les différents processus difficile. Plusieurs solutions ont été imaginées et mises en œuvre. Nous avons traité, dans ce chapitre d'un ensemble d'algorithmes concernant la gestion du temps dans les systèmes distribués qui facilitent l'exploitation de mécanismes de coopération entre les composants du système tels que l'exclusion mutuelle, l'élection d'un coordinateur, les consensus, les transactions distribuées et la diffusion atomique. Dans le chapitre suivant nous allons-nous intéresser aux réseaux de capteurs sans fils qui sont un type spécifique et pratique des systèmes distribués.

Chapitre 3 : Les réseaux de capteurs sans fil

3.1 **Introduction.**

Au cours de ces dernières décennies, l'apparition de la technologie de miniaturisation de plus en plus poussée, a contribué à faire apparaitre et évoluer de nouvelles générations de réseaux de capteurs sans qui ont attiré l'attention des chercheurs et des industriels. Les réseaux de capteurs sans fils sont devenus un domaine de recherche très important car ils ouvrent la porte sur de larges domaines d'applications, on peut noter le militaire, l'environnemental, le médical, le domestique et l'industriel. Les réseaux de capteurs sans fils (RCSF), en anglais Wireless sensor networks (WSNs) sont considérés comme un type général de systèmes distribués et plus particulièrement des réseaux ad hoc. Un RCSF est composé d'un grand nombre de nœuds capteurs dispersés aléatoirement dans une zone géographique qui n'est, généralement, pas accessible par l'être humain. Un nœud capteur est équipé de petites unités de traitement, de stockage, d'acquisition et de transmission de l'information et d'une source d'alimentation en énergie (généralement une batterie). Son rôle principal est de collecter les informations à partir de l'environnement dans lequel il est déployé afin de les router vers une station de base (SB) à travers des liaisons sans fil. Dans les réseaux de capteur sans fil, la minimisation de la consommation d'énergie est considérée comme un critère de performance important pour garantir une durée de vie maximale au réseau. Dans ce chapitre, nous traiterons des réseaux de capteurs sans fil hiérarchiques en nous focalisant sur la coopération entre les nœuds capteurs pour effectuer les opérations de clustering, l'élection d'un cluster-head et pour établir les itinéraires de routage des données collectées à partir de l'environnement.

3.2 **Architectures des réseaux de capteurs sans fil**

Un réseau de capteurs sans fil (RCSF) se compose généralement d'un grand nombre de nœuds capteurs à faible coût, distribués aléatoirement dans une zone géographique limitée. Ces nœuds capteurs sont équipés généralement d'une petite batterie d'énergie non rechargeable et généralement irremplaçable en raison de l'environnement hostile de

déploiement. L'architecture des réseaux de capteurs se base sur l'interaction entre les trois éléments suivants [38] :

- **nœud capteur :** c'est l'élément de base qui compose le système d'acquisition de données à partir de l'environnement.
- **Champs de capture** : Appelé aussi « champ de captage » ; c'est une zone géographique où sont placés les capteurs du réseau pour superviser ou surveiller des phénomènes divers.
- **stations de base (Sink)** : Nommée aussi nœud-puits où l'écoulement des données se termine. Ces nœuds-puits sont des nœuds spéciaux qui possèdent plus de ressources matérielles et permettent de collecter et de stocker les informations issues des capteurs**.**

L'interaction entre ces éléments suit une séquence d'opérations telles que le déploiement, la détection d'évènements et la transmission des données. Ces opérations doivent être exécutées dans l'ordre comme indiqué figure 3-1.

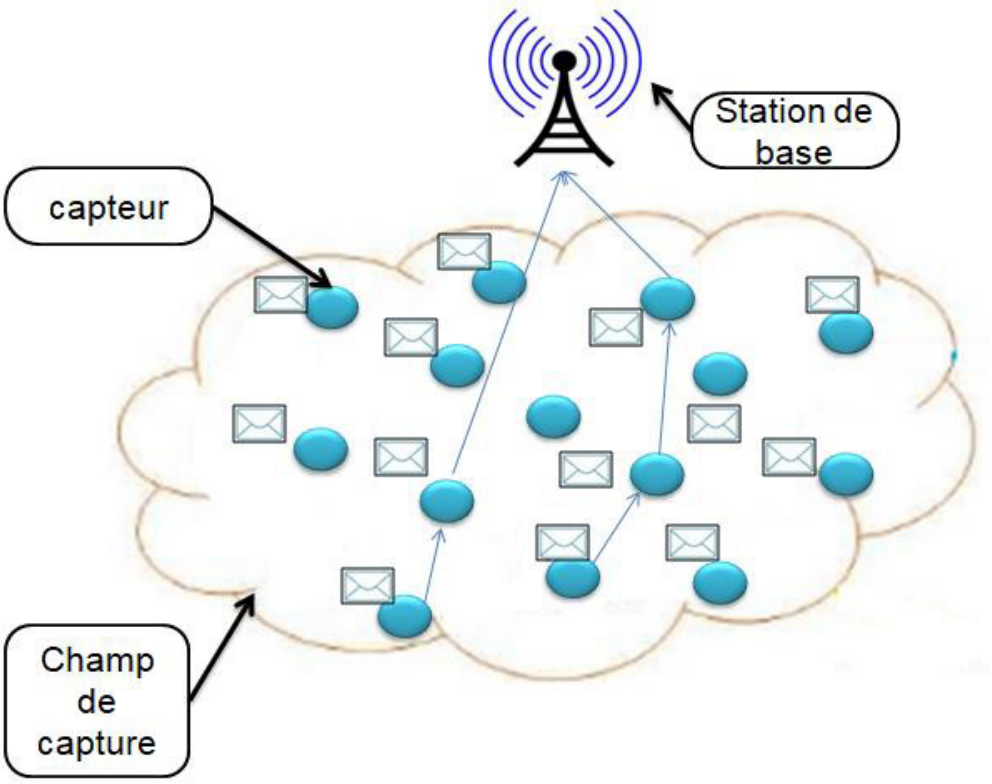

Figure 3-1 Fonctionnement des réseaux de capteur.

1. Les capteurs sont déployés aléatoirement dans un champ limité appelé zone d'intérêt
2. Le capteur surveille son environnement et lorsqu'il détecte un événement il le traite localement.
3. Les données acquises par le capteur sont transmises à la station de base via un chemin direct (un seul saut) ou via d'autres capteurs (par multiples sauts)

3.3 **Facteurs et contraintes liées aux réseaux de capteurs**

Les domaines d'applications de la technologie des RCSF sont multiples. On peut citer loin d'être exhaustif : le domaine militaire et la détection pour la collecte d'informations sur les positions de l'ennemi, la surveillance des zones hostiles (contaminées), la détection d'agents chimiques et bactériologiques. L'environnement : pour la détection des feux de forêt, pour le contrôle de la qualité de l'eau ou de l'air. Le bâtiment, lors de tremblements de terre, pour aider les secouristes à retrouver les victimes (capteurs emprisonnés dans le béton à la construction qui détectent le niveau de bruit)**.** Le domaine de l'industrie dans la gestion des stocks et le domaine médical ; par exemple pour contrôler le rythme cardiaque des patients**.** Pour constituer un RCSF qui fonctionne correctement et qui réponde aux besoins des applications, il faut tenir compte d'un grand nombre de contraintes et facteurs. Ces facteurs représentent la base de la conception de protocoles ou d'algorithmes pour les réseaux de capteurs.

3.3.1 **L'agrégation des données**

Les nœuds des réseaux de capteurs sont déployés dans des zones géographiques limitées appelées zones de couverture, ce qui peut impliquer souvent la redondance de certaines données collectées par ces nœuds. L'agrégation des données collectées par des nœuds intermédiaires avant la transmission à la station de base est une tâche importante pour construire des approches efficaces en termes de conservation d'énergie lors des transmissions des données.

3.3.2 **La tolérance aux pannes**

Certains nœuds capteurs peuvent être sujets à des pannes à cause d'un manque d'énergie, d'un matériel défectueux, lors d'interférences environnementales ou autres causes possibles et parfois imprévisibles. La panne d'un nœud capteur ne doit pas affecter le fonctionnement global du réseau. C'est le problème de fiabilité ou de tolérance aux pannes. La tolérance aux pannes est donc la capacité à maintenir les fonctionnalités du réseau sans interruption

due à une panne d'un nœud capteur et c'est là une contrainte importante dans la conception des RCSFs.

3.3.3 **La consommation d'énergie**

Le nœud capteur est un dispositif micro-électronique, alimenté par une batterie de faible puissance. En général, les RCSF sont dispersés dans des environnements hostiles ; qui ne sont pas accessibles par un être humain. Le remplacement d'une batterie est alors une tâche impossible. En conséquence, la durée de vie du capteur est donc dépendante de la durée de vie de sa batterie. La gestion de l'énergie est, ainsi, une contrainte clé dans les réseaux de capteurs [39].

3.3.4 **La limitation des ressources**

Une des caractéristiques des nœuds capteurs est leur taille très réduite ce qui leur confère plusieurs avantages tels que la flexibilité de déploiement et un encombrement réduit leur permettant d'être facilement incrustés, voire dissimulés, dans divers systèmes et environnements. Cependant, une taille réduite d'un nœud peut limiter la puissance de batterie utilisée pour alimenter le capteur et réduire les capacités de traitement et de stockage, bien que vues les récents développements technologiques de grandes performances peuvent être atteintes sur des puces dont les surfaces sont de plus en plus réduites. Malgré tous les efforts et les améliorations consentis, le problème majeur des RCSFs concerne les contraintes de durée de vie du réseau et les faibles capacités des nœuds. Ainsi, malgré la miniaturisation et la réduction du coût de fabrication, ces capteurs sont généralement dotés de ressources limitées en termes de puissance de transmission, de capacité de traitement et de stockage des données et d'énergie. Ces limitations présentent des grands défis pour les industriels [40].

3.3.5 **Topologie dynamique**

Dans plusieurs applications, la topologie des réseaux de capteurs sans fil peut être organisée de façon. Les nœuds capteurs peuvent changer de position après leur déploiement. Permettant, ainsi, des configurations et organisations du réseau à des instants imprévisibles et d'une manière rapide. Dans certaines applications, ou des déplacements permanents des nœuds capteurs sont nécessaires comme c'est le cas pour le suivi des déplacements des animaux, les capteurs du réseau sont attachés à des unités mobiles qui se déplacent d'une façon libre et arbitraire [41].

3.3.6 **Facteur d'échelle**

Il n'est pas rare de trouver des milliers voire même des millions de nœuds capteurs dispersés dans un champ de captage. Le passage à l'échelle est une contrainte importante dans les RCSF afin de créer des grands réseaux répondant aux besoins des applications. Ces réseaux sont censés fonctionner avec des densités de capteurs très grandes. Par conséquent, ils engendrent beaucoup de schémas pour pouvoir garantir un bon fonctionnement [42]. Plusieurs solutions ont été proposées pour permettre aux RCSF de passer à l'échelle. Parmi ces dernières, on peut citer le clustering et l'établissement des protocoles de routage.

3.3.7 **Coûts du réseau**

Le nombre, parfois, très important de nœuds capteurs influence grandement les coûts du réseau. La réduction de ces coûts est un autre défi à relever et dont l'optimisation n'est pas chose aisée. De plus ; d'autres contraintes doivent être envisagées telles celles liées aux possibles extensions du réseau, l'utilisation de matériel de précision pour des applications spécifiques, la prévision de pannes en dédoublant éventuellement le nombre de nœuds ainsi que les couts de maintenance et de remplacement (la durée de vie des nœuds d'un réseau est dictée par la nature intrinsèque des nœuds capteurs le composants). Toutes ces préoccupations doivent être prises en considération lors de l'étape de conception du réseau.

3.3.8 **Sécurité**

Les réseaux de capteurs sans fil sont utilisés dans une vaste gamme de domaines tels que dans des applications liées à la surveillance des infrastructures critiques comme les applications militaires et médicales. Ces applications nécessitent un fonctionnement parfait sans faute mais ce n'est pas le cas dans le monde réel, où les nœuds peuvent être attaqués et endommager le bon fonctionnement du réseau. Garantir la sécurité de ce type de réseau est une tâche difficile et un autre défi pour les RCSFs.

3.4 **Architecture matérielle d'un capteur**

Parmi les définitions générales d'un nœud capteur trouvées dans la littérature :'c'est un petit dispositif électronique sans fil qui produit une réponse mesurable à un changement dans une condition physique ou chimique peut effectuer de nombreuses tâches, y compris la détection d'événements, le traitement de données et la communication de données' [43]. Ainsi, un nœud capteur peut être défini comme étant un petit appareil sans fil capable de collecter des données à partir de son environnement, les traiter localement et les

communiquer. Généralement, un nœud capteur est composé d'un ensemble des quatre unités essentielles présentées ci-dessous :

3.4.1 **Unités d'acquisition ou de capture**

Ces unités sont composées de deux sous-unités : le capteur qui produit un signal analogique suite à un phénomène observé et l'ADC (Analogique Digital Converter) qui est un convertisseur qui transforme les signaux analogiques en signaux numériques afin de les traiter par les unités de calcul. [44]

3.4.2 **Les unités de traitement**

Ces unités sont considérées comme étant le cerveau d'un capteur, elles constituent un élément très important pour le capteur. Elles sont composées, généralement, de micro contrôleurs et d'une unité de stockage. Elles assurent au nœud la gestion et l'exécution de taches de calcul et autres procédures assurent la collaboration avec les autres nœuds pour accomplir des taches parallèles et concurrentes [45]. Ces unités de traitement sont dotées deux interfaces : une interface qui contrôle les unités d'acquisition et la seconde pour le contrôle des unités de transmission. Elles reçoivent les données en provenance des premières unités, puis elles effectuent les traitements appropriés et elles envoient les données ainsi traitées aux unités de transmission [46].

3.4.3 **Les unités de transmission (ou unités de communication)**

Les unités de transmission fournissent des moyens de communication entre les nœuds du réseau ; elles assurent les échanges de données (émissions et réceptions) entre ces nœuds via des supports de communication radio [45] et en utilisant des protocoles de communication.

3.4.4 **L'unité d'alimentation (Batterie)**

Cette unité constitue la source d'énergie du nœud capteur, c'est l'unité qui alimente les trois unités citées précédemment et fournit toute l'énergie nécessaire à leur fonctionnement [47]. La gestion de cette unité est devenue un sujet de recherche très intense vue que les nœuds capteurs peuvent être déployés dans des environnements où il n'est pas possible de remplacer ou de recharger ces batteries une fois leur énergie épuisée.

Il existe d'autres composants complémentaires qui peuvent être reliés aux capteurs comme les systèmes de localisation GPS (Global Position System) et les mobilisateurs qui permettent le déplacement des capteurs, etc. [46]. La figure 3-2 représente l'architecture matérielle d'un capteur.

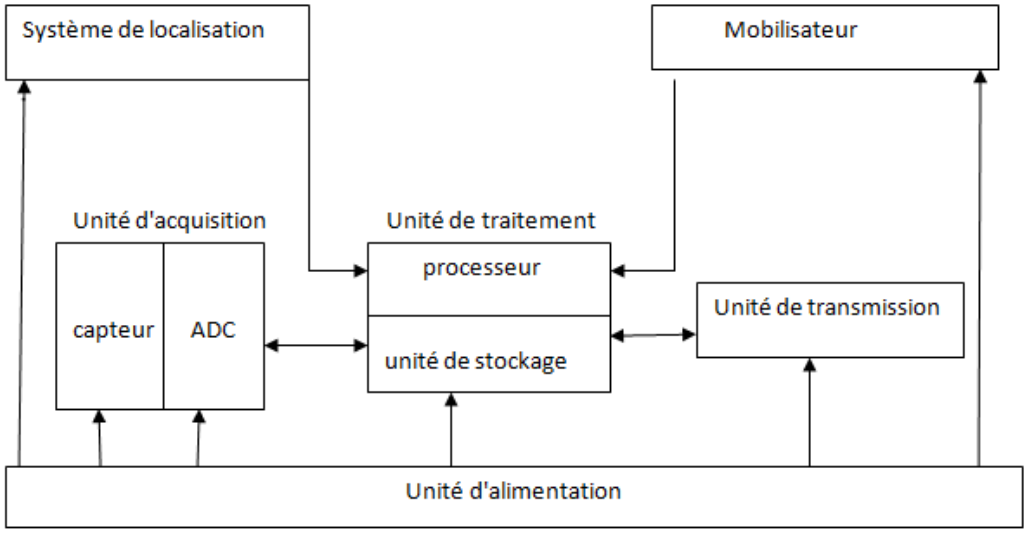

Figure 3-2 Architecture matérielle d'un capteur [46].

3.5 **Consommation d'énergie dans les RCSF**

Comme les nœuds capteurs sont de dispositifs qui possèdent de petites batteries, la conservation d'énergie est un défi majeur et l'exigence la plus critique pour prolonger la durée de vie des réseaux de capteurs sans fils. Afin de maximiser la durée de vie globale du réseau, il faut éviter toutes les sources de surconsommations et de dissipation d'énergie telles que : les collisions, la sur écoute, la surcharge et l'écoute à vide. Il convient d'exploiter les techniques et mécanismes de la conservation d'énergie comme l'organisation des échanges des données en minimisant le nombre d'émissions et de réceptions, limiter les accusées de réception et utiliser des architecture hiérarchique [48]. Plusieurs mécanismes et techniques sont proposés pour la conservation d'énergie. Dans cette section nous citons les plus connus et les plus efficaces qui sont classés comme suit : techniques de conservation au niveau des données, au niveau de la communication, au niveau des liens radio et au niveau du capteur. Cette classification est résumée figure 3-3

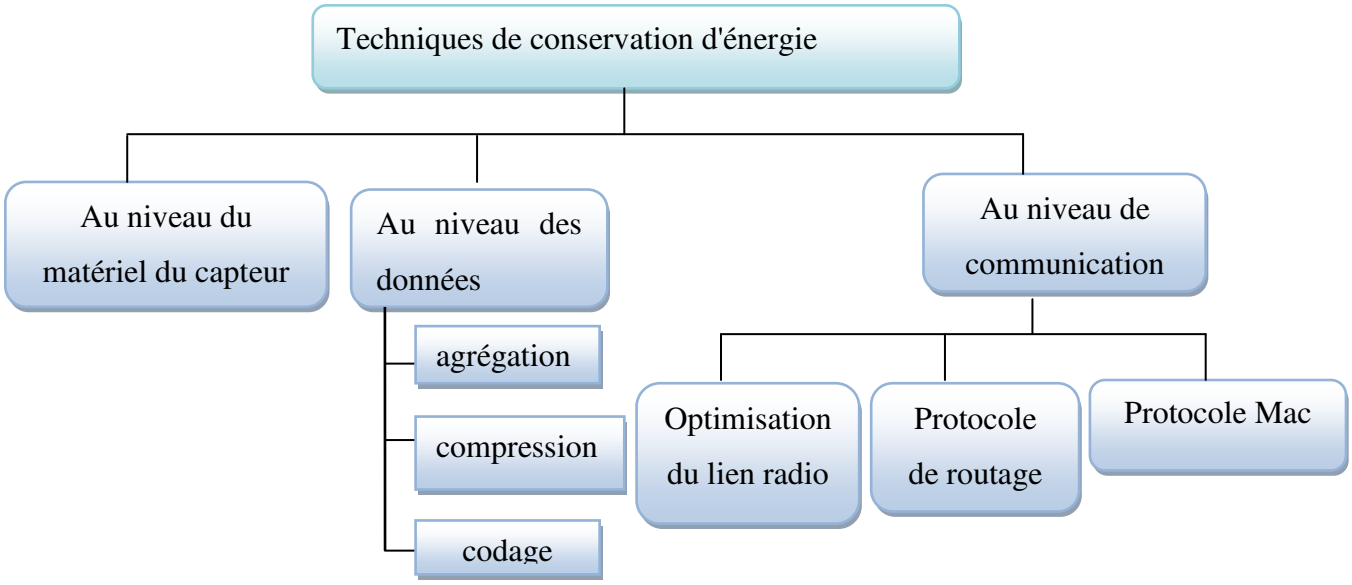

Figure 3-3 Les techniques de conservation d'énergie

3.6 **Le routage dans les réseaux de capteurs**

Dans les réseaux de capteur sans fil, la minimisation de la consommation d'énergie est considérée comme un critère de performance important pour assurer une durée de vie maximale du réseau. Par conséquent, comment minimiser la consommation d'énergie des capteurs est une question importante et difficile, qui est aussi au centre de la conception des protocoles de routage de données dans les RCSFs. Le routage des données dans les réseaux de capteurs sans fils est reconnu comme constituant un domaine de recherches très actif car c'est l'élément fondamental dans la communication réseau. L'objectif principal d'un protocole de routage pour un réseau de capteurs sans fils est l'établissement correct et efficace d'itinéraires entre une paire de nœuds afin que les messages puissent être acheminés, il s'agit d'assurer la livraison des messages d'une source vers certaines destinations. Tous les flux qui transitent sur le réseau ont pour unique destination la station de base. Cela implique deux fonctions : la découverte des routes vers les destinations et la transmissions des messages via les routes découvertes. Les protocoles de routage développés pour le RCSF ont pris en compte la contrainte d'énergie. En effet, ces protocoles tentent de conserver le maximum d'énergie sans perdre l'efficacité de routage. Malgré la performance de ces protocoles, il existe toujours une amélioration tant qu'il y'a une consommation d'énergie. Un grand nombre de protocoles de routage ont été conçus spécialement pour les réseaux de capteurs sans fils et plusieurs classifications sont proposées pour ces protocoles. Parmi les plus connues ; celle basées sur la manière d'établissement des routes qui inclus trois classes : réactifs, proactifs et hybrides. Une autre

classification très étudiée est basée sur la structure logique du réseau où trois grandes familles de structures sont citées : plat, hiérarchique et géographique. Cette classification constituera un premier axe d'étude dans ce qui suit. La figure 3-4 présente une classification pour les protocoles de routage dans les RCSFs.

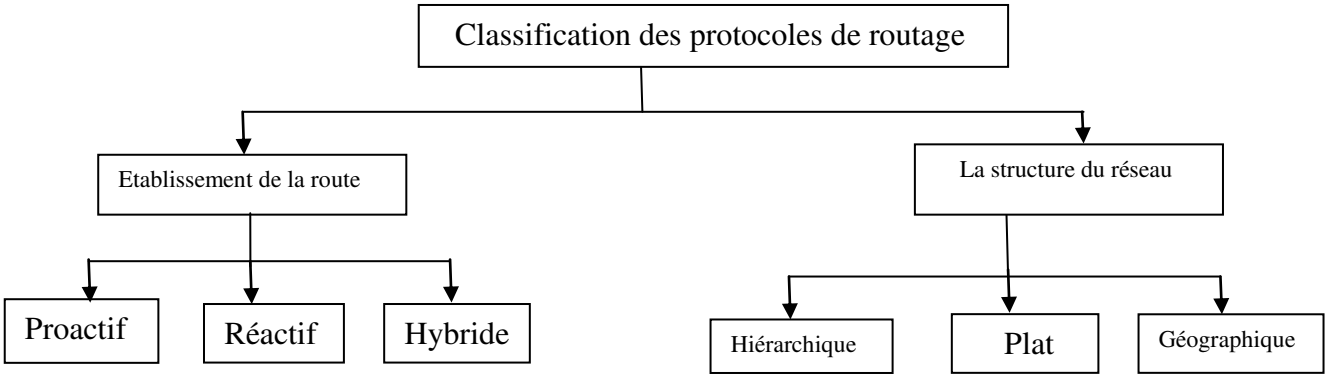

Figure 3-4 Classification des protocoles de routage pour les RCSFs

3.6.1 **Le routage plat**

Dans les protocoles de routage à plat, tous les nœuds sont identiques en termes de fonctions et de responsabilités. Les itinéraires établis dans un réseau de capteur sans fils avec ce type de routage sont généralement des routes multi-sauts qui sont choisies en se basant sur des critères comme l'énergie, la distance ou autres.

3.6.2 **Le routage géographique**

Dans un routage géographique (ou de localisation), une zone est ciblée comme destination des messages. Cette zone comporte plusieurs nœuds, n'importe quel nœud peut jouer le rôle de destinateur du message émis vers cette zone. Dans les RCSFs, le routage géographique prend tout son sens dans la manière avec laquelle les nœuds capteurs sont dispersés dans une zone géographique limitée.

3.6.3 **Le routage hiérarchique**

Dans le routage hiérarchique, les nœuds sont organisés en groupes appelés clusters, pour chaque cluster un nœud est élu comme un cluster head pour gérer la communication des données de ses membres et afin de les transmettre à la station de base. Un protocole de routage hiérarchique se compose d'au moins deux phases : la phase de clustering des nœuds (pour l'élection du ou des clusters head) et la phase de transmission de données (après leurs traitements éventuels). La technique du clustering permet de réduire la complexité du routage. LEACH (Low Energy Adaptative Clustering Hierarchy) est parmi les premiers protocoles de routage hiérarchique les plus répandus proposé pour les RCSFs.

Plusieurs variantes de LEACH ont été proposées pour minimiser les problèmes liés à la version originale et améliorer la performance du protocole.

3.6.4 **LEACH**

LAECH (Low Energy Adaptive Clustering Hierarchy) [49] est le premier et le plus populaire protocole hiérarchique, qui fonctionne par des tours. Chaque tour est composé de deux phases principales. La première phase ; appelée phase de configuration (setup-state), concerne la création des clusters et l'élection de leurs CHs. La seconde phase est la phase de routage (steady state) ; elle concerne la transmission de données vers la station de base. Le but principal de LEACH est de minimiser la consommation énergétique des éléments du réseau

3.6.4.1 La phase de configuration (setup)

Apres le déploiement des nœuds dans le champ de capture, Les CHs (Cluster Heads) sont générés aléatoirement. Dans la phase de configuration et de construction des clusters, les CHs informent leur voisinage de leur élection en utilisant la même puissance. En s'appuyant sur la puissance des signaux reçus, le nœud choisit le cluster le plus proche à rejoindre et renvoie un message au CH correspondant. Le CH alloue un intervalle de temps de communication pour chaque nœud membre dans le cluster en utilisant le protocole TDMA (Time Division Multiple Access). Afin de minimiser la consommation d'énergie, les nœuds peuvent ainsi mettre en veille leur système de communication en attendant leur tranche de temps. La méthode de sélection des CHs dans LEACH peut être exprimée comme suit : chaque nœud sélectionne un nombre entre 0 et 1 de façon aléatoire. Si la valeur de ce nombre est inférieure à la valeur d'un seuil calculé T (n), le nœud devient cluster head. T (n) est représenté par l'équation (3).

$$T(n) = \begin{cases} \frac{P}{1-P*[r \bmod \left(\frac{1}{P}\right)]} & n \in G \\ 0 & autre \end{cases} \qquad (3)$$

3.6.4.2 La phase de transmission des données (Steady state)

L'opération de transmission des données est partitionnée en trames (figure 3-5), les nœuds ne transmettent leurs données au CH qu'une seule fois pendant la durée de chaque trame. Le temps alloué à chaque nœud dépend du nombre de nœuds sous un CH. Après l'expiration du TDMA et la réception de toutes les données par le CH, ce dernier effectue des traitements sur les données reçus (compression, agrégation,..), puis transmet le résultat directement à la station de base en utilisant le protocole CSMA/CD.

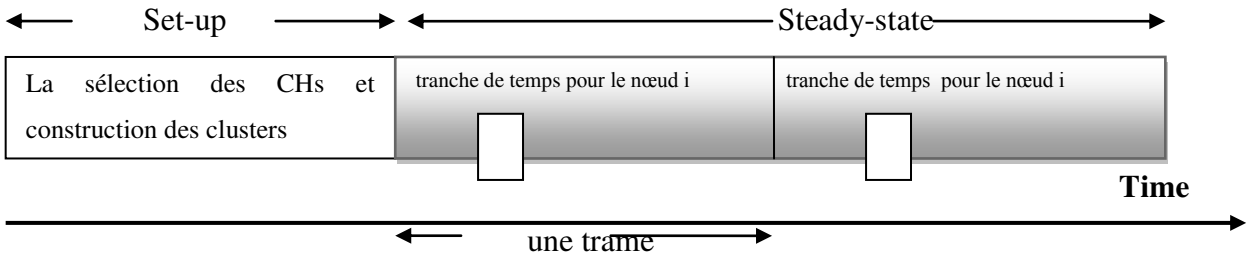

Figure 3-5 Les opérations de LEACH représentées dans le temps.

3.7 **Clustering**

Le clustering est une approche parmi les plus importantes pour organiser la structure logique des réseaux. Il permet la formation d'une architecture spécifique pour améliorer l'utilisation des ressources et augmenter la performance du réseau. Plusieurs protocoles de routage utilisent cette approche, elle est considérée comme étant une solution parmi les plus populaires et les plus importantes proposées pour la conservation d'énergie et maximiser la durée de vie des réseaux de capteurs sans fils. Elle a aussi prouvé son efficacité est présente de nombreux avantages en terme d'agrégation des données, passage à l'échelle et stabilité des réseaux. En outre, elle améliore les performances des algorithmes de routage. Son principe est basé sur le partitionnement du réseau en plusieurs couches ou plusieurs groupes appelés clusters.

3.7.1 **Définition**

Le clustering est un arrangement des nœuds capteurs dans des groupes appelés clusters. Pour chaque cluster, un leader doit être sélectionné pour devenir le cluster head (CH), il gère la communication des données collectées par les membres de son cluster. Cet arrangement et cette sélection du CH sont basés sur des métriques telles que la distance et l'énergie. La construction des clusters et la sélection des CHs peuvent se faire de manière centralisée ou distribuée. Dans un réseau regroupé en clusters les nœuds ordinaires détectent des événements dans leurs champ de capture et collectent des informations utiles concernant ces évènements afin de les transmettre au cluster head, ce dernier les agrége et les transmet à la station de base (SB).

3.7.2 **Caractéristiques du clustering**

Afin d'étudier et classifier les protocoles de clustering proposés, il faut étudier leurs caractéristiques. Selon [50], on peut distinguer trois types de caractéristiques : celles du cluster, celles du cluster head et enfin, celles du processus de clustering lui-même. La figure 3-6 présente les différentes caractéristiques de clustering.

3.7.2.1 Les caractéristiques du cluster

En fonction des spécifications du cluster, quatre propriétés sont définies pour caractériser un cluster : le nombre de clusters, la taille du cluster, la communication intra-cluster et la communication inter-cluster. Une brève explication de chacune de ces caractéristiques est présentée ci-dessous :

- Le nombre de clusters : Le nombre de clusters formés dans un réseau hiérarchique dépend du protocole de clustering exécuté. Il peut être constant (prédéfini) ou variable. En général dans les approches où la sélection des CHs est aléatoire, le nombre de clusters est variable.
- Taille du cluster : la taille du cluster est définie par le nombre de nœuds appartenant au cluster. Elle définit aussi la distribution de charge dans le réseau. La taille du cluster est définie en fonction de la manière de la formation des clusters ou de la façon de la division du réseau utilisées par l'algorithme de clustering exécuté. Dans un même réseau ces tailles peuvent être différentes.
- Communication intra-cluster : La communication intra-cluster est la manière dont sont effectuées les communications entre les nœuds au sein d'un même cluster. Elle est définie sur la base de l'algorithme de clustering, elle peut être directe ou multi-sauts. Dans la communication multi-sauts les nœuds du cluster envoient leurs données collectées au CH à travers des nœuds intermédiaires. Dans certains algorithmes de clustering et lorsque le nombre de CH est faible et la taille des clusters est grande, une communication multi-sauts entre le CH et les membres peut être nécessaire pour éviter les interférences causées par les transmissions à longues distances.
- Communication inter-cluster : la communication inter-cluster concerne la manière de communication entre les CHs et la SB. Elle peut être directe ou multi-sauts. L'approche multi-sauts est un mécanisme qui est utilisé dans les RCSFs lorsque la taille du réseau est grande. Habituellement, dans les réseaux à petite échelle et les approches traditionnelle, on utilise surtout la communication directe entre les CH et les BS.

3.7.2.2 Les caractéristiques des CHs

L'élection des CHs est une tache fondamentale dans tout algorithme de clustering. Les CHs élus ont un effet substantiel sur la performance de l'algorithme de clustering. Nous listons ci-dessous quelques caractéristiques des CHs :

- Mobilité : les CHs peuvent être fixes ou mobiles. Les CHs mobiles peuvent se déplacer dans un rayon dont la distance est limitée. En général, le processus de gestion de la topologie des CHs mobiles est plus difficile à mettre en œuvre que celui exécuté dans un réseau avec des CHs stationnaires.
- Type de nœuds : les nœuds capteurs déployés peuvent être homogènes ou hétérogènes. Dans le cas d'hétérogénéité, les nœuds ont différentes capacités en termes d'énergie, de capture, de traitement et de stockage. Dans ce cas ; le CH élu a plus de capacités que les autres nœuds. Dans le cas où tous les nœuds sont homogènes, les CHs sont choisis parmi les nœuds réguliers.
- Rôle : selon l'algorithme de clustering exécuté, les CHs élus peuvent jouer différents rôles dans le réseau. Un CH peut agir comme un simple nœud relais, c'est le cas, par exemple pour le contrôle du trafic de données et de la synchronisation comme il peut s'agir d'un leader qui effectue des opérations importantes d'agrégation et de fusion de données.

3.7.2.3 Les caractéristiques des processus de clustering

Le processus de clustering comprend deux tâches principales qui sont la formation des clusters et la sélection des CHs. Chaque protocole de clustering suit son propre processus. Nous présentons ci-après quelques caractéristiques des processus de clustering.

- Méthode : Un algorithme de clustering peut être distribué ou centralisé. Puisque les RCSFs sont en général des réseaux à grande échelle, les approches distribuées ont gagné plus de popularité que les approches centralisées.
- Objectifs : La construction des clusters des nœuds dans les RCSFs peut avoir des objectifs différents. Parmi ceux-ci : la tolérance aux fautes, le passage à l'échelle, la stabilité du réseau, l'équilibrage de charges et d'autres qui seront présentés dans la section suivante.
- La sélection des CHs : Chaque algorithme de clustering propose son propre mécanisme d'élection des CHs. Cependant et de façon générale, les algorithmes d'élection des CHs peuvent être classés en trois catégories : méthodes prédéfinies, aléatoires et basées sur des métriques. Dans les approches prédéfinies, les CHs sont élus avant le déploiement des nœuds dans le champ de capture. Dans les approches aléatoires, les CHs sont sélectionnés au hasard et les algorithmes basés sur des métriques les sélectionnent en fonction de certaines de leurs caractéristiques comme l'énergie résiduelle et leur éloignement de la SB.

- La complexité de l'algorithme : La complexité de l'algorithme prend en considération la manière dont un algorithme converge. Cette complexité est calculée en fonction des spécifications du réseau comme le nombre de CHs. Certains algorithmes convergent en temps variable alors que d'autres convergent à instants constants, quelles que soient les spécifications du réseau.
- La nature du clustering : De nombreux algorithmes de clustering ont été proposés pour les RCSFs dans la littérature. Un petit nombre de ces approches, appelées réseaux réactifs, sont basées sur la méthode data centric. La plupart des approches proposées sont proactives et ne supportent pas la réactivité. Il existe aussi les approches hybrides.
- Le dynamisme des clusters : un algorithme de clustering peut être dynamique ou statique. Le dynamisme peut être considéré au niveau des nœuds capteurs ou à celui de la station de base. Les algorithmes dynamiques agissent en temps réel. Les CHs sont choisis en fonction des conditions actuelles du réseau.

3.7.3 **Objectifs du Clustering**

L'établissement de l'architecture hiérarchique et le clustering des nœuds dans les réseaux de capteurs sans fils sont effectués avec un certain nombre d'objectifs. La conservation d'énergie est l'objectif le plus important et le plus commun entre tous les algorithmes de clustering dans les RCSFs. Dans la suite, nous expliquerons brièvement les objectifs du clustering les plus recommandés.

3.7.3.1 Passage à l'échelle

Le nombre de nœuds capteurs dispersés dans un champ de captage peut être de l'ordre de centaines, des milliers voire des millions d'unités. L'architecture hiérarchique permet le passage à l'échelle afin de créer de grands réseaux répondant aux besoins des applications. Ceci est réalisé en divisant les réseaux en zones ou couches virtuelles et dans chacun des groupes, des nœuds (clusters) sont constitués. Les nœuds des différents clusters peuvent communiquer entre eux grâce aux CHs. Le résultat est l'augmentation de la taille du réseau et la réduction significative du nombre de chemins de routage vers la station de base [51].

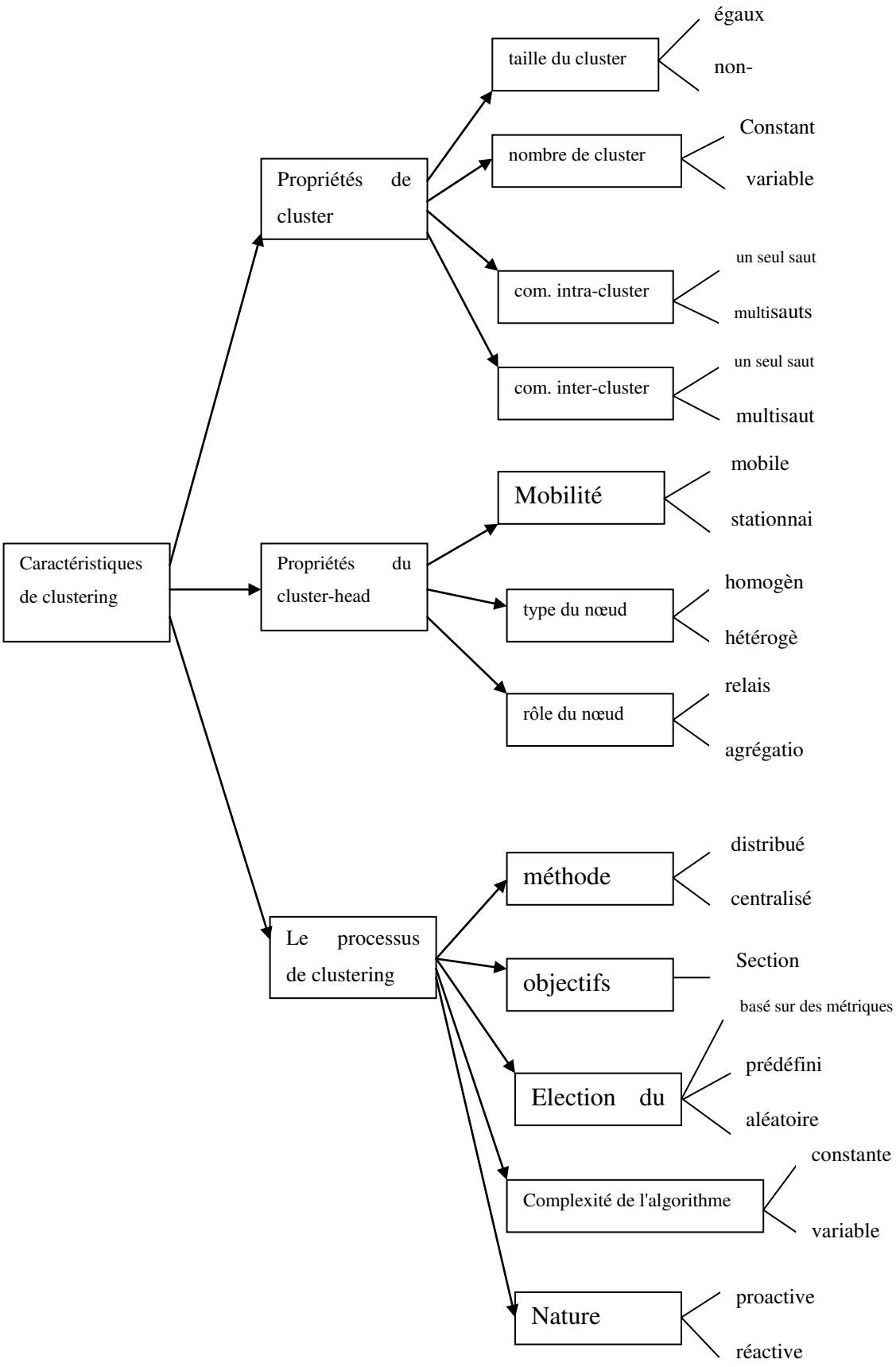

Figure 3-6 Les caractéristiques de clustering

3.7.3.2 Tolérance aux pannes

Les réseaux de capteurs sans fils sont dispersés généralement dans des environnements hostiles et inaccessibles pour l'être humain, donc la tolérance aux fautes est une caractéristique critique pour ces réseaux où l'échec de certains nœuds ne devrait pas affecter le travail du réseau. Le clustering des nœuds est une approche efficace pour rendre les RCSFs plus sûrs et plus tolérants [52].

3.7.3.3 L'agrégation des données

Etant donnée qu'une grande quantité des données dans les réseaux de capteurs sans fils peut être sensiblement la même (des capteurs peuvent collecter les mêmes informations), l'agrégation des données est une approche efficace pour éviter de retransmettre des données redondantes dans le réseau. La mise en œuvre du clustering permet de fusionner les données au niveau du CH de chaque cluster, ceci réduit considérablement la charge totale du réseau ; ce qui permet plus d'économie d'énergie [53].

3.7.3.4 Stabilité du réseau

Dans un réseau hiérarchique, où les nœuds sont regroupés en clusters, chaque CH possède des informations sur les membres de son cluster comme l'énergie et la position. Il faut établir une bonne gestion du réseau en termes de clustering, de sélection des CHs et pour la communication entre les différents nœuds pour assurer la stabilité du réseau.

3.7.3.5 Equilibrage de charge

Le clustering doit assurer le traitement des données et la communication intra-nœuds avec une faible consommation d'énergie ; cependant, les CH sélectionnés pendant la phase de clustering sont responsables des transmissions des données collectées par les membres vers la SB après son agrégation. Cette opération consomme beaucoup d'énergie des CHs. Ainsi, il est préférable de faire pivoter ce rôle parmi tous les nœuds du réseau. L'utilisation de l'équilibrage de charge permet d'obtenir un réseau plus économe en énergie [54].

3.7.3.6 Maximisation de la durée de vie du réseau

Comme mentionné précédemment, le principal défi dans les RCSF est de prolonger autant que possible la durée de vie du réseau. Une méthode de clustering qui satisfait les objectifs cités peut prolonger la durée de vie du réseau.

3.7.3.7 Réduction les délais du routage

Dans certaines applications des RCSFs, spécialement dans les réseaux de capteurs sans fil multimédia. Il importe de garantir une date limite (time_out) stricte pour le routage de données, sinon celle-ci seront obsolètes et l'opération aboutirait à des dommages importants. C'est le cas, par exemple, des systèmes temps réels tels que le e-health. Par conséquent, réduire un délai de routage est un autre défi pour répondre aux exigences de qualité de service du réseau [55]

3.7.3.8 Evitement des collisions

L'évitement de collision dans les RCSFs est une tâche importante lors des transmissions des données, car chaque collision peut causer la perte de certains paquets et il faut les retransmettre à nouveau. Cela augmente la consommation d'énergie dans le réseau et ne convient pas aux contraintes imposées. Lors du clustering des nœuds, l'utilisation de certains protocoles de couche MAC, tels que TDMA (Time Division Multiple Access), permet de réaliser des transmissions sans collisions [56].

3.8 **Les algorithmes de clustering**

L'établissement de l'architecture hiérarchique et la formation des clusters dans les RCSFs est un domaine de recherches très actif. L'intérêt porté aux techniques de clustering augmente au fils des années vu qu'elles ont prouvé leur performance dans la minimisation de la consommation d'énergie et la prolongation de la durée de vie du réseau. Les algorithmes de clustering développés pour les réseaux de capteurs sans fils prennent en compte la contrainte d'énergie. En effet, ces protocoles tentent de conserver le maximum d'énergie afin de prolonger la durée de vie du réseau. Malgré les performances sans cesse accrues de ces protocoles, les chercheurs essaient toujours d'apporter des améliorations pour obtenir des gains meilleurs en termes de conservation d'énergie. Dans cette section, nous présentons une vue générale concernant l'état de l'art (de 2010 à 2016) concernant les algorithmes de clustering proposés pour les RCSFs et la manière de les classifications.

On trouvera dans [57] un état de l'art assez complet traitant des algorithmes de clustering. Dans cette étude, les auteurs résument les dernières recherches sur les protocoles de routage basés sur la construction des clusters et ils les classent en quatre catégories principales : clustering basé sur les données, clustering hiérarchique, clustering basé sur la localisation et clustering basé sur la qualité de service (QoS). Les auteurs discutent de l'influence du placement des nœuds sur le fonctionnement et la performance des RCSFs. A la fin de cette étude, les auteurs comparent les algorithmes discutés en fonction de la

taxonomie proposée et d'autres critères significatifs. Une autre étude est présentée dans [58] qui tente de décrire et d'analyser les derniers algorithmes de clustering pour les RCSFs en utilisant différentes métriques de mesure, telles que l'élection des CH, la manière de communication dans les cluster et la mobilité des nœuds. Les auteurs proposent également une taxonomie pour classifier les algorithmes de clustering étudiés. Les catégories suivantes sont proposées : clusters dynamiques, homogènes, hétérogènes, clusters basé sur la conservation d'énergie, clusters pour l'équilibrage de charge et clusters basés sur la mobilité des nœuds. Les auteurs présentent les avantages et les inconvénients des algorithmes et se basent sur des critères de performance. Des analyses quantitatives et qualitatives sont effectuées. Dans [59] Une étude détaillée sur les techniques de clustering pour les différents réseaux de capteurs coopératifs (RCSF, MANET et VANET) est présentée. Les auteurs discutent les dernières idées novatrices pour chaque type de réseau de capteurs coopératifs et classent tous les algorithmes discutés en sept catégories : optimisation du cluster, mobilité, K-sauts, Équilibrage de charge, conservation d'énergie, multicritères et accès. Une étude comparative entre les protocoles de clustering avec station de base mobile et les protocoles de clustering avec station de base statique est réalisée dans [60]. On trouve également, dans cette recherche, une analyse de différentes techniques de chargement des nœuds de capteurs dans le réseau. Les avantages et les inconvénients de chaque technique sont indiqués. Une étude de recherche sur les approches et les caractéristiques de clustering dans les RCSFs est présentée dans [61], les auteurs classent les différents algorithmes de clustering selon l'architecture du réseau centralisé ou distribué. Parmi les algorithmes discutés LEACH, SOP, SEP, HEAP et WSNCBC, LEACH-C, BCDCP et OLE. A la fin de leur article, les auteurs ont présenté les avantages et les inconvénients des protocoles entrant dans chaque catégorie. Un autre état de l'art sur le clustering pour les RCSFs hétérogènes est présenté dans [62] où certains protocoles à clusters hétérogènes sont décrits avec des analyses, on y trouve notamment les protocoles : AD-LEACH, DEEC, Modified DEEC, MG-LEACH et autres. Une comparaison est effectuée entre ces approches. À la fin de cette étude, les auteurs discutent de l'amélioration obtenue par chaque approche. Dans [63], un bref aperçu sur les protocoles de routage basés sur la construction des clusters est présentée. Les algorithmes discutés sont classés selon le type de nœuds déployés. Les auteurs déterminent deux types de nœuds pour les RCSFs : les RCSFs statiques où les nœuds sont fixes et les RCSFs dynamiques où ces derniers sont mobiles. Les protocoles étudiés sont comparés en prenant en considération plusieurs critères. Dans [64], les auteurs donnent un aperçu sur les mécanismes qui fournissent ou

améliorent la propriété de tolérance aux pannes dans les réseaux de capteurs sans fils tels que les mécanismes de routage et clustering pour conserver l'énergie des capteurs et effectuer l'agrégation et la compression des données. Les auteurs proposent une nouvelle classification basée sur la taille du réseau car la performance de la majorité des mécanismes dépend de la gestion de la taille du réseau en termes de zone géographique couverte et du nombre de nœuds. Une étude limitée sur les algorithmes de clustering qui utilisent les techniques méta-heuristiques est présentée dans [65], où différentes méthodes méta-heuristiques sont analysées telles que PSO, Optimisation par Colonie de Fourmis (ACO : Ant Colony Optimisation), Algorithmes Génétiques (GA), (HS), et (SA), les versions d'optimisation et les combinaisons de ces techniques sont aussi discutées. A la fin de cette recherche les avantages et les inconvénients de chaque méta-heuristique sont présentés et une comparaison entre les algorithmes étudiés est effectuée. Plusieurs états de l'art sont discutés concernant les améliorations du protocoles de base LEACH Le tableau 3-1 en présente une synthèse rapide.

Tableau 3-1 Les derniers états de l'art sur le clustering dans les RCSFs

Ref.	Year	Classification
Maimour et al.	2010	Proactive, reactive
Lotf et al.	2010	Basé sur LEACH
Xu and Gao	2011	Basé sur LEACH
Joshi and Lakshmi Priya	2011	Basé sur LEACH
Wei et al.	2011	Predefine, adaptatif
Liu	2012	La manière de clustering, la manière de transmission des données
Aslam et al.	2012	Basé sur LEACH
Kumar et al.	2013	Basé sur la logique floue
Subhai et al.	2013	Basé sur les réseaux neurones
Jindal and Gupta	2013	Basé sur LEACH
Padmavati and Aseri	2014	clustering basé sur la mobilité de la station de base
Triana Mugia et al.	2014	Les protocoles de sécurité basé sur LEACH
Mehdi Afsar et al.	2014	Taille du cluster: égaux, non égaux
M.Di Francesco et al.	2014	Routage plat direct, routage par intermédiaire

S.R Jain and N.V Thakur	2015	Réseau statique et réseau dynamique
Samira Chouikhi et al.	2015	Fault tolerance scheme in small WSN, Fault tolerance scheme in large WSN (network size)
Twinkle and Nihar	2015	Clustering des réseaux hétérogènes
Zanjireh and Larijani	2015	Clustering centralisé, clustering distribué
Hifzan and Narenda	2015	Station de base statique et clustering avec station de base dynamique
Victor Sucasas et al.	2016	optimisation du cluster, mobilité, K-sauts, Équilibrage de charge, conservation d'énergie, multicritères, basé sur l'accès
Dharmanshu et al.	2016	Les techniques méta-heuristique PSO, ACO, GA, HS, SA,
Asim Zeb et al.	2016	clusters dynamiques, homogènes, hétérogènes, clusters basé sur la conservation d'énergie, l'équilibrage de charge et la mobilité des nœuds
Gherbi et al	2016	clustering basé sur les données, hiérarchique, la localisation et sur la qualité de service (QoS)

3.9 **La taxonomie proposée pour les algorithmes de clustering**

Le clustering des nœuds capteurs est devenu une tâche très importante dans les protocoles hiérarchiques pour les réseaux de capteurs sans fil (RCSF) en raison de leur applicabilité dans de nombreux domaines : militaire, villes intelligentes, médicales, commercial et industriel. Comme nous avons vu dans la section précédente de nombreuses études et recherches sur les algorithmes de clustering pour les RCSF ont été menées beaucoup ont abouti à des mises en œuvre pratiques. De même, plusieurs taxonomies sont proposées. Dans cette section, nous présentons notre propre essai de classification. Les critères principaux de classification que nous avons pris en considération concernent les paramètres du réseau de déploiement et les paramètres des nœuds capteurs déployés. Ainsi, nous avons déterminé les six catégories suivantes : les algorithmes de clustering basés sur les types de nœuds (homogènes ou hétérogènes), la méthode de clustering (centralisée ou

distribuée), la mobilité des nœuds (statique ou dynamique), la taille du réseau, le mode de transmission (un seul ou multi sauts), la taille des clusters. La figure 3-7 présente notre taxonomie et les algorithmes analysés pour chaque catégorie. Le tableau 3-2 résume le principe et l'objectif de chaque algorithme.

3.9.1 **Algorithmes de clustering selon le type de nœuds**

Deux types de nœuds sont définis pour cette classe : homogène et hétérogène. Dans les réseaux homogènes, tous les nœuds ont les mêmes capacités en termes d'énergie, de traitement et de stockage. Par conséquent, les CHs sont sélectionnés de manière aléatoire à partir des nœuds réguliers. Alors que dans les réseaux hétérogènes, les nœuds sont de types différents et par conséquent, les CHs sont choisis parmi les capteurs qui présentent les meilleures capacités.

3.9.1.1 Cluster homogène

- ECWC (An efficient clustering method using weighting coefficients in homogeneous wireless sensor network): c'est une approche efficace de clustering pour les réseaux de capteurs sans fil homogènes utilisant des coefficients de pondération; l'algorithme se concentre sur l'énergie résiduelle de chaque nœud. ECWC donne une amélioration en termes de conservation d'énergie par rapport à LEACH et HEED [66].
- FLCHH (Fuzzy Logic Based Cluster Head Selection for Homogeneous Wireless Sensor Networks) : Il s'agit d'un protocole de clustering pour les réseaux de capteurs sans fil homogènes où la sélection des CHs est basée sur la logique floue avec trois autres métriques qui sont : la distance, le nombre total de nœuds dans un cluster et le flux de communication [67].

3.9.1.2 Cluster hétérogène

- MSEEC (Multi-level stable and energy-efficient clustering protocol in heterogeneous wireless sensor networks) : Il s'agit d'un protocole de clustering à plusieurs niveaux pour les réseaux de capteurs hétérogènes. C'est une extension du protocole SEEC, ou chaque cluster possède un nœud maitre plus puissant que les autres nœuds esclaves déployés d'une manière aléatoire dans le cluster. L'avantage des architectures multi-niveaux est la couverture des zones de détection distantes. Dans un niveau d'hétérogénéité, Chaque type de nœud son rôle et un nombre optimal de nœuds maitres sont disponible pour atteindre la consommation minimale d'énergie du réseau [68].

- E2HR (An Energy-Efficiet Heterogeneous Ring Clustering Routing Protocol for Wireless Sensor Networks) : Il s'agit d'un protocole de routage à clusters hétérogènes constitués sous forme d'un anneau. C'est un protocole qui minimise la consommation totale d'énergie et assure l'équilibrage de charge des clusters. Des messages relatifs sont implémentés sur la structure originale des messages de protocole de routage RPL. E2HR assure un meilleure performance globale par rapport à RPL [69].

3.9.2 **Algorithmes de clustering selon le type de réseau**

Dans les réseaux de capteurs sans fils, deux méthodes sont possibles pour la formation des clusters : les méthodes centralisées et les méthodes distribuées. Dans les méthodes centralisées, la station de base nécessite des informations globales (localisation et niveau d'énergie) des nœuds capteurs déployés dans le réseau. Dans les méthodes distribuées, les nœuds collaborent entre eux et ne nécessitent aucune information globale sur le réseau.

3.9.2.1 Clustering distribué

- LEACH (Low Energy Adaptive Clustering Hierarchy) : est le premier et le plus populaire protocole hiérarchique, qui fonctionne en utilisant la notion de tours. Chaque tour est composé de deux phases principales. La première ; appelée phase de configuration (setup-state), concerne la création des clusters et l'élection de leurs CHs. La seconde phase est la phase de routage (steady state) et elle concerne la transmission de données vers la station de base [10]. l'avantage principal de LEACH est de minimiser la consommation énergétique des éléments du réseau [49].
- DCA (Distributed Clustering Strategies in Industrial Wireless Sensor Networks) : Il s'agit d'un algorithme de clustering distribué proposé pour les réseaux de capteurs industriels sans fil. La stratégie de clustering étudiée partitionne le réseau de capteurs en un nombre non fixe de clusters selon des critères concernant le réseau de communication, la topologie et la distribution des nœuds capteurs [70].

3.9.2.2 Clustering centralisé

- LEACH-C (Low Energy Adaptive Clustering Hierarchy Centralized) : Il s'agit d'une version centralisée de LEACH, il fonctionne aussi en tours. Chaque tour est divisé en deux phases : la phase de clustering et la phase de routage. Dans ce protocole, la station de base effectue le clustering des nœuds après la réception de leurs informations de localisation et de niveau d'énergie. LEACH-C a prouvé son

efficacité en termes conservation d'énergie notamment dans la phase de clustering qui prend en compte le niveau d'énergie des capteurs dans la selection des CHs, chose qui n'est pas prise en compte dans LEACH [71].

- GEAR-CC (A Centralized Clustering Geographic Energy Aware Routing for Wireless Sensor Networks). Il s'agit d'un protocole de clustering centralisé pour les RCSF basé sur l'énergie totale du cluster. GEAR-CC utilise l'algorithme de Dijikstra pour optimiser le routage dans la transmission des données [72].

3.9.3 **Algorithmes de clustering en fonction de la mobilité dans le réseau**

Selon la mobilité des nœuds, deux types de réseaux sont définis. Les réseaux statiques où tous les nœuds sont situés dans des endroits fixes durant toute la durée de vie du réseau. Dans les réseaux mobiles, les nœuds capteurs se déplacent au sein de la zone de capture. Pour réduire le problème de reclustering, les nœuds ayant des vitesses identiques sont regroupés pour former des clusters dans le but de minimiser la consommation d'énergie ce qui prolonge de la durée de vie du réseau.

3.9.3.1 Clustering dans les WSN mobiles

- CAGM (Clustering Algorithm Based on Glow worm Swarm Optimization) : C'est un protocole de clustering qui a été développé pour équilibrer la consommation d'énergie et maximiser la durée de vie des réseaux de capteurs sans fils. Le protocole CAGM utilise l'algorithme d'optimisation à essaim glowworm pour diviser le réseau en plusieurs clusters et sélectionner le meilleur CH pour chaque cluster en minimisant la consommation d'énergie. CAGM utilise la mobilité de la station de base pour équilibrer la consommation d'énergie entre les nœuds de capteurs [73].
- EMMS (Energy Management Algorithm with Multiple Sinks) : c'est un algorithme de clustering proposé pour la gestion d'énergie dans les RCSF en utilisant plusieurs stations de base mobiles. Le fonctionnement de EMMS est composé de deux tâches principales : le contrôle de la mobilité des stations de base et la gestion d'énergie du réseau. Le contrôle de la mobilité est réalisé comme suit : un tour fermé pour chaque SB est déterminé de sorte que la longueur de tous les tours fermés pour les stations de base soient presque égaux pour équilibrer la charge de travail entre les différentes SB. Puis l'emplacement des différentes SB est déterminé. Le protocole fonctionne en cycle comme le protocole LEACH, à la différence que chaque cycle

comprend trois phases : phase de calcul du temps, phase de collecte des données et phase de construction des routes [74].

3.9.3.2 Clustering pour les RCSF statiques

- EECBRP (Energy Efficient Cluster Based Routing Based Routing Protocol) : c'est un algorithme de clustering qui tente de conserver l'énergie des nœuds capteurs. La phase de clustering est initiée par la sélection des CHs, qui est basée sur certains paramètres comme l'énergie résiduelle et le nombre des nœuds voisins. Puis la formation de clusters en fonction de cette sélection des CHs. EECBRP suppose que les nœuds capteurs sont statiques et capables d'agréger des données, de calculer leur propre énergie résiduelle et de trouver leur emplacement géographique [75].
- DE-LEACH (Distance and Energy Aware LEACH) : C'est une version améliorée du protocole de routage LEACH. Il fonctionne en cycles. Le réseau est partitionné en deux zones en fonction de la distance entre les nœuds capteurs et la station de base. La sélection des CHs dans DE-LEACH est effectuée sur la base de la distance entre les nœuds et de leurs énergies résiduelles. Les CH effectuent l'agrégation des données et les transmettent à la station de base à travers d'autres CHs (communication multi-sauts inter-cluster) [76].

3.9.4 **Algorithmes de clustering en fonction de la taille du réseau**

Chaque protocole de routage hiérarchique est adapté à un certain type de réseau. En fonction de la taille du réseau, on distingue trois types : réseaux à grande échelle, à petite échelle et de taille moyenne. Le clustering est une solution importante pour le passage à l'échelle dans les RCSF.

3.9.4.1 Clustering pour les grands RCSF

- SODCC (Second-Order Data-Coupled Clustering for large Wireless Sensor Networks) : C'est un algorithme de clustering proposé pour les grands RCSFs, il utilise une combinaison d'algorithmes de traitement du réseau et de clustering pour obtenir une meilleure configuration. SODCC utilise un algorithme de clustering basé sur les statistiques des processus à second ordre et un algorithme de compression distribuée qui maximise la variance dans les composants résolus [50].
- JCR (Join Clustering and Routing protocol for reliable and efficient data collection in large-scale wireless sensor network) : C'est un protocole de clustering et de routage qui assure une collecte de données fiables et efficaces pour les RCSFs à grande échelle. Les résultats montrent que le routage multi-saut dans JCR peut

conduire à une sélection de CHs non équilibrée. Ensuite, la solution est fournie pour optimiser la durée de vie du réseau en prenant en compte le gradient des nœuds voisins pour générer une topologie adéquate. [77].

3.9.4.2 Clustering pour les petits RCSF

- Improved- LEACH : C'est un protocole de routage hiérarchique qui présente une version amélioré du protocole LEACH. L'idée principale de ce protocole est de sélectionner un vice-CH dans chaque cluster pendant la phase de clustering. C'est un assistant qui vise à prolonger la durée de vie du CH et, ainsi, celle du cluster en diminuant l'énergie consommé par les communications entre le CH et les nœuds membres [78].
- LEACH-G (an Optimal Cluster-heads Selection Algorithm based on LEACH) : C'est un algorithme de clustering optimal pour la sélection des CHs, il se base sur le protocole LEACH. LEACH-G calcule le nombre optimal des CHs en se basant sur le modèle énergétique pour LEACH [79].

3.9.5 **Algorithmes de clustering selon le mode de transmission**

La communication entre les nœuds dans les RCSF peut être faite en un seul saut ou par multi sauts. Dans le routage à un seul saut, les nœuds transmettent leurs données directement au CH. Cependant, dans le routage multi-sauts, les nœuds membres transmettent leurs données au CH via des nœuds relais.

3.9.5.1 Clustering multi-sauts

- MRRCE (multi-hop routing algorithm using Steiner points for reducing energy consumption in wireless sensor networks) : C'est un algorithme de routage hiérarchique qui utilise le mode de transmission multi-sauts pour réduire la consommation d'énergie dans les RCSFs. Il se base sur l'énergie résiduelle et la distance pour la formation des clusters ainsi que pour la sélection des CHs. Dans MRRCE, les tours dans lesquels le premier nœud meurt et le dernier nœud meurt sont grandement augmentés. Dans les RCSF hiérarchique utilisant le routage multi-sauts, chaque CH peut avoir plusieurs chemins vers la BS, l'algorithme de routage MRRCE cherche le meilleur chemin entre CH et la SB. Cela évite de perdre de l'énergie supplémentaire dans des chemins inefficaces et procure, ainsi, ainsi une prolongation de la durée de vie du réseau [80].
- DAMHR (Delay Aware Adaptive Multi Hop Routing Protocol) : c'est un protocole de routage hiérarchique multi-sauts pour les RCSF. Il s'agit d'une approche

heuristique qui définit les chemins optimaux pour la transmission des données collectées par les nœuds capteurs, cela minimise l'énergie consommée. DAMHR est un protocole adaptatif aux délais de retard, il utilise un récepteur mobile pour la sélection optimale des chemins, ce qui augmente la quantité totale des données et réduit la consommation d'énergie [81].

3.9.5.2 Clustering à saut unique

- SCEEP (A Single-Hop Clustering and Energy Efficient Protocol for Wireless Sensor Networks) : C'est un protocole de clustering qui utilise la méthode de communication à un seul saut. il est principalement envisagé pour réduire les surcharges de communication et les réélections inutiles des CHs pendant les cycles [82].
- ICP (Instantaneous Clustering Protocol for Wireless Sensor Networks) : C'est un protocole de clustering proposé pour les RCSF, qui tient compte des contraintes des délais de transmission et d'énergie. C'est un protocole de clustering instantané qui regroupe les nœuds de capteurs en clusters et utilise la transmission parallèle à un seul saut. ICP a prouvé sa performance dans la conservation d'énergie dans les RCSF [83].

3.9.6 **Algorithmes de clustering selon de la taille du cluster**

Les protocoles de routage hiérarchiques pour les RCSF peuvent regrouper les nœuds capteurs en clusters de taille égale et non égale. Dans les protocoles à clusters de tailles égales, tous les clusters ont la même taille, tandis que dans les protocoles à clusters inégaux, les clusters ont des tailles différentes. En général, des protocoles de clustering inégaux sont utilisés pour équilibrer la charge entre les nœuds.

3.9.6.1 Clustering à clusters de taille égale

- DSBCA: C'est un algorithme de clustering proposé pour équilibrer la charge dans les RCSF. Dans ce protocole les clusters des nœuds sont formés en fonction de leur distance par rapport à la station de base et la densité de distribution des nœuds. Par conséquent, dans la distribution uniforme des nœuds, la taille des clusters augmente si la station de base est distante par rapport aux nœuds. Dans la distribution non uniforme, les rayons du cluster sont définis par les deux paramètres cités précédemment (distance et densité) [84].
- LCM (Link-aware Clustering) *:* il s'agit d'un protocole de clustering pour les RCSFdans lequel les CHs sont élus en évaluant l'état des nœuds. Les auteurs

proposent une métrique, appelée nombre de transmissions prédites (PTX) pour évaluer les conditions d'une demande d'un nœud capteur émetteur. Le PTX représente la capacité d'un nœud pour une transmission persistante vers un nœud voisin spécifique [85].

3.9.6.2 Clustering de taille inégale

- EDDUCA (Energy Degree Distance Unequal Clustering Algorithm): C'est un algorithme de clustering dont les tailles des clusters sont inégales qui vise à équilibrer la consommation d'énergie et à maximiser la durée de vie du réseau. EDDUCA utilise la méthode du "triangle de Sierpinski" pour diviser le réseau en clusters non égaux. Les résultats obtenus indiquent qu'EDDUCA peut équilibrer efficacement la consommation d'énergie et donc prolonger la durée de vie du réseau [86].
- EBCAG (Energy Balancing unequal Clustering Approach for Gradient-based routing) : Cette approche utilise le nombre de sauts pour contrôler la distribution des CHs. Chaque nœud maintient une valeur de gradient définie comme son nombre minimal de sauts à la BS. Les CHS candidats sont choisis au hasard pour chaque valeur de gradient. Les données recueillies à partir des membres du cluster suivent la direction du gradient descendant pour atteindre la BS. La taille d'un cluster est déterminée par la valeur de gradient de ses CHs qui est estimée sur la base des données reçues des nœuds avec un gradient plus élevé [87].

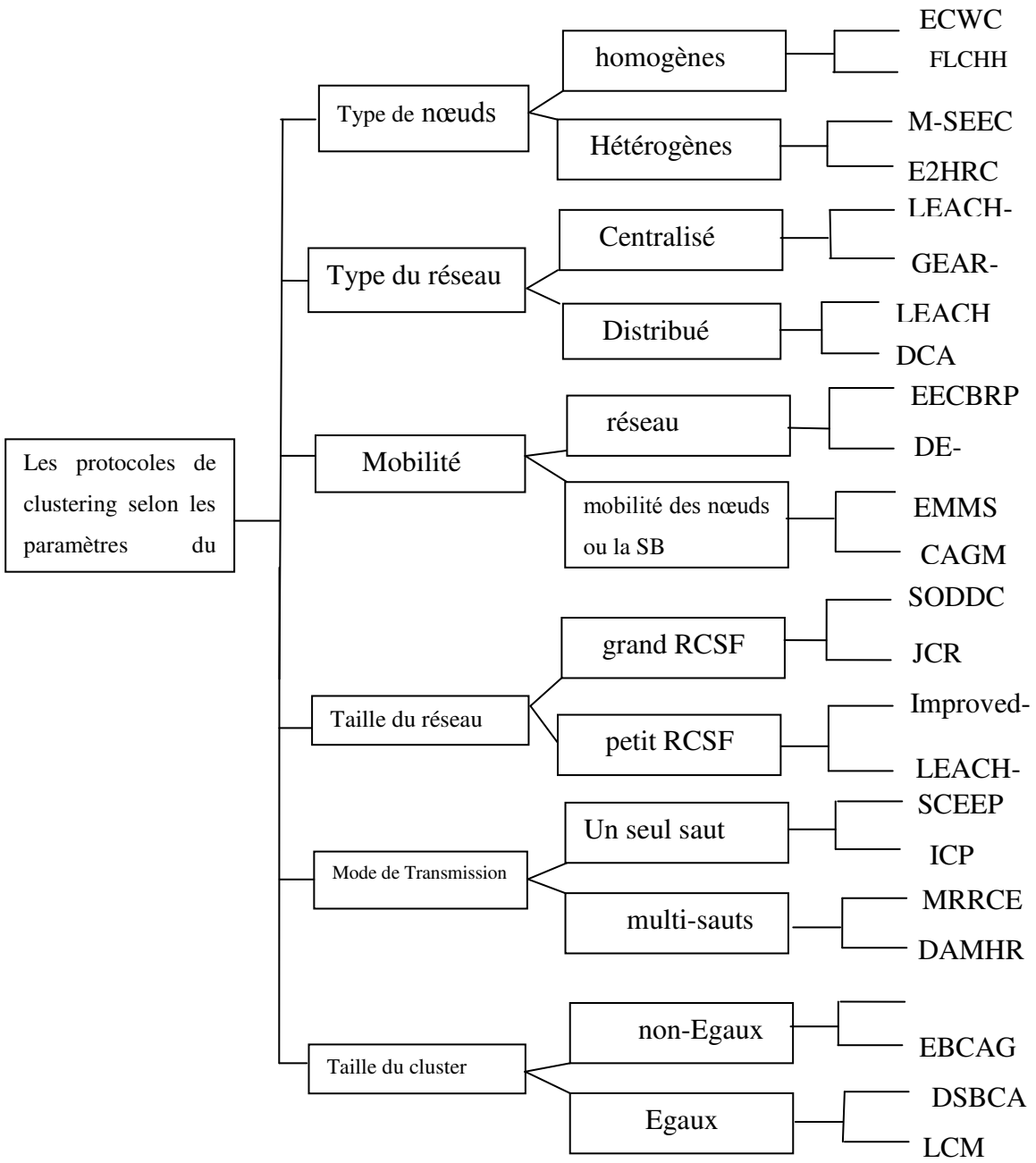

Figure 3-7 La taxonomie proposée pour les algorithmes de clustering

Tableau 3-2 Le principe et l'objectif des algorithmes de clustering

Protocole	Année	Référence	Principe	Objectifs
ECWC	2017	[13]	Clustering homogène	Maximiser la durée de vie du réseau et la conservation d'énergie.
FLCHH	2015	[14]	Clustering homogène	Maximiser la durée de vie du réseau et la conservation d'énergie.

M--SEEC	2014	[68]	Clustering hétérogène	Maximiser la durée de vie du réseau et la conservation d'énergie.
E2HRC	2017	[69]	Clustering hétérogène	Maximiser la durée de vie du réseau et la conservation d'énergie.
GEAR-CC	2013	[72]	Clustering centralisé	Maximiser la durée de vie du réseau et la conservation d'énergie.
DCA	2016	[70]	Clustering distribué	Maximiser la durée de vie du réseau et la conservation d'énergie.
EECBRP	2014	[75]	Clustering des RCSFs statiques	Maximiser la durée de vie du réseau et la conservation d'énergie.
DE-LEACH	2014	[76]	Clustering des RCSF statiques	Maximiser la durée de vie du réseau et la conservation d'énergie.
SODCC		[50]	Clustering des grands RCSF	Maximiser la durée de vie du réseau et la conservation d'énergie.
JCR	-	[77]	Clustering des grands RCSF	Maximiser la durée de vie du réseau et la conservation d'énergie.
Improved LEACH	2012	[78]	Clustering des petits RCSF	Maximiser la durée de vie du réseau et la conservation d'énergie.
LEACH-G	2013	[79]	Clustering des petits RCSF	Maximiser la durée de vie du réseau et la conservation d'énergie.
SCEEP	-	[82]	Communication à un seul saut	Maximiser la durée de vie du réseau et la conservation d'énergie.
ICP	2016	[83]	Communication à un seul saut	Maximiser la durée de vie du réseau et la conservation d'énergie.
MRRCE	2016	[80]	Communication multi-sauts	Maximiser la durée de vie du réseau et la conservation d'énergie.
DAMHR	2016	[81]	Communication multi-sauts	Maximiser la durée de vie du réseau et la conservation d'énergie.
CAGM	2015	[73]	Clustering des RCSF mobiles	Maximiser la durée de vie du réseau, la conservation d'énergie et l'équilibrage de charge
EMMS	2016	[74]	Clustering des RCSF mobiles	Maximiser la durée de vie du réseau et l'équilibrage de charge
EBCAG	2012	[87]	Unequal Clustering	Maximiser la durée de vie du réseau, équilibrage de charge
EDDUCA	2012	[86]	Unequal Clustering	Maximiser la durée de vie du réseau, équilibrage de charge

DSBCA	2013	[84]	Equal Clustering	Maximiser la durée de vie du réseau, équilibrage de charge
LCM	2013	[85]	Equal Clustering	Maximiser la durée de vie du réseau
LEACH	2000	[49]	Random Distributed Clustering	Maximiser la durée de vie du réseau
LEACH-C	2002	[71]	Centralized Clustering	Maximiser la durée de vie du réseau

3.10 **Conclusion**

Dans ce chapitre nous avons procédé à l'étude des réseaux de capteurs sans fils. Nous avons présenté une généralité sur les réseaux de capteurs sans fils, quelques définitions, l'architecture des RCSF et l'architecture d'un nœud capteur ainsi que certaines contraintes principales imposées pour la conception d'un réseau de capteurs sans fil. En fin de chapitre, nous nous sommes focalisés particulièrement sur les protocoles de routages hiérarchiques qui utilisent la technique de clustering pour établir les itinéraires minimisant la consommation d'énergie dans les RCSF. Nous avons proposé une taxonomie pour les protocoles de clustering selon les paramètres du réseau de déploiement avec une analyse de ces protocoles.

Chapitre 4 : Contributions

4.1 **Introduction**

Un réseau de capteurs sans fil est composé d'un grand nombre de nœuds capteurs dispersés dans une zone géographique limitée. Ces nœuds capteurs sont de petits dispositifs caractérisés par de petites batteries pour leur alimentation en énergie et une faible puissance de traitement et de stockage. Le défi majeur de ce type de réseaux est comment conserver le maximum d'énergie afin de prolonger la durée de vie du réseau étant donné que l'épuisement de l'énergie d'un capteur implique son inactivité dans le réseau. Les protocoles de routage hiérarchiques où les nœuds sont clustérisés sont considérés parmi les plus populaires et importantes solutions pour conserver l'énergie des capteurs et assurer ainsi la prolongation de la durée de vie du réseau. Une bonne stratégie de routage associée à la technique de clustering ont présenté de nombreux avantages en termes d'efficacité énergétique, d'agrégation des données, de stabilité de la topologie du réseau et de l'augmentation de sa durée de vie avec possibilité de passage à l'échelle. Le protocole LEACH et sa version centralisée LEACH-C sont parmi les premiers protocoles qui ont été proposés, étudiés et appliqués. Ils consistent à partitionner le réseau en clusters puis des nœuds cluster-heads (CH) sont constitués puis utilisés comme relais pour atteindre la destination en optimisant la consommation d'énergie. Dans ce chapitre, nous présentons nos deux propositions. La première est un algorithme de clustering pour les réseaux de capteurs sans fils basé sur le centre de densité des clusters nommé WDC-LEACH-C (Weighted Density Center-Low Energy Adaptive Clstering Hierarchy). Notre deuxième proposition concerne un protocole de routage multi sauts pour les RCSF basé sur les colonies de fourmis nommé Improved-ACORP (Improved Ant Colony Optimization Protocol). Nous commençons d'abord par présenter les spécifications générales utilisées dans les travaux connexes, ensuite nous présentons les détails de nos propositions avec les modélisations opérationnelles et formelles. A la fin de chaque contribution, nous présentons l'évaluation des performances.

4.2 **Motivation**

Dans notre travail, nous nous intéressons aux mécanismes de conservation d'énergie des nœuds capteurs qui sont déployés dans des environnements hostiles où les batteries sont non rechargeables.

Notre étude concernant l'état de l'art effectuée dans le chapitre 3 a montré qu'une grande puissance d'énergie est dissipée lors de l'opération de transmissions des données. Donc ; si la transmission des données n'est pas effectuée de manière fiable, l'épuisement énergétique sera un point critique et un problème majeur qui affecte la performance du réseau. La transmission non fiable des données induite par plusieurs sources telles que la redondance des paquets envoyés, la perte de données, les collisions et les interférences causées par les transmissions sur de longs chemins font qu'une grande partie de l'énergie des capteurs est inutilement gaspillée et par conséquent affecte la durée de vie du réseau. Pour améliorer la tâche de transmission des données, il faut construire des protocoles de routage efficaces évitant les sources de dissipation d'énergie. Comme nous avons vu dans le chapitre précédent ; plusieurs solutions sont proposées pour améliorer les performances des protocoles de routage.

Dans ce chapitre, nous présentons deux nouveaux protocoles de routage hiérarchiques. La première proposition nommée WDC-LEACH-C (Weighted Density Center- Low Energy Adaptive Clstering Hierarchy) qui est un protocole de routage hiérarchique, considère le mode de transmission à un seul saut dans lequel les messages sont directement transmis à partir des membres vers les CHs. La pierre angulaire de cette proposition est la sélection des CHs qui est basée sur le critère d'énergie et le centre de densité du cluster. Le protocole WDC-LEACH-C choisit le point le plus proche de tous les nœuds du cluster comme CH. cette idée réduit considérablement la distance de communication entre les nœuds ce qui évite la perte des données causée par des transmissions sur de longs chemins. La transmission à saut unique réduit le nombre de chemins à établir et diminue, ainsi, la consommation d'énergie en augmentant la durée de vie du réseau. Notre deuxième proposition concerne un protocole de routage hiérarchique intelligent pour les réseaux de capteurs sans fils homogènes auto-organisés basé sur les colonies de fourmis nommé Improved-ACORP (Ant Colony Optimization Protocol). L'idée principale de Improved ACORP est inspirée du comportement des colonies de fourmis réelles prenant en considération la distance entre les nœuds et le nombre de sauts pour établir les chemins. Improved ACORP introduit le critère d'énergie dans la sélection des chemins vers le CH et

il intègre, ainsi, une stratégie de gestion d'énergie des CHs. Une autre approche dénommée CHMS (Cluster Head Management Strategy) concerne une stratégie qui intervient lorsque le niveau d'énergie des CHs a atteint un seuil critique. La conservation d'énergie ainsi que la maximisation de la durée de vie du réseau ont été les buts principaux pour la conception de nos protocoles.

4.3 **Spécifications générales**

Dans cette section, nous spécifions les hypothèses, les modèles du réseau et de radio utilisés dans ce travail.

4.3.1 **Le modèle du réseau**

Dans notre travail, nous considérons un réseau de capteurs hiérarchiques à architecture clustérisée constituée de *n* nœuds capteurs homogènes et une station de base (*SB*). Ces nœuds capteurs et la station de base sont supposés être stationnaires une fois qu'ils sont déployés dans l'environnement. Le réseau est partitionné en groupes nommés clusters, et chaque cluster est géré par un cluster-head (*CH*). Initialement, tous les nœuds capteurs ont la même quantité d'énergie qui est limitée contrairement à la station de base qui n'est pas limitée en termes d'énergie, de mémoire et de puissance de calcul qui sont considérés robustes et dotés de ressources inépuisables. Le canal radio est symétrique de telle sorte que la consommation d'énergie de la transmission des données à partir du nœud *X* au nœud *Y* soit la même que celle de la transmission du nœud *Y* au nœud *X*. Chaque nœud capteur peut fonctionner soit en mode de détection pour contrôler les paramètres de l'environnement et les transmettre à la station de base, soit en mode CH pour recueillir les données, les compresser et les transmettre à la station de base. Nous supposons que pour minimiser l'overhead de communication dans le réseau, uniquement les CHs peuvent communiquer directement avec la *SB*, les autres nœuds ordinaires communiquent seulement avec les nœuds capteurs (et aussi avec leur CH) de leur cluster correspondant. La figure 4-1 représente ce modèle de réseau.

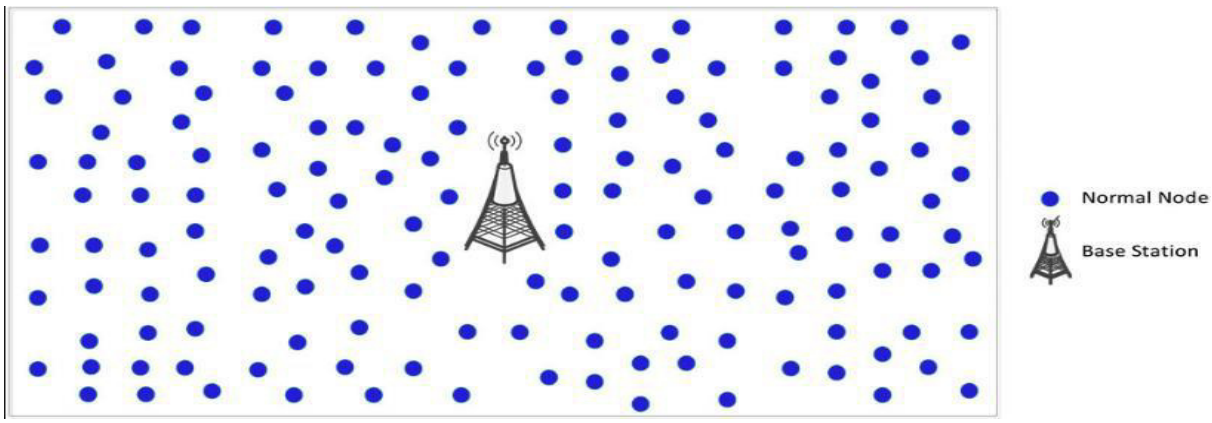

Figure 4-1 Le modèle réseau

4.3.2 **Le modèle radio**

Le modèle de consommation d'énergie, qui est utilisé dans le protocole proposé est présenté dans la figure 4-2. C'est le même modèle radio, qui est utilisé dans LEACH-C [71]. L'énergie dissipée par l'émetteur pour une transmission de k bits données est donnée par l'équation 4

$$E_{tx}(k,d) = \begin{cases} E_{elec} \times k + E_{amp} \times k \times d^2, & d < d_0 \\ E_{elec} \times k + E_{amp} \times k \times d^4, & d \geq d_0 \end{cases} \quad (4)$$

Ou

$E_{tx}(k,d)$: énergie dissipée par l'émetteur

E_{elec} : énergie des appareils électroniques

E_{amp} : énergie de l'amplificateur

d: distance de transmission

d_0: is the threshold distance that depends on the environment.

k: le hombre de bits transmis

Pour la réception de k bits de données par chaque nœud de capteur, l'énergie dissipée par le récepteur E_rx est donnée par l'équation 5

$$E_{rx}(k,d) = E_{elec} \times k \quad (5)$$

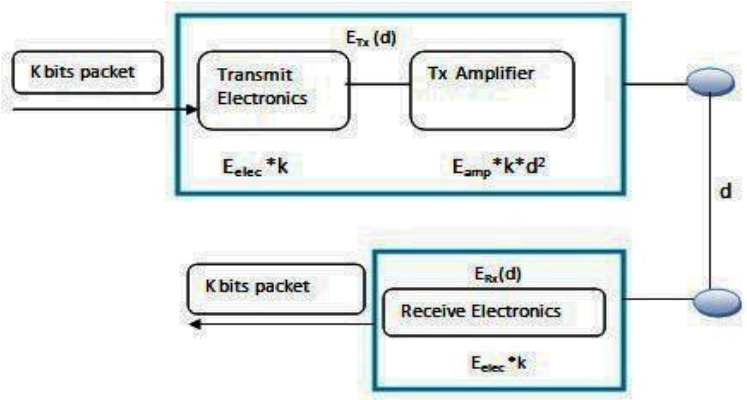

Figure 4-2 Le modèle radio [71]

4.4 **Protocole de clustering proposé WDC-LEACH-C**

Dans cette section nous présentons notre protocole proposé pour le clustering des nœuds dans les réseaux de capteurs sans fils. Nous commencerons par donner une vue d'ensemble sur l'algorithme ensuite nous le détaillerons.

4.4.1 **Vue d'ensemble du protocole proposé**

Le protocole proposé WDC-LEACH-C est un protocole de routage à architecture hiérarchique clustérisée basé sur LEACH-C. Il fonctionne en cycles ou chaque cycle est divisé en deux phases. La première est la phase de configuration (setup-state), cette phase consiste dans la formation des clusters, l'élection des CHs et l'établissement de l'ordonnancement de la transmission des nœuds dans chaque cluster. Cet ordonnancement est créé en utilisant le protocole TDMA (Time Division Multiple Access). Le protocole proposé effectue une élection des CH basée sur l'énergie initiale, l'énergie résiduelle et sur le centre de densité des nœuds (WDC). L'utilisation du WDC dans la sélection de CH rend ce dernier le nœud le plus proche de tous les nœuds membres du cluster. Par conséquent, les messages sont directement transmis depuis les nœuds membres du cluster vers le cluster head (CH) en minimisant le nombre de chemins de transmission tout en évitant les interférences causées par les transmissions sur de longues distances. Ce qui réduit, par voie de conséquence la consommation d'énergie. La figure 4-3 donne une comparaison entre une communication à plusieurs sauts et une communication à un seul saut avec l'utilisation de WDC en termes de chemins. Lorsque la phase de configuration est terminée, la deuxième phase qui concerne le routage (steady state) est entamée par la collecte des données à partir de l'environnement, ensuite l'agrégation et la transmission de ces données vers la station de base par les CHs. L'organigramme présenté figure 4.4 résume le déroulement général du protocole WDC-LEACH-C.

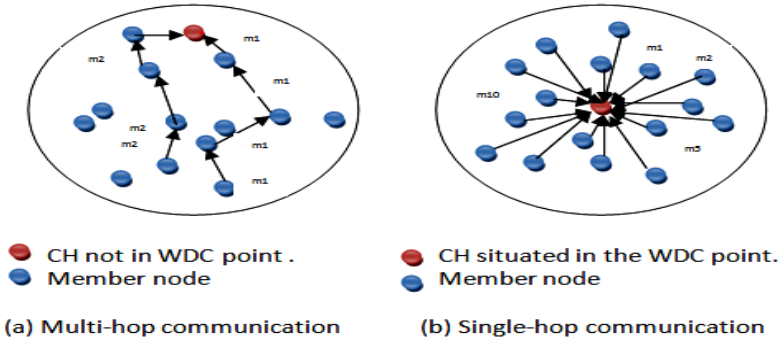

Figure 4-3 Le nombre de chemins dans la communication avec un seul saut et celui utilisant multi-saut avec la technique WDC

4.4.2 **Détails du protocole**

Notre protocole évolue en deux phases : la phase de configuration qui se déroulée selon trois étapes : initialisation, partitionnement du réseau et formation du cluster et sélection des CHs. La deuxième phase consiste dans le routage et la transmission intra et inter clusters.

4.4.3 **Phase de configuration (setup state)**

1. Initialisation

C'est l'étape de préparation des nœuds pour le clustering. Après le déploiement de nœuds dans le champ de capture désiré, chaque capteur envoie ses informations (informations de localisation et niveau d'énergie) à la station de base distante (ces données peuvent être s directement par la SB grâce au GPS).

2. Partitionnement du réseau et formation des clusters

Après avoir reçu ces informations, une division du réseau en régions égales est réalisée pour former des clusters équilibrés en termes de nombre de nœuds.

3. La sélection de CH

La sélection des CHs est la phase la plus importante dans notre proposition. Une fois les clusters formés, un CH doit être choisi dans chaque groupe. Les caractéristiques de ce nœud particulier sont sa grande puissance énergétique et le fait qu'il soit le plus proche de tous les autres nœuds du cluster. Ainsi, la sélection du CH se fait de la manière suivante :

- Calculer les distances entre chaque nœud i et tous les nœuds du même cluster, ensuite calculer la distance moyenne de ces distances pour chaque nœuds. On obtient, ainsi, une distance moyenne pour chaque nœud qui représente sa position par rapport aux autres nœuds membres de son cluster.

- On choisit la distance la plus courte parmi les distances moyennes calculées, qui représente le nœud le plus proche par rapport aux autres nœuds membres du cluster. C'est ce qu'on a appelé le nœud WDC.
- Pour une meilleure sélection de CH, on va choisir le nœud le plus proche au WDC et qui a le maximum de réserve énergétique. Pour cela un groupe G est créé qui contient un certain nombre de nœuds qui sont les plus proches du point WDC. Ce nombre est calculé sur la base d'une distance seuil déterminée.
- Puis, on choisit le nœud qui a la plus haute puissance d'énergie parmi les nœuds du groupe G

La procédure de sélection des CHs en utilisant la méthode WDC est présenté par les figure 4-5 et 4-6.

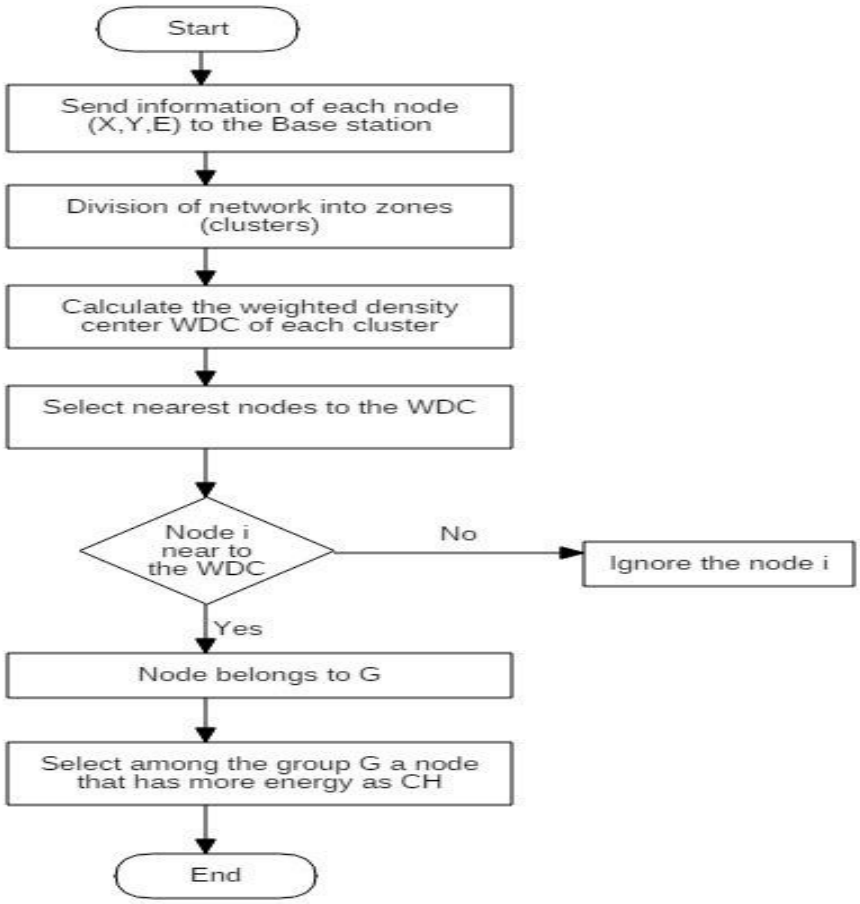

Figure 4-4 L'organigramme du protocole proposé.

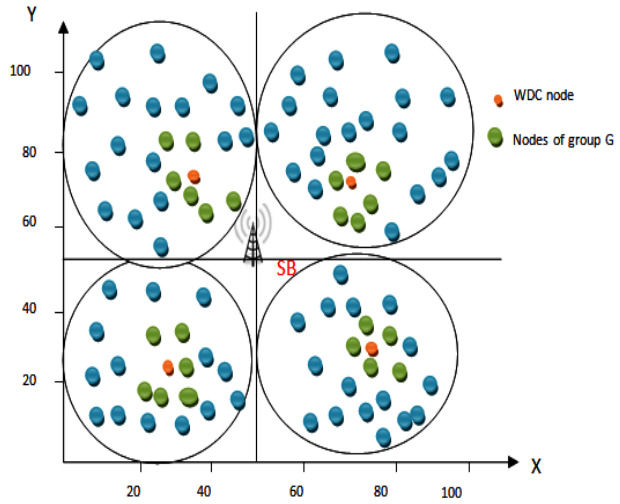

Figure 4-5 Calcule de WDC et groupe G

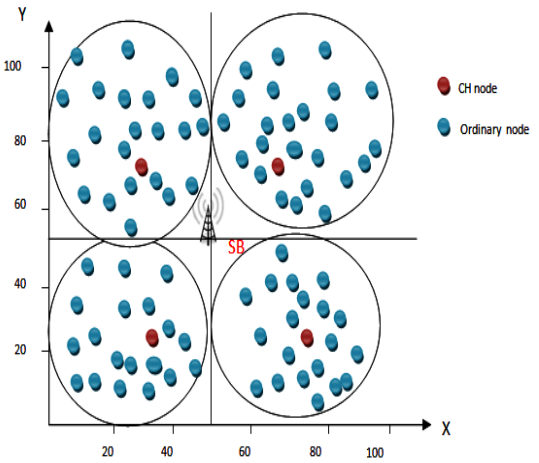

Figure 4-6 Sélection de CH

4. Phase de routage (steady state)

Le routage dans un protocole à architecture clustérisée est divisé en deux phases : le routage intra cluster c'est la communication entre les nœuds au sein d'un même cluster et le routage inter cluster qui concerne la communication entre les CHs et la station de base. Deux manières de communication sont possibles, la communication à un seul saut où les messages sont directement envoyés à la destination via des nœuds relais. Dans la technique WDC, dans notre proposition, la communication intra cluster à un seul saut est préférable parce que le CH est le nœud le plus proche de tous les membres du cluster, donc les distances entre le CH et les nœuds sont petites et par conséquent les communications sont effectuées par des courtes distances. Ce qui évite les interférences causés par les transmissions par longues distances et minimise le nombre de chemins pour l'envoi d'un message. L'acheminement des données inter clusters est effectué à un seul saut par les CHs directement vers la station de base.

4.5 La modélisation de notre proposition WDC-LEACH-C

La mise en œuvre finale d'une solution efficace d'un protocole de routage hiérarchique doit se faire en suivant des règles strictes de modélisation et de simulation. Ce sont là des difficultés qu'il est nécessaire de surmonter. La modélisation est devenue une tâche importante lors de la construction des réseaux de capteurs sans fils. Ces difficultés sont liés principalement au choix du protocole de routage ainsi qu'au nombre, parfois très important de nœuds déployés dans le réseau. Examiner toutes les possibilités peut être une tâche impossible vue leur nombre (explosion combinatoire). Il y'a lieu d'établir des modèles

après la proposition de la solution, de bien les analyser avant d'opiner pour la décision d'implémentation et de déploiement. La figure 4-7 indique quelques étapes pour construire une solution acceptable.

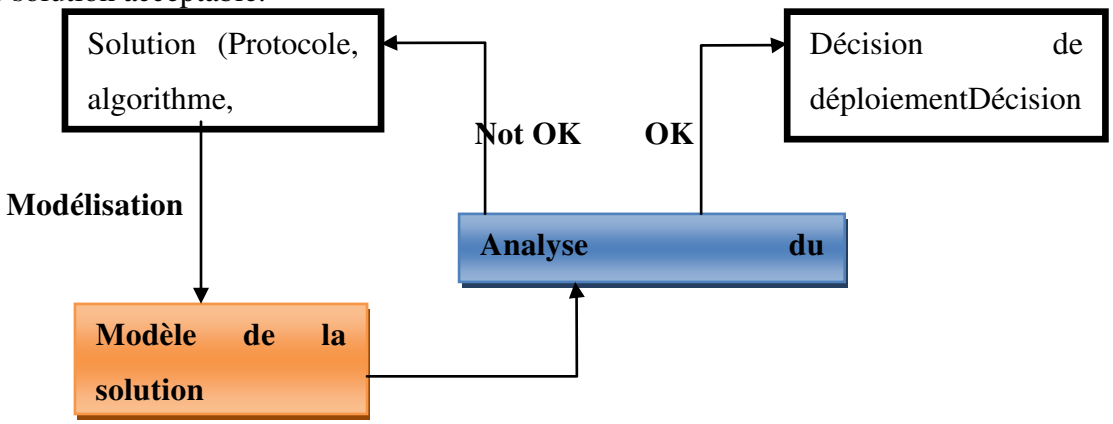

Figure 4-7 Les étapes suies pour construire une solution

La modélisation des données est une représentation abstraite de la structure, des relations, des noms et des formats de données pertinents. Un modèle est une représentation simplifiée d'une réalité. Il permet de capturer les éléments les plus importants pour répondre à un objectif défini. Notre étude tente de mettre en place un protocole de routage efficace qui assure la connectivité du réseau et la couverture du champ surveillé et assurer une délivrance fiable et rapide. Le défi le plus important est la consommation du moins d'énergie possible. Il existe plusieurs approches de modélisation. Il est nécessaire d'utiliser des techniques formelles et informelles pour saisir les diverses facettes du protocole proposé qu'on veut modéliser sans le dénaturer. Afin de modéliser nos propositions, nous avons utilisé trois approches de modélisation qui sont la modélisation objet avec UML, la modélisation opérationnelle par SMA et la modélisation formelle par les automates temporisées

4.5.1 **La modélisation Objet utilisant le langage UML**

4.5.1.1 Le langage UML [88]

UML est l'abréviation de Unified Modeling Language, c'est à dire langage unifié pour la modélisation. UML est un langage qui permet de représenter des modèles par un ensemble d'outils permettant d'exprimer des modèles en faisant abstraction de l'implémentation

4.5.1.2 Diagramme de Classe

Le diagramme de classe compte parmi les diagrammes les plus importants de toutes les méthodes orientées objet, il concerne la structure interne du système (figure 4-8).

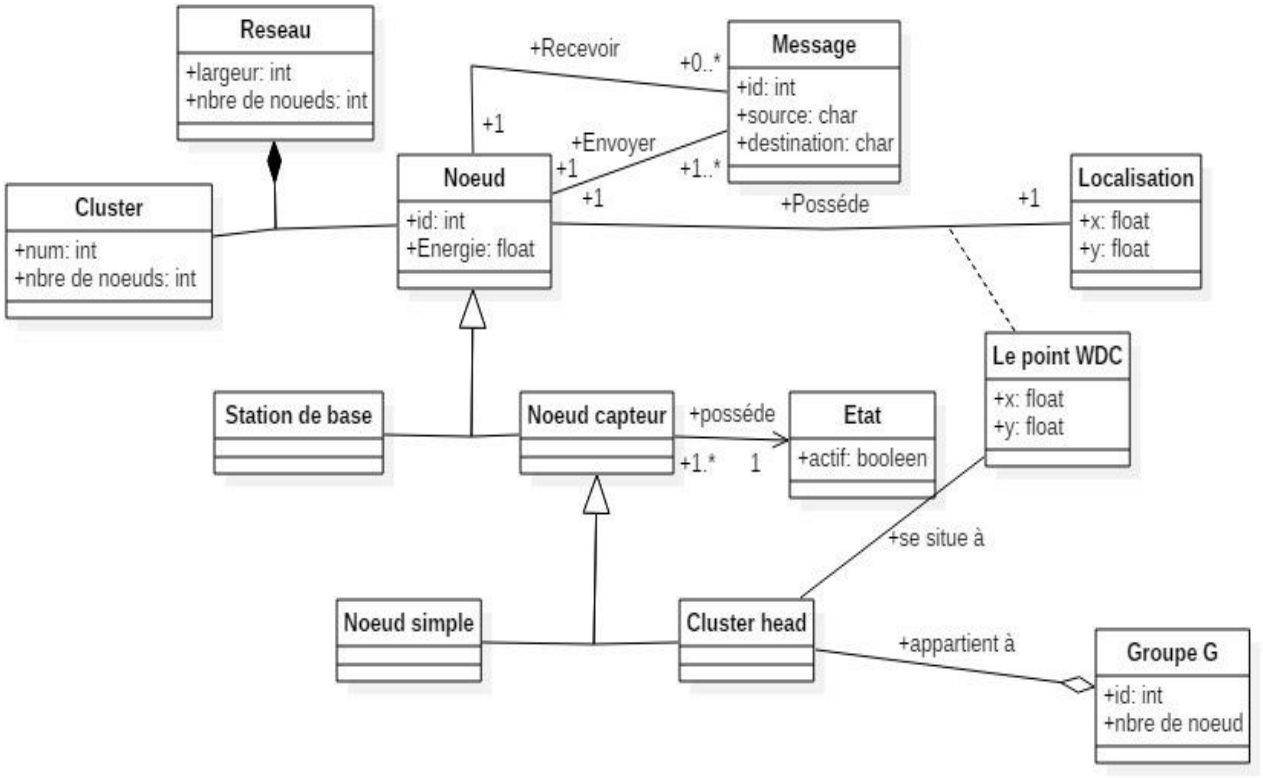

Figure 4-8 Le diagramme de classe du protocole WDC-LEACH-C

4.5.1.3 Les diagrammes des cas d'utilisation

Les diagrammes des cas d'utilisation montrent le système du point de vue acteurs. Il permet de recenser les grandes fonctionnalités des différents opérateurs dans le système.

a- Le diagramme de cas d'utilisation global

But : Le déroulement global du protocole

Acteur : Utilisateur.

Description du traitement nominal : L'utilisateur peut :

- Définir le modèle réseau et le modèle radio.
- Déployer le réseau.
- Simuler le réseau.
- Consulter l'état du réseau.

(voir figure 4-9)

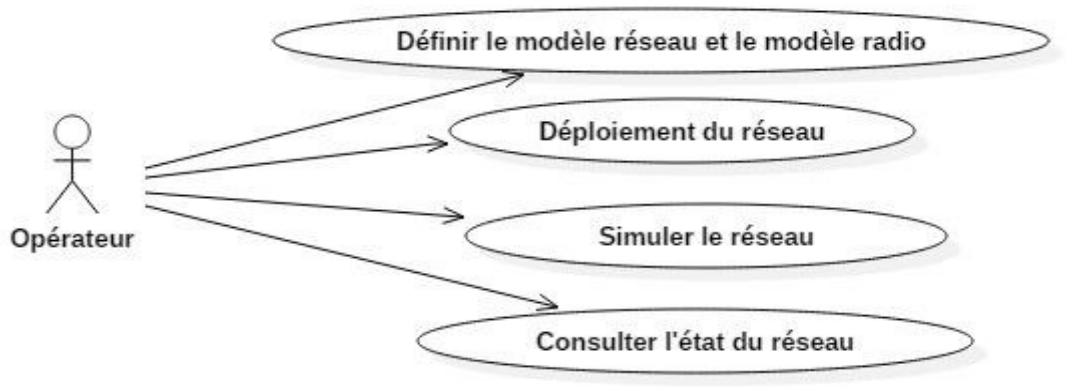

Figure 4-9 Diagramme de cas d'utilisation globale du protocole

b- Le diagramme de cas d'utilisation pour le clustering

But : le clustering des nœuds et la sélection du CH pour chaque cluster

Acteur : station de base.

Pré conditions : L'operateur doit Etre authentifié.

Description du traitement nominal : la station de base peut :

- Traiter les informations des nœuds.
- Diviser le réseau.
- Calculer le WDC de chaque cluster.
- Créer le groupe G.
- Gérer le groupe G.
- Sélectionner le CH à partir du groupe G.

(voir figure 4-10)

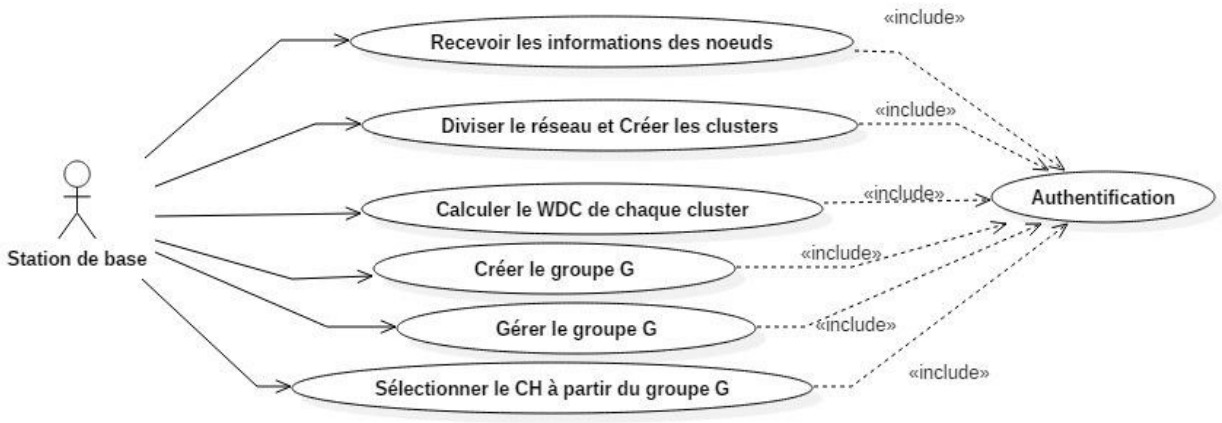

Figure 4-10 Diagramme de cas d'utilisation pour le clustering

c- Le diagramme de cas d'utilisation pour le routage inter-cluster

But : le routage inter-cluster des données

Acteur : station de base.

Pré conditions : L'opérateur doit être authentifié.

Description du traitement nominal : la station de base peut:

- Réception des données à partir des CHs.
- Traiter les données reçues.
- Envoyer les données traitées à l'utilisateur.
- Sélectionner des nouveaux CHs pour un nouveau cycle.

(voir figure 4-11)

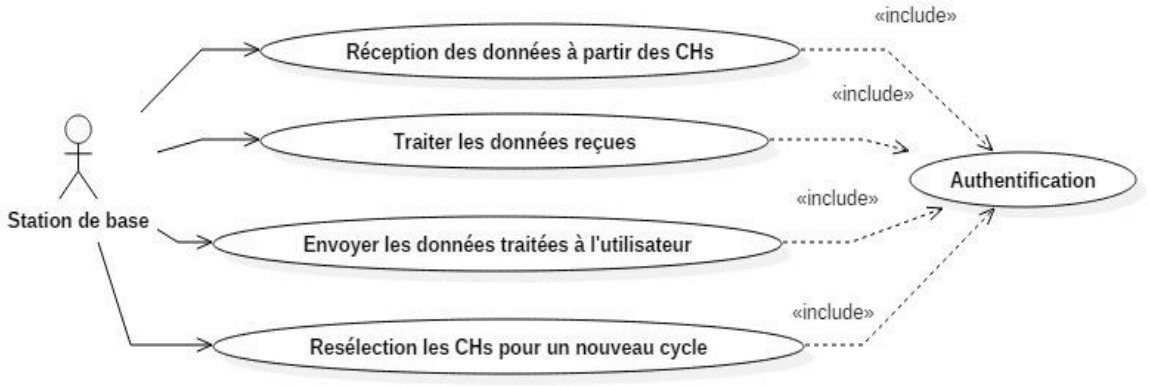

Figure 4-11 Diagramme de cas d'utilisation pour le routage inter cluster des données

d- Le diagramme de cas d'utilisation pour le routage intra-cluster des données

But : le routage intra-cluster des données

Acteur : cluster head

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : le cluster head peut :

- Recevoir les informations collectées par les nœuds.
- Agrégation des données.
- Envoyer les données à la station de base

(voir figure 4-12)

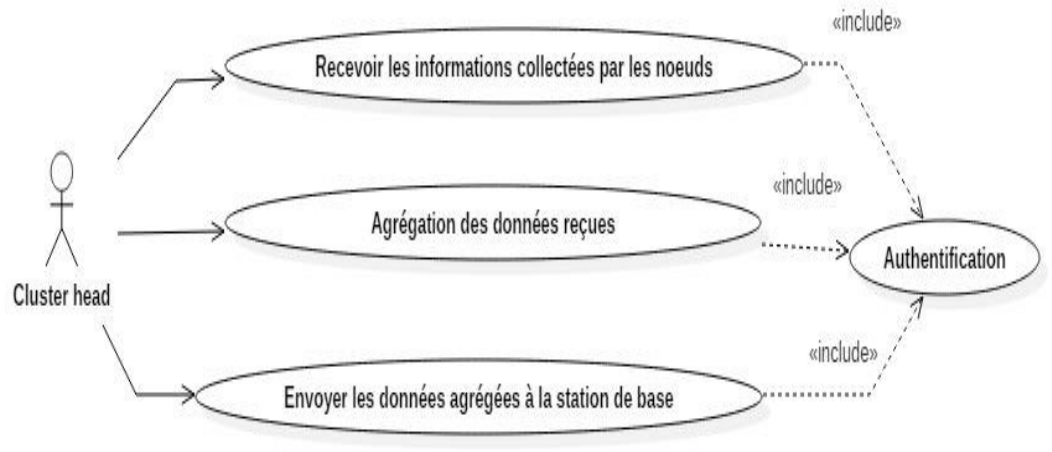

Figure 4-12 Diagramme de cas d'utilisation pour le routage intra-cluster des données

e- Le diagramme de cas d'utilisation pour les opérations d'un nœud capteur

But : définir les opérations d'un nœud capteur pendant le déroulement du protocole

Acteur : Station de base.

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : un nœud capteur peut :

- Envoyer les informations GPS et énergie à la SB.
- Capter les données de l'environnement.
- Collecter les données captées.
- Envoyer les données collectées au CH.

(voir figure 4-13)

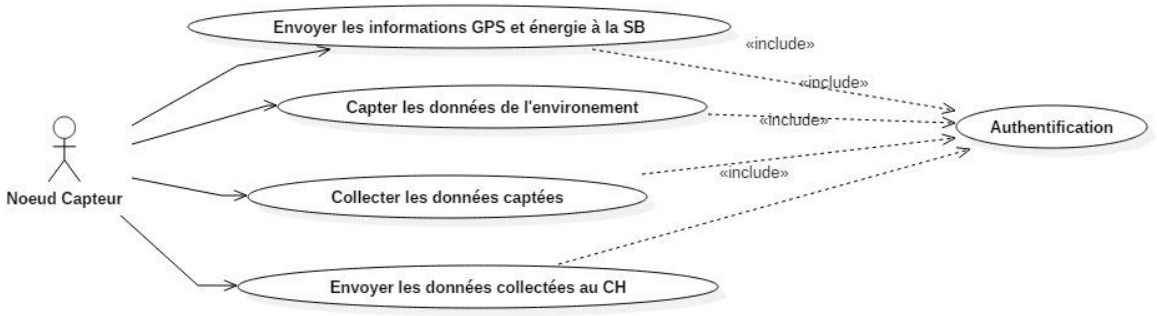

Figure 4-13 Diagramme de cas d'utilisation pour les opérations d'un nœud capteur

4.5.1.4 Diagramme de séquence

Le diagramme de séquence est un diagramme temporel qui indique les interactions entre plusieurs partenaires de communication. Il établit le pont entre le point de vue fonctionnel et statique. Il apporte un aspect dynamique au système. Dans notre protocole 3 entités sont en interaction qui sont : le nœud capteur, le cluster head et la station de base (Figure 4-14).

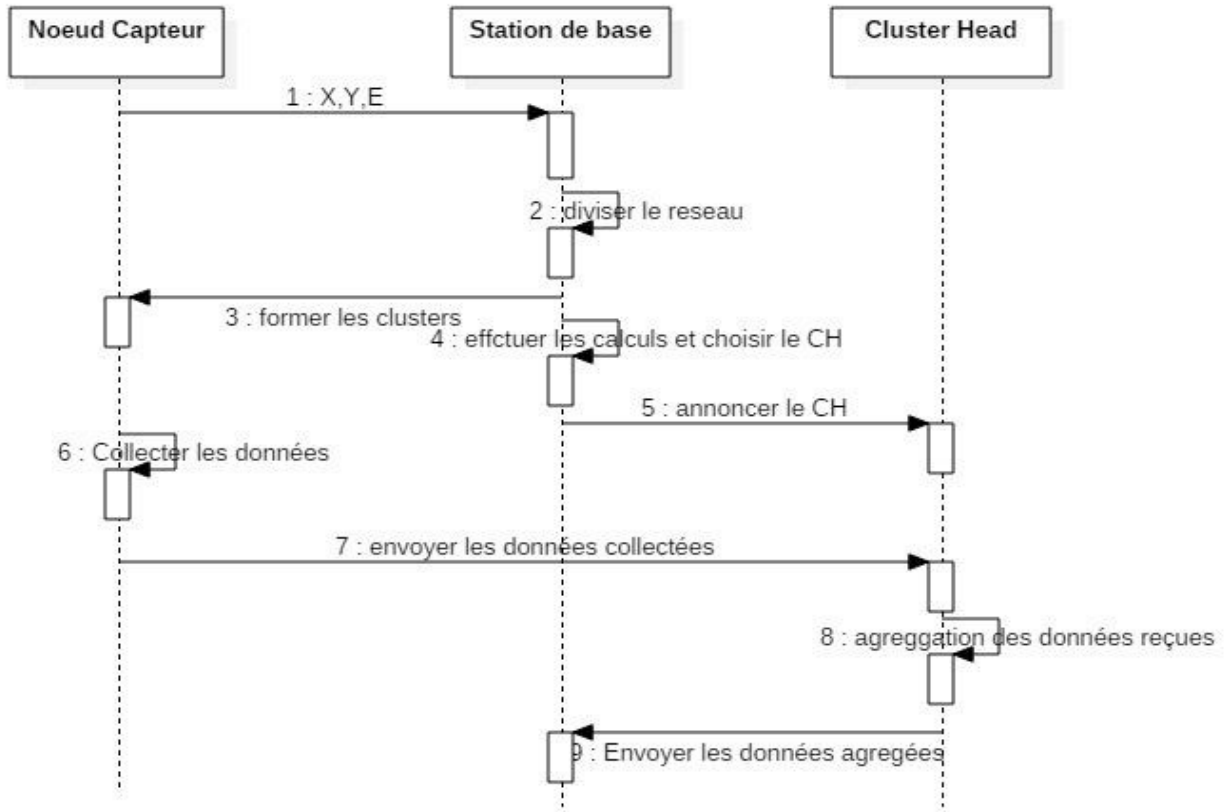

Figure 4-14 Diagramme de séquence des événements du WDC-LEACH-C

4.5.2 **La modélisation formelle**

On peut modéliser formellement la phase la plus importante du protocole WDC-LEACH-C pour la sélection de CH comme suit :

- Le calcul de la distance moyenne $d_{avg}^{(i)}$ séparant chaque nœud des autres membres de son cluster se fait selon l'équation (6)

$$d_{avg}^{(i)} = \frac{1}{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} d_{ij} \qquad (6)$$

Où

$d_{avg}^{(i)}$: est la distance moyenne

d_{ij}: est la distance entre le nœud i et le nœud j

n : est le nombre total des nœuds dans le cluster

- La sélection du nœud WDC (qui présente la plus courte distance moyenne) est donnée par l'équation (7)

$$d_{WDC} = \min\left\{d_{avg}^{(i)}\right\}_{i=1}^{n} \quad (7)$$

- La construction du groupe G est effectuée par le test suivant :

```
{
Init G=0;

For (i=1…n)
If  d_avg^(i) < d_Threshold  then
      G = G + 1;
End}
```

Où $d_{Threshold}$ est donné par la formule proposée suivante qui fournit une meilleure sélection du groupe G dans chaque cluster

$$d_{Threshold} = \frac{d_{WDC} + \frac{1}{n}\sum_{i=1}^{n} d_{avg}^{(i)}}{2} \quad (8)$$

- Le CH est le nœud qui a le maximum d'énergie dans le groupe G. Cette sélection est faite de la manière suivante :

1- Calcul de la probabilité de l'énergie résiduelle de chaque nœud du groupe G par l'équation 9.

$$P_j = \frac{E_{residual}^{(j)}}{E_{initial}^{(j)}} \qquad j \in G \quad (9)$$

2- Ensuite on va choisir la plus grande probabilité. Ceci est donné par l'équation (10)

$$E_{max} = max\left\{P_j\right\}^{G} \quad (10)$$

Le nœud j devient alors un CH.

4.5.3 **Modélisation par automates temporisés**

La phase de modélisation du système a pour but de fournir une représentation formelle du système étudié. De nombreux modèles ont été proposés. Leur sémantique est définie sous la forme de systèmes de transitions où les nœuds représentent les états du système tandis que les transitions décrivent les évolutions possibles d'un état à un autre. On peut citer les structures de Kripke, les automates d'états finis, les automates temporisés et les réseaux de Pétri. Dans cette section, nous modélisons les différents composants de notre contribution par les automates temporisées. Les figures 4-15, 4-16, 4-17 représentent les automates d'un nœud capteur, un CH et la SB.

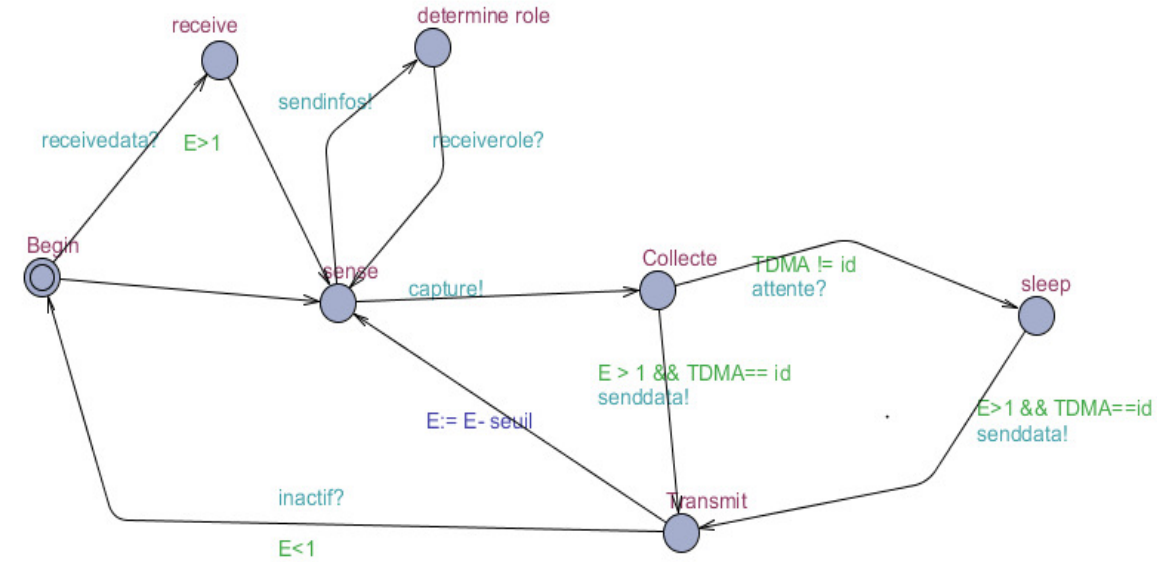

Figure 4-15 Automate représentant un nœud capteur

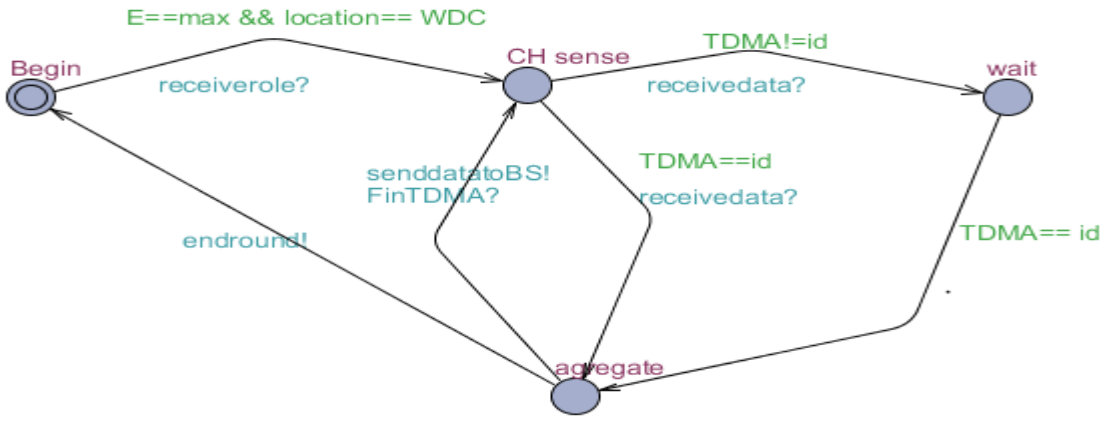

Figure 4-16 Automate représentant un cluster head

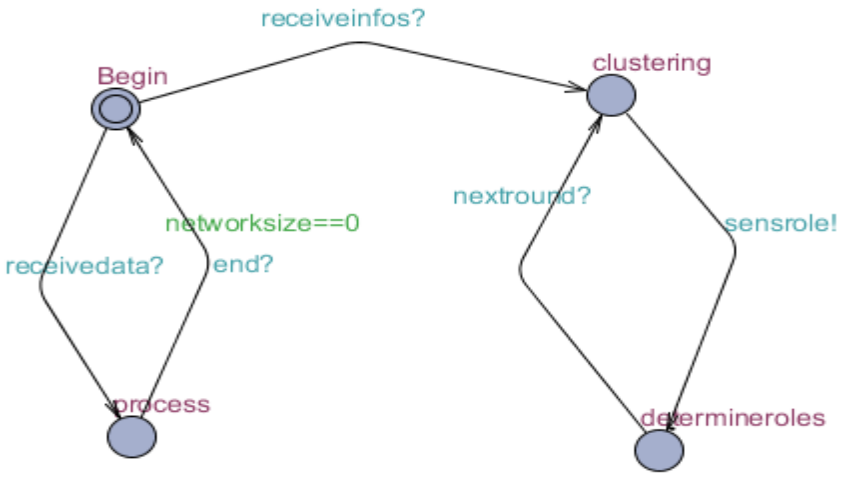

Figure 4-17 Automate représentant la station de base

4.6 **Simulation et évaluation des performances**

Après avoir terminé la conception et la modélisation de notre proposition, nous entamons la dernière phase qui concerne la simulation. C'est une phase qui consiste dans la mise en pratique de tout ce qui a été fait auparavant. Dans cette section, nous effectuons une étude de simulation pour démontrer l'aspect pratique et l'efficacité de notre protocole de clustering que nous avons dénommé nommé WDC-LEACH-C. Afin de valider les performances de l'algorithme WDC-LEACH-C proposé, une comparaison avec les algorithmes LEACH-C, Impro-LEACH et DE-LEACH a été réalisée. Nous évaluons les performances de notre protocole en termes de l'efficacité énergétique, la durée de vie du réseau, le nombre des nœuds vivants lors du déroulement du protocole, la quantité des données et le nombre de paquets envoyées à la station de base. Le protocole est implémenté avec le simulateur NS2. Le tableau 4-1 présente les paramètres de simulation.:

Tableau 4-1 Les paramètres de simulation

Paramètres	**Valeurs**
La zone de simulation	(100 m× 100 m)
La taille du réseau	100 nœuds
Location de la SB	(50,50)
Energie minimum	0.001 joule
Période maximum de simulation	3600 secondes
Taille d'un paquet	25 Octets

4.6.1 **Durée de vie du réseau**

La durée de vie du réseau est définie selon les trois critères suivants :

- FND (First Node Died) : qui est également appelée période de stabilité, il s'agit de l'intervalle de temps entre le début de la simulation et la mort du premier nœud.
- HLD (Half Node Died) : représente l'intervalle de temps entre le début de la simulation et la mort de la moitié des nœuds.
- LND (Last Node Died) : représente l'intervalle de temps entre le début de la simulation et la mort du dernier nœud.

La figure 4-18 représente la durée de vie du réseau dans laquelle FND, HND et LND sont définis pour LEACH-C, Improved-LEACH, DE-LEACH et pour le protocole proposé WDC-LEACH-C. A partir de cette figure, nous remarquons que le temps de mort des premiers, de la moitié et derniers nœuds dans WDC-LEACH-C est plus long que celui dans LEACH-C, Impro-LEACH, et DE-LEACH. Par conséquent, WDC-LEACH-C offre une meilleure durée de vie au réseau grâce à la minimisation du nombre et de la longueur des chemins de transmission, ce qui optimise la consommation d'énergie des capteurs.

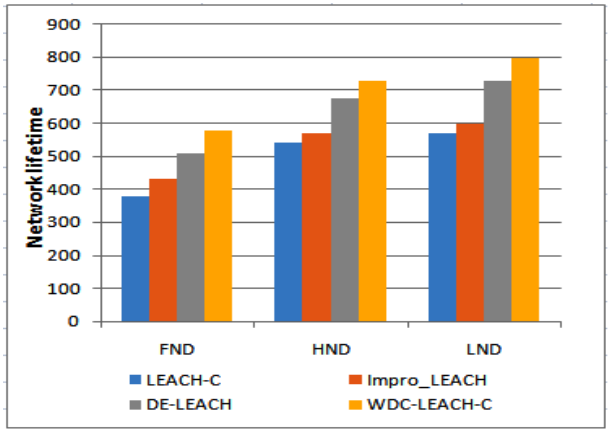

Figure 4-18 La durée de vie du réseau

4.6.2 **Nombre de nœuds vivants par cycle**

La mesure du nombre de nœuds vivants à chaque cycle permet de connaitre l'état du déroulement du protocole. Cette mesure représente la durée de stabilité des nœuds et les moments où ils meurent et permettent, ainsi, de caractériser la manière de la disparition (lente ou rapide).

La figure 4-19 montre le nombre de nœuds vivants à chaque cycle dans le réseau. Comme nous pouvons le voir sur cette figure, la disparition des nœuds vivants diminue lentement dans l'algorithme proposé WDC-LEACH-C, comparé aux algorithmes LEACH-C,

Improved-LEACH et DE-LEACH. Lorsque tous les nœuds sont inactifs car toute l'energie a été epuisée dans LEACH-C, Impro-LEACH et DE -LEACH, les nœuds dans WDC-LEACH-C peuvent encore fonctionner pendant plusieurs cycles supplémentaires. Par conséquent, la stabilité du réseau et la durée de vie dans l'algorithme proposé sont meilleures comparée à celles des algorithmes DE-LEACH, Impro-LEACH et LEACH-C.

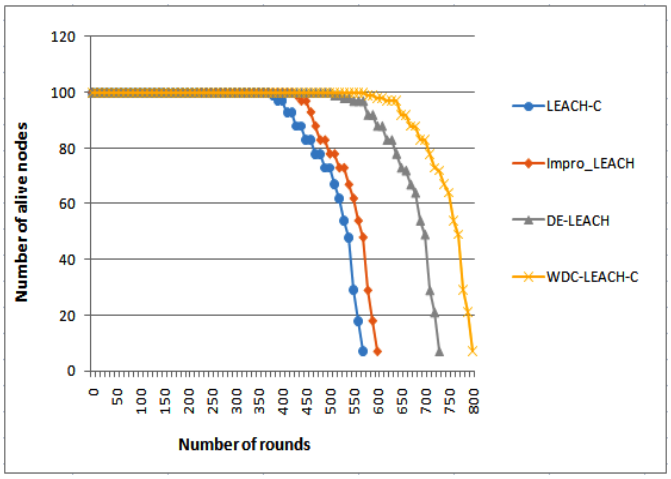

Figure 4-19 Le nombre des nœuds vivants

4.6.3 **L'énergie consommée**

Cette mesure représente l'énergie consommée des nœuds au cours des cycles de simulation. L'objectif principal d'un protocole de routage est de conserver l'énergie des capteurs car si cette dernière est épuisée, le nœud capteur sera inactif et sa durée de vie sera terminée. L'énergie est donc le facteur clé à prendre en compte dans les protocoles de routage.

La figure 4-20 illustre l'energie consommée des nœuds dans le réseau. A partir de cette figure, nous pouvons voir que notre protocole WDC-LEACH-C consomme moins d'énergie que LEACH-C, Improved-LEACH et DE-LEACH au cours des cycles de simulation. Cela peut s'expliquer par le fait que l'algorithme proposé offre un meilleur clustering et une selection efficace des CH. Puisque le CH est le nœud qui se situe au centre de densité des autres noeuds du cluster et qui a dispose du maximum d'energie, les transmissions sont effectuées sur de courtes distances. Ainsi, les performances du réseau ont été grandement améliorées.

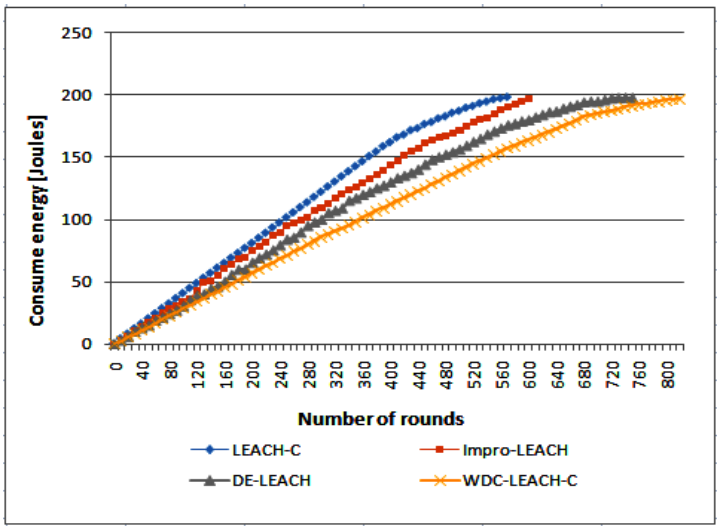

Figure 4-20 L'énergie consommée

4.6.4 **La quantité de données reçues par la BS**

La quantité des données reçues par la station de base est mesurée en octets. Chaque paquet de données reçu par la station de base correspond à 25 octets. La périodicité des paquets initiaux qui comportent les informations des nœuds (localisation et niveau d'énergie), ces paquets sont ignorés en temps de simulation car notre algorithme de clustering n'inclut pas les transmissions périodiques à la station de base.

La figure 4-21 montre la quantité de données reçues par la station de base, nous pouvons voir que la quantité de données reçues est beaucoup plus grande dans le protocole proposé par rapport à DE-LEACH, Impro-LEACH et LEACH-C.

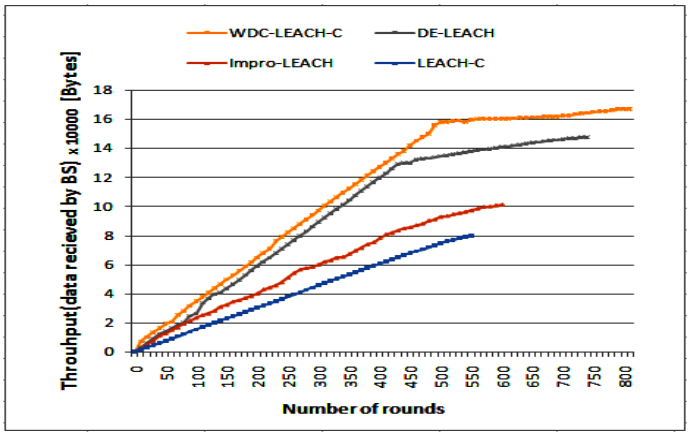

Figure 4-21 Quantité de données reçus par la SB

La figure 4-22 montre que le nombre total de paquets envoyés à la station de base est plus élevé dans WDC-LEACH-C que dans les algorithmes DE-LEACH, Improved-LEACH et LEACH-C. Ces résultats incluent les paquets initiaux d'énergie et la position GPS

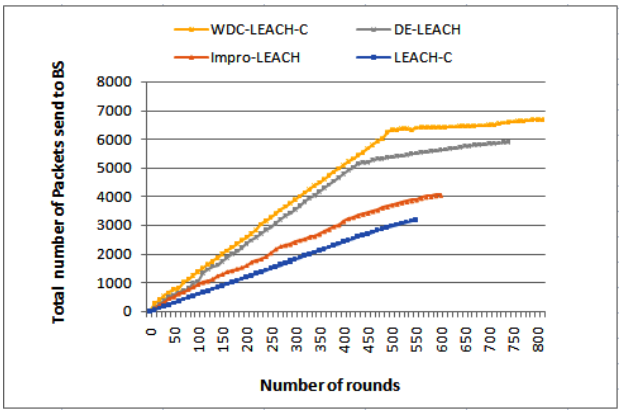

Figure 4-22 Le nombre de paquets reçus par la SB

Le tableau 4-2 résume les résultats obtenus. Comme nous pouvons le voir dans ce tableau, le WDC-LEACH-C est plus performant que les autres algorithmes. Cette performance est en termes de consommation d'énergie, la durée de vie du réseau, les données et les paquets reçus par le BS

Tableau 4-2 Les résultats de simulation

Protocol	First node died in round	Last node died in round	Total quantity of data sent to the BS (Bytes)	Total number of packets sent to BS
LEACH-C	382	569	79845	3193
Impro-LEACH	429	600	101434	4057
DE-LEACH	511	732	147599	5903
WDC-LEACH-C	581	801	167132	6680

4.7 **Le deuxième protocole de routage hiérarchique proposé Improved-ACORP**

Dans cette section nous présentons un second protocole que nous proposons pour le routage hiérarchique à structure clustérisée des nœuds dans les réseaux de capteurs sans fils. Nous commencerons par présenter le protocole de base ACORP et après nous donnerons une vue d'ensemble sur l'algorithme puis nous le détaillerons.

4.7.1 **Le protocole de base ACORP**

ACORP (Ant Colony Optimization Routing Protocol) est un protocole de routage hiérarchique intelligent pour les réseaux de capteurs sans fils homogènes auto-organisés basé sur les colonies de fourmis. L'idée principale de ACORP est inspirée du comportement des colonies de fourmis réelles. Il est exécuté en cycles où chaque cycle est composé de deux phases : la phase de clustering et la phase de routage. Dans la première phase une division circulaire dynamique est effectuée pour construire des clusters équilibrés en termes de nombre de nœuds. Ensuite et selon l'énergie résiduelle des nœuds, les CHs sont élus. La figure 4-23 résume les étapes de la phase de clustering. La pierre angulaire de cette proposition est le routage intra cluster multi-sauts dans laquelle la technique ACO est utilisée pour établir les routes afin d'acheminer les données collectées par les nœuds capteurs vers les CHs à travers des nœuds relais. Dans le premier cycle, la sélection du prochain saut est basée sur la distance entre les nœuds. Dans les autres cycles, la technique ACO est utilisée dans laquelle la quantité de phéromones déposée sur les chemins représente par le nombre de sauts. La figure 4-24 résume les étapes de la phase de routage (inter et intra cluster). La figure 4-25 représente l'organigramme de ACORP.

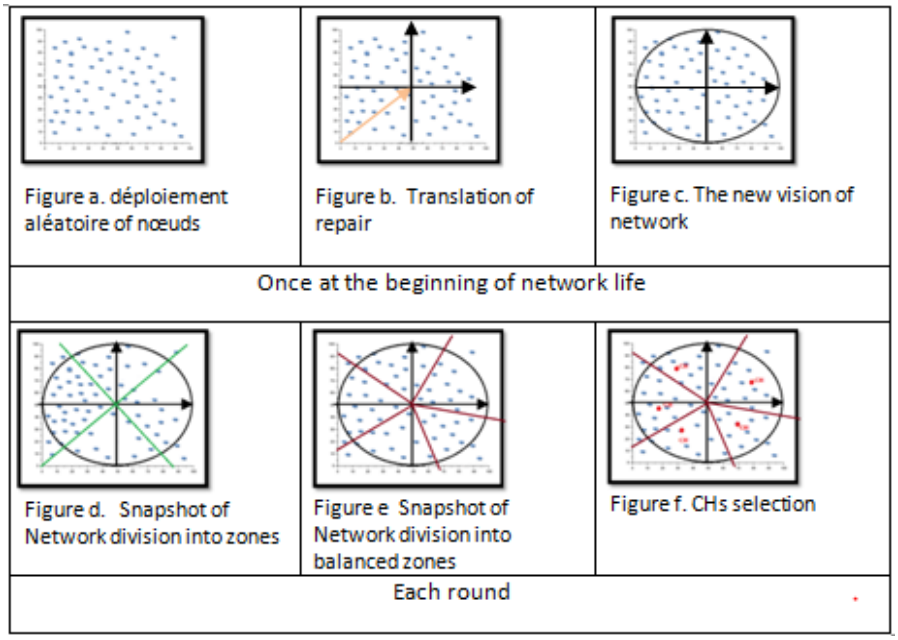

Figure 4-23 Les étapes de la phase de clustering de ACORP

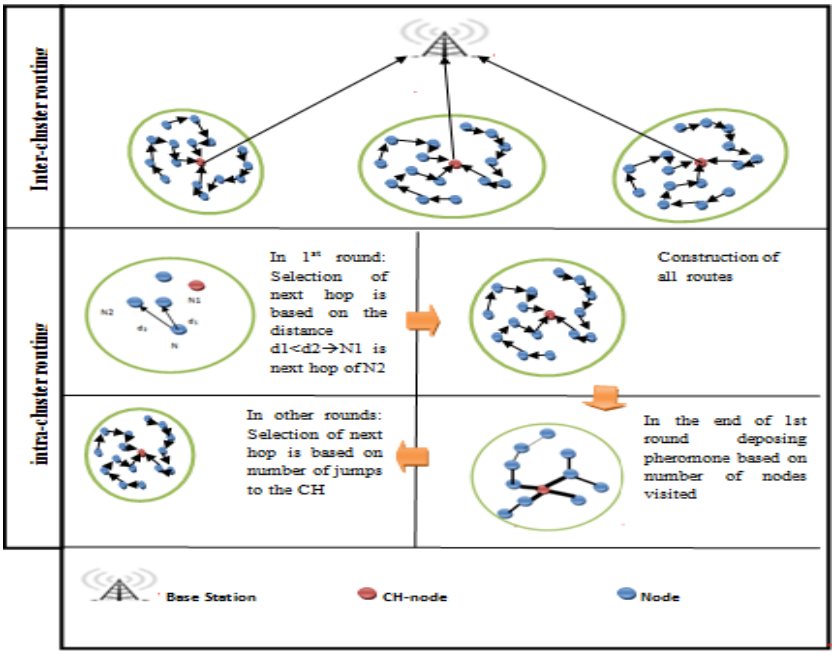

Figure 4-24 Les étapes de la phase de routage de ACORP

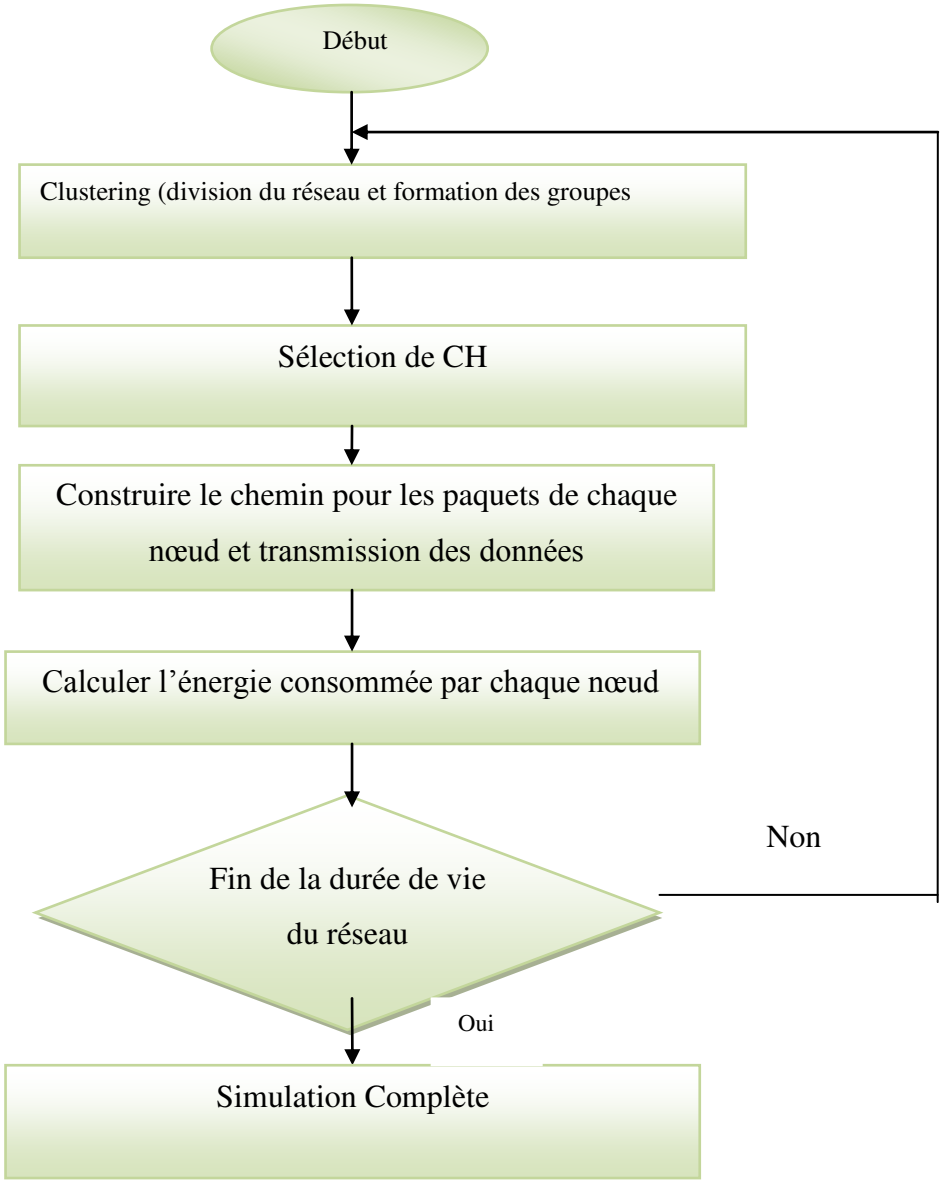

Figure 4-25 L'organigramme de ACORP

4.7.2 **Vue d'ensemble du protocole proposé**

Le protocole proposé Improved-ACORP est un protocole de routage à architecture hiérarchique clustérisée. C'est une version améliorée du protocole ACORP. Improved ACORP est exécuté en cycles comme ACORP. Chaque cycle est divisé en deux phases : la phase de clustering et la phase de routage. Improved ACORP a rajouté plusieurs améliorations à ACORP afin d'augmenter ses performances. La sélection des CHs dans Improved ACORP diffère entre le premier cycle et les autres cycles. Dans le premier cycle, la distance entre la station de base et les nœuds est calculée afin de choisir le nœud le plus proche de la station de base en tant que CH ce qui n'est pas considéré dans ACORP où la sélection du CH dans le premier cycle est aléatoire parce que tous les nœuds au début sont

homogènes en terme de puissance d'énergie. Dans les autres rounds, en plus de la distance à la station de base, l'énergie résiduelle est considérée dans la sélection du CH. Dans la phase de routage intra cluster, Improved ACORP considère le critère d'énergie en plus de la distance et le nombre de sauts pour la sélection des chemins vers le CH. Lors de cette phase, une stratégie de gestion d'énergie des CHs (CHMS : Cluster Head Management Strategy) est établie afin de surveiller le niveau d'énergie du CH et fournir un assistant si le niveau à atteindre un seuil critique. La figure 4.26 représente l'organigramme de Improved-ACORP.

4.7.3 **Détails du protocole**

Le protocole proposé évolue en deux phases : la phase de clustering qui se déroule en deux étapes ; le partitionnement du réseau pour la formation des clusters et la sélection des CH. La deuxième phase consiste dans le routage et la transmission intra et inter clusters. Au cours de cette phase, une stratégie de gestion d'énergie des CHs est élaborée. La phase de partitionnement du réseau avec la division circulaire est similaire à ACORP.

4.7.3.1 La phase de clustering

1. La formation de clusters

L'étape de partitionnement du réseau afin de former les clusters est similaire à celle utilisée dans ACORP, où une division circulaire dynamique est utilisée.

2. La sélection des CHs

Dans la phase de clustering, Improved ACORP diffère de ACORP au niveau de l'étape de sélection des CHs. Il ne considère pas l'effet aléatoire de la sélection dans le premier cycle en ce sens qu'il se base sur la distance entre les nœuds et la station de base afin de choisir le nœud le plus proche de cette dernière pour choisir le CH. Cela permet d'éviter une dissipation abusive d'énergie qui serait causée par des transmissions effectuées sur de longues distances. Dans les autres cycles, en plus de la distance à la station de base, l'énergie résiduelle des nœuds est considérée dans la sélection du CH.

1. La sélection du nœud superviseur

Le protocole proposé dans Improved ACORP a introduit une stratégie de gestion d'énergie des CHs dans les situations critiques afin de conserver leur énergie résiduelle et maximiser, ainsi, la durée de vie du cluster. Dans cette stratégie, un nœud superviseur est sélectionné dans chaque cluster il joue le rôle du contrôleur qui surveille le CH et il reçoit un message d'alerte en cas d'urgence lorsque l'énergie du CH atteint un seuil critique.

4.7.3.2 Phase de routage (steady state)

Dans la phase de routage de Improved ACORP, une combinaison du mode de communication multi-sauts, la technique ACO et la stratégie CHMS est proposée pour réduire la consommation d'énergie. Dans la phase de routage, la construction des chemins est la tâche la plus importante. Elle consiste à créer des chemins entre les nœuds sources émetteurs et le CH à travers des nœuds intermédiaires. La sélection du prochain saut pour un noeud émetteur ou un nœud intermédiaire diffère entre le premier cycle et les autres cycles. Dans le premier cycle, un nœud émetteur choisit son prochain saut en considérant la distance qui le sépare du nœud le proche de lui pour le considérer comme son prochain nœud dans le chemin vers le CH. Dans les autres cycles, la technique ACO est utilisée, Improved ACORP rajoute le critère de l'énergie résiduelle des nœuds en plus du nombre de sauts et de la distance entre les nœuds considérés dans ACORP. Pour éviter les collisions des transmissions, le protocole TDMA (Time Division Multiple Access) est utilisé dans l'établissement de l'ordre des transmissions entre nœuds.

4.7.4 **La stratégie de gestion d'énergie des CHs (CHMS)**

La stratégie CHMS est un mécanisme de gestion d'énergie des CH qui vise la conservation de leur énergie et prolonger, ainsi, la durée de vie du cluster en assurant le fonctionnement du réseau de capteurs d'une manière continue et efficace en cas de puissance d'énergie critique des CHs. Cette stratégie est appliquée lors de la phase de routage. Deux types de nœuds interviennent dans cette approche : le nœud superviseur (supervisor node) et le nœud assistant (assistant node) qui sont situés au niveau de chaque cluster assurant la distribution du fonctionnement du CH en cas de situations critiques. La stratégie fonctionne comme suit : lors de la phase de routage intra ou inter cluster lorsque le niveau d'énergie du CH atteint un niveau critique il envoie un message d'alerte au nœud superviseur qui est sélectionné après la phase de clustering et qui est le nœud le plus proche du CH. Ce nœud superviseur envoie une demande à la station de base pour identifier le nœud le plus proche au CH et qui a une énergie résiduelle supérieure (supérieure à celle de CH) afin de l'élire comme assistant du CH. La station de base envoie l'identifiant du nœud demandé au nœud superviseur. Ce dernier envoie un message au nœud assistant pour l'informer qu'il est considère en tant que nœud assistant du CH qui effectue les transmissions des données à la SB.

L'approche proposée est modélisée avec un système multi agents (MAS). Cette solution maximise la durée de vie du réseau et minimise la consommation d'énergie des capteurs.

4.7.5 **La modélisation formelle du protocole proposé Improved ACORP**

On peut modéliser formellement les phases de Improved ACORP pour la sélection de CH, le routage des données et la stratégie CHMS comme suit :

- Apres réception des coordonnées des nœuds par la station de base. Cette dernière va calculer la distance entre elle et chaque nœud du réseau.
- Le choix du CH dans le premier cycle est basé sur la distance entre les nœuds et la station de base. Le nœud le plus proche de la station de base est sélectionné comme CH et ceci est effectué grâce à l'équation (11) de probabilité suivante :

$$P_i = \frac{1}{dSB_i} \quad (11)$$

- Ensuite la plus grande probabilité est calculée. Ceci est donné par l'équation (12)

$$P_{CH} = max\{P_i\} \quad (12)$$

- Pour la sélection du CH dans les autres rounds, un calcul de la probabilité de l'énergie résiduelle et la distance de chaque nœud dans chaque cluster grâce à l'équation 13.

$$P_i = \frac{1}{dSB_i} * \frac{E_{residual}^{(i)}}{E_{initial}^{(i)}} \quad (13)$$

- Ensuite la plus grande probabilité est choisie. Ceci est donné par l'équation (14)

$$P_{CH} = max\{P_i\} \quad (14)$$

Le nœud i devient alors un CH.

- Pour la sélection du nœud superviseur la distance entre le CH et tous les nœuds du cluster est calculée. Le nœud le plus proche au CH est sélectionné comme nœud superviseur

$$P_i = \frac{1}{dCH_i} \quad (15)$$

- Ensuite la plus grande probabilité est calculée. Ceci est donné par l'équation (16)

$$P_{superviseur} = max\{P_i\} \quad (16)$$

Le nœud i devient alors un superviseur.

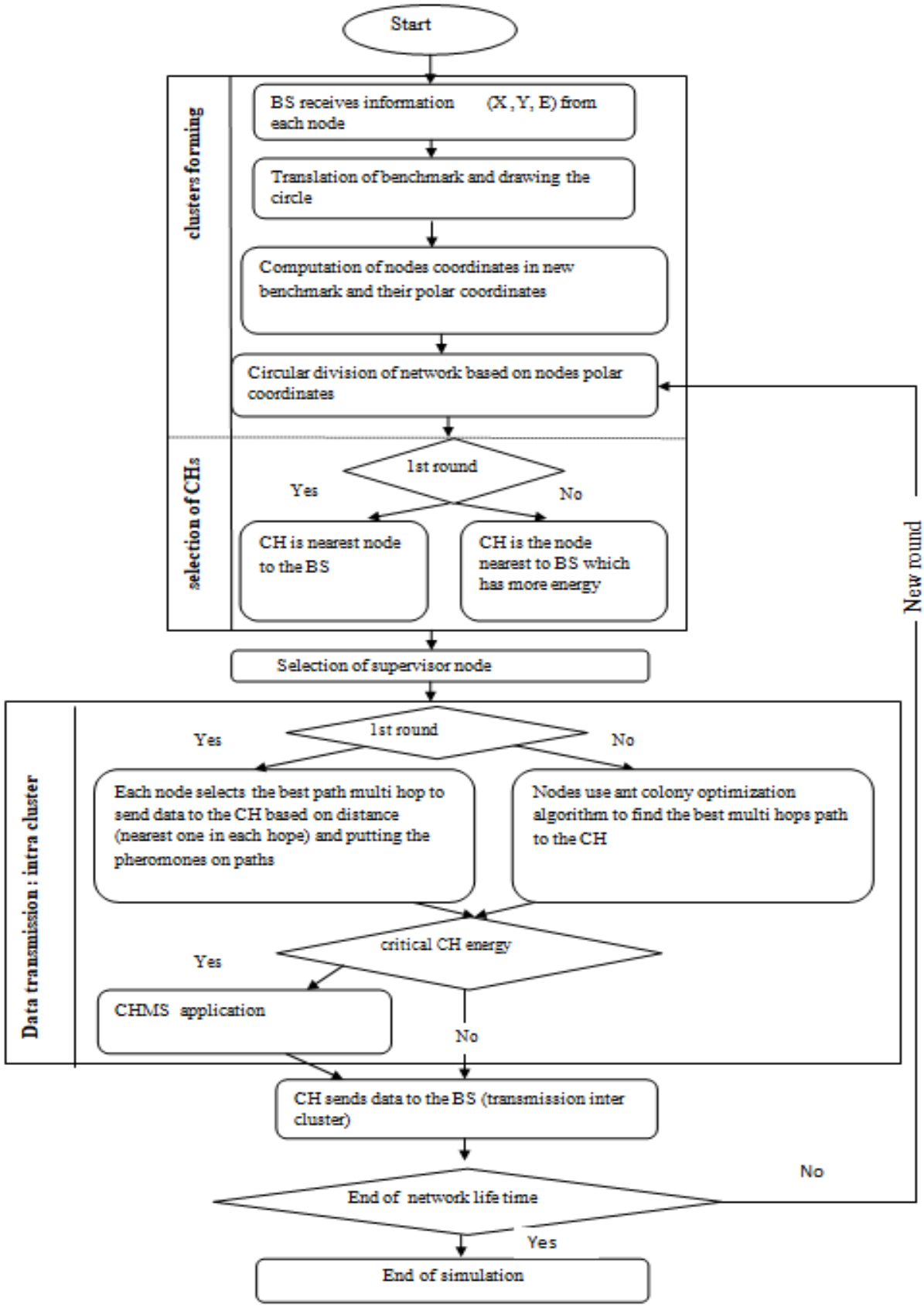

Figure 4-26 L'organigramme de Improved-ACORP

- Dans la phase de routage, la construction des chemins au premier cycle se fait selon l'algorithme suivant

Algorithme 2
1.For $i =$ $1 \dots \dots nombre\ de\ noeuds\ vivants\ dans\ le\ réseau$ i=1……numberofalivenodesinnetwork
2.Calculer la distance entre le $CH\ et\ i$ dans vecteur-distance
3. Trier le vecteur-distance de façon décroissante
4. For $i = 1 \dots \dots nombre\ de\ noeuds\ vivants\ dans\ le\ réseau$ i=1……numberofalivenodesinnetwork
$TDMA[i] = distance[i]$
$TDMA[nombre\ de\ noeuds\ vivants\ dans\ le\ réseau] = CH$

Dans les autres cycles, la sélection du prochain saut pour construire les chemins vers le CH s'effectue selon l'équation 17

$$p = \frac{\tau_{ij}{}^{\alpha}.\mu_{ij}{}^{\beta}.\delta^{\gamma}}{\sum_{j\epsilon N_i} \tau_{ij}{}^{\alpha}.\mu_{ij}{}^{\beta}.\delta^{\gamma}} \quad (17)$$

Où

$$\tau_{ij} = \frac{1}{\text{pheromone(j,ch)}} \quad (18)$$

Où phéromone (j,ch) : représente le nombre de sauts entre le nœud j et le CH et est défini par l'équation suivante

$$Pheromone(i, CH) = 1/L(packet) \quad (19)$$

La valeur de phéromone déposée sur tous les chemins (qui est modélisée par le nombre de sauts) est calculée à la fin du premier cycle.

Où

i : est le nœud source,

$Pheromone$: est la quantité de phéromone du noeud source i au CH,

L : est le nombre de sauts effectués par un paquet pendant son chemin.

τ_{ij}: est la quantité de phéromone du prochain saut.

$$\mu_{ij} = \frac{1}{d(i,j)} \quad (20)$$

μ_{ij} : est l'information heuristique liée à la distance entre les nœuds.

$$\delta = E_{\text{Residuel}} \quad (21)$$

N_i: représente la liste des voisins du nœud.

α, β and γ : Trois paramètres, qui contrôlent respectivement l'importance du phéromone et la longueur du chemin entre les nœuds i et j et sont donnés comme suit :

$$\alpha = 0.5 \ ; \beta = 0.2 \ ; \gamma = 0.6$$

- L'énergie résiduelle critique est calculée comme suit :

$$E_{critique} = 2*(Er+ E_{agrégation}+E_{transmission}) \quad (22)$$

- La sélection du nœud assistant est basée sur la distance minimale entre les nœuds du cluster et le CH ou le nœud le plus proche au CH et qui a une énergie supérieure est devenu un nœud assistant de CH

$$P_i = \frac{1}{dCH_i} * \frac{E_{residual}^{(i)}}{E_{initial}^{(i)}} \quad (23)$$

- Ensuite la plus grande probabilité est calculée . Ceci est donné par l'équation (24)

$$P_{assistant} = max\{P_i\} \quad (24)$$

4.7.6 **La modélisation Objet utilisant le langage UML**

4.7.6.1 Diagramme de Classe

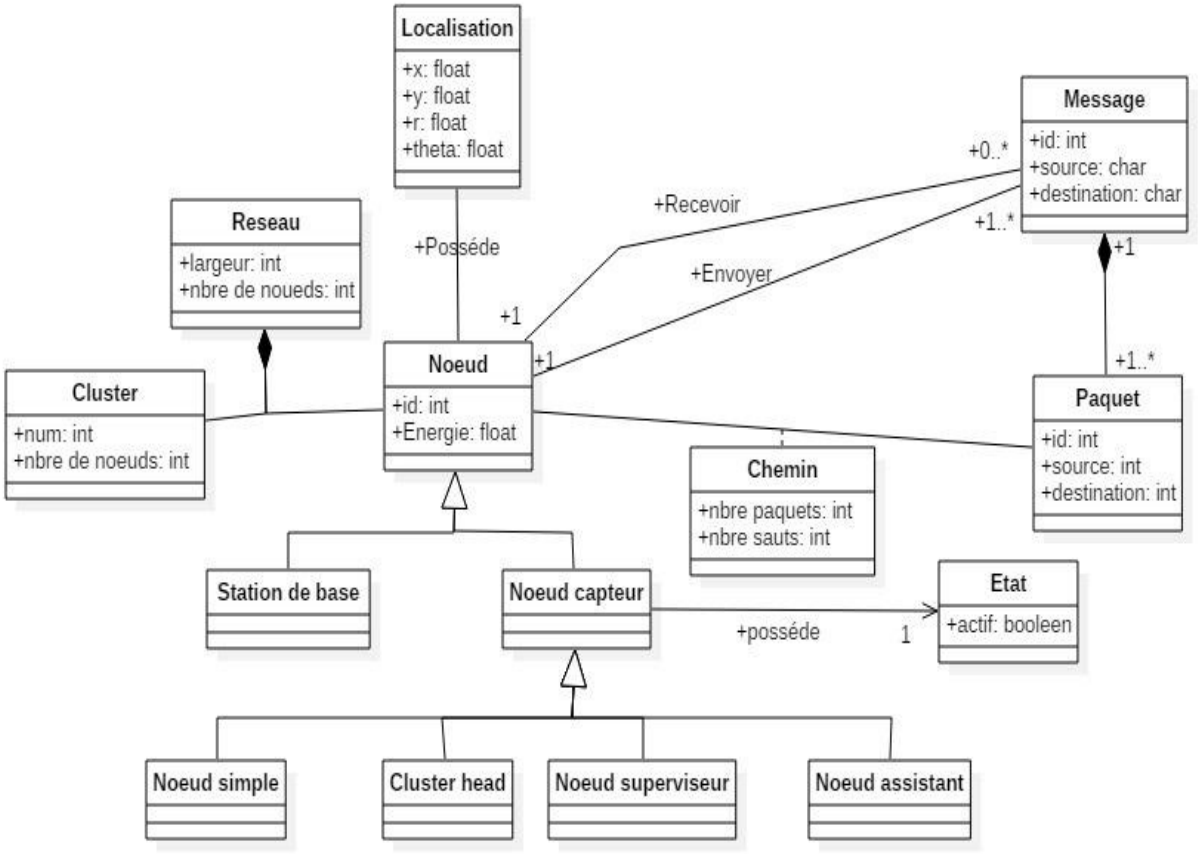

Figure 4-27 Le diagramme de classe du protocole Improved-ACORP

4.7.6.2 Diagramme de cas d'utilisation

a- Le diagramme de cas d'utilisation globale

But : Le déroulement global du protocole

Acteur : Utilisateur.

Description du traitement nominal : L'utilisateur peut :

- Définir le modèle réseau et le modèle radio.
- Déploiement du réseau.
- Simuler le réseau.
- Consulter l'état du réseau.

(voir figure 4-28)

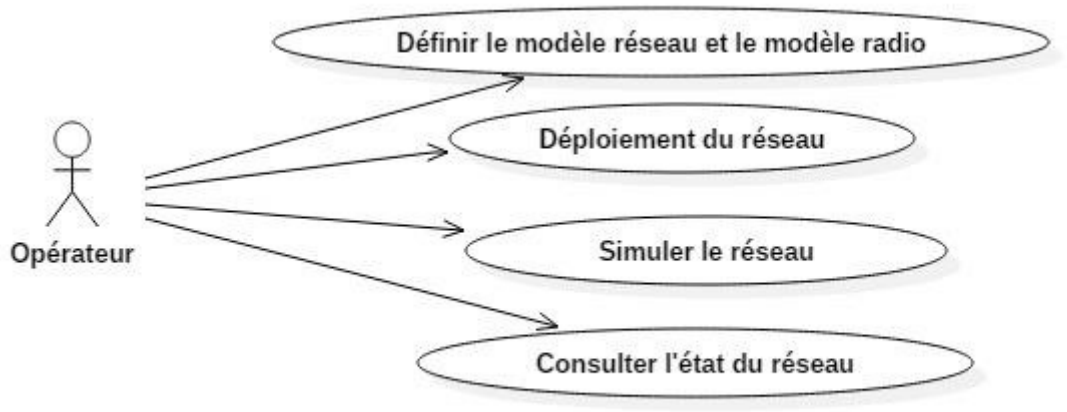

Figure 4-28 Diagramme de cas d'utilisation globale du protocole

b- Le diagramme de cas d'utilisation pour le clustering

But : le clustering des nœuds et la sélection du CH pour chaque cluster

Acteur : station de base.

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : la station de base peut :

- Recevoir les informations des nœuds.
- Translation du repère.
- Calculer les coordonnées polaires.
- Créer les clusters.
- Sélectionner les CHs.
- Sélectionner les nœuds superviseurs.

(voir figure 4-29)

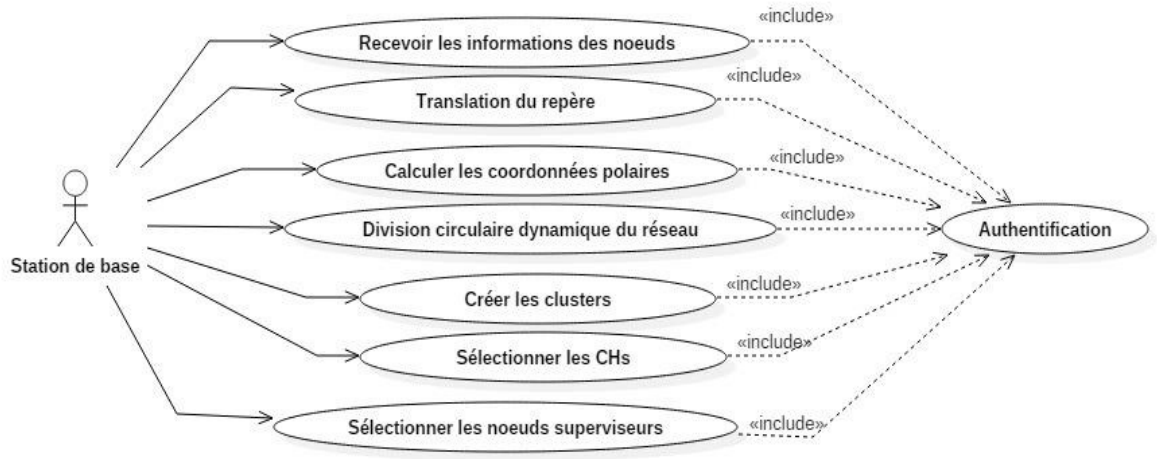

Figure 4-29 Diagramme de cas d'utilisation pour le clustering

c- Le diagramme de cas d'utilisation pour le routage inter-cluster

But : le routage inter-cluster des données

Acteur : station de base.

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : la station de base peut :

- Réception des données à partir des CHs ou des assistants.
- Traiter les données reçues.
- Envoyer les données traitées à l'utilisateur.
- Sélectionner des nouveaux CHs et superviseurs pour un nouveau cycle

(voir figure 4-30).

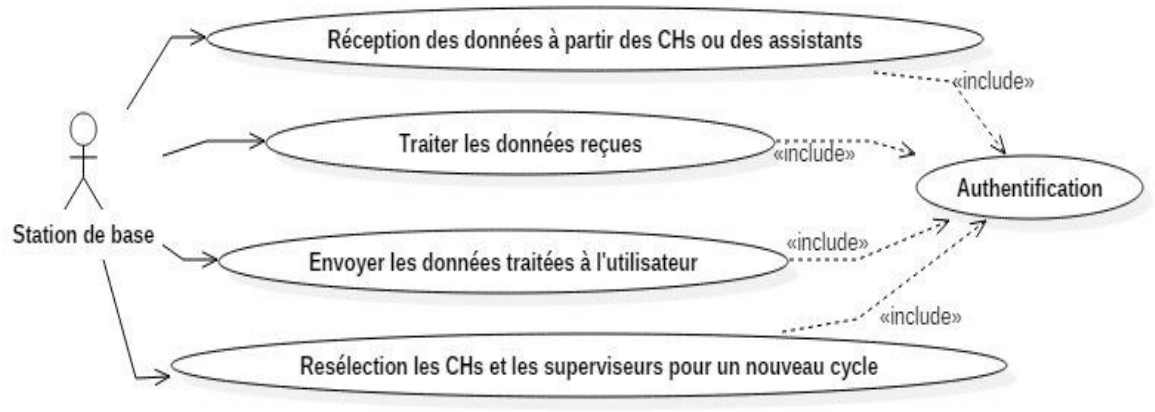

Figure 4-30 Diagramme de cas d'utilisation pour le routage inter cluster des données

d- Le diagramme de cas d'utilisation pour le routage intra-cluster des données

But : le routage intra-cluster des données

Acteur : cluster head

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : le cluster head peut :

- Recevoir les informations collectées par les nœuds.
- Agrégation des données.
- Envoyer les données agrégées à la station de base

(voir figure 4-31)

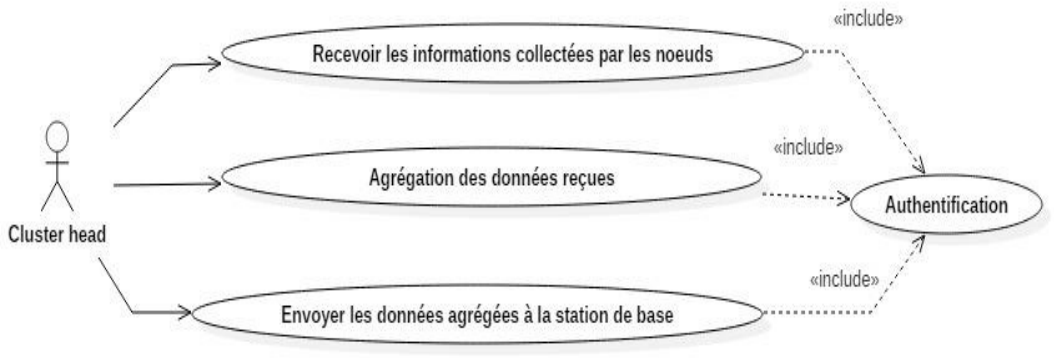

Figure 4-31 Diagramme de cas d'utilisation pour le routage intra-cluster des données.

e- Le diagramme de cas d'utilisation pour le routage intra-cluster des données avec l'application du CHMS

But : le routage intra-cluster des données dans les situations critiques.

Acteur : cluster head.

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : un nœud capteur peut :

- Envoyer un message d'alerte au nœud superviseur.
- Recevoir l'id du nœud assistant.
- Recevoir les données collectées par les nœuds.
- Agrégation des données reçues.
- Envoyer les données agrégées au nœud assistant.

(voir figure 4-32)

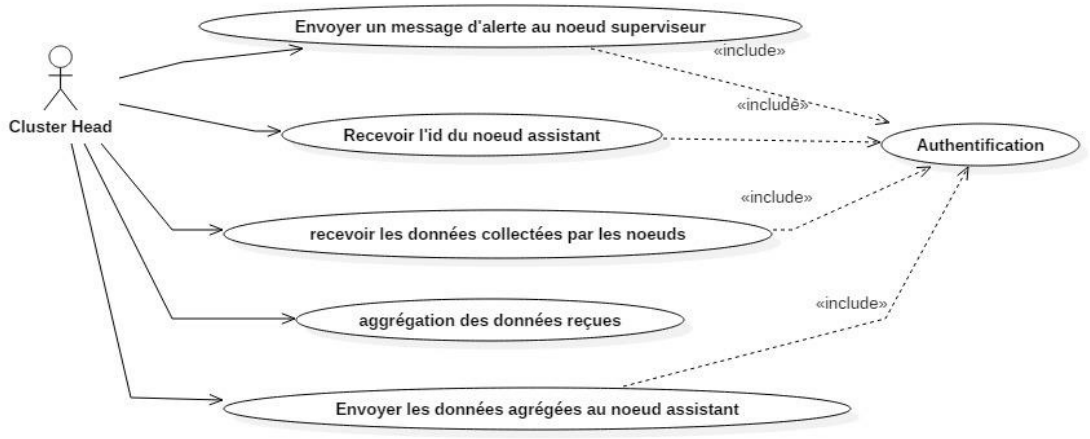

Figure 4-32 Diagramme de cas d'utilisation pour le routage intra-cluster des données lors de l'application de CHMS

f- Le diagramme de cas d'utilisation pour les opérations d'un nœud capteur

But : définir les opérations d'un nœud capteur pendant le déroulement du protocole

Acteur : nœud capteur.

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : un nœud capteur peut :

- Envoyer les informations GPS et énergie à la SB.
- Capter les données de l'environnement.
- Collecter les données captées.
- Envoyer les données collectées au CH.

(voir figure 4-33)

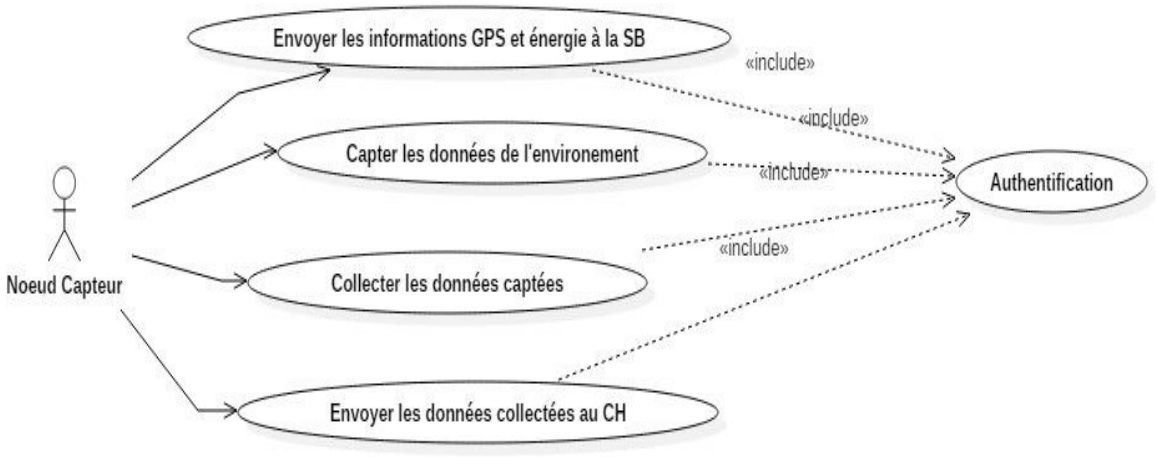

Figure 4-33 Diagramme de cas d'utilisation pour les opérations d'un nœud capteur

g- Le diagramme de cas d'utilisation pour les opérations d'un nœud superviseur

But : définir les opérations d'un nœud superviseur lors de l'application de CHMS

Acteur : nœud superviseur.

Pré conditions : L'opérateur doit Etre authentifié.

Description du traitement nominal : un nœud capteur peut :

- Recevoir un message d'alerte du CH.
- Envoyer une demande d'id à la SB pour un assistant au CH.
- Recevoir l'id de l'assistant.
- Envoyer des messages d'information au CH et à l'assistant.

(voir figure 4-34)

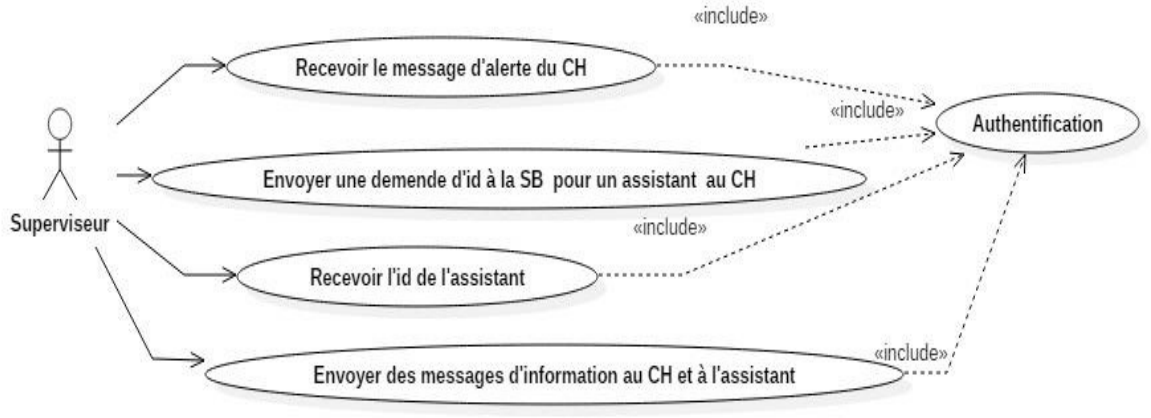

Figure 4-34 Diagramme de cas d'utilisation pour les opérations d'un nœud superviseur

h- Le diagramme de cas d'utilisation pour les opérations d'un nœud assistant

But : définir les opérations d'un nœud assistant lors de l'application du CHMS

Acteur : nœud assistant.

Pré conditions : L'opérateur doit être authentifié.

Description du traitement nominal : un nœud capteur peut :

- Recevoir un message d'information de la part du nœud superviseur.
- Recevoir les données de la part de CH.
- Envoyer les données à la station de base.

(voir figure 4-35)

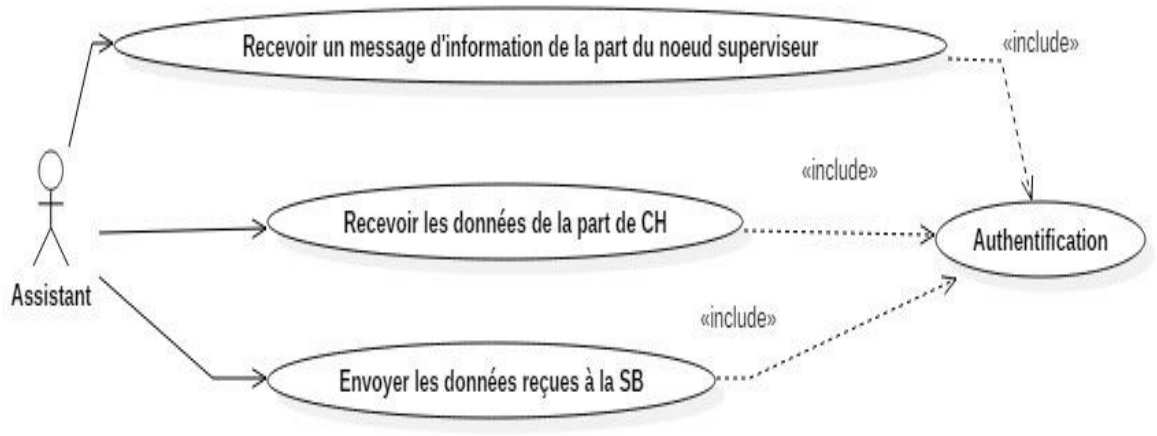

Figure 4-35 Diagramme de cas d'utilisation pour les opérations d'un nœud assistant

4.7.6.3 Diagramme de séquence

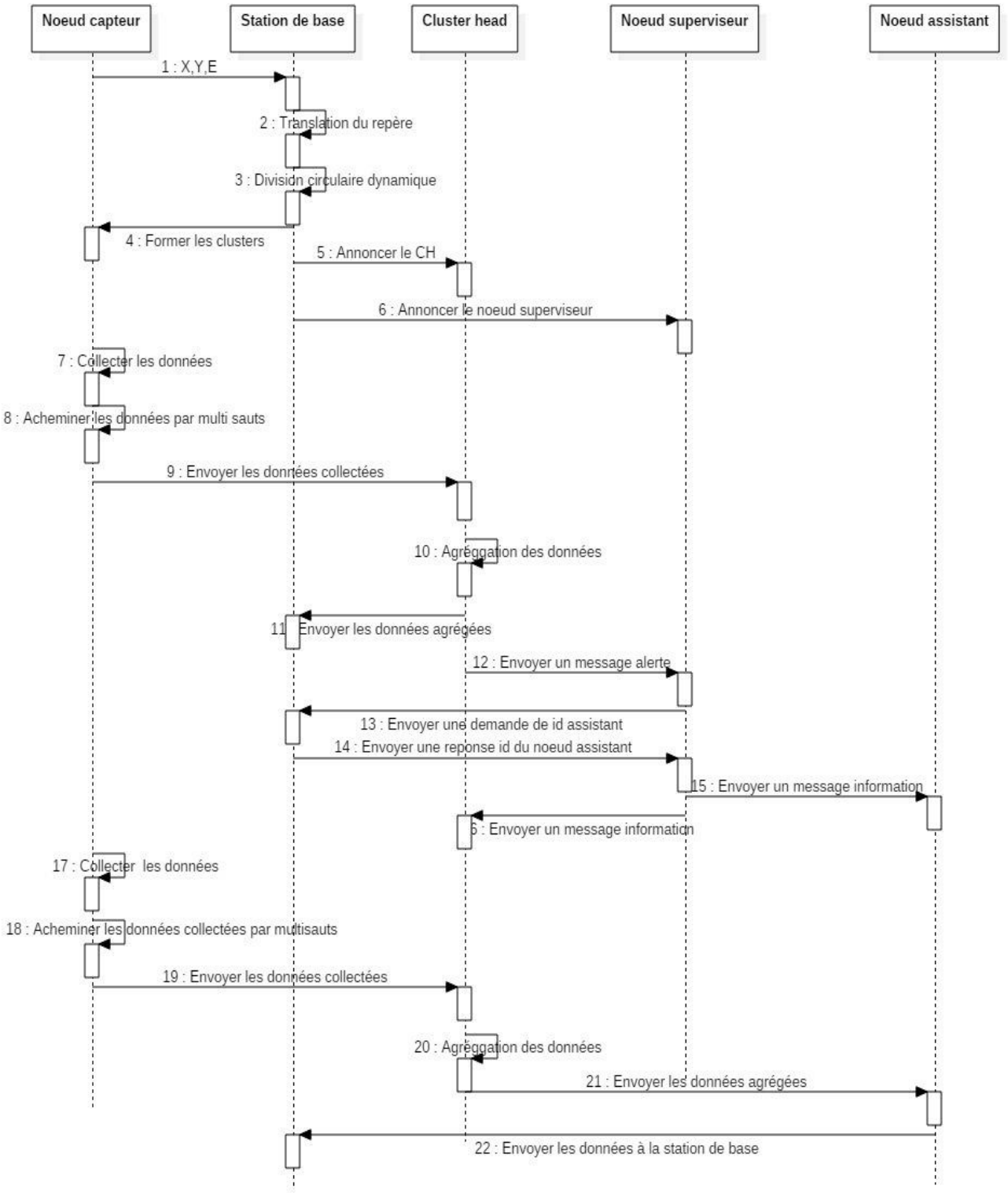

Figure 4-36 Diagramme de séquence des événements dans Improved ACORP

4.7.7 **Modélisation de la stratégie CHMS dans Improved-ACORP par les systèmes multi agents**

Les systèmes multi agents proposent des solutions importantes dans plusieurs domaines et particulièrement dans celui de l'intelligence artificielle distribuée et en informatique de façon générale. Un agent peut être défini comme étant une entité logicielle ou matérielle qui évolue dans un environnement, qui est dotée d'autonomie d'exécution et pourvue d'entrées pour la perception de cet environnement ainsi que de sorties pour agir sur cet environnement. Les agents peuvent communiquer entre eux par tableau noir, par message ou au travers d'environnements partagés. Ils peuvent être réalisés à travers la programmation matérielle ou logicielle. Un système multi-agents est composé d'un certain nombre et pouvant être de même type ou de types hétérogènes et pouvant exécuter des tâches spécifiques. Dans cette section, nous donnons un aperçu de l'approche CHMS modélisée avec un système multi-agents. Notre système est basé sur les contraintes suivantes :

• Chaque agent peut être dans l'un des deux états suivants :

Actif : c'est un nœud vivant présentant présence d'énergie résiduelle.

Inactif : c'est un nœud mort avec une réserve d'énergie nulle.

• L'échange de messages est autorisé entre les agents qui appartiennent à un même groupe et également dans des groupes différents.

Nous avons mis en œuvre l'approche de CHMS à l'aide de systèmes multi-agents comme suit :

• Un agent appelé Agent_Sink qui est situé dans la station de base.

• Un agent CH situé dans chaque groupe.

• Un agent Agent_Assistant situé dans chaque groupe. Cet agent est situé dans le nœud assistant de chaque groupe.

• Les agents simples dans chaque groupe présentent presque le quart 1/4 du nombre total d'agents. Chaque agent est situé dans un nœud du capteur

• Un agent-superviseur. Cet agent est situé dans le nœud superviseur de chaque groupe.

La figure 4-37 représente le diagramme de classe d'agents pour la stratégie CHMS

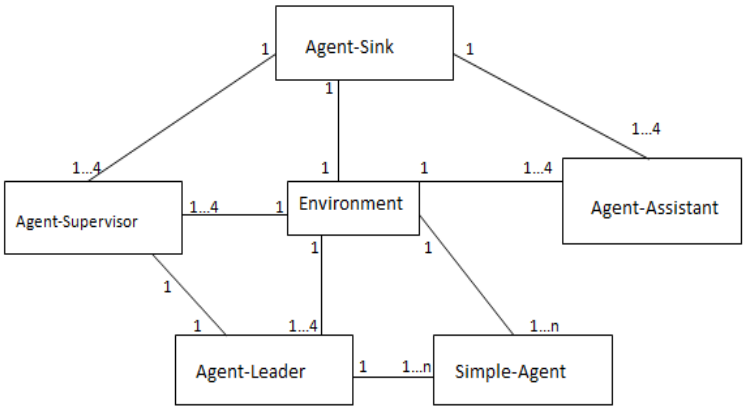

Figure 4-37 Diagramme de classe agents pour la stratégie CHMS

4.7.7.1 La procédure de CHMS

La procédure de CHMS est basée sur les étapes suivantes :

1. Alerte_message : lorsque le niveau d'énergie de l'agent_leader (CH) chute au_dessous d'un seuil donné ; il envoie un message d'alerte à l'Agent_Superviseur.
2. L'Agent_Superviseur demande à l'Agent_sink l'ID de l'agent simple le plus proche au CH et qui a une énergie résiduelle supérieure à celle du CH.
3. L'agent_sink envoie l'identifiant demandé à l'agent-superviseur.
4. Message_info: l'Agent-Superviseur envoie un message à l'Agent Assistant pour l'informer qu'il devient assistant de l'Agent_Leader.
5. Message_info: L'Agent-Superviseur envoie l'ID de l'Agent_Assistant à l'Agent_Leader.

La figure 4-38 résume les différentes étapes de CHMS

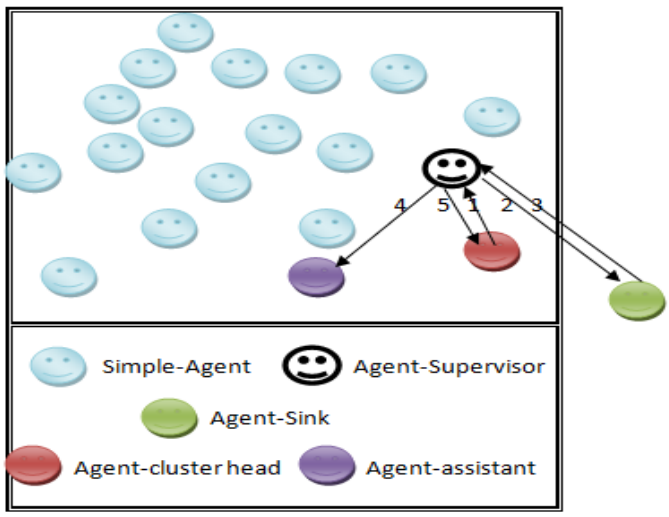

Figure 4-38 Les étapes de CHMS modéliser par les agents

- **Agent_sink**

Cet agent est situé dans la station de base qui dispose d'une énergie et des capacités de traitement et de stockage illimitées. C'est le gestionnaire du système, responsable du regroupement des agents simples dans les groupes. Il sélectionne l'agent_ CH et l'agent_superviseur pour chaque groupe. L'Agent_Sink reçoit toutes les données collectées de l'environnement par les Agents_CH ou les Agents_Assistants (voir le script Agent-Sink).

- **Agent leader (cluster head)**

Cet agent est situé dans le cluster head. L'Agent_CH est l'agent le plus proche de l'agent_sink et qui a plus d'énergie. Il représente le leader. Il est prêt à recevoir et agréger les données des membres de son groupe afin de les envoyer à l'Agent_Sink. Il envoie un message d'alerte à l'agent_Superviseur lorsque son niveau d'énergie baisse faiblement (voir Script Agent_CH).

- **Agent Simple**

Cet agent est situé dans le nœud de capteur. Après son déploiement dans l'environnement, il envoie les informations GPS et le niveau d'énergie à l'Agent_Sink. Sa fonction principale est de détecter l'environnement et d'envoyer les données collectées à son Agent_CH (voir Script Simple_Agent).

- **Agent de contrôle des CHs (Agent_Superviseur)**

Afin de réaliser le protocole de routage, la stratégie de gestion des CHs est intégrée. Cette stratégie est fournie par l'Agent_Superviseur. La principale fonction de cet agent est d'assurer une surveillance des CHs lorsque leur niveau d'énergie diminue en sélectionnant un nœud assistant qui est l'agent le plus proche de l'Agent_cluster head et qui a une énergie résiduelle supérieure (voir Script Agent-Superviseur).

- **Agent_ Assistant**

Cet agent est le plus proche de l'Agent_cluster head dans chaque groupe et qui a une énergie résiduelle supérieure à celle de l'agent cluster head. Lorsque cet agent reçoit une information de l'agent superviseur, il devient responsable de la communication intra-cluster et achemine les données de son groupe vers l'agent_sink (voir le script Agent-Assistant).

Algorithm 1 Agent-Sink script

BEGIN

si (receive [X,Y,E] from all actives Simple_Agents) **then**

-dividing the sensing environment

- regrouping Simple-Agents into balanced groups by a circular dynamic division

- select the Agent-Leader

- select the Agent-Supervisor

end if

receive data from Agent_CH

if(receive request-id-assistant from Agent_Supervisor) then

send id-assistant to the Agent_Supervisor

endif

END

Algorithm 2 Agent_CH script

BEGIN

while (round is not finished) **then**

If(energy level dips low) **then**

send alert-message to the Agent-Supervisor

endif

if (receive data from Simple-Agents of group) **then**

aggregate data

if (receive message-info from the Agent-supervisor) **then**

send aggregated data to the Agent-Assistant

else

send data to the Agent-Sink

end if

endif

endwhile

END

Algorithm 3 Simple-Agent script

BEGIN

send [X,Y, E] to the Agent-Sink

while (state is active) **then**

sensing and collect data from environment

send collected data to the next hop (intermediate node) or Agent_CH

endwhile

END

Algorithm 4 Agent_Supervisor script

BEGIN

if (receive alert-message from the Agent_CH) **then**

send (request id-assistant) to the Agent_Sink

if (receive id-assistant from Agent_Sink) **then**

send message-info to the assistant : ”you are the Agent-Assistant of the Agent-Leader”

send message-info to the Agent_CH: “id-assistant is your assistant”

Algorithm 5 Agent-Assistant script

BEGIN

if (receive (message-info from the Agent_Supervisor) **then**

if (receive data from the Agent_CH) **then**

send data to the Agent_Sink

end if

END

4.8 **Simulation et évaluation des performances**

Après avoir terminé la conception et la modélisation de notre proposition, nous entamons la dernière phase qui concerne la simulation du système proposé. Dans cette section, nous effectuons une étude de simulation pour démontrer l'aspect pratique et l'efficacité de notre protocole de clustering pour les RCSFs nommé Improved-ACORP. Afin de valider les performances de ce nouvel algorithme, une comparaison avec les algorithmes LEACH-C, Improved-LEACH (Impro-LEACH) et DE-LEACH a été réalisée. Nous évaluons les performances de notre protocole en termes d'efficacité énergétique, de durée de vie du réseau, du nombre des nœuds vivants lors du déroulement du protocole et de la quantité des données envoyées à la station de base. Le protocole est implémenté en utilisant le simulateur NS2. Le tableau 4-3 présente les paramètres de simulation.

Tableau 4-3 Les paramètres de simulation

Parameters	**Values**
Length of network	(100 m× 100 m)
Network size	100 sensors
Initial Energy	2 joules
Packet size	25 Bytes
Minimum Energy	0.001 joule
Maximum duration of the simulation	3600 seconds
Max	25
Min	10

4.8.1 **Durée de vie du réseau**

La figure 4-39 représente la durée de vie du réseau en termes de FND, HND et LND pour les protocoles LEACH-C, DE-LEACH, DHCR et Improved-ACORP. Selon cette figure, nous notons que Improved-ACORP a prouvé sa performance par rapport aux autres

protocoles en termes de la durée de vie et ceci est dû à la minimisation du nombre de chemins et de ses longueurs ce qui optimise la consommation d'énergie des capteurs.

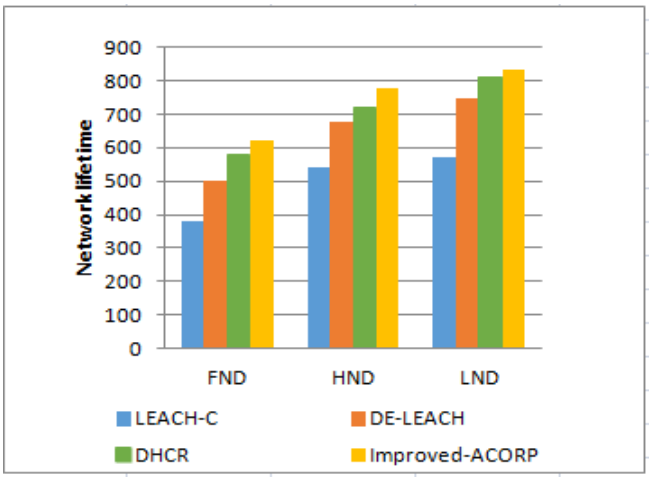

Figure 4-39 La durée de vie du réseau

4.8.2 **Nombre de nœuds vivants par cycle**

La figure 4-40 montre le nombre de nœuds actifs par tour dans le réseau. Comme nous pouvons le voir à partir de cette figure, la disparition des nœuds vivants diminue lentement dans l'algorithme Improved-ACORP proposé par rapport aux algorithmes LEACH-C, DE-LEACH et DHCR. Par conséquent, la stabilité du réseau et la durée de vie du réseau sont bien meilleures par rapport à DHCR, DE-LEACH et LEACH-C.

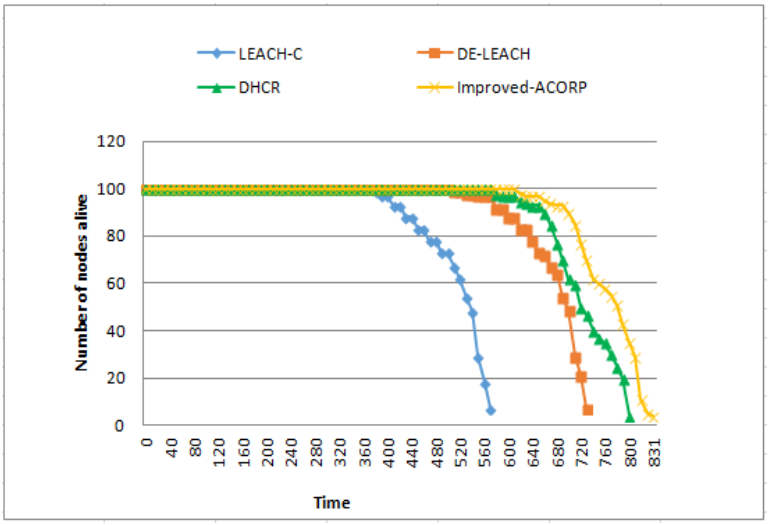

Figure 4-40 Le nombre de nœuds vivants

4.8.3 **Energie consommée**

La figure 4-41 montre l'énergie consommée par les nœuds du réseau. Comme le montre cette figure, il est clair que la consommation d'énergie est répartie plus uniformément entre

tous les nœuds de l'algorithme proposé par rapport aux algorithmes LEACH-C DE-LEACH et DHCR.

Ceci peut s'expliquer par le fait que l'algorithme proposé fournit un meilleur clustering des nœuds en termes d'équivalence et d'équilibrage. Par conséquent, la performance du réseau a été grandement améliorée.

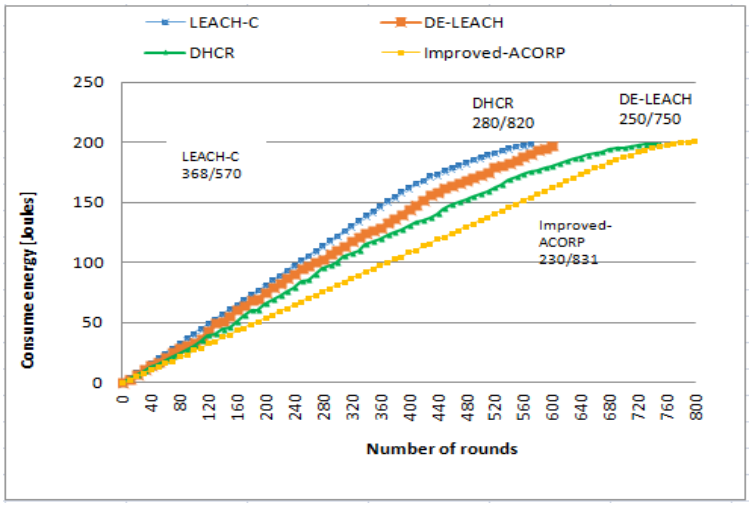

Figure 4-41 L'énergie consommée

4.8.4 **La quantité de données reçues par la BS**

La figure 4-42 montre la quantité de données reçues par la station de base. Comme le montre cette figure, la quantité de données reçues par la station de base est beaucoup plus grande dans le protocole Improved-ACORP proposé par rapport à DHCR, DE-LEACH et LEACH-C.

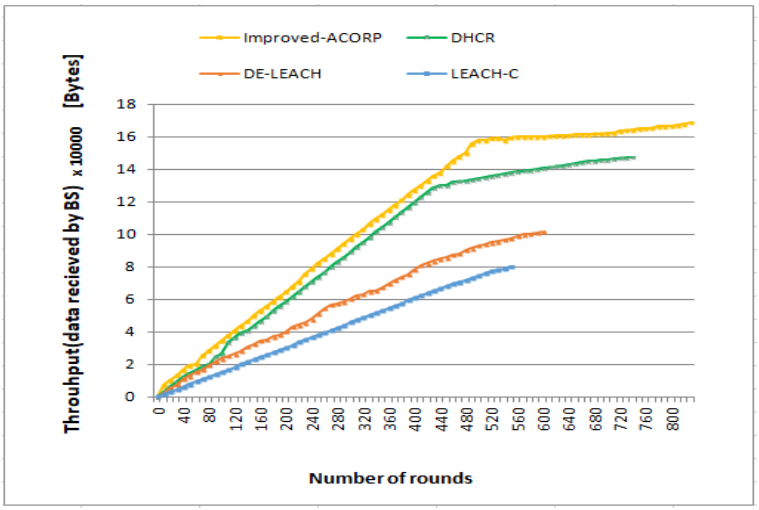

Figure 4-42 La quantité de données reçus par la SB

Le tableau 4.4 résume les résultats obtenus. Nous remarquons que Improved-ACORP surpasse les autres algorithmes en termes de quantité de données reçues par la BS et durée de vie du réseau.

Tableau 4-4 Les résultats de simulation

	First node died in round	Last node died in round	Total quantity of data sent to the BS(Bytes)
LEACH-C	382	569	70833
DE-LEACH	511	732	101434
DHCR	580	820	139677
Improved-ACORP	620	831	169566

4.9 **Conclusion**

Dans ce chapitre nous avons proposé deux contributions. La première concernant le protocole de clustering pour les réseaux de capteur sans fils basé sur le centre de densité des nœuds WDC-LEACH-C. La deuxième concerne un protocole de routage hiérarchique baptisé IMPROVED-ACORP qui est une amélioration du protocole ACORP, où une stratégie de gestion d'énergie est intégrée afin de conserver l'énergie des CH et de prolonger la durée de vie du réseau. Ces propositions ont été comparées avec d'autres travaux et les résultats obtenus sont encourageants car ils montrent que l'on obtient de meilleures performances par rapport aux autres travaux en termes d'énergie, de durée de vie et de quantité de données reçues la station de base.

Conclusion générale et perspectives

Les systèmes distribués sont devenus un domaine de recherche très actif dû à la grande importance de leurs applications et vus l'expansion très importante des réseaux de communication. Les systèmes distribués ont, ainsi, été largement étudiés et la répartition a été rapidement considérée comme une option naturelle pour répondre à une forte demande de : calcul, simulation, modélisation, base de données et serveurs multimédia.

Le domaine de réseaux de capteurs soulève d'importantes problématiques de recherche en termes d'exploitation des données récoltées, la conservation d'énergie et la prolongation de la durée de vie du réseau ainsi que la facilité des échanges et de la coopération entre les différents composants du système. Les travaux présentés dans cette thèse s'inscrivent dans ces cadres.

Dans la première partie de cette thèse consacrée à l'établissement d'un état de l'art de la question, nous avons entamé une étude générale des systèmes distribués en présentant les différents concepts de base et les techniques de communication ensuite avons détaillé les algorithmes qui modélisent la coopération et l'échange dans les différentes opérations effectuées dans un système distribué. Cette étude a mis en lumière les différentes facettes des réseaux de capteurs qui sont un type des réseaux qui apporte des bénéfices aux systèmes distribués ainsi que leurs caractéristiques intrinsèques où la limitation en énergie est la plus critique et considérée comme un défi majeur en relation avec leurs vulnérabilités aux différentes attaques. La deuxième partie de cette thèse est consacrée aux contributions, nous avons décrit deux propositions dans les protocoles de routage : La première proposition nommée WDC-LEACH-C (Weighted Density Center- Low Energy Adaptive Clstering Hierarchy) concerne un protocole de routage hiérarchique qui adopte un mode de transmission à un seul saut dans lequel les messages sont directement transmis à partir des membres vers les CHs. La pierre angulaire de cette proposition est la sélection de CHs qui est basée sur le critère d'énergie et le centre de densité du cluster. Le protocole WDC-LEACH-C choisit le point le plus proche des nœuds du cluster comme CH. Cette idée réduit considérablement la distance de communication entre les nœuds ce qui évite la perte des données causée par des transmissions sur de longs chemins. La transmission à saut

unique réduit le nombre de chemins établis et réduit la consommation d'énergie en augmentant la durée de vie du réseau. Notre deuxième proposition concerne un protocole de routage hiérarchique intelligent pour les réseaux de capteurs sans fils homogènes auto-organisés basé sur les colonies de fourmis nommé Improved-ACORP (Ant Colony Optimization Protocol). L'idée principale de Improved-ACORP est basée sur ACORP qui est inspiré du comportement des colonies de fourmis réelles. Improved-ACORP a introduit le critère d'énergie dans la sélection des chemins vers le CH et il intègre, ainsi, une stratégie de gestion d'énergie de CHs. La stratégie HCMS développée ici intervient lorsque le niveau d'énergie d'un CH a atteint un seuil critique. La conservation d'énergie ainsi que la maximisation de la durée de vie du réseau ont été les buts principaux pour la conception de nos protocoles.

Perspectives :

Les systèmes distribués et les réseaux de capteurs constituent un axe de recherche très fertile et ont de nombreuses perspectives d'application dans des domaines très variés : Il reste encore de nombreux problèmes à résoudre dans ce domaine afin de pouvoir les utiliser dans les conditions réelles. En outre, chaque application a ses propres contraintes. De ce fait, la conception d'un réseau de capteurs est une tâche très difficile parce qu'elle devra combiner les contraintes propres aux systèmes distribués et aux systèmes embarqués. Pour cette raison, les perspectives ouvertes par ces travaux sont nombreuses et variées.

Dans l'axe des protocoles de routage, plusieurs critères peuvent être considérés pour améliorer l'efficacité des chemins établis. Les algorithmes d'ordonnancement jouent un rôle très important dans la conservation d'énergie s'ils sont intégrés dans les protocoles de routage

D'autres axes de recherches sont très importants pour construire des réseaux de capteurs sans fils en conditions réelles comme le passage à l'échelle et la sécurité de ces réseaux.

Bibliographie

[1] Coulouris, G. F., Dollimore, J., & Kindberg, T. (2005). *Distributed systems: concepts and design*. pearson education.

[2] Tanenbaum, A. S., & Besse, L. (1994). *Systèmes d'exploitation: systèmes centralisés, systèmes distribués*. InterEditions.

[3] Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, *21*(7), 558-565.

[4] Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed systems: principles and paradigms*. Prentice-Hall.

[5] Jia, W., & Zhou, W. (2004). *Distributed Network Systems: From Concepts to Implementations* (Vol. 15). Springer Science & Business Media.

[6] Xiao, X., Xie, G., Xu, C., Fan, C., Li, R., & Li, K. (2018). Maximizing reliability of energy constrained parallel applications on heterogeneous distributed systems. *Journal of computational science*, *26*, 344-353. http://dx.doi.org/10.1016/j.jocs.2017.05.002

[7] Sari, A., & Rahnama, B. (2013, November). Addressing security challenges in WiMAX environment. In *Proceedings of the 6th International Conference on Security of Information and Networks* (pp. 454-456). ACM.http://dx.doi.org/10.1145/2523514.2523586

[8] Koren, I., & Krishna, C. M. (2010). *Fault-tolerant systems*. Elsevier.

[9] Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed systems: principles and paradigms*. Prentice-Hall.

[10] Sarr, I. (2010). *Routage des transactions dans les bases de données à large échelle* (Doctoral dissertation, Université Pierre et Marie Curie (Paris VI))

[11] Taboada, G. L., Touriño, J., & Doallo, R. (2012). F-MPJ: scalable Java message-passing communications on parallel systems. *The Journal of Supercomputing*, *60*(1), 117-140.. https://doi.org/10.1007/s11227-009-0270-0

[12] Waldo, J. (1998). Remote procedure calls and java remote method invocation. *IEEE concurrency*, *6*(3), 5-7.

[13] Hadim, S., & Mohamed, N. (2006). Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE distributed systems online*, (3), 1.

[14] Valls, M. G., & Val, P. B. (2014). Comparative analysis of two different middleware approaches for reconfiguration of distributed real-time systems. *Journal of Systems Architecture*, *60*(2), 221-233.."

[15] Alho, K. (1998). A comparison of corba, dcom and rmi. *Helsinki University of Technology*.

[16] Plášil, F., & Stal, M. (1998). An architectural view of distributed objects and components in CORBA, Java RMI and COM/DCOM. *Software-Concepts & Tools*, *19*(1), 14-28.

[17] Li, K., & Hudak, P. (1989). Memory coherence in shared virtual memory systems. *ACM Transactions on Computer Systems (TOCS)*, *7*(4), 321-359.

[18] Bernard, G. (1999, June). Applicabilité et performances des systèmes d'agents mobiles dans les systèmes répartis. In *Première Conférence Française en Systèmes d'Exploitation (CFSE'1), Rennes, France* (Vol. 24).

[19] Zhang, Y. H., Lin, Y., Gong, Y. J., & Zhang, J. (2015, December). Particle swarm optimization with minimum spanning tree topology for multimodal optimization. In *2015 IEEE Symposium Series on Computational Intelligence* (pp. 234-241). IEEE.

[20] Yang, W., & Fu, M. (2017). A proportional integral estimator-based clock synchronization protocol for wireless sensor networks. *ISA transactions*, *71*, 148-160.

[21] Freris, N. M., Graham, S. R., & Kumar, P. R. (2011). Fundamental limits on synchronizing clocks over networks. *IEEE Transactions on Automatic Control*, *56*(6), 1352-1364.

[22] Gusella, R., & Zatti, S. (1989). The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3 BSD. *IEEE transactions on Software Engineering*, *15*(7), 847-853.

[23] Cristian, F., & Fetzer, C. (1994, October). Probabilistic internal clock synchronization. In *Proceedings of IEEE 13th Symposium on Reliable Distributed Systems* (pp. 22-31). IEEE.

[24] Lamport, L. (2001). Paxos made simple. *ACM Sigact News*, *32*(4), 18-25.

[25] Mattern, F. (1988). *Virtual time and global states of distributed systems* (pp. 215-226). Univ., Department of Computer Science.,"

[26] Chandy, K. M., & Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems (TOCS)*, *3*(1), 63-75.

[27] Lejeune, J., Arantes, L., Sopena, J., & Sens, P. (2013, January). Un algorithme équitable d'exclusion mutuelle distribuée avec priorité. In *9ème Conférence Française sur les Systèmes d'Exploitation (CFSE'13), Chapitre français de l'ACM-SIGOPS, GDR ARP*.

[28] Ricart, G., & Agrawala, A. K. (1980). *An Algorithm for Mutual Exclusion in Computer Networks* (No. TR-869). MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE.

[29] Maekawa, M. (1985). A√ N algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems (TOCS)*, *3*(2), 145-159.

[30] Ricart, G., & Agrawala, A. K. (1983). Author's response to 'On mutual exclusion in computer networks' by Carvalho and Roucairol. *Communications of the ACM*, *26*(2), 147-148.

[31] Boudrahem, M., Bouiche, K., & Sadi, M. (2017). *Implémentation d'un algorithme d'élection de leader dans les réseaux ad hoc*(Doctoral dissertation, Université abderrahmane mira béjaia).

[32] Chang, E., & Roberts, R. (1979). An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Communications of the ACM*, *22*(5), 281-283.

[33] Fischer, M. J., Lynch, N. A., & Paterson, M. S. (1982). *Impossibility of distributed consensus with one faulty process*(No. MIT/LCS/TR-282). MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE.

[34] Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *4*(3), 382-401.

[35] Fowler, M. (2014). *U.S. Patent No. 8,838,534*. Washington, DC: U.S. Patent and Trademark Office.

[36] Chandy, K. M., & Misra, J. (1984). The drinking philosopher's problem. *ACM transactions on programming languages and systems*, *6*(4), 632-646.

[37] Arévalo, S., Anta, A. F., Imbs, D., Jiménez, E., & Raynal, M. (2015). Failure detectors in homonymous distributed systems (with an application to consensus). *Journal of Parallel and Distributed Computing*, *83*, 83-95.

[38] Alippi, C., Anastasi, G., Galperti, C., Mancini, F., & Roveri, M. (2007, October). Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems* (pp. 1-6). IEEE.

[39] Braginsky, D., & Estrin, D. (2002, September). Rumor routing algorthim for sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications* (pp. 22-31). ACM.

[40] Benaziz, B. (2017). *Un Modèle Agent Situé Pour Les Réseaux De Capteur* (Doctoral dissertation, Université Mohamed Khider–BISKRA).

[41] Brustoloni, J. C. (1991). *Autonomous agents: Characterization and requirements*. School of Computer Science, Carnegie Mellon University.

[42] Anastasi, G., Conti, M., Passarella, A., & Pelusi, L. (2008). Mobile-relay forwarding in opportunistic networks. *Adaptation and Cross Layer Design in Wireless Networks*, *21*, 389.

[43] Calle Torres, M. G. (2006). *Energy consumption in wireless sensor networks using GSP* (Doctoral dissertation, University of Pittsburgh).

[44] Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., & Culler, D. E. (2002). SPINS: Security protocols for sensor networks. *Wireless networks*, *8*(5), 521-534.

[45] Intanagonwiwat, C., Govindan, R., & Estrin, D. (2000, August). Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*(pp. 56-67). ACM.

[46] Ye, F., Chen, A., Lu, S., & Zhang, L. (2001). A scalable solution to minimum cost forwarding in large sensor networks. In *Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No. 01EX495)* (pp. 304-309). IEEE.

[47] Saad, C. (2008). *Quelques contributions dans les réseaux de capteurs sans fil: Localisation et Routage* (Doctoral dissertation, Université d'Avignon).

[48] Li, X., Li, D., Wan, J., Vasilakos, A. V., Lai, C. F., & Wang, S. (2017). A review of industrial wireless networks in the context of Industry 4.0. *Wireless networks*, *23*(1), 23-41.

[49] Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. (2000, January). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd annual Hawaii international conference on system sciences* (pp. 10-pp). IEEE.

[50] Afsar, M. M., & Tayarani-N, M. H. (2014). Clustering in sensor networks: A literature survey. *Journal of Network and Computer Applications*, *46*, 198-226.

[51] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications magazine*, *40*(8), 102-114.

[52] Zhou, Y., Fang, Y., & Zhang, Y. (2008). Securing wireless sensor networks: a survey. *IEEE Communications Surveys & Tutorials*, *10*(3), 6-28.

[53] Rajagopalan, R., & Varshney, P. K. (2006). Data aggregation techniques in sensor networks: A survey.

[54] Pantazis, N. A. (2007). Survey on power control issues in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, *9*(4), 86-107.

[55] Ehsan, S., & Hamdaoui, B. (2012). A survey on energy-efficient routing techniques with QoS assurances for wireless multimedia sensor networks. *IEEE Communications Surveys & Tutorials*, *14*(2), 265-278.

[56] Dong, Q., & Dargie, W. (2013). A survey on mobility and mobility-aware MAC protocols in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, *15*(1), 88-100.

[57] Gherbi, C., Aliouat, Z., & Benmohammed, M. (2017). A survey on clustering routing protocols in wireless sensor networks. *Sensor Review*, *37*(1), 12-25.

[58] Zeb, A., Islam, A. M., Zareei, M., Al Mamoon, I., Mansoor, N., Baharun, S., ... & Komaki, S. (2016). Clustering analysis in wireless sensor networks: the ambit of performance metrics and schemes taxonomy. *International Journal of Distributed Sensor Networks*, *12*(7), 4979142.

[59] Sucasas, V., Radwan, A., Marques, H., Rodriguez, J., Vahid, S., & Tafazolli, R. (2016). A survey on clustering techniques for cooperative wireless networks. *Ad Hoc Networks*, *47*, 53-81.

[60] Ahmad, H., & Kohli, N. (2015, November). A survey on various static and mobile base stations and wireless charging of sensor nodes in WSNs. In *2015 International Conference on Communication Networks (ICCN)* (pp. 16-22). IEEE.

[61] Zanjireh, M. M., & Larijani, H. (2015, May). A survey on centralised and distributed clustering routing algorithms for WSNs. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)* (pp. 1-6). IEEE.

[62] Tiwari, T., & Roy, N. R. (2015, May). Heirarchical clustering in heterogeneous wireless sensor networks: A survey. In *International Conference on Computing, Communication & Automation* (pp. 1385-1390). IEEE.

[63] Jain, S. R., & Thakur, N. V. (2015). Overview of cluster based routing protocols in static and mobile wireless sensor networks. In *Information Systems Design and Intelligent Applications* (pp. 619-626). Springer, New Delhi.

[64] Chouikhi, S., El Korbi, I., Ghamri-Doudane, Y., & Saidane, L. A. (2015). A survey on fault tolerance in small and large scale wireless sensor networks. *Computer Communications*, *69*, 22-37.

[65] Gupta, G. P., & Jha, S. (2018). Biogeography-based optimization scheme for solving the coverage and connected node placement problem for wireless sensor networks. *Wireless Networks*, 1-11.

[66] Zahedi, A. (2018). An efficient clustering method using weighting coefficients in homogeneous wireless sensor networks. *Alexandria Engineering Journal*, *57*(2), 695-710.

[67] Khan, F. U., Shah, I. A., Jan, S., Khan, I., & Mehmood, M. A. (2015, December). Fuzzy logic based cluster head selection for homogeneous wireless sensor networks. In *2015 International Conference on Open Source Systems & Technologies (ICOSST)*(pp. 41-45). IEEE.

[68] Farouk, F., Rizk, R., & Zaki, F. W. (2014). Multi-level stable and energy-efficient clustering protocol in heterogeneous wireless sensor networks. *IET Wireless Sensor Systems*, *4*(4), 159-169.

[69] Zhang, W., Li, L., Han, G., & Zhang, L. (2017). E2hrc: An energy-efficient heterogeneous ring clustering routing protocol for wireless sensor networks. *IEEE Access*, *5*, 1702-1713.

[70] Cenedese, A., Luvisotto, M., & Michieletto, G. (2017). Distributed clustering strategies in industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics*, *13*(1), 228-237.

[71] Heinzelman, W. B., Chandrakasan, A. P., & Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on wireless communications*, *1*(4), 660-670.

[72] Tang, B., Wang, D., & Zhang, H. (2013, October). A centralized clustering geographic energy aware routing for wireless sensor networks. In *2013 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1-6). IEEE.

[73] Wang, J., Gao, Y., Yin, X., Li, F., & Kim, H. J. (2018). An Enhanced PEGASIS Algorithm with Mobile Sink Support for Wireless Sensor Networks. *Wireless Communications and Mobile Computing*, *2018*.

[74] Wang, T., Li, Y., Wang, G., Cao, J., Bhuiyan, M. Z. A., & Jia, W. (2017). Sustainable and efficient data collection from WSNs to cloud. *IEEE Transactions on Sustainable Computing*.

[75] Roy, S., & Das, A. K. (2014, August). Energy efficient cluster based routing protocol (EECBRP) for Wireless Sensor Network. In *2014 First International Conference on Networks & Soft Computing (ICNSC2014)* (pp. 25-29). IEEE.

[76] Kumar, S., Prateek, M., Ahuja, N. J., & Bhushan, B. (2014). De-leach: Distance and energy aware leach. *arXiv preprint arXiv:1408.2914*.

[77] Xu, Z., Chen, L., Chen, C., & Guan, X. (2016). Joint clustering and routing design for reliable and efficient data collection in large-scale wireless sensor networks. *IEEE Internet of Things Journal*, *3*(4), 520-532.

[78] Zhao, F., Xu, Y., & Li, R. (2012). Improved LEACH routing communication protocol for a wireless sensor network. *International Journal of Distributed Sensor Networks*, *8*(12), 649609.

[79] Chen, H., Zhang, C., Zong, X., & Wang, C. (2013). LEACH-G: an Optimal Cluster-heads Selection Algorithm based on LEACH. *journal of software*, *8*(10), 2660-2668.

[80] Rezaei, E., Baradaran, A. A., & Heydariyan, A. (2016). Multi-hop routing algorithm using Steiner points for reducing energy consumption in wireless sensor networks. *Wireless Personal Communications*, *86*(3), 1557-1570.

[81] Papithasri, K., & Babu, M. (2016, January). Efficient multihop dual data upload clustering based mobile data collection in Wireless Sensor Network. In *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)* (Vol. 1, pp. 1-6). IEEE.

[82] Kandukuri, S., Murad, N., & Lorion, R. (2015, September). A single-hop clustering and energy efficient protocol for wireless sensor networks. In *2015 Radio and Antenna Days of the Indian Ocean (RADIO)* (pp. 1-2). IEEE.

[83] Liao, Y., Qi, H., & Li, W. (2013). Load-balanced clustering algorithm with distributed self-organization for wireless sensor networks. *IEEE sensors journal*, *13*(5), 1498-1506.

[84] Wang, S. S., & Chen, Z. P. (2013). LCM: a link-aware clustering mechanism for energy-efficient routing in wireless sensor networks. *IEEE sensors journal*, *13*(2), 728-736.

[85] Guiloufi, A. B. F., Nasri, N., & Kachouri, A. (2016). An energy-efficient unequal clustering algorithm using 'Sierpinski Triangle'for WSNs. *Wireless Personal Communications*, *88*(3), 449-465.

[86] Liu, T., Li, Q., & Liang, P. (2012). An energy-balancing clustering approach for gradient-based routing in wireless sensor networks. *Computer Communications*, *35*(17), 2150-2161.

[87] Morley, C., Hugues, J., & Leblanc, B. (2000). *UML pour l'analyse d'un système d'information-le cahier des charges du maître d'ouvrage* (No. BOOK). Dunod.