République Algérienne Démocratique et Populaire
Ministre de l'Enseignement Supérieur de la Recherche Scientifique

Université A/Mira de Béjaïa
Faculté des Sciences Exactes Département d'Informatique

Mémoire de fin de Cycle

En vue de L'obtention d'un diplôme de Master professionnel en Informatique

Option : **A**dministration et **S**écurité des **R**éseaux

Thème

Adaptation du simulateur multi-agents NetLogo pour modéliser et simuler le routage dans un réseau mobile Ad hoc

[Couche Réseau , Protocole AODV]

Réalisé par :

M*lle*. Bouchebbah Razika.

M*lle*. Djermani Hanane.

Soutenu le 17 septembre 2020

Devant le jury composé de :

Président	**M***r*	M. MOKETFI	M.A.A	U. A/Mira Bejaia.
Examinateur	**M***me*	S.SABRI	M.C.B	U. A/Mira Bejaia.
Encadrant	**M***me*	A. HOUHA	M.A.A	U. A/Mira Bejaia.

Promotion : 2019/2020

Remerciement

Avant tout nous remercions dieu pour la santé, la volonté et la patience qui nous ont accompagnées durant le cursus universitaire afin de réaliser ce modeste travail.

On tient à exprimer nos vifs remerciements et notre sincère gratitude à notre encadrant « M^{me} Amel Houha» de son suivi, ses conseils avisés, sa gentillesse, son encouragement, sa disponibilité et sa patience. Nous vous témoignons ici toute notre reconnaissance.

Nos remerciements chaleureux et anticipés vont aux membres du jury qui ont consacré une partie de leur temps pour examiner et juger notre travail.

Nous tenons aussi à remercier tous les enseignants de notre département pour leurs efforts fournis durant notre cursus universitaire.

Nos remerciements vont également à tous ceux qui nous ont aidées de loin comme de près à l'aboutissement de ce travail, soit avec leur support, leur amitié ou leur amour.

Nous passons également notre très vif remerciement à nos camarades de la promotion 2020 : Dahbia, Massilia, Dina, Zohra, AbdElHaq, Sofiane …etc.

Dédicace

A

Ma chère mère & Mon cher père

Vous êtes pour moi une source de vie car sans vos sacrifices, votre tendresse et votre Affection je ne pourrais jamais arriver jusqu'au bout. Je me réjouis de cet amour filial. Que Dieu vous garde afin que votre regard puisse suivre ma destiné.

A

Mes grands-parents & Mes frères & Ma sœur

*(**T**oufik, **K**hellaf, **S**abrina)*

Pour leurs disponibilités à entendre mes frustrations et les sources de mon stresse

Avec mes souhaits de bonheur et de réussit dans leurs vies.

A

Toute ma famille

A

Tous mes amis et mes camarades

En témoignage de notre amitié sincère.

*A ma chère amie **W**ISSAM, je te remercie pour ton amitié chère à mon cœur, et je te souhaite tout le bonheur du monde. Toute mon affection pour ton admirable famille, que je remercie beaucoup.*

*A ma très chère binôme **H**ANANE, pas de mots pour exprimer mes remerciements envers toi, pour ton soutien, tes encouragements que tu m'as donné et ta patience avec moi pendant cette année.*

A

Tous ceux que j'aime et qu'ils m'aiment

Qu'ils trouvent dans ce travail l'expression de mes sentiments les plus affectueux.

Je dédié ce travail.

Razika

Dédicace

Je dédie le fruit de ce modeste travail accompagné d'un profond amour :

***A** celle qui m'a arrosé de tendresse et d'espoirs, à la source d'amour incessible,*

*à la mère des sentiments fragile qui ma bénie par ces prières **MA Douce MAMAN**.*

***A** mon support dans la vie, qui m'a appris m'a supporté et m'a dirigé Vers la gloire mon*

***Adorable PAPA.** Aucune dédicace ne saurait exprimer, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Ce travail est le fruit de vos sacrifices que vous avez consentis pour mon éducation et ma formation.*

J'espère que vous êtes fière de votre fille.

***A** mes chers frères : **N**abil, **N**assim et **M**ouloud pour leur soutiens et encouragements*

***M**erci d'être là pour moi. Je vous souhaite un avenir plein de joie, de bonheur, de réussite, et de sérénité .Je vous exprime à travers ce travail mes sentiments de fraternité et d'amour.*

***A** mes belles sœurs et en particulier à mes bébés d'amour mes neveux :*

*à Ma princesse **L**eya, à **H**ind et Houda et à mon petit ange **A**ris*

***A** ma très chère binôme **R**AZIKA, pas de mots pour exprimer mes remerciements envers toi, pour ton soutien, tes encouragements que tu m'as donné et ta patience avec moi Pendant ces cinq dernières années.*

***A** toutes les personnes de ma grande famille, et surtout à La mémoire de mes grands-parents, que dieu les accueils dans son vaste paradis*

***A** tous mes cher cousins et cousines : Lila, Kahina, Yousra, Sarah, Alice … et à la mémoire de ma belle cousine fahima.*

Hanane

Table des matières

Table des matiéres I

Liste des figures VIII

Liste des tableaux X

Liste des abréviations XI

Introduction Générale 1

Chapitre I: Système Multi-Agents

I.1. Introduction 4

I.2. Agent et Système Multi-Agents 4

I.2.1. Agent 4

I.2.1.1. Définition 4

I.2.1.2. Propriétés 5

I.2.1.3. Architectures 7

I.2.1.4. Cycle de vie 9

I.2.2. Système Multi-Agents 10

I.2.2.1. Définition 10

I.2.2.2. Propriétés 12

I.2.2.3. Interactions 11

I.2.2.4. Caractéristique 12

I.2.2.5. A + E + I + O : l'approche Voyelles 12

I.2.2.5. Coopération 13

I.2.2.6. Conception 15

I.2.2.7. Notions importantes 15

I.2.2.8. Domaines d'application 16

I.2.2.9. Avantages 17

I.3. Technologie agent et les réseaux Ad Hoc 17

I.3.1. Intérêt ……. 17
I.3.2. Agents mobiles et le routage dans les réseaux Ad hoc 18
I.3.3. SMA et le routage dans les réseaux Ad Hoc 18
I.3.4. Protocoles de routage basé agents 19
Conclusion 21

Chapitre II: Réseaux Ad Hoc et protocoles de routage

II.1. Introduction 21
II.2. Définition du réseau sans fil 21
II.3. Catégories des réseaux sans fil 21
II.3.1 Selon la zone de couverture 22
a. Réseaux personnels sans fil (WPAN) 22
b. Réseaux locaux sans fil (WLAN) 22
c. Réseaux métropolitains sans fil (WMAN) 22
d. Réseaux sans fil étendus (WWAN) 22
II.3.2. Selon l'infrastructure 23
a. Réseaux sans fil avec infrastructure 23
b. Réseaux sans fil sans infrastructure (Ad hoc) 24
II.4. Réseaux Ad Hoc 25
II.4.1. Définition 25
II.4.2. Modélisation 26
II.4.3. Caractéristiques 26
II.4.4. Avantages et inconvénients 29
II.4.5. Applications 29
II.5. Routage dans les réseaux Ad Hoc 30
II.5.1. Définition du routage 30
II.5.2. Définition du routage dans les réseaux Ad Hoc 31
II.5.3. Difficulté 32

II.5.4. Contraintes 32
II.5.5. Routage dans le modèle OSI 33
II.6. Problèmes de conception des protocoles de routage pour les réseaux ad hoc 34
II.6.1. Architecture de routage 34
II.6.2. Prise en charge des liens unidirectionnels 35
II.6.3. Routage de la qualité de service (QoS) 36
II.6.4. Prise en charge de la multidiffusion 36
II.7. Classification des protocoles de routage 37
II.7.1. Critères….. 37
II.7.1.1. Évaluation de topologie, de destination ou de position pour le routage 37
II.7.1.2. Structures de routage (hiérarchique ou plat) 37
II.7.1.3. Routage proactif, réactif ou hybride 39
II.7.1.4. Métriques exploitées dans le routage 39
II.7.1.5. Techniques de routage 39
II.7.1.6. Algorithmes de routage 40
II.7.1.7. Inondation 40
II.7.1.8. Concept du groupe 41
II.7.1.9. Routage uniforme ou non uniforme 41
II.7.2. Classification des protocoles de routage 42
II.7.2.1. Protocoles proactifs 42
II.7.2.2. Protocoles réactifs 44
II.7.2.3. Protocoles hybrides 46
II.7.3. Comparaison des protocoles de routage 48
Conclusion 49

Chapitre III: Simulation et Simulateurs Réseaux

III.1.Introduction 50
III.2. Simulation 50

III.2.1. Définition 50
III.2.2. Types……..... 52
III.2.2.1. Systèmes de simulation discrets 52
III.2.2.2. Systèmes de simulation continus 52
III.2.3. Concepts liés à la simulation..... 52
III.2.4. Avantages et inconvénients 52
III.2.5. Simulation en tant que processus expérimental 53
III.2.6. Simulation à base d'agents 56
III.2.6.1. Définition..... 56
III.2.6.2. Domaines d'application..... 56
III.3. Simulateurs de réseau les plus utilisés 57
III.3.1. Network Simulator-2 (NS2)..... 57
III.3.2. Network Simulateur-3 (NS3)..... 58
III.3.3. OMNET++ (Objective Modular Network Testbed in C++)..... 59
III.3.4.NetLogo.....60
III.4. Etude comparative 63
III.4.1. Tableau de comparaison 63
III.4.2 Comparaison entre les simulateurs NS-2 et OMNET++..... 64
III.4.2.1. Selon les caractéristiques 64
III.4.2.2. Selon deux protocoles 802.11b, 802.15.4 65
III.4.3. Comparaison entre NS-2 et le simulateur NetLogo 68
III.4.4. Discussion..... 71
Conclusion 72
Chapitre IV: Simulation du protocole routage AODV
IV.1. Introduction..... 73
IV.2. Analyse du problème & solution proposée..... 73
IV.2.1. Choix d'AODV..... 74

IV.2.2. Objectif de simulation d'AODV 74
IV.3. Conception 74
IV.3.1. Type du système utilisé 74
IV.3.2. Description de la tâche 74
IV.3.3. Protocole de routage utilisé (AODV) 75
IV.3.4. Modélisation à base de système multi-agents 75
IV.3.5. Modélisation d'un réseau des nœuds à base de graphe 76
IV.4. Mise en œuvre 78
IV.4.1. Simulation 78
IV.4.2. Environnement de simulation NETLOGO 79
IV.4.2.1. Choix 79
IV.4.2.2. Définition 79
IV.4.2.3. Présentation 79
IV.4.3. Simulation du protocole de routage AODV dans NetLogo 80
IV.4.3.1. Pourquoi simuler avec NetLogo 81
IV.4.3.2. Paramètres de simulation 81
IV.4.3.3. Description des tortues utilisées 82
IV.4.3.4. Description de l'environnement 85
IV.4.4. Présentation de notre simulation 86
IV.4.4.1. Description de l'interface de l'application 86
IV.4.4.2. Description de la communication 89
IV.4.4.3. Communication entre les agents-nœuds 92
IV.4.4.4. Implémentation des actions 95
IV.4.5. Interprétation des résultats de la simulation 98
Conclusion 99

Chapitre V: Développement d'Extensions dans NetLogo

V.1. Introduction 100

V.2. Simulateur NetLogo et Extensibilité 100
V.3. Extensions 101
V.3.1. Présentation 101
V.3.2. Développement 102
V.3.2.1. Java 102
V.3.2.2. Scala 102
V.3.3. Plugin sbt 103
V.3.4. Emplacement 103
V.3.5. Utilisation 104
V.3.6. Création d'extensions prédéfinies avec sbt Windows 104
V.4. Conseils de développement d'une extension 105
V.4.1. Instanciation 105
V.4.2. Chemin de classe 105
V.4.3. Débogage d'extensions 106
V.4.4. Tests de langue & performances 106
V.4.5. Niveaux des JAR 106
V.4.6. Documentation du l'extension 106
V.5. Exemple explicatif pour la création d'une extension en Java 106
V.5.1. Création d'un dossier d'extension 107
V.5.2. Développement des primitives 107
V.5.3. Développement d'une classe ClassManager 108
V.5.4. Ecriture d'un fichier manifeste 108
V.5.5. Création d'un fichier JAR 109
V.5.6. Utilisation de l'extension crée dans un modèle 110
V.6. Quelques extensions existantes sur le gestionnaire d'extensions NetLogo 111
V.7. Mise en œuvre de notre extension 113
V.7.1. Objectif de création 113

V.7.2. Etapes de création avec l'IDE Eclipse 113
V.7.3. Représentation 113
V.7.3.1. Extension Table de routage 114
V.7.3.2. Extension Voisins 114
V.7.4. Utilisation 114
V.7.4.1. Extension Table de routage 115
V.7.4.2. Extension Voisins 115
V.7.5. Discussion 116
Conclusion 117
Conclusion Générale 118
Bibliographie 120
Annexes 127

Liste des Figures

Figure I. 1: Représentation classique d'un agent et de son environnement 5
Figure I. 2: Agent réactif. 7
Figure I. 3: Agent cognitif. 8
Figure I. 4: Agent hybride. 9
Figure I. 5: Etats d'un agent. 9
Figure I. 6: Représentation schématique d'un système multi-agents. 11
Figure I. 7: Présentation de niveau macro et micro d'un SMA 15
Figure II. 1: Classification des réseaux sans fil 22
Figure II. 2: Classement des réseaux sans fil selon la portée 23
Figure II. 3: Mode infrastructure et mode Ad Hoc 23
Figure II. 4: Réseau sans fil avec infrastructure 24
Figure II. 5: Modèle d'un réseau sans fil sans infrastructure 25
Figure II. 6 : Modélisation d'un réseau Ad Hoc 26
Figure II. 7: Changement de la topologie d'un réseau Ad Hoc 27
Figure II. 8: Durée de vie de batterie des nœuds 27
Figure II. 9: Applications de secours des réseaux Ad Hoc 30
Figure II. 10: Chemin utilisé dans le routage entre la source et la destination 31
Figure II. 11: Illustration du routage unicast, multicast et broadcast 31
Figure II. 12: Modèle OSI. 33
Figure II. 13: Routage hiérarchique 38
Figure II. 14: Routage plat 38
Figure II. 15: Mécanisme d'inondation 40
Figure II. 16: Décomposition du réseau en groupe 41
Figure II. 17: Schéma récapitulatif des critères de classification des protocoles de routage. 42
Figure II. 18: Classification des protocoles de routage. 42
Figure II. 19: Avantage de l'utilisation des MPR 44
Figure III. 1: Simulation informatique selon MICHEL 51
Figure III. 2: Etapes de conception d'une simulation informatique 55
Figure III. 3: Description architecturale du simulateur NS2 58
Figure III. 4: Interface du simulateur NS-3 59
Figure III. 5: Lancement du simulateur OMNeT++ 60
Figure III. 6: Logo du simulateur NetLogo v 6.1.1. 61

Figure IV. 1: Environnement d'un nœud. 76
Figure IV. 2: Modélisation d'un réseau de nœud. 76
Figure IV. 3: Changement de la topologie des réseaux Ad hoc. 77
Figure IV. 4: Phases principales et initiales du travail. 78
Figure IV. 5: Lancement du simulateur NetLogo v 6.1.1. 79
Figure IV. 6: Interface du NetLogo. 80
Figure IV. 7: Tortues utilisées. 85
Figure IV. 8: Environnement de simulation. 85
Figure IV. 9: Interface de notre application. 86
Figure IV. 10: Plots de nombre de paquets envoyés et reçus dans notre application. 88
Figure IV. 11: Plot de l'énergie consommée dans notre application. 88
Figure IV. 12: Description de communication. 90
Figure IV. 13: Diagramme d'états et actions de communication du protocole AODV. 92
Figure IV. 14: Format du paquet de contrôle RREQ. 93
Figure IV. 15: Format du paquet de contrôle RREP. 93
Figure IV. 16: Format de paquet de contrôle RERR. 93
Figure IV. 17: Format de la table de routage. 94
Figure IV. 18: Communication entre agents-nœuds (Emission du paquet de données)…….94
Figure V. 1: Classe IntegerList du reporter first-n-integers 107
Figure V. 2: Classe Manager du reporter first-n-integers 108
Figure V. 3: Fichier manifest du reporter first-n-integers. 109
Figure V. 4: Schéma explicatif des étapes de création d'une extension. 111
Figure V. 5: Exemple d'utilisation de l'extension Table de routage. 115
Figure V. 6: Exemple d'utilisation de l'extension Voisins. 116

Liste des tableaux

Tableau II. 1: Avantages et inconvénients des réseaux Ad Hoc. 29
Tableau II. 2: Tableau comparatif des différents protocoles de routage Ad Hoc 48
Tableau III. 1: Avantages et inconvénients de la simulation. 53
Tableau III. 2: Avantages et limites des simulateurs 63
Tableau III. 3: Comparaison entre NS2 et OMNET++ 67
Tableau III. 4: Comparaison entre NS-2 et NetLogo. 70
Tableau IV. 1: Description de paramètres de simulation dans notre travail. 81
Tableau IV. 2: Paramètres de notre travail avec NetLogo. 82
Tableau IV. 3: Phases d'AODV. 91

Liste des Abréviations

SMA	**S**ystème **M**ulti-**A**gents
MARP	**M**ulti-**A**gent **R**outing **P**rotocol
MWAC	**M**ulti-**W**ireless-**A**gent **C**ommunication
WPAN	**W**ireless **P**ersonal **A**rea **N**etwork
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WMAN	**W**ireless **M**etropolitan **A**rea **N**etwork
WWAN	**W**ireless **W**ide **A**rea **N**etwork
MANET	**M**obile **A**d Hoc **N**ETWORK
OSI	**O**pen **S**ystem **I**nterconnexion
OLSR	**O**ptimized **L**ink **S**tate **R**outing
MPR	**M**ulti **P**oint **R**elay
DSDV	**D**estination-**S**equenced **D**istance-**V**ector
DSR	**D**ynamic **S**ource **R**outing
AODV	**A**d Hoc **O**n demand **D**istance **V**ector
RREQ	**R**oute **R**equest
RREP	**R**oute **R**eply
RERR	**R**oute **E**rror
ZRP	**Z**one **R**outinier **P**rotocol
SBA	**S**imulation **b**ased-**A**gent
NS2	**N**etwork **S**imulator-2
NS3	**N**etwork **S**imulateur-3
OMNET++	**O**bjective **M**odular **N**etwork **T**estbed in C++
JAR	**J**ava **A**rchive

Introduction Générale

Introduction Générale

Les systèmes informatiques n'ont pas arrêté de s'accroitre ces dernières décennies. Leurs utilisations, implémentations et déploiements deviennent de plus en plus vastes et complexes. Il est devenu essentiel de gérer de plus en plus l'interactivité et la mobilité dans leurs utilisations les plus répandues. Ainsi, l'apparition des **S**ystèmes **M**ulti-**A**gents, que nous désignons par la suite **SMA**, apporte une nouvelle dimension au concept de modélisation utilisé pour représenter une application du monde réel avec un degré approprié de complexité et de dynamisme. Cette discipline est à la connexion de plusieurs domaines dont les systèmes distribués, le génie logiciel et en particulier l'Intelligence Artificielle Distribuée (IAD). Elle est particulièrement intéressante à utiliser dans des environnements de nature distribué et dynamique comme les réseaux ad hoc. Un SMA semble donc approprié pour gérer le routage dans les réseaux ad hoc et obtenir de meilleures performances.

La notion du réseau sans infrastructure où réseau **Ad Hoc** peut être définie comme un ensemble d'entités mobiles interconnectées par une technologie sans fil formant un réseau temporaire sans l'aide de toute administration ou de tout support fixe.

La méthode adoptée dans le routage, doit offrir le meilleur acheminement des données en respect des différentes métriques de coûts utilisées. Pour cela les réseaux ad hoc utilisent des protocoles de routages spécifiques, chaque protocole essaye de maximiser les performances du réseau en minimisant le délai de livraison des paquets, l'utilisation de la bande passante et la consommation d'énergie. Ces protocoles se divisent en trois catégories proactives, réactives et hybrides. Parmi les protocoles de routage des réseaux ad hoc le plus connus de la famille des protocoles réactives, nous trouvons le protocole **AODV** (**A**d hoc **O**n demand **D**istance **V**ector) qui est le protocole le plus utilisé dans le domaine des réseaux Ad hoc .Au vu de ses caractéristiques, ce protocole est devenu très connu et beaucoup de travaux ont déjà été réalisés à son propos. Il est tout à fait adapté aux réseaux mobiles Ad Hoc par sa prise en charge de la mobilité des nœuds dans le réseau, un autre avantage de ce protocole est sa simplicité. Ensuite son ancienneté et sa maturité, AODV existe depuis longtemps.

Ce travail rentre dans le cadre de l'étude d'un simulateur Multi-Agents **NetLogo** largement utilisé dans divers domaines et fournit une bibliothèque de modèles riche. Excepte dans le domaine du réseau, son utilisation apparait actuellement en émergence et il est largement apprécié par les chercheurs en raison de sa facilité d'installation, de compréhension, d'apprentissage, de traitement et d'autres avantages, que les programmeurs peuvent reconnaître lorsqu'ils l'utilisent, néanmoins en raison des pénuries des modules du réseau et d'implémentation de protocole, il existe encore de nombreuses

limitations lors de la simulation dans le domaine du réseau. De ce fait l'adaptation de ce simulateur au réseau est le sujet de notre thème de recherche. Pour commencer nous simulerons le protocole AODV afin de voir le fonctionnement de NetLogo, de le comprendre, d'offrir à la bibliothèque de NetLogo plus de modèles dans ce domaine et de voir les possibilités de changement à réaliser qui nous aidera à atteindre notre objectif principal.

Ce mémoire se divise en cinq chapitres :

- Le **premier chapitre** présente les notions Agent et SMA, leurs caractéristiques, leurs avantages, et le rapport entre la technologie agent et le réseau ad hoc ainsi que quelques protocoles de routage basés sur les agents.
- Le **deuxième chapitre** expose les réseaux ad hoc, leurs principaux caractéristiques, leurs avantages et inconvénients ainsi que leurs domaines d'applications, ensuite nous donnons une présentation sur les techniques de routage avec les difficultés rencontrées, et finalement nous abordons les différentes catégories des protocoles dans le réseau ad hoc (proactif, réactif et hybride), et nous détaillons quelques protocoles les plus connus tout en indiquant leurs avantages et leurs inconvénients.
- Le **troisième chapitre** donne un aperçu sur le concept de simulation, leurs types, leurs avantages et inconvénients ainsi que son processus d'expérimentation, et une vue globale sur la simulation à base d'agent, puis nous présentons une étude comparative entre les simulateurs réseaux les plus utilisés dont nous fixons notre choix sur le simulateur multi-agent « **NetLogo** ».
- Le quatrième **chapitre** est consacré pour la simulation du protocole de routage le plus connu dans la famille des protocoles ad hoc réactifs : Ad hoc On demand Distance Vector (**AODV**) par le simulateur choisi « **NetLogo** ».
- Le cinquième **chapitre** explique la notion d'extension en donnant un aperçu global sur les différentes étapes à suivre pour sa création. Nous présentons ainsi son intérêt, son utilisation, et quelques conseils pour développer une extension. Par la suite, nous avons exposé l'objectif et le principe de création de notre extension.

Enfin nous terminons ce mémoire par une conclusion générale qui résume nos objectifs estimés et des perspectives.

Chapitre I

Système Multi-Agents

I.1. Introduction

Durant ces dernières décennies, les Systèmes Multi-Agents se sont imposés comme étant le paradigme le plus approprié pour résoudre les problèmes récurrents que ce soit dans le domaine de l'intelligence artificielle, dans le domaine des systèmes distribués, dans le domaine de la robotique, ou même dans ce nouveau champ disciplinaire qu'est la vie artificielle. Le thème des SMA est de nos jours un champ de recherche très répondu. Le SMA est une discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes appelées Agents.

En outre le paradigme multi-agents est apparu comme une solution appropriée aux défis des réseaux ad hoc. Pour cela nous allons présenter dans de ce chapitre les concepts d'agents et SMA ainsi leurs rapport avec le réseau ad hoc.

I.2. Agent et Système Multi-Agents

Le système multi-agents (SMA) est composé d'un ensemble d'agents, situés dans un certain environnement et interagissent selon certaines relations pour résoudre ou réaliser une tâche, alors qu'un agent présente un comportement autonome qui est la conséquence de ses perceptions, de ses représentations et de ses communications.

I.2.1. Agent

I.2.1.1. Définition

Il n'y a encore aucun consensus, entre les différents chercheurs, quant à la définition même du mot « **Agent** ». Nous citons quelques-unes dans ce qui suit :

« On appelle agent un **système mécanique**, **biologique** ou **logiciel** qui interagit avec son environnement ». Anne Nicole.

La définition la plus adoptée et la plus complète d'un agent est celle de J.FERBER [1] qui est la suivante :

On appelle agent une entité réelle ou virtuelle plongée dans un environnement sur lequel elle est :

- Capable d'**agir**.
- Qui peut **communiquer** directement avec d'autres agents.
- Qui est mue par un ensemble de **tendances** (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser).
- Qui possède des **ressources** propres.
- Qui est capable de **percevoir** (mais de manière limitée) son environnement.

- Qui ne dispose que d'une représentation **partielle** de cet environnement (éventuellement aucune).
- Qui possède des **compétences** et offre des **services**.
- Qui peut éventuellement **se reproduire**.
- Dont le **comportement** tend à satisfaire ses **objectifs**, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

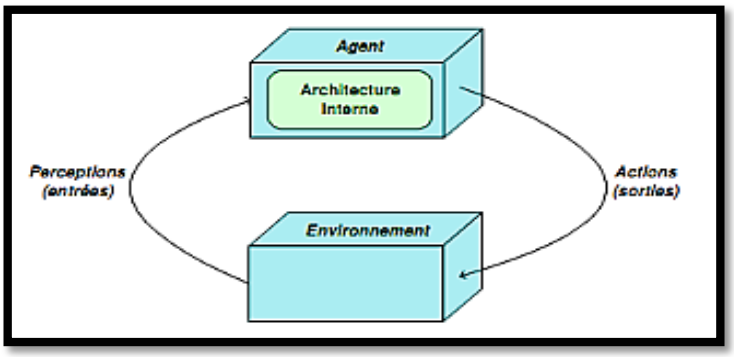


Figure I. 1: Représentation classique d'un agent et de son environnement [2].

La figure I.1 montre qu'un agent est défini par un ensemble de **perceptions** (entrées), un ensemble **d'actions** (sorties) et on parle de l'architecture interne de l'agent pour désigner les mécanismes qui définissent sa dynamique intrinsèque.

Après cette définition nous pouvons ressortir les principales caractéristiques d'un agent :

I.2.1.2. Propriétés

- **Autonome**

  Son comportement est en fonction de ses **perceptions** qui agissent sur son état, et de sa représentation de l'**environnement** dans lequel il évolue. Aucun super contrôleur ne peut le piloter de l'extérieur [3].

- **Proactif**

  Il peut prendre des initiatives afin de satisfaire ses **buts**. Pour le faire, il n'est pas soumis à l'invocation d'une autre entité pour agir mais peut agir sur sa propre **initiative** [3].

- **Flexible**

  Caractérise un agent intelligent. Être flexible signifie que l'agent est capable : de **percevoir** son environnement et **répondre** en temps **voulu** aux changements qui s'y produisent (**réactivité**); d'avoir un comportement dirigé par son but, d'être **opportuniste** et **capable** d'initiative quand c'est approprié (**proactivité**); et d'interagir quand c'est approprié, avec d'autres agents artificiels ou humains de manière à compléter leur résolution de problème et d'aider les autres dans leurs

activités (**sociabilité**) [4]**.**

- **Social**

Il a la capacité d'interagir pour atteindre ses buts ou pour aider d'autres agents dans leurs activités [3].

- **Situé**

Capacité à **percevoir** l'environnement à travers les métriques **spatio-temporelles** dans lequel il peut agir de façon limitée [3].

- **Coopération**

Les agents doivent s'efforcer d'atteindre et de satisfaire un objectif commun ou personnel.

- **Communication**

Un agent communique d'une manière directe ou indirecte avec les autres agents par **coopération** ou par **compétition**, le but est de réaliser une tâche donnée. Nous distinguons deux modèles de communication.

1. **Communication par partage d'information**

Communication via le tableau noir. Elle se fait par un tableau commun pour tous les agents, ces derniers communiquent en déposant ou en récupérant des informations sur ce tableau.

2. **Communication par envoi de messages**

- Agents en liaison directe.
- Envoi directement et explicitement au destinataire.
- Disposer d'un langage commun pour pouvoir coopérer pour la résolution d'un problème.
- Langage (protocole) primitives connues par chaque entité.

- **Mobilité**

Il y a deux types de mobilité :

Mobilité relative (ou par requêtes) : dans ce cas il n'y pas un réel **déplacement de l'agent**, celui-ci lance une succession de requêtes à destination de différents serveurs.

Mobilité réelle (de l'agent) : le processus agent se déplace d'un serveur à un autre sur le réseau. Le code de l'objet ainsi que ses données sont transportés sur une nouvelle machine où il continu son exécution [5]**.**

- **Agent Mobile**

Les agents mobiles sont des agents qui ont la possibilité de se déplacer entre les machines du réseau. Un agent mobile exécuté sur un ordinateur du réseau peut suspendre son exécution sur cet ordinateur, puis se déplacer vers un autre ordinateur sur le réseau, puis poursuivre l'exécution. À un moment donné, l'agent mobile est composé de son code, de son contexte d'exécution et de ses données. Le programme d'agent mobile hérite de deux technologies, à

savoir la technologie d'agent et la technologie de code mobile, qui comprennent la définition de code de sorte que les programmes compilés avec le code puissent être déplacés et exécutés sur n'importe quel ordinateur.

- **Avantages des agents mobiles**

- Réduire la charge du réseau.
- Hétérogène, robuste et tolèrent.
- Surmonter le temps de latence sur le réseau.
- S'exécuter de manière asynchrone et autonome.
- S'adapter dynamiquement aux environnements d'exécution [6].

I.2.1.3. Architectures

Il existe trois grandes familles d'agents : les agents réactifs, les agents cognitifs et les agents hybrides. La différence entre celle-ci se fond sur le mécanisme décisionnel et sur l'environnement de chaque agent.

a- Agent réactif

Comme son nom l'indique, un agent réactif ne fait que réagir aux changements qui surviennent dans l'environnement. Autrement dit, un tel agent ne fait **ni délibération ni planification**, il se contente simplement d'acquérir des perceptions et de réagir à celles-ci (Figure I.2). Étant donné qu'il n'y a pratiquement **pas de raisonnement**, ces agents peuvent **agir** et **réagir** très rapidement [7].

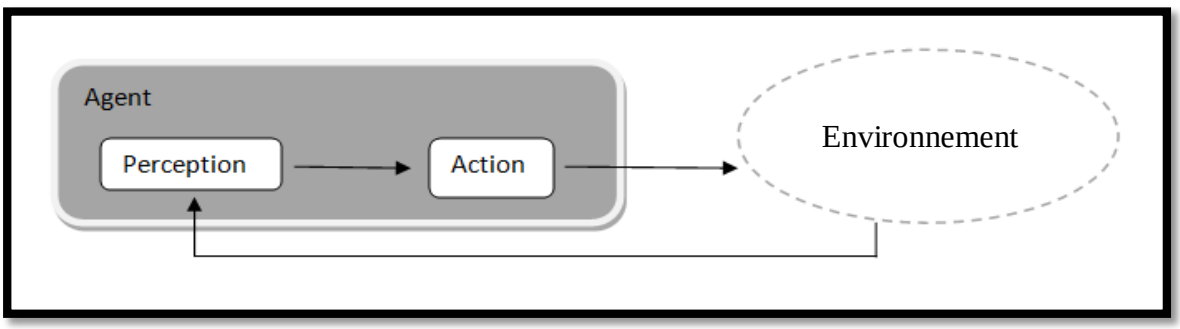


Figure I. 2: Agent réactif.

L'agent réactif (figure I.2) présente les caractéristiques suivantes :

- Absence de la mémoire.
- Pas de représentation explicite.
- Prise de décision se focalise sur le fait du Stimulus/Réponse.
- Simplicité de mise en œuvre.

b- Agent cognitif

Ces agents possèdent une représentation explicite de leur environnement, des autres agents et d'eux-mêmes. Ils sont aussi dotés de capacités de **raisonnement** et de **planification** ainsi que de **communication** [7] . Ce type d'agent porte sur les principes de **BDI :**

- Croyance (Belief **B**): C'est l'ensemble de connaissance varies d'un agent sur les agents de son environnement et sur l'environnement lui-même.
- Désir (Desire **D**) : Ce sont les états de l'environnement et parfois de lui-même, qu'un agent aimerait voir réaliser. Ce sont les objectifs que se fixe un agent [7].
- Intention (Intention **I**) : Ce sont les actions qu'un agent a décidé de faire pour accomplir ses désirs. Ils forment des ensembles de plans qui sont exécutés tant que l'objectif correspondant n'est pas atteint [7].

Le modèle BDI a inspiré beaucoup d'architectures d'agents cognitifs.

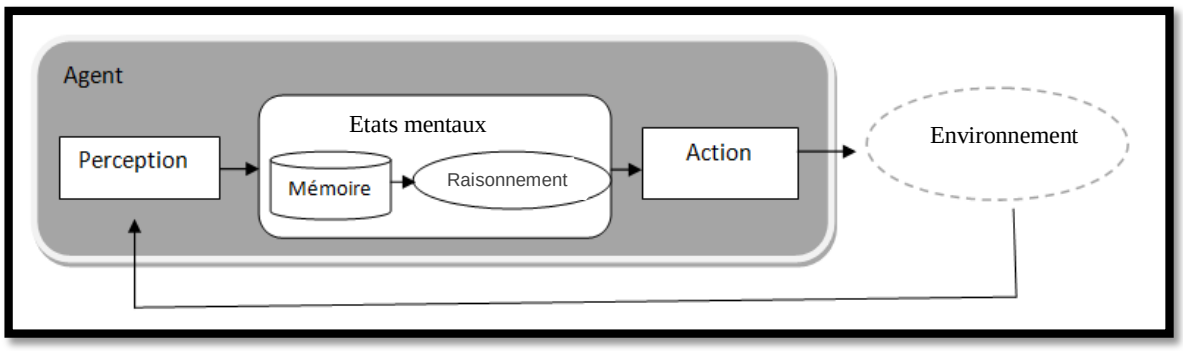


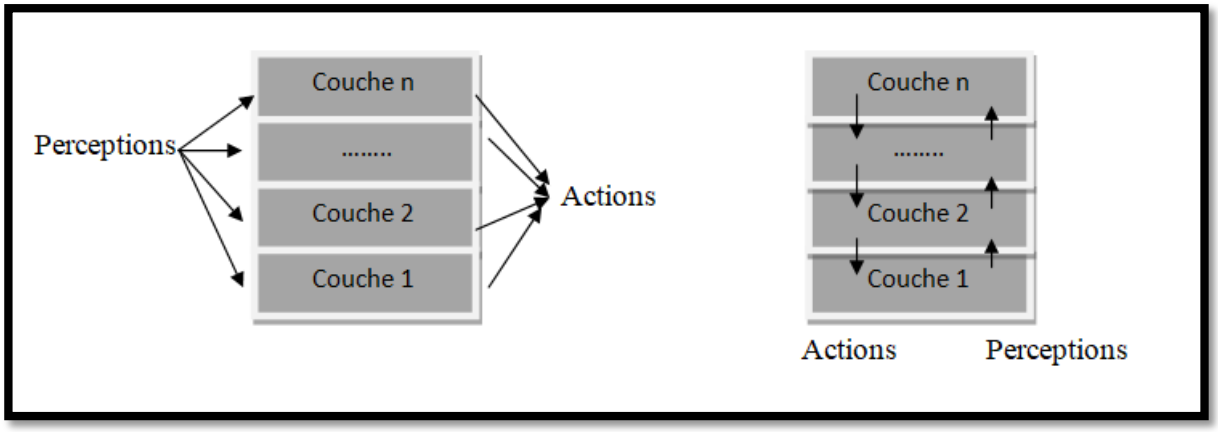
Figure I. 3: Agent cognitif.

Ce type d'agent (figure I.3) se caractérise par :

- Une représentation explicite de l'environnement et du monde auquel ils appartiennent.
- Planification de la réaction.
- Des informations et du savoir-faire qui se trouvent dans une base de connaissance.
- Présence d'une mémoire pour sauvegarder les états précédents.

c- Agent hybride

En général, la différence entre des agents réactifs et des agents cognitifs peut être expliquée par le compromis **efficacité/complexité**. La complexité des systèmes réactifs exige le développement de nouvelles théories dans le domaine de la **coopération**, de la **communication** et de la **compréhension** de nouveaux phénomènes telles que **l'émergence**. Toutefois, il est maintenant possible de concevoir des systèmes hétérogènes comportant les deux types de comportements (cognitif et réactif) : on parlera alors d'agents hybrides [8]. Généralement, nous trouvons deux présentations pour cette architecture « horizontale et verticale ».



(a) Architecture horizontale (b) Architecture verticale

Figure I. 4: Agent hybride.

Ce type d'agent (Figure I.4) se caractérise par :

- Combinaison des capacités d'agent réactif et d'agent cognitif.
- Adaptation du comportement en temps réel à l'évolution de l'environnement.
- Un agent est composé d'une architecture multicouche qui se base sur la hiérarchie de niveaux.
- Puisque différents composants peuvent fonctionner en même temps, la puissance de traitement peut être améliorée.

I.2.1.4. Cycle de vie

Chaque agent a le même cycle de vie, composé des étapes suivantes :

Tant que l'agent est en vie **faire**

Début

Attendre des informations ;

Traiter ces informations ;

Construire d'éventuels messages pour d'autre agents ;

Fin tant que

Dans son cycle de vie, l'agent passe par trois états, comme la montre la figure ci-dessous :

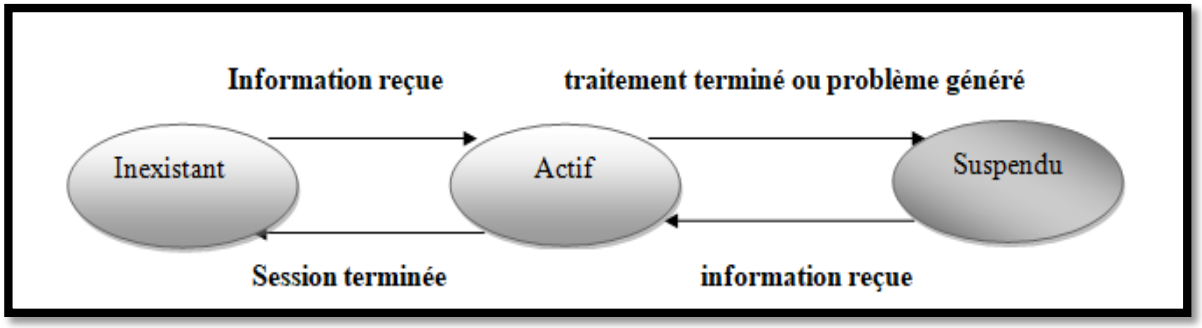


Figure I. 5: Etats d'un agent.

Au début de la session, tous les agents sont dans un état **passif** appelé **inexistant**. C'est la réception d'un message qui va les activer, c'est-à-dire passé d'un état **inexistant** à un état **actif**. Les agents actifs attribueront le traitement en fonction de leurs compétences et des informations reçues. A l'issue de son traitement, l'agent attendra des informations et son état sera **suspendu**.

Un agent est suspendu dans deux cas :

- Soit il est **suspendu** parce qu'il a terminé, il communique éventuellement des informations à d'autres agents avant de devenir inactif. C'est la réception des messages qui aura pour effet de le réveiller et de le mettre dans un état **actif**.
- Soit il est **suspendu** parce qu'il est bloqué dans sa session. Il a engendré des tâches qu'il a communiqué à d'autres agents puis s'est mis en attente de réponses**.**

I.2.2. Système Multi-Agents

La section précédente a présenté un système avec un seul agent, mais dans la plupart des cas pratiques, l'agent n'est pas seul dans son environnement, il y a d'autres agents autour. Par conséquent, nous devons adopter un système dans lequel les agents doivent interagir les uns avec les autres pour effectuer leurs tâches. Un tel système est appelé " **SMA**".

I.2.2.1. Définition

Ce système composé d'un ensemble d'agents au moins partiellement **autonomes**, situés dans un certain environnement et interagissant selon certaines relations (de **coopération**, de **concurrence** voir de conflit) [9] **.** SMA a une fonction à réaliser ou une tâche à résoudre. Ses principales caractéristiques sont que le système est **ouvert** ou **fermé**, **homogène** ou **hétérogène** et **autonome** ou non. Ces agents constituent un système complexe qui comprend une intelligence que l'on pourrait qualifié de collective. Ceci est un exemple d'une colonie de fourmis composée d'entités (fourmis) qui n'ont pas leurs propres capacités cognitives, mais qui atteignent un haut degré d'**organisation** et d'**adaptabilité** comme s'il s'agissait d'une logique raisonnable collective.

En plus, SMA désigne plusieurs agents appartenant à un même groupe qu'est définit par un ensemble de caractéristiques communes à tous les éléments du groupe [10]**.**

- Un **environnement** pour les contenir :
  – L'environnement c'est l'espace commun aux agents d'un SMA, doté d'un ensemble d'objets et de possibilité de perception et d'action.
- Des **interactions** pour :
  – Echanger des connaissances.
  – Transmettre des croyances.

– Coordonner entre les actions.

- Des **organisations** pour les structurer.

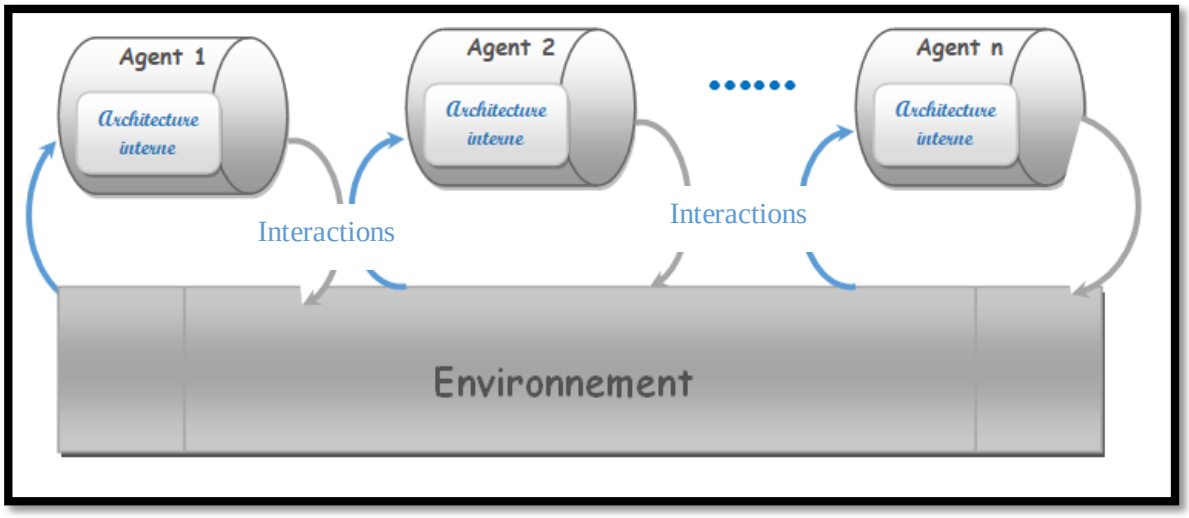


Figure I. 6: Représentation schématique d'un système multi-agents.

La figure I.6 présente un SMA avec ses deux principaux composants agents et environnement, en précisant que chaque agent possède une architecture interne et interagit avec d'autres agents dans un environnement commun.

Un **SMA** peut être [11] **:**

- **Ouvert :** Les agents y entrent et sortent librement (ex : un café).
- **Fermé :** L'ensemble des agents reste le même (ex : un match de football).
- **Homogène :** Tous les agents sont construits sur le même modèle (Exemple : une colonie de fourmis).
- **Hétérogène :** Les agents sont de modèles différents, de granularités différentes (ex : l'organisation universitaire).

I.2.2.3. Interactions

Les SMA mettent en œuvre des interactions complexes, comme la **coopération, la coordination, la collaboration, la compétition, et la négociation** [4] [11]**:**

- La **coopération :** Lorsque les agents travaillent ensemble pour atteindre un but commun.
- La **coordination :** Consiste à organiser les activités liées à la résolution d'un problème de manière à éviter les interactions néfastes et à exploiter les interactions bénéfiques.
- La **collaboration :** Les agents du système partagent un même but de façon intermittente.
- La **compétition :** Les agents du système ont des buts incompatibles
- La **négociation :** A pour objectif la gestion des conflits entre agents, c'est à dire qu'elle consiste à aboutir à un accord acceptable par l'ensemble des agents impliqués.

I.2.2.2. Propriétés

- Chaque agent à des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent à un point de vue partiel [12].
- Il n'y a aucun contrôle global du SMA [12].
- Les données sont décentralisées [12] .
- Le calcul est asynchrone [12].
- Un système composé des éléments suivants :

  Un environnement E, un ensemble d'objets O, un ensemble d'agents A, un ensemble de relations R, un ensemble d'opérations Op.

I.2.2.4. Caractéristique

ELFAZZIKI Et all ont cité les principales caractéristiques d'un SMA, voir [13]:

- **L'hétérogénéité** des agents : les messages doivent être mutuellement compréhensibles.
- **L'échange de savoir** : un agent coopératif au sein du SMA doit pouvoir exprimer ses différents types de connaissances.
- **Le contrôle local** : les agents doivent être autonomes, c'est-à-dire, leur comportements ne doit dépendre ni d'un planificateur central ni d'interactions prédéfinies.

I.2.2.5. A + E + I + O : l'approche Voyelles

A. Agent

[12]**:** Un agent est une entité qui **perçoit son environnement** et **agit** sur celui-ci.

B. Environnement

- Espace commun aux Agents d'un SMA, doté d'un ensemble d'objets et de possibilités de perception et d'action.
- Tout ce qui n'est pas Agent dans un SMA.
- Médium de l'interaction.
- Lieu dans lequel les actions sont réalisées et dans lequel les perceptions sont perçues.
- Une source de données pour les Agents.
- L'environnement d'un Agent en particulier est constitué de tout ce qui l'entoure [13]**.**

C. Interaction

Les interactions entre agents peuvent se résumés à ce qui suit [13]**:**

- Toute tâche qui affecte l'agent dans la réalisation de son but, de son objectif.
- Mise en relation dynamique d'agents par le biais d'un ensemble d'actions réciproques.

- Il y a existence d'une interaction dès lors que la dynamique propre d'un agent est perturbée par les influences des autres agents.
- La communication est un type d'interaction.

D. Organisation

L'organisation sociale d'un SMA est la manière dont le groupe est constitué, à un instant donné, pour pouvoir fonctionner. Elle décrit l'ensemble des composants fonctionnels du système, leurs natures, leurs responsabilités et leurs besoins en ressources ainsi que les liens de communication entre les agents. Cette organisation peut être statique ou dynamique.

Une société d'agent est constituée de trois éléments : un ensemble d'agents, un ensemble de taches à réaliser et un ensemble d'objets associés à l'environnement.

Un agent peut prendre la responsabilité d'effectuer une tache s'il en a la capacité. Il prend alors un rôle dans le groupe. La réalisation d'une tache suppose la manipulation d'objets de l'environnement. L'organisation peut être dynamique : le groupe se réorganise à chaque fois en fonction de la tâche à accomplir [14]**.**

I.2.2.5. Coopération

On dira que plusieurs agents coopèrent, ou encore qu'ils sont dans une situation de coopération, si l'une des deux conditions est vérifiée [1]**:**

1. L'ajout d'un nouvel agent permet d'accroitre différentielle-ment les performances du groupe.
2. L'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels.

On peut citer sept méthodes de coopérations :

- **Le regroupement :** Signifie que les agents forment des groupes homogènes dans leur environnement, ces groupes comprennent tous les spécialistes dans divers domaines et facilitent la communication entre les agents.
- **La multiplication** : Il s'agit d'une augmentation du nombre d'agents dans le système.
- **La communication :** Le système de communication qui lie un ensemble d'agents agit comme une sorte de système nerveux qui met en contact des individus parfois séparés. La communication en effet agrandit les capacités perceptives des agents en leur permettant de bénéficier des informations et du savoir-faire des autres agents. Les communications sont indispensables à la coopération et il est difficile de concevoir un système d'agents coopérants s'il n'existe pas un système permettant aux agents d'échanger des informations ou de véhiculer des requêtes. Dans les systèmes cognitifs, les communications s'effectuent par envoi de messages, alors que dans les systèmes réactifs, elles résultent de la diffusion d'un signal

dans l'environnement. La communication constitue l'un des moyens fondamentaux pour assurer la répartition des tâches et la coordination des actions [14].

La communication entre les individus peut se faire directement ou indirectement. Quand deux individus interagissent indirectement en modifiant l'environnement, on parle du principe de la **stigmergie :**

La **stigmergie** est un mécanisme de coordination indirecte entre les agents. Le principe est que la trace laissée dans l'environnement par l'action initiale stimule une action suivante, par le même agent ou un agent différent. De cette façon, les actions successives ont tendance à se renforcer et ainsi conduisant à l'émergence spontanée d'activité cohérente, apparemment systématique [15].

- **La spécialisation :** La spécialisation est un processus dans lequel les agents s'adaptent de plus en plus à leurs tâches. Il est souvent difficile de développer des agents spécialisés pour chaque tâche.
- **La collaboration par partage de tâches et de ressources:** Nous appellerons collaboration l'ensemble des techniques permettant à des agents de répartir des tâches, des informations et des ressources de manière à réaliser une œuvre commune. Résoudre un problème de collaboration consiste donc à répondre à la question "qui fait quoi ?" par rapport à un travail donné [16].La collaboration c'est la capacité d'un agent à travailler en groupe, elle est la condition nécessaire à la coopération ; si l'agent ne veut pas collaborer il n'aura pas de coopération. La collaboration peut être imposée (dans ce cas on parle de **soumission** de comportement qu'on retrouve chez les animaux.), comme elle peut être du propre chef de l'agent (dans ce cas on parle de **volontarisme**. Comportement qu'on retrouve chez les humains, par exemple : construction d'une école par l'ensemble des villageois).
- **La coordination d'actions :** Comme les agents se partagent l'environnement ou ils évoluent, la coordination est nécessaire car l'environnement dispose des ressources limitées et pour éviter de réessayer ou de négliger certaines tâches. La coordination est basée sur la communication et est nécessaire à la coopération. Les agents qui ne peuvent pas coordonné leurs activités ne peuvent jamais parvenir à une coopération.
- **La résolution de conflit par arbitrage et négociation :** L'arbitrage et la négociation sont deux des moyens utilisés par les SMA pour résoudre les conflits [14].

I.2.2.6. Conception

Pour concevoir un SMA, il faut définir :

- **Un modèle de SMA :**

  Le modèle de chacun des agents qui vont entrer en action (niveau microscopique), définir leur environnement et leurs interactions (niveau macroscopique) et définir les organisations sociales (niveau macro) qui les structurent [12].

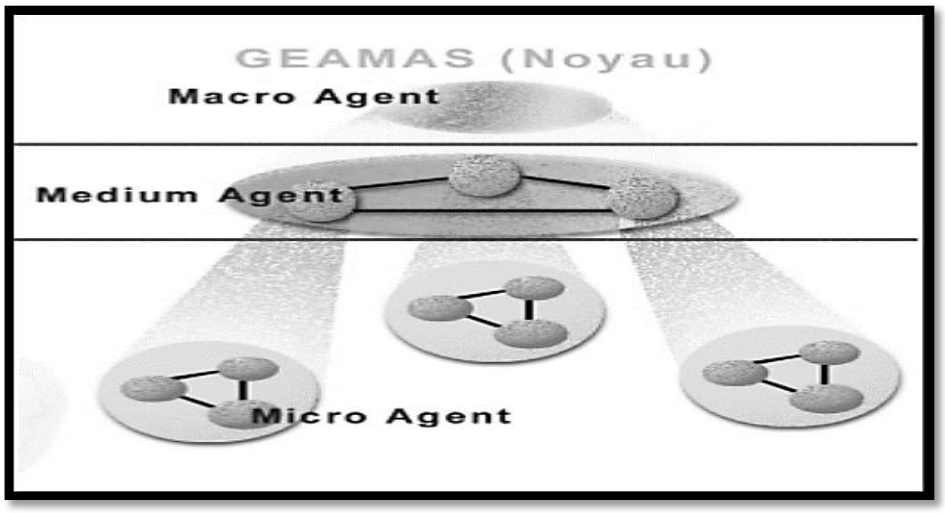


Figure I. 7: Présentation de niveau macro et micro d'un SMA [3].

La figure I.7 illustre les 3 niveaux d'un SMA [3]:

- ♦ Niveau Macro : Représente l'ensemble du SMA.
- ♦ Niveau Medium : Emergence comportementale, Modèle Hiérarchique et Structure de Groupe.
- ♦ Niveau Micro : Entité autonome proactive de granularité la plus fine.

- **Un modèle concret de SMA :**

  Qui crée, initialise les agents, installe leur organisation et lance les agents qui doivent intervenir pour une exécution particulière [12]**.**

I.2.2.7. Notions importantes dans SMA

A. Adaptation

Un système multi-agents adaptatif est donc défini comme un SMA qui est capable de changer son comportement en cours de fonctionnement pour l'ajuster dans un environnement dynamique, soit pour réaliser la tâche pour laquelle il a été conçu, soit pour améliorer sa fonction ou ses performances. Il est caractérisé par le fait d'être plongé dans un environnement dynamique, de réaliser une tâche (fonction) et d'être composé d'agents en interaction. La conception de ces systèmes nous a amenée à intéresser aux notions d'**émergence** et d'**auto-organisation** [17]**.**

B. Emergence

La notion d'émergence peut être définie de manière intuitive comme une propriété macroscopique d'un système qui ne peut pas être inféré à partir de son fonctionnement microscopique. Cette notion est étudiée depuis fort longtemps dans les disciplines telles que la philosophie, la physique, la thermodynamique… [17]**.**

C. Auto organisation

Il existe plusieurs définitions d'auto-organisation nous allons citer quelques une [18] **:**

- Motif ou fonction du niveau global d'un système (SMA) qui émerge suite aux interactions entre composants de plus bas niveau (agents).
- Transformation autonome de la topologie d'un système (exp : connexions réseau) par ses composants résultant du fonctionnement de ce réseau.
- Processus par lequel un système change son organisation interne pour s'adapter aux changements de ses buts et de l'environnement sans contrôle externe explicite. L'auto-organisation résulte souvent en un comportement émergent désirable ou non.
- Le système doit pouvoir s'adapter à son environnement.
- Auto-organisation des composants ➔ adaptation**.**

I.2.2.8. Domaines d'application

Les différentes applications des SMA relèvent de plusieurs domaines, voir [5] [19] **:**

- **La résolution de problèmes par émergence :** Faire émerger une solution à un problème complexe à partir de comportements simples tels que la gestion des réseaux de télécommunication, système de transport.
- **Le contrôle de systèmes complexes :** Les SMA sont aussi utilisés pour le contrôle ou le pilotage d'outils ou de composants immergés dans un environnement dynamique, telles les chaînes de production ou encore les robots tels que les robots automates mobiles.
- **La simulation des systèmes complexes :** C'est là que les caractéristiques d'autonomie, de proactivité, des agents interviennent. elle est utilisée pour pouvoir jouer sur des paramètres difficilement modifiables, voir non modifiables, dans les cas réels pour observer leurs influences, tel que la simulation individu-centrée.
- **Le travail collaboratif assisté par ordinateur :** Comme les agents assistants, agents médiateurs.
- **La télématique** (internet) : Comme les agents " intelligents", agents d'interface.
- **Les systèmes multi-capteurs.**
- **Les jeux vidéo** (intelligence des caractères).

I.2.2.9. Avantages

Ils possèdent les avantages traditionnels de la résolution distribuée et concurrente de problèmes [20]:

- **La modularité :** Permet de rendre la programmation plus simple. Elle permet, de plus, aux SMA d'être facilement extensibles, parce qu'il est plus facile d'ajouter de nouveaux agents à un SMA que d'ajouter de nouvelles capacités à un système monolithique.
- **La vitesse :** Est principalement due au parallélisme, car plusieurs agents peuvent travailler en même temps pour la résolution d'un même problème.
- **La fiabilité :** Peut être également atteinte, dans la mesure où le contrôle et les responsabilités étant partagés entre les différents agents, le système peut tolérer la défaillance d'un ou de plusieurs agents. Si une seule entité contrôle le tout, alors une seule défaillance de cette entité fera en sorte que tout le système tombera en panne.

I.3. Technologie agent et les réseaux Ad Hoc

La technologie agent aura un rôle de plus en plus important à jouer dans les télécommunications grâce à ses propriétés, notamment, d'**autonomie**, d'**intelligence** et/ou de **mobilité**. Les réseaux **Ad Hoc** joueront également un rôle de plus en plus important offrant un accès omniprésent au réseau, favorisant ainsi la mobilité de l'utilisateur [21].

I.3.1. Intérêt

Les agents sont principalement utilisés dans les réseaux **Ad Hoc** pour améliorer les méthodes de localisation et les protocoles de mobilité existants, pour contrôler la signalisation sur le réseau, réduire les accès et adapter le handover[1] aux besoins de l'utilisateur.

La technologie agent permet à l'utilisateur de changer de point d'accès ou de réseau d'accès en fonction de ses besoins. D'autres domaines d'application des agents dans les télécommunications sont, également, importants tels que la proposition ou la composition dynamique de services personnalisés aux usagers. Dans ce cas, les technologies sans fil peuvent, encore, jouer un rôle important en tant que réseau d'accès [21].

1 Le handover ou transfert intercellulaire : est un mécanisme fondamental dans les communications mobiles cellulaires (GSM, CDMA, UMTS ou LTE par exemple)

I.3.2. Agents mobiles et le routage dans les réseaux Ad hoc

Les protocoles de routage **Table-Driven**[2] exigent de connaître la topologie du réseau entier. Pour que les nœuds restent au courant des changements dans la topologie des informations de mise à jour doivent être fréquemment propagés dans tout le réseau. Les protocoles **Table-Driven** utilisent donc en permanence une grande partie de la capacité du réseau pour garder les informations de routage à jour ce qui les rend inadaptés pour les réseaux ad hoc. Dans l'autre côté les protocoles de routage **On-demand**[3] utilisent une procédure de découverte de route qui génère un grand volume de trafic et produit la congestion du réseau et provoque la perte des paquets, ce qui les rend aussi ces protocoles inadaptés pour les réseaux ad hoc.

Les agents mobiles sont une solution pour découvrir la topologie du réseau et mettre en œuvre un protocole de routage sans générer trop de trafic dans le réseau. Les agents mobiles se déplacent dans le réseau et recueillent les informations nécessaires pour le routage, arrivent à un nœud ils vont mettre à jour sa table de routage par les dernières informations recueillies. Un nœud dans le réseau reçoit des agents des informations sur la topologie du réseau ce qui lui permet de calculer des routes correctes pour communiquer avec les autres nœuds [4].

I.3.3. SMA et le routage dans les réseaux Ad Hoc

Le système de gestion de routage dans **les réseaux ad hoc** nécessite une distribution des opérations, car il n'existe aucun élément central fixe qui peut organiser le réseau, il nécessite la **coopération** entre les différents nœuds mobiles car un nœud mobile ne peut communiquer qu'avec les nœuds proches de lui, il a donc besoin de coopérer avec les autres nœuds mobiles qui vont l'aider à transmettre les informations.

Le système de gestion de routage dans les réseaux ad hoc doit aussi être **robuste**, il doit pouvoir continuer à fonctionner normalement en cas de défaillance d'un ou plusieurs nœuds, il doit être **intelligent**, il doit choisir les meilleures routes pour transmettre les paquets et réagir rapidement aux évènements qui peuvent intervenir. Les SMA prennent en compte les aspects de **coopération, d'autonomie**, de **distribution** et d'**intelligence** ils nous semblent donc appropriés pour gérer le routage dans les réseaux ad hoc [22]**.**

2Le protocole Table-driven : son principe est de maintenir à jour des tables de routage qui indiquent les routes vers chaque destination du réseau.

3Le protocole On-demand : crée et maintient les routes selon les besoins.

I.3.4. Protocoles de routage basé agents

L'idée d'utiliser les agents pour le routage dans les **réseaux Ad hoc** a été explorée récemment dans quelques travaux. Nous allons présenter dans la suite de ce chapitre quelques protocoles de routage ad hoc basé sur les agents [4]**.**

- **Le protocole MARP**

MARP ***(*Multi-Agent Routing Protocol)** [23] est un protocole de routage pour les réseaux ad hoc basé sur les agents. MARP découvre les itinéraires à la demande et se base sur les nœuds de négociation en tant qu'agents intelligents. Les agents agissent de manière autonome dans le MARP. Le routage dans les MARP est basé sur la négociation des agents pour la livraison des paquets de données. Chaque nœud a des caractéristiques incertaines et une connaissance limitée des agents, des autres nœuds. Les connaissances de l'agent sont mise à jour par la négociation avec d'autres agents, et par la transmission de paquets de données.

Dans MARP, chaque agent stocke ses connaissances dans quatre types de tableaux qui consistent en : Table d'États, Table de routage, Table de voisins, et Table des croyances.

- Le "Table des États" comprend les dernières informations de l'agent.
- Le "Table de routage" contient les dernières informations sur les agents de destination et les agents voisins qui sont utilisés pour livrer des paquets de données à la destination souhaitée.
- La "Table de voisin" comprend une liste des voisins, qui est mis à jour périodiquement.

Le routage MARP comprend les trois phases suivantes :

1) **Découverte des routes :** Les nouveaux itinéraires sont trouvés dans cette phase. Les agents sont responsables de la livraison des paquets de données d'un nœud source à un nœud destination, tout en essayant de trouver une route optimale.
2) **Maintenance des routes :** La deuxième phase de routage est l'entretien des itinéraires, qui est chargé de maintenir les routes pendant la tâche de transmission.
3) **Traitement des défaillances :** La troisième phase de l'acheminement concerne la gestion des défaillances potentielles, qui sont souvent dues à la mobilité des nœuds, et sont parfois dus au fait que la batterie des nœuds qui contribuent à une tâche de transmission est faible.

Le Protocole MWAC

MWAC **(Multi-Wireless-Agent Communication),** dans ce protocole on associe à chaque nœud un agent. Les agents vont s'organiser d'une manière dynamique en groupes ayant une structure hiérarchique afin de localiser au mieux l'inondation et ainsi obtenir un gain en ressources. L'organisation des agents va émergée de leurs interactions et de l'évolution de leurs états.

Un groupe dans ce protocole contient :

– Un agent représentant qui va gérer les communications au sein de son groupe,

– Un ou plusieurs agents de liaison : ils appartiennent à plusieurs groupes et permettent aux différents représentants de communiquer entre eux,

– Aucun où plusieurs simples membres qui n'ont aucun rôle particulier dans le groupe sauf la réception et le traitement des paquets qui leurs sont destinés ou la transmission leurs propres paquets [4]**.**

Conclusion

Dans ce chapitre nous avons présenté en premier lieu les notions agent et SMA, nous avons vu qu'un SMA est un ensemble d'entités autonomes nommées **« agents »** qui évoluent dans un environnement commun. L'ensemble d'agents interagissent dans un environnement en se basant sur des propriétés comme la **coopération**, l'**autonomie** et l'**intelligence.**

Certains domaines requièrent l'utilisation de plusieurs entités séparées ou distantes, par conséquent, les SMA sont très bien adaptés à ce type de situations. A cet égard nous avons aussi cité quelques domaines d'application tels que les systèmes distribués, interfaces homme/machine et les réseaux.

En second lieu, nous avons exposé le rapport entre la technologie agent et le réseau **Ad hoc** ayant un très grand impact sur l'évolution de ces réseaux. Finalement, nous avons présenté la relation entre les réseaux **Ad hoc** et les **SMA**, en citant quelques protocoles de routage inspiré de communautés d'agent.

Dans notre travail, les SMA vont nous être utile surtout pour la conception et la mise en œuvre d'un simulateur réseau basé agent. Dans le prochain chapitre nous allons parler sur les généralités sur les réseaux Ad Hoc ainsi que le concept du routage dans ce domaine.

Chapitre II

Réseaux Ad Hoc et
Protocoles de routage

II.1. Introduction

Le développement de la technologie sans fil a ouvert de nouvelles perspectives dans le domaine des télécommunications. Les réseaux basés sur la technologie sans fil connaissent aujourd'hui une forte expansion. Ils offrent une grande flexibilité d'emploi, et ils permettent aussi aux utilisateurs de se déplacer librement tout en continuant normalement leurs communications. Les réseaux sans fil peuvent être classés en deux catégories : les réseaux avec infrastructure fixe préexistante, et les réseaux sans infrastructure (**Ad Hoc**) qui est un ensemble de nœuds où l'acheminement de l'information d'une source vers une destination nécessite des protocoles de routage qui établissent des routes entre les nœuds du réseau.

Dans ce chapitre nous visions à donner une vue globale sur les réseaux Ad Hoc et un aperçu général sur le routage dans Ad Hoc, nous commençons par définir le réseau sans fil et ses catégories, le réseau Ad Hoc ainsi que ces principaux caractéristiques, enfin nous citons les domaines d'application de ce réseau. Nous allons aussi définir la notion du routage et le principe du routage dans les réseaux Ad Hoc. Par la suite nous allons voir les problèmes de conception des protocoles de routage, citer la classification des protocoles de routage de ces réseaux et à la fin nous allons présenter quelques protocoles de routage les plus connus dans ce domaine.

II.2. Définition du réseau sans fil

Un réseau sans fil est un réseau informatique ou numérique qui connecte différents postes ou systèmes entre eux par ondes radios. Le réseau sans fil peut être associé à un réseau de télécommunication pour réaliser des interconnexions entre nœuds [24]**.**Ces réseaux de communications permettent aux utilisateurs de profiter de tous les services traditionnels des réseaux indépendamment de leurs positions géographiques. Le rayonnement géographique des ondes est relativement limité étant donné la faible puissance d'émission des solutions matérielles actuelles. Pour cette raison, les réseaux sans fil se sont avant tout développés comme réseaux internes, propres à un bâtiment, soit comme réseau d'entreprise, soit comme réseau domestique [25]**.**

II.3. Catégories des réseaux sans fil

Les réseaux sans fil peuvent être classifiés selon deux critères. Le premier est la zone de couverture du réseau. Au vu de ce critère il existe quatre catégories : les réseaux personnels, les réseaux locaux, les réseaux métropolitains et les réseaux étendus. Le second critère est l'infrastructure ainsi que le modèle adopté. Par rapport à ce critère on peut diviser les réseaux sans fil en : réseaux avec infrastructures et réseaux sans infrastructure (Ad Hoc) [26]**.**

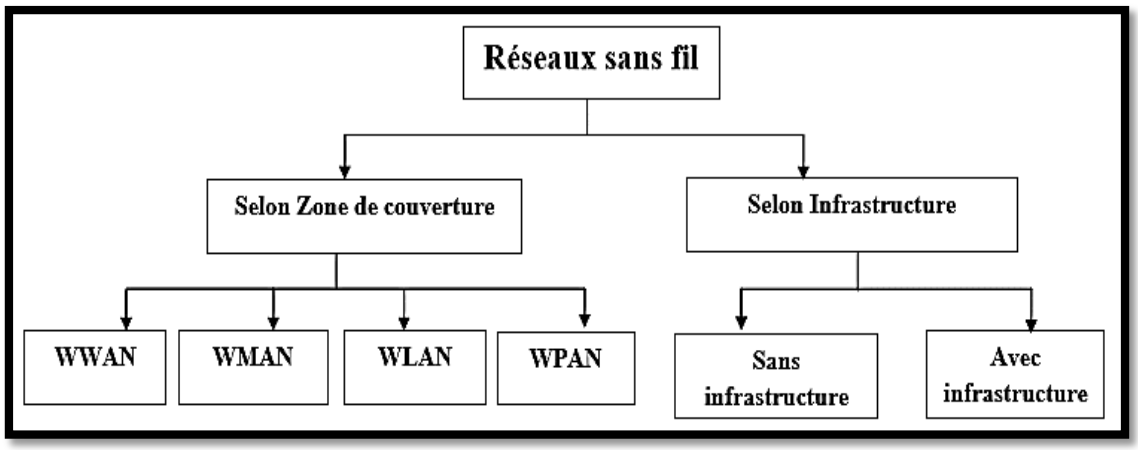


Figure II. 1: Classification des réseaux sans fil [27].

Cette figure représente la classification des réseaux sans fil selon la zone de couverture et la présence ou non d'infrastructure.

II.3.1 Selon la zone de couverture

a. Réseaux personnels sans fil (WPAN)

Les réseaux personnels sans fil ou Wireless Personal Area Network **(WPAN)**, sont des réseaux sans fil à très faible portée, de l'ordre de quelques dizaines de mètres. Ce type de réseau sert généralement à relier des appareils électroniques personnels .Par exemple : Bluetooth, ZigBee (IEEE 802.15.4), Liaisons infrarouge [25]**.**

b. Réseaux locaux sans fil (WLAN)

Depuis le développement des normes qui offrent un haut débit, les réseaux locaux sans fil ou Wireless Local Area Network **(WLAN)** sont généralement utilisés à l'intérieur d'une entreprise, d'une université, mais également chez les particuliers. Par exemple : IEEE 802.11, WiFi (Wireless Fidelity), Hiperlan 1 & 2 [26]**.**

c. Réseaux métropolitains sans fil (WMAN)

Il est connu aussi sous le nom de Boucle Locale Radio **(BLR).** Il convient de rappeler que la **BLR** permet en plaçant une antenne parabolique sur le toit d'un bâtiment, de transmettre par voie hertzienne de la voix et des données à haut débit pour l'accès à l'internet et la téléphonie. Il existe plusieurs types de réseaux **WMAN** dont le plus connu est : les réseaux **WiMAx** (Worldwide interoperability for Microwave Access).

d. Réseaux sans fil étendus (WWAN)

Il est connu sous le nom de réseau cellulaire mobile et les principales technologies sont les suivantes : **GSM** (Global System for Mobile Communication), **GPRS** (General Packet Radio Service), **UMTS** (Universal Mobile Télécommunication System), (voir figure II.2).

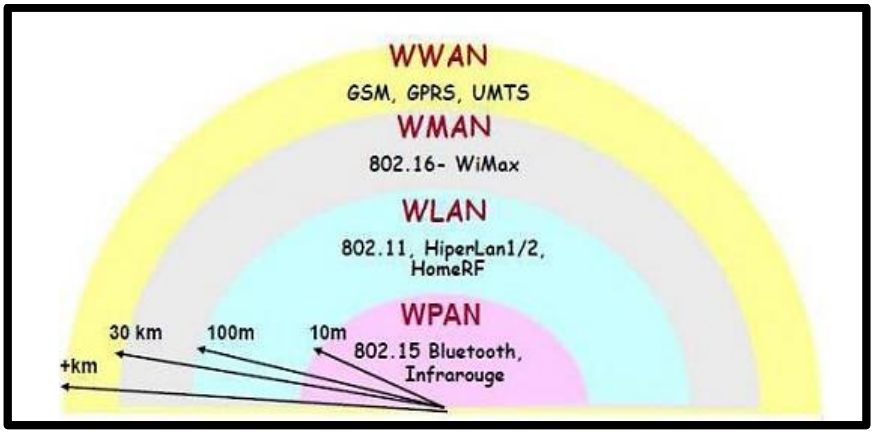


Figure II. 2: Classement des réseaux sans fil selon la portée [28].

II.3.2. Selon l'infrastructure

Les environnements sans fil sont des systèmes qui permettent à leurs utilisateurs d'accéder à l'information indépendamment de leurs positions géographiques. Les réseaux sans fil, peuvent être classés en deux classes : les réseaux avec infrastructure et les réseaux sans infrastructure. La figure II.3 montre la différence d'utilisation des réseaux sans fil en mode infrastructure fixe et en mode ad hoc.

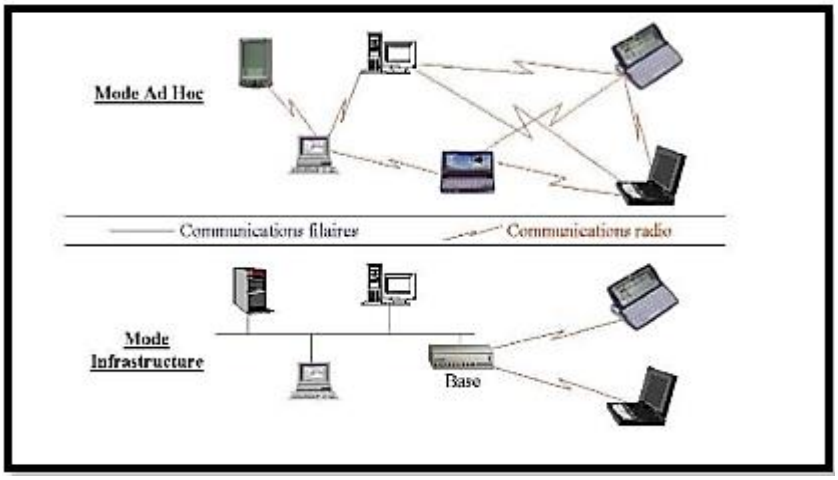


Figure II. 3: Mode infrastructure et mode Ad Hoc [29].

a. Réseaux sans fil avec infrastructure

Le modèle de système intégrant des sites mobiles et qui a tendance à se généraliser, est composé de deux ensembles d'entités distinctes : les "**sites fixes**" d'un réseau de communication filaire classique (**wired network**), et les "**sites mobiles**" (**Wireless network**).

Certains sites fixes, appelés stations support mobile (Mobile Support Station) ou station de base (**SB**) sont munis d'une interface de communication sans fil pour la communication directe avec les sites ou unités mobiles (**UM**), localisés dans une zone géographique limitée, appelée cellule [27], (voir figure II.4).

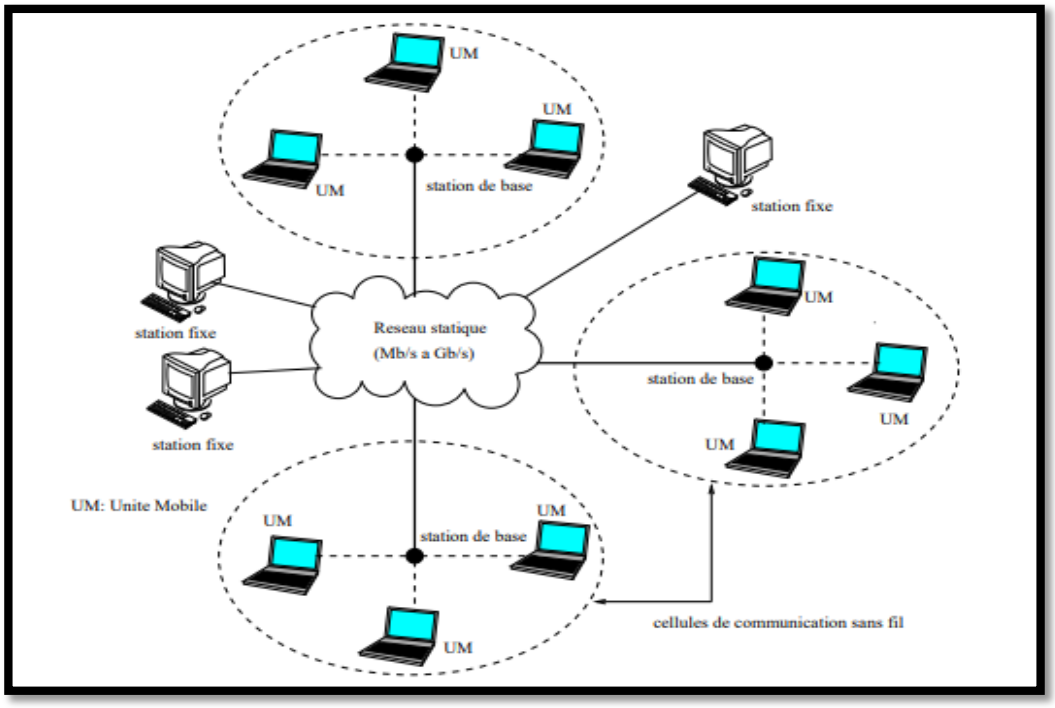


Figure II. 4: Réseau sans fil avec infrastructure [30].

La figure ci-dessus illustre le modèle avec infrastructure qui contient une station de base et des unités mobiles et qui se trouve au prés de cette SB formant des cellules.

A chaque station de base correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire, généralement fiable et d'un débit élevé. Les liaisons sans fil ont une bande passante limitée qui réduit sévèrement le volume des informations échangées. Dans ce modèle, une unité mobile ne peut être, à un instant donné, directement connectée qu'à une seule station de base. Elle peut communiquer avec les autres sites à travers la station à laquelle elle est directement rattachée [30].

b. Réseaux sans fil sans infrastructure (Ad hoc)

Ne comporte pas de site fixe, tous les sites du réseau communiquent d'une manière directe en utilisant leurs interfaces de communication sans fil (voir figure II.5). L'absence d'infrastructure ou de réseau filaire composé de stations de base, oblige les unités mobiles à se comporter comme des routeurs qui participent à la découverte et la maintenance des chemins pour les autres hôtes du réseau [30].

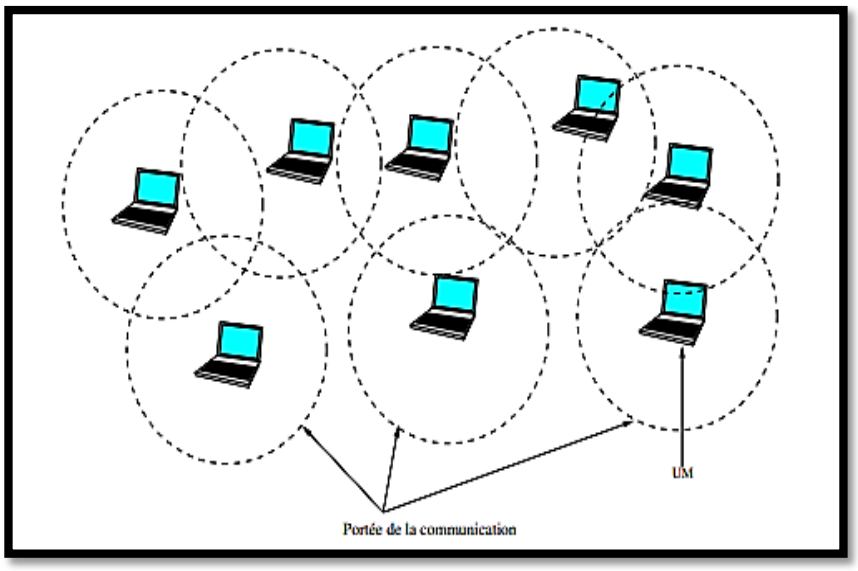


Figure II. 5: Modèle d'un réseau sans fil sans infrastructure [30].

II.4. Réseaux Ad Hoc

II.4.1. Définition

Les réseaux sans fil **Ad Hoc** sont composés de systèmes informatiques divers, plus ou moins complexes, appelés nœuds par la suite, ayant la possibilité de communiquer de manière autonome par ondes radio. Les nœuds interagissent et peuvent coopérer pour s'échanger des services [29]**.**

Chaque nœud dans le réseau Ad Hoc communique avec un autre nœud directement (en utilisant son interface sans fil), si ce dernier est dans sa portée de transmission, ou indirectement par l'intermédiaire d'autres nœuds du réseau dans le cas contraire. Ce dernier doit se comporter comme un terminal, et aussi comme un routeur, et participer à la découverte et la maintenance des routes entre les nœuds du réseau [31]**.**

Ces réseaux sont dits ad hoc dans la mesure où ils ne nécessitent pas d'infrastructure fixe**,** ils offrent une grande flexibilité d'emploi et une grande robustesse et peuvent se déployer très rapidement [31]**.**

De plus, un réseau Ad Hoc est principalement utilisé dans un contexte de mobilité, on parle alors sur les réseaux ad hoc mobiles qui sont un cas particulier du ce réseau.

Une définition formelle des réseaux Ad Hoc **MANET** (**Mobile Ad Hoc NETWORK**) est donnée par la RFC 2501. Il s'agit de réseaux sans fil composé d'un ensemble relativement des nœuds mobiles qui se déplacent librement dans une certaine zone géographique sans aucune infrastructure fixe préexistante [27] .Un protocole adapté doit être utilisé et doit supporter la mobilité des nœuds. Chaque nœud exécute ce protocole pour calculer le prochain nœud relais. Suivant ce protocole, les

nœuds échangent des messages de signalisation entre eux pour avoir connaissance de la topologie en créant un graphe du réseau ou pour créer un chemin vers la destination [32].

II.4.2. Modélisation

Un réseau ad hoc peut être modéliser par un graphe **Gt = (Vt,Et)** .

Où : Vt représente l'ensemble des nœuds (i.e. les unités ou les hôtes mobiles) du réseau.

Et modélise l'ensemble des connections qui existent entre ces nœuds.

Si **e = (u, v) ∈ Et**, cela veut dire que les nœuds **u** et **v** sont en mesure de communiquer directement à l'instant **t** [33].

La figure suivante représente un réseau ad hoc de 10 unités mobiles sous forme d'un graphe :

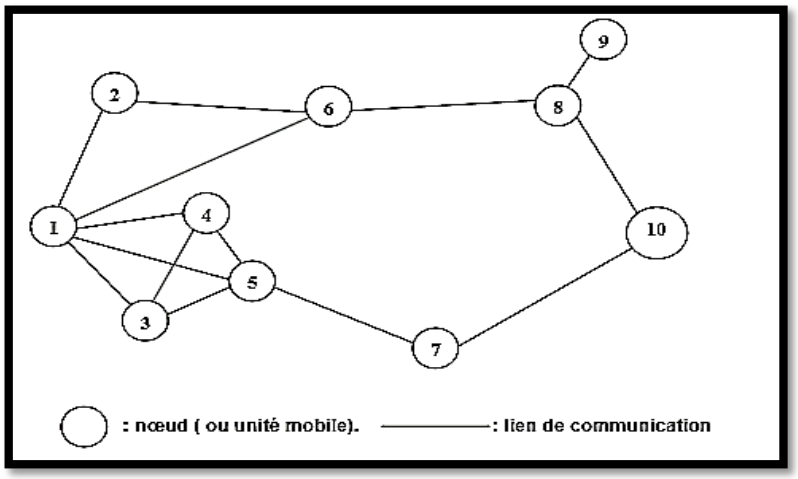


Figure II. 6: Modélisation d'un réseau Ad Hoc [33].

II.4.3. Caractéristiques

Les réseaux ad hoc sont caractérisés principalement par :

- ✓ **Une topologie dynamique :** La topologie des réseaux Ad hoc change rapidement, et aléatoirement, ceci est causé par la mobilité arbitraire des nœuds du réseau. Le changement de la topologie change les routes entre les nœuds et provoque la perte de paquets. La figure II.7 illustre la topologie d'un réseau Ad Hoc avant et après le déplacement d'un nœud [34]**.**

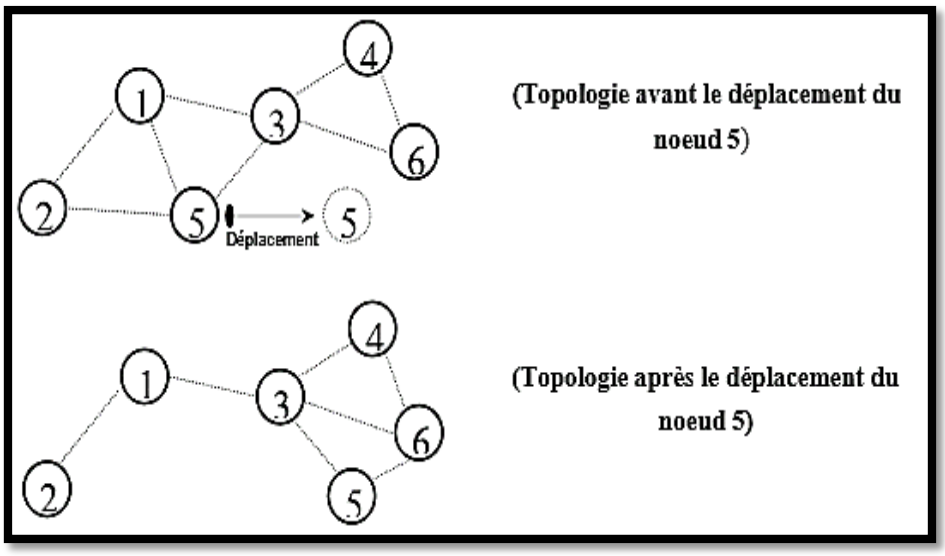


Figure II. 7: Changement de la topologie d'un réseau Ad Hoc [31].

- ✓ **Une Bande passante limitée :** La communication dans les réseaux Ad hoc se base sur le partage d'un médium sans fil (onde radio). Ce qui induit une bande passante modeste pour chaque hôte du réseau [34]**.**
- ✓ **Des contraintes d'énergie :** Les nœuds mobiles dans les réseaux Ad hoc sont alimentés par des sources d'énergie autonomes comme les batteries ou les autres sources consommables, la consommation d'énergie devient alors un problème important [34], (voir figure II.8).

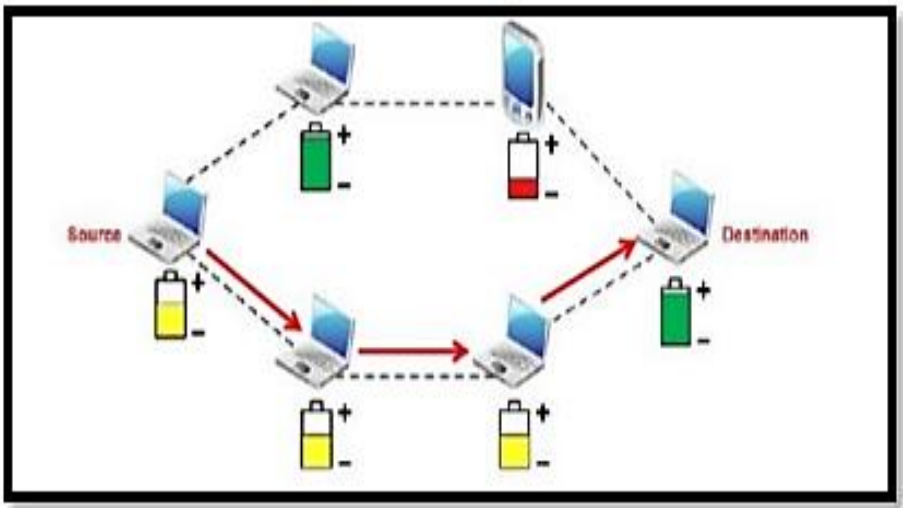


Figure II. 8: Durée de vie de batterie des nœuds [27].

- ✓ **Une sécurité physique limitée :** Les réseaux ad hoc sont plus touchés par le paramètre de sécurité, que les réseaux filaires classiques. Cela se justifie par les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé [28]**.**

- ✓ **Absence d'infrastructure :** Les réseaux ad hoc ne dépendent d'aucune infrastructure préétablie, ceci rend la gestion du réseau plus complexe [34]**.**
- ✓ **Rapidité de déploiement :** Les réseaux Ad hoc peuvent être facilement installés dans les endroits difficiles à câbler, ce qui élimine une bonne part du travail et du coût généralement liés à l'installation et réduit d'autant le temps nécessaire à la mise en route [35]**.**
- ✓ **Equivalence des nœuds du réseau** : Dans un réseau classique, il existe une distinction nette entre les nœuds terminaux (stations, hôtes) qui supportent les applications et les nœuds internes (routeurs par exemple) du réseau, en charge de l'acheminement des données. Cette différence n'existe pas dans les réseaux Ad hoc car tous les nœuds peuvent être amenés à assurer des fonctions de routage [36]**.**
- ✓ **Communication par lien radio** : Les communications entre les nœuds se font par l'utilisation d'une interface radio. Il est alors important d'adopter un protocole d'accès au médium qui permet de bien distribuer les ressources radio et ceci en évitant le plus possible les collisions et en réduisant les interférences. Les technologies de communication sans fil sont alors indispensables à la mise en place d'un réseau Ad hoc [35]**.**
- ✓ **Auto-configuration :** L'absence d'une entité centrale d'administration exige que les nœuds doivent s'auto-configurer et s'auto-organiser afin de garantir la flexibilité et l'adaptabilité requises [34]**.**
- ✓ **Auto-organisation :** Elle permet d'émerger un comportement global à partir des interactions des nœuds locaux sans aucune administration centrale ni hiérarchie globale [37]**.** Et le terme d'auto-organisation dans les réseaux ad hoc est utilisé dans le sens autonome. Ainsi, qu'un protocole de routage peut être dit auto-organisé car les nœuds collaboreront de façon autonome pour établir une fonction du routage, sans intervention extérieure, sans paramétrage manuel. Ainsi, tout protocole conçu pour les réseaux ad hoc est auto-organisé par nature puisqu'il doit fonctionner librement, et s'adapter aux insertions ou suppressions de nœuds [38]**.**

II.4.4. Avantages et inconvénients

Les réseaux Ad Hoc se caractérisent par plusieurs avantages et inconvénients :

Avantages	Inconvénients
Pas de câblage : L'absence d'un câblage et cela en éliminant toute les connexions filaires qui sont remplacées par des connexions radio.	**Topologie non prédictible :** L'activité permanant et les déplacements fréquents des nœuds d'un réseau Ad Hoc rendent son étude très difficile.
Déploiement facile : L'absence du câblage donne plus de souplesse et permet de déployer un réseau Ad Hoc facilement et rapidement et ceci peut être justifié par l'absence d'une infrastructure préexistante permettant ainsi d'économiser tout le temps de déploiement et d'installation du matériel nécessaire.	**Capacités limitées** : Dans un tel réseau Ad Hoc la configuration de la portée de communication des nœuds est important et il faut qu'elle soit suffisante pour assurer la connectivité du réseau, mais plus on accroit la portée des mobiles plus les communications demandent de l'énergie. Il faut donc trouver un compromis entre la connectivité du réseau et la consommation énergétique.
Consommation énergétique : Un nœud émet plus de message en modes Ad Hoc qu'en mode infrastructure puisqu'il doit à la fois transmettre ses propres paquets mais également les paquets des autres nœuds.	**Taux d'erreur important :** Les risques de collision augmentent avec le nombre de nœuds qui partagent le même médium.
La mobilité : Les réseaux Ad Hoc permettent une certaine mobilité à leurs nœuds, de ce fait ces derniers peuvent se déplacer librement à condition de ne pas s'éloigner trop les uns des autres pour garder leur connectivité.	**Sécurité :** Un autre problème des réseaux Ad Hoc et qui attire la curiosité des chercheurs et des spécialistes de ce domaine est la notion de sécurité, un réseau Ad Hoc ne permet pas d'assurer la confidentialité de l'information échanger entre les nœuds contrairement au réseau filaire.
Coût : Le déploiement d'un réseau Ad Hoc ne nécessite pas d'installer des stations de base. Il nécessite que des nœuds, ce qui conduit à la réduction de son coût d'une manière significative.	

Tableau II. 1: Avantages et inconvénients des réseaux Ad Hoc [34].

II.4.5. Applications

Les applications ayant recours aux réseaux Ad hoc couvrent un très large spectre, incluant les applications militaires et de tactique, l'enseignement à distance, les opérations de secours…etc. D'une façon générale, les réseaux ad hoc sont utilisés dans toutes applications où le déploiement d'une infrastructure réseau filaire est trop contraignant soit parce qu'elle est difficile à mettre en place, soit parce que la durée d'installation du réseau ne le justifie pas [39].

On distingue entre autre :

- **Les services d'urgence :** Opérations de recherche et de secours des personnes, tremblement de terre, feux, inondation, dans le but de remplacer l'infrastructure filaire. Le déploiement d'un réseau Ad Hoc est indispensable pour permettre aux unités de secours de se communiquer [39] , (voir, figure II.9).

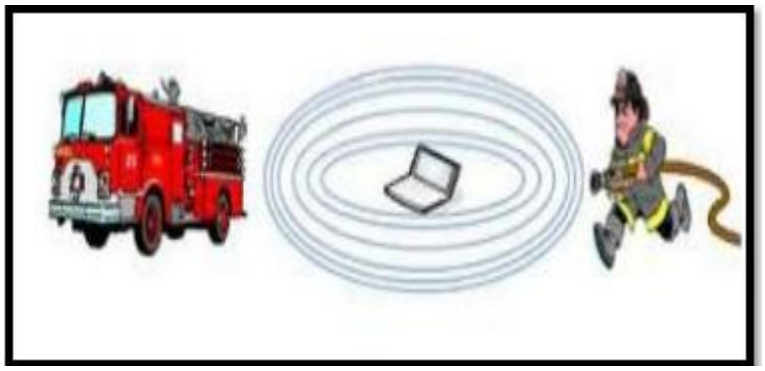


Figure II. 9: Applications de secours des réseaux Ad Hoc [35].

- **L'informatique embarquée :** Dans des véhicules communiquant par exemple.
- **Les entreprises :** Dans le cadre d'une réunion ou d'une conférence [34]**.**
- **Les gares et aéroports :** Pour la communication et la collaboration entre les membres du personnel [34]**.**

II.5. Routage dans les réseaux Ad Hoc

II.5.1. Définition du routage

Le routage est une méthode d'acheminement des informations vers la bonne destination à travers un réseau de connexion donnée, il consiste à assurer une stratégie qui garantit, à n'importe quel moment, un établissement de routes qui soient correctes et efficaces entre n'importe quelle paire de nœud appartenant au réseau, ce qui assure l'échange des messages d'une manière continue. Vu les limitations des réseaux ad hoc, la construction des routes doit être faite avec un minimum de contrôle et de consommation de la bande passante [33]**.**

Son intérêt consiste à trouver le chemin optimal au sens d'un certain critère de performance (bande passante, délai, etc.). Il doit aussi être capable de s'adapter aux événements venant perturber le réseau (panne, congestion, etc.).Son problème réside dans l'effet de trouver l'investissement de moindre coût en capacités nominales et de réserves qui assure le routage du trafic nominal et garantit sa serviabilité en cas de n'importe quelle panne d'arc ou de nœud [33]**.**

Exemple : Si on suppose que les coûts des liens sont identiques, le chemin indiqué dans la figure suivante est le chemin optimal reliant la station source et la station destination. Une bonne stratégie de routage utilise ce chemin dans le transfert des données entres les deux stations.

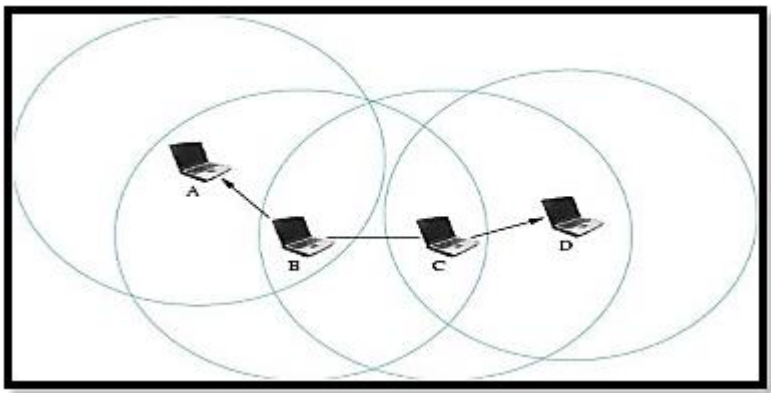


Figure II. 10: Chemin utilisé dans le routage entre la source et la destination [40].

II.5.2. Définition du routage dans les réseaux ad hoc

Le routage est la tâche d'acheminement de flux des données à partir des nœuds sources vers les nœuds destinations. Si une seule destination est impliquée dans la communication, alors il s'agit d'un **"routage unicast"**, si encore tous les nœuds du réseau ou juste un sous ensemble sont concernés par la réception des données alors on parle du **"broadcast"** et du **"routage multicast",** respectivement [41], (voir figure II.11)**.**

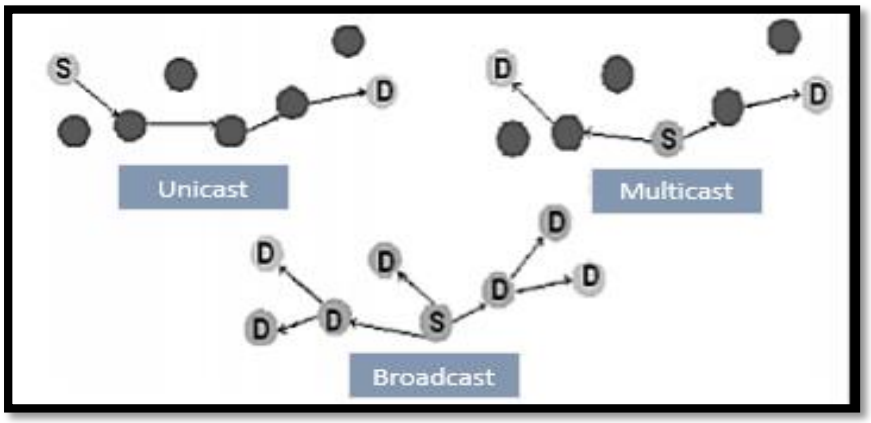


Figure II. 11: Illustration du routage unicast, multicast et broadcast [41].

La figure ci-dessus montre le routage pour une seule destination 'unicast', 'multicast' plusieurs destinations, 'broadcast' tous les nœuds sont destinations.

L'objectif principal des protocoles de routage est l'établissement et la maintenance des chemins, pour que les données soient correctement délivrées dans le réseau [42]**.** La conception des protocoles de routage pour les réseaux Ad hoc est loin d'être un problème simple. Des nouvelles approches de routage sont nécessaires pour effectuer un routage de données sûr et efficace. L'instabilité du médium de communication sans fil, la limitation d'énergie et de la bande passante, ainsi que la mobilité des nœuds introduisent plus de difficulté et de complexité à la conception des protocoles de routage pour les réseaux Ad hoc.

II.5.3. Difficulté

De fait qu'un réseau ad hoc est un ensemble de nœuds qui peuvent être mobiles, qui sont dynamiquement et arbitrairement éparpillés d'une manière ou l'interconnexion entre les nœuds peut changer à tout moment. Il se peut qu'un hôte destination soit hors de la portée de communication d'un hôte source, ce qui nécessite l'emploi d'un routage interne par les nœuds intermédiaires afin de faire acheminer les paquets de message à la bonne destination [43].

En effet, la topologie évoluant constamment en fonction des mouvements des mobiles, Le problème qui se pose dans le contexte des réseaux Ad hoc est l'adaptation de la méthode d'acheminement utilisée avec le grand nombre d'unités existant dans un environnement caractérisé par de modestes capacités de calcul et de sauvegarde. D'ailleurs dans la pratique il est impossible qu'un hôte puisse garder les informations de routage concernant tous les autres nœuds, dans le cas où le réseau serait volumineux [43].

II.5.4. Contraintes

L'étude et la mise en œuvre d'algorithmes de routage pour assurer la connexion des réseaux Ad hoc au sens classique du terme (tout sommet peut atteindre tout autre), est un problème complexe. L'environnement est dynamique et évolue donc au cours du temps, la topologie du réseau peut changer fréquemment. Il semble donc important que toute conception de protocole de routage doive étudier les problèmes suivants [44]:

- **Minimisation de la charge du réseau :** L'optimisation des ressources du réseau renferme deux autres sous problèmes qui sont l'évitement des boucles de routage, et l'empêchement de la concentration du trafic autour de certains nœuds ou liens.
- **Offrir un support pour pouvoir effectuer des communications multipoints fiables :** Le protocole doit permettre d'effectuer des échanges de données fiables entre les différents nœuds du réseau.
- **Nécessite une distribution des opérations :** Le protocole doit être entièrement distribué [31].
- **Assurer un routage optimal :** La stratégie de routage doit créer des chemins optimaux et pouvoir prendre en compte différentes métriques de coûts (bande passante, nombre de liens, ressources du réseau,... etc.). la stratégie de routage doit assurer une maintenance efficace de routes avec le moindre coût possible.

- **Le temps de latence :** La qualité des temps de latence et de chemins doit augmenter dans le cas où la connectivité du réseau augmente**.**

II.5.5. Routage dans le modèle OSI

Le modèle de référence de connexion entre systèmes ouverts **OSI** (reference model of Open System Interconnexion) a pour objectif de permettre la création de logiciels réseaux modulaires et réutilisables indépendamment des techniques mises en œuvre. Le modèle OSI propose une architecture composée de sept couches (figure II.12).Chaque couche propose un service aux couches qui lui sont adjacentes [31]**.**

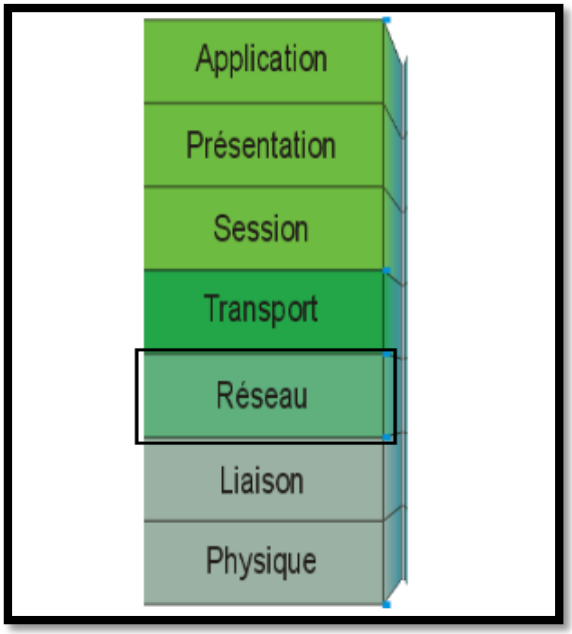


Figure II. 12: Modèle OSI.

La figure II.12 représente les sept couches du modèle de référence OSI en mettant l'accent sur la couche réseau qu'est associée essentiellement au routage.

Chaque couche du modèle de référence OSI a une fonction spécifique :

- **La couche physique** : Sert à transmettre les données via un canal de communication et comprend le matériel nécessaire à la réalisation de cette fonction tel que : les moyens électriques, mécaniques, optiques ou hertziens.
- **La couche liaison** : Gère la fiabilité du transfert de bits d'un nœud à l'autre du réseau, comprenant, entre autres, les dispositifs de détection et de correction d'erreurs, ainsi que les systèmes de partage des supports.

- **La couche réseau :** Détermine le meilleur chemin afin d'échanger les données entre les nœuds du réseau .Le routeur fonctionne au niveau de cette couche. Elle recourt à des méthodes d'adressage logique qui peuvent être gérées par un administrateur. Elle utilise les méthodes d'adressage IP (Internet Protocol), IPX...etc [45]**.**
- **La couche transport :** Regroupe les règles de fonctionnement de bout en bout, Elle traite notamment l'établissement des connexions et la fiabilité du transport.
- **La couche session** : Réunit les procédures de dialogue entre les applications : établissement et interruption de la communication, cohérence et synchronisation des opérations.
- **La couche présentation** : Traite les formes de représentation des données, permettant la traduction entre machines différentes.
- **La couche application :** Source et destination de toutes les informations à transporter, la couche application rassemble toutes les applications qui ont besoin de communiquer par le réseau : messagerie électronique, transfert de fichiers, gestionnaire de bases de données, etc.

Les protocoles de routage sont associés à la couche réseau du modèle OSI car c'est dans cette couche qu'on détermine les routes qui doivent être empruntés par les paquets de données [31]**.**

II.6. Problèmes de conception des protocoles de routage pour les réseaux ad hoc

Dans cette section nous aborderons les principaux problèmes de conception des réseaux ad hoc mobile.

II.6.1. Architecture de routage

L'architecture de routage des réseaux auto-organisés peut être soit hiérarchique, soit plate. Dans la plupart des réseaux auto-organisés, les hôtes agissent comme des routeurs indépendants, ce qui implique que l'architecture de routage doit être conceptuellement plate (c'est-à-dire que chaque adresse sert uniquement d'identifiant et ne transmet aucune information sur la localisation topologique d'un hôte par rapport à un autre nœud). Dans un réseau plat auto-organisé, la gestion de la mobilité n'est pas nécessaire puisque tous les nœuds sont visibles les uns par rapport aux autres via des protocoles de routage [46].

Dans un algorithme de routage **plat**, les frais généraux de routage augmentent plus rapidement lorsque la taille du réseau augmente. Par conséquent, pour contrôler la réutilisation des canaux dans l'espace (en termes de fréquence, de temps ou de code d'étalement) et réduire le surcroît d'informations de routage, il convient d'utiliser une forme de schéma hiérarchique. Le regroupement est la technique la plus couramment utilisée dans les architectures de routage hiérarchique.

L'idée du routage **hiérarchique** est de diviser les hôtes des réseaux auto-organisés en un certain nombre de clusters qui se chevauchent ou sont disjointes. Un nœud est élu comme chef de cluster pour chaque cluster. Ce chef de cluster conserve les informations sur les membres du cluster. Les autres nœuds présents dans le cluster seront traités comme des nœuds ordinaires. Lorsque ces nœuds veulent envoyer un paquet, ils peuvent envoyer le paquet à la tête du cluster qui l'achemine vers la destination. Le routage par commutation de passerelle de tête de cluster (**CGSR** Cluster Switch Gateway Routing) et le protocole de routage basé sur le cluster (**CBRP** Cluster Based Routing Protocol) appartiennent à ce type de schéma de routage. Le routage hiérarchique implique la gestion des clusters et la gestion des adresses/mobilité [46].

II.6.2. Prise en charge des liens unidirectionnels

Même si l'on suppose que tout protocole de routage est bidirectionnel, un certain nombre de facteurs feront que les liens sans fil seront unidirectionnels [46]**:**

- **Différentes capacités radio :** Différents nœuds peuvent avoir des puissances d'émission et de réception différentes au sein d'un réseau.
- **Interférences :** Elles sont dues soit à des brouilleurs hostiles, soit à des interférences amicales, qui réduisent la sensibilité d'un récepteur proche. Par exemple, l'hôte X peut recevoir des paquets de l'hôte Y car il y a très peu d'interférences à proximité de X. Cependant, Y peut se trouver à proximité d'un nœud d'interférence et ne peut donc pas recevoir de paquets de X. Ainsi, la liaison entre X et Y est dirigée de Y vers X.
- **Exigence de diffusion du message :** Pour les liaisons ascendantes, des émetteurs par satellite sont utilisés. Les liaisons montantes utilisent différents types de chemins alternatifs.
- **Mode silencieux :** Un cas extrême, applicable uniquement dans les réseaux mobiles tactiques, est celui où les hôtes ne peuvent pas transmettre en raison d'une menace imminente. Dans ce cas, ils doivent quand même recevoir des informations, mais ne peuvent pas participer à des communications bidirectionnelles.
- **L'état de la direction de la liaison est variable dans le temps :** Dans un diagramme d'état, si la communication sans fil est représentée, l'état de la liaison sans fil peut être soit un phénomène persistant, soit un phénomène transitoire. La durée du séjour dans un état particulier peut dépendre d'une fonction du trafic offert, de l'environnement et de la disponibilité de l'énergie dans les nœuds.

II.6.3. Routage de la qualité de service (QoS)

Dans le cadre d'une qualité de service, le but du protocole de routage est de trouver la meilleure route selon les critères précis de la qualité de service souhaitée (délai, taux de perte, quantité de bande passante, ...), et reposant sur des liens fiables. Ce dernier point est à la fois important et difficile à assurer dans le cas des réseaux ad hoc en raison de leur topologie dynamique.

Les nœuds constituant le réseau ad hoc doivent stocker et mettre à jour les états des liens dans un environnement qui est mobile. Ce processus est donc très complexe et couteux car des ruptures de liens peuvent intervenir à tout moment beaucoup plus fréquemment que dans des réseaux classiques [47].

Le routage QoS repose sur des paramètres de la qualité de service [48] tels que :

-Délai : C'est le temps écoulé entre l'envoie d'un paquet par la source et sa réception par le destinataire.

-Latence : Représente le retard entre l'émission et la réception d'un paquet.

-Débit : Le débit binaire ou par abus de langage, la bande passante entre deux entités communicantes est le nombre de bit que le réseau est capable d'accepter ou de délivrer par unité de temps.

-Taux de perte de paquet (packet loss ratio) : Correspondant le rapport du nombre de paquets non livrés sur le nombre total de paquets transmis.

-Dé-séquencement : Il s'agit d'une modification de l'ordre d'arrivée des paquets.

Il est souhaitable de choisir l'itinéraire avec le moindre coût sur la base de ces mesures, plutôt que de ne fournir que l'itinéraire le plus court en fonction de la distance de saut.

II.6.4. Prise en charge de la multidiffusion

La multidiffusion prend en charge les communications de groupe, en particulier dans le cas des réseaux ad hoc, où le réseau est auto-organisé, où la largeur de bande est limitée et où l'énergie est restreinte. Ces réseaux se composent de plusieurs groupes de travail coopératifs. Le déploiement du routage multidiffusion dans des réseaux auto-organisés permettra une visualisation et une conférence multimédia en collaboration ainsi que la diffusion d'informations dans des situations critiques telles que des catastrophes ou des scénarios militaires [46].

Un réseau auto-organisé est mieux adapté à la multidiffusion qu'au routage de l'unicast en raison de ses caractéristiques de diffusion. Le routage de la multidiffusion dans les réseaux auto-organisés pose de nouveaux défis. Les protocoles de multidiffusion traditionnels ne sont pas adaptés à cet environnement pour les raisons suivantes [46]:

1. Les protocoles orientés source sont inefficaces car la source à l'origine de la demande d'acheminement se déplace.

2. Lorsque les nœuds des réseaux auto-organisés se déplacent, ils modifient la topologie des réseaux, de ce fait le routage peut être difficile.

3. Des boucles transitoires peuvent se former lors de la reconfiguration de l'arborescence.

4. Comme la communication s'effectue vers un groupe de nœuds, le maintien d'un trop grand nombre d'informations d'état liées au multicast exerce une forte pression sur la capacité de stockage et la puissance, et ces ressources sont fortement limitées dans les appareils portables des réseaux auto-organisés.

II.7. Classification des protocoles de routage

II.7.1. Critères

Afin de résoudre le problème de difficulté de routage dans les réseaux ad hoc, les stratégies existantes utilisent une variété de techniques. Nombreuse classifications sont apparues suivant ces critères, parmi lesquelles nous allons citer :

II.7.1.1. Évaluation de topologie, de destination ou de position pour le routage

Selon le type d'information exploité dans le routage on distingue [49] [50]**:** les protocoles basés topologie, basés destination ou basés position. Dans un protocole de routage basé topologie, chaque nœud maintient une image complète de la topologie du réseau pour pouvoir prendre ses décisions de routage tandis que dans un protocole de routage basé destination, un nœud maintient uniquement des informations sur ses sauts prochains vers chacune des destinations possibles.

Un protocole de routage basé position utilise les informations sur la localisation du nœud source par rapport au nœud destination, et sur la mobilité de ces derniers dans les phases de découverte de chemins ou de routage des données.

II.7.1.2. Structures de routage (hiérarchique ou plat)

a. Protocoles de routage hiérarchique

Cette approche est basée sur la formation de clusters (zones communes). Le principe est de router les données récoltées par chaque nœud du cluster à son chef de zone (Cluster Head), et après des traitements sur leurs parties communes, les transmettra à la prochaine destination (Si le CH ne pourra pas atteindre directement la station de destination, les informations seront routées vers le prochain chef de zone). L'avantage est la réduction des coûts en communication et en

énergie en minimisant le nombre de messages circulant sur le réseau, étant donné que les CHs appliquent des fonctions d'agrégat sur les données du cluster ce qui permet de les combiner. L'inconvénient concerne la taille du réseau. En outre, quand la taille du réseau augmente, le processus d'élection du Cluster Head devient critique et gourmand en ressources [51]**,** (voir la figure II.13)**.**

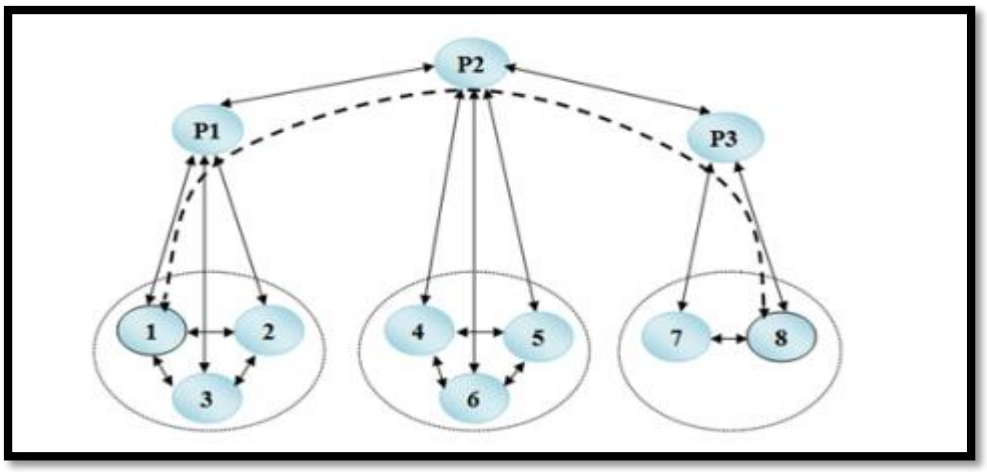


Figure II. 13: Routage hiérarchique [52].

b. Protocoles de routage à plat

Ces protocoles considèrent que tous les nœuds sont identiques, c'est à dire ont les mêmes fonctions à exécuter sauf le nœud de contrôle (sink) qui est chargé de collecter toutes les informations issues des différents nœuds capteurs pour les transmettre vers l'utilisateur final. La décision d'un nœud de router des paquets vers un autre dépendra de sa position et pourra être remise en cause au cours du temps [53], (voir la figure II.14).

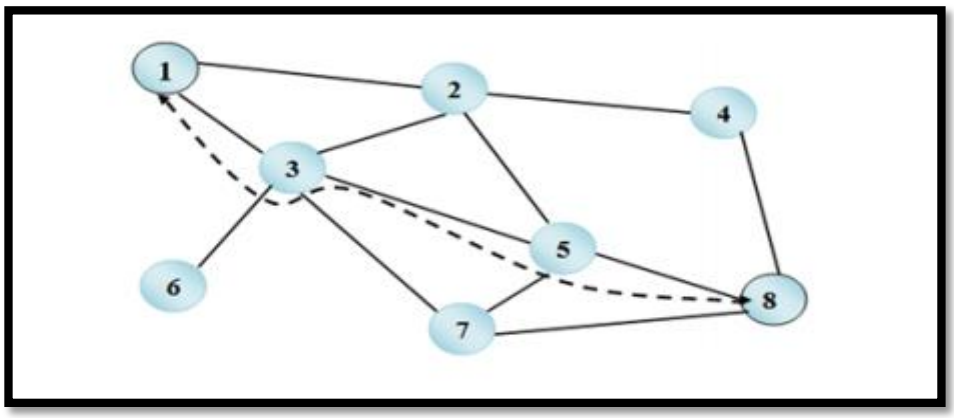


Figure II. 14: Routage plat [52].

Ces protocoles de routage sont des protocoles non hiérarchiques dans lesquels les nœuds jouent le même rôle et ont la même tâche. Parmi les protocoles utilisant cette technique, on cite DSDV, DSR, OLSR [54].

II.7.1.3. Routage proactif, réactif ou hybride

Parmi les méthodes les plus utilisées pour la différentiation des protocoles de routage dans les réseaux ad hoc, on retrouve celle basée sur la façon d'acquisition et du maintien des informations de routage par les nœuds mobiles. En utilisant cette méthode, les protocoles de routage peuvent être classifiés en des protocoles proactifs, réactifs ou hybrides [49] [50] [53]. Les protocoles proactifs effectuent des mises à jour périodiques des chemins pour qu'un paquet, dès que nécessaire, soit immédiatement transmis. Les protocoles de routage réactifs, quant à eux, n'effectuent des découvertes de chemins qu'à la demande. L'approche hybride repose sur une organisation hiérarchique des nœuds dans le réseau et applique à chaque niveau de la hiérarchie une approche de routage appropriée (proactive ou réactive).

II.7.1.4. Métriques exploitées dans le routage

Les métriques utilisées dans la construction des chemins peuvent être utilisées également comme un critère de classification des protocoles de routage [55]. Plusieurs protocoles de routage sont basés sur le nombre de sauts comme métrique en choisissant le chemin avec le nombre de sauts minimal. D'autres protocoles de routage prennent en considération la stabilité des liens lors de la phase de construction des chemins. Un protocole de routage peut prendre en compte plusieurs métriques comme le délai, la bande passante et le taux d'erreurs pour pouvoir satisfaire les différents requis de QoS (Qualité de Service) de certaines applications.

II.7.1.5. Techniques de routage

a. Routage à la source

Le routage à la source ou << **source Routing** >> consiste à indiquer dans le paquet routé l'intégralité du chemin que devra suivre le paquet pour atteindre sa destination. L'entête de paquet va donc contenir la liste des différents nœuds relayeur vers la destination. Le protocole le plus connu basant sur cette classe est : DSR [56].

b. Routage saut par saut

Le routage saut par saut ou <<**hop by hop**>> consiste à donner uniquement à un paquet l'adresse du prochain nœud vers la destination. AODV fait partie des protocoles qui utilisent cette technique [56].

II.7.1.6. Algorithmes de routage

a. Vecteur de distance

Ils sont utilisés dans des petits réseaux, et ils ont un nombre limité de sauts, alors que les protocoles à état de lien peuvent être utilisés dans des réseaux plus grands, et ils ont un nombre illimité de sauts. Il signifie que les routes sont échangées entre routeurs en indiquant **la direction et la distance.**

b. Etat de lien

Dans le protocole de routage **à état de lien**, chaque nœud construit une carte de chaque connectivité autour d'un routeur. Chaque routeur a une connaissance complète du routeur auquel il est connecté et ajoute les meilleures routes à leurs tables de routage en fonction de la métrique utilisée.

II.7.1.7. Inondation

L'inondation ou la diffusion pure, consiste à répéter un message dans tout le réseau. Un nœud qui initie l'inondation envoie le paquet à tous ses voisins directes, de même si un nœud quelconque de réseau reçoit le paquet pour la première fois, il le rediffuse à tous les voisins, Ainsi de proche en proche le paquet inonde le réseau. Notons que les nœuds peuvent appliquer (durant l'inondation) certains traitements de contrôle dans le but d'éviter certains problèmes, tels que le blocage et la duplication des messages, Le mécanisme d'inondation est utilisé généralement dans la première phase du routage plus exactement dans la procédure de découverte des routes, et cela dans le cas où le nœud source ne connait pas la localisation exacte de la destination [57] , (voir, figure II.15).

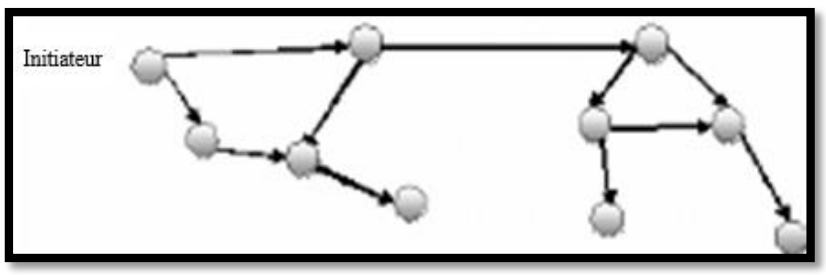


Figure II. 15: Mécanisme d'inondation [56].

II.7.1.8. Concept du groupe

Dans la communication de groupes, les messages sont transmis à des entités abstraites ou groupes, les émetteurs n'ont pas besoin de connaître les membres du groupe destinataire. La gestion des membres d'un groupe dynamique permet à un élément de se joindre à un groupe, de quitter ce groupe, se déplacer ailleurs puis rejoindre le même groupe. C'est en ce sens que la communication de groupe assure une indépendance de la localisation ce qui la rend parfaitement adaptée à des topologies de réseaux reconfigurables telles que les architectures avec sites mobiles [28] , (voir figure II.16).

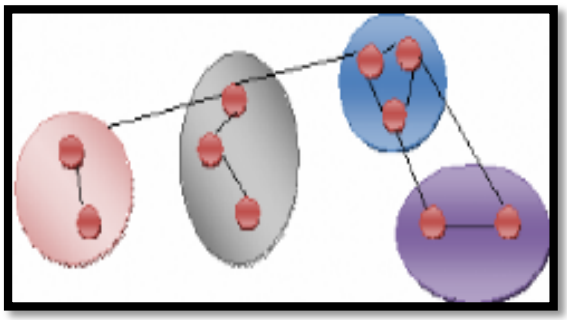


Figure II. 16: Décomposition du réseau en groupe [56].

II.7.1.9. Routage uniforme ou non uniforme

Une autre méthode de classification est basée sur le rôle que jouent les nœuds dans un schéma de routage. Selon ce critère de classification, on distingue les protocoles de routage uniformes et non uniformes [55] [58]. Dans un protocole de routage uniforme, tous les nœuds mobiles ont le même rôle, importance et fonctionnalité. Dans un protocole de routage non uniforme, certains nœuds ont des fonctions de gestion et de routage particulières. Les protocoles non uniformes à leur tour peuvent être subdivisés selon l'organisation des nœuds mobiles et selon les stratégies de gestion et de routage en : des protocoles basés sur les zones, basés sur les clusters ou basés sur des nœuds noyaux [55].

Pour donner une vue plus claire sur les critères de classification des protocoles de routage, nous allons présenter ci-dessous un schéma explicatif.

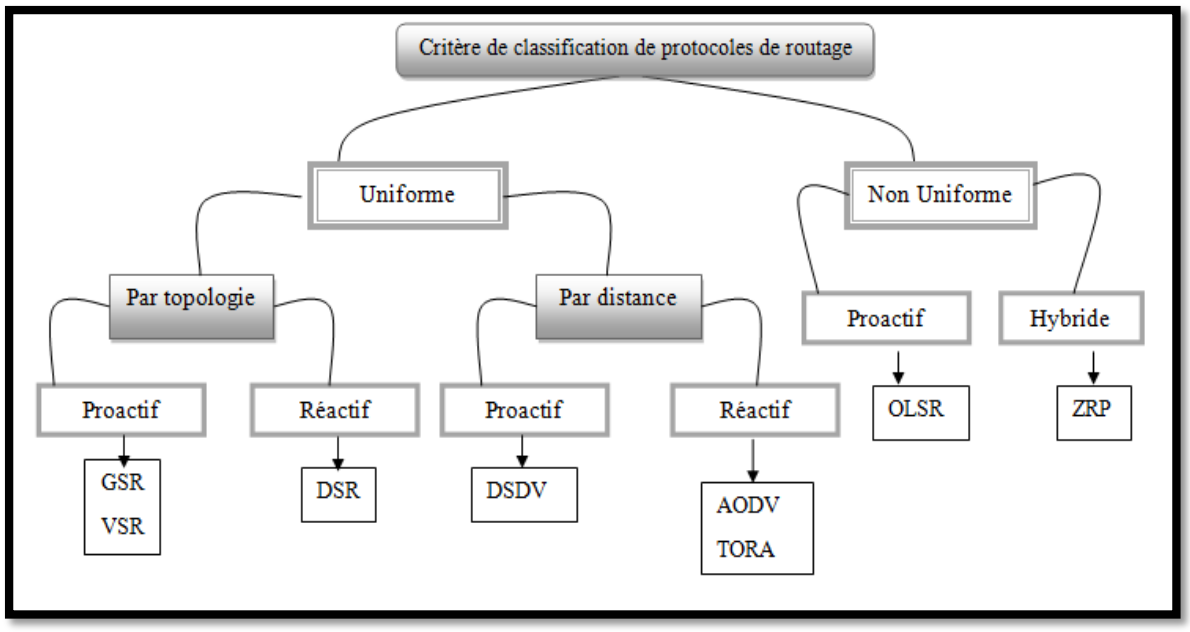


Figure II. 17: Schéma récapitulatif des critères de classification des protocoles de routage.

II.7.2. Classification des protocoles de routage

Suivant la manière de création et de maintenance de routes lors de l'acheminement des données, les protocoles de routage peuvent être séparés en : **Proactif**, **Réactif** et **Hybride**. Comme il est illustré dans la figure II.18.

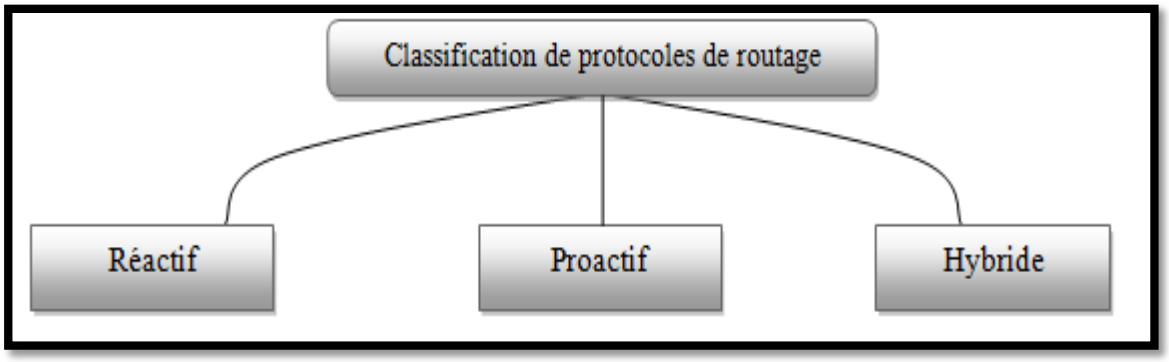


Figure II. 18: Classification des protocoles de routage.

De nombreux protocoles et algorithmes ont été proposés pour rendre la communication dans les réseaux Ad Hoc plus efficace. Et leurs performances ont été analysées dans différentes situations.

II.7.2.1. Protocoles proactifs

Les protocoles de cette catégorie sont basés sur les algorithmes classiques d'état de liens et de vecteur de distance. Les protocoles de routage proactifs essaient de maintenir les meilleurs chemins existants vers toutes les destinations possibles au niveau de chaque nœud du réseau. Les

routes sont sauvegardées même si elles ne sont pas utilisées. La mise à jour permanente des tables de routage, est assurée par un échange continu des messages de mise à jour des chemins. Lorsqu'un nœud reçoit un paquet de contrôle, il met à jour ses tables de routages.

Ainsi, de nouvelles routes seront construites sur la base des informations topologiques transportées par les trames de contrôle. Ce processus est déclenché aussi à chaque changement de topologie pour reconstruire à nouveau les routes [59]**.**Les protocoles basés sur ce principe sont entre autre : DSDV, WRP, OLSR, TBRPF, GSR, FSR, HSR, ZHLS, CGSR, DREAM et LSR.

II.7.2.1.1. Avantages

- Pas de temps de réaction.
- Adaptés aux réseaux denses de taille moyenne.
- Adaptés aux réseaux à forte mobilité.

II.7.2.1.2. Inconvénients

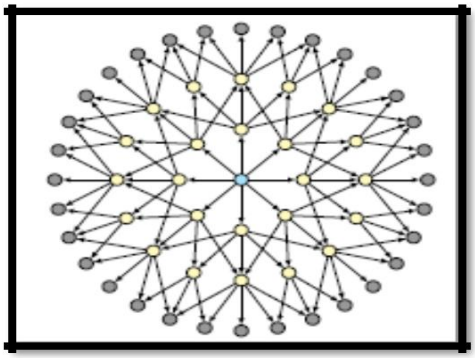
- Trafic de contrôle important.
- Capacité d'échange du réseau limitée.

II.7.3.1.3. Quelque protocoles de routage proactif

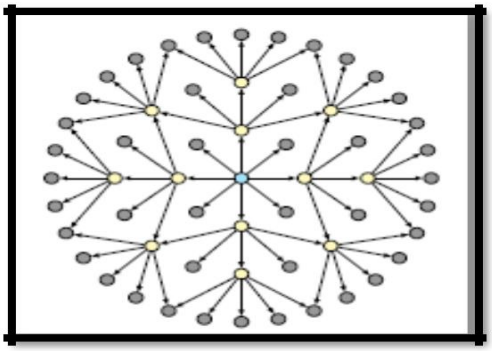
a. OLSR (Optimized Link State Routing)

OLSR [60] est un protocole proactif à état de lien (Link state), il apporte certaines améliorations sur le principe de base de l'état de lien en vue d'atteindre de meilleures performances dans un contexte Ad Hoc :

Il permet de minimiser l'inondation du réseau en diminuant les retransmissions redondantes dans la même région du réseau et réduit la taille des paquets échangés. Pour cela, OLSR se base essentiellement sur la notion de relais multipoint (MPR, Multi Point Relay), un sous ensemble des voisins à un saut qui permet d'atteindre la totalité des voisins à deux sauts. Ainsi, lors d'une diffusion, tous les voisins reçoivent et traitent le message mais seulement les nœuds choisis comme MPR le retransmettent ce qui diminue considérablement le nombre de messages émis dans le réseau, (voir figure II.19).



a. Routage par inondation (24 transmissions pour atteindre tous les nœuds à 3 sauts).



b. Routage avec les nœuds MPR(12 transmissions pour atteindre tous les nœuds à 3 sauts).

Figure II. 19: Avantage de l'utilisation des MPR [60].

OLSR étant proactif, chaque nœud construit en permanence une vision de la topologie du réseau sous forme d'un graphe où les arcs constituent les liens entre les nœuds. La cohérence de cette vision est assurée grâce à des diffusions périodiques des liens sortants. Ainsi, un nœud recevant ces informations met à jour sa vision de la topologie et applique l'algorithme du plus court chemin pour choisir le prochain saut en direction de chaque destination [61]**.**

b. DSDV (Destination-Sequenced Distance-Vector)

DSDV [59] est l'un des premiers protocoles de routage Ad hoc proactifs à vecteur de distance. Il se base sur l'algorithme distribué Bellman-Ford DBF (Distributed Bellman-Ford) [62] qui a été modifié pour s'adapter aux réseaux Ad Hoc. Comme il s'agit d'un protocole proactif, chaque nœud, à chaque instant a une vision complète du réseau. Pour ce faire, chaque nœud récupère les distances le séparant de chaque autre nœud du réseau et ne garde que le plus court chemin. Ceci est fait grâce à des échanges périodiques d'informations sur leurs tables de routage respectives.

II.7.2.2. Protocoles réactifs

Les protocoles de routage réactifs, dits aussi : protocoles de routage à la demande, représentent les protocoles les plus récents proposés dans le but d'assurer le service du routage dans les réseaux sans fil. Les protocoles de routage appartenant à cette catégorie, créent et maintiennent les routes selon les besoins. Lorsque le réseau a besoin d'une route, une procédure de découverte globale de routes est lancée, et cela dans le but d'obtenir une information [63]**.**

Les protocoles de routage proactifs engendrent un trafic très important ce qui conduit, souvent, à la saturation rapide du réseau. Pour y remédier, les protocoles réactifs évitent au maximum les

inondations qui consomment beaucoup de ressources. Les protocoles basés sur ce principe sont entre autre : CBRP, DSR, AODV, TORA, ABR, SSR, LAR, RDMAR, EARP et CEDAR [49].

II.7.2.2.1. Avantages

- Trafic de contrôle faible.
- Adaptés aux grands réseaux.
- Consommation énergétique réduite.

II.7.2.2.2. Inconvénients

- Temps de réaction long.
- Problème en cas de forte mobilité des nœuds.

II.7.2.2.3. Quelques protocoles réactifs

a. DSR (Dynamic Source Routing)

DSR [64] [61] est un protocole à routage source, il est composé de deux mécanismes : la découverte des chemins et la maintenance des chemins. Un nœud qui veut transmettre des données vers une destination, alors qu'il ne maintient aucun chemin vers cette dernière dans sa cache initie une découverte de chemins en diffusant un paquet **RREQ** vers tous ses voisins. Un nœud intermédiaire qui reçoit ce paquet peut répondre de sa cache s'il connaît un chemin vers la destination, sinon il inclut son adresse dans le paquet et le rebroadcast.

Le nœud qui répond au paquet RREQ, s'il est la destination, génère un paquet **RREP** et il inclut dans ce dernier la séquence des nœuds enregistrés dans le RREQ. Sinon, il effectue une concaténation de la séquence des nœuds incluse dans le RREQ avec le chemin enregistré dans sa cache. Si le nœud générant le RREP maintient un chemin vers la source, il envoie le RREP sur ce chemin. Un nœud qui ne maintient pas un chemin pareil transmet le RREP en suivant le chemin inverse de celui dans le RREQ si les liens sont bidirectionnels, sinon une nouvelle découverte de chemins est initiée en incluant la réponse dans un paquet RREQ.

b. AODV (Ad Hoc On demand Distance Vector)

Un des protocoles les plus connus dans cette catégorie est le protocole **AODV** [65] [61]qui n'est en réalité qu'une combinaison entre les deux protocoles DSDV et DSR. En effet, c'est une combinaison de la demande de base de la découverte et le maintien de route de DSR et le routage de saut par saut et des numéros de séquence de DSDV. La route retenue est bidirectionnelle et correspond au plus court chemin (en nombre de sauts) entre la source et la

destination. Chaque nœud maintient une table de routage dont les entrées mémorisent, pour une destination :

- L'identifiant de cette destination.
- L'identifiant du prochain nœud vers cette destination.
- Le nombre de nœuds jusqu'à cette destination.

Quand un nœud source **S** veut atteindre la destination **D** pour laquelle il ne possède pas de route, il inonde le message de demande de route **RREQ** à ses voisins. Le paquet RREQ contient le numéro de séquence pour cette destination, si le numéro de séquence n'est pas connu, la valeur nulle sera prise par défaut, il contient aussi la valeur du numéro de séquence du nœud source. Le RREQ sera propagé jusqu'à ce que le paquet atteigne un nœud qui a une route à la destination. Chaque nœud intermédiaire expédie la demande et crée une route inversée vers **S** (mémorise une route vers la source).

Quand un nœud intermédiaire a une route vers **D**, il produit une **réponse RREP** qui contient le nombre de saut et le numéro de séquence pour **D** (le plus récent). Les nœuds qui portent la réponse vers **S** créent une route vers **D** mais seulement avec le prochain saut et non pas la route toute entière. Cependant, AODV maintient les chemins d'une façon distribuée en gardant une table de routage au niveau de chaque nœud de transit appartenant au chemin recherché.

Pour la mise à jour des routes, le protocole AODV exige des **messages HELLO** toutes les quelques secondes. Un lien est considéré invalide si trois messages HELLO consécutifs ne sont pas reçus (à travers ce même lien).

Quand un lien devient invalide, tout nœud expédiant à travers celui-ci est informé par un paquet **Route Error RERR** avec une métrique égale à l'infini. Ce qui conduit à l'élancement d'une opération de découverte de route.

II.7.2.3. Protocoles hybrides

Dans ce type de protocole, on peut garder la connaissance locale de la topologie jusqu'à un nombre prédéfini (a priori petit) de sauts par un échange périodique de trame de contrôle, autrement dit par une technique proactive. Les routes vers des nœuds plus lointains sont obtenues par schéma réactif, c'est-à-dire par l'utilisation de paquets de requête en diffusion. Un exemple de protocoles appartenant à cette famille est **DSR** (Dynamic Source Routing), qui est réactif à la base

mais qui peut être optimisé s'il adopte un comportement proactif. Un exemple de ces protocoles est le protocole **ZRP** *(***Z**one **R**outinier **P**rotocol*)* [66]**.**

II.7.2.3.1. Avantages et inconvénient des protocoles hybrides

Le protocole hybride est un protocole qui se voit comme une solution mettant en commun les avantages des deux approches précédentes en utilisant une notion de découpe du réseau [57]**.**

Cependant, il rassemble toujours quelques inconvénients des deux approches proactives et réactives.

II.7.2.3.2. Quelques protocoles hybrides

ZRP (Zone Routinier Protocol)

ZRP [66] [61] est un protocole de routage hybride où le réseau est décomposé en plusieurs zones de routages chevauchées. La zone de routage d'un nœud est définie comme l'ensemble des nœuds qui se trouvent à une distance inférieure ou égale au rayon de la zone. Les nœuds qui se trouvent exactement à une distance égale au rayon de la zone sont appelés "nœuds périphériques".

Le routage intra-zone IARP (IntrAzone Routing Protocol) peut être assuré par n'importe quel protocole de routage proactif, à condition qu'il soit modifié pour que la portée des mises à jour soit restreinte au rayon de la zone de routage. Le routage interzone IERP (IntErzone Routing Protocol) est assuré par un cycle **RREQs-RREPs**. Quand un nœud reçoit un paquet RREQ s'il n'est pas destination et si encore la destination ne réside pas dans sa zone, il renvoie le RREQ uniquement vers les nœuds périphériques. Cela limite considérablement le nombre des RREQs propagés dans le réseau.

Les performances de ZRP dépendent de la valeur choisie pour le rayon des zones. Pour des grandes valeurs, ZRP se comporte comme un protocole de routage purement proactif tandis que pour des petites valeurs, il se comporte comme un protocole de routage purement réactif [67]**.**

II.7.3. Comparaison des protocoles de routage

Dans ce qui suit nous allons énumérer quelque protocole de routage en spécifiant pour chacun sa catégorie, type de route et d'autres caractéristiques.

Contrainte de performance	DSDV	OLSR	AODV	DSR	ZRP
Catégorie	Proactif	Proactif	Réactif	Réactif	Hybride
Route sans cycle	Oui	Oui	Oui	Oui	Oui
Routes multiples	Oui	Non	Non	Non	Oui
Multicast	Oui	Oui	Oui	Non	Oui
Surcharge réseau	Minimale	Minimale	Modérée	Modérée	Modérée
Diffusion périodique	Possible	Possible	Possible	Possible	Possible
Principale caractéristique	Des informations sur les destinations avec un numéro de séquence. Envoi périodique aux voisins.	Messages de contrôle pour la détection de liaison, détection des voisins (MPR), et un calcul des routes.	Découverte des routes, avec une recherche, à la poursuite du chemin.	Demande et découverte des routes, des routages à la source, et la maintenance de route.	Chaque nœud identifié les voisins, pour découvrir les routes, IERP est utilisé à la demande pour chercher les routes.

Tableau II. 2: Tableau comparatif des différents protocoles de routage Ad Hoc [68].

Le tableau ci-dessus présente une comparaison entre les différents protocoles de routage Ad-Hoc en mettant l'accent sur les différences et les propriétés pour chaque protocole de routage.

Conclusion

Après avoir abordé dans ce chapitre la description des réseaux sans fil en énumérant leurs types, nous nous sommes intéressées beaucoup plus à une catégorie de réseaux sans fil sans infrastructure particulièrement les réseaux Ad hoc qui font partie d'un domaine de recherche très actif avec l'apparition de plusieurs technologies utilisant la communication en mode Ad hoc notamment les réseaux de capteurs sans fil, la robotique et les réseaux véhiculaires. Pour cela, nous avons traité les réseaux Ad hoc avec des détails sur leurs caractéristiques et l'intérêt qu'apporte ce type de réseaux.

Par la suite, nous avons présenté la notion et les problèmes du routage dans les réseaux Ad hoc. Comme nous avons vu, le problème de routage est loin d'être évident dans cet environnement, où ce dernier impose de nouvelles limitations par rapport aux environnements classiques. Les stratégies de routage doivent tenir compte des changements fréquents de la topologie, de la consommation de la bande passante qui est limitée, ainsi d'autres facteurs.

Finalement, nous avons présenté la classification et les différents types de protocoles de routage dans les réseaux ad hoc, avec quelques exemples pour les protocoles de routage proactifs, réactifs et hybrides qui ont été conçu pour ses réseaux.

Dans le chapitre suivant, nous allons comparer entre les simulateurs les plus utilisés dans le domaine réseau et nous mettons l'accent sur le simulateur adapté à notre thème.

Chapitre III

Simulation et
Simulateurs Réseaux

III.1.Introduction

De nos jours, la simulation connait un essor considérable, et ce grâce à l'intérêt que présente les modèles informatique des systèmes simulés. Elle consiste essentiellement à modéliser un système quelconque, en offrant une représentation de toutes les entités de ce système, leurs comportements propres et ainsi leurs interactions. Et grâce aux progrès réalisés dans le domaine du développement et de techniques de programmation, il devient possible de réaliser un simulateur pour un environnement de programmation donnée [69]**.**

Un simulateur dit réseau est une technique de mise en œuvre du réseau dans l'ordinateur. Grâce à lui, le comportement du réseau est calculé soit par l'interconnexion des entités du réseau en utilisant des formules mathématiques, ou en captant et en lisant des observations à partir d'un réseau de production.

Avant de commencer la réalisation d'un projet, il faut choisir les outils nécessaires pour son implémentation. Ainsi pour la réalisation de notre travail nous avons choisi le simulateur NetLogo, et afin de justifier notre choix nous définissons dans l'ensemble de ce chapitre quelques simulateurs réseaux et leurs différences avec celui choisi.

III.2. Simulation

III.2.1. Définition

Le terme simulation vient du latin « **simulatio »** qui signifie : **apparence** et **imitation.** Si nous nous en tenons, dans notre cadre, au terme d'imitation, la simulation est donc l'imitation d'un phénomène du monde réel. **Mais quel besoin l'homme a-t-il d'imiter les phénomènes du monde réel ?** [70]

La réponse à cette question n'est pas unique, mais, comprendre le passé, les situations présentes ou prédire et mieux planifier les états futurs d'un système sont les objectifs partagés par de nombreux outils d'analyse et de simulation [70]**.**

Définition1: La simulation est la programmation et la manipulation d'un modèle sur un ordinateur, ainsi que l'analyse des résultats, et elle est largement utilisée pour le développement de nouvelles architectures de communication et de protocoles de réseau [71]**.**Nous appellerons temps de simulation le temps de calcul utilisé pour simuler une expérience, l'une des étapes de la simulation est la description d'un système qui consiste à:

- Identifier les éléments (entités).
- Déterminer les attributs.
- Spécifier les activités (processus à l'origine des changements).

Définition 2:"Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output." [72]

Cette deuxième définition donne un aperçu global et intuitif de ce qu'est un processus de simulation par ordinateur.

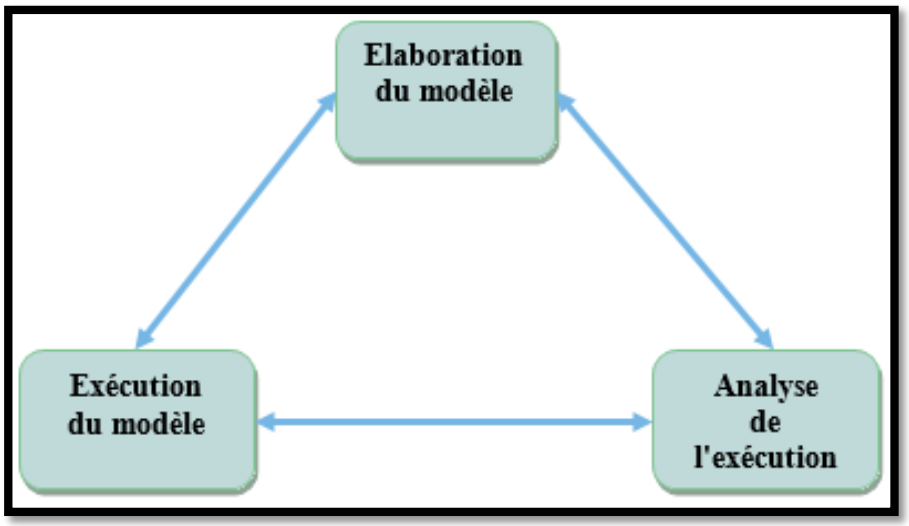


Figure III. 1: Simulation informatique selon MICHEL [72].

Cette figure ci-dessus représente les trois tâches fondamentales d'une simulation.

MICHEL définit ainsi cette discipline comme un processus itératif non linéaire composé de 3 tâches fondamentales, fortement interdépendantes [72] :

1. L'élaboration du modèle.
2. L'exécution du modèle sur ordinateur.
3. L'analyse de l'exécution du modèle et des résultats obtenus.

Encore une fois, cette définition fait clairement apparaitre l'importance du **modèle**, et donc de son élaboration, dans la conception d'une simulation. Par ailleurs, il est intéressant de remarquer que cet énoncé s'attache principalement à définir quelles sont les taches fondamentales qui sont associées à cette démarche scientifique. En effet, au-delà des objectifs sous-tendus par ce type d'approche, il est dans un deuxième temps important de définir les différentes phases qui caractérisent sa mise en œuvre [72].

Remarque :

Il est important de distinguer les deux termes **"modèle"** et **"simulation"** car la différence entre le modèle et la simulation peut être considérable.

Le **modèle** représente la structure conceptuelle du système observé, c'est-à-dire la structure des agents et autres entités et la relation qui lie les composants au niveau de l'organisation. Il décrit également le comportement individuel, c'est-à-dire le mécanisme des changements d'état et la façon dont les activités se succèdent dans le temps.

Alors que, la **simulation** correspond à l'implémentation de ce modèle sur une machine, ce qui permet de l'exécuter. « Un simulateur désigne ainsi tout système de calcul capable d'exécuter le modèle de manière à générer son comportement » [73]**.**

III.2.2. Types

En fonction du type d'évènements dans la simulation, nous distinguons deux types de systèmes de simulation : les systèmes discrets et les systèmes continus.

III.2.2.1. Systèmes de simulation discrets

Il s'agit d'un système dans lequel les variables liées à la simulation ne changent d'état qu'en plusieurs points de l'axe du temps. Ces systèmes sont également appelés : systèmes de simulation à événements discrets.

III.2.2.2. Systèmes de simulation continus

Les variables de ces systèmes peuvent changer d'état à tout moment pendant la simulation.

III.2.3. Concepts liés à la simulation

1. **Evènement** : L'événement est caractérisé par la date à laquelle l'état du système a changé (date de l'événement). Il existe des événements internes et externes (également appelés événements endogènes et exogènes).
2. **Activité** : Dans chaque événement, des objets associés participent à l'opération, ces opérations initiées (ou terminées) sur chaque événement sont appelées activités. Toute activité est limitée à son début et sa fin par évènement.
3. **Processus** : Un processus est le groupement d'une séquence d'événements dans l'ordre chronologique dans lequel ces évènements se produiront. Comme les évènements peuvent initier des activités, un processus peut inclure aussi des activités [74].

III.2.4. Avantages et inconvénients

La simulation repose sur plusieurs avantages et malgré ses intérêts, la simulation possède ainsi des inconvénients et le tableau suivant les expliquent :

Avantages	Inconvénients
Observations des états du Systèmes	La conception de modèles peut nécessiter des compétences spéciales
Etudes des points de fonctionnement d'un système	Une autre forme d'analyse plus proche de la réalité est peut être nécessaire
Etudes de systèmes à échelle de temps variable	Résultats difficilement interprétables
Etudes de l'impact des variables sur les performances du système	Résultats pas forcément généralisable
Etude d'un système sans les contraintes matérielles	Résultats sont en fonction des entrées du système
Analyse l'enchaînement des événements dans le système	
Amélioration d'un système déjà existant, en Testant différentes optimisations	

Tableau III. 1: Avantages et les inconvénients de la simulation.

Ce tableau montre que la simulation présente de nombreux avantages, tels que l'étude du point de fonctionnement du système et la visualisation de l'état du système pour étudier son comportement sous surveillance ou aide à la décision. Le tableau répertorie également certains inconvénients, tels que le besoin de compétences particulières lors de la conception de modèles et Il est difficile d'interpréter les résultats.

III.2.5. Simulation en tant que processus expérimental

La définition que nous avons donnée précédemment suffit la plupart du temps à illustrer clairement et simplement les différentes étapes qui constituent l'élaboration d'une simulation par ordinateur. Cependant, il est peut être intéressant de rentrer dans le détail de ces étapes. Pour cela, on peut encore une fois se référer à **MICHEL** [72] qui propose de distinguer les étapes de conception suivantes dans le **processus** de simulation :

1. **Définition du problème**

   Dans cette phase initiale, il s'agit clairement de définir les objectifs de l'étude. **Quelles sont les questions auxquelles on souhaite apporter une réponse ?**

2. **Planification du projet**

   Il s'agit ici de s'assurer que nous avons les ressources humaines et le matériel nécessaire pour l'étude.

3. Définition du système

À ce stade, l'objectif est de déterminer les aspects du système que vous souhaitez étudier afin de pouvoir le définir de manière pertinente dans l'expérience. Le modèle sera ensuite développé en fonction des objectifs fixés.

4. Formulation du modèle conceptuel

Durant cette phase, le premier modèle sera développé graphiquement ou en pseudo code. Il s'agit de définir les différentes entités qui composent le système : composants, variables, interactions entre composants, etc.

5. Analyse préliminaire de l'expérimentation

Il faut ici déterminer quels sont les critères qui permettront d'évaluer la qualité de l'expérimentation : quels sont les paramètres que l'on souhaite faire varier, avec quelle amplitude et sur combien d'exécutions. Combien d'expériences seront nécessaires à l'expérimentation dans son ensemble.

6. Constitution des paramètres initiaux

A ce stade, il s'agit de déterminer et de collecter les données nécessaires à l'élaboration des valeurs initiales, qui seront utilisées pour le paramétrage du modèle.

7. Transcription du modèle

Cette étape consiste à convertir le modèle élaboré dans un langage de simulation de manière à permettre son implémentation sur machine.

8. Vérification et validation

Il s'agit ici de vérifier d'abord que le simulateur exécute correctement le modèle (debugging) puis de vérifier les résultats obtenus par le modèle. Sont-ils acceptables et représentent-ils le système que nous voulons étudier ?

9. Analyse finale de l'expérimentation

À ce stade de la conception, la cinquième étape doit être reconsidérée. En effet, la connaissance du modèle ayant considérablement augmentée, il est nécessaire de mettre à jour ses conclusions.

10. Expérimentation

La simulation proprement dite est exécutée de manière à récupérer les résultats désirés et à effectuer une analyse de sensibilité du modèle aux paramètres initiaux.

11. Analyse et interprétation des résultats

Une fois les simulations effectuées, il s'agit d'inférer des conclusions sur le modèle à partir des résultats obtenus.

12. Utilisation et documentation

Les conclusions tirées de l'expérimentation, le modèle et son utilisation doivent être clairement documentés**.**

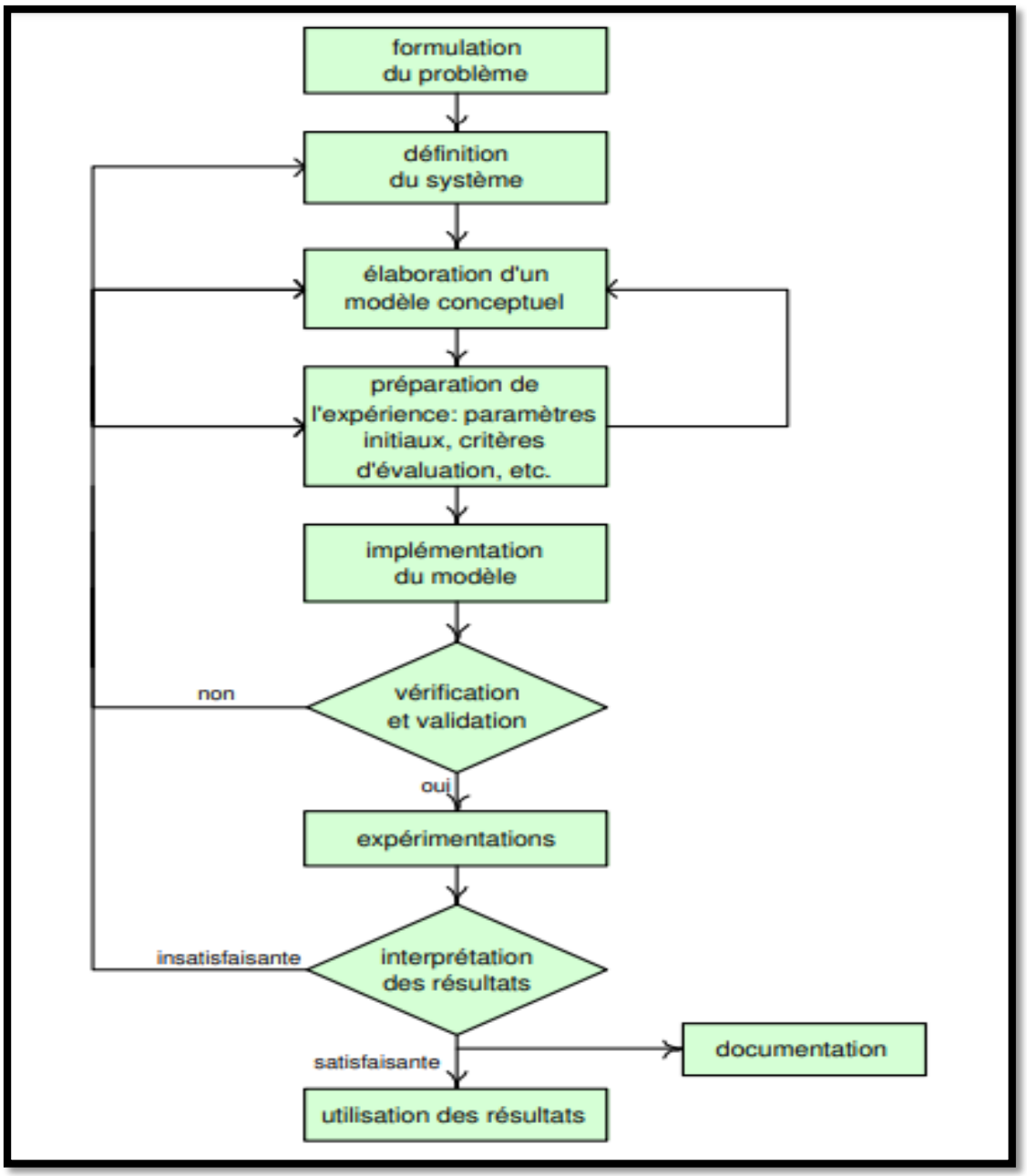


Figure III. 2: Etapes de conception d'une simulation informatique [72].

Cette figure résume de manière schématique les principales étapes de conception liées à cette approche. Selon la vérification, la validation et les résultats obtenus si on n'arrive pas à valider ou bien les résultats ne sont pas satisfaisant, nous pouvons revenir aux étapes précédentes pour faire des changements adéquats.

III.2.6. Simulation à base d'agents

III.2.6.1. Définition

La simulation multi-agents (ou simulation **orientée agent**) utilise les agents pour représenter les entités ou les agrégats du phénomène étudié. Elle permet notamment de se focaliser sur l'émergence de caractéristiques globales à partir d'actions et d'interactions locales entre les agents [70]**.**

Elle consiste à concevoir, construire et modéliser un système complexe constitué d'entités individualisées disposant d'un certain degré d'**autonomie** et interagissant les unes avec les autres. Une telle approche est proposée, aujourd'hui, à la fois pour la résolution de problèmes en **intelligence** artificielle et pour la représentation de processus en économie, écologie, géographie, ou physique, elle est de plus en plus utilisée pour représenter des systèmes dynamiques.

La simulation à base d'agents **(SBA)** a pour but de reproduire la dynamique d'un système en modélisant les entités par des agents, dont le comportement et les interactions ont été définis.

III.2.6.2. Domaines d'application

Les simulations multi-agents sont largement utilisées dans de nombreux domaines comme notamment [70]:

- **Simulations d'écosystèmes et de gestion de ressources naturelles :** dont la problématique est de tenter d'étudier et de résoudre les problèmes liés à l'accès et à l'utilisation des ressources naturelles et renouvelable.
- **Simulations urbaines :** dont le but est l'étude et la compréhension de l'évolution des cités et de l'impact de leur aménagement sur l'environnement et les flux de populations.
- **Simulations en transport :** dont l'objectif est d'aider à la compréhension des flux urbains dans une agglomération.
- **Simulations de mouvement de foules :** qui permettent l'étude de la dynamique des comportements lors de la concentration d'un nombre important de personnes dans un espace restreint.
- **Simulations de phénomènes naturels** : dont le but est d'étudier des phénomènes comme l'infiltration, l'érosion ou de formation de la houle.

III.3. Simulateurs de réseau les plus utilisés

Il existe plusieurs simulateurs de réseau tels que : **NS2**, **OMNET++, OPNET, NS3, GLOMOSIM**, **JSIM** …etc. Parmi ceux, nous allons citer quelques simulateurs les plus utilisés comme **NS2**, **NS3** et **OMNET++.**

III.3.1. Network Simulator-2 (NS2)

NS-2 est un simulateur à **évènements discrets** le plus utilisé et le plus répondu dans le domaine de la recherche liés au réseau, l'utilisation de l'appellation NS-2 précise la version du simulateur NS**.** Il a commencé comme un simulateur de réseau général, et ainsi la prise en charge des réseaux mobiles ad hoc sans fil ont été ajoutée plus tard [75]**.** NS-2 constitue un support important pour la simulation de protocoles TCP, protocoles de routage et protocoles de multicast.

Le développement de NS suit une approche orienté objet **(OO),** qui utilise deux langages de programmation **C++**, **OTcl** (Object Tool Command Langage) comme interpréteur. Le noyau de simulation, les modèles, les protocoles et les autres composants sont implémentés en C++, mais sont également accessibles depuis OTcl. Les scripts OTcl sont utilisés pour la configuration du simulateur, la mise en place de la topologie du réseau, la spécification des scénarios, l'enregistrement des résultats de la simulation, etc [76]**.**

NS-2 possède un outil appelé **NAM** (Network Animator) qui permet de visualiser graphiquement le déroulement de la simulation. Au cours de la simulation, NS-2 génère un fichier spécial de trace Nam. Le script OTcl est utilisé pour sélectionner les informations qui doivent être enregistrées dans ce fichier [77]**.**

La figure suivante illustre les différents composants internes du simulateur :

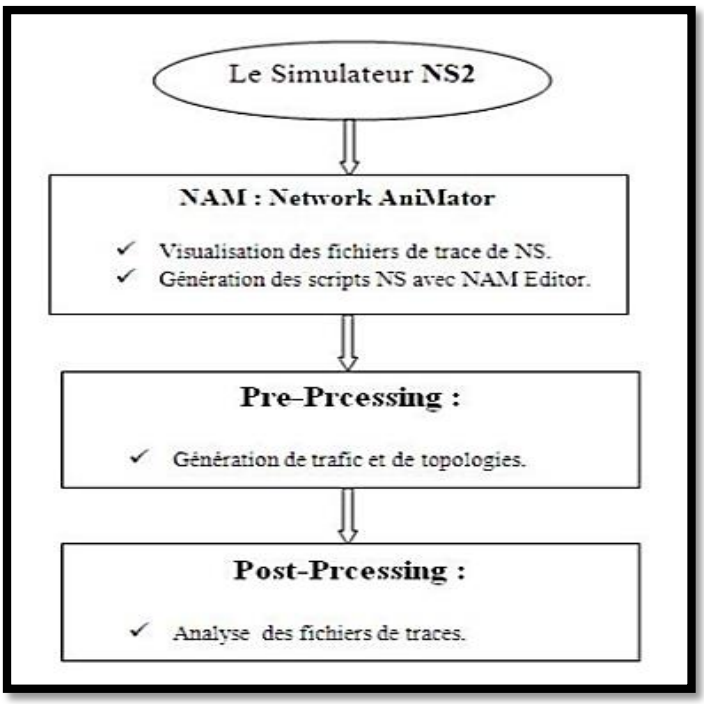


Figure III. 3: Description architecturale du simulateur NS2 [69].

L'utilisation de **NS-2** comme simulateur de réseau de capteur sans fil (**WSN** Wireless Sensor Network) présente quelques inconvénients [77]**:**

- Le modèle de détection n'existe pas.
- Les formats de paquets et les protocoles MAC (Medium Access Control) sont différents de ceux que l'on trouve sur les plateformes WSN classiques.
- Le modèle énergétique est trop simple.

III.3.2. Network Simulateur-3 (NS3)

NS-3 est un simulateur de réseau à **événements discrets** open source[4]. Il est considéré comme un remplaçant de NS-2, et non comme une extension. Il s'appuie sur le **C++** pour la mise en œuvre des modèles de simulation [78]**.**

Comme NS-2, il n'a pas d'API (Application Programming Interface) OTcl, NS-3 n'utilise plus les scripts OTcl pour contrôler la simulation, abandonnant ainsi les problèmes qui ont été introduits par la combinaison de C++ et OTcl dans NS-2.

La version **NS-3.10** supporte la simulation parallèle et possède un ensemble de fonctionnalités avancées. De plus, les simulations de réseaux NS-3 peuvent être implémentées en **C++** pur, tandis que certaines parties de la simulation peuvent aussi être écrites en **Python** [78]**.**

[4]C'est-à-dire les possibilités de libre redistribution, d'accès au code source et de création de travaux dérivés

La figure III.4 illustre l'interface du simulateur NS-3.

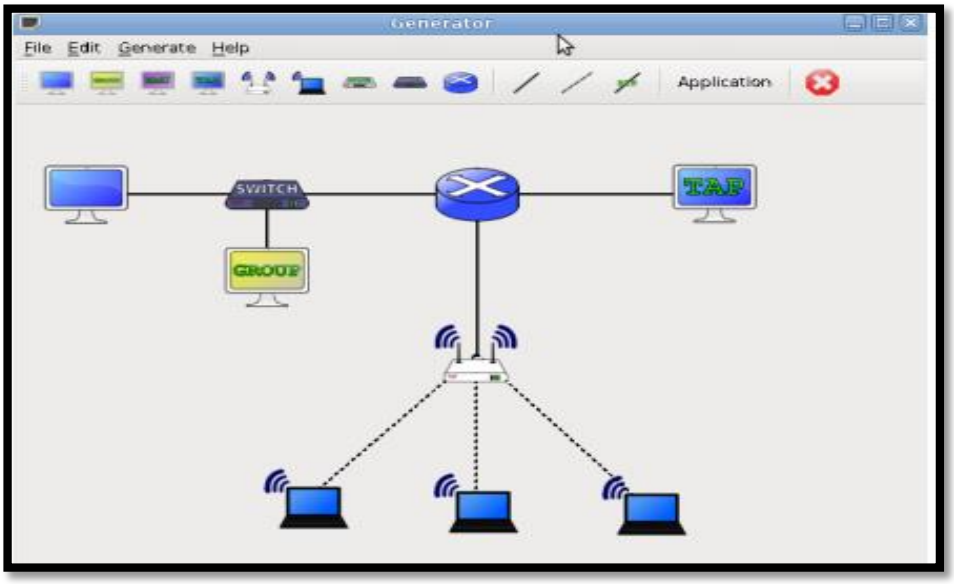


Figure III. 4: Interface du simulateur NS-3 [78].

NS-3 supporte à la fois la simulation et l'émulation[5] des sockets utilisateurs, Il génère également des traces pcap[6] qui peuvent aider au débogage. Pour analyser le trafic réseau, des outils standards comme **Wireshark** peuvent être utilisés pour lire les fichiers de trace. Il fournit ainsi un environnement réaliste et son code source est bien organisé [78]**.**

III.3.3. OMNET++ (Objective Modular Network Testbed in C++)

OMNeT++ est un simulateur **d'événements discrets** open source écrit en C++ [76]**.**Il ne s'agit pas d'un simulateur réseau de capteur sans fil **(WSN),** ni même d'un simulateur de réseau, mais d'une plateforme de simulation riche sur laquelle divers groupes indépendants peuvent construire leurs propres simulateurs, OMNET++ est un cadre général de simulation basé sur des composants (modulaires) et des architectures ouvertes [78]**.**

C'est un environnement qui fournit des outils pour la création et la configuration des modèles de réseaux**(**le fichier **NED**[7] **(**Network Description**)** et le fichier de configuration **INI** (initialisation)**)** et des outils pour l'exécution d'un lot de programmes ainsi que pour l'analyse des résultats de simulation [79]**.**OMNeT++ semble séduire de plus en plus la communauté scientifique et un nombre croissant de modèles sont disponibles.

[5]Consiste à substituer à un élément de matériel informatique tel qu'un terminal informatique, un ordinateur ou un logiciel.
[6]C'est une interface de programmation permettant de capturer un trafic réseau.
[7] Il permet à l'utilisateur de déclarer des modules simples, de les connecter et de les assembler en modules composés.

Figure III. 5: Lancement du simulateur OMNeT++ [69].

III.3.4.NetLogo

NetLogo a été créé par Uri Wilensky au Center for Connected Learning and Computer-Based Modeling (CCL) en 1999, puis à l'Université Tufts dans la région de Boston. NetLogo est né de StarLogoT, qui a été écrit par Wilensky en 1997 [80] .La partie «Logo» est due au fait que NetLogo est un dialecte du langage Logo[8], la partie «Net» est destiné à évoquer la nature décentralisée et interconnectée des phénomènes qui peuvent être modélisé avec NetLogo, y compris les phénomènes du réseau.

NetLogo est un outil de simulation multiplateforme et fournit une API permettant de piloter son simulateur ce qui simplifie son intégration. Un modèle NetLogo est composé d'une interface graphique qui permet à l'utilisateur d'interagir avec la simulation et de voir son évolution, et d'un script (en langage NetLogo) qui décrit le comportement des agents (appelés turtles), la dynamique de l'environnement, et plus généralement quelques actions qui seront réalisées à chaque pas de simulation [80].

[8] C'est un langage de programmation orientée objet réflexif. Plus lisible que le Lisp, il en est une adaptation, ce qui lui a valu le surnom de « Lisp sans parenthèses ».

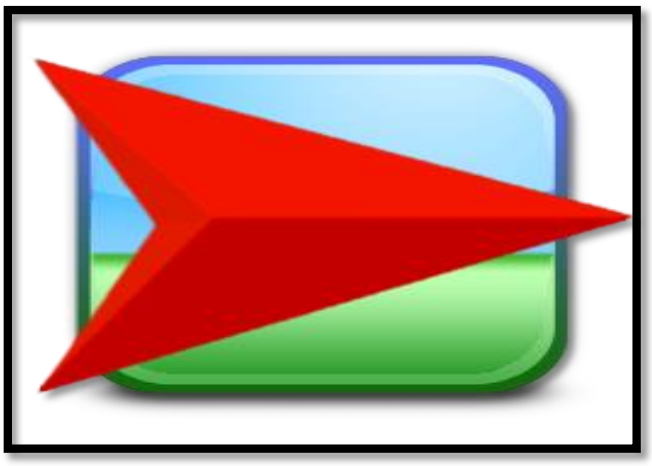


Figure III. 6: Logo du simulateur NetLogo v 6.1.1.

C'est un simulateur **multi-agents** qui est utilisé pour développer des simulations dans les domaines qui comprennent : Art, Biologie, Chimie et Physique, Informatique, Mathématiques, et les sciences sociales. Notamment, il est non adapté **aux réseaux** mais n'empêche pas que son utilisation dans la simulation des réseaux informatiques est toujours en évolution, ainsi que plusieurs travaux de recherche sont développés dont nous pouvons cités :

- ✓ **WSimulNET, Un modèle pour la simulation de grands réseaux sans fil** : Les réseaux à grande échelle sont difficiles à simuler dans les simulateurs de réseaux traditionnels, car ils peuvent nécessiter des ressources de calcul élevées. De plus, en raison de leur complexité dans les détails, les simulateurs traditionnels n'offrent pas un environnement convivial pour les nouveaux étudiants. WSimulNET, c'est un modèle conçu pour simuler les réseaux hertziens à grande échelle. Il est un moyen simple et rapide d'introduire les principaux concepts de réseaux sans fil aux étudiants (sous-) diplômés puisqu'il tire parti de la faible complexité de l'environnement de NetLogo.

  Ce modèle implémente quelques protocoles des couches liaison de données (MAC) et réseau. L'objectif principal de ce modèle est de fournir un moyen de simuler des réseaux à grande échelle avec un temps de mise en place court [81].

- ✓ **NetLogo - Une autre façon de simuler les réseaux mobiles ad hoc** : Les caractéristiques du réseau mobile ad hoc obligent à utiliser la simulation pour vérifier les nouveaux concepts proposés dans les réseaux mobiles ad hoc. Les simulateurs de réseaux traditionnels sont orientés vers la simulation de bas niveau, ce qui entraîne diverses difficultés lorsque ces simulateurs sont utilisés pour la simulation des aspects de haut niveau des réseaux mobiles ad hoc.

Cette optique propose l'utilisation du langage de programmation multi-agents NetLogo pour simuler des aspects de haut niveau des réseaux mobiles ad hoc. L'applicabilité de cet outil est démontrée en simulant et en évaluant les critères de sécurité de diverses approches d'infrastructure à clé publique dans un réseau mobile ad hoc dans NetLogo [82]**.**

- ✓ **Simulation d'une approche de modélisation routière dans l'environnement VANET à l'aide du NetLogo :** La nouvelle version modifiée du réseau mobile ad hoc appelé "Vehicular Ad hoc Network" (VANET), vise à employer et à utiliser des technologies ad hoc pour réduire la congestion routière en temps réel. Dans l'environnement de VANET, le mot "véhicule" représente un nœud intelligent capable de communiquer avec les voisins mobiles du réseau. VANET présente plus de défis que le "réseau mobile ad hoc" (MANET) en raison de la grande mobilité des nœuds et des changements rapides de topologie dans VANET. Dans cette étude, une approche développée est suggérée en fonction des fondamentaux de la table.

Le simulateur NetLogo est suggéré pour être utilisé dans la programmation, la conception, la création et la mise en œuvre de plusieurs structures pour différentes cartes routières. Il est proposé de mettre en œuvre et de tester trois nouveaux formulaires de modèles routiers avec le système VANET. La simulation sera utilisée pour observer et mesurer leurs comportements en générant et en collectant les données, les facteurs et les paramètres requis. Les résultats finaux ont montré que le nombre de véhicules, la vitesse des véhicules, l'intensité du trafic et la zone de couverture ayant un effet tangible sur les performances des réseaux [83].

III.4. Etude comparative

III.4.1. Tableau de comparaison

Après avoir brièvement décrit les trois simulateurs réseaux (NS2, NS3 et OMNET ++) nous allons faire une étude comparative entres ces simulateurs en termes d'avantages et limites de chacun. Les résultats de cette étude sont présentés dans ce tableau :

Simulateur	Avantages	Limites
NS2	1- Fournit un modèle de circulation et de mouvement facile en incluant un modèle d'efficacité énergétique. 2- Fournit un ensemble de modèles de mobilité randomisée. 3- Scénarios complexes peuvent être facilement testés.	1- Recompilation à chaque fois, s'il y a un changement dans le code d'utilisateur. 2- Système réel est trop complexe pour modéliser à savoir l'infrastructure complexe. 3- Le mélange de la compilation et l'interprétation et il est difficile d'analyser et de comprendre le code.
NS3	1-Haute modularité que son ancêtre NS2. 2- Prise en charge de TCP, UDP, ICMP, IPv4, routage multicast, P2P et les protocoles CSMA. 3- Une large gamme d'utilisation à la fois l'optimisation et l'expansion des réseaux existants.	1- Souffre encore d'un manque de crédibilité. 2- Besoin de beaucoup de soutiens spécialisés afin de profiter du bien-fondé de NS2. 3- Les mainteneurs actifs sont nécessaires pour répondre aux questions des utilisateurs et des rapports de bugs.
OMNET++	1- Fournit un environnement graphique puissant 2- Le débogage est beaucoup plus facile que d'autres simulateurs. 3- Précision des modèles les plus matériels et comprennent la modélisation des phénomènes physiques. 4- Supporte le parallélisme.	1- Il ne propose pas une grande variété du modèle, des ressources, des algorithmes, etc [84]. 2- L'extension de la mobilité relativement incomplète. 3- Pauvres en termes d'analyse et de gestion de performances.

Tableau III. 2: Avantages et limites des simulateurs [84].

Dans les sections suivantes nous détaillerons particulièrement les simulateurs NS2 et OMNET++ et NetLogo, en ignorant le simulateur NS3 qui n'est pas un produit officiellement sponsorisé par une entreprise. Le seul support de NS3 repose sur les efforts déployés sur les listes de diffusion des groupes d'utilisateurs de NS3.

III.4.2. Comparaison entre les simulateurs NS-2 et OMNET++

Dans cette section, nous récapitulerons les points forts et les points faibles des simulateurs NS-2 et OMNET ++ étant les simulateurs réseaux les plus connues. NS-2 est un outil puissant pour la simulation du protocole TCP, du protocole de routage et du protocole de diffusion. Alors qu'OMNET ++ est une bibliothèque de simulation C ++ extensible, basée sur des modules et basée sur des composants, principalement utilisée pour construire des simulateurs du réseau.

III.4.2.1. Selon les caractéristiques

OMNeT++, a une architecture modulaire et extensible basée sur des composants, pour la simulation des réseaux ad hoc sans fil. Il s'appuie sur deux extensions externes et importantes sont : le cadre INET[9](IF) et le cadre de mobilité (MF). Il est installé à l'aide des scripts de **configure**[10] et **make**[11], avant de l'installer les deux packages tcl/tk et BTL doivent être installés, et le temps nécessaire pour l'installer et de l'importer sur éclipse d'environs de trois jours .C'est le seul simulateur avec visualisation en ligne, un utilisateur de ce simulateur peut inspecter ou même modifier directement les valeurs dans les modèles. L'une des principaux inconvénients de l'OMNeT++ est qu'il ne dispose pas d'un modèle énergétique [76].

NS2 dispose d'un modèle énergétique. En ce qui concerne son installation, il existe deux façons soit en téléchargeant les package tout-en-un ou seulement certains composants et bibliothèques. Et l'un des principaux avantages de NS2 est l'énorme quantité de fonctionnalités disponible, offrant un grand nombre de protocoles externes déjà mis en œuvre. Afin de visualiser le comportement du réseau dans NS2, il faut d'abord appeler deux scripts : un pour générer un fichier de trace de trafic et un autre pour créer un fichier de trace de mouvement. La simulation génère ensuite un fichier journal qui peut être ensuite visualisé à l'aide d'animateur (Nam) de NS2 qui est similaire à celui de OMNeT ++ [76].

[9]INET Framework est une bibliothèque de modèles open-source pour l'environnement de simulation OMNeT ++.

[10]Le script configure détecte les logiciels installés et la configuration de système. Il écrit les résultats dans le fichier Makefile.inc, qui sera lu par les makefiles pendant le processus.

[11]Le script make est utilisé pour le débogage et la construction des versions optimisée, c'est créer l'environnement OMNET.

III.4.2.2. Selon deux protocoles 802.11b, 802.15.4

Dans notre travail nous nous intéressons à la simulation des réseaux, il est utile donc d'exposer la différence dans les technologies 802.11 et 802.15 :

IEEE 802.11b : cette norme vient améliorer les vitesses de transmission insuffisantes du 802.11, permet une transmission point à multipoint en ligne de vue sur une distance allant jusqu'à 300 mètres avec des débits dépendant de la qualité du signal de 1, 2, 5.5 ou 11 Mbit/s [85].

Protocole 802.15.4 : est un protocole de communication défini par l'IEEE. Il est destiné aux réseaux sans fil de la famille des LR WPAN (Low Rate Wireless Personal Area Network) du fait de leur faible consommation, de leur faible portée et du faible débit des dispositifs utilisant ce protocole [86].

Pour ces deux protocoles (**802.11b, 802.15.4**), lorsque le même scénario est implémenté, le temps passé à envoyer et à recevoir les trames de données dans OMNeT++ est toujours supérieur à celui de NS2. Par conséquent, lorsque le même scénario est mis en œuvre, l'énergie consommée dans OMNeT++ est supérieure à l'énergie consommée dans NS2. Les résultats des scénarios de consommation d'énergie indiquent que OMNeT++ a des résultats cohérents sur les deux plateformes utilisées dans l'évaluation, c'est-à-dire Windows et Linux. Il s'agit d'une caractéristique importante pour les simulateurs qui fonctionnent sur plusieurs plates-formes. Le NS2 est pris en charge surtout sur les plates-formes Linux. OMNeT++ possède une interface graphique intégrée. Elle permet à l'utilisateur d'exécuter graphiquement les scénarios et de déboguer facilement le code source. Il possède également un module de développement intégré [75].

L'environnement de développement (IDE) aide le développeur à identifier les erreurs et à vérifier la syntaxe avant de compiler le code, ce qui cause un grand problème lors de l'utilisation du simulateur NS2.

L'évaluation des critères qualitatifs des deux simulateurs du réseau sans fil cités sont résumés dans le tableau au-dessous.

Critère	**NS2**	**OMNeT++**
Nature de simulateur	Simulateur	Simulateur
Type de simulateur	Événement discret	Événement discret
Licence	Licence GNU GPLv2.	Licence publique académique. Les modèles INET sont sous licence LGPL ou GPL.

Interface utilisateur	GUI : Oui, via Nam. Langage de programmation supporté : C++ et OTcl.	GUI : Oui, une interface graphique intégrée est disponible. Langage de programmation pris en charge : C++ et NED.
Plateformes supportées	Linux, MacOs et FreeBSD.	Windows, Linux et Mac OSX.
Hétérogénéité	Non	Oui
Niveau de détail	Niveau de code	Niveau de paquet
Modèles de mobilité	Oui	Oui
Modélisation	Disponible	Disponible
Modèle d'énergie	Modèle de batterie : Uniquement pour la batterie idéale. États RF [12]: Oui Limitations : Impossible de modéliser les unités de détection et de traitement.	Modèle de batterie : Oui États RF : Oui Limitations : Impossible de modéliser les unités de détection et de traitement.
Modèle du support sans fil	Modèles de perte de chemin : ombres, sol à deux rayons, espace libre.	Modèles de perte de chemin : espace libre, ombrage log-normal, évanouissement de la lumière du soleil, sol à deux rayons, ... Autres modèles : Bruit de fond, modèles de perte d'obstacle et de propagation.

[12] États RF : tels que l'inactivité, le sommeil, la réception et la transmission.

Technologies Et protocoles pris en charge	Couche Applicat-ion	DHCP, Telnet, FTP, HTTP.	HTTP, DHCP, BitTorrent, Streaming vidéo P2P, de la voix
	Couche transport	TCP, UDP, SCTP, XCP, TFRC, RAP, RTPM.	TCP, UDP, SCTP, RTP, RTCP.
	Couche Réseau	IPv4, IPv6.	ARP, HIP, IGMPv2, IGMPv3, IPv4, IPv6, MCoA, MIPv6.
	Couche liaison de données	HDLC, GAF, MPLS, LDP, Diffserv, DropTail, RED, RIO, WFQ, SRR, SemanticPacket Queue, REM, CSMA, 802.11b, 802.15.4, Satellite Aloha.	802.11, 802.11p, 802.1e, WiMAX, LDP, LTE, PPP.
	Protocoles de Routage	RIP, AODV, Click, DSDV,DSR, NixVectorRouting, OLSR	AODV, BGP, GPSR, routage à l'état de lien, OSPF, OSPFv2, PIM, RIP.

Tableau III. 3: Comparaison entre NS2 et OMNET++ [75].

- **Problèmes de ces simulateurs**

Il existe plusieurs simulateurs pour les réseaux sans fil, qui proposent différents modèles, et fonctionnalités. Le choix du simulateur doit être déterminé par les exigences des modèles a simulés. Les simulateurs peuvent gérer complètement le réseau, il est donc très pratique de les utiliser surtout parce qu'ils facilitent leur suivi. De plus, pour sélectionner un simulateur est une question délicate, la réponse dépend en grande partie des besoins d'utilisation.

Pour **NS2** la tâche de modélisation reste complexe : il n'y a pas d'interface graphique et il faut une forte technicité requise pour utiliser ce simulateur. Le problème majeur de ce simulateur est la difficulté d'installation car il nécessite d'abord une bonne maitrise de la plateforme linux, l'installation des packages… etc. A tous ses problèmes s'ajoutent que dans quelque cas il nécessite une analyse plus proche de la réalité, les résultats sont difficilement interprétables, pas forcément généralisable et surtout ils sont en fonction des entrées du système.

Pour **OMNET++**il y'a peu de modèles pour les réseaux sans fil, Il ne propose pas une grande variété des ressources, des algorithmes, …etc. L'extension de la mobilité est relativement incomplète, il a une pauvre analyse et gestion de performances, description des modèles en langage NED. En plus de tout ce qu'était énumérer s'ajoute la difficulté d'installation.

Suivant notre cadre du travail et à la fin de cette étape de comparaison entre OMNET++ et NS-2, nous nous intéressons au simulateur NS-2 qui est déjà plus utilisé qu'OMNET++ dans les réseaux sans fil et contrairement à ce dernier, NS-2 offre des modèles pour ces réseaux. Et Omnet++ nécessite plus d'apprentissage que ns2.

III.4.3. Comparaison entre NS-2 et le simulateur NetLogo

Les réseaux à grande échelle sont difficiles à simuler dans les simulateurs de réseaux traditionnels (OMNeT++, NS3… etc), car ils peuvent nécessiter des ressources de calcul élevées. De plus, en raison de leur complexité dans les détails, les simulateurs traditionnels n'offrent pas un environnement convivial pour les nouveaux étudiants.

NetLogo est un environnement multi-agents qui permet de simuler des phénomènes naturels et sociaux, utilisé dans la modélisation des systèmes complexes. Les modèles créés dans NetLogo permettent la création de règles de comportement pour des centaines d'agents qui fonctionnent indépendamment ou en fonction des interactions avec leur environnement. Cela permet d'explorer la relation entre les comportements individuels des agents et d'observer les modèles de comportement émergent découlant des interactions individuelles.

NS-2 est l'un des simulateurs sans fil le plus répondu accepté par les chercheurs et les développeurs. De plus c'est un simulateur WSN (Réseau de capteur sans fil) génériques et devrait être utilisé pour l'évaluation d'algorithmes, de protocoles et d'applications de haut niveau avant de passer à une plateforme spécifique. Cependant, il s'appuie sur des modules d'extension tiers pour fournir des caractéristiques spécifiques au WSN telles que les protocoles MAC et les modèles de détection.

- **Similarités**

NetLogo et **NS2** sont des outils de simulation libre. Ils s'agissent des logiciels gratuits, tout le monde peut les télécharger gratuitement et de construire des modèles sans restriction.

NetLogo est accompagné d'une documentation complète et des tutoriels et une grande collection d'échantillons de modèles, ainsi que NS2.

- **Différences**

-Ils fonctionnent sous différents plateformes notamment Linux (Unix) pourNS2, Windows et les systèmes Mac pour NetLogo.

-NetLogo est un produit stable et rapide. NS2 est un produit non rapide en termes d'installation.

-NetLogo est une application autonome écrite en Java, NS-2 est écrit en C++.

-Interface utilisateur de NS-2 GUI est assurée via Nam. Langage de programmation supporté : C++ et OTcl, tandis que GUI de NetLogo est une interface graphique d'interaction avec le modèle et de représentation des résultats et des états du « monde » adaptée et supporte le langage Logo.

Dans ce tableau nous citrons les plus grands écarts entre **NS-2** et **NetLogo**.

Simulateur	**Network Simulator 2 – NS2**	**NetLogo**
Type	Logiciel de simulation multicouche.	Logiciel de simulation multicouche.
Interface	Interface de programmation en OTcl (Tool Command Langage) et noyau écrit en C++.	Interface de programmation en Logo, et noyau écrit en Java
Développement	Développement orienté objet.	Développement multi-agents.
Adaptative	Adapté aux petits réseaux.	Adapté pour la modélisation de systèmes complexes évoluant dans le temps.
Support de traçage	Pas de traçage en temps réel.	Génère des traces.
Modèles de mobilité	Oui	Oui
Modélisation	Disponible	Disponible

Modèle d'énergie	Modèle de batterie : Uniquement pour la batterie idéale. états RF : Oui Limitations : Impossible de modéliser les unités de détection et de traitement.		Modèle de batterie : la présence d'un dispositif de stockage est indispensable pour assurer l'alimentation des récepteurs d'une façon permanant.
Technologies et protocoles pris en charge	Couche Application	DHCP, Telnet, FTP, HTTP	
	Couche transport	TCP, UDP, SCTP, XCP, TFRC, RAP, RTPM.	
	Couche Réseau	IPv4, IPv6	
	Couche liaison de données	HDLC, GAF, MPLS, LDP, Diffserv, DropTail, RED, RIO, WFQ, SRR, SemanticPacket Queue, REM, CSMA, 802.11b, 802.15.4, Satellite Aloha.	
	Protocoles de Routage	RIP, AODV, Click, DSDV, DSR, and OLSR NixVectorRouting.	

Tableau III. 4: Comparaison entre NS-2 et NetLogo.

Malgré que NetLogo ne possède pas l'essentiel des couches et des protocoles, mais il existe des travaux dans la littérature qui sont en cours de développement.

III.4.4. Discussion

Nous pouvons dire que **NetLogo** est plus simple vu que son monde est composé d'agents qui peuvent exécutés des instructions, (il y a quatre types d'agents dans le NetLogo : les tortues (qui peuvent être mobiles), les patchs, les liens et l'observateur), voir que l'application **NS2** est composée de deux éléments fonctionnels : un interpréteur et un moteur de simulation, qui le classe parmi les simulateurs difficile à travailler dessus. Ce qui rend dans notre contexte le simulateur NetLogo plus facile a utilisé grâce à la non-nécessité de connaissance préalable du son langage de programmation.

De plus par son comportement multi-agents, NetLogo permet d'obtenir des résultats plus proches de la réalité que les outils de simulation utilisés jusqu'à présent. La courte courbe d'apprentissage de l'environnement de NetLogo et son langage font que cet outil peut être utilisé dans des cours d'introduction ou de perfectionnement et avec cet outil on peut simuler plusieurs types de réseaux sans fil.

Conclusion

Dans ce chapitre nous avons vu que la simulation est une méthode qui consiste à créer un univers artificiel à partir de théories, de lois ou d'hypothèses et à observer le comportement d'un système sous quelques aspects lorsque le modèle est soumis à des variations, notamment temporelles [87]**.** La simulation est une expérimentation sur un modèle, ce dernier étant une reproduction du système ou du phénomène que l'on désire étudier. Nous avons vu aussi Les simulations multi-agents qui utilisent les agents pour représenter les entités ou les agrégats du phénomène étudié. Elles permettent notamment de se focaliser sur l'émergence de caractéristiques globales à partir d'actions et d'interactions locales entre les agents.

Par la suite, nous avons présenté les différents simulateurs existants et les plus utilisés dans le domaine du réseau tels que : NS2, OMNET++, NS3 et **NetLogo.** Finalement nous avons défini les principales différences entre ces simulateurs NS-2, OMNET++ et NetLogo, en se basant sur des critères de comparaison possible. Et parmi ces simulateurs, notre choix a été fixé sur NetLogo essentiellement à cause de sa rapidité d'apprentissage et son environnement graphique simple.

Dans le chapitre suivant, nous allons présenter la simulation d'un protocole de routage ad hoc AODV dans le simulateur à base d'agents NetLogo.

Chapitre IV

Simulation du protocole de routage AODV

IV.1. Introduction

Depuis toujours la nature humaine rêve de connaitre le futur et dans le cas des chercheurs du réseau de connaitre les différents comportements d'un protocole avant même de l'implémenter et de le commercialiser, la simulation offre une vue presque réelle. Elle est essentiellement une méthode pédagogique d'intérêt, ayant déjà fait ses preuves dans différents domaines, très largement acceptée et souhaitée, tant par les étudiants que par les enseignants [88].

La nécessité de passer par la simulation a été déjà démontré précédemment. Notre étude sur des simulateurs nous a conduits à choisir un simulateur en particulier « **NetLogo** » pour divers avantages qu'il présente.

Dans le cadre de notre travail, nous avons opté pour la simulation d'un protocole de routage le plus connu dans la famille des protocoles ad hoc réactifs : Ad hoc On demand Distance Vector (**AODV**). Par son aptitude de découverte et de maintenance de route selon les besoins, ce protocole illustre parfaitement le concept de routage à la demande.

Dans ce chapitre nous présenterons l'essentiel de notre travail qui consiste en : une analyse du problème et la solution proposée, puis la conception de cette solution et enfin sa mise en œuvre.

IV.2. Analyse du problème & solution proposée

Après une étude des simulateurs réseaux les plus connus dans ce domaine, nous avons remarqué que l'apprentissage et la maitrise de ces simulateurs est une tâche difficile qui prend une durée considérable de temps, en raison des similarités entre les réseaux et les SMA, ainsi que la facilité du langage de NetLogo nous allons opter pour ce simulateur.

Notre problème principal est d'adapter le simulateur multi-agents (NetLogo) à un simulateur réseau, étant donné qu'il représente une plateforme de simulation utilisée dans plusieurs domaines. Ce dernier a rencontré un grand accueil par les chercheurs du réseau grâce à sa facilité d'installation, de compréhension, d'apprentissage, de manipulation et d'autres avantages qui peuvent être reconnus par le programmeur lors de son utilisation. Mais reste toujours qu'il est confronté à de nombreuses limitations lorsqu'il est utilisé pour la simulation des réseaux en raison du manque de modules et d'implémentations de protocoles du réseau. En vue d'atteindre notre objectif nous avons commencé par la simulation d'un protocole de routage réactif le plus populaire des réseaux Ad hoc AODV.

IV.2.1. Choix d'AODV

Le choix du protocole AODV est justifié par :

- Il est le plus populaire des protocoles de routage réactif pour réseau Ad hoc.
- Il permet aux nœuds mobiles d'obtenir des routes à la demande et n'exige pas le maintien de routes vers des destinations, ce qui réduit sensiblement la taille des tables de routage au niveau de chaque nœud ainsi que le trafic de contrôle qui s'y rapporte.
- Il minimise sensiblement le nombre de diffusions de messages.
- Il est adapté aux réseaux à topologie fortement dynamique.
- Il est peu gourmand en énergie et ne nécessite pas de grande puissance de calcul, il est donc facile à installer sur de petits équipements mobiles.

IV.2.2. Objectif de simulation d'AODV

Nous pouvons résumer notre objectif dans les lignes suivantes :

- Comprendre convenablement le principe d'AODV.
- Avoir une vue proche et presque réel du comportement d'AODV.
- Examiner les messages de contrôle d'AODV.
- Détecter les manques du simulateur NetLogo en comparant aux autres simulateurs réseaux où ce protocole est implémenté (NS2…etc.)
- Voir le fonctionnement et les différentes primitives de NetLogo.

IV.3. Conception

IV.3.1. Type du système utilisé

Dans notre cas, nous avons utilisé une architecture réactive décentralisée, cette architecture est utilisée pour éviter les inconvénients des architectures cognitives et centralisées qui sont :

- La complexité des agents cognitifs à cause de leur capacité de raisonnement.
- La vitesse de réponse dépend de la dimension du système (c'est-à-dire lorsque le nombre d'agent augmente la vitesse des communications décroit).
- Le système est peu robuste, car il est sensible aux fautes du décideur central.
- Il faut que l'agent central dispose des informations globales à chaque instant, ce qui n'est pas toujours réaliste.

IV.3.2. Description de la tâche

On dispose d'un ensemble d'agents-nœuds répartis sur un espace bidimensionnel, au moment où un nœud source demande une route, il crée dynamiquement des routes et les maintient tant que la source a encore besoin.

Chaque agent-nœud possède les propriétés suivantes :

- Une forme,
- Une position,
- Une direction (qui sera utilisée s'il y a une mobilité),
- Une couleur,
- Une taille,
- Un label.

Les agents-nœuds peuvent être dans l'un des états suivants :

- Communication :

  La communication est réalisée via le protocole AODV
  - De transmission de paquets de données.
  - De réception de paquets de données.
- Mobilité :
  - De déplacement

IV.3.3. Protocole de routage utilisé (AODV)

Dans notre travail le réseau Ad hoc est formé par des agents-nœuds mobiles qui communiquent les uns avec les autres via des connexions sans fil. Il n'y a pas d'infrastructure, tous les agents-nœuds sont équivalents et il n'y a pas de centralisation de contrôle, tous les agents-nœuds peuvent servir les autres, et les paquets de données sont transmis de l'un à l'autre dans un mode multi-saut.

Le protocole AODV utilise quatre types de message ou de paquets de contrôle, voir chapitre II (AODV) :

1. **RREQ** Route Request : message de demande de route.
2. **RREP** Route Reply : message de réponse à un RREQ.
3. **Hello** message : message diffusé périodiquement vers le nœud immédiatement voisin pour voir s'il est encore là, s'il n'y a pas de message Hello qui arrive d'un nœud particulier, le voisin suppose que ce nœud est déplacé et marque ce lien comme interrompu.
4. **RERR** Route Error : message qui signale la perte d'une route.

IV.3.4. Modélisation à base de système multi-agents

Le SMA est de plus en plus utilisé pour étudier la complexité des phénomènes naturels ou sociaux. Il y a de nombreuses plate-forme de simulation multi-agents comme : CORMAS, NETLOGO, REPAST, ASCAPE, MADKIT, SWARM, etc.

Comme nous avons mentionné auparavant, les SMA sont des systèmes distribués composés d'un ensemble d'agents interagissant, le plus souvent, selon des modes de coopération, de concurrence.

C'est le modèle que nous avons choisi pour la spécification de notre problème, les agents représentent les agents-nœuds et les interactions entre les agents représentent la communication entre les agents-nœuds.

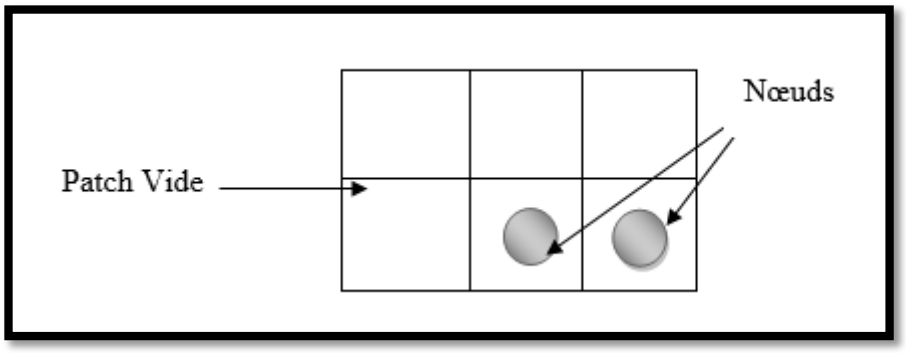


Figure IV. 1: Environnement d'un nœud.

IV.3.5. Modélisation d'un réseau des nœuds à base de graphe

Un réseau d'agents peut être modélisé par un graphe $G_t = (V_t, E_t)$. Où :

V_t représente l'ensemble des nœuds du réseau et E_t modélise l'ensemble des connexions qui existent entre ces nœuds. Si e = (u, v) appartient à E_t, cela veut dire que les nœuds u et v sont en mesure de communiquer directement à l'instant t. La figure IV.2 représente un réseau de 8 nœuds sous forme d'un graphe.

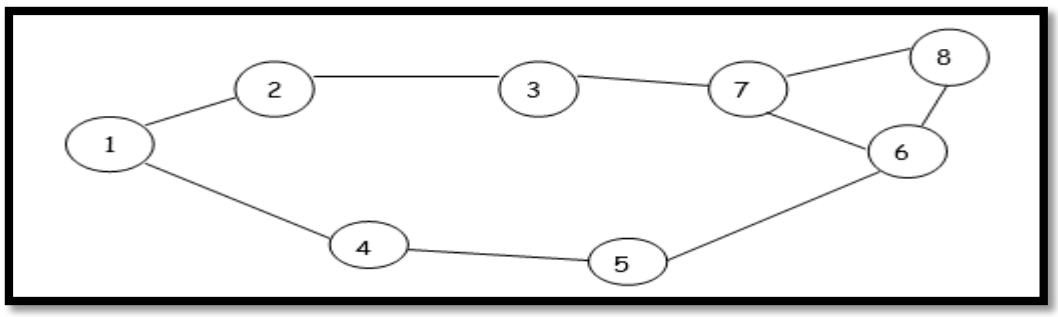


Figure IV. 2: Modélisation d'un réseau de nœud.

Dans la figure ci-dessus les nœuds sont représentés par des cercles et liens de communication par des traits.

La topologie du réseau peut changer à tout moment (voir la figure IV.3), elle est donc dynamique et imprévisible ce qui fait que la déconnexion des nœuds soit très fréquente. Après le déplacement des nœuds, la topologie du réseau de la figure IV.2 peut devenir à un moment donné comme suit :

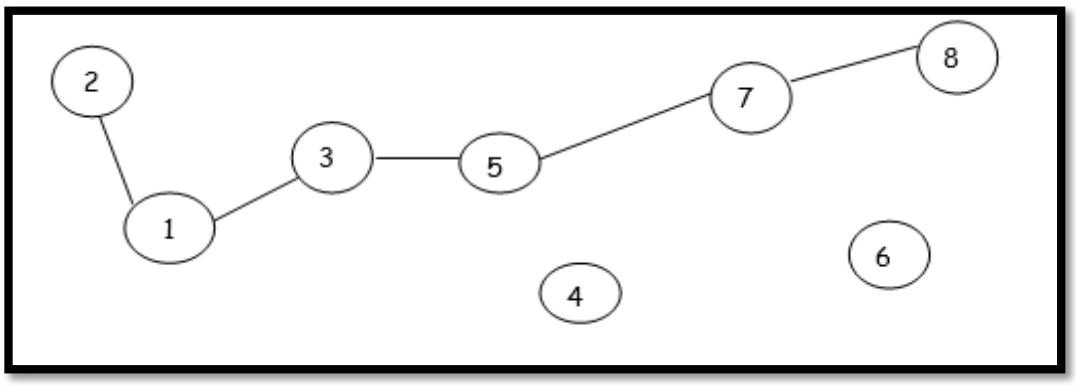


Figure IV. 3: Changement de la topologie des réseaux Ad hoc.

Dans ce cas, l'ancienne route qui existait entre le nœud 6 et le nœud 8 est perdue car le lien entre ces deux nœuds est cassé.

La conception et la formulation de notre modèle se déroule sur deux phases :

Phase 1 : Cette phase nécessite l'identification du comportement du système désiré et ses paramètres (nombre d'entités autonomes).

Phase 2 : Construction du système artificiel, cette phase est divisée en deux :

- **Modélisation des entités** : par identification de ces entités, en définissant leurs caractéristiques (états, buts, …) et leur voisinage qui nécessite le calcul d'une certaine mesure (distance) permettant de définir la zone d'interaction entre les entités autonomes dans la procédure de communication, ainsi la définition des phases de découverte et de maintenance de route (par l'utilisation du protocole AODV).
- **Modélisation de l'environnement** : par identification de ces caractéristiques et définition de sa représentation.

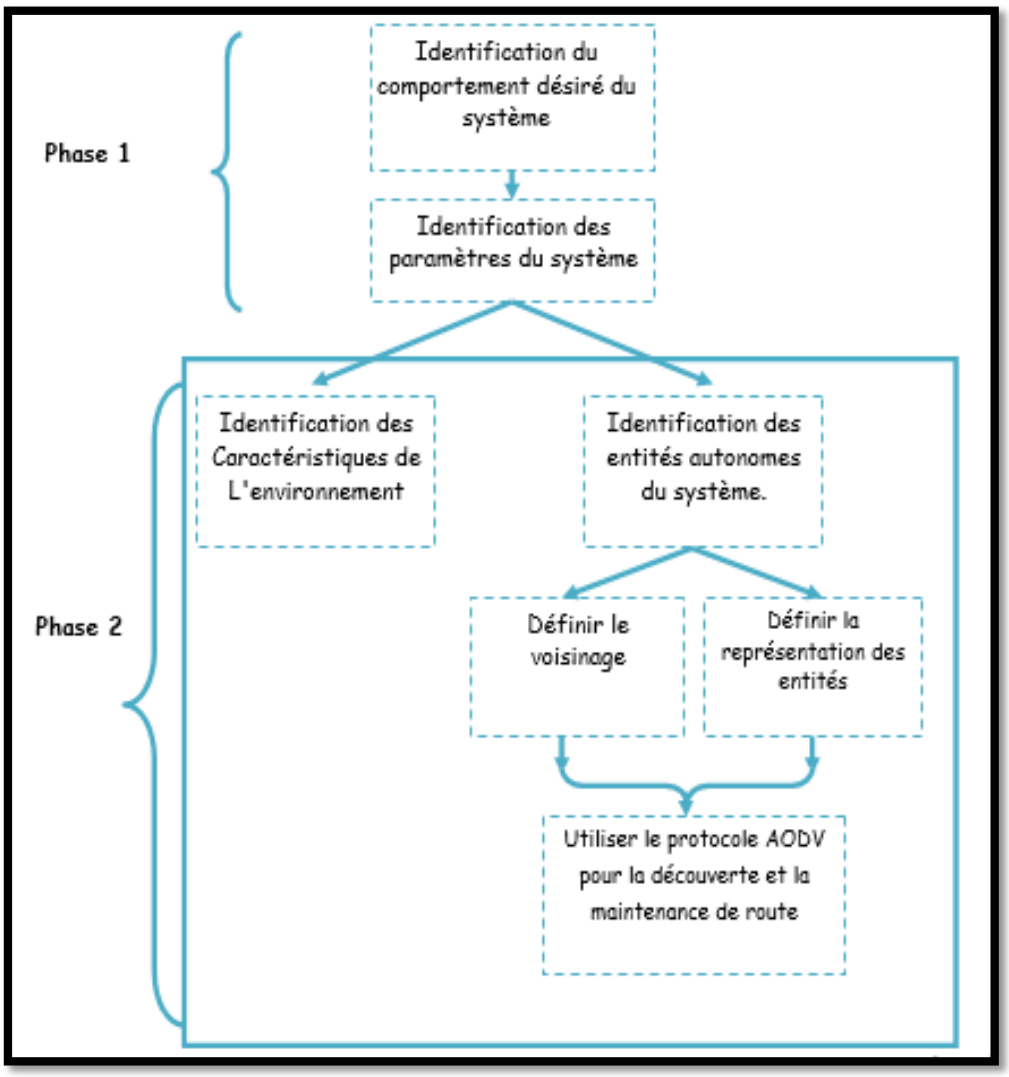


Figure IV. 4: Phases principales et initiales du travail.

La figure IV.4 illustre le schéma représentant les phases principales et initiales du travail. La première phase consiste à déterminer le comportement désiré du système d'un point de vue général puis nous passons aux paramètres du système comme le nombre des nœuds à utiliser, la deuxième phase contient la définition des caractéristiques de l'environnement de l'autre côté les caractéristiques des entités autonomes du système et la perception de ces entités en utilisant le voisinage afin de découvrir des chemins pour garantir la communication et la poursuite de ces derniers.

IV.4. Mise en œuvre

IV.4.1. Simulation

La simulation est l'étude du comportement dynamique d'un système grâce à des modèles qui évoluent dans le temps selon des règles clairement définies à des fins prédictives. Pour plus de détails voir le chapitre III.

IV.4.2. Environnement de simulation NETLOGO

Pour notre simulation, nous avons utilisé le simulateur NetLogo v6.1.1. Le choix du simulateur NetLogo est dû aux nombreux avantages qu'il présente par rapport aux simulateurs NS2 et OMNET ++, en termes de sa rapidité d'apprentissage et son environnement graphique simple.

IV.4.2.1. Choix

Comme nous avons mentionné auparavant, nous avons opté pour la plateforme NetLogo afin de simuler le protocole AODV. Ce logiciel possède le premier avantage d'être conçu en JAVA (langage orienté objet), ce qui le rend portable sur n'importe quelle machine. D'autre part, il utilise une extension du langage LOGO ce qui permet à un utilisateur non informaticien de développer des programmes sans connaissance des langages informatiques classiques. En outre il est relativement simple à manipuler et possède des outils intégrés permettant un traitement aisé des résultats comme les illustrations graphiques. Mais l'avantage principal de ce langage c'est qu'il permet la modélisation d'un système complexe composé d'un nombre infini d'entités autonomes situées dans un environnement statique ou dynamique, et fonctionnant en parallèle. Ainsi il permet la simulation des interactions entre entités. Finalement on peut dire que NetLogo permet la modélisation des comportements des entités autonomes au niveau micro et le comportement émergent au niveau macro.

IV.4.2.2. Définition

NetLogo est une application autonome écrite en Java, qui peut donc s'exécuter sur tous les principaux systèmes d'exploitation. NetLogo est un produit stable et rapide. Il s'agit d'un logiciel gratuit, tout le monde peut le télécharger gratuitement et de construire des modèles sans restriction. Il est accompagné d'une documentation complète et des tutoriels et une grande collection d'échantillons de modèles.

Figure IV. 5: Lancement du simulateur NetLogo v 6.1.1.

IV.4.2.3. Présentation

NetLogo se compose d'une seule fenêtre depuis laquelle sont accessibles différentes interfaces via des onglets. Le monde de NetLogo est constitué d'agents. Les agents sont des êtres capables d'exécuter des instructions. Chaque agent peut avoir sa propre activité, mais tous agissent simultanément.

Cet environnement de développement repose principalement sur quatre types d'agents [89]:

- **Les Tortues** : ce sont des agents mobiles, munis de capacités décisionnelles, se déplaçant et interagissant avec leur environnement.
- **Les Patches** : ils constituent l'environnement cellulaire sur lequel se déplacent les tortues.
- **Les Liens** : un type particulier d'agent qui met en relation deux agents et il est représenté comme une ligne tracée entre ces agents. Ce lien peut être orienté ou non.
- **L'Observateur** : il offre une vue « extérieure » du monde constitué par les patches et les tortues. Il permet une analyse du comportement du collectif d'entités.

Dans cet environnement de simulation, tous les agents « tortues » fonctionnent en parallèle sur une grille de patches (c'est à dire, un monde cellulaire).

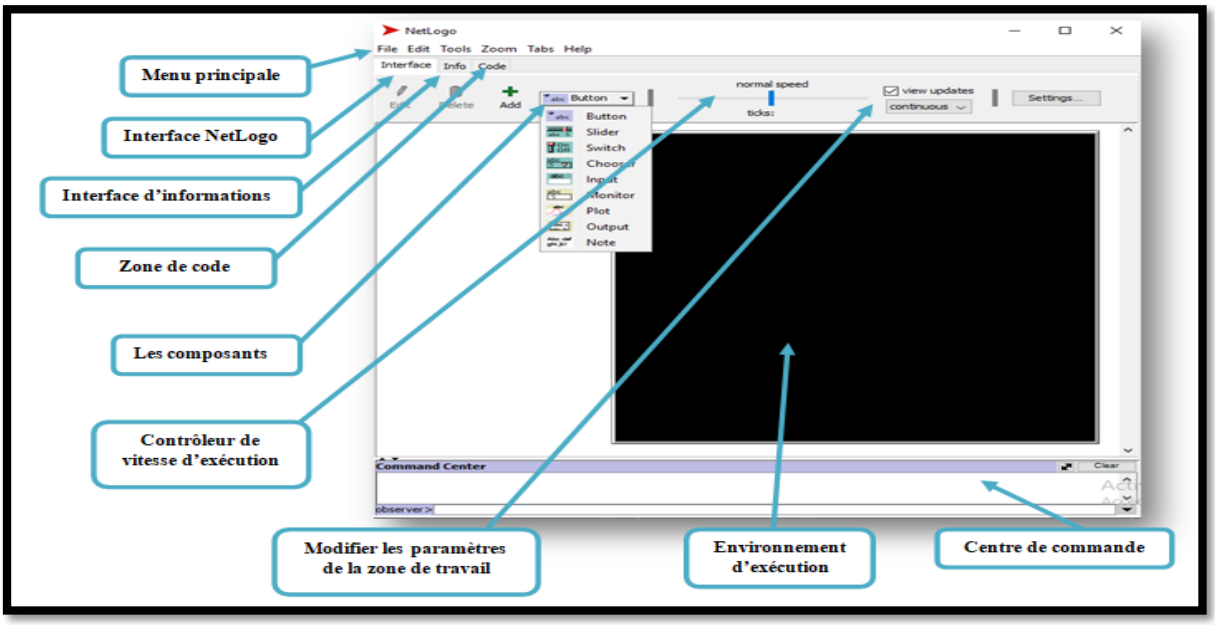


Figure IV. 6: Interface du NetLogo.

L'interface graphique est divisée en trois parties [89]**:**

- Une première partie présente le modèle graphique associé au monde cellulaire des patches.
- Une deuxième partie contient les différents outils (boutons, indicateurs…) permettant à l'utilisateur d'interagir avec le modèle.
- Une troisième partie permet de visualiser le déroulement des différentes procédures.

IV.4.3. Simulation du protocole de routage AODV dans NetLogo

NetLogo s'exécute sur la machine virtuelle Java, il peut donc s'exécuter sur toutes les machines virtuelles plateformes principales (Mac, Windows, Linux, etc.). La simulation d'AODV est implémentée sur la plateforme multi-agents NetLogo pour bénéficier de différentes possibilités fourni par cet outil.

IV.4.3.1. Pourquoi simuler avec NetLogo

Les réseaux ad hoc sont caractérisés par une forte **dynamique** et **auto-organisation**. Afin de prouver que ce type du réseau peut présenter un comportement global (chaque nœud a une vue sur l'ensemble du réseau), nous avons choisi le simulateur NetLogo.

Ce simulateur est particulièrement adapté à la simulation de systèmes complexes qui évolue au fil du temps. Les modélisateurs peuvent guider (donner des instructions a) des centaines d'agents de NetLogo indépendants et concurrents. Cela permet d'explorer les liens entre les comportements individuels (niveau micro) et donc de montrer le comportement global (niveau macro).

IV.4.3.2. Paramètres de simulation

Toutes les procédures manipulant les agents dans NetLogo sont écrites via l'interface « Code » et seront exécutées soit grâce à des boutons que l'on peut crées par le biais de l'interface graphique, soit en tapant leur nom dans le centre de commande. Chacune des entités autonomes est représentée par une tortue possédant des commandes qui lui sont propre.

Le tableau suivant contient des informations sur notre modèle de simulation :

Paramètres de simulation	Paramètres de simulation dans notre travail
Objectif à atteindre	En vue d'adapter NetLogo à un simulateur réseau nous avons choisi un des protocoles les plus connus de la famille des protocoles réactifs pour le simuler, ce qui nous permettra de mieux comprendre le fonctionnement de ce simulateur et de détecter ses manques.
Type de simulation utilisée	Simulations dynamiques • Système qui change dans le temps. • Simulations Stochastiques. • Les entrées et sorties sont aléatoires. (Deux simulations ne donnent pas le même résultat). Nous allons utiliser la simulation à évènement discret (changement d'état qu'en plusieurs points de l'axe du temps) ou à pas de temps. Le pas de temps représente un événement temporel qui est le temps nécessaire pour faire le tour sur tous les nœuds du réseau pour les consulter, et détecter s'il y a des évènements produits qui nécessitent du traitement au niveau de chaque nœud.
Modèle utilisé	Les tortues « agent-nœuds » de NetLogo : Monde artificiel composé d'agents en interaction dans un environnement simulé, La dynamique du système est le résultat des actions locales et interactions entre agents

Tableau IV. 1: Description de paramètres de simulation dans notre travail.

Comme nous l'avons vu auparavant dans notre travail nous allons simuler le protocole AODV, dans ce qui suit nous représenterons les paramètres essentiels de ce travail.

Les paramètres de notre travail	Les paramètres avec NetLogo
Les agents-nœuds, les RREQS les RREPS les RERRS et les paquets de données	Sont représentées par des tortues caractérisés par : Une couleur : **color** une taille : **size** une direction : **heading** une position : **xcor** et **ycor** une forme : **shape**
Chaque ensemble du nœud, possèdes ses propres caractéristiques.	Le travail collectif est assuré avec NetLogo par la création des races : la primitive « breed »
chaque ensemble d'agents peut avoir ses propres propriétés comme il peut avoir des propriétés communes avec les autres ensembles.	Chaque race peut avoir ses propres variables déclarées dans le bloc **turtles-own** [] (Nœuds-own, RREQS-own, RREPS-own, RERRs-own, et paquets-own), et il peut avoir des variables communes déclarées dans le bloc **Globals** [].
Nombre des agents-nœuds à utiliser	Selon l'utilisateur : un **Input** est utilisé pour saisir le nombre des agents-nœuds.
Les interactions entre les agents-nœuds	Sont représentées par des **links** entres les agents-nœuds.
Dans notre modèle, l'environnement où se déplacent les agents-nœuds est représenté par un ensemble d'entités fixes appelées « les cellules ».	Chaque cellule correspond à un patch, l'ensemble des patches forme l'environnement qui est caractérisé par une couleur et une taille.
Les actions des agents	Les actions des tortues sont implémentés dans des **procédures** ; la procédure commence par le mot clé « To » et fermé par « **end** »
L'exécution des actions des agents	Les procédures sont exécutées soit grâce à des **boutons** que l'on peut crées par le biais de l'interface graphique, soit en tapant leur nom dans le **centre de commande.**

Tableau IV. 2: Paramètres de notre travail avec NetLogo.

IV.4.3.3. Description des tortues utilisées

L'approche présentée est une approche décentralisée, les agents sont homogènes : tous les agents sont identiques et disposent des mêmes capacités de perception et d'action, ainsi la communication est explicite.

Les agents-nœuds doivent pouvoir acheminer des paquets de données sous certaines contraintes du fonctionnement à savoir :

- L'environnement est borné par une bordure que les agents-nœuds ne peuvent pas dépasser.
- Un patch ne peut contenir qu'un agent-nœud comme il peut rester vide.

1) Agents-nœuds

Les agents-nœuds sont représentés par des tortues, chaque nœud a des propriétés montrées dans le tableau suivant :

Propriétés d'agent-nœuds	Les propriétés du nœud avec NetLogo
Ensemble des agents-nœuds	Breed [nœuds nœud]
Une forme de cercle	set-default-shape noeuds "circle"
Une couleur gris	set color gray
Une direction	set heading one-of [0 90 180 270]
Une taille	set size 1
Un identificateur	[who] of nœud
Une position	setxy xcor random-xcor setxy ycor random-ycor
Un label	set label who
Une couleur de label noire	set label-color black

2) Route Request RREQs

Les RREQS sont représentés par des tortues, chaque RREQ a des propriétés montrées dans le tableau ci-dessous :

Propriétés de RREQ	Les propriétés de RREQ avec NetLogo
Ensemble des RREQS	Breed [RREQS RREQ]
Une forme d'enveloppe	set-default-shape RREQS " letter sealed "
Une couleur jaune	set color yellow
Un identificateur	[who] of RREQ
Une taille	set size 1

3) Route Replay RREPs

Les RREPS sont représentés par des tortues, chaque RREP a des propriétés montrées dans le tableau ci-dessous :

Propriétés de RREP	Les propriétés de RREP avec NetLogo
Ensemble des RREPS	Breed [RREPS RREP]
Une forme de symbole ok	set-default-shape RREPS " check "
Une couleur turquoise	set color turquoise + 2
Un identificateur	[who] of RREP
Une taille	set size 1

4) Route Error RERRs

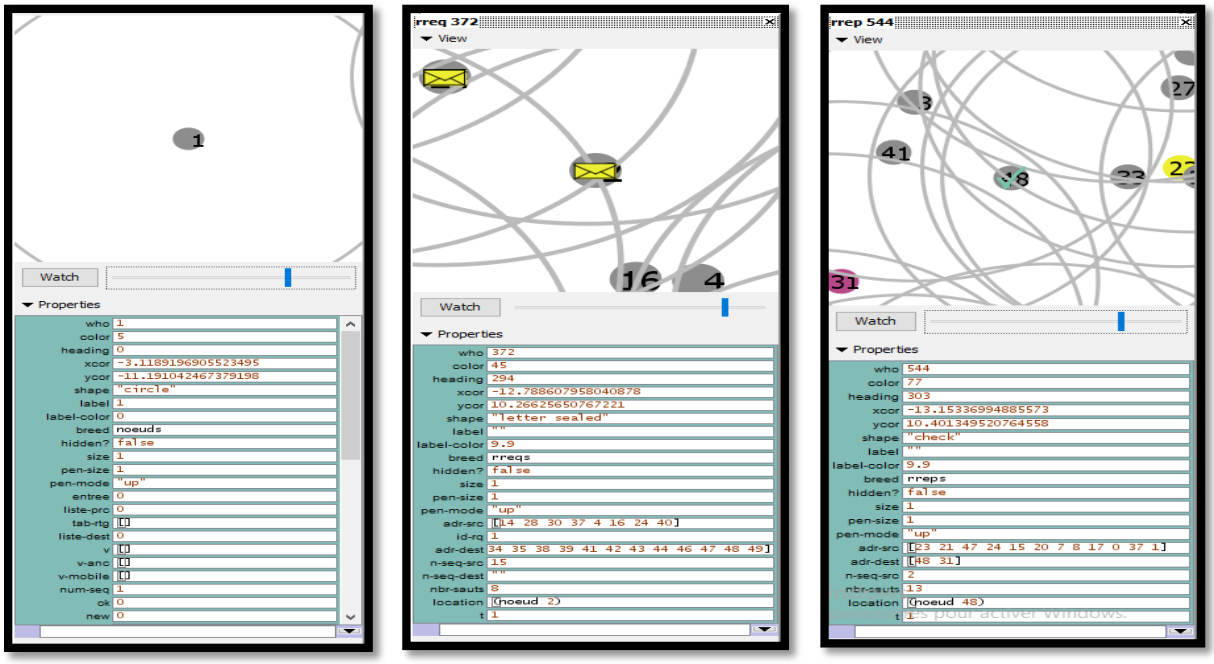
Les RERRs sont représentés par des tortues, chaque RERR a des propriétés montrées dans le tableau ci-dessous :

Propriétés de RERR	Les propriétés de RERR avec NetLogo
Ensemble des RERRS	Breed [RERRs RERR]
Une forme de symbole erreur	set-default-shape RERRS " x"
Une couleur rouge	set color red
Un identificateur	[who] of RERR
Une taille	set size 1

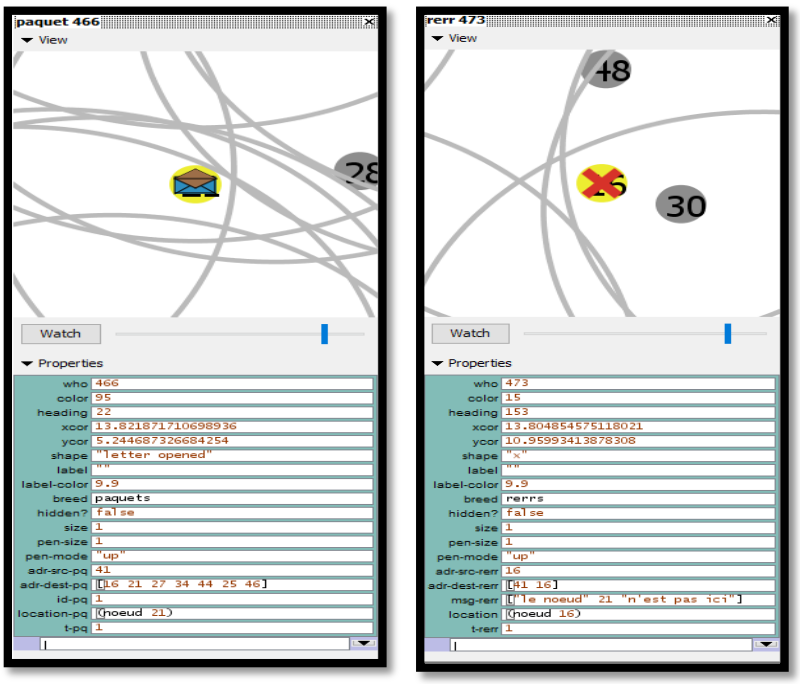
5) Les paquets de données

Les paquets sont représentés par des tortues, chaque paquet a des propriétés montrées dans le tableau suivant :

Propriétés de Paquet	Les propriétés de Paquet avec NetLogo
Ensemble des paquets	Breed [paquets paquet]
Une forme d'enveloppe ouverte	set-default-shape paquets "letter opened"
Une couleur bleu	set color sky
Un identificateur	[who] of paquet
Une taille	set size 1



(a) (b) (c)



(d) (e)

Figure IV. 7: Tortues utilisées.

« a : Nœud (gris), b : RREQ (jaune), c : RREP (turquoise), d : Paquet (bleu), e : RRER (rouge) ».

IV.4.3.4. Description de l'environnement

L'environnement est considéré comme un espace à deux dimensions divisé en cellules, ce dernier permet aux agents-nœuds de se déplacer et de communiquer entre eux dans le but d'exécuter des tâches complexes.

Dans notre cas nous utilisons un environnement ouvert de taille (20*20) comme la montre la figure ci-dessous :

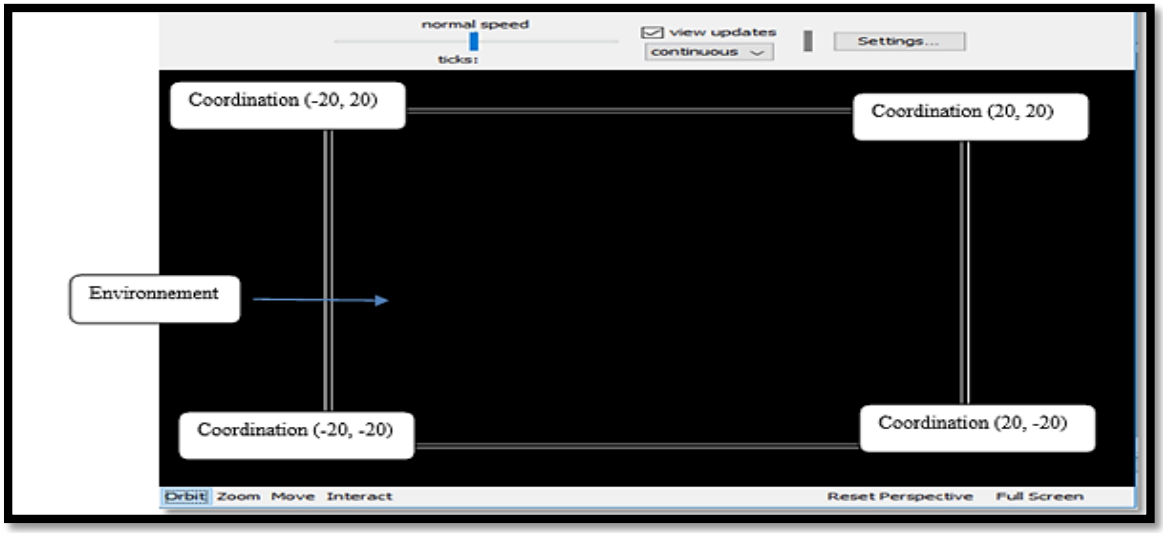


Figure IV. 8: Environnement de simulation.

IV.4.4. Présentation de notre simulation

IV.4.4.1. Description de l'interface de l'application

L'interface de notre application englobe un ensemble d'options à partir desquelles on peut accéder à notre système, la figure IV.9 montre ses différentes options.

Dans notre contexte du travail, nous pouvons divisée notre application de simulation dans NetLogo en trois parties :

Première partie : elle contient les options d'initialisation de la simulation.

Deuxième partie : elle englobe les options principales qui correspondent aux fonctions liées au protocole simulé AODV qui est divisé en deux phases (découverte, maintenance de route) et aux tracés utilisés.

Troisième partie : c'est une partie supplémentaire, elle contient un protocole simple qui consiste à envoyer directement un message aux voisins d'un nœud, de plus elle contient des procédures permettant de découvrir les voisins, de construire la table de routage et de rechercher un nœud.

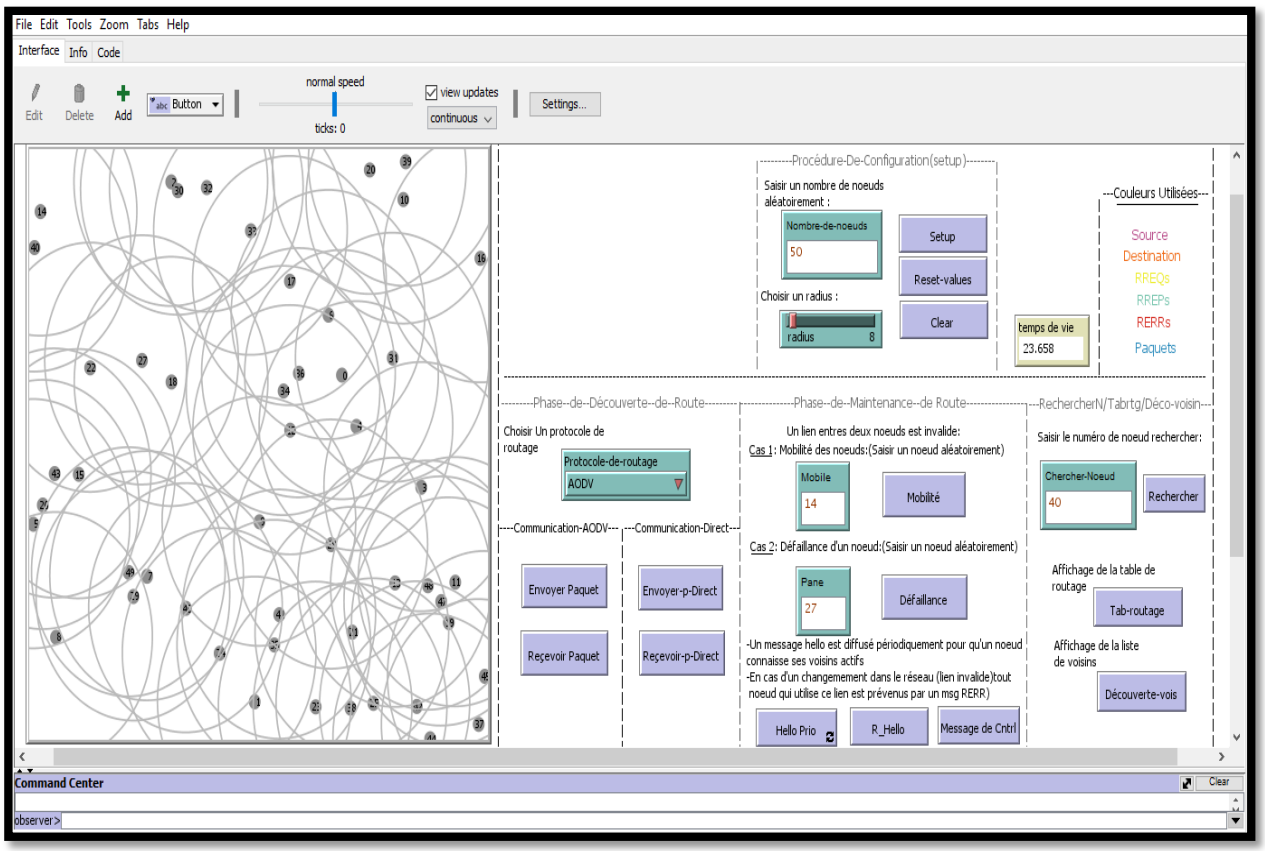


Figure IV. 9: Interface de notre application.

Description de la figure ci-dessus :

A. Première partie

1. Environnement de simulation : c'est un espace limité, qui contient les agents-nœuds, les sources et les destinations, les liens et les zones de portée, et différents types de paquets (données, contrôle).

2. L'input Nombre-de-Nœud : permet d'introduire le nombre d'agents-nœuds dans le réseau.

3. Le bouton Setup : permet d'initialiser le modèle, c'est-à-dire définir les conditions de départ de modèle, comme la taille de la zone de portée, la création des types d'agents (nœuds, RREQs, RERRs) …etc.

4. Le bouton Reset-value : permet la réinitialisation des valeurs des variables globales et tous les tracés (c'est à dire les mettre à zéro).

5. Le bouton Clear : permet la réinitialisation de l'environnement.

6. Le glissière (slider) radius : permet de définir la taille de la zone de portée d'un nœud dans le réseau.

B. Deuxième partie

Découverte de route

1. Le sélecteur (chooser) protocole-de-routage : permet de choisir un des deux protocoles de routage existant (**AODV**, **Direct**).

2. Le bouton Envoyer paquet : une fois que l'utilisateur a choisi le protocole de routage **AODV**, ce bouton permet à un nœud qui veut communiquer avec une destination de découvrir une route s'il ne possède pas de route valide qui mène vers cette dernière. Dans le cas où il n'y a pas de chemin vers cette destination une boite de dialogue sera affichée.

3. Le bouton Recevoir paquet : permet de retourner une boite dialogue qui confirme la réception du paquet de donnée par la destination et indique son ignorance au niveau des nœuds intermédiaires.

Maintenance de route

1. L'input Mobile et le bouton Mobilité : permet à un utilisateur de choisir un nœud et de le faire déplacer ce qui peut engendrer une coupure d'une route utilisée pour le routage des paquets ou d'une route existante dans la table de routage.

2. L'input Pane et le bouton Défaillance : permet à un utilisateur de choisir un nœud et le rendre dans un état inactif dans le réseau de ce fait un lien peut devenir invalide pour l'échange des paquets.

3. Le bouton Hello-perio : un message hello est diffusé périodiquement pour que le nœud connaisse ses voisins actifs (qui sont dans sa portée radio) et les insèrent dans sa table de

routage. Dans le cas d'une coupure de route causée par la mobilité ou la défaillance d'un nœud, les nœuds utilisant ce lien sont prévenus par un message d'erreur RERR envoyé par un des nœuds voisins du nœud causant cette coupure.

4. Le bouton R-Hello : le nœud source qui reçoit un RERR constate que le chemin vers la destination voulu est invalide donc il initié la procédure de communication à nouveau.

Tracés (plots)

Les fonctions de traçage de NetLogo nous permettent de créer des tracés pour nous aider à comprendre ce qui se passe dans un modèle. Dans notre simulation ces deux plots permettent de tracer le nombre de messages envoyés et celui reçus pour chaque paquet qui traverse le réseau.

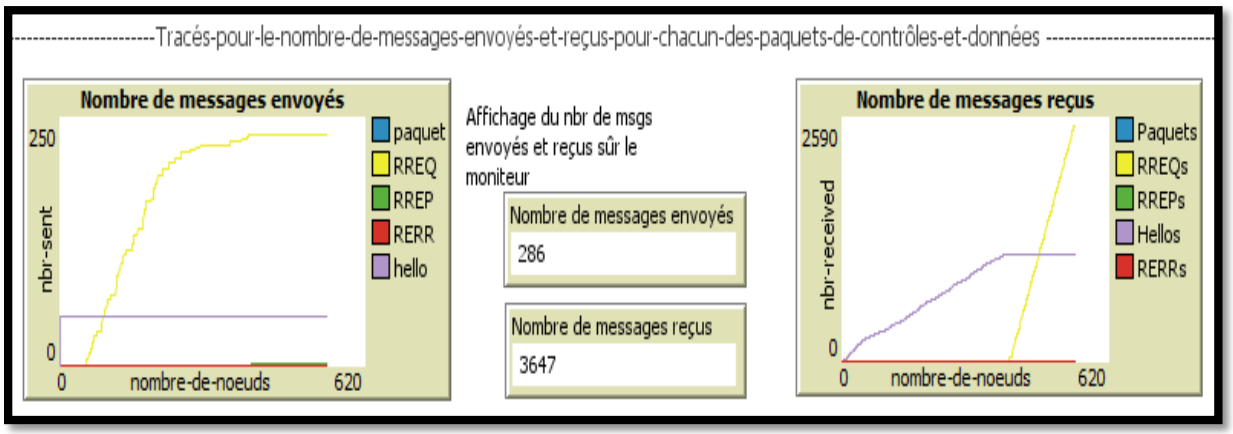


Figure IV. 10: Plots du nombre des paquets envoyés et reçus dans notre application.

La figure IV.11 présente le tracé d'énergie consommée par les nœuds qui interagissent dans le réseau.

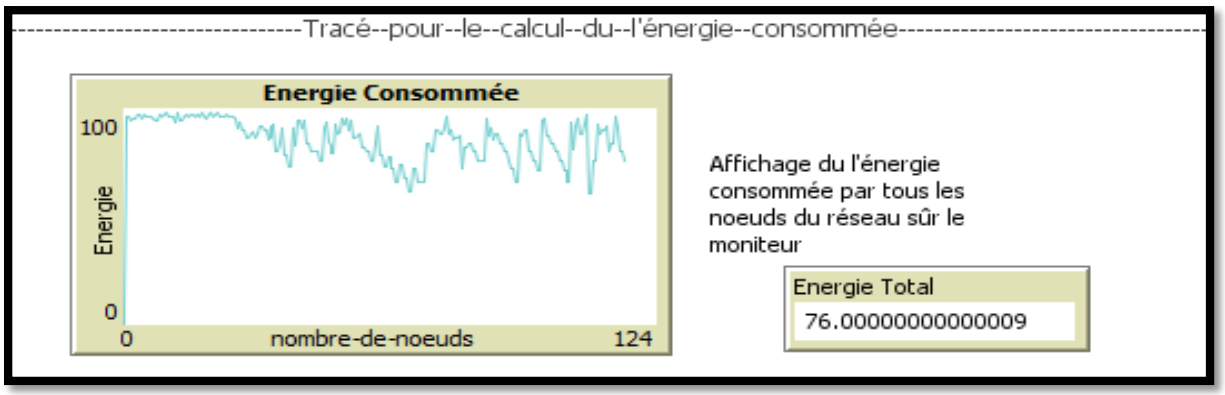


Figure IV. 11: Plot de l'énergie consommée dans notre application.

C. Troisième partie

1. Le bouton Envoyer-p-direct : après le choix du protocole de routage **direct,** ce bouton permet d'envoyer un paquet de donnée d'une source à une destination dans le cas où celle-ci est dans sa zone de portée donc il existe un lien direct entre les deux nœuds.

2. Le bouton Recevoir-p-direct : une boite de dialogue indiquant la réception du paquet de données par la destination sera affichée pour confirmer sa réception.

3. L'input Chercher-Nœud et le bouton Rechercher : ils sont utilisés pour la recherche d'un nœud dans l'environnement, et il rend sa couleur violet pour faciliter la tâche de recherche dans le réseau.

4. Le bouton tab-routage : affiche la table de routage de chaque nœud.

5. Le bouton découverte-vois : pour chaque nœud du réseau il sélectionne les nœuds se trouvant dans sa portée.

IV.4.4.2. Description de la communication

Afin d'établir la communication entre les agents-nœuds, nous allons utiliser les éléments et les actions de base sur lesquelles repose le protocole AODV.

Les actions suivantes sont englobées dans le cadre de cette communication :

1. Emission de RREQ, lorsqu'un nœud veut communiquer avec une destination et il ne possède pas une route vers cette dernière, une requête de demande de route est diffusée sur tout le réseau jusqu'à l'arrivé à la destination.

2. Réception de RREQ, un nœud qui reçoit un RREQ soit qu'il est une destination ou qu'il à la route vers cette destination, prépare une réponse contenant le plus court chemin qui mène au nœud destinataire.

3. Emission de RREP, le nœud destinataire où un nœud intermédiaire ayant la plus courte route vers le nœud cible répond par un RREP à la source suivant ce chemin inversé.

4. Réception de RREP, lorsque un nœud source reçois le RREP récupère le chemin traversé par ce dernier et l'inverse afin de l'utiliser pour l'envoie du paquet de donnée.

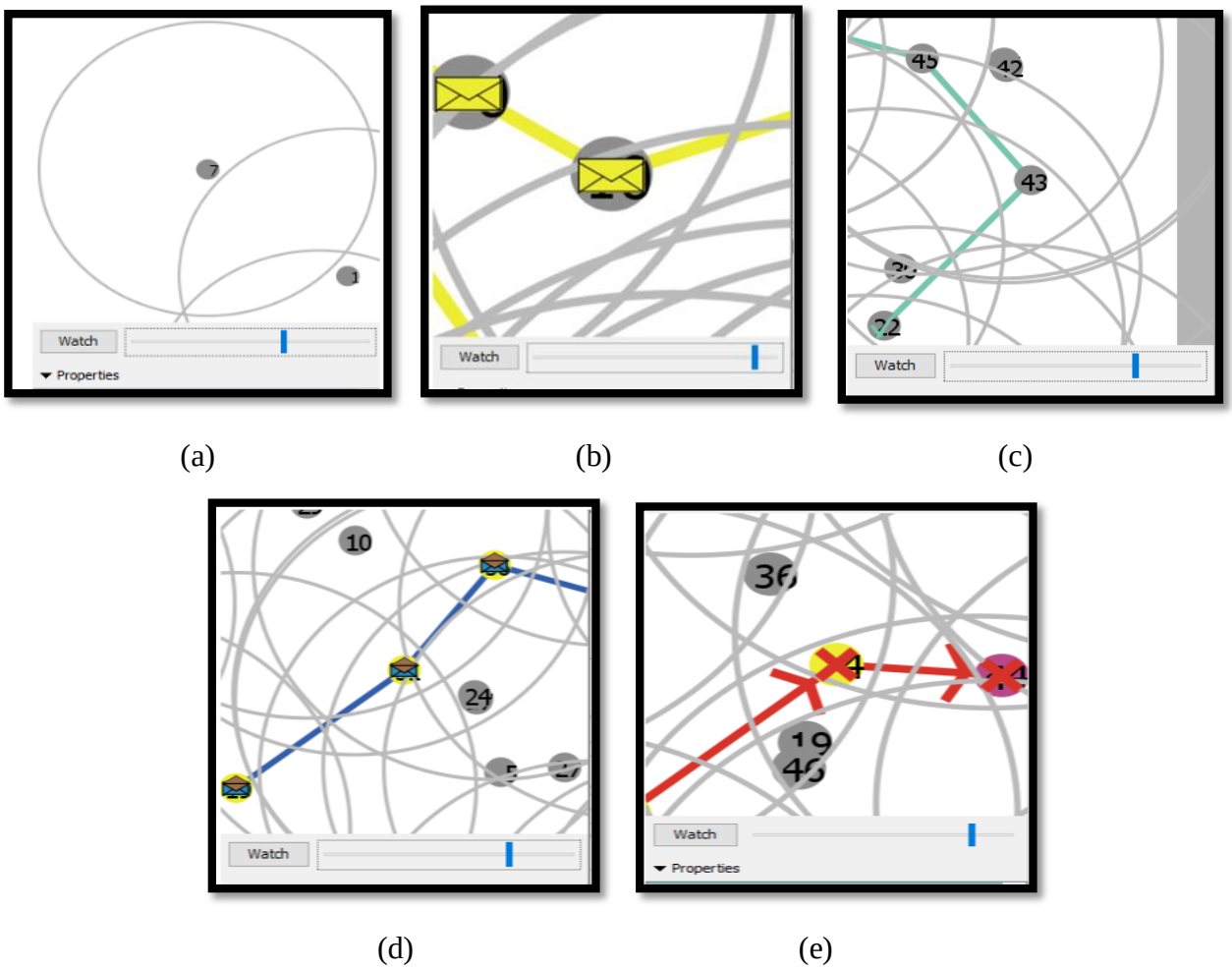
5. Emission de paquet de donnée, la source envoie le message sur le plus court chemin.

6. Réception de paquet de donnée, une fois le paquet est reçu par le destinataire une boite de dialogue apparaitra qui indique la bonne réception de ce paquet et son négligence au niveau des autres nœuds.

7. Emission de Hello, est l'action de diffusion périodique du message Hello pour la découverte des voisins, s'il trouve un changement de ceux-ci il met à jour la table de routage.

8. Emission de RERR, le rôle de ce message apparait lors d'une coupure d'une route utilisée qui est causée soit par la mobilité ou par la défaillance d'un nœud, un RERR est envoyé au nœud utilisant ce lien pour lui indiquer sa non-disponibilité.

9. Réception de RERR, dès qu'un nœud reçoit ce paquet de contrôle qu'indique la non-validité de ce chemin, il rediffuse le message RREQ.



(a) (b) (c)

(d) (e)

Figure IV. 12: Description de communication.

Cette figure illustre les différentes actions exécutées afin de permettre les échanges entre les agents-nœuds dans le réseau :

(a) Présente la zone de portée d'un nœud qui est l'étendue où un paquet est reçu avec succès s'il n'y a pas d'interférences.

(b) Emission/ réception RREQ.

(c) Emission/ réception RREP.

(d) Emission/ réception paquet.

(e) Emission/ réception RERR.

Comme mentionné auparavant le protocole **AODV** est divisé en deux phases qui englobent les actions qu'un nœud peut exécuter. Le tableau ci-dessous présente la partition de ces actions :

Découverte de route	Maintenance de route
Emission de RREQ Réception de RREQ Emission de RREP Réception de RREP	Emission de Hello Emission de RERR Réception de RERR

Tableau IV. 3: Phases d'AODV.

Dans notre simulation, l'agent-nœud prend la décision de l'action qu'il va effectuer selon son état actuel qui est stocké dans une file appelée « file-attente », le nœud peut être dans plusieurs états qui sont exécutées suivant le processus ci-dessous :

- ➔ Le nœud ne peut pas exécuter l'action **émission RREQ** si sa file d'attente contient déjà une action **émission RREQ** c.à.d. le nœud ne peut pas diffuser deux RREQS au même temps.
- ➔ Le nœud ne peut pas exécuter l'action **émission RREP** sans avoir l'état **réception RREQ** dans sa file d'attente c.à.d. le nœud ne peut pas répondre à une requête qui n'a pas encore reçue.
- ➔ Le nœud ne peut pas exécuter l'action **réception RREP** sans avoir l'état **émission RREQ** dans sa file d'attente c.à.d. le nœud ne peut pas recevoir une réponse à une requête qui n'a pas diffusé.
- ➔ Le nœud ne peut pas exécuter l'action **émission paquet** sans avoir l'état **réception RREP** dans sa file d'attente c.à.d. le nœud ne peut pas envoyer le paquet de donnée avant la réception de RREP (avant de connaitre la route au destinataire).
- ➔ Le nœud ne peut pas exécuter l'action **réception du paquet** sans avoir l'état **émission RREP** dans sa file d'attente c.à.d. le nœud ne peut pas recevoir un paquet de donnée avant de répondre à la requête.
- ➔ Le nœud ne peut pas exécuter l'action **émission RERR** sans avoir un cas de mobilité où de défaillance qui cause une coupure de route.
- ➔ Le nœud ne peut pas exécuter l'action **réception RERR** sans avoir l'état **émission RERR** c.à.d. le nœud ne peut pas recevoir le paquet RERR avant qu'il soit émis par le nœud détectant la coupure.

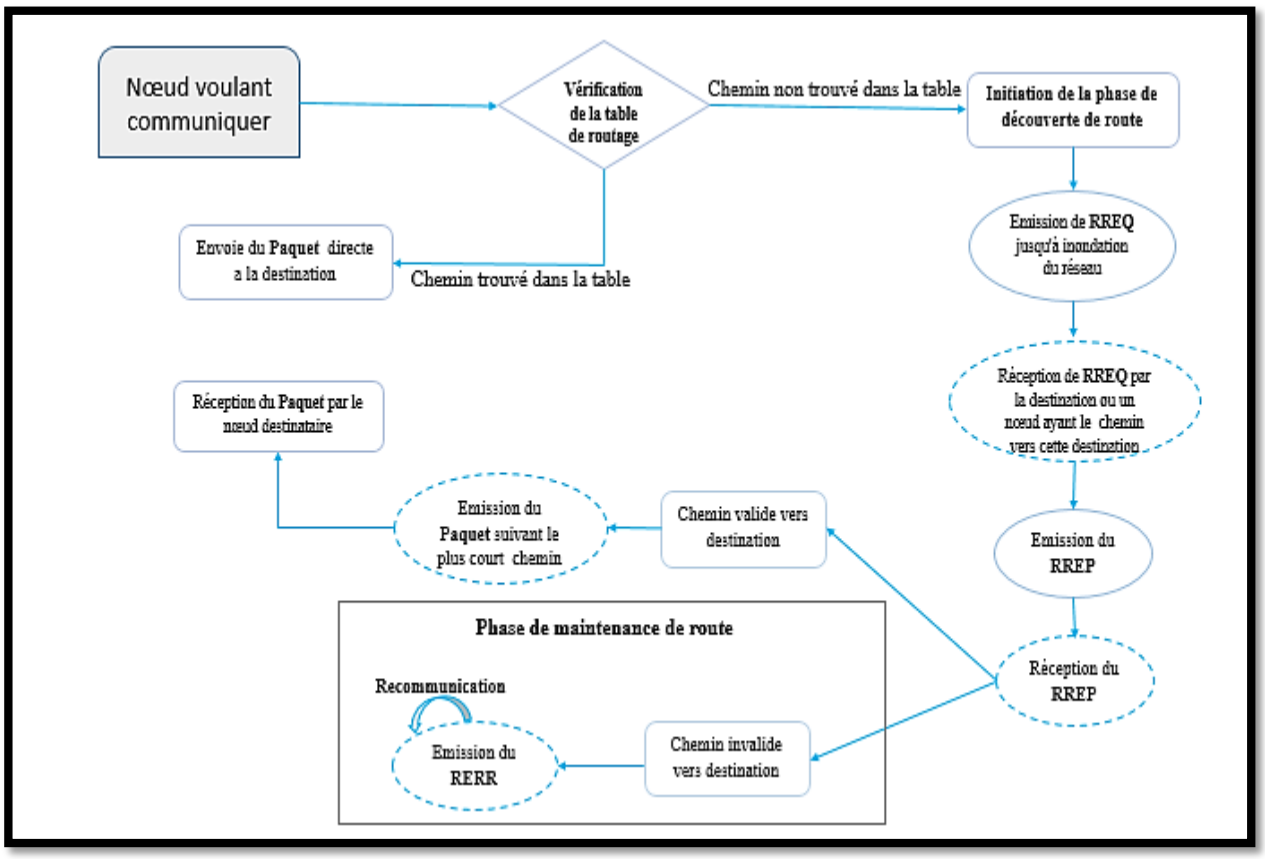


Figure IV. 13: Diagramme d'états et actions de communication du protocole AODV.

Cette figure présente l'ensemble des états et les actions utilisées par un nœud source désirant communiquer avec un nœud destinataire.

IV.4.4.3. Communication entre les agents-nœuds

La communication entre les agents-nœuds dans le protocole **AODV** se déroule comme suit :

La découverte de route

- Si le nœud lançant la procédure de communication trouve le nœud destinataire dans sa table de routage, il l'envoie directement le paquet de données suivant le prochain saut.
- Si le nœud veut communiquer avec un autre nœud et il ne possède pas de chemin vers lui, il diffuse un RREQ en mode « Broadcast » (à ses voisins actifs) afin d'obtenir la plus courte route qui permet d'envoyer le paquet de donnée au destinataire (dans notre cas un des autres agents-nœuds est le destinataire).
- Si le nœud trouve son identificateur dans la liste des destinataires, donc il incrémente son numéro de séquence puis il forme un RREP et il continue à diffuser le RREQ aux autres agents-nœuds tan qu'il n'est pas le nœud destinataire recherché ou qu'il ne connait pas le chemin pour l'atteindre.

- Chaque destinataire répond à la requête par l'envoie de RREP qui est formée de l'adresse source (le nœud lui-même), l'adresse destination (le nœud qui est à l'origine de la requête), son numéro de séquence et le nombre de sauts.
- Si le nœud qui est à l'origine de la requête reçoit le RREP, il insère le destinataire dans sa table de routage avec : son numéro de séquence, nombre de sauts, liste des agents-nœuds précurseurs, ensuite il envoi le paquet de donnée au nœud inséré dans sa table de routage suivant le plus court chemin enregistré par le RREP.

Adresse source	Identificateur de la requête	Adresse destination	Numéro de séquence de destination	Nombre de sauts

Figure IV. 14: Format du paquet de contrôle RREQ.

Adresse source	Adresse destination	Numéro de séquence source	Nombre de sauts

Figure IV. 15: Format du paquet de contrôle RREP.

La maintenance de route

- Périodiquement un message hello est diffusé afin de vérifier la connectivité ou bien l'activité des routes donc la disponibilité des voisins (actifs).
- Si un nœud se déplace ou quitte le réseau, des coupures de route peuvent empêchées la procédure de communication ce qui nécessite l'envoie d'un paquet RERR pour les nœuds qui utilise ce lien.
- Si un nœud reçoit un RERR, il met à jour sa table de routage et fait appel une autre fois à la procédure de communication.

Adresse source	Adresse destination	Message du Paquet de contrôle

Figure IV. 16: Format du paquet de contrôle RERR.

Toute communication se fait à base d'une table de routage, la figure IV.17 présente le format de la table de routage utilisée pour notre simulation.

Adresse destination	Numéro de séquence de la destination	Prochain saut	Nombre de sauts	Liste des précurseurs
…….	…….	…….	…….	…….

Figure IV. 17: Format de la table de routage.

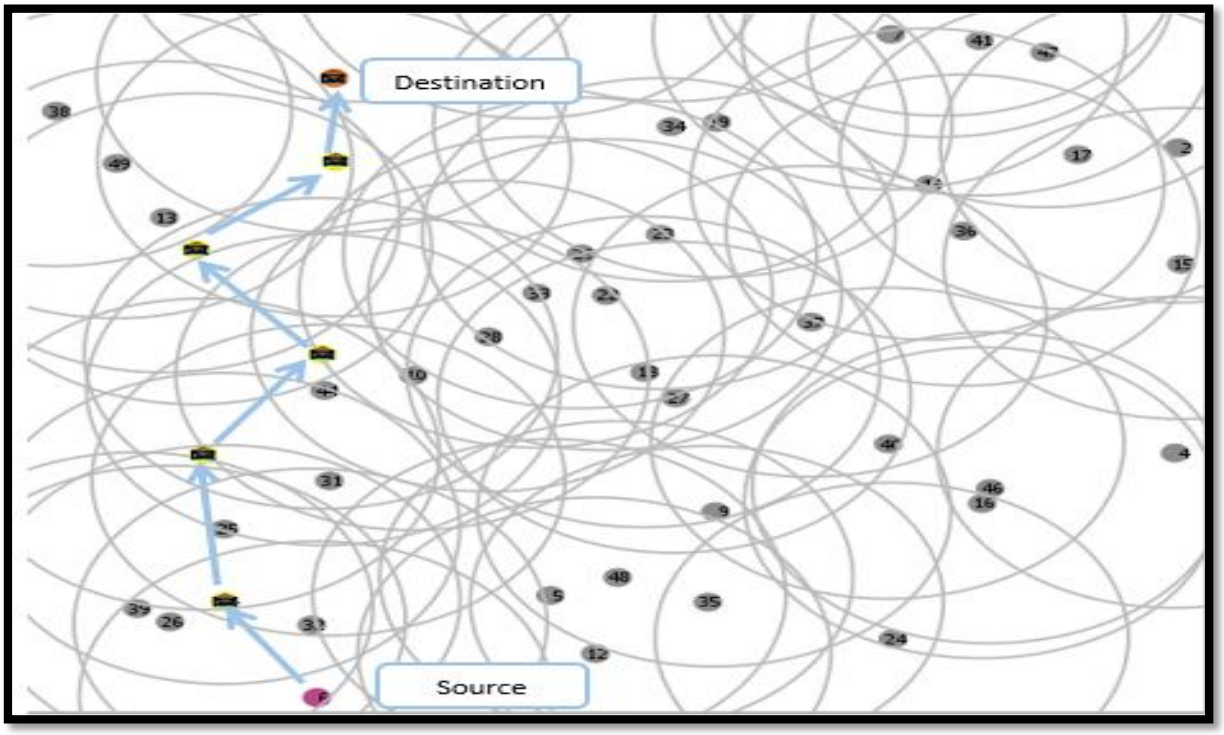


Figure IV. 18: Communication entre agents-nœuds (Emission du paquet de données).

La figure ci-dessus montre l'émission du paquet de données d'une source vers une destination suivant le plus court chemin.

IV.4.4.4. Implémentation des actions

Nous avons implémenté les agents-nœuds par des turtles de breed « nœud », Les 11 actions qui peuvent-être exécutés par les agents-nœuds sont implémentées comme suit :

Procédure émission_RREQ [nd desti]

Début

Demander au nœud source numéro [nd]

Si sa file d'attente ne contient pas une émission RREQ Et le nœud veut communiquer **Alors**

Envoyer RREQ aux voisins actifs pour les diffuser jusqu'à atteindre nœud [desti]

Mis à jour de sa file d'attente par l'ajout de l'action émission RREQ

Sinon

Sauvegarder la requête pour l'envoyer ultérieurement

Fin

Procédure réception_RREQ [desti]

Début

Demander au nœud destination numéro [desti]

S'il a reçu une requête RREQ **Alors**

Si le couple [adresse source et identificateur de la requête] est reçu pour la première fois **Alors**

Incrémenter le nombre de sauts de RREQ

Former le paquet RREP

Mis à jour sa file d'attente par l'ajout de l'action réception RREQ

Sinon

Ignorer la requête RREQ

Sinon

Rien à faire

Fin

Procédure émission_RREP [desti]

Début

Demander au nœud destination numéro [desti]

Si sa file contient l'action de réception RREQ **Alors**

Envoyer le paquet RREP au nœud qui est la source de la requête RREQ via

Le chemin inverse de RREQ

Mis à jour sa file d'attente par l'ajout de l'action émission RREP

Sinon

Rien à faire

Fin

Procédure réception_RREP [nd]

Début

Demander au nœud source numéro [nd]

Si sa file contient déjà l'action d'émission de RREQ Alors

Insérer dans sa table de routage le nœud qui est la source du paquet RREP

Sinon

Rien à faire

Fin

Procédure émission_Paquet [nd desti]

Début

Demander au nœud destination numéro [desti] récupérer le plus court chemin

Demander au nœud numéro [nd]

Si sa file d'attente contient déjà l'action réception de RREP **Alors**

Envoyer le paquet de donnée au nœud destination sur le plus court chemin enregistré dans sa table de routage

Supprimer les actions d'émission de RREQ et réception RREP de la file d'attente

Sinon

Rien à faire

Fin

Procédure réception_Paquet

Début

Demander aux nœuds

Si sa file d'attente contient déjà l'action émission RREP **Alors**

Si le nœud est le destinataire **Alors**

Afficher le message « Je suis la destination, Bien reçu »

Sinon

Afficher le message « Je ne suis pas la destination »

Tuer le paquet

Supprimer les actions de réception RREQ et émission RREP de la file d'attente

Sinon

Rien à faire

Fin

Procédure Emission-RRER [nd]

Début

Demander au nœud numéro [nd]

Si nœud-mobile = vrai **ou** nœud-pane = vrai **Alors**

Construire un RRER

Envoyer RERR aux nœuds utilisant ce nœud pour la construction de route

Sinon

Rien à faire

Fin

Procédure Réception-RRER [nd]

Début

Demander au nœud numéro [nd]

Si déjà reçu un RERR **Alors**

Mettre à jour la table de routage

Appel à la procédure de communication pour trouver un autre chemin vers la destination

Sinon

Rien à faire

Fin

Procédure Déplacer [nd]

Début

Demander au nœud numéro [nd]

Si la case en face, à gauche ou à droite est vide (selon sa position) **Alors**

Se déplacer

Fin

Procédure Communication [nd desti]

Début

Demander au nœud numéro [nd]

Si le nœud desti appartient à la table de routage **Alors**

Envoyer le paquet

Sinon

Découverte-de-route nd desti

Fin

Procédure Découverte-de-route [nd desti]

Début

Répéter

Demander au nœud numéro [nd]

Emission-RREQ nd desti

Demander aux nœuds

Réception-RREQ desti

Emission-RREP desti

Jusqu'à atteindre la destination

Si Destination trouvé **Alors**

Demander au nœud numéro [nd]

Réception-RREP nd

Emission Paquet nd desti

Fin

IV.4.5. Interprétation des résultats de la simulation

La phase de simulation joue un rôle très important pour la modélisation de comportement d'un réseau sur une machine. L'AODV est un protocole de routage dans les réseaux ad hoc basé sur la demande de route. Nous avons pu simuler ce protocole sur un simulateur multi-agents, permettant de modéliser des phonèmes complexes ce qui lui permet donc de tolérer un nombre important de nœuds et d'interactions entre eux, et même il supporte une forte mobilité. Ce simulateur non adapté au réseau, est le sujet de notre étude pour pouvoir récolter les informations essentielles qui nous permettent de comprendre le fonctionnement de NetLogo et de voir les manques que présente ce simulateur dans le domaine réseau.

Notre simulation de ce protocole, nous a permis de comprendre le principe du protocole de routage et de voir la nécessité ou bien le besoin fort des procédures propres au simulateur qui nous permettra de l'utiliser d'une façon un peu plus simple et rapide, dont nous trouvons :

- La table de routage étant une liste de listes peut poser une contrainte de perte de temps pour son remplissage et l'utilisation des informations stockées dedans.
- La découverte de voisins dans la portée peut-être une fonction intégré dans le simulateur NetLogo.
- La mise en œuvre d'un mécanisme d'inondation facilitera la diffusion d'un paquet de contrôle.

De ce fait l'adaptation du ce simulateur au domaine réseau est un travail intéressant pour le rendre parmi les simulateurs les plus utilisés dans ce domaine.

Conclusion

La simulation est une étape primordiale pour les chercheurs, et le choix d'un simulateur et de phénomène a simulée repose sur le but que nous voulons atteindre. Dans ce chapitre nous avons expliqué la démarche que nous avons suivie pour résoudre notre problème.

Pour la simulation du protocole AODV, nous avons modélisé le fonctionnement de ce protocole qui est divisé en deux phases consistant à découvrir des chemins et de les maintenir, grâce à des paquets de contrôles RREQs, RREPs, et RERRs.

Après, nous avons simulé AODV dans le simulateur NetLogo. Et nous avons exposé la description de notre application de simulation, ainsi que les différents paramètres, les agents, et l'environnement utilisés.

Dans le chapitre suivant nous allons mettre les points sur la façon d'adapter ce simulateur multi-agents à un simulateur réseau en ajoutant des extensions.

Chapitre V

Développement
d'Extensions dans NetLogo

V.1. Introduction

NetLogo est un outil de recherche puissant. À une certaine époque, ce simulateur était une plateforme fermée. Les utilisateurs ne pouvaient pas le modifier ou l'étendre, ni le contrôler à partir d'un code externe. Aujourd'hui, cela a changé NetLogo devient extensible.

Dans ce chapitre, nous allons voir pourquoi utiliser les extensions et montrer comment utiliser dans un modèle les extensions qui sont soit réutilisées téléchargeables ou crées, ainsi nous allons citer quelques exemples des extensions déjà crées et nous allons mettre l'accent sur les extensions que nous envisageons intégrer afin de rendre NetLogo adapté au réseau.

V.2. Simulateur NetLogo et Extensibilité

Le langage **NetLogo** est un membre de la famille Lisp [13](List Processing) qui supporte les agents et la concurrence. Il est utilisé pour construire une variété infinie de simulations (Voir le chapitre IV).

Dans cette section, nous décrivons comment le NetLogo est récemment devenu **extensible** grâce à l'ajout de nouvelles facilités d'extensions. Auparavant, nous avons décrit NetLogo comme un environnement intégré ou "All-in-one". Son environnement complet regroupe de nombreux composants : un langage de programmation, un compilateur, un interprète, un éditeur de syntaxe, un constructeur d'interface, un moteur graphique, un espace de comportement, etc.

L'inconvénient de l'approche " All-in-one " est que cette approche peut se transformer en "All-or-Nothing". D'où un utilisateur risque que si un composant ne répond pas à ses besoins, celui-ci ne pourra utiliser aucun des composants, car ils sont tous liés entre eux. Pour éviter ce piège du "All-or-Nothing ", les utilisateurs peuvent étendre ou remplacer les parties du NetLogo qui ne conviennent pas à leurs besoins.

De cette façon, même les utilisateurs qui ont des besoins uniques, ou simplement des besoins auxquels les constructeurs de ce simulateur n'ont pas pensé ou que n'ont pas encore pris le temps d'aborder, peuvent construire eux-mêmes ce dont ils ont besoin, et ils obtiendront toujours le bénéfice du reste de travail de ces constructeurs. Une nouvelle API est une étape vers cet objectif.

L'environnement NetLogo intégré fournit des fonctionnalités de base, cette API permettra aux utilisateurs avancés de sortir de ce noyau. En rendant NetLogo extensible, ils comblent le fossé entre les environnements de modélisation intégrés (faciles à utiliser, mais potentiellement restrictifs) et les outils de modélisation (plus flexible, mais beaucoup plus difficiles à utiliser) [90].

[13] C'est la plus ancienne famille de langage de programmation à la fois impératives et fonctionnelles. Développé initialement en tant que modèle pratique pour représenter des programmes.

V.3. Extensions

NetLogo a toujours été un langage de programmation qui permet aux utilisateurs d'écrire des procédures dans ce dernier et les utiliser ensuite comme des commandes intégrées .Depuis NetLogo 2.0.1. NetLogo offre une application interface de programmation (**API**) pour les extensions afin que les utilisateurs puissent ajouter et écrire de nouvelles commandes et de nouveaux reporters (primitives) en Java et dans d'autres langages à utiliser dans les modèles NetLogo [91] .Cela permet aux utilisateurs d'ajouter de nouveaux types de fonctionnalités à ce dernier.

Par exemple, un utilisateur peut laisser les agents créer des sons et de la musique en utilisant les capacités MIDI[14] (Musical Instrument Digital Interface) de Java, ou communiquer avec des ordinateurs distants, et bien d'autres choses encore.

N.B : En plus des nombreuses extensions fournies avec NetLogo, de nombreuses autres extensions peuvent être installées via le gestionnaire d'extensions [91] .

V.3.1. Présentation

Chaque extension de NetLogo se compose d'un dossier portant le même nom que l'extension, entièrement en minuscules. Ce dossier doit contenir un fichier **JAR** portant le même nom que le dossier. Par exemple, l'extension **sound** est stockée dans un dossier appelé **sound** avec un fichier à l'intérieur appelé **sound.jar**.

Ce fichier JAR doit avoir à son tour le contenu suivant :

- Une ou plusieurs classes qui implémentent org.nlogo.api.**Primitive**.
- Une classe principale qui implémente org.nlogo.api.**ClassManager**.
- Un fichier **manifeste** d'extension NetLogo, avec les quatre balises suivantes :
  - Manifest-Version, toujours 1.0.
  - Extension-Name, le nom de l'extension.
  - Class-Manager, le nom complet d'une classe implémentant org.nlogo.api.ClassManager.
  - NetLogo-Extension-API-Version, la version de l'API d'extension NetLogo à laquelle ce fichier JAR est destiné. Si un utilisateur ouvre l'extension avec NetLogo qui a une version d'API d'extension différente, un message d'avertissement est émis.

[14] La norme MIDI (Musical Instrument Digital Interface) définit un protocole de communication pour les appareils de musique électroniques, tels que les instruments à clavier électronique et les ordinateurs personnels.

Certaines extensions dépendent de fichiers supplémentaires. Ces fichiers seront dans le dossier de l'extension avec le fichier JAR. Le dossier peut également contenir d'autres fichiers tels que la documentation et des exemples de modèles.

Remarque :

Pour créer une extension, il faudra inclure NetLogo.jar dans le chemin de classe. De plus, le lib répertoire (également de la distribution NetLogo) doit être au même emplacement que NetLogo.jar ; il contient des bibliothèques supplémentaires utilisées par NetLogo.jar [92].

V.3.2. Développement

Le développement d'une extension NetLogo nécessite une connaissance du langage **Java** et les outils associés. Il est également possible d'utiliser à partir d'autres langages pour la machine virtuelle Java, comme **Scala.**

V.3.2.1. Java

C'est un langage de programmation **orienté objet** créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java. Une particularité de Java est que les logiciels écrits dans ce langage sont compilés vers une représentation binaire intermédiaire qui peut être exécutée dans une machine virtuelle Java (**JVM**) en faisant abstraction du système d'exploitation [93].

V.3.2.2. Scala

C'est un langage de programmation **multi-paradigme** conçu à l'École polytechnique fédérale de Lausanne (EPFL) pour exprimer les modèles de programmation courants dans une forme concise et élégante. Son nom vient de l'anglais Scalable language qui signifie à peu près « langage adaptable » ou « langage qui peut être mis à l'échelle ». Il peut en effet être vu comme un métalangage.

Scala intègre les paradigmes de programmation orientée objet et de programmation fonctionnelle, avec un typage statique. Il concilie ainsi ces deux paradigmes habituellement opposés (à de rares exceptions près, telle que le langage OCaml (Objective Categorical Abstract Machine Language) et offre au développeur la possibilité de choisir le paradigme le plus approprié à son problème. Il est prévu qu'il soit compilé en bytecode Java (exécutable sur la JVM), ou .NET. Seule la plateforme Java est supportée officiellement par l'EPFL [94].

V.3.3. Plugin sbt

Une extension sur NetLogo peut être crée à partir de n'importe quel système de génération Java tel qu'Ant, Gradle ou Maven. Le CCL[15] (Center for Connecting Learning) utilise et recommande sbt[16] avec le plugin d'extension NetLogo . Ce plugin gère les étapes suivantes [95]:

- Ajout du fichier JAR NetLogo approprié en fonction du paramètre du NetLogoVersion dans le fichier build.sbt.
- Configuration sbt pour attacher le manifeste approprié aux fichiers JAR dans le cadre du package task.
- Empaqueter l'extension et toutes les dépendances de jar dans un fichier .zip qui peut être distribué aux utilisateurs. Alternativement, un utilisateur peut choisir de déplacer le fichier .jar et toutes les dépendances dans la racine du projet.

Étant donné que le CCL utilise le plugin d'extension NetLogo, il sera toujours mis à jour avec les dernières étapes de construction nécessaires pour utiliser une extension sans nécessité d'intervention de l'utilisateur.

V.3.4. Emplacement

NetLogo recherchera des extensions à plusieurs endroits [91]:

1. Dans le dossier du modèle actuel.
2. Dans le dossier des extensions du l'installation de NetLogo. Pour les installations NetLogo typiques :
   - Sous **Mac OS X** : /Applications/NetLogo 6.1.1/extensions.
   - Sous **Windows** 64 bits avec NetLogo 64 bits Ou Windows 32 bits avec NetLogo 32 bits: C:\Program Files\NetLogo 6.1.1\app\extensions.
   - Sur Windows 64 bits avec NetLogo 32 bits: C:\Program Files (x86)\NetLogo 6.1.1\app\extensions.
   - Sous **Linux** : le sous-répertoire app\extensions du répertoire NetLogo extrait de l'installation.tgz.
3. Ce sous-dossier (relatif à votre répertoire personnel) :

[15] Le Centre pour l'apprentissage connecté et la modélisation informatique (CCL) de Northwestern est dédié à l'utilisation créative de la technologie pour approfondir l'apprentissage.

[16] C'est un outil de construction (build) pour la JVM, à l'instar de Maven ou Gradle. À partir d'un projet, il permet, entre autres, de gérer ses dépendances, de compiler, d'exécuter des tests et de publier les artefacts sur des dépôts.

- Sous **Mac OS X** : Library/Application Support/NetLogo.
- Sous **Windows** : AppData\NetLogo.
- Sous **Linux**: .NetLogo.

4. Le sous-dossier .bundled du dossier extensions mentionné au point 2(par exemple, /Applications/NetLogo 6.1.1/extensions/.bundled sur Mac OS X).

V.3.5. Utilisation

L'utilisation d'extensions indique à NetLogo de rendre les commandes et les reporters des extensions spécifiées disponibles dans le modèle actuel, comme s'il s'agissait de primitives NetLogo intégrées.

Pour utiliser une extension dans un modèle, il faut ajouter le mot - clé « **extensions** » au début de l'onglet Code, avant de déclarer des races ou des variables.

Extensions : est suivi d'une paire de crochets contenant une liste de noms d'extension. Par exemple : **extensions [**sound speech**]** [92]**.**

Ainsi pour son utilisation, il faut suivre les règles suivantes [91] :

- La façon la plus simple d'installer de nouvelles extensions consiste à utiliser le gestionnaire d'extensions.
- Si une extension n'est pas disponible via le gestionnaire d'extensions, il faut la télécharger manuellement et la placer dans l'emplacement décrit par l'élément 1 ou l'élément 2 ci-dessus.
- La modification manuelle du contenu des éléments 3 et 4 ci-dessus n'est pas prise en charge.
- L'ordre de l'emplacement des extensions indiqué ci-dessus est la **priorité** qu'un gestionnaire d'extension utilisera. Cela signifie que si le gestionnaire d'extensions trouve une extension demandée pour un modèle installé manuellement dans le dossier des extensions, il ne vérifiera pas la bibliothèque d'extensions pour voir s'il existe des versions mises à jour à installer. En tant que tel, l'utilisateur est verrouillé sur la version installée manuellement jusqu'à ce qu'il décide de la supprimer. En outre, toutes les extensions qu'il a installées via le gestionnaire d'extensions remplaceront les extensions fournies avec NetLogo.

V.3.6. Création d'extensions prédéfinies avec sbt Windows

Parfois, un utilisateur voudra télécharger le code d'extension à partir de GitHub et le construire lui-même (éventuellement dans le but d'apporter des modifications). La méthode suivante fonctionnera pour les extensions construites avec sbt package, telles que les extensions NetLogo fournies. Ceux-ci sont attachés au NetLogo GitHub et peuvent être trouvés en recherchant les référentiels NetLogo contenant le mot "Extension". Tout d'abord, le modélisateur accède à

l'extension qu'il souhaite télécharger sur GitHub. Puis il télécharge le référentiel sous forme d'un fichier ZIP et ensuite il le décompresse. Et s'il souhaite modifier le code, Il faut qu'il ouvre le répertoire /src et il modifie les fichiers texte à l'intérieur en utilisant son éditeur de texte ou l'IDE de son choix [95].

N.B : Certaines étapes d'installation et de configuration sont requises pour les outils de création d'une extension.

Après ceci l'utilisateur sera prêt à créer l'extension :

- Il ouvre l'invite de commande (recherche «cmd» dans le menu Démarrer).
- Il entre la commande "cd [répertoire]" pour accéder à l'extension (par exemple, "cd C : \ Extension GoGo").
- Ensuite, il entre "sbt package " et son code devrait être compilé à cet étape. S'il y a des erreurs dans son code, l'invite de commande affichera des messages d'erreur.
- Enfin, l'utilisateur devra trouver un fichier .jar dans le répertoire d'extension. Et le copie dans le répertoire approprié dans NetLogo (par exemple C: \ Program Files (x86) \ NetLogo 6.0 \ Java \ extensions \ gogo). La prochaine fois qu'il exécutera NetLogo, il devra avoir la nouvelle version de l'extension [95].

V.4. Conseils de développement d'une extension

V.4.1. Instanciation

Le gestionnaire de classe est instancié dans un nouveau chargeur de classe JVM au moment où un modèle utilisant l'extension est chargé. Cela est fait pour que les extensions soient déchargeables, de sorte que lorsqu'un utilisateur ouvre un nouveau modèle, les extensions utilisées par le modèle précédent puissent être déchargées et la mémoire utilisée a été récupérée [92].

V.4.2. Chemin de classe

Il faut inclure NetLogo.jar dans le chemin de classe lors de la compilation. Il s'agit d'une erreur la plus courante commise par les nouveaux auteurs d'extensions. Si le compilateur ne trouve pas NetLogo.jar, l'utilisateur obtiendra des messages d'erreur concernant les classes du package org.nlogo.api non trouvées. Si le répertoire lib n'est pas au même emplacement que NetLogo.jar, il obtiendra des erreurs sur les autres classes non trouvées [92].

N.B : Si un utilisateur utilise le plugin sbt d'extension NetLogo, la dépendance NetLogo sera ajoutée automatiquement à son build.

V.4.3. Débogage d'extensions

Il existe des primitives NetLogo spéciales pour aider l'utilisateur dans le développement et le débogage de son extension. Celles-ci sont considérées comme expérimentales et peuvent être modifiées ultérieurement, tels que :

- print__dump-extensions : Imprime des informations sur les extensions chargées.
- print __dump-extension-prims : Imprime des informations sur les primitives d'extension chargées.
- __reload-extensions : Oblige NetLogo à recharger toutes les extensions lors de la prochaine compilation d'un modèle. Sans cette commande, les modifications apportées à votre JAR d'extension ne prendront effet que lorsqu' un utilisateur ouvrira un modèle ou redémarrera NetLogo [92].

V.4.4. Tests de langue & performances

Un utilisateur peut exécuter le test de langue à partir de la session sbt de son extension, exemple l'extension NW l' utilise. De même, Il peut exécuter des tests de performances à partir de la session sbt de son extension [95].

V.4.5. Niveaux des JAR

Si l'extension d'un utilisateur dépend du code stocké dans un fichier JAR distinct, il copie les fichiers JAR supplémentaires dans le répertoire de l'extension. Chaque fois qu'une extension est importée, NetLogo met tous les fichiers JAR de son dossier à la disposition de l'extension.

S'il prévoie de distribuer son extension à d'autres utilisateurs de NetLogo, il doit s'assurer de fournir des instructions d'installation [92].

V.4.6. Documentation du l'extension

Il n'y a aucun moyen pour le modélisateur d'obtenir une liste de commandes et de reporters fournis par une extension, il est donc important que le modélisateur fournis une documentation adéquate [95].

V.5. Exemple explicatif pour la création d'une extension en Java

Pour apprendre la façon d'écrire une extension, nous allons présenter un exemple d'une extension qui fournit un seul reporter appelé first-n-integers, qui prendra une seule entrée numérique n et rapportera une liste des entiers de 0 à n- 1. Les étapes de cette écriture sont énumérées dans ce qui suit [92] :

V.5.1. Création d'un dossier d'extension

Puisqu'une extension est un dossier avec plusieurs éléments, d'abord il faut créer un dossier. Dans cet exemple, il est appelé « example ». Tout le travail sera fait dans ce dossier. Egalement, il faut créer un sous-dossier src/main/java pour le code Java.

V.5.2. Développement des primitives

Les primitives sont implémentées comme une ou plusieurs classes Java. Les fichiers .java de ces classes doivent être placés dans le sous-dossier src/main/java.

Une commande exécute une action, un reporter rapporte une valeur. Pour créer une nouvelle commande ou un nouveau reporter, il faut crée une classe qui implémente l'interface org.nlogo.api.Command ou org.nlogo.api.Reporter, qui étend org.nlogo.api.Primitive.

DefaultReporter nécessite l'implémentation :

```
Object report (Argument args[], Context context)
  throws ExtensionException;
```

Puisque le reporter prend un argument, il faut implémenter également :

```
Syntaxe getSyntax ();
```

Voici l'implémentation du reporter, dans un fichier appelé src/main/java/IntegerList.java :

```
import org.nlogo.api.*;

public class IntegerList extends DefaultReporter
{
    // take one number as input, report a list
    public Syntax getSyntax() {
        return Syntax.reporterSyntax(
            new int[] {Syntax.TYPE_NUMBER}, Syntax.TYPE_LIST
        );
    }

    public Object report(Argument args[], Context context)
        throws ExtensionException
    {
        // create a NetLogo list for the result
        LogoList list = new LogoList();

        int n ;
        // use typesafe helper method from
        // org.nlogo.api.Argument to access argument
        try
        {
            n = args[0].getIntValue();
        }
        catch( LogoException e )
        {
            throw new ExtensionException( e.getMessage() ) ;
        }

        if (n < 0) {
            // signals a NetLogo runtime error to the modeler
            throw new ExtensionException
              ("input must be positive");
        }

        // populate the list
        // note that we use Double objects; NetLogo numbers
        // are always doubles
        for (int i = 0; i < n; i++) {
            list.add(Double.valueOf(i));
        }
        return list;
    }
}
```

Figure V. 1: Classe IntegerList du reporter first-n-integers [92].

Remarque :

- Les objets numériques qui sont met dans la liste sont des doubles, pas des entiers. Tous les nombres utilisés comme valeurs NetLogo doivent être de type Double, même s'ils n'ont pas de partie fractionnaire.
- Pour accéder aux arguments, il faut utiliser les méthodes « typesafe helper » de org.nlogo.api.Argument, telles que getDoubleValue ().
- Le Lancement de l'org.nlogo.api.ExtensionException permet de signaler une erreur d'exécution NetLogo au modélisateur.
- Une commande est identique à un reporter, sauf que les reporters implémentent Object report (...) tandis que les commandes implémentent void perform (...).

V.5.3. Développement d'une classe Class Manager

Chaque extension doit inclure, en plus d'un certain nombre de classes de commandes et de reporters, une classe qui implémente l'interface org.nlogo.api.ClassManager. La classe ClassManager indique à NetLogo quelles primitives font partie de cette extension. Dans les cas simples, il faut étendre de la classe abstraite org.nlogo.api.DefaultClassManager, qui fournit des implémentations vides des méthodes ClassManager dont probablement le modélisateur n'a pas besoin.

Voici le gestionnaire de classe pour l'exemple d'extension src/main/java/SampleExtension.java

```
import org.nlogo.api.*;


public class SampleExtension extends DefaultClassManager {
    public void load(PrimitiveManager primitiveManager) {
        primitiveManager.addPrimitive
          ("first-n-integers", new IntegerList());
    }
}
```

Figure V. 2: Classe Manager du reporter first-n-integers [92].

AddPrimitive () : dit à NetLogo que le reporter existe et quel est son nom.

V.5.4. Ecriture d'un fichier manifeste

L'extension doit également inclure un manifeste. Le manifeste est un fichier texte qui indique à NetLogo le nom de l'extension et l'emplacement du fichier ClassManager.

Le manifeste doit contenir quatre balises :

- Manifest-Version, toujours 1.0.
- Extension-Name, le nom de l'extension.

- Class-Manager, le nom complet d'une classe implémentant org.nlogo.api.ClassManager.
- NetLogo-Extension-API-Version, la version de l'API d'extension NetLogo à laquelle ce fichier JAR est destiné. Pour savoir quelle version de l'API d'extension que NetLogo prend en charge, il suffit de choisir l'élément "À propos de NetLogo" dans le menu "Aide", puis une clique sur l'onglet Système. Ou bien par le lancement du fichier NetLogo.jar avec l'extension-api-version argument.

N.B : La version de l'API est rarement la même que la version NetLogo. La version NetLogo 6.0 aura une version API 6.0, mais NetLogo 6.0.1 et tout à fait possible NetLogo 6.1 auront également probablement la version API 6.0. Ce numéro de version augmentera lorsqu'une modification incompatible en amont est introduite dans le package org.nlogo.apiou org.nlogo.core. Voici un manifeste pour cette exemple d'extension appelé manifest.txt :

```
Manifest-Version: 1.0
Extension-Name: example
Class-Manager: SampleExtension
NetLogo-Extension-API-Version: 4.0
```

Figure V. 3: Fichier manifest du reporter first-n-integers [92].

La ligne NetLogo-Extension-API-Version doit correspondre à la version réelle de l'API NetLogo Extension que l'utilisateur utilise.

V.5.5. Création d'un fichier JAR

Si un utilisateur utilise le plugin sbt NetLogo, il peut demander à sbt de créer le package d'extension pour lui en exécutant sbt package. Il trouvera alors un fichier .zip à la racine du projet, qui peut être décompressé dans le répertoire d'extensions NetLogo pour installer l'extension. S'il souhaite construire le jar à la main, les instructions sont les suivantes :

Pour créer le fichier JAR d'une extension, Il faut d'abord compiler les classes comme d'habitude, à partir de la ligne de commande ou à l'aide d'un IDE.

N.B : Pour chacune des exemples d'extensions sur GitHub inclut une construction en ligne de commande, invoquée avec la commande sbt package.

Voici un exemple de l'apparence de la compilation de l'extension à partir de la ligne de commande sans l'utilisation du sbt ou make :

```
$ mkdir -p classes     # create the classes subfolder if it does not exist
$ javac -classpath NetLogo.jar -d classes src/main/java/IntegerList.java src/main/java/SampleExtension.java
```

L'utilisateur doit également modifier l'argument classpath pour pointer vers le fichier NetLogo.jar à partir de son installation NetLogo. Par exemple, sur Mac OS X, il faut faire :

```
javac -classpath "/Applications/NetLogo 6.0-BETA2/Java/netlogo-6.0.0-BETA2.jar" -d
classes src/main/java/IntegerList.java src/main/java/SampleExtension.java
```

Cette ligne de commande compilera le .java et placera les fichiers .class dans le sous-dossier classes. Par la suite, l'utilisateur crée un JAR contenant les fichiers de classe résultants et le manifeste.
Par exemple :

```
$ jar cvfm example.jar manifest.txt -C classes .
```

V.5.6. Utilisation de l'extension crée dans un modèle

Pour l'utilisation de cet exemple d'extension, l'utilisateur doit placer le dossier « example » dans le dossier d'extensions NetLogo ou dans le même répertoire que le modèle qui utilisera l'extension. En haut de l'onglet Code, il doit écrire :

```
extensions [example]
```

Et maintenant, l'utilisateur peut utiliser example : first-n-integers comme s'il s'agissait d'un reporter NetLogo intégré.
Par exemple, il sélectionne l'onglet Interface et il tape dans le Centre de commande :

```
observer> show example:first-n-integers 5
observer: [0 1 2 3 4]
```

Le schéma ci-dessous récapitule les différentes étapes à suivre pour le développement d'une extension sur NetLogo.

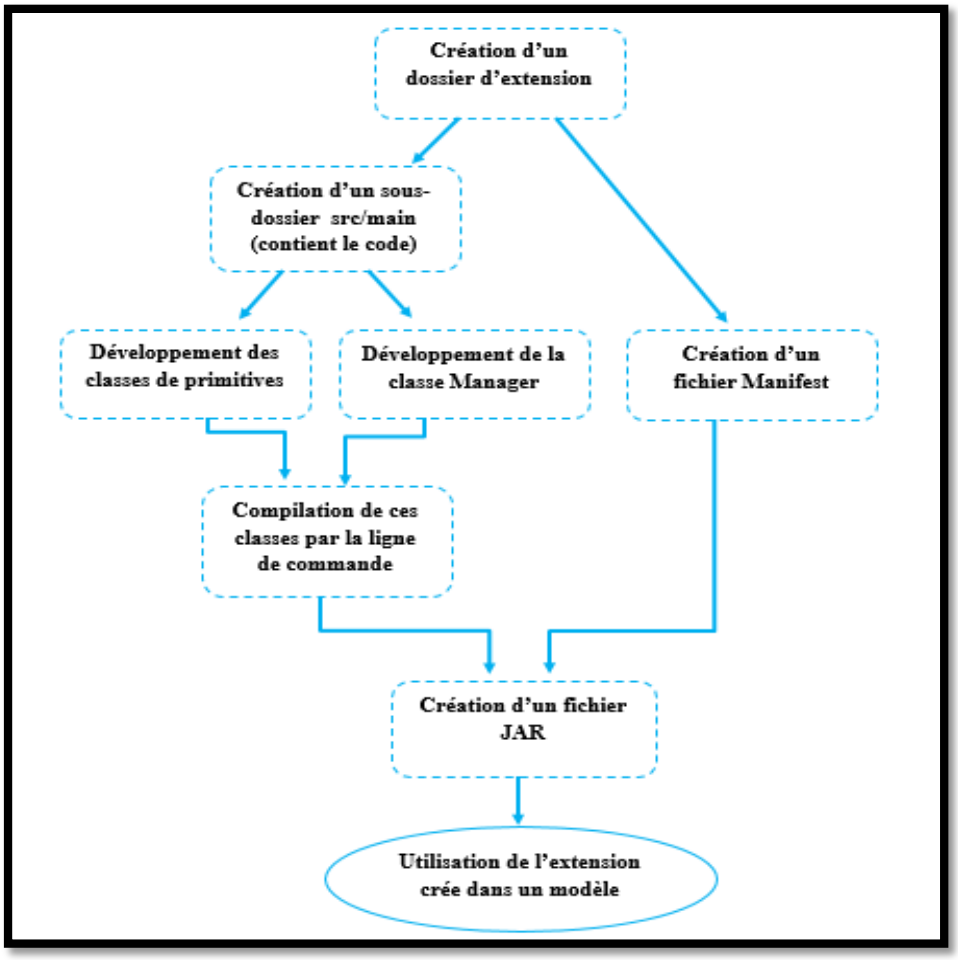


Figure V. 4: Schéma explicatif des étapes de création d'une extension.

V.6. Quelques extensions existantes sur le gestionnaire d'extensions NetLogo

1. Extension NetLogo NW pour l'analyse du réseau

Cette extension fournit un large ensemble d'outils d'analyse du **réseau** à utiliser dans **NetLogo**. Elle s'efforce de ne jamais calculer les choses deux fois. Elle se souvient de tous les chemins, distances et centralités qu'elle calcule pour être rapide dans autant de circonstances que possible. De plus, elle garde une trace des valeurs qu'elle vient juste de calculer en cours de route. Il y a quelques circonstances où l'extension **NW** doit oublier certaines valeurs. Si le réseau change du tout (par l'ajout des tortues ou des liens, ou par la suppression des tortues ou des liens), elle doit tout oublier. Pour les primitives pondérées, si la valeur de la variable de pondération change pour l'un des liens du réseau, elle oubliera les valeurs associées à cette variable de pondération. Si un utilisateur travaille sur un réseau qui peut changer régulièrement, il essaye de faire tous ses calculs du réseau en même temps, puis tout son réseau change en même temps.

Plus ses calculs de réseau entrelacés et ses changements de réseau sont nombreux, plus l'extension **NW** ne devra recalculer les valeurs.

Dans de rares cas, l'utilisateur ne souhaite pas que l'extension **NW** mémorise les valeurs. Par exemple, s'il travaille sur un réseau extrêmement volumineux, la mémorisation de toutes ces valeurs peut prendre plus de mémoire qu'il n'en a. Dans ce cas, il peut simplement appeler nw: set-context (first nw:get-context) (last nw:get-context) pour forcer l'extension **NW** à tout oublier immédiatement [96].

Les primitives d'extension **NW** considèrent toutes les tortues et tous les liens comme faisant partie du réseau actuel, parmi ces primitives nous pouvons citées quelques-unes [96]:

Chemin et distance

nw:turtles-in-radius , nw:distance-to , nw:path-to ,nw:turtles-on-path-to.

Gestion du contexte

nw:set-context , nw:get-context, nw:with-context.

Importer et exporter

nw:save-matrix , nw:load-matrix , nw:save-graphml ,nw:load-graphml … .

2. Extension Arduino pour NetLogo

Arduino est un espace de développement intégré (EDI) qui vous permet d'écrire, de compiler et d'envoyer du code sur le circuit imprimé du même nom. Pour rappel, la carte Arduino contient un microcontrôleur que l'on peut programmer dans le but d'effectuer des tâches variées [97].

Cette extension fournit une communication entre NetLogo et un Arduino connecté. Elle offre une fonctionnalité d'écriture simple de NetLogo sur la carte, ainsi que la possibilité de lire les données qui ont été envoyées de la carte à l'ordinateur et stockées dans une table de recherche de paires nom-valeur [98].

Un modèle NetLogo utilisant cette extension doit fonctionner en conjonction avec un Arduino Sketch. Ces deux points d'extrémité communiquent au moyen d'un protocole d'application qu'ils définissent. Par exemple, si le modèle NetLogo envoie un octet «1» sur le fil, cela peut signifier quelque chose pour l'Arduino Sketch, qui répondra en conséquence.

Quelques primitives de cette extension :

arduino:open, arduino:close, arduino:get, arduino:write-string… .

3. Extension NetLogo GoGo

L'extension GoGo permet à un utilisateur de connecter NetLogo au monde physique, à l'aide de capteurs, de moteurs, d'ampoules, de LED, de relais et d'autres appareils. L'extension GoGo pour NetLogo fournit des primitives pour communiquer avec une carte GoGo via une interface série.

Une carte GoGo est une carte open source, facile à construire, à faible coût et à usage général, spécialement conçue pour être utilisée dans des projets éducatifs. Une carte GoGo possède 8 ports

de capteur et 4 ports de sortie, ainsi qu'un connecteur pour des cartes complémentaires (comme un écran ou un module de communication sans fil). En utilisant l'extension GoGo, les modèles NetLogo peuvent interagir avec le monde physique de deux manières. Premièrement, il peut collecter des données sur l'environnement, telles que la température, la lumière ambiante ou les entrées utilisateur. Ces informations peuvent être utilisées par le modèle pour modifier ou calibrer son comportement. Deuxièmement, il peut contrôler les périphériques de sortie - NetLogo pourrait contrôler les moteurs, les jouets, les voitures télécommandées, les appareils électriques, les ampoules et les équipements de laboratoire automatisés [99] .

Quelques primitives de cette extension [99] :

gogo-open , gogo-close, gogo-ports, output-port-off… **.**

V.7. Mise en œuvre de notre extension

Dans les sections précédentes, nous avons vus comment créer une extension d'une façon formelle dans ce qui suit nous allons expliquer comment et pourquoi et qu'elle est le principe de notre extension.

V.7.1. Objectif de création

Les réseaux sont actuellement un domaine de recherche très actif dans la communauté de modélisation basée sur les agents. Les modèles de travail en réseau sont déjà possibles dans NetLogo, mais nous voulons les rendre plus faciles à construire, notamment en facilitant l'exploitation des capacités des outils d'analyse et de visualisation des réseaux existants. Et afin d'adapter NetLogo aux simulations des réseaux nous avons développé des primitives facilitant son utilisation dans ce domaine.

V.7.2. Etapes de création avec l'IDE Eclipse

1- Ajouter le NetLogo 6.1.1. jar à la bibliothèque de l'Eclipse.
2- Création de la classe Manager et les classes des primitives (comme cité auparavant).
3- Création du fichier Manifest.
4- Compilation des classes créées avec la ligne de commande.
5- Création du fichier jar qui porte le même nom que l'extension et le mettre dans un dossier.
6- Ajouter ce dossier portant le nom de l'extension au dossier des extensions de NetLogo.
7- Appeler cette extension sur NetLogo : extension [nom-extension].
8- Utiliser les primitives de l'extension dans NetLogo.

V.7.3. Représentation

Nous avons développé deux extensions Voisins et Table de routage ; **Voisins** contient une primitive pour vérifier si un nœud est un voisin pour le nœud appelant. **Table de routage** retourne une table sous forme de liste de listes contenant les nœuds voisins et quelque autre informations (nombre de sauts, numéro de séquence…etc.).

V.7.3.1. Extension Table de routage

Notre extension se compose de certaines primitives tels que :

TabledeRoutage:Table [List-voisins]

Signale une nouvelle table (liste de listes) avec le contenu de la liste donné en entrée. List-voisins doit contenir les voisins d'un nœud.

TabledeRoutage:Get [tab-voisins] item

Signale un élément avec le contenu de la table se trouvant dans l'index item. tab-voisins doit contenir table d'un nœud (crée avec la primitive Table). Item doit contenir un entier indiquant la position de l'élément voulu.

TabledeRoutage:Set [tab-voisins] item [list]

Modifie l'élément se trouvant dans l'index item de la table par un autre élément list. tab-voisins doit contenir table d'un nœud (crée avec Table). Item doit contenir un entier indiquant la position de l'élément a modifié. List doit contenir la valeur du nouvel élément.

TabledeRoutage:Remove [tab-voisins] item

Supprime un élément dont l'index item du contenu de la table. tab-voisins doit contenir table d'un nœud crée avec Table. Item doit contenir un entier indiquant la position de l'élément a supprimé.

TabledeRoutage:Find [tab-voisins] [list]

Retourne un booléen, soit à vrai si la table contient la liste ou faux si cette liste ne figure pas dedans. tab-voisins doit contenir table d'un nœud crée avec Table. List c'est la liste recherchée.

V.7.3.2. Extension Voisins

Cette extension contient la primitive :

Voisins:isNeighbors [radius, x1, y1, x2, y2]

Retourne vrai si un nœud est voisin du nœud appelant sinon elle retourne faux. Radius contient la valeur la zone de portée du nœud appelant. x1 et y1 c'est les coordonnés du nœud appelant et x2, y2 c'est les coordonnées d'un autre nœud.

V.7.4. Utilisation

V.7.4.1. Extension Table de routage

En général, tout ce que nous pouvons faire avec une table de routage d'un protocole dans NetLogo, nous pouvons simplement utiliser une liste de listes. Mais nous pouvons envisager d'utiliser les primitives d'extension Table de routage à la place pour des raisons de vitesse et simplicité d'utilisation. Les listes et les tableaux ont des caractéristiques de performances différentes, nous pouvons donc peut-être accélérer l'exécution de notre modèle en sélectionnant la structure de données appropriée. Pour la simulation d'un protocole de routage (AODV), il serait plus facile d'appeler la primitive Table pour générer la table de routage de chaque nœud du réseau que de la programmer sur NetLogo ce qui prend beaucoup plus de temps.

L'extension Table de routage permet à un utilisateur débutant de mieux s'adapter au simulateur NetLogo dans le domaine réseau grâce aux facilités qu'elle offre. Notre extension est développée de façon à combler les manques de ce simulateur dans le coté réseau, nous avons donc pensé à ajouter des commandes permettant de construire des modèles pour les réseaux (protocoles…) de la manière la plus aisée et simple. Nous avons créé des primitives pour la manipulation de cette table notamment : l'ajout, la mise à jour, la récupération et la recherche dans cette table.

Contrairement aux listes de NetLogo, cette table est «modifiable». Cela signifie que nous pouvons la modifier directement, plutôt que de construire une copie modifiée comme avec des listes. Si la table est utilisée à plusieurs endroits dans le code, toutes les modifications que nous apportons apparaîtront partout.

La figure ci-dessous illustre le résultat lors de l'utilisation des différentes primitives de cette extension.

```
observer> show TabledeRoutage:Table [1 2]
observer: [["Nd" "Nseq" "ProchSaut" "nbrsaut" "listPrecs"] [1 1 1 1 [1]] [2 2 2 1 [2]]]
observer> show TabledeRoutage:Get [[1 1 1 1 [1]] [2 2 2 1 [2]]] 1
observer: [2 2 2 1 [2]]
observer> show TabledeRoutage:Set [[1 1 1 1 [1]] [2 2 2 1 [2]]] 1 [3 3 3 1 [3]]
observer: [[1 1 1 1 [1]] [3 3 3 1 [3]]]
observer> show TabledeRoutage:Remove [[1 1 1 1 [1]] [2 2 2 1 [2]]] 0
observer: [<null> [2 2 2 1 [2]]]
observer> show TabledeRoutage:Find [[1 1 1 1 [1]] [2 2 2 1 [2]]] [1 1 1 1 [1]]
observer: true
observer> show TabledeRoutage:Find [[1 1 1 1 [1]] [2 2 2 1 [2]]] [4 4 4 1 [4]]
observer: false
```

Figure V. 5: Exemple d'utilisation de l'extension Table de routage.

V.7.4.2. Extension Voisins

En générale la découverte des voisins dans NetLogo se fait par la primitive In-radius ce qui nécessite la création d'une liste et le parcours de tous les nœuds du réseau ce qui demande beaucoup du temps et de ressources, tandis que des fois nous avons besoin de savoir seulement est ce que un nœud est voisin ou non. C'est pour cela que nous avons créé la primitive isNeighbors retournant un booléen à True dans le cas ou deux nœuds sont voisins False dans le cas contraire.

La figure ci-dessous illustre le résultat lors de l'utilisation de la primitive isNeighbors de cette extension.

```
observer> ask noeud 22 [let x xcor let y ycor ask noeud 45 [show  voisins:isNeighbors 7 xcor ycor x y  ]]
(noeud 45): true
observer> ask noeud 11 [let x xcor let y ycor ask noeud 45 [show  voisins:isNeighbors 7 xcor ycor x y  ]]
(noeud 45): false
```

Figure V. 6: Exemple d'utilisation de l'extension Voisins.

V.7.5. Discussion

Avec l'apparition du concept d'extension dans NetLogo, plusieurs utilisateurs avancés de ce simulateur ont pu créer des extensions dans divers domaines. Dans notre domaine de recherche, l'extension la plus proche de ce que nous voulons réaliser est l'extension Array qui donne un tableau à partir d'une liste permettant d'obtenir une meilleure performance par rapport aux listes ce qui englobe ainsi son objectif.

L'extension « Array » donne un tableau à une dimension suivant le même ordre de la liste donnée en entrée ce qui ne correspond pas aux tables de routage. Cette extension est utile lorsque nous avons besoin d'une collection de valeurs dont la taille est fixe. Elle permet un accès rapide ou une modification de n'importe quel élément d'un tableau en connaissant sa position.

Par contre notre extension table de routage est créé pour résoudre le problème de manque du module propre au domaine du réseau qui consiste a donné une table sous forme d'une liste de listes donc c'est presque un tableau a deux dimensions ce qui se rapproche a une table de routage réel. De plus, elle donne la possibilité de chercher ou de supprimer un élément dans la table en donnant sa position. Cette extension sert à tous les utilisateurs de NetLogo d'obtenir une table de routage avec le moyen le plus rapide.

Conclusion

Dans ce chapitre nous avons expliqué la démarche que nous avons suivie pour résoudre notre problème qui consiste à adapter le simulateur NetLogo à un simulateur réseau. NetLogo est devenue récemment extensible, ce qui donne la possibilité aux utilisateurs de l'étendre et de le modifier à partir d'un code externe .C'est cette notion d'extensibilité qui nous a permis de réaliser le travail demandé. Pour cela nous avons expliqué dans l'ensemble de ce chapitre la notion d'extension en donnant une vue générale sur les différents étapes à suivre pour sa création, son utilisation, son intérêt ainsi les conseils de développements pour son écriture.

Par la suite, nous avons montré une représentation de nos deux extensions. La première consiste en un ensemble de primitives pour construire une table de routage à partir d'une liste de voisins et la maintenir lorsque si nécessaire. Et la deuxième contient une primitive de vérification de voisin existant dans la zone de portée.

Après avoir mené notre travail de recherche pour adapter NetLogo au réseau, nous avons pu ressortir quelques points que nous aimerons partager avec d'autres utilisateurs de ce simulateur qui sont : la consultation des modèles de simulation existant sur la bibliothèque des modèles de NetLogo afin d'ajouter un modèle dans le domaine réseau dans le but d'offrir plus d'exemples de ce domaine qui pourrait être utile pour des chercheurs réseau. De plus, il est important de consulter les spécifications de l'API NetLogo pour plus de détails sur ses classes, interfaces et méthodes et pour voir la méthode qui permet de développer une extension. Et ainsi de consulter le code source du NetLogo sur GitHub, le télécharger et comprendre son enchainement afin de pouvoir effectué des modifications d'une façon à ajouter des commandes et des primitives qui servent le domaine du réseau.

Conclusion Générale

Conclusion Générale

Aujourd'hui, les réseaux sans fil sont en plein essor et ont atteint un niveau d'utilisation élevé. La demande de mobilité a stimulé le développement de la technologie sans fil dans deux catégories : les réseaux avec ou sans infrastructure (appelé les réseaux ad hoc).

Les réseaux Ad hoc ont pris une part intéressante dans les recherches scientifiques pour les utiliser dans des secteurs aussi stratégiques que l'industrie, l'agriculture et le domaine militaire, ces réseaux n'exigent aucune infrastructure ou administration centralisée pour leurs déploiement, ils sont caractérisés par une très grande taille, fortement hétérogènes en termes de technologie de communication, protocoles, services et ils sont très dynamiques en raison des changements continuels de la topologie, cette technologie Ad hoc est déployée sur différentes structures matériels comme dans les réseaux véhiculaires qui portent le nom de VANETs, les réseaux mobiles MANETs et les réseaux de capteurs sans fil WSN…etc.

Dans le monde réel, la simulation du réseau est la méthodologie la plus utile et la plus courante pour évaluer différents réseaux, protocole …etc, avant de les utilisés. On trouve plusieurs simulateurs réseaux chacun prouvant ses performances par rapport à l'autre, malgré ceux leurs utilisations reste difficile. Plusieurs autres simulateurs simple et facile servent d'outil de simulation pour les réseaux dont nous pouvons citer le simulateur **NetLogo** qui a prouvé sa place par ses divers avantages.

L'objectif de ce mémoire était d'adapter ce simulateur multi-agents (NetLogo) qu'est une plateforme de simulation dans divers domaines à un simulateur réseau, pour le rendre : un des simulateurs les plus utilisés par les chercheurs réseaux, facile à utiliser et à manipuler.

La contribution dans ce travail consisté donc, à répondre à cette question, déclinée en deux parties :

D'abords, nous avons commencé par la simulation d'un protocole de routage réactif le plus populaire des réseaux Ad hoc AODV. Ensuite, nous avons essayé d'ajouter des extensions à ce simulateur dans le domaine réseau pour pouvoir atteindre notre but initial.

Ce mémoire était structuré en cinq chapitres :

Le **premier chapitre** met en évidence l'intérêt des SMA et défini les concepts clés utilisés dans ce travail. L'objectif du **chapitre deux** est de donner un aperçu générale sur un des types des réseaux sans fil sans infrastructure dit ad hoc, cette phase nous a permet ainsi de bien connaitre les principales protocoles de routage et nous avons met l'accent sur le protocole AODV qui va être utilisé dans les chapitres suivants. Dans le **chapitre trois,** nous avons parlé sur la notion de simulation ainsi qu'une étude comparative entre les différents simulateurs les plus utilisées dans le domaine réseau. En se basant sur cette étude NetLogo a été employé par la suite pour simuler le protocole de routage dont

nous avons met l'accent auparavant c'est le sujet de **chapitre quatre**. Enfin une extension est ajoutée à ce simulateur pour l'adapter au réseau dans le **chapitre cinq**.

Après toute cette étude nous avons pu approfondir nos connaissances dans le domaine du réseau sans fil, voir la relation entre les SMA et les réseaux ad hoc, apprendre à utiliser NetLogo et pouvoir modifier le fonctionnement d'un simulateur (l'ajout d'une extension).

En termes de perspective, nous envisageons d'ajouter d'autres extensions pour donner à l'utilisateur plus de fonctionnalités, l'ajout d'autres modèles des simulations des réseaux complexes et des protocoles de routage à base d'agent à la bibliothèque de NetLogo.

Bibliographie

Bibliographie

[1] J.FERBER, «Les systèmes multi-agents, vers une intelligence artificiel,» Inter Editions, 1995.

[2] TW.Baudouin, «Modélisation des systèmes d'élevage et simulation multi-agents d'une épidémie animale en milieu rurale,» Mémoire de Master , Université de YAOUNDE I, 2011.

[3] R.COURDER, «Systèmes Multi-Agents,» chez support du cours:Partie2 Agents et Systèmes Multi-Agents, Université de la Réunion, p. 1–58.

[4] N.BOUKHECHEM, «Routage dans les réseaux mobile Ad hoc par approche à base d'agents,» Mémoire Présenté en vue de l'obtention du diplôme de Magister, Université de Constantine, 2008.

[5] S.CHOUHA&S.ZEROUALA, «Adaptation de protocole de routage AODV dans un réseau de robots,» projet de fin d'études pour l'obtention du diplôme d'ingénieur d'état en informatique, Université de Batna, 2010.

[6] C.BOURJOIT, «Système multi-agents : Modélisation et simulation informatique de comportement collectifs, Chapitre II : Les différents concepts et composants d'un SMA,» Master M2 Sciences Cognitives, Université de Nancy2.

[7] K.BENHAMZA, «Conception d'un système multi -agents adaptatif pour la résolution de problème,» Thèse en vue de l'obtention du diplôme de Doctorat en Sciences, Université de BadjiMokhtar-Annaba, 2016.

[8] N. KABACHI, «Intelligence Artificielle Distribuée Et Systèmes Multi-Agents », support de cours : extrait de mémoire de thèse de doctorat, partie état de l'art,pp.33,2008.

[9] S.DJILALI, «La contribution des systèmes Multi-Agent dans la technologie du datamining distribue,» Mémoire en vue de l'obtention du diplôme de Magister, Université des Sciences et de la Technologie d'Oran, 2016.

[10] T.GUYET, «Systèmes multi agents (SMA) introduction et applications biomédicales,» cours IMTC, 2007.

[11] D.DAVID, «Théorie des SMA,» M1 STIC, 2010.

[12] C.SIBERTIN-BLANC&C.HANACHI, «Introduction aux systèmes Multi-Agents Objectifs,» Université Toulouse I & IRIT.

[13] A.ELFAZZIKI&A.NEJEOUI&M.SADGAL, «Une Approche multi-agents pour la modélisation et l'optimisation des Systèmes de gestion de transport maritime,» Thèse département d'informatique FSSM Marrakech, Maroc, 2005.

[14] S.ALAM, «Approche multi agents pour contrôler l'inondation dans un réseau de capteurs,» Mémoire de fin d'étude pour l'obtention du diplôme d'ingénieur d'état en informatique, Ecole nationale supérieure d'informatique, Oued-Smar, Alger, 2009.

[16] J.FERBER, «Coopération réactive et émergence,» Revue de l'association pour la recherche cognitif pp-35, 1994.

[17] MP.GLEIZES, «Vers la résolution de problèmes par émergence,» habilitation à diriger des recherches de l'Université Paul Sabatier, 2004.

[18] C.BERNON, «Émergence fonctionnelle par auto-organisation dans les systèmes artificiels,» Institut de Recherche en Informatique de Toulouse, 2007.

[19] B.MARZOUGUI, «Contribution à la modélisation et à la vérification des systèmes multi a agents,» Thèse de Doctorat, École Doctorale Informatique Télécommunications et Électronique(Paris), 2014.

[21] B.BENNAMER&F.KRIEF, «La technologie agent et les réseaux sans fil,» Laboratoire LIPN, Université du Paris 13.

[22] G.BEURIER, «Codage indirect de la forme dans les Systèmes Multi-Agents,» Thèse présentée pour obtenir le diplôme de Doctorat, Université des Sciences et Techniques du Languedoc, 2007.

[23] M.TAGHIKHEIRABADI&H.MOHAMMEDI, «MARP: A Multi-Agent Routing Protocol for Ad-hoc Network,» conférence internationale sur les communications et les réseaux en Chine (CHINACOM) pp.234-238, 2007.

[25] S.ATHMANI, «Protocole de sécurité pour les Réseaux de capteurs sans Fil,» Mémoire de Magister, Université Hadj Lakhder - Batna, 2010.

[26] A.BOUDJAADAR, «Plateforme basée Agents pour l'aide à la conception et la simulation des réseaux de capteurs sans fil,» Thèse de Magister, Université de Skikda, 2010.

[27] A.BOUTHAINA&Z.HEMAIZIA, «Un protocole de routage optimisé dans les réseaux Ad Hoc,» Mémoire de Master, Université de Tébessa, 2016.

[28] H.SAIDA&A.BERRABAH, «Balancement de charges dans les réseaux Ad Hoc,» Mémoire de Master, Université Abou Bakr Belkaid- Tlemcen, 2013.

[29] V.GAYRAND&L.NUAYMI&all, «La Sécurité dans les Réseaux Sans Fil Ad Hoc,» ENST Bretagne.

[30] BS.HAGGAR, «Auto-organisation et routage dans les réseaux mobiles Ad Hoc,» pour l'obtention du grade de Docteur, Université de Reims Champagne-Ardenne.

[31] N.BOUKHECHEM, «ROUTAGE DANS LES RESEAUX MOBILES Ad Hoc PAR UNE APPROCHE A BASE D'AGENT,» Mémoire de Magister, Université de Constantine, 2008.

[32] N.HARRAG, «Optimisation des paramètres d'un réseau Ad Hoc par algorithmes génétiques,» Memoire de Magister, Université Ferhat ABBAS – Sétif, 2011.

[33] T.LEMLOUMA, «Le routage dans les réseaux mobiles Ad Hoc,» Mini projet, Université d'Alger USTHB, 2000.

[34] H.BOUKHALFA&N.MOUICI, «L'impact des attaques sur la fiabilité du routage dans les réseaux Ad Hoc,» Mémoire de MASTER, Université de Tébessa, 2016.

[35] OS.OUBBATI, «Proposition et Simulation d'un Algorithme de partage de Ressources dans les Manets Basé sur l'Algorithme de Naimi et Tréhel,» Mémoire de Master , Université Ammar Telidji-Laghouat, 2011.

[36] JM.PERCHER&B.JOUGA, «Détection D'intrusions dans les réseaux Ad Hoc,» Ecole Supérieure d'Electronique de l'Ouest (ESEO).

[37] S.BITAM, «Une nouvelle approche pour la découverte de la topologie dans les réseaux mobiles ad hoc inspirée de la communication dans les essaims d'abeilles,» Thèse de Doctorat, Université Mohamed Khider de Biskra, 2011.

[38] F.THEOLEYRE, «Une auto-organisation et ses applications dans les réseaux ad hoc et hybrides,» Thèse de Doctorat, Institut National Des Sciences Appliquées de Lyon, 2006.

[39] F.KETTOUCHE&H.LATROCHE, «Protocole de routage multi-chemins EAOMDV avec consommation d'énergie dans les réseaux sans fil Ad Hoc,» Mémoire de Master, Université Abdelhamid Ibn Badis-Mostaganem, 2013.

[40] D.RAFFO, «Schémas de sécurité pour le protocole OLSR pour les réseaux AD Hoc,» Thèse de Doctorat, Université Paris 6 UPMC, 2005.

[41] B.TAVLI&W.HEINZELMAN, «Mobile Ad Hoc Networks: Energy-Efficient Real-Time Data Communications,» a book published by Springer, University of Rochester-NY-USA, 2006.

[42] R.MERAIHI, «Gestion de la qualité de service et contrôle de topologie dans les réseaux Ad Hoc,» Thèse de doctorat, École nationale supérieure des télécommunications (Paris), 2004.

[43] D.MONDONGA, «Etude sur les protocoles de routage d'un réseau sans fil en mode Ad Hoc et leurs impacts cas de protocoles OLSR et AODV,» pour l'obtention du titre d'ingénieur Informaticien, Institut supérieur de l'informatique, Programmation et Analyse.

[44] J.CARSIQUE&N.DAUJEARD&A.LALLEMAND&R.LADJADJ,«Le routage dans les réseaux mobiles Ad hoc,» 2003.

[45] G.ROMAN, «Modèle OSI et Routage,» Révision, 2002.

[46] C.PUTTAMADAPPA&SK.SARKAR&TG.BASAVARAJU,«Ad Hoc Mobile Wireless Networks,» Second Edition, Taylor&Francis Group-New York, 2013.

[47] G.BENREDJEM, «Le routage avec QOS dans les Réseaux Ad Hoc,» Université Badji Mokhtar-Annaba.

[48] P.MÜHLETHALER&P.HIPERCOM, «Routage dans les réseaux ad hoc,» vol.33,n.ref.article : te7520,pp. 1–7, Europe, 2002.

[49] G.Jayakumar&G.Gopinath, «Ad Hoc Mobile Wireless Networks Routing Protocols-A Review,» Journal of Computer Science,Vol. 3, No. 8, pp. 574-582, 2007.

[50] M.Mauve&J.Widmer, «A Survey on Position-Based Routing in Mobile Ad Hoc Networks,» IEEE Network,Vol.15,pp. 30-39, 2001.

[51] O.BOUDAA, «Conception et réalisation d'un protocole de routage pour les réseaux de capteurs sans fil,» Mémoire de Master en Informatique, Université Abderrahmane Mira – Béjaïa.

[52] M.Frikha, «Réseaux ad hoc,routage,qualité de service et optimisation,» Lavoisier , 2010.

[53] S.BELKHEY, «Etude d'un protocole de routage basé sur les colonies de Fourmis dans les réseaux de capteurs sans fil,» Mémoire de fin d'études Master,Option: Modèle Intelligent et Décision (M.I.D), Université de Telmcen, 2013.

[54] A.ABDESSELAM&M.BELOUATEK, «Conception d'un algorithme de routage basé sur l'heuristique du recuit simulé pour les réseaux de capteurs à grande échelle Réalisé,» Mémoire de fin d'études, Université de Telmcen, 2012–2013.

[55] J.Kaiser&C.Liu, «A Survey of Mobile Ad Hoc network Routing Protocols,» Technical Report, University of Ulm-Germany, 2003.

[57] S.ALLOUACHE&R.TAYEBCHERIF, «Etude Comparative entre AODV et AOMDV dans le Cadre des Réseaux Ad Hoc,» Mémoire de Master, Université Abderrahmane Mira-Béjaia, 2017.

[58] N.MOGHIM&F.HENDESSI, «Ad Hoc Wireless Network Routing Protocols and improved AODV,» The Arabian Journal for Science and Engineering,Vol.28,No.2C,pp.99-114, 2003.

[59] CE.PERKINS&P.BHAGWAT, «Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers,» of the conference on Communications architectures, protocols and applications (SIGCOMM'94), pages 244–254. ACM, University of Maryland, 1994.

[60] T.Clausen&P.Jacquet, «Optimized Link State Routing Protocol OLSR,» Project Hipercom,INRIA,RFC3626, 2003.

[61] MA.AYAchi, «Contributions à la détection des comportements malhonnêtes dans les réseaux ad hoc AODV par analyse de la confiance implicite,» Thèse de doctorat, Université de Rennes 1, 2011 .

[62] D.Bertsekas&R.Gallager,«Data networks,» Prentice-hall Englewood Cliffs, NJ, Massachusetts Institue of Technology, 1987.

[63] A.Derhab&T.Lemlouma&N.Badache&D.Djenour, «Les Protocoles de routage dans les réseaux mobiles Ad Hoc,» Revue d'Information Scientifique et Technique pp.77–112., 2004.

[64] DB.Johnson&DA.Maltz, «Dynamic source routing in ad hoc wireless networks,» In Mobile Computing, chap. 5,Kluwer Academic Publishers, pp. 153–181, 1996.

[65] C.Perkins&E.Belding-Royer&S.Das, «Ad Hoc On-Demand Distance Vector (AODV) Routing,» RFC3561, 2003.

[66] ZJ.Hass&R.Pearlman, «Zone routing protocol for ad-hoc networks,» Internet Draft, work in progress, 1999.

[67] M.Abolhasan&T.Wysocki&E.Dutkiewicz, «A review of routing protocols for mobile ad hoc networks,» Ad Hoc Networks 2, pp. 1-22, 2004.

[68] B.Abdelkrim, «Impact des modèles de mobilités sur les performances des protocoles de routage en milieu urbain réaliste dans les réseaux VANET (V2V),» 2015.

[69] L.NIAR, «Analyse Graphique pour la surveillance dans ce réseau de capteurs sans fil(RCSF) Simulateur : OMNet++,» Mémoire de Magister, Spécialité : Informatique, Université d'Oran, 2012.

[70] S.FOURNIER, «Intégration de la dimension spatiale au sein d'un modèle multi-agents à base de rôles pour la simulation : Application à la navigation maritime,» pour obtenir le grade de Docteur, Université de Rennes 1, 2005.

[71] A.KUMAR&all, «Simulators for Wireless Networks : A Comparative Study,» International Conference on Computing Sciences, 2012.

[72] F.MICHEL, «Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents,» Thèse pour obtenir le diplôme de doctorat, Université Montpellier II, 2004.

[73] P.BOMMEL, «Définition d'un cadre méthodologique pour conception de modèles multi-agents adaptée à la gestion des ressources renouvelables,» Thèse pour obtenir le grade de docteur, Université Montpellier II, 2009.

[74] B.BELATTAR, «Simulation et modélisation des systèmes,» support de cours, Université de Batna, 2000.

[75] M.BAKNI&all, «Methodology to Evaluate WSN Simulators : Focusing on Energy Consumption Awareness,» Conference on Wireless &Mobile Network, ESTIA Institue of Technology(France), 2019.

[76] J.LESSMANN&all, «Comparative Study of Wireless Network Simulators,» Seventh International Conference on Networking, University of Paderborn, 2015.

[77] M.JEVETIC&N.Zogovic&all, «Evaluation of Wireless Sensor Network Simulators,» Conference in Institute Mihajlo Pupin, 2009.

[78] A.KHAN&B.SARDAR&M.OTHMAN, «A Performance Comparison of Network Simulators for Wireless Networks,» Faculty of Computer Science & Information Technology, University of Malaya.

[79] C.MALLANDA&all, «Simulating Wireless Sensor Networks with OMNeT ++,» The University of Akron.

[80] T.PARIS&L.CIARLETTA&V.CHEVRIER,«Co-simulation à base d'outils multi-agents: un cas d'étude avec NetLogo,» 26èmes Journées Francophones sur les Systèmes Multi-Agents, France, 2018.

[81] V.ROSSET, «WSimulNET-A Model for Simulating Large Wireless Networks,» Short Guide-V.6.26-Basic, University of São Paulo(Brazil), 2018.

[82] M.BABIS&P.MAGULA, «NetLogo – an alternative way of simulating mobile ad hoc networks,» Slovak University of Technology in Bratislava.

[83] S.TALIBHASSON&Z.YASEEN, «Simulating Road Modeling Approach's in Vanet Environment Using NetLogo,» College of Information Technology, University of Babylon, 2016.

[84] M.BELLAHCENE&M.BOUAZZA, «Une version améliorée du protocole GPSR pour un routage efficace dans les Vanets,» Mémoire de Master, Université de Tlemcen, 2016.

[87] M.PIRON, «Modélisation et simulation de systèmes complexes,» Diapos du Séminaire du Groupe de Recherches en Analyse de l'Information Territoriale, 2007.

[88] R.Jouffroy&G.Khélifi&M.Fontaine&all, «Apport de la simulation pour la prise en charge des urgences vitales,» Article, Université Paris Descartes-Paris5.

[90] S.Tisue&U.Wilensky, «NetLogo: Design and implementation of a multi-agent modeling environment,» Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence, Chicago IL, 2004.

Références webographique

[15] https://fr.wikipedia.org/wiki/Stigmergie, (Consulté le 07/07/2020).

[20] http://www.damas.ift.ulaval.ca/~coursMAS/ComplementsH10/Agents-SMA.pdf,(Consulté le 07/07/2020).

[24] https://fr.wikipedia.org/wiki/R%C3%A9seau_sans_fil,(Consulté le 09/07/2020).

[56] https://www.memoireonline.com/01/09/1878/m_Les-technologies-sans-fil-Le-routage-dans-les-reseaux-ad-hoc-OLSR-et-AODV3.html, (Consulté le 10/07/2020).

[85] https://fr.wikipedia.org/wiki/IEEE_802.11b,(Consulté le 09/07/2020).

[86] https://fr.wikipedia.org/wiki/IEEE_802.15.4,(Consulté le 09/07/2020).

[89] https://ccl.northwestern.edu/netlogo/docs/,(Consulté le 11/07/2020).

[91] https://ccl.northwestern.edu/netlogo/docs/extensions.html ,(Consulté le 11/08/2020).

[92] https://romainmejean.fr/manuel_netlogo/extensions.html ,(Consulté le 12/08/2020).

[93] https://fr.wikipedia.org/wiki/Java_(langage),(Consulté le 11/08/2020).

[94] https://fr.wikipedia.org/wiki/Scala_(langage),(Consulté le 11/08/2020).

[95] https://github.com/NetLogo/NetLogo/wiki/Extensions-API?fbclid=IwAR3JcrYbcAxrloFM_lagAE_nv3ilCkNi9hpQNRssvhXMt35Fg9Owq3olhPg, (Consulté le 15/08/2020).

[96] https://github.com/NetLogo/NW-Extension, ,(Consulté le 19/08/2020).

[97] https://www.01net.com/telecharger/windows/Programmation/creation/fiches/131502.html, (Consulté le 17/08/2020).

[98] https://github.com/NetLogo/Arduino-Extension, ,(Consulté le 19/08/2020).

[99] https://ccl.northwestern.edu/netlogo/3.0/docs/gogo.html, ,(Consulté le 19/08/2020).

Annexes

Table de routage des nœuds

```
(noeud 0): [[20 20 20 1 [0]] [22 22 22 1 [0]] [17 17 17 1 [0]] [11 12 11 1
[0]]]
(noeud 1): [[28 29 28 1 [1]] [18 19 18 1 [1]] [10 11 10 1 [1]]]
(noeud 2): [[8 9 8 1 [2]]]
(noeud 3): [[23 24 23 1 [3]] [13 13 13 1 [3]] [6 6 6 1 [3]]]
(noeud 4): [[18 18 18 1 [4]] [23 23 23 1 [4]] [19 20 19 1 [4]] [29 30 29 1
[4]] [15 16 15 1 [4]]]
(noeud 5): [[9 10 9 1 [5]]]
(noeud 6): [[13 13 13 1 [6]] [3 4 3 1 [6]]]
(noeud 7): [[8 8 8 1 [7]] [16 17 16 1 [7]]]
(noeud 8): [[2 2 2 1 [8]] [16 17 16 1 [8]] [7 8 7 1 [8]]]
(noeud 9): [[10 11 10 1 [9]] [5 5 5 1 [9]] [18 19 18 1 [9]]]
(noeud 10): [[18 18 18 1 [10]] [9 9 9 1 [10]] [19 20 19 1 [10]] [1 1 1 1
[10]]]
(noeud 11): [[25 25 25 1 [11]] [0 0 0 1 [11]] [14 14 14 1 [11]] [20 20 20 1
[11]] [27 28 0 5 [0 17 15 21]]]
(noeud 12): []
(noeud 13): [[3 4 3 1 [13]] [6 7 6 1 [13]]]
(noeud 14): [[11 12 11 1 [14]]]
(noeud 15): [[29 29 29 1 [15]] [17 18 17 1 [15]] [21 21 21 1 [15]] [19 19 19 1
[15]] [4 4 4 1 [15]] [22 23 22 1 [15]]]
(noeud 16): [[7 7 7 1 [16]] [8 8 8 1 [16]] [26 27 26 1 [16]]]
(noeud 17): [[0 0 0 1 [17]] [15 15 15 1 [17]] [22 22 22 1 [17]]]
(noeud 18): [[9 9 9 1 [18]] [28 28 28 1 [18]] [1 1 1 1 [18]] [4 5 4 1 [18]]
[10 11 10 1 [18]] [19 20 19 1 [18]]]
(noeud 19): [[10 10 10 1 [19]] [18 18 18 1 [19]] [4 4 4 1 [19]] [15 16 15 1
[19]] [22 23 22 1 [19]]]
(noeud 20): [[0 0 0 1 [20]] [25 25 25 1 [20]] [11 12 11 1 [20]]]
(noeud 21): [[15 16 15 1 [21]] [26 26 26 1 [21]] [27 27 27 1 [21]] [29 29 29 1
[21]]]
(noeud 22): [[0 0 0 1 [22]] [19 19 19 1 [22]] [15 15 15 1 [22]] [17 18 17 1
[22]]]
(noeud 23): [[4 5 4 1 [23]] [3 3 3 1 [23]]]
(noeud 24): []
(noeud 25): [[11 12 11 1 [25]] [20 20 20 1 [25]]]
(noeud 26): [[21 22 21 1 [26]] [27 27 27 1 [26]] [16 16 16 1 [26]]]
(noeud 27): [[26 27 26 1 [27]] [21 22 21 1 [27]]]
(noeud 28): [[1 1 1 1 [28]] [18 19 18 1 [28]]]
(noeud 29): [[15 16 15 1 [29]] [21 22 21 1 [29]] [4 4 4 1 [29]]]
```

Figure 1 : Table de routage des nœuds du notre simulation enregistré dans fichier word.

```
////////////////////////////////////////////////////classe manager//////////////////////////////////////////////////////
public class TabledeRoutage extends DefaultClassManager {
     public void load(PrimitiveManager primitiveManager) {//ajout des primitives
          primitiveManager.addPrimitive("Table", new Addtable());
          primitiveManager.addPrimitive("Remove", new  Removetable());
          primitiveManager.addPrimitive("Set", new Settable());
          primitiveManager.addPrimitive("Get", new Gettable());
          primitiveManager.addPrimitive("Find", new Findtable());

        }
```

Figure 2 : Classe Manager de l'extension Table de Routage.

```
////////////////////////////////////////////////////les classes des primitives//////////////////////////////////////////////////////
     public class Addtable implements Reporter {// Reporter => interface implémenté par la classe addtable

     public Syntax getSyntax() {
     //c'est une méthode de l'interface reporter, elle permet de faire passé les arguments de la primitives
         return Syntax.reporterSyntax
             (new int[]{Syntax.ListType()}, Syntax.ListType());//le types de arguments
               }

      public Object report(Argument args[], Context context)
      //c'est une méthode de l'interface reporter, elle contient le code de la primitive
      throws ExtensionException , LogoException {//extentionexpetion elle indique l'erreur
      LogoListBuilder list = new LogoListBuilder();// créé de liste pour les résultat de NetLogo, donc la liste passé en résultat
      LogoListBuilder list1 = new LogoListBuilder();
      LogoListBuilder list2 = new LogoListBuilder();
      LogoListBuilder list3 = new LogoListBuilder();

   ////////////////////////////// Récupération des entrées//////////////////////////////////////////////
         LogoList listes = new LogoList(null);listes =args[0].getList();//recuppere la liste donné en entré
         int liste;liste = listes.size();// retourne la taille de la liste donné en entrée

   //////////////les informations de la premiere case de la table de routage//////////////////////
                  list2.add("Nd");
                  list2.add("Nseq");
                  list2.add("ProchSaut");
                  list2.add("nbrsaut");
                  list2.add("listPrecs");
                  list.add(list2.toLogoList());
```

Figure 3 : La classe de la primitive addTable d'extension Table de Routage.

```
import org.nlogo.api.*;

public class Voisins implements Reporter{//Reporter => interface implémenté par la classe voisin
static public int sqr(int a) {//permet de claculer le carré d'un entier
    return a*a;
}
 static public  int distance(int x1, int y1, int x2,int y2) {//permet de calculer la distance entre 2 point
    return (int) Math.sqrt(sqr(y2 - y1) + sqr(x2 - x1));
}
@Override
 public Syntax getSyntax() {
    //c'est une méthode de l'interface reporter, elle permet de faire passé les arguments de la primitives
    return SyntaxJ.reporterSyntax
  (new int []{Syntax.NumberType(),Syntax.NumberType(), Syntax.NumberType(), Syntax.NumberType() , Syntax.NumberType()}
  ,Syntax.BooleanType());//le types de arguments
  }
 @Override
 public Object report(Argument[] args, Context context)//c'est une méthode de l'interface reporter elle contient le code de la primi
 throws ExtensionException {//extentionexpetion elle indique l'erreur
//creation des variable pour les entrées
 int raduis;    ////pour la récuppération de la portée radius
 int x; int y; //pour la récuppération des coordonnées de 1er agent
 Boolean reslt = false;
    try {
         raduis = args [0].getIntValue();
         x = args[1].getIntValue();;
         y = args[2].getIntValue();
      }
      catch(LogoException e) {
        throw new ExtensionException( e.getMessage() ) ;
      }
    if (raduis < 0) {// donne le cas ou la portée est négative
        throw new ExtensionException
          ("input must be positive");
    }
```

Figure 4 : La classe de la primitive d'extension Voisins.

Résumé

La technologie sans fil progresse rapidement et de nouvelles améliorations sont proposées régulièrement avec l'apparition de divers types de réseaux, notamment les réseaux sans infrastructure dit Ad hoc. Dans les réseaux Ad hoc, de nouveaux protocoles non testés ne peuvent être lancés à grande échelle en raison de l'incertitude quant à leur succès. D'où la nécessité d'une phase de simulation. La simulation du réseau est sans aucun doute l'une des méthodologies d'évaluation les plus prédominantes dans le domaine des réseaux informatiques. Elle est largement utilisée pour le développement de nouvelles architectures de communication et de protocoles de réseau. La variété des simulateurs du réseau présente de lourds inconvénients.

Les réseaux à grande échelle sont difficiles à simuler dans les simulateurs des réseaux traditionnels, car ils peuvent nécessiter des ressources de calcul élevées. De plus, en raison de leur complexité dans les détails, les simulateurs traditionnels n'offrent pas un environnement convivial pour les nouveaux étudiants, contrairement au simulateur NetLogo qui est un outil de simulation multi-agents facile à utiliser et il demande peu de temps pour apprendre son fonctionnement. Dans cette optique, notre travail est porté sur l'adaptation de ce simulateur à un simulateur réseau.

Mots clés : Réseaux, Ad hoc, Protocoles de routage, Multi-agents, Simulation, NetLogo.

Abstract

Wireless technology is advancing rapidly and new improvements are being proposed regularly with the emergence of various types of networks, including networks without infrastructure so-called ad hoc. In Ad-hoc networks, new untested protocols cannot be launched on a large scale because of the uncertainty as to their success. Hence the need for a simulation phase. Network simulation is undoubtedly one of the most predominant evaluation methodologies in the field of computer networks. It is widely used for the development of new communication architectures and network protocols. The variety of network simulators has serious disadvantages.

Large-scale networks are difficult to simulate in traditional network simulators because they can require high computational resources. In addition, because of their complexity in detail, traditional simulators do not provide a user-friendly environment for new students, unlike the NetLogo simulator, which is an easy to use multi-agent simulation tool and requires short time to learn how it works. With this in mind, our work is focused on adapting this simulator to a network simulator.

Keywords: Networks, Ad hoc, Routing protocols, Multi-agent, Simulation, NetLogo.