MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'Obtention du Diplôme de Master en Informatique

Option : **Réseaux et Systèmes**

Présenté par :

Cissé Moussa Almamy

THEME :

Algorithme de construction de structure virtuelle pour la gestion du routage dans MANET

Soutenu le : 19/06/2021

Devant le jury composé de :

BENHAMED S.	Grade	Université de Mostaganem	Président
BAHNES N.	Grade	Université de Mostaganem	Examinateur
BESSAOUD K.	Grade	Université de Mostaganem	Encadreur

Année Universitaire 2020-2021

Remerciements

Tout d'abord , je remercie Allah de m'avoir donné la force, le courage et la volonté de mener à bien ce travail.

Un grand merci à ma familles, surtout mes parents qui m'ont épaulées durant tout mon cursus universitaire et soutenues tout au long de ce projet.

A mes chères et fidèles amies qui ont toujours été présents pour moi quand j'ai eu besoin surtout durant ce projets.

Ensuite je remercie mon encadreur Dr. Karim Bessaoud, un professeur extraordinaire, pour tout le temps qu'il m'a consacré, pour ces précieux conseils et pour tout son aide, son appui et son soutient durant la réalisation de ce projet et durant mon cursus.

Aussi à tous les enseignants et employés du département d'informatique à qui je doit les connaissances acquis durant mon cursus universitaire.

En fin, je tiens aussi à remercier également tous les membres du jury pour avoir accepté d'évaluer mon travail.

Je dédie ce travail

A mes très chers parents, source de vie, d'amour et d'affection. Au symbole de douceur, de tendresse, d'amour et d'affection, ma mère pour son soutient incéssant. A mon père, pour les sacrifices qu'il a consentis pour mon éducation. Que ce travail soit le témoignage de l'amour que je les avoues. A toute ma famille, source d'espoir et de motivation. A mes chers frères et leurs enfants, source de joie et de bonheur. A la plus adorable de mes nièces "Aminata Cissé". A mes chers amis qui m'ont soutenu, encouragé et m'ont surtout permis de travailler dans une atmosphère joviale.

♡

Résumé

Un Mobile Ad hoc NETwork (MANET) est un type de réseau décentralisé sans infrastructure prédéfini composé de nœuds mobiles connectés entre eux par une liaison sans fil qui forme une topologie arbitraire. Cette absence d'infrastructure combinée à la mobilité des nœuds qui se déplacent librement et de manière hasardeux provoque des rupture très fréquente des liaisons de communication réduisant considérablement les performances générales du réseaux. Plusieurs approches visant à minimiser les impactes de l'absence d'infrastructure centrale d'administration et/ou de control et la mobilité dans les MANETs sur les performances du réseau ont été proposés dans la littérature scientifique. L'une de ces approches consiste à construire dans MANET une structure virtuelle pour gérer les taches complexes telles le routage, etc. Nous présentons dans ce mémoire un algorithme distributé de construction d'épine dorsale (ou backbone) comme structure virtuelle pour la gestion du routage dans MANET.

Mot-clés : MANET ; Modèle de mobilité, Ensemble dominant connecté (CDS) ; routage réseau ad hoc sans fil ;

Abstract

A Mobile Ad hoc NETwork (MANET) is a type of decentralized network without a predefined infrastructure composed of mobile nodes connected to each other by a wireless link that forms an arbitrary topology. This absence of infrastructure associated with the mobility of the nodes which move freely and in a hazardous manner causes very frequent breaks of the Communication links govern the general performance of the networks. Several approaches aiming to minimize the impacts of the absence of central administration and/or control infrastructure and mobility in MANETs on network performance have been proposed in the scientific literature. One of these approaches consists in building in MANET a virtual structure to manage complex tasks such as routing, etc. We present in this thesis a distributed backbone construction algorithm as a virtual structure for the management of routing in MANET.

Keywords : MANET ; Mobility model, Connected Dominanting Set (CDS) wireless ad hoc network routing ;

Table des figures

Liste des abbréviations

CDS	Connected Dominating Set
IS	Independent Set
MANET	Mobile Ad-hoc Network
MIS	Maximum independent Set
MM	Mobility Model
MWCDS	Minimum Weight Connected Dominating Set
LWCDS	Low Weight Connected Dominating Set

Table des matières

Introduction général

De nos jours, le marché des réseaux de télécommunication est envahi par les technologies. La communication entre utilisateurs , à n'importe quel moment et n'importe où (à la maison, au bureau, dans les voitures, aux conféréences, etc.) est plus que nécessaire.

Les réseaux filaires n'étant plus en mésure d'assurer les besoins de mobilités et d'autonomies des utilisateur, plusieux dispositifs portables de communication sans fil (ordinateur portable, téléphone mobiles, PDA, Tablette, etc.) on vu le jour. Parmi les technologie sans fil qui ont été rapidement adoptés par les utilisateurs dû à leur flexibilité d'utilisation, nous avons le bluetooth et le Wi-Fi.

Ces technologies permettent de deployer un réseau personnel qui interconnecte plusieurs périphériques de communication tels que l'imprimante, le clavier, le téléphone mobile, l'appareil photo, etc. On trouve également la technologie 802.11, plus connu sous le nom Wi-Fi (Wireless Fidélity), qui permet d'augmenter le débit de la connexion entre les équipements et d'utiliser l'internet dans différents lieu public tels que les universités, les entreprises, aéroports, etc.

Bien que les efforts de recherche et de développement consacrés aux réseaux sans fil traditionnels sont toujours considérables, l'intérêt de la communauté scientifique et industrielle dans le domaine des télécommunications s'est récemment orientée vers des scénarios plus complexes dans lesquels un groupe d'unités mobiles équipées par de cartes sans fil communiquent sans aucune infrastructure fixe préalable. PRNET (Packet Radio Network) [1] est le premier réseau qui ne repose sur aucune infrastructure ou liaison filaire, il a été construit par la Darpa (Defense Advanced Research Projects Agency) en 1987. C'est l'un des bases des réseaux Ad hoc actuels.

Les PRNETs ont connu plusieurs appellations au fil du temps. L'un de ses nom est MANET (**M**obile **A**d-hoc **NET**work). Les réseaux mobile ad-hoc sont constitués d'entité pouvant se dépalcement indépendamment les uns des autres et s'auto-organiser de sorte

que le réseau fonctionne sans aucune infrastructure fixe.

Malgré l'énorme avantage qu'offre les MANETs, ils ne restent pas moins sans problème, et l'un de ses principaux problèmes est la variation constante de la qualité des liens (radio) entre ses nœud, essentiellement due à leurs mouvements. Vue le réseau est mobile, la topologie change continuellement. Les nœuds peuvent apparaître et disparaître, ce qui produit un état imprevisible des liens radio. Rajouter à cela,les caractéristiques des MANETs comme la limitation d'énergie et de la bande passante, la gestion du routage multi-sauts est une tâche très complexe.Pour résoudre problèmes et/ou minimiser l'impacts de ses problèmes sur les performances du réseau, plusieurs approchés basées sur la prédiction de la qualité de lien, du mouvement ou du moment de rupture de lien ont été proposés.

L'objectif de ce mémoire est de proposer et d'évaluer un algorithme distribué de construction d'un ensemble dominant connexe comme structure virtuelle pour le routage dans un réseau MANET (Mobile Ad hoc NETworks). Construire un ensemble dominant connexe revient à choisir un certain nombre de nœuds pour jouer le rôle de dominants. Ce mémoire est organisé comme suit :

Dans le premier chapitre, nous ferons une brève présentation des réseaux mobiles adhoc (MANETs), ensuite nous aborderons le concept de mobilité qui est l'un des problèmes majeurs rencontrer fréquemment dans les MANETs, nous parlerons également de quelques modèles de mobilité utilisé dans ce type de réseau, et enfin nous parlerons du routage et de quelques protocoles de routages utilisés dans les MANETs.

Dans le second chapitre, nous ferons une étude des ensemble dominants connectés (CDS) et de quelques algorithmes de construction de CDS rencontrés fréquemment dans la littérature.

Dans le troisième chapitre nous feront la modélisation d'un simulateur de MANET, que nous allons concevoir afin de pouvoir tester et évaluer l'algorithme de construction de backbone que nous allons présenter dans le quatrième chapitre.

Dans le quatrième chapitre, nous allons décrire succinctement notre algorithme de construction de backbone et présenter brièvement notre simulateur et quelques unes de ses fonctionnalités avec des captures d'écran. Puis nous terminerons ce dernier chapitre par une évaluation des performances de notre solutions à travers une série de simulation.

Enfin nous terminerons ce mémoire par une conclusion général dans laquelle nous

présentons les acquis tirés à ce projet et les perspectives de notre travail.

Réseaux Mobile Ad-hoc (MANETs)

1. Introduction

Un réseau mobile ad-hoc ou MANET (**M**obile **A**d hoc **Net**work) est un type de réseau sans fil décentralisé qui ne s'appuie sur aucune infrastructure définie au préalable et dans lequel les nœuds sont totalement libre de leur mouvement. Le déplacement aléatoire et souvent instantané des nœuds d'un tel réseau fait qu'il est quasiment impossible d'avoir une topologie fixe ou prévisible dans ce type de réseau.

Les réseaux mobile ad-hoc sont facile à mettre en place tant par rapport à leur configuration que par rapport à leur coût et au delai de déploiment. Combiné à cela, l'absence d'oragane central de controle, ceci confère aux réseaux mobile ad-hoc le status de réseau le mieux adapter aux situations d'urgences telles que les catastrophes naturelles, les champs de batailles militaires, les missions de sauvetage, etc. Les réseaux mobile ad-hoc sont également utilisés dans les domaines de récherche.

Dans ce chapitre, nous ferons une brève présentation des réseaux mobiles ad-hoc (MANETs), ensuite nous aborderons le concept de mobilité qui est l'un des problèmes majeurs rencontrer fréquemment dans les MANETs, nous parlerons également de quelques modèles de mobilité utilisé dans ce type de réseau, et enfin nous parlerons du routage et de quelques protocoles de routages utilisés dans les MANETs et nous terminerons ce chapitre par une conclusion.

2. Réseaux mobiles Ad-hoc (MANETs)

La commnunication dans un MANET se fait en mode point à point ou en mode multi-sauts, une paire de nœuds ne peuvent communiquer que s'ils sont à partée l'un de l'autre. Les messages sont transmis soit en un seul saut si l'émetteur et le recepteur sont à portée l'un de l'autre (mode point à point), ou via des nœuds intermédiare dans le cas contraire (mode multi-sauts) [2].

2.1. Definition & Histoire

Opposé aux réseaux sans fil d'infrastructure où chaque nœud communique directement avec un point d'accès ou une station de base, un réseau mobile ad-hoc, ou MANET est un type de réseau ad-hoc sans fil qui ne s'appuie sur aucune infrastructure fixe ou organe central d'administration ou de controle pour ses opérations [3].

A l'origine, dans les années 1970, les réseaux mobile ad-hoc étaient appélés des PRNETs (Packet Radio Network). Ils ont par la suite évoulé pour devenir des SURAN (Survivable Adaptive Radio Network) [1].

L'essort dans les années 1990 des réseaux sans fil principalement dû à la mobilité, la simplicité d'installation, la topologie, le coût réduit et la fiabilité qu'offre ce type de réseau aux usagers à motivé la recherche de technologie émergente capable de s'adapter a des situations d'urgence telles que les champs de batailles millitaire. La Defense Advanced Research Projects Agency (DARPA)[1] qui cherchait le moyen d'optimiser les commnunications dans le domaine millitaire lance des programmes de recherches dans ses réseaux. Une illustration des réseaux ad-hoc avec infrastructure et des MANETs est donnée par la figure I.I.1.

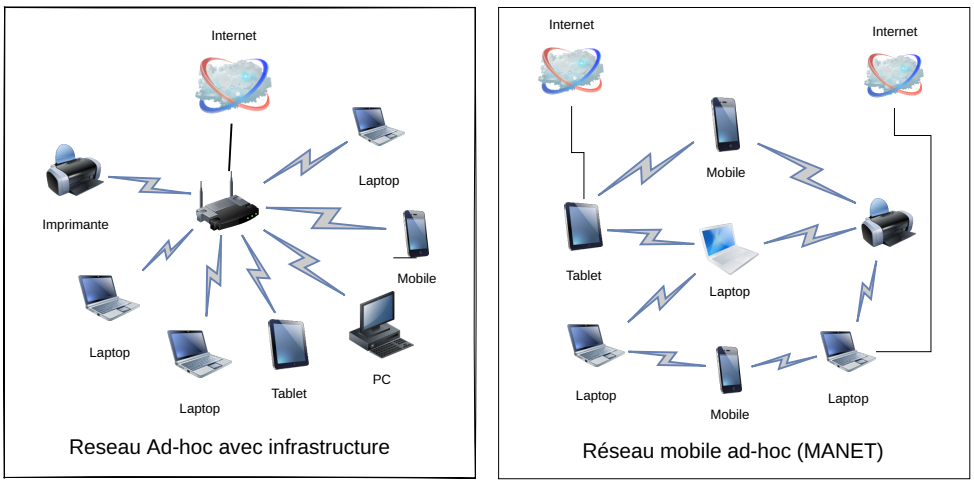


FIGURE I.1 – Topologie ad-hoc avec infrastructure et sans infrastructure (MANET)

2.2. Type de réseaux ad-hoc

Il existe divers type de réseaux ad-hoc, parmi lesquels, nous pouvons cité :

2.2.1. Les réseaux personnels (PAN)

Un réseau personnel (PAN) est un réseau domestiques restreint formés entre divers appareils mobile réliés entre entre eux de manière ad-hoc. C'est également un réseau autonome interconnectant divers appareils sans fil à la maison parmi lesquels, nous avons le Bleutooth, le Zigbee [1], l'infrarouge, etc.

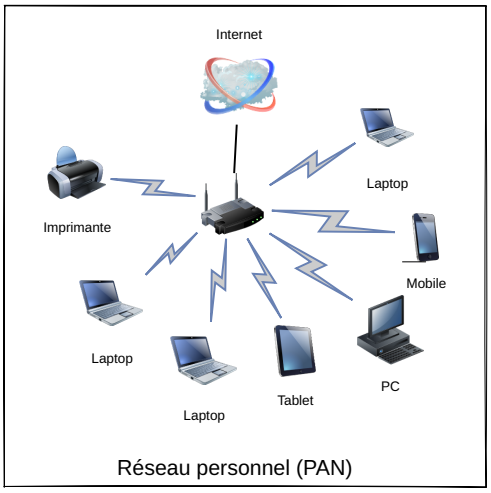


FIGURE I.2 – Réseau Personel (PAN)

2.2.2. Les réseaux de capteurs sans fil

Un réseau de capteurs sans fil est un réseau composé d'un très grand nombre de petits capteurs qui peuvent être utilisés pour capter des grandeurs physiques (chaleur, humidité, pression, toxines, pollution, etc.) d'une zone et de les transformés en grandeurs numériques, puis les envoyée à une unité de traitement informatiques ou de stockage de données via une transmission sans fil. Les capacités de chaque capteur sont très limités, et chacun doit s'appuyer sur d'autres afin de transmettre des données à un ordinateur central, d'où l'intérêt des réseaux ad-hoc.

2.2.3. Les réseaux véhiculaire (VANETs)

Vehicular Ad hoc Network (ou VANET), est une sous-classe des réseaux mobiles ad hoc (MANETs). Ces des réseaux qui n'ont également pas d'infrastructure fixe, dépendent des véhicules eux-mêmes pour fournir des fonctionnalités réseaux [4]. Cependant, Ces réseaux présentent plusieurs caractéristiques qui sont totalement différents de nombreux MANET

1. ZigBee est un protocole de haut niveau permettant la communication d'équipements personnels ou domestiques équipés de petits émetteurs radios à faible consommation ; il est basé sur la norme IEEE 802.15.4 pour les réseaux à dimension personnelle (Wireless Personal Area Networks : WPAN)

génériques, principalement dû aux contraintes de mobilités et aux comportements du conducteur.

2.3. Mode de communication dans les réseaux Ad Hoc

Dans les commnunications réseaux nous distinguons trois mode de commnunications :

- **Communication point à point (Unicast) :** Une seule paire de nœud (émetteur et récepteur) est concerner par ce mode de communication.
- **Communication multipoints (Multicast) :** Ce mode de communication permet à un nœud (source) d'envoyé un messages à plusieurs autres nœuds du réseaux.
- **Diffusion (Broadcast) :** Ce mode de communication permet permet à un nœud (source) d'envoyé un messages à tous les autres nœuds du réseaux.

La figure I.I.3 donne une illustration de ses trois modes de communications. Le nœud **A** est le nœud source.

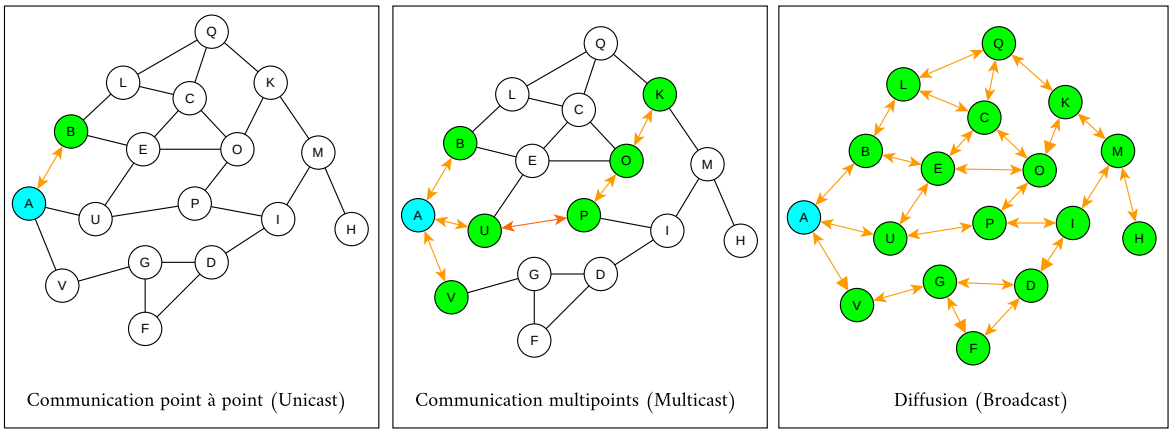


FIGURE I.3 – Mode de commnunication dans les réseaux Ad Hoc

2.4. Caractéristiques des réseaux mobiles ad hoc

Les principales caractéristiques d'un MANET sont :

- **Absence d'infrastructure :** Les MANETs se distinguent des autres réseaux par la propriété d'absence d'infrastructure. Il n'existe pas d'organe central de control pour gérer la communication entre les nœuds du réseau. Les nœuds s'auto-organisent pour créer et maintenir des routes dynamiques entre eux afin d'assurer la commnunication dans le réseau.

- **Limitations d'énergie :** Les nœuds d'un MANET sont généralement petits et léger. Ils embarquent une source d'énergie (batterie, ou autre source d'énergie consommable) limités.
- **Sécurité physique limitée :** Contrairement au réseau filaire, les réseau sans fil sont extrêmement sensibles aux interférences. Les signaux radio , les radiations et tout autre type similaire peuvent provoquer un dysfonctionnement ou des interférences à un réseau sans. Les réseau sans fil peuvent être accessibles à n'importe quel équ sans fil à la portée du signal du réseau. Ils sont alors enclins aux ménaces d'écoute clandestine, d'usurpation d'identité, déni de service, etc.
- **Limitations de la bande passante :** Les réseaux basés sur la communication sans fil communique via un méduim radio partagé. Ils sont sensible à l'atténuation du signal, aux interférence ainsi qu'aux trajets multiple. La capacité des liens est limitée et variable.
- **Topologie dynamiques :** Les nœuds d'un MANET se déplacent de façon libre et arbitaire. Ce qui peut provoquer un changement rapide et imprévisible de la topologie du réseau. Les liens entre les nœuds peuvent alors devenir bidirectionnels ou unidirectionnels.

2.5. Domaines d'application des MANETs

Les MANETs (ou les réseaux ad-hoc en général) peuvent être utilisés partout où le déploiement d'une infrastructure réseau fixe est trop contraignant, que ça soit par rapport à la difficulté de la mise en place, au coût d'installation, à la durée d'installation, etc. Parmi ces domaines d'application nous pouvons citer :

- **Operation de sauvetage :** En cas de castastrophe (incendie, inondation, tremblement de terre, etc) dans une zone, les infrastructures des communication de cette zone sont dans la plupart des cas endommagées. Les opérations de sauvetage peuvent alors devenir très délicate sans moyen de communications entre les équipe d'urgence. Il devient primordiale dans de telle situation de trouver rapidement un moyen de communications. Un réseau mobile ad-hoc (MANET) peut être utiliser dans ces genre de situations pour assurer la communication. Les informations seront relayées par des petits ordinateur de poche que les agents de l'équipe peuvent avoir avec eux sans être gêner dans leur mission de sauvetage.

- **Conflits millitaire :** Sur un champ de bataille, les MANETs peuvent être utilisés par les soldats avec un accent sur le déploiement rapide, les réseaux sans infrastructure et totalement sans fil (pas de tour radio fixée), la robustesse (les ruptures de liaison ne sont pas un problème), la sécurité, la portée, et le fonctionnement instantané. Les MANETs peuvent être utilisés dans les mines « sautantes » de l'armée, dans des pelotons où les soldats communiquent en terrain étranger, leur donnant la supériorité sur le champ de bataille.
- **Institution Universitaire et commercial (Entreprise) :** Les réseaux MANETs peuvent être utilisés dans le cadre de travail collaboratif (une réunion par exemple) ou pour assurer la communication dans les entreprises.

3. Modélisation d'un MANET

Les réseaux mobiles ad-hoc ou MANETs peuvent être modéliser à l'aide d'un graphe non orienté $G = (V, E)$ où V désigne l'ensemble des nœuds mobile (les terminaux du réseaux) et E ($E \subseteq V$) l'ensemble des arêtes (les liaison radio entre les nœuds mobile) [5]. En effet lorsqu'un nœud (sommet) A peut émettre par liaison radio un paquet vers un nœud B alors le nœud A et B seront relié par un arc dirigée du sommet A vers le sommet B. Cependant si les nœuds ont la même portée de transmission r, alors le graphe résultant peut être considéré comme non orienté, puisque s'il existe un arc dirigée du sommet A vers le sommet B, et un autre du sommet B vers le sommet A, on a dans ce cas un arc bidirectionnel, autrement, une arête.

3.1. Modélisation à l'aide de graphes de disques unitaires (UDG)

Dans la théorie des graphes, un graphe de disques (ou disk graph en anglais) est le graphe d'intersection d'une collection de disques. Quand tous les disques ont le même diamètre, le graphe est qualifié de graphe de disque-unité (Unit-disk graph an anglais). Tout sous-graphe induit d'un graphe de disque-unité est également un graphe de disque-unité. Lorsque tous les nœuds du graphe ont la même plage de transmission (même puissance d'émission, etc.) et que le système radio est idéal, alors un tel réseau peut donc être représenté par un Unit Disk Graph (UDG)[2].

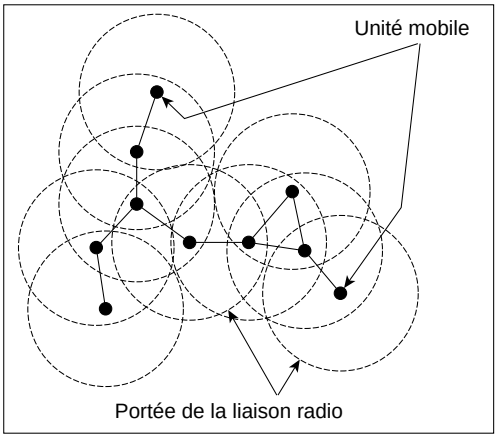


FIGURE I.4 – Modélisation de la topologie d'un MANET par des graphiques de disques unitaires

3.2. Mobilité dans les réseaux mobile ad-hoc (MANETs)

3.2.1. Concepte de mobilité

La mobilité est l'un des facteurs majeurs qui influe sur les performances d'un MANET. L'étude des modèles de mobilité via des simulation plus proche de la réalité est le meilleur moyen d'évaluer les performances d'un MANET à travers les différents protocoles de routage proposés pour ces types de réseaux. Le mouvement réel étant beaucoup plus riche et plus beaucoup complexe, il est très difficile d'élaborer un modèle de mobile plus réalistes [6]. Néanmoins, plusieurs modèle de mobilités ont été proposés dans la littérature pour les MANETs. Chacun de ses modèles de mobilité possèdes des avantages et également des lacunes car les protocoles ad-hoc sont rarement élaborés pour une application très précise.

3.2.2. Modèles de mobilités pour les MANETs

Il existe une grande variété de modèles synthétiques répartis en deux classes suivant leur caractéristiques de mobilité[6].

Dans les modèles synthétiques le mouvement des nœuds est modélisé sous formes d'équations mathématique utilisant des processus aléatoire. Les modèles de mobilités peuvent être différencier selon leurs dépendances spatiales et temporelles :

- **Dépendance spatiale :** c'est une mesure de la façon dont deux nœuds sont dépendant de leur mouvement. Si deux nœuds se déplacent dans la même direction alors ils ont une forte dépendance spatiale.[7].

- **Dépendance temporelle :** c'est une mesure de la façon dont la vitesse courant(amplitude et direction) sont liées à la vitesse précédente. Les nœuds ayant la même vitesse ont une forte dépendance temporelle.[7]

3.2.2.1. Modèles de mobilité d'entité Il exite plusieurs modèles de mobilité d'entité[7]. Nous citons ici quelques uns d'entre eux.

3.2.2.1.1. Point d'attente aléatoire (Random Waypoint) : Le modèle Random Waypoint est le modèle de référence dans la communauté de recherche [7]. Dans ce modèle de mobilité, un nœud choisit au hasard une destination dans l'espace de mouvement et une vitesse uniformément distribué entre une vitesse maximale et une vitesse minimale de déplacement et se déplace vers cette destionation. Lorsque le nœud atteint la destination, il observe un temps de pause déterminé, puis choisit à nouveau une destination et un temps d'attente aléatoire et répète l'ensemble du processus jusqu'à la fin de la simulation.

Dans la plupart des cas, la distribution des positions initiales est choisie de façon aléatoire et uniforme dans l'espace de mouvement. Les mouvements, les changement de vitesses et/ou d'angle sont espacé de pause [8]. La figure I.5 illustre un cas de mouvement suivant le modèle du point d'attente aléatoire.

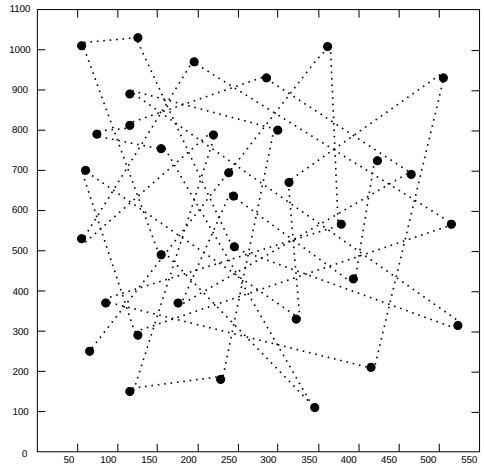


FIGURE I.5 – Déplacement d'un nœud suivant le modèle du point d'attente aléaoire

3.2.2.1.2. Marche aléatoire (Random Walk) : Le modèle de marche aléatoire a été initialement proposé par Einstein en 1926 pour décrire le mouvement brownien [1]

1. Le mouvement brownien, ou processus de Wiener, est une description mathématique du mouvement aléatoire d'une « grosse » particule immergée dans un fluide et qui n'est soumise à aucune autre interaction que des chocs avec les « petites » molécules du fluide environnant.

[9]. Il est largement utilisé dans divers domaine. Dans ce model de mobilité chaque nœud sélectionne indépendamment de ses voisins une destination avec un angle et une vitesse prise équi-probablement entre des valeurs minimales et maximales. Chaque mouvement s'éffectue sur un intervalle de temps constant [10].

Le processus du modèle de mobilité de marche aléatoire est sans mémoire [11] (c'est à dire qu'il ne conserve aucune information sur l'état antérieur du nœud lors du passage à la décision future). Quand le nœud atteint la limite de la zone de mouvement, il choisit aléatoirement et uniformément une direction $\theta(t)$ dans la plage prédéfinie de 0 à 2π et une vitesse V_t entre 0 et V_{max} pour se déplacer vers un nouvel emplacement. Après une constante intervalle de temps t, une nouvelle direction et une nouvelle vitesse sont calculées et assigné au nœud . Si le nœud atteint la bordure de la de mouvement zone, il rebondi d'un angle $\theta(t)$ ou $\pi - \theta(t)$ [12, 6]. L'ullistration de la figure I.I.5 est assez simulaire au cas de mouvement suivant le modèle de marche aléatoire si nous supprimons le temps de pause.

3.2.2.1.3. Modèle de mobilité de Manhattan : Le modèle de mobilité de Manhattan utilisé pour modéliser le mouvement des nœuds mobiles dans une zone urbaine. Il imite le mouvement des nœuds dans les rues [13]. Dans ce modèle de mobilité, les nœuds se déplacent uniquement le long de la grille horizontales et verticales de la rue avec une vitesse V_t entre 0 et V_{max}. Lorsqu'un nœud arrive à l'intersection d'une rue horizontale et d'une rue verticale, il peut virer à gauche, à droite ou continué tout droit suivant une certaine probabilité. La figure I.I.6 illustre un cas de mouvement suivant le modèle Manhattan.

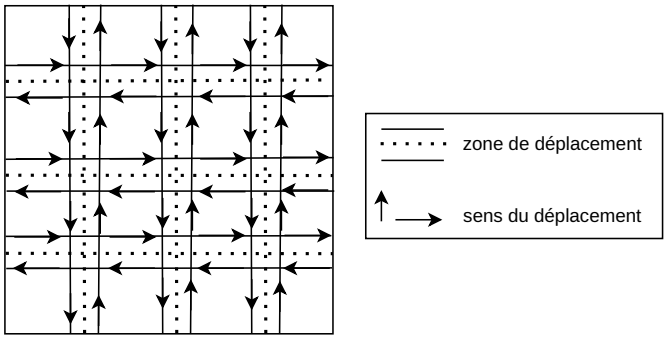


FIGURE I.6 – Déplacement d'un nœud suivant le modèle du point de Manhattan

3.2.2.1.4. Modèle de mobilité de Gauss-Markov : Ce modèle de mobilité a été proposé par LIANG et HAAS. Il permet de déterminer les vitesses de mouvement d'un

nœud à des intervalles de temps régulier. Il démarre de l'hypothèse que le mouvement suit un processus gaussien [1] et qu'un processus de Gauss-Markov peut le décrire. Dans ce modèle de mobilité, chaque nœud est initialisé par une vitesse et une direction courante [10]. A des intervalle de temps régulier le mouvement de chaque nœud est décrit par la mise à jour de la vitesse et de la direction. Plus précisément, la valeur de la vitesse et la direction à la n^{ime} instance sont calculées sur la base de la valeur de la vitesse et de la direction du $(n-1)^{ime}$ instance et d'une variable aléatoire en utilisant l'équations suivantes :

$$s_n = \alpha s_{n-1} + (1+\alpha)\bar{s} + \sqrt{(1+\alpha^2)s_{x_{n-1}}}$$
$$d_n = \alpha d_{n-1} + (1+\alpha)\bar{d} + \sqrt{(1+\alpha^2)d_{x_{n-1}}}$$

où s_n et d_n sont la nouvelle vitesse et la nouvelle direction du nœud à l'intervalle de temps n ; α $(0 \leq \alpha \leq 1)$ est le paramètre de réglage utilisé pour faire varier le caractère aléatoire ; $\bar{s}$ et $\bar{d}$ sont des constantes représentant la valeur moyenne de la vitesse et direction lorsque $n \to \infty$ $s_{x_{n-1}}$ et $d_{x_{n-1}}$ sont des variables aléatoires d'une distribution. Si $\alpha = 0$, la vitesse à un instant donné est indépendant de l'instant précédent. Le mouvement devient linéaire lorsque $\alpha = 1$

L'une des caractéristiques clé de ce modèle de mobilité est l'absence de temps de pause et changement brusque d'angle et de vitesse, contrairement au modèle du point d'attente aléatoire. Le mouvement peut devenir plus réaliste par l'ajout de facteur de pondération avec les valeurs précédentes de l'angle et de la vitesse. La figure I.7 montre une illustration du mouvement d'un nœud suivant le modèle Gauss-Markov.

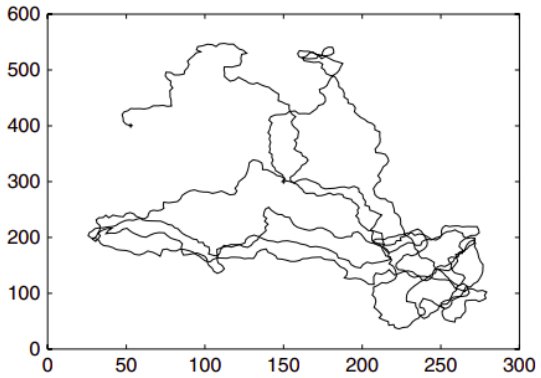


FIGURE I.7 – Mouvement d'un nœud suivant le modèle Gauss-Markov[6]

3.2.2.2. Modèle de mobilité de groupe

1. Un processus stochastique X sur un ensemble fini de sites S est dit gaussien si, pour toute partie finie $A \subset S$ et toute suite réelle (a) sur $A, \sum s \epsilon A a_s X(s)$ est une variable gaussienne.

3.2.2.2.1. Modèle de mobilité de colonne (Column MM) : Le modèle de colonne représente un ensemble de nœuds mobiles (par exemple, des robots) qui se déplacent dans une certaine direction fixe. Ce modèle de mobilité peut être utilisé dans les activités de recherche et de balayage, telles que la destruction de mines par des robots militaires [15]. La figure I.8 montre une illustration du mouvement suivant le modèle de mobilité de colonne.

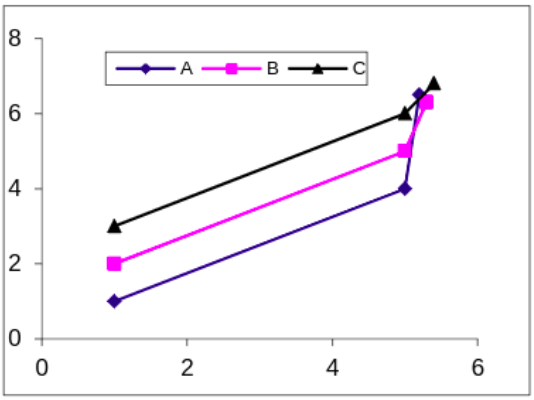


FIGURE I.8 – Mouvement d'un nœud suivant le modèle de mobilité de colonne[6]

3.2.2.2.2. Points de référence de groupe : La mobilité de groupe points de référence peut être utilisée dans la communication sur le champ de bataille militaire. Chaque groupe a un centre logique (chef de groupe) qui détermine le comportement de mouvement du groupe. Au départ, chaque membre du groupe est uniformément réparti dans le quartier du chef de groupe. Par la suite, à chaque instant, chaque nœud a une vitesse et une direction qui sont dérivées en s'écartant aléatoirement de celle du chef de groupe [15]. La figure I.I.9 montre une illustration du mouvement suivant le modèle points de référence de groupe.

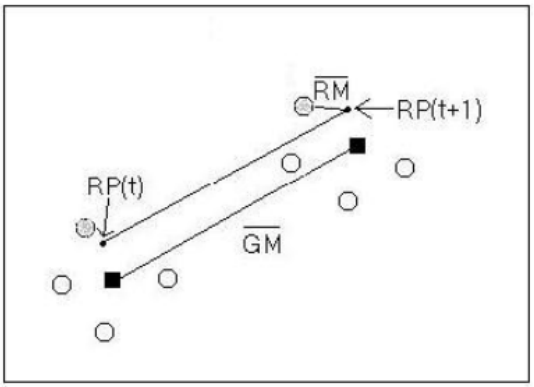


FIGURE I.9 – Mouvement d'un nœud suivant le modèle de mobilité points de référence de groupe [6]

3.2.2.2.3. Modèle de mobilité de poursuite (Pursue MM) : Le modèle de mobilité de poursuite émule des scénarios dans lesquels plusieurs nœuds tentent de capturer un seul nœud mobile à l'avance. Ce modèle de mobilité pourrait être utilisé dans le suivi des cibles et l'application de la loi. Le nœud poursuivi se déplace librement selon le modèle Random Waypoint. La mobilité des nœuds de poursuite est dirigée vers la position et le mouvement du nœud cible [15].

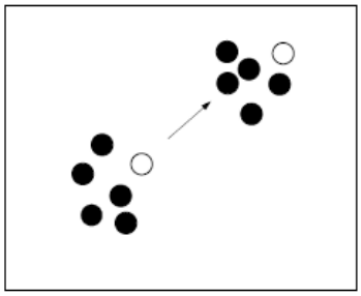


FIGURE I.10 – Mouvement d'un nœud suivant le modèle de mobilité de poursuite[6]

4. Routage dans les réseaux mobiles ad-hoc

Dans les réseaux ad-hoc, il n'existe pas d'infrastructure ou d'organe central d'administration ou de control (routeur) qui co-ordonne l'acheminement des données entre nœuds . Le traffic entre une source et une destination est donc acheminer soit directement, si l'émetteur et le récepteur sont à portée l'un de l'autre, ou retransmis de proche en proche par les nœuds intermédiaires dans le cas contraire [2]. Chaque nœud se comporte donc comme un routeur.

4.1. Principe du routage

Le routage est le mécanisme qui permet d'établir des chemins dans un réseau par lesquels les paquet de ce réseau vont être achéminer vers un autre réseau.

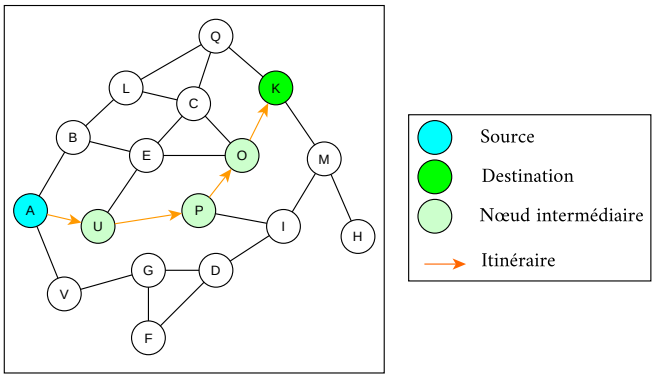


FIGURE I.11 – Chemin utilisé dans un Routage

4.2. Problématique du routage

Les caractéristiques des réseaux ad-hoc ne permettent pas le déploiement des protocoles de routage traditionnels utiliser dans les réseaux filaires. En effet, l'absence d'infrastructure ou d'administration fixe impose un fonctionnement distribué dans ses réseaux. Le déplacement des nœuds mobiles peut remettre en cause la validité des informations de routage. Chaque nœud participe alors au routage en retransmettant les paquets d'un nœud qui n'est pas en mesure d'atteindre sa destination.

Dans les réseaux ad-hoc, la limitation des ressources matérielles des nœuds mobile conduit à l'exclusion des algorithmes de routages qui sont exigeant par rapport à la capacité de mémoire et/ou de traitement des nœuds . De plus la taille des réseaux ad-hoc peut devenir très importante, créant ainsi un problème d'adaptation des méthodes d'acheminement de paquets pouvant être utiliser dans ces réseaux [16].

4.3. Protocole de routage

Un protocole de routage est un système qui permet à un routeur de déterminer la route par laquelle les paquets vont être achimnier vers un autre réseau.

4.4. Classification des protocoles de routage ad-hoc

Le problème de routage dans les réseaux ad-hoc est l'un des défis fondamentaux de la communauté scientifique de recherche dans ce domaine [17].

Pour résoudre ce problème plusieurs techniques (protocoles) ont été proposés. Les protocoles de routage dédiés aux réseaux ad hoc peuvent être caractérisés par leur catégorie

(protocoles proactifs, réactifs ou hybrides), leur architecture (plate ou hiérarchique) ainsi que par leur type de méthode (à état de lien ou vecteur de distance).

4.4.1. Protocole de routage proactif, réactif et hybride

On distinguent trois catégorie de protocoles de routages :

4.4.1.1. Protocole de routage proactif : Dans ce type protocole de routage, les routes vers toutes les destinations sont établies dans une table de routage avant qu'il y ait une demande de transmission. En effet chaque nœud maintient à jour une table de routage à travers des échanges périodiques d'informations de routage avec les autres nœuds mobiles du réseau. Ces protocoles de routage présentent comme avantage la disponible permanente des informations de routage. Cependant, ils ont comme inconvénient majeur, l'encombrement permanente du réseau par un trafic (dont une partie importante est inutile) de contrôle concernant les routes non modifiées et/ou non utilisées.

4.4.1.2. Protocole de routage réactif : Dans ce type de protocole de routage, une route n'est calculée que si elle est nécessaire à la communication entre deux nœuds . En effet plutôt que de s'échanger périodiquement des informations sur la topologie, une recherche de chemin vers un nœud destinateur est éffectué uniquement quand c'est nécessaire, c'est à dire, lorsqu'un nœud source veut émettre vers un nœud destinateur.

La durée d'obtention d'une route est souvent plus longue, néanmoins le trafic de contrôle réquis pour obtenir une route est minimisé car les mécanismes ne sont déployés qu'en cas de besoin [18, 19]. Les protocoles de routage DSR, RPL sont des protocole de routage réactif.

4.4.1.3. Protocole de routage hybrides : Les protocoles de routage hybrides tentent de réunir les avantages des protocole de routage proactif et réactif [18].

4.4.2. Protocole de routage à état de lien et à vecteur de distance

4.4.2.1. Méthode à état des liens : Dans le protocole de routage à état de lien, chaque nœud construit une carte connectivité de chaque nœud (routeur) 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. $OSPF$ est un protocole de routage interne IP de type « à état de liens ».

4.4.2.2. Protocol à vecteur de distance : Dans un routage à vecteurs de distance Les nœuds maintiennent une table de routage à travers une diffusion périodiquement d'un vecteur (table) où chaque entrée de la table comprend une destination, le prochain saut (nœud intermédiaire) pour atteindre la destination et une métrique (nombre de sauts pour atteindre la destination). Un nœud met à jour sa table de routage s'il trouve une route plus courte que celle qu'il a dans sa table, ou si le nœud à travers lequel il peut atteindre une destination donnée change la distance vers cette destination, ou encore s'il trouve un nœud qui n'existe pas dans sa table. Les protocoles de routage RIP(**R**outing **I**nformation **P**rotocol), *AODV*, *DSDV* fonctionnent suivant ce principe.

4.4.3. Protocole plat ou hiérarchique

On distinguent deux type de protocole de routage dans la classification basée sur l'architecture :

4.4.3.1. Protocole plat : Dans ce type de protocole de routage, les nœuds ont les mêmes fonctionnalités par rapport au routage, la décision d'un nœud de router des paquets vers un autre nœud dépendra de sa position. Les protocoles de routage *DSR* et *AODV* fonctionnent suivant ce principe.

4.4.3.2. Protocole hiérarchique : Dans ce type de protocole de routage, certains nœuds prennent en charge la fonctionnalité de routages par rapport aux autres nœuds du réseaux.

4.5. Protocole de routage basé sur une dorsale (backbone)

Le protocole de routage OLSR (**O**ptimized **L**ink **S**tate **R**outing protocol) est un protocole de routage à état de lien optimisé [20]. C'est un protocole de routage à base de backbone (épine dorsale).

Le protocole *OLSR* utilise le concept de relais multi-points *MPR* (MultiPointRelays) pour optimiser la diffusion des messages topologiques sur le réseau. Les nœuds de l'ensemble *MPR* sont les seuls autorisés à retransmettre les messages topologiques, ils forment l'épine dorsale (ou le backbone). Les nœuds sont sélectionnés parmi les voisins à un saut de sorte l'ensemble des *MPR* permettent l'accès mutuel aux nœuds à un saut les uns des autres. Ainsi, chaque nœuds , pour savoir s'il peut retransmettre un paquet de contrôle reçu, doit disposer de la liste des nœuds qui l'ont choisit comme *MPR* et

qui représentent ses sélectionneurs multi-points. Pour sélectionner un ensemble de voisins qui forment ses $MPRs$, un nœud doit disposer de la liste de ses voisins directs et de ses voisins à deux sauts. Le concept de backbone sera détaillé dans le chapitre II.

4.6. Protocole de routage basé sur des clusters

Le Protocole de Routage Basé sur les Clusters appelé $CBRP$ (Cluster BasedRouting Protocol) [21] est un protocole hiérarchique basée sur la clustérisation. L'ensemble des nœuds du réseau est rassemblé en des clusters. Le protocole de routage $CBRP$ se compose principalement de deux phases : la formation de clusters et le routage.

La première phase consiste à grouper les nœuds dans des clusters. Le nœud qui a le plus petit identifiant (ID) parmi tous ses voisins est élu comme tête de cluster (cluster-head). Le cluster formé est composé du cluster-head et de tous ses voisins.

Conclusion

Dans ce chapitre nous avons présenté les MANETs dans leur généralité à travers leurs historiques, leurs caractéristiques, leurs applications, leurs types etc. Nous avons aussi étudier quelques modèles de mobilités proposés dans la littérature scientifique dans le but de maître en évidence les impacts de la mobilité des nœuds sur les performances général d'un MANET. Et enfin avons terminer ce chapitre par l'étudie du routage et de quelques de routages utilisé dans les MANETs.

Dans le prochain chapitre nous étudierons les structures virtuelles pouvant servir d'épine dorsale afin de combler l'impact de l'absence d'infrastructure centrale d'administration ou de control dans les MANETs et facilité certaines taches complexes comme le routage. Nous étudierons également quelques algorithmes de construction d'épine dorsale proposé dans la littérature scientifiques.

Algorithme de construction d'épine dorsale (backbone)

1. Introduction

Dans le domaine des réseaux informatiques, une épine dorsale (ou backbone en anglais) est l'ensemble des supports de transmission et de communication à partir du commutateur d'abonné ; il supporte la partie la plus importante du trafic avec une bande passante importante. Dans les MANETs, l'absence d'infrastructure centrale d'administration rend certaines tâches telle que le routage très difficiles.

La communication dans un MANET se fait en mode « point à point » ou en mode « multi-sauts », c.à.d, une paire de nœuds ne peuvent communiqués que s'ils sont à portée l'un de l'autre. Les messages sont donc transmis soit par un saut si l'émetteur et le récepteur sont à portée l'un de l'autre (mode point à point), ou à travers des nœuds intermédiaires dans le cas contraire (mode multi-sauts)[2]

Dans un réseau de grande taille, la communication (routage de paquet) entre deux nœuds très éloignés nécessite l'implication de plusieurs nœuds intermédiaires. Cela peut constituer un énorme problème surtout si nous prenons en comptes les limitations des MANETs en termes de ressources materielles (mémoire, énergie, puissance de calcul, bande passante, etc.), de l'hétérogénéités des nœuds et l'absence d'infrastructure. L'utilisation de structures virtuelles hiérarchiques telle que les ensembles dominants connectés (CDS) : Connected Dominating Set) en tant qu'infrastructures virtuelles des réseaux sans fil ad hoc peut facilité certaines tâches du réseau telle le routage.

Dans ce chapitre nous étudierons les principales structures virtuelles proposé dans la littérature scientifique, et qui peuvent servir d'épine dorsale (backbone ou CDS) dans un MANET dans le but de faciliter les tâches complexe du réseau. Nous étudierons ensuite quelques algorithmes de construction de CDS.

2. Structures virtuelles pour les MANETs

Dans un MANET, le changement permanent de la topologie du réseau dû à la mobilité des nœuds influe énormément sur les performances du réseau. Une solution efficace à ce problème est de définir une architecture d'auto-organisation. Cette structure faciliterais la gestion des tâches telle que le routage. Parmi les techniques d'organisation des MANETs, il y a la définition d'un backbone virtuel (épine dorsale) qui couvre tous les nœuds du réseaux. Un backbone est une vision logique de la topologie du réseau. Il a pour but de minimiser la charge du réseau en optimisant les processus de diffusion. Parmi les principales types de structure virtuelles couramment utilisés dans les MANETs nous pouvons citer :

2.1. Cluster

Le clustering est le principe de répartition des nœuds du réseau en plusieurs groupes (appelés cluster). Pour un graphe $G = (V, E)$, un cluster représente un sous-ensemble de ce graphe.

Dans un cluster, le nœud dominant ou « tête de cluster » (le clusterhead) est désigné à la suite d'une élection basé sur des algorithmes d'élection de tête de cluster, comme celui dans [22]. Une fois élu, le clusterhead se charge alors du bon fonctionnement du cluster dont il en est le chef. Les autres nœuds (nœuds régulier) hormis ceux qui sont situés à la jonction de plusieurs clusters sont appelé « cluster member ». Quand aux nœuds qui sont à la jonction de plusieurs portent le nom de « passerelle du cluster ou cluster gateway ». La figure II.II.1 donne une illutstration de l'architecture d'un cluster.

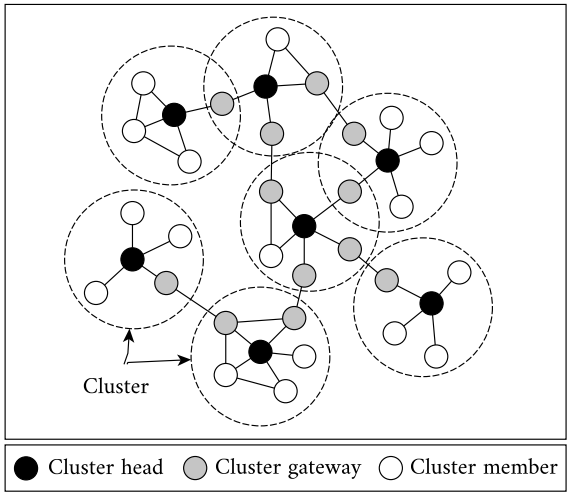


FIGURE II.1 – Structure d'un cluster

2.2. Relais Multipoint (MPR : Multipoint Relay)

Introduite pour la première fois dans la norme HIPERLAN (High-Performance Radio Local Area Network) de type 1 [23], les relais multipoints sont utilisés pour prévenir l'inondation du réseau par des messages de contrôles inutiles en réduisant leur nombre. Il fonctionne suivant une règle appelé « règle multipoint » dans lequel chaque nœud du réseau, pour atteindre tout le voisinage à deux sauts, choisit un sous-ensemble minimum de nœuds appelés « relais multipoint (MPR) » parmi ses voisins symétriques à un saut. Les autres nœuds qui ne sont pas dans l'ensemble MPR peuvent lire, mais pas retransmettre, les paquets de diffusion [23]. La figure II.II.2(f) donne une illustration de l'architecture d'un relais multipoint.

2.3. Ensemble dominant (DS)

Un ensemble dominant (DS : Dominating Set) V' dans un graphe $G = (V, E)$ est ensemble de sommets (nœuds) dans lequel tout sommet du graphe est voisin d'au moins un sommet de cet ensemble. Une illustration d'un DS est donnée par la figure II.II.2(a).

2.4. Ensemble dominant connectés (CDS)

Dans la terminologie des graphes, un ensemble dominant (Dominant Set ou DS) V' d'un graphe $G = (V, E)$ est un sous-ensemble de sommets tel que chaque sommet du graphe est soit dans l'ensemble, soit a un lien vers un sommet de l'ensemble. Si en plus de cela les sommets de cet DS sont connectés, cet ensemble porte alors le nom d'ensemble

dominant connecté (CDS). Une illustration d'un CDS est donnée par la figure II.II.2(b).

2.5. Ensemble indépendant (IS)

Un ensemble indépendant (IS : Indépendent Set) V' d'un graphe $G = (V, E)$ est un ensemble de sommets (nœuds) dans lequel aucune paire de sommets de cet ensemble ne son voisin dans le graphe. Une illustration d'un IS est donnée par la figure II.II.2(c).

2.6. Ensemble indépendant maximal (MIS)

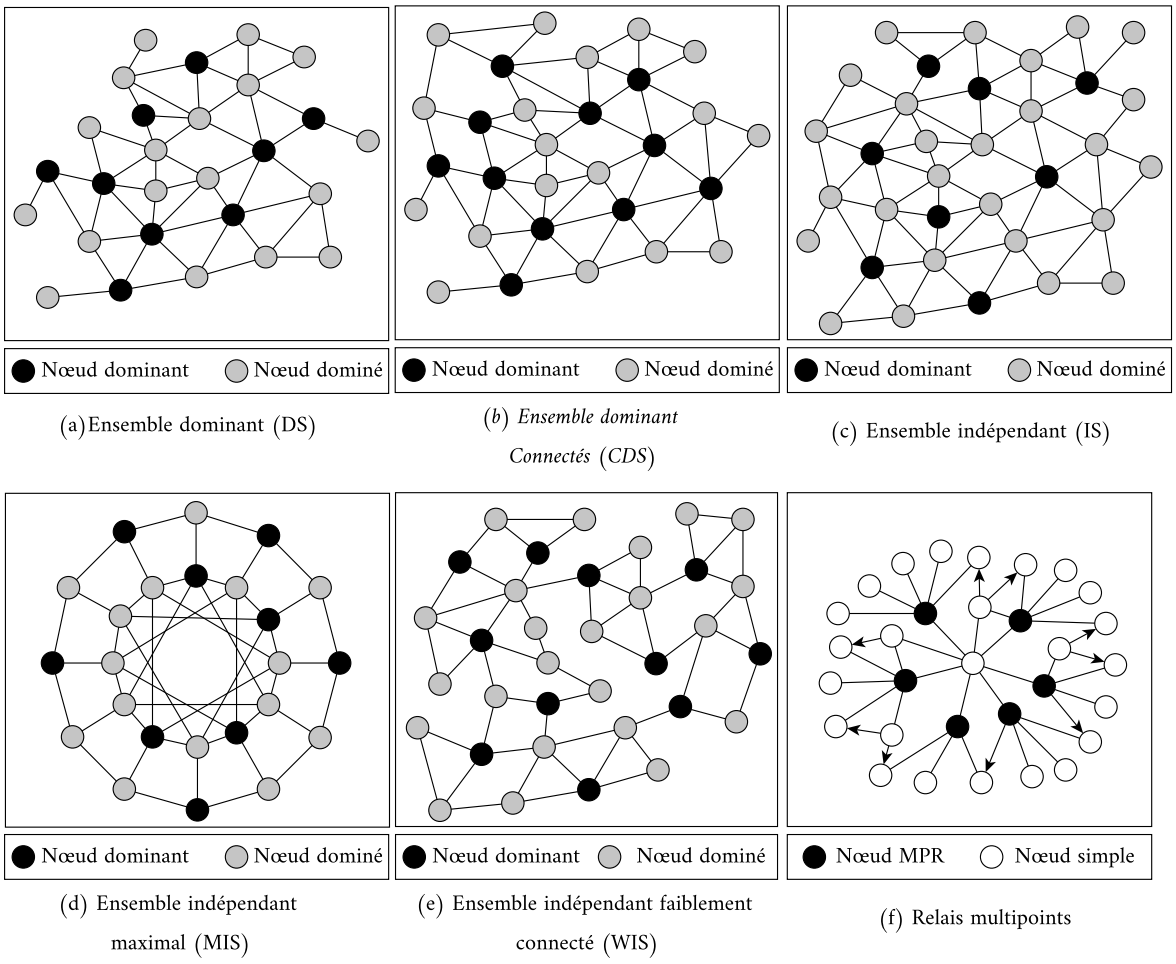
Un ensemble indépendant maximal (MIS : Maximal Independent Set) V' d'un graphe $G = (V, E)$ est un ensemble indépendant de plus grande taille possible pour ce graphe. Une illustration d'un MIS est donnée par la figure II.II.2(d).

2.7. Ensemble dominant faiblement connecté ($WCDS$)

Un ensemble dominant faiblement connecté ($WCDS$: Weakly Connected dominating Set) V' d'un graphe $G = (V, E)$ connexe est un ensemble dominant de sommets du graphe tel que les arêtes non incidentes à un sommet de V' ne séparent pas le graphe [24]. Une illustration d'un $WCDS$ est donnée par la figure II.II.2(e).

2.8. Ensemble dominant connectés pondéré minimum ($MWCDS$)

Un ensemble dominant connectés pondéré minimum ($MWCDS$: Minimum Weighted Connected dominating Set) d'un graphe $G = (V, E)$ est un sous-ensemble V' de V tel que V' est un CDS avec un poids minimum.[25].



(a) Ensemble dominant (DS)

(*b*) *Ensemble dominant Connectés* (*CDS*)

(c) Ensemble indépendant (IS)

(d) Ensemble indépendant maximal (MIS)

(e) Ensemble indépendant faiblement connecté (WIS)

(f) Relais multipoints

FIGURE II.2 – Ensemble dominant, ensemble indépendant, Ensemble dominant connectés

3. Algorithmes de construction de CDS

Dans les revues scientifiques en rapport avec les MANETs plusieurs algorithmes (centralisés ou distribués) de constructions de backbone basés sur différentes contraintes (consommation d'énergie, contrainte de mobilité, puissance de calcul et/ou de transmission, etc) on été proposés. Nous présentons dans cette section certains de ces algorithmes.

3.1. Algorithmes de Guha et Khuller

Guha et Khuller ont Proposé deux alogrithmes dans [26], qui permettent de construire un CDS. Le premier algorithme fonctionne suivant le principe suivant : L'algorithme part d'un arbre dont les sommets (nœuds) sont initialement coloriés en blanc. Il parcourt l'arbre à la recherche de sommet de degré maximum (nœud dominant) et les colories en noir.

Les sommets adjacents au sommets coloriés en noir (ses voisins) sont coloriés en gris (ces nœuds sont appélés « les dominés »). A l'itération suivante l'algorithme choisi parmi les sommets (nœuds) colorés en gris (les dominés) qui a le plus de voisins coloriés en blanc et le coloriés en noir. L'algorithme continue ainsi son itération jusqu'à ce qu'il n'y ait plus de sommet de degré maximum colorié en gris avec des voisins coloriés en blanc.

L'exécution de l'algorithme s'arrête lorsque l'arbre ne contient plus que des sommets de degré minimum (sommet avec une seule arête ou nœud feuille). A la fin de l'exécution de l'algorithme, les sommets coloriés en noir forme le CDS. Le principe de fonctionnement de cet algorithme est analogue à celui proposé par Das, B., Sivakumar, R., & Bharghavan, V. dans [27]. Une illustration du premier algorithme de Guha et Khuller est donnée à la figure II.II.3

Le second algorithme de Guha et Khuller dans [26] s'exécute en deux phases : Au début de la première phase de cet algorithme, comme dans l'algorithme précédent, tous les nœuds sont coloriés en blanc. De même que le premier algorithme, dans la première phase, les nœuds de l'ensemble dominant sont coloriés en noir et leurs voisins adjacent sont coloriés en gris.

L'objectif de la première phase est d'avoir un nombre minimum de pièces. Une pièce est définie comme étant soit un nœud blanc ou un ensemble de noir connectés [3]. A chaque itération de l'algorithme dans la première phase, il sélectionne un nœud pour le colorié en noir, ce qui conduit à une réduction maximale du nombre de pièces.

L'algorithme s'exécute ainsi jusqu'à ce qu'il ne reste plus de blanc, autrement, jusqu'à ce que tous les nœuds soient coloriés noir ou en gris. Il résulte de cette première phase, un ensemble de nœuds colorié en noirs et en gris.

Dans la deuxième phase, les paires de nœuds coloriés en noir sont reliés récursivement entre eux en colorant les chaînes de nœuds gris en noir. L'ensemble de nœuds colorié en noir et connectés entre eux forme le CDS.

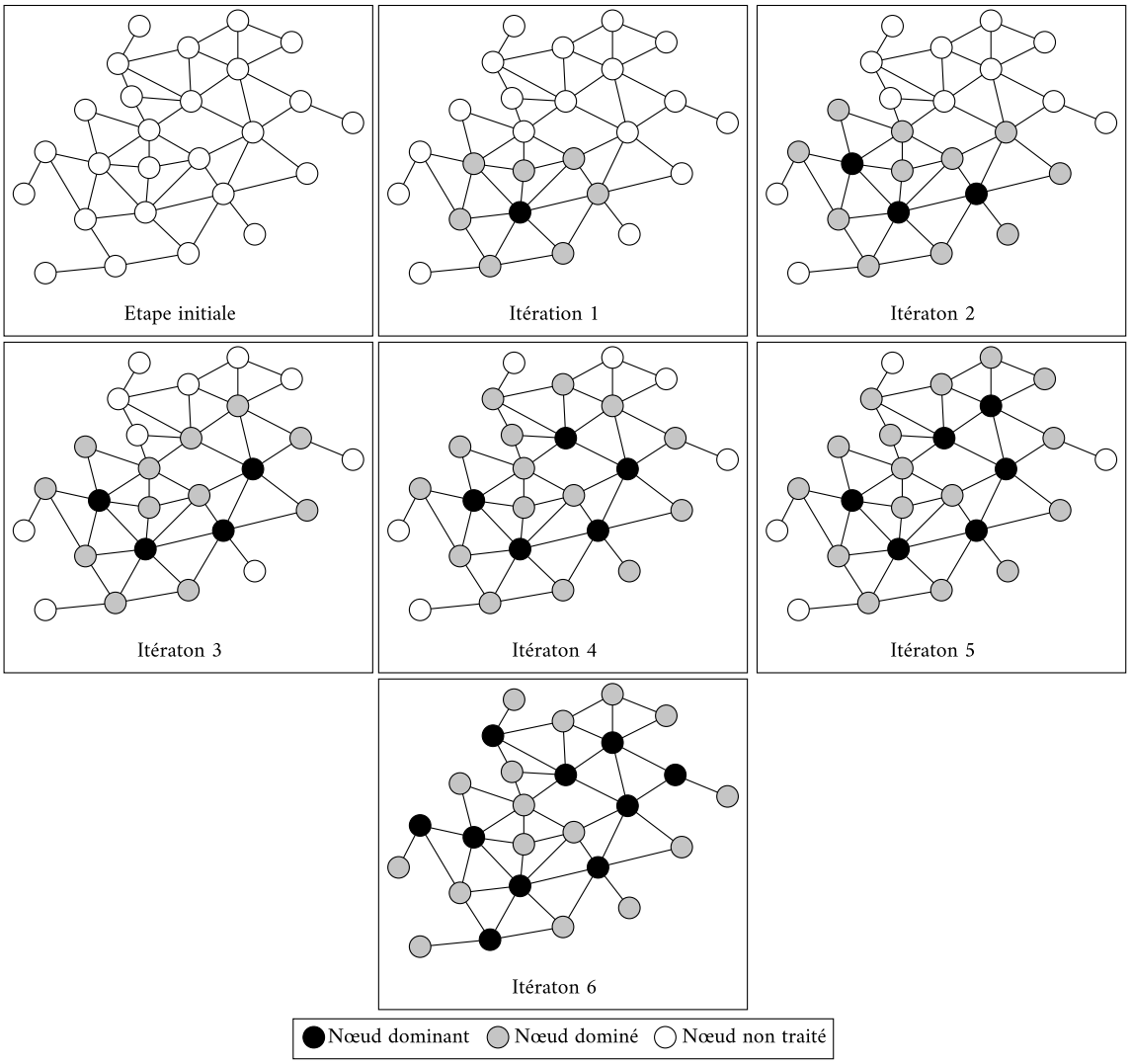


FIGURE II.3 – Premier algorithme de Guha et Khuller

3.2. Algorithme de Yu et al.

Proposé dans [28], cet algorithme est une amélioration de [29]. Il se compose de deux étapes. Une étape de calculs d'un DS et une étape de réduction.

L'algorithme commence par calculer un DS à haute densité à travers l'algorithme proposé dans [29]. Après le calcul du CDS à haute densité, une étape de réduction est effectué pour réduire la densité du CDS de sorte que chaque paire de dominateurs à moins de trois sauts soit connectés par un chemin de connecteurs. A la fin de l'étape de réduction les dominateurs et connecteurs résultant forment un CDS.

3.3. Algorithme de Cardei et al.

Proposé dans [30], au départ de cet algorithme comme celui décris à la « section II.II.3.1. », chaque sommet (nœud) est de couleur blanche, un nœud dominant est de couleur noire, tandis qu'un nœud dominé est de couleur grise. L'algorithme s'exécute en deux phases. Il suppose dans la première phase que chaque sommet connaît ses voisins de distance-un et leurs degrés effectifs [1].

L'algorithme désigne également un nœud comme chef de file en utilisant un algorithme distribué d'élection de leader. L'algorithme colore ensuite ce chef en noir. Le nœud comme chef diffuse le message « DOMINATOR ». Tout nœud blanc qui reçoit un message « DOMINATOR » pour la première fois se colore en gris et diffuse le message « DOMINATEE ». Un nœud blanc recevant un message « DOMINATEE » devient actif. Un nœud blanc actif avec un id et un degré élevé parmi tous ses voisins blancs actifs se colorera en noir et diffusera le message « DOMINATOR ». Un nœud blanc diminue son degré effectif de 1 et diffuse le message « DEGREE [2] » chaque fois qu'il reçoit un message « DOMINATEE ». Un nœud blanc recevant un message « DEGREE » mettra à jour ses informations de voisinage en conséquence. Chaque sommet gris diffusera un message « NUMOFBLACKNEIGHBORS » lorsqu'il détecte qu'aucun de ses voisins n'est blanc. La phase 1 se termine lorsqu'il n'y a plus de sommet blanc.

Lorsque le nœud chef (ou leader) reçoit le message « NUMOFBLACKNEIGHBORS » de tous ses voisins gris, il démarre la phase 2 en diffusant le message « M ». Un nœud est « prêt » à être exploré s'il n'a pas de voisins blancs.

L'algorithme utilise un « arbre de Steiner » pour connecter tous les hôtes noirs générés lors de la phase 1 en choisissant ces sommets gris qui se connectent à de nombreux voisins noirs. Un sommet noir sans aucun dominateur est actif. Au départ de de cette deuxième phase, aucun sommet noir n'a de dominateur et tous les hôtes sont inexplorés. Le message M contient un champ qui spécifie l'hôte suivant à explorer.

Un sommet gris avec au moins 1 voisin noir actif est efficace. Si M est construit par un sommet noir, son champ suivant contient l'identifiant du voisin gris inexploré qui se connecte au nombre maximum d'hôtes noirs actifs. L'ensemble des nœuds noirs forment un CDS.

1. Le degré effectif d'un sommet est le nombre total de voisins blancs
2. Le message « DEGREE » contient le degré effectif actuel de lexpéditeur

3.4. Algorithme de Ben-Othman et al.

Proposé dans [25], il s'agit d'un algorithme auto-stabilisant de construction de $LWCDS$ axé sur la sauvegarde de l'énergie des nœuds d'un réseau de capture sans fil. Cet algorithme s'exécute en trois principales étapes :

La première étape concerne la construction d'un arbre couvrant basé sur le poids des nœuds , dans lequel le chemin dans le graphe entre la racine et tout nœud quelconque est le chemin avec le poids minimum. Chaque nœud stocke alors le poids du son chemin vers son parent.

La deuxième étape de l'algorithme démarre à la fin de la construction de l'arbre couvrant. Cette deuxième étape consiste en la formation d'un DS (ou Dominating Set) qui se fait suivant le principe suivant : Un nœud fait partie du DS si tous ses voisins l'ont élu comme étant « Dominant [1] ». Au debut de cette phase tous les nœuds peuvent être « dominant », mais à la fin de la phase, seule les nœuds avec le plus faible poids dans son voisinage demeure dominant. A l'issue de cette étape, nous obtenu un DS, connecter ou partiellement connecté ou non connectés du tout, constitué de nœud avec un chemin local de poids, dans lequel un nœud appartient soit au $LWDS$ ou soit il est voisin avec des nœuds dont les chemins de poids élevé que le sien qui appartiennent au $LWDS$.

Enfin, la dernière étape de cet algorithme consiste à sélection les nœuds pères des nœuds dominants, appélés « Connecteur » pour faire partie du backbone.

3.5. Algorithme de Bessaoud

Proposé dans [2], cet algorithme construit un CDS avec des nœuds spécifiques et/ou des caractéristiques de liens comme un CDS avec des nœuds éco-énergétiques, un CDS avec des liens fiables ou un CDS avec des nœuds éco-énergétiques et des liens sécurisés.

L'algorithme utilise une structure d'arborescence construite à partir d'un nœud racine. La sélection du nœud racine dépend des applications ou de l'environnement. Il peut être le puits dans le cas des réseaux de capteurs sans fil ou de tout autre nœud qui est sélectionné via un algorithme d'auto-stabilisation d'élection de leader.

L'exécution de l'algorithme se fait en deux étapes : la construction du spanning tree [2]

1. Un nœud est élu comme « Dominant », lorsqu'il le plus faible poids local parmi tous ses voisin et qu'il n'a pas de parent dominant.

2. Le spanning tree (ou arbre couvrant) est un arbre dans lequel, le chemin entre n'importe quel nœud u et la racine r est le chemin avec le minimum poids agrégé dans le graphe

et la construction du CDS.

Pour la construction du CDS, l'algorithme sélectionne les nœuds avec le minimum poids agrégé localement. Un nœud u appartient au CDS s'il n'est pas une feuille de l'arbre, c'est-à-dire s'il est le père d'au moins un nœud dans son voisinage. Le CDS final est obtenu par la suppression des feuilles de la dorsale. Les nœuds à l'extérieur de la dorsale sont à un saut de la dorsale car au moins leurs pères sont dans la dorsale.

4. Conclusion

Dans ce chapitre, nous avons étudier les principales structures virtuelles pouvant être dans un MANETs telles que les clusters, les CDS, etc pour faciliter certaines tâches complexes telles que le routage hiérarchique, le contrôle de la topologie, etc. Nous avons également étudier quelques algorithmes de construction de CDS.

Dans un contexte de mobilité, la construction et la maintenance d'un backbone dans un MANET est un grand défie à relevé. Les nœuds d'un MANET sont constamment en mouvement, ce qui peut parfois casser partiellement ou entièrement le backbone et provoquer ainsi une rupture de communication temporaire avec certains nœuds du réseau ou entre tous les nœuds du réseau, réduisant ainsi les performances générales du réseaux. Pour étudier l'impactes de la mobilité sur la construction et la maintenance du backbone dans un MANET par notre algorithme nous avons choisi de concevoir notre propre simulateur MANET implémentant le modèle de mobilité de « marche aléatoire (décris à la section II.3.2.2.1.1.) », afin de pouvoir évaluer les performances de notre algorithmes.

Dans le chapitre suivant, nous allons faire l'analyse et la conception, à l'aide du langage de modélisation, d'un simulateur que nous utiliserons ensuite pour tester et évaluer les performances de notre algorithme de construction de backbone par rapport à l'algorithme de construction de backbone pour MANET de Jalel Ben-Othman et al. (décris à la section II.3.4.)

Modélisation et conception

1. Introduction

L'objectif de ce chapitre est de faire une modélisation du simulateur que nous allons utilisés pour évalué les performances de notre algorithme de construction de cds, qui sera décrit au chapitre « Réalisation de l'application (IV.) », en présentant les étapes fondamentales de son évolution et de son développement.

Le recours à la modélisation est depuis longtemps une pratique indispensable au développement d'application, car un modèle est prévu pour anticiper les résultats du développement.

Un modèle est en effet une abstraction du résultat, dont le but est de documenter, de prévoir, d'étudier, de collecter ou d'estimer les informations d'un système. Associé au processus de développement, un modèle représente la vue sur une spécification ou sur une solution de système, pris à un niveau de détail pertinent pour exprimer ou concevoir la cible de l'étape en cours.

Il est en effet impossible de produire un modèle représentant quelques milliers de lignes de code sans passer par les étapes d'avancement qui permettent d'organiser judicieusement le volume d'informations collectées. Ces différentes étapes forment le ńProcessus Unifié ż, car il regroupe les activités à mener pour transformer les besoins d'un utilisateur en un système logiciel quelque soit la classe, la taille et le domaine d'application de ce système.

Le processus unifié utilise et vient compléter le langage UML (Unified Modeling Language), qui est un langage de modélisation très complet permettant de modéliser un problème de façon standard.

Dans ce chapitre, nous allons d'abord présenter le processus unifié, ensuite nous reviendrons sur la présentation d'UML dont quelques diagrammes seront utilisés dans certaines phase et activité du processus unifié.

2. Modélisation du simulateur

La modélisation objet décompose les systèmes en objets collaborants, formés par des données et des fonctions qui opèrent sur ces données. La notion de classe est utilisée pour saisir la structure et le comportement des objets [31]. La partie des données d'une classe est définie par un ensemble d'attributs et la partie fonctionnelle par un ensemble d'opérations, appelées méthodes.

2.1. UML

UML (Unified Modeling Language) [32] est une méthode de modélisation visuelle comme moyen de conception et/ou de création de logiciels orientés objet. La notation UML est un langage visuel constitué d'un ensemble de schémas, appelés « diagrammes », qui fournissent chacun une vision différente du projet à traiter.

La modélisation du système en UML [31] est structurée autour de plusieurs diagrammes qui sontutilisés aux différentes étapes du développement. Il définit cependant neuf types de diagrammes, dont les plus utilisés sont résumer par la figure III.1.

Ces diagrammes se complètent pour modéliser un système, et sont regroupés dans deux grands ensembles :

- Les diagrammes statiques (ou structurel)
  - ✓ **Les diagrammes de classes**, qui expriment de manière générale la structure statique d'un système, en termes de classes et de relations entre ces classes.
  - ✓ **Les diagrammes de cas d'utilisation**, qui décrivent la façon dont le système sera utilisé. Ils montrent les relations entre les acteurs (ou agents externes au système à concevoir) et les cas d'utilisation du système.
  - ✓ **Les diagrammes de composants**, qui sont utilisés pour modéliser les différents composants du système et leurs relations. Ces composants peuvent être des modules, des sous-programmes, des tâches et des sous-systèmes.
  - ✓ **Les diagrammes de déploiement**, qui montrent l'organisation des composants matériels et le rattachement du logiciel aux dispositifs matériels.
  - ✓ **Les diagrammes d'objets**, qui sont des instances des diagrammes de classes, utilisés pour présenter un contexte particulier du problème.

- Les diagrammes dynamique (ou comportemental)
  - ✓ **Les diagrammes de séquence**, qui montrent l'interaction entre plusieurs objets organisés dans le temps. Cette interaction est un ensemble de messages échangés entre les objets pour effectuer une opération ou obtenir un résultat.
  - ✓ **Les diagrammes d'activités**, qui sont des variantes des diagrammes d'états, organisés par rapport aux actions (ou opérations). Ils sont utilisés pour représenter le comportement interne d'une méthode ou d'un cas d'utilisation.
  - ✓ **Les diagrammes d'état-transition**, qui montrent le comportement des classes. Ce sont des automates hiérarchiques, permettant une certaine représentation du parallélisme.
  - ✓ **Les diagrammes de collaboration**, qui présentent l'interaction organisée autour des objets et de leurs liaisons. A la différence des diagrammes de séquence, ils ne montrent pas le temps comme une dimension séparée. Ces diagrammes, ainsi que les diagrammes de séquence, sont appelés diagrammes d'interaction.

Enfin UML possèdes des points forts et des points faibles qui sont :

✔ **Point fort**

- ✓ C'est un langage formel et normalisé clair et précis [33] ;
- ✓ C'est un support de communication performant qui facilite l'analyse et la compréhension de représentations abstraites complexes [33] ;
- ✓ Il possède une sémantique propre, une syntaxe composée de graphe et permet de construire plusieurs modèles.

✔ **Point faible**

- ✓ La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation ;
- ✓ Ne constitue qu'une étape (n'est pas à l'origine des concepts objets [33].

Nous avons utilisé les diagrammes suivants pour modéliser notre système :

✓ Les diagrammes de cas d'utilisation.

✓ Les diagrammes de classe.

✓ Les diagrammes de séquence.

✓ Les diagrammes de collaboration.

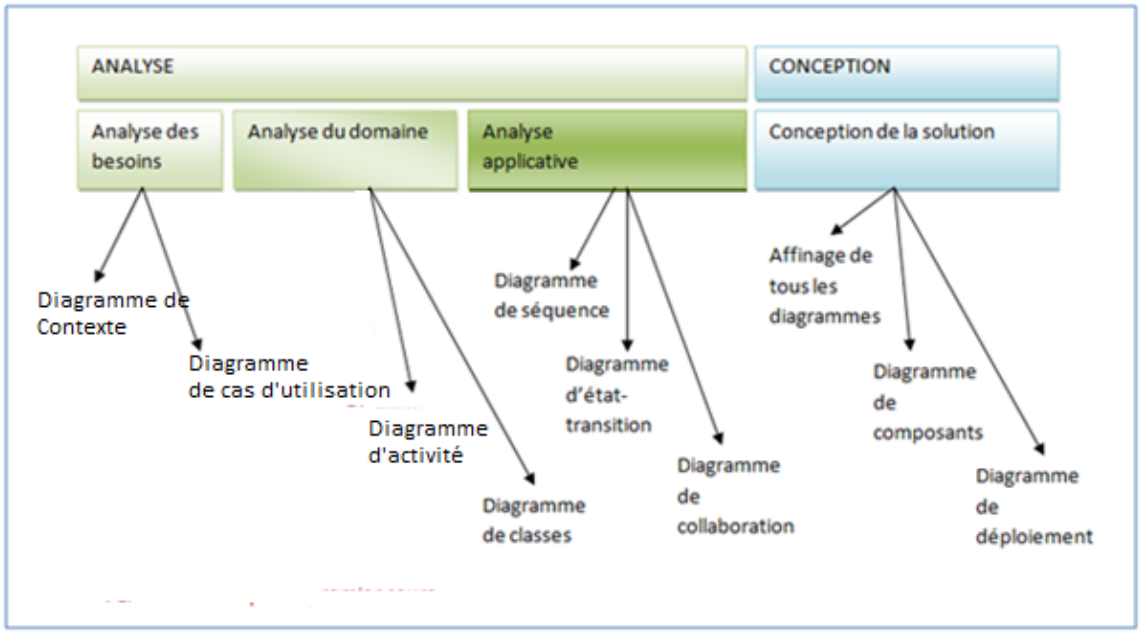


FIGURE III.1 – Modélisation UML : Les différents types de diagramme

2.1.1. Diagrammes de l'application

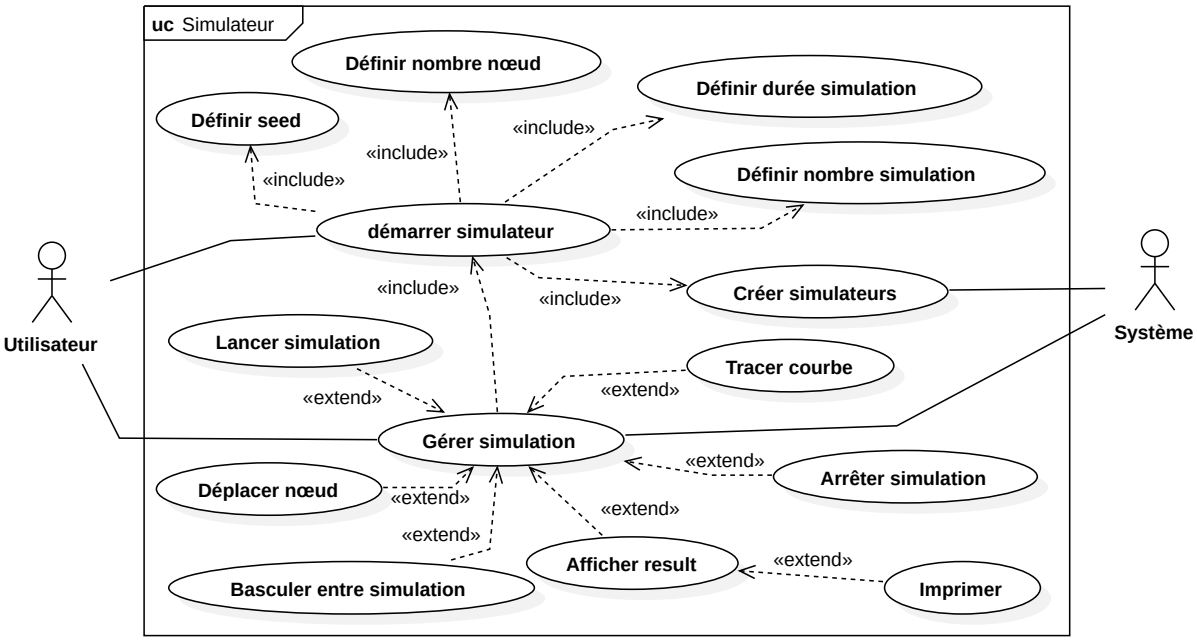


FIGURE III.2 – Diagramme de cas d'utilisation - Global

2.1.1.1. Démarrer simulateur

- **Diagramme de cas d'utilisation :** La figure III.3 donne une illustration des cas d'utilisation de la procédure de démarrage du simulateur.

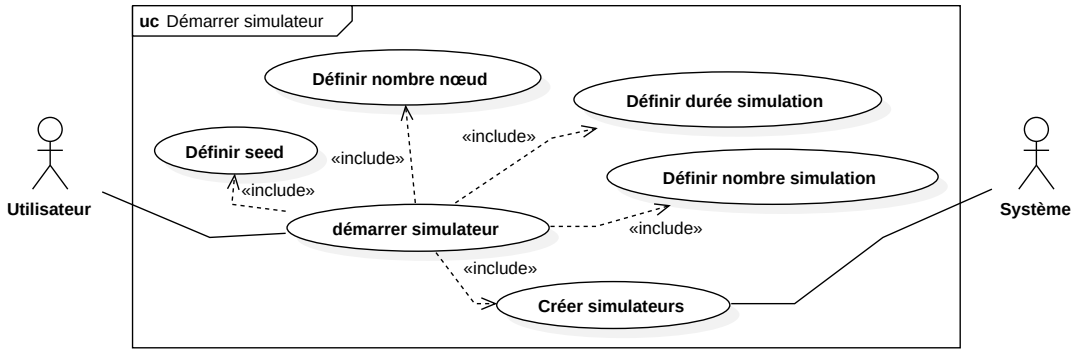


FIGURE III.3 – Diagramme de cas d'utilisation - Démarrer simulateur

- **Acteur principal :** Utilisateur/Système.
- **Objectif :** Lancer le simulateur en définissation les paramètre de simulation.
- **Scénario nominal :**
  1. L'utilisateur decide de lancer une ou plusieurs simulation.
  2. L'utilisateur saisi la valeur du seed [1], le nombre du nœuds , le nombre de simulation et la durée de chaque simulation.
  3. L'utilisateur clique ensuite sur valider pour démarrer le simulateur.
  4. Le système démarre le simulateur.
- **Scénario alternatif :**
  1. L'utilisateur oublie ou decide de ne pas fournir la valeur du seed, ou le nombre de nœuds , ou le nombre de simulation ou la durée d'une simulation.
  2. Le système affiche un message d'erreur et invite l'utilisateur à fournir les valeur obligatoire manquantes.

- **Diagramme de séquence :** La figure III.4 donne une illustration séquentielle de la procédure de démarrage du simulateur.

1. Un seed ou « graine » en français (graine aléatoire ou un état de graine) est un nombre (ou un vecteur) utilisé pour initialiser un générateur de nombres pseudo-aléatoires.

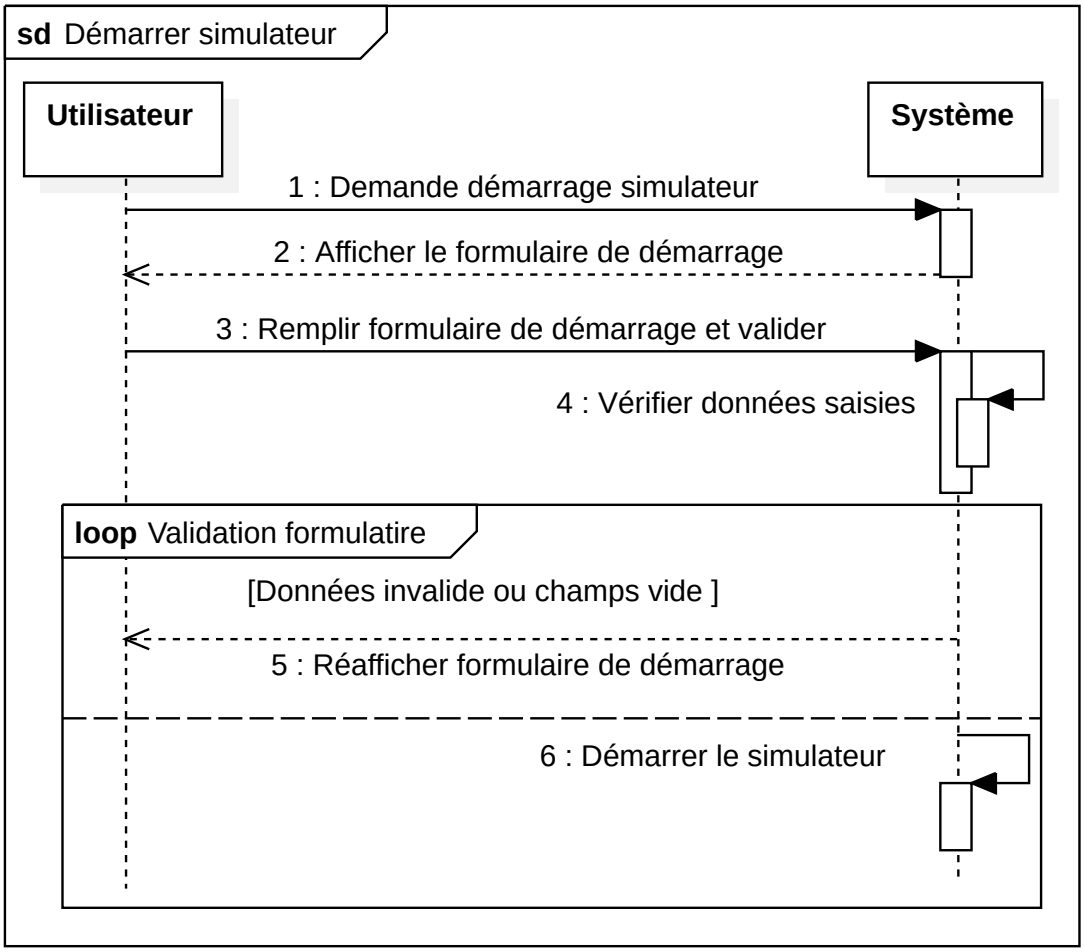


FIGURE III.4 – Diagramme de séquence - Démarrer simulateur

2.1.1.2. Gérer simulation

- **Diagramme de cas d'utilisation :** La figure III.5 donne une illustration des cas d'utilisation de la procédure de gestion du simulateur.

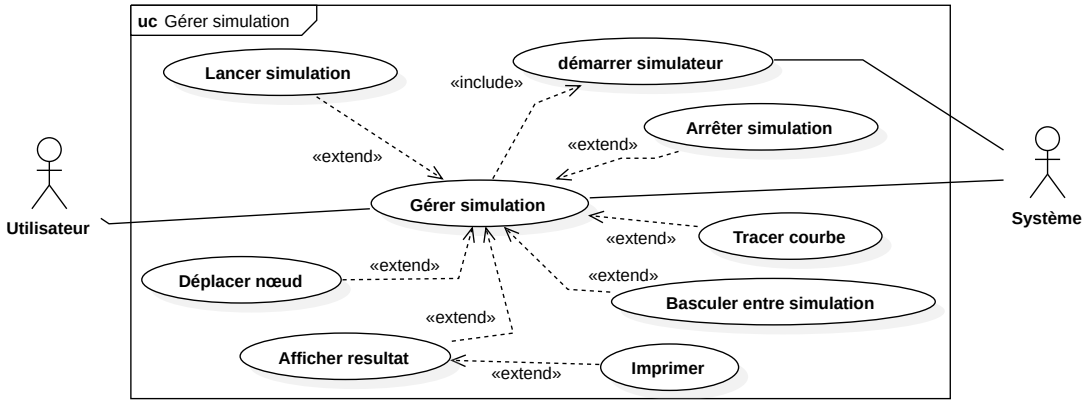


FIGURE III.5 – Diagramme de cas d'utilisation - Gérer simulation

- **Acteur principal :** Utilisateur/Système.

- **Objectif :** Gérer la simulation courante (visible à l'écran).
- **Scénario nominal :**
  1. L'utilisateur décide de lancer la simulation en cliquant sur « Valider » après avoir rempli le formulaire de démarrage.
  2. L'utilisateur [1] ou le système [2] décide d'arrêter la simulation.
  3. L'utilisateur [3] ou le système [4] décide de déplacer un nœud .
  4. L'utilisateur [3] ou le système [2] décide de basculer vers une autre simulation lorsqu'il y a plusieurs simulation à faire.
  5. Le système trace la (les) courbe(s) de variations des paramètre suivie (pourcentage cds, nombre de nœuds dominant [5], etc.) à la fin de la (ou des) simulations.
  6. L'utilisateur d'afficher ou d'imprimer dans un fichier (pdf ou text) les résultat d'une (ou plusieurs) simulation(s).
- **Scénario alternatif :**
  1. La simulation courante échoue [3], le système bascule automatique à la simulation dans le cas où il y'a plusieurs simulation à faire.
  2. Le simulateur crashe [6], le système s'arrête.

- **Diagramme de séquence :** La figure III.6 donne une illustration séquentielle de la procédure de gestion du simulateur.

1. Pour interrompre la simulation ou pour une raison quelconque.
2. Lorsque la durée de la simulation est atteint.
3. Pour une raison quelconque
4. Par le modèle de mobilité
5. nœuds coloré en noir
6. Suite à un plantage du système ou une saturation de la mémoire (dans ce cas le système d'exploitation arrête le simulateur).

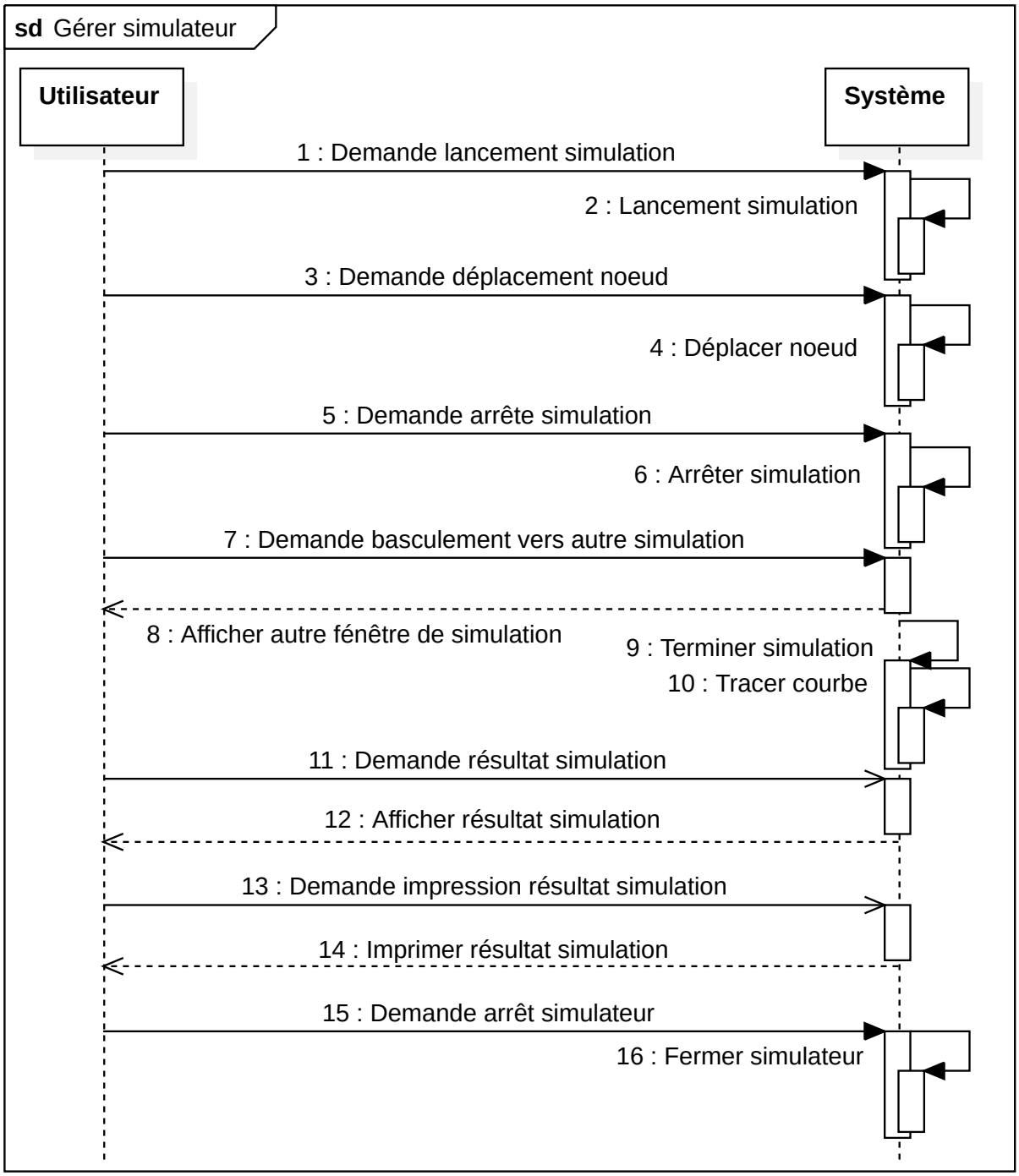


FIGURE III.6 – Diagramme de cas de séquence - Gérer simulation

2.1.1.3. Diagramme de classes La figure III.7 donne une illustration séquentielle de la procédure de gestion du simulateur.

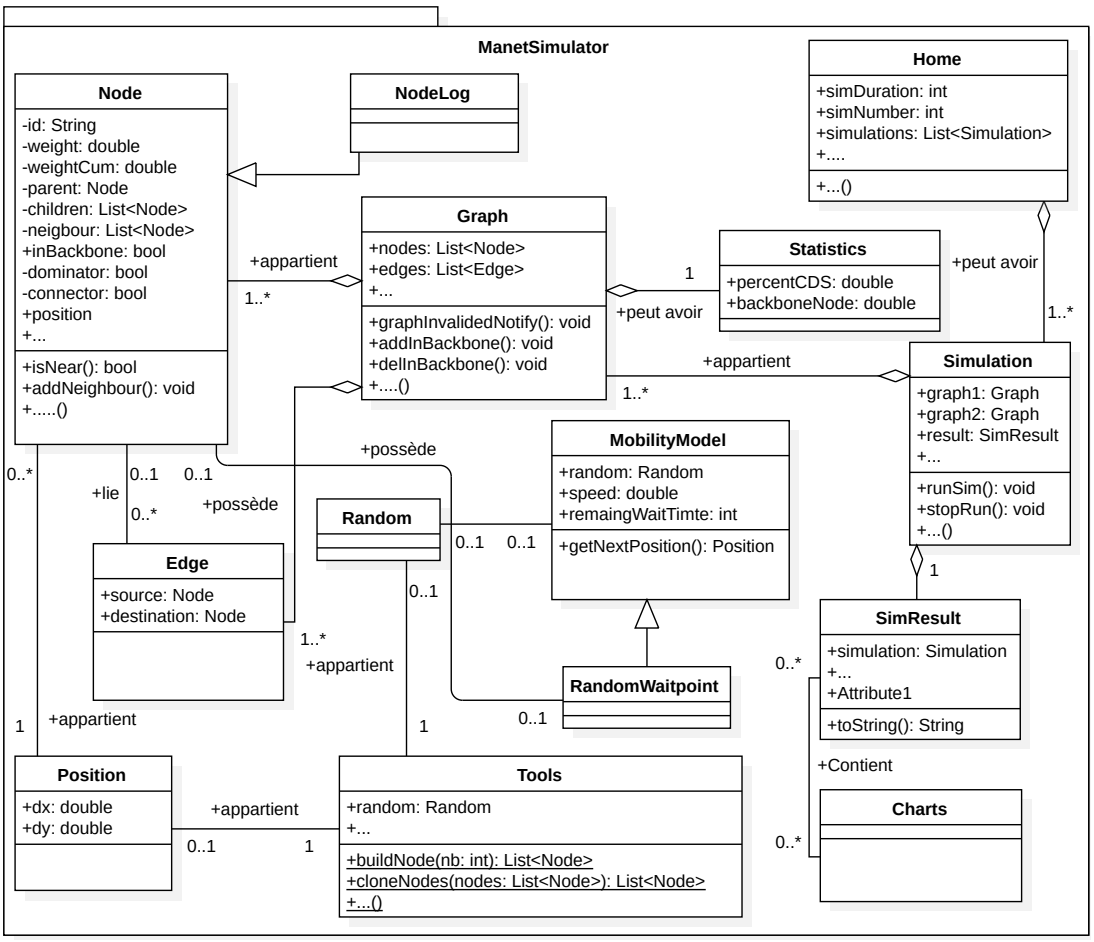


FIGURE III.7 – Diagramme de classes

2.2. Modèle du système

Dans la section I.3. de ce mémoire, nous avons montré qu'un MANET peut être modéliser par un graphe de disque dans lequel les nœuds (ou sommets) de ce graphe répresente les processus du réseau et les arêtes (ou arcs), les liens de communication. Nous avons supposé notre système comme un UDG, dans le cadre de l'implémentation de notre algorithme. Cependant, Notre système peut aussi bien fonctionner comme un UDG qu'un DGB.

2.2.1. Contexte de l'application

Notre projet consiste à proposé un algorithme de construction de CDS pour gérer le routage dans un MANET. Nous avons pour cela décider de concevoir un simulateur capable de simuler un MANET et ainsi tester et évaluer les performances de notre algorithmes. L'algorithme et le fonctionnement général du simulateur seront présentés dans le chapitre IV de ce mémoire.

2.2.1.1. Besoins fonctionnels Notre simulateur permet de simuler un MANET. Pour ce faire, il a besoin de certaines données (le seed, le nombre de simulation, le nombre de nœuds et la durée d'une simulation), l'utilisateur, doit préciser ses valeurs pour pouvoir lancer le simulateur.

- ✓ Un seed [1] est un nombre (ou un vecteur) utilisé pour initialiser un générateur de nombres pseudo-aléatoires. Fixer la valeur du seed, permet de générer les mêmes séries de nombre aléatoire dans plusieurs simulation. Cela est utile, lorsque nous souhaitons comparer les performance de plusieurs algorithme, car il permet de mettre tous les algorithmes dans les mêmes conditions d'exécutions.
- ✓ Le nombre de nœuds est utilisé pour définir simuler la taille du réseau.
- ✓ Le nombre de simulation, permet de juger de la performance d'un algorithme par rapport à un autre, car nous pourrons en effet calculer une moyenne de performance de chaque algorithmes dans les différentes simulations.
- ✓ La durée d'une simulation, permet l'exécution infini d'une simulation et de récueillir les résultats à la fin de cette durée.

2.2.1.2. Besoins non fonctionnels Les besoins non fonctionnels décrivent toutes les contraintes auxquelles est soumis le système pour sa réalisation et son bon fonctionnement.

- ✓ L'ordinateur hôte doit disposer d'un bon processeur et une bonne capacité mémoire ;
- ✓ Ergonomie et souplesse : l'application doit offrir une interface conviviale et ergonomique exploitable par l'utilisateur en envisageant toutes les interactions possibles à l'écran du support tenu ;
- ✓ L'application doit optimiser les traitements et l'utilisation de ses ressources(cpu, mémoire disponible) afin de ne pas perturber l'exécution des autres programmes sur l'ordinateur hôte ou ralentir ce dernier ;
- ✓ Rapidité du traitement : l'application doit assurer une rapidité de traitement pour qu'elle s'approche au maximum d'une exécution en temps réel.

2.2.2. Mobilité des nœuds dans le simulateur

Les nœuds d'un MANET sont mobiles. Nous devons donc tenir compte de la mobilité des nœuds dans la construction du backbone par notre simulateur afin de maintenir la

1. ou « graine » en français (graine aléatoire ou un état de graine)

stabilité de ce dernier. La mobilité des nœuds dans le simulateur fonctionnement suivant le modèle de mobilité « Random Way Point » décris à la section 3.2.2.1.1. du chapitre I. Les deux paramètres essentiels dans ce modèle de mobilité sont la vitesse et le temps d'attente (ou temps de pause). Dans notre simulation, la vitesse et le temps de pause sont corrélés au poids du nœuds .

3. Conclusion

Dans ce chapitre, nous avons effeectué la modélisation de notre simulateur. La conception a été faite en utilisant le langage de modélisation UML. Les différents diagrammes de cas d'utilisation (Use Case), de séquences et de classes ont été présentés pour comprendre le déroulement des différentes tâches effectués par l'application. Nous avons ce chapitre par la présentation d'un model de notre simulateur.

Une fois la conception faite, nous arrivons à la phase d'implémentation de l'application qui s'appuyera sur la conception faite dans ce chapitre.

Réalisation de l'application

1. Introduction

Dans ce chapitre, nous allons dans un premier temps présenter notre environnement de développement ainsi que les technologies et les langages de programmation que nous avons utilisé, ensuite nous entamerons la dernière activité du « Processus Unié » du Langage de Modélisation Unifié (UML), présenter dans le chapitre précédent, qui se composé de deux parties (Implémentation et Test), dans le but d'obtenir un simulateur capable de simuler un MANET et nous permettre ainsi, d'évaluer les performances de notre algorithme. Nous ferons ensuite une description détaillé de notre algorithme de construction de backbone (CDS) et de son principe de fonctionnement. Nous terminerons ce chapitre par une présenterons ensuite notre application final à travers des capture d'écran de ses différentes interface.

2. Environnement de développement

Pour le développement de notre simulateur, nous avons choisi d'utiliser un framework [2] de développement d'applications multi-plateforme appélés « Flutter [3] ». En effet, depuis bien longtemps déjà, les programmes développés pour Windows ne suffisent plus ; aujourd'hui, pour s'imposer, les application doivent pouvoir fonctionner sur des smartphones Android, sous iOS et, mieux encore, dans un navigateur web.

Pour y arriver, il fallait jusqu'ici ajuster et compiler un nouveau code de programmation pour chacune de ces plateformes. Dans de nombreux cas, il fallait même parfois procéder à des modications de taille pour conférer à ses applications des particularités propres à chaque système d'exploitation et reproduire l'aspect et les fonctionnalités des interfaces utilisateurs usuelles. Flutter simplie le développement multi-plateforme en mettant à disposition une base de code commune pour toutes les plateformes.

2. Un Framework est, comme son nom l'indique en anglais, un cadre de travail. L'objectif d'un Framework est généralement de simplier le travail des développeurs informatiques (les codeurs si vous préférez), en leur offrant une architecture prête à l'emploi et qui leur permette de ne pas repartir de zéro à chaque nouveau projet.

3. Flutter est un framework de développement d'applications multi-plateforme, conçu par Google, dont la première version a été publiée sous forme de projet open source à la n de l'année 2018.

3. Présentation du système

Un MANET est un système distribué constitué d'un ensemble de processus déployés sans aucune infrastructure et basés sur la technologie de communication sans fil [2]. Dans un système distribué, chaque nœud a un état local formé par son ensemble de variables.

3.1. Définitions

- **Système distribué** : Un système distribué [2] est un ensemble de processus qui échangent des messages via un système de communication selon des règles définies par le modèle de communication utilisé. Dans un système distribué, chaque nœud a un état local formé par son ensemble de variables.
- **Algorithme distribué** : Un algorithme distribué [2] est exécuté localement par chaque nœud du système distribué, il vise à effectuer une tâche globale en utilisant des informations locales.

3.2. Description de l'algorithme

3.2.1. Construction du backbone

Les algorithmes de constructions de backbone [2] proposés dans la littérature se concentrer le plus souvent sur la taille du backbone, c'est-à-dire le nombre de nœuds du backbone (nœuds dominants). Notre algorithme suit également cette logique sans toute que cela soit une contrainte majeure au déroulement de l'algorithme.

Notre algorithme est conçu pour être utilisé dans un context où les nœuds sont pondérés. Il est basé sur une structure arborescente dont la construction debut à partir d'un nœud racine. Cette pondération est utlisé par notre simulateur pour gérer la mobilité. Le principe de la mobilité des nœud dans notre simulateur sera détaillé un plus bas dans le document.

Notre algorithme s'exécute en deux phase. Il commence d'abord par construire un arbre couvrant [1] dans lequel, les nœuds avec un poids élevé son renvoyer au feuilles. Ensuite il séléctionne le nœud de l'arbre qui sont des parents d'autres nœuds pour faire parti du

1. Un arbre couvrant d'un graphe non orienté et connexe est un arbre inclus dans ce graphe et qui connecte tous les sommets du graphe.

backbone.

Notre algorithme utilise un système de coloration pour distingués les nœuds fils des nœuds parents. Les nœuds fils sont colorés en gris, tandis que les nœuds parents sont colorés en noir.

Ce système de coloration est gérer par un système d'échange de messages comportant deux types de messages « weight (poids) » et « parent » :

- **Message de type weight** : Ce type de message est envoyé par tous les nœuds à leur voisin à un saut de façon périodique ou lorsqu'un nœud change de parent.
- **Message de type parent** : Ce type de message est envoyé par un nœud au nœud qu'il a choisi comme parent pour notifier à ce dernier qu'il lui a choisi comme parent. Le nœud destinateur de ce message, c'est à dire celui qui a été choisi comme parent, se colore alors en noir, tandis que le nœud émetteur se colore en gris.

Initialement chaque nœud calcul localement son parent en se basant sur les messages réçu (périodiquement) de ses voisins à un saut. La procédure du calcul du parent est donnée par l'algorithme 1.

Une fois que le nœud a calculer son parent, il se colore en gris et envoie un message de type « parent » au nœud qu'il a choisi comme parent. Il calcul ensuite son poids cumulé, c'est à dire la somme de son poids avec celui de son parent, et le diffuse à tous ses voisins à un saut.

Lorsqu'un nœud reçoit un message de type « weight », il exécute l'algorithme 1 pour avoir un parent temporaire (*tmp*). Il calcule un nouveau poids cumulé (temporaire) avec le poids de ce nouveau parent. Il compare ensuite ce nouveau poids cumulé avec son poids cumulé courant. Si cette nouvelle valeur est inférieur à l'ancienne, le nœud met alors à jour son parent avec avec le parent tmp. Il envoie un message de type « parent » à *tmp*, puis diffuse son nouveau poids cumulé à tous ses voisins à un saut. Lorsqu'un nœud réçoit un message de type « parent », il se colore en noir et met à jour sa liste de fils avec l'émetteur de ce message.

La première phase de l'exécution de l'algorithme prend fin, lorsque tous les nœuds ont choisi un parent et qu'il n'y a plus aucune évolution dans le graphe. Une fois l'arbre construit,l'algorithme choisi les nœuds noirs pour faire parties du backbone. La figure IV.1 illustre la construction du backbone par notre algorithme.

Algorithm 1 Calcul du parent

```
1 : procedure CalParent()
2 :     Debut
3 :     if neighbour.length > 0 then
4 :         parent ← 1                          ▷ Nous supposons le voisin 0 comme parent
5 :         i ← 0                                                          ▷ Compteur
6 :         while i < neighbour.length do
7 :             weightCum1 ← parent.weightCum + this.weight
8 :             weightCum2 ← neighbour[i].weightCum + this.weight
9 :             if weightCum2 < weightCum1 then
10 :                parent ← neighbour[i]
11 :            i ← i + 1
12 :        return parent                                  ▷ Retourne le nouveau du parent
13 :    return null    ▷ Le nœud n'a aucun voisin. e.g, il n'est pas connecté au reste du
    graphe
14 :    Fin
```

3.2.2. Context de mobilité

Lorsqu'un nœud se déplace, suivant sa position dans l'arbre, il peut parfois causer la rupture partielle ou total du backbone. Dans ce cas, tous les nœuds adjacents à ce nœud avant son mouvement ré-exécute localement notre alogrithme de construction de CDS (décris à la section 3.2.1. de ce chapitre) et diffuser leur poids cumulé à leur voisins qui font de même jusqu'à réformer à nouveau le backbone. Lorsque le nœud en déplacement atteint sa destination, il exécute à son tour l'algorithme et diffuse ses nouvelles données à ses nouveaux voisins, qui ré-calcule, si nécessaire, leur parent.

3.3. Gestion de la mobilité

La gestion de mobilité des nœuds par la procédure de gestion de la mobilité dans le simulateur est régit par le principe suivant : "A chaque instant t de la simulation en cours, l'algorithme de mobilité (algorithme 2) calcul pour chaque nœuds un nombre aléatoire et compare la valeur de ce nombre à la valeur du poids du nœud . Si cette valeur est inférieur à la valeur du poids du nœud alors le nœud est considéré à cet instant comme mobile par l'algorithme de mobilité et traiter en conséquence. Cette action est répété ainsi jusqu'à la fin de la simulation." Au lancement d'une simulation, la procédure de gestion de la mobilité des nœuds n'est lancé qu'une fois que le backbone est entièrement construite pour la première fois. Son exécution se poursuit alors jusqu'à la fin de la simulation.

Algorithm 2 Mobilité des nœuds suivant le modèle de mobilité Random Waypoint

```
1 : procedure getNextPosition()
2 :     Déclaration
3 :     int minWaitTime
4 :     int maxWaitTime
5 :     int minSpeed
6 :     int maxSpeed
7 :     double move
8 :     Position nextPosition
9 :
10 :    Initialisation
11 :
12 :    move ← random.nextDouble()
13 :    t1 ← AppConfig.minWaitTime
14 :    t2 ← AppConfig.maxWaitTime
15 :    v1 ← AppConfig.minSpeed
16 :    v2 ← AppConfig.maxSpeed
17 :
18 :    Debut
19 :    if move < node.weight || node.remainingWaitingTime >= 0 then
20 :                ▷ Le temps d'attente du nœud est pas écoulé ou le nœud a atteint sa
    destination
21 :        if node.endMove && node.remainingWaitingTime < 0 then
22 :            node.speed ← intInRange(min : v1, max : v2)
23 :                                        ▷ Retourne la nouvelle destination du nœud
24 :            node.remainingWaitingTime = intInRange(min : t1, max : t2)
25 :                                          ▷ Calcul de la nouvelle position du nœud
26 :            nextPosition ← getNextWayPoint()
27 :                                        ▷ Retourne la nouvelle destination du nœud
28 :            return nextPosition
29 :                                    ▷ Le temps d'attente du nœud n'est pas écoulé
30 :        else if node.remainingWaitingTime > 0 then
31 :                                    ▷ Le nœud attend l'échéance du temps d'attente
32 :            node.remainingWaitingTime ← node.remainingWaitingTime − 1
33 :
34 :            return node.position                ▷ Le nœud reste à la même position
35 :    return node.position                        ▷ Le nœud reste à la même position
36 :
37 :    Fin
```

3.4. Calcul des nœuds du backbone

Au cours d'une simulation, les nœuds du backbone (nœud dominant colorié en noir) sont calculés en temps réel de simulation à l'aide d'une procédure récursive du programme de simulation nommée « getBackboneNodes », régit par l'algorithme 3 et qui prend en paramètre une file (Queue) et l'adresse d'un compteur. Chaque nœuds du simulateur possède un attribut « isInBackbone » qui prend la valeur « true » si le nœuds est dans le backbone et « false », sinon.A la première appel de cette procédure, la racine de l'arbre [1] est enfilé dans la file et le compteur du nombre de nœuds du backbone est initialisé à zéro et la valeur 0 est placé dans l'attribut « colorIndex » de tous les nœuds . La procédure « getBackboneNodes » défile ensuite cet élément pour le traiter. Le traitement d'un nœud par la procédure consiste placé 2 dans l'attribut « colorIndex » de ce nœud , et à parcourir la liste de ces voisins pour les marqués à l'aide d'un système d'indice de couleur et/ou à les comptabiliser. Le système d'indice de couleur est défini comme suit :

- 0 : si le nœud n'a pas été traiter ;
- 1 : si le nœud voisin possède la valeur « true » dans son attribut « isInBackbone » et n'a pas été marqué comme traite ;
- 2 : si le nœud a été marqué comme traiter et comptabiliser.

4. Présentation de l'application

Notre nouveau simulateur implemente deux algorithmes de construction de backbone, l'algorithme [2] proposé dans [25] (décris à la section II.3.4.), et notre algorithme [3] (décrit précedemment à la section IV.3.2.1.). Une fenêtre de simulation dans notre simulateur correspond donc, à l'exécution de ces deux algorithmes, répartit sur deux ongles, nommé respectivement dans lesimulateur par « Algorithme 1 » et « Algorithme 2 »

1. Le nœud qui a le poids le plus petit
2. Nommé « Algorithme 2 dans le simulateur »
3. Nommé « Algorithme 1 dans le simulateur »

Algorithm 3 Calcul du nombre de nœuds dans le backbone

```
1 : procedure getBackboneNodes(file, &compteur)
2 :
3 :     Debut
4 :     node ← file.removeFirst()                    ▷ Défiler le premier élément de la le
5 :     i ← 0                                        ▷ Compteur de position
6 :     while i < node.neighbour.length do
7 :         if node.neighbour[i].isInBackbone&&node.neighbour[i].colorIndex = 0 then
8 :             node.neighbour[i].colorIndex ← 1     ▷ Marqué le voisin i comme non traiter
9 :             file.add(node.neighbour[i])          ▷ Enfiler le voisin i dans la le
10 :        else
11 :            node.neighbour[i].colorIndex ← 2     ▷ Marqué le nœud défiler comme traiter
12 :            compteur ← compteur + 1              ▷ Comptabiliser le voisin i
13 :        i ← i + 1
14 :    node.colorIndex ← 2                          ▷ Comptabiliser le nœud défiler
15 :    if file.length > 0 then
16 :        getBackboneNodes(file, &compteur)
17 :
18 :    Fin
```

4.1. Interface de démarrage

Au lancement du simulateur, une popup de lancement est proposé à l'utilisateur dans lequel il est invité à fournir certaines informations dont les plus importantes et obligatoires sont le nombre de nœuds qui participe à la simulation et le nombre de simulation. Si l'utilisateur, ne précise pas la valeur du seed et/ou la durée d'une simulation, alors une valeur par défaut fixée à 0 pour le seed et une valeur par défaut [1] fixé à 100 comme durée d'une simulation seront utilisé parle simulateur. La figure IV.1 montre une illustration de la fenêtre de lancement du simulateur.

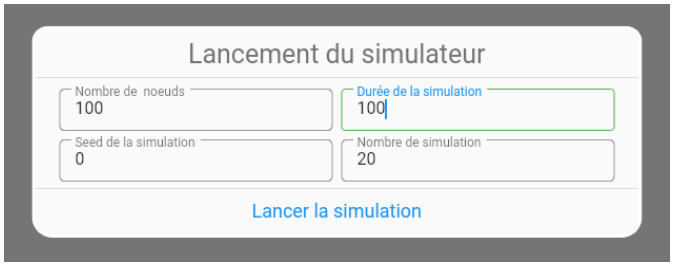


FIGURE IV.1 – Simulateur MANET - Interface de démarrage

1. en seconde.

4.2. Interface de simulation

Après la definition et la validation des paramètres de démarrage, le simulateur basculer sur une interface de simulation [1], qui varie suivant la valeur du nombre de simulation saisi par l'utilisateur, dans laquelle sont les statistiques de l'algorithme en cours de simulation sont enrégistrés et afficher en temps réel :

— **Simulation unique** :

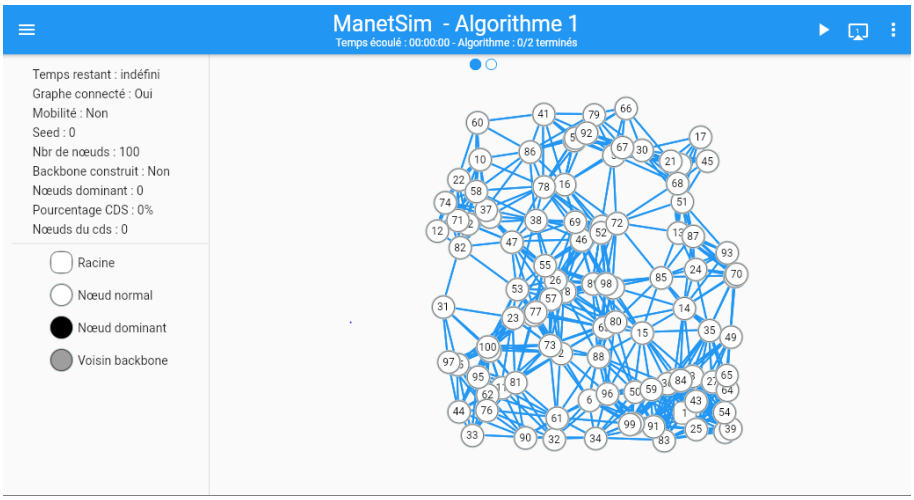


FIGURE IV.2 – Simulateur MANET - Interface de simulation unique

— **Simulation Multiple** :

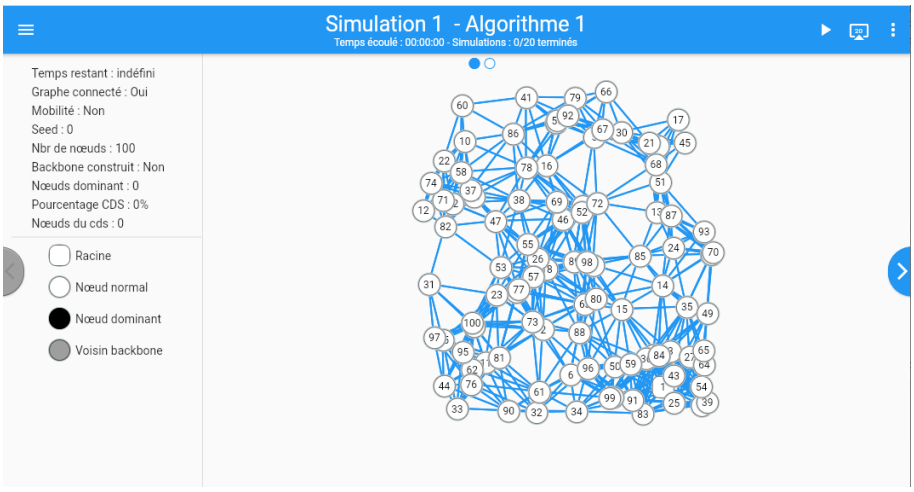


FIGURE IV.3 – Simulateur MANET - Interface de simulation multiple

1. Le simulateur est programmé pour basculer automatiquement d'un algorithme à un algorithme et d'une simulation à une autre dans le cas où il y a plusieurs simulation à faire

4.3. Interface des statistiques

A la fin d'une simulation ou d'une série de simulations les statistiques de simulations enrégistrer au cours de la simulation sont utilisé pour dessiner un graphe qui montre l'évolution des pourcentage du CDS et du nombre de nuds dans le CDS au fil du temps pour chaque algorithme à l'aide d'une série de couleur pour chaque donnée. Les figures IV.5 et IV.6 donnent une illustration des graphes d'évolutions de la simulaton au fil du temps.

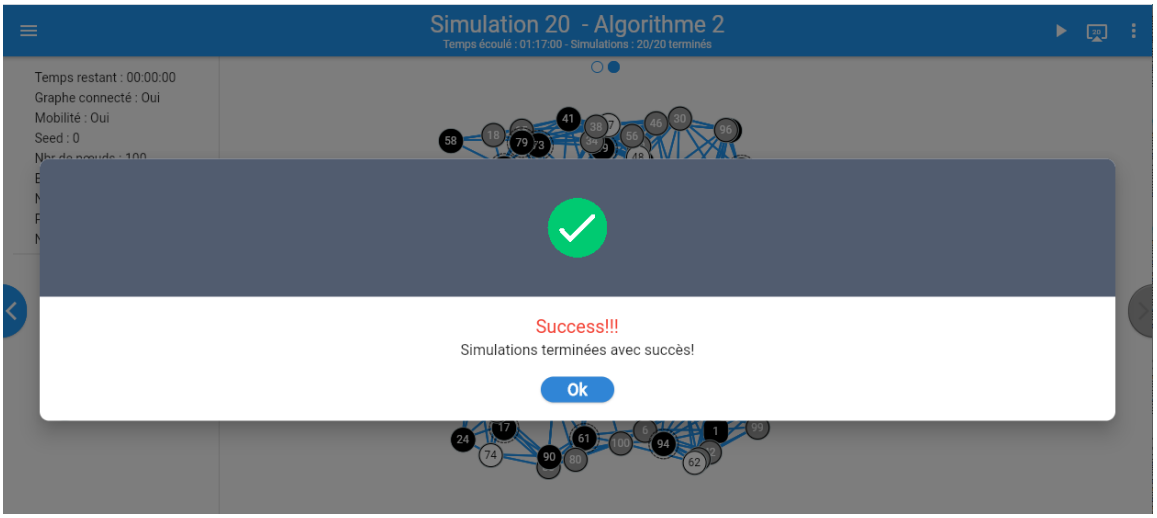


FIGURE IV.4 – Simulateur MANET - Interface fin simulation

FIGURE IV.5 – Simulateur MANET - Interface des statistiques (résultat)

FIGURE IV.6 – Simulateur MANET - Interface des statistiques (graphe)

5. Simulation et discussion

L'évalution de notre algorithme de construction de backbone a été réalisée à l'aide du simulateur [1] que nous avons conçu pour cet but. Comme nous l'avons souligner à la section 2.2. du chapitre III, un MANET peut être modéliser par un graphe de disque (UDG ou DGB) et notre système peut aussi bien fonctionner comme un UDG qu'un DGB. A cet effet, donc, nous allons décrire un scénario d'évaluation dans un UDG pour étudier l'efficacité de notre algorithme. Nous présentons ensuite les paramètres de simulation et analysons les résultats obtenue de ce scénario, puis nous discuterons des principaux avantages de notre solution. A travers ces deux scénarios expérimentaux nous allons démontrer la flexibilité de notre algorithme.

5.1. Scénario : Création d'un CDS de poids faible dans un UDG

Dans ce scénario, seuls les poids des nœuds sont prisent en compte. Un poids de nœud peut en effet refléter différents caractéristiques telles que l'énergies dans les réseaux de capteurs sans fils (WSN) ou la confiance, etc. des nœuds .

1. décris à la section 4. du chapitre IV.

5.2. Simulation et résultat

Pour évaluer notre proposition d'algorithme de construction de CDS dans un MANET, nous la comparons à l'algorithme de Ben-Othman et al. dans [25]. Pour ce faire, nous effectuons une série de 20 simulations en tenant compte de chaque paramètre de simulation. Nous supposons que les nœuds sont dispersés au hasard dans une zone carrée de 600m par côté. Nous nous assurons que le graphe obtenu est connecté pour chaque simulation.

Pour construire un graphe UDG, nous considérons la plage de transmission des nœuds comme défini dans le tableau IV.1. Les poids des nœuds sont séléctionnés dans l'intervalle $[0,1]$ en fonction de la loi uniforme. Les paramètres de simulation sont résumés dans le tableau IV.1.

Paramètres de simulation	Evaluer
Coté de la zone carré	600m
Nombre de noeud	100 à 150
Porté de transmission	45m
Durée d'une simulation	100s

Table IV.1 – Tableau des paramètres

Pour réaliser notre expérimentale, nous avons observé la variation moyenne des nœuds dominant dans le backbone, le pourcentage de temps de simulation pendant lequel le backbone est construit à 100% ainsi que la variation du pourcentage de nœuds dans le backbone.

Les résultats obtenue à la figure IV.7 de l'étude d'évaluation de notre scénario montre d'une part que la densité du réseau augmente avec la taille du réseau indépendamment de l'évaluation de la solution adaptée. D'un autre côté, notre algorithme surpasse de loin l'autre solution grâce à la construction arborescente qui isole les nœuds de poids élevé et donc les plus mobiles en feuille dans l'arbre. Ces nœuds de poids élevés ne peuvent donc jamais être séléctionnés dans le backbone.

5.3. Discussion

Afin de construire le backbone, le graphe illustrer par la figure IV.2 doit être connexe. Le déploiement des nœuds étant aléatoire, nous testons d'abord si le graphe obtenu est

connecté. L'opération de déploiement des nœuds est répétée jusqu'à ce que le graphe soit connecté.

Dans le cas de cadre de notre scénario, notre simulateur construit d'abord le CDS pour les deux algorithmes. Une fois le CDS construit pour la première fois, le simulateur lance la mobilité et commence alors à la collecter les résultats. Le processus de construction avec la mobilité s'exécute de façon permanente sans jamais s'arrêter. Néanmoins pour la comparaison des deux algorithmes, nous avons limité le temps de collecte de données à une certaines.

Moyennes des Simulations

	Algoritme 1	Algoritme 2
Moy nœuds CDS	93.87	87.65
Nœuds backbone	35.91	34.84
Nombre de nœuds	120	120
% temps cds construct	65.69	50.69
Temps simulation	100	100

FIGURE IV.7 – Simulateur MANET - Moyennes des simulations (graphe)

FIGURE IV.8 – Simulateur MANET - Moyennes des simulations (Résultat)

Dans un contexte de mobilité, notre algorithme donne de meilleur résultats par rapport à l'algorithme de BEN-OTHMAN et al. dans [25]. Car à la différence de l'algorithme de

Ben-Othman et al., notre algorithme isole les nœuds les plus mobiles en feuilles et ne leur permet donc jamais de faire partie du backbone. Ce qui permet de minimiser l'impacte du déplacement de ses nœuds sur le backbone. C'est ce qui explique les resultats obtenu à la figure IV.7.

6. Conclusion

Dans ce chapitre, nous avons commencé par présenter notre algorithme à travers une description détaillé de la manière qu'il utilise pour construire et maintenir le backbone dans un contexte de mobilité, ensuite nous avons présenté notre environnement de développement ainsi que les raisons qui ont motivé notre choix pour cet environnement. Nous avons terminé ce chapitre par la présentation du fruit de notre travail qui est un simulateur d'algorithme de construction de backbone dans un contexte de mobilité.

Conclusion générale

Le but de ce mémoire était de proposé un algorithme de construction de structure virtuelle pour la gestion du routage dans un MANET.

Dans cette perspective, nous avons commencé dans le premier chapitre par faire un bref parcours des MANETs à travers leurs historiques, leurs caractéristiques, leurs domaine d'applications, leurs types, etc., ensuite nous avons terminé étudier le concepts de mobilité dans un MANET à travers un parcours de quelques modèles de mobilité existants dans la littérature scientique.

Dans le deuxième chapitre nous nous sommes intéressés aux structures virutelles rencontrées le plus souvent dans le cadre des MANETs (ou des réseaux ad-hoc en général), ainsi algorithmes de construction de structure virtuelles telles les « backbone » et y avons fait une bref études des travaux réalisés dans ce domaine.

Dans le troisième chapitre nous avons fait la modélisation d'un simulateur de MANET, que nous avons conçu an de pouvoir tester et évaluer l'algorithme de construction de backbone que nous proposés au debut du quatrième chapitre. Nous avons terminé ce dernier chapitre par la présentation et la description du fonctionnement de notre simulateur.

Enn ce mémoire, nous a permis d'apprendre et d'acquérir de nouvelles connaissances sur les notions de structures virtuelles et les algorithmes de constructions dans le domaine des MANETs. La conception et l'implémentation d'un simulateur de MANET nous a permis d'une part de réviser et d'approfondir nos connaissance en la modélisation UML, surtout le processus unié et d'autre part à apprendre et à maîtriser le framework « Flutter » qui était un framework totalement nouveau pour nous. Grâce à ce mémoire nous pourrons développer aisément nous futur application multi-plateforme avec Flutter.

Le simulateur qu'on a conçu dans le cadre de ce mémoire peut être amélioré et enrichi de fonctionnalités an d'en faire un simulateur complèt et plus performant des MANETs. Parmi les perspectives à prendre en compte pour améliorer le simulateur, nous citons

notamment :

- L'ajout de la possibilité de pouvoir exécuter n'importe quel algorithme de construction de backbone grâce à la mise en place d'un système d'API [1] permettant aux chercheurs dans le domaine des MANETs de pouvoir ajouter, simuler et évaluer les performances de l'exécution de leur algorithme de construction de backbone ;
- L'ajout d'autres de modèles de mobilités que nous rétrouver le plus couramment dans la littérature ;
- L'ajout de la possibilité de pouvoir exporter les resultats de (ou des) simulation(s) dans plusieurs format de chiers dans le but de pouvoir appliquer des traitement spéciques à des resultats de simulation ;
- etc.

1. Une API (ou Application Programming Interface) est un ensemble de dénitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications.

Bibliographie

[1] Mohit Kumar et Rashmi Mishra. « An overview of MANET : History, challenges and applications ». In : *Indian Journal of Computer Science and Engineering (IJCSE)* 3.1 (2012), p. 121-125.

[2] Karim Bessaoud. « Self-stabilizing Algorithm for Generic Aggregated Weighted Connected Dominating Set. » In : *Adhoc & Sensor Wireless Networks* 45 (2019).

[3] Jeroen Hoebeke et al. « An overview of mobile ad hoc networks : applications and challenges ». In : *Journal-Communications Network* 3.3 (2004), p. 60-66.

[4] Saleh Yousefi, Mahmoud Siadat Mousavi et Mahmood Fathy. « Vehicular ad hoc networks (VANETs) : challenges and perspectives ». In : *2006 6th International Conference on ITS Telecommunications*. IEEE. 2006, p. 761-766.

[5] Jie Wu et Hailan Li. « On calculating connected dominating set for efficient routing in ad hoc wireless networks ». In : *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. 1999, p. 7-14.

[6] Anuj K Gupta, Harsh Sadawarti et Anil K Verma. « Performance analysis of MANET routing protocols in different mobility models ». In : *International Journal of Information Technology and Computer Science (IJITCS)* 5.6 (2013), p. 73-82.

[7] Bhavyesh Divecha et al. « Impact of node mobility on MANET routing protocols models ». In : *J. Digit. Inf. Manag.* 5.1 (2007), p. 19-23.

[8] Josh Broch et al. « A performance comparison of multi-hop wireless ad hoc network routing protocols ». In : *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. 1998, p. 85-97.

[9] Ioannis Karatzas et Steven E Shreve. « Brownian motion ». In : *Brownian Motion and Stochastic Calculus*. Springer, 1998, p. 47-127.

[10] Tracy Camp, Jeff Boleng et Vanessa Davies. « A survey of mobility models for ad hoc network research ». In : *Wireless communications and mobile computing* 2.5 (2002), p. 483-502.

[11] Ben Liang et Zygmunt J Haas. « Predictive distance-based mobility management for PCS networks ». In : *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*. T. 3. IEEE. 1999, p. 1377-1384.

[12] Xiaoyan Hong et al. « A group mobility model for ad hoc wireless networks ». In : *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*. 1999, p. 53-60.

[13] Aditya Bakshi, AK Sharma et Atul Mishra. « Significance of mobile AD-HOC networks (MANETS) ». In : *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* 2.4 (2013), p. 1-5.

[14] Ben Liang et Zygmunt J Haas. « Hybrid routing in ad hoc networks with a dynamic virtual backbone ». In : *IEEE Transactions on Wireless Communications* 5.6 (2006), p. 1392-1405.

[15] F Bai et A Helmy. « Wireless ad hoc and sensor networks ». In : *A survey of mobility models in wireless adhoc networks (Springer, 2006)* (2004).

[16] Geetha Jayakumar et Ganapathy Gopinath. « Ad hoc mobile wireless networks routing protocols–a review ». In : *Journal of Computer science* 3.8 (2007), p. 574-582.

[17] Imrich Chlamtac, Marco Conti et Jennifer J-N Liu. « Mobile ad hoc networking : imperatives and challenges ». In : *Ad hoc networks* 1.1 (2003), p. 13-64.

[18] Zygmunt J Haas et Marc R Pearlman. « The performance of a new routing protocol for the reconfigurable wireless networks ». In : *ICC'98. 1998 IEEE International Conference on Communications. Conference Record. Affiliated with SUPERCOMM'98 (Cat. No. 98CH36220)*. T. 1. IEEE. 1998, p. 156-160.

[19] Prince Samar, Marc R Pearlman et Zygmunt J Haas. « Independent zone routing : an adaptive hybrid routing framework for ad hoc wireless networks ». In : *IEEE/ACM Transactions On Networking* 12.4 (2004), p. 595-608.

[20] Thomas Clausen et Philippe Jacquet. *Rfc3626 : Optimized link state routing protocol (olsr)*. 2003.

[21] Tim Daniel Hollerung. « The cluster-based routing protocol ». In : *project group Mobile Ad-Hoc Networks Based on Wireless LAN-2003/4* (2003).

[22] Arwa Zabian, Ahmed Ibrahim et Fadi Al-Kalani. « Dynamic head cluster election algorithm for clustered Ad-Hoc networks ». In : *Journal of Computer Science* 4.1 (2008), p. 42-50.

[23] Ou Liang, Y Ahmet Sekercioglu et Nallasamy Mani. « A survey of multipoint relay based broadcast schemes in wireless ad hoc networks ». In : *IEEE Communications Surveys & Tutorials* 8.4 (2006), p. 30-46.

[24] Jean E Dunbar et al. « On weakly connected domination in graphs ». In : *Discrete Mathematics* 167 (1997), p. 261-269.

[25] Jalel Ben-Othman et al. « Self-stabilizing algorithm for efficient topology control in wireless sensor networks ». In : *Journal of Computational Science* 4.4 (2013), p. 199-208.

[26] Sudipto Guha et Samir Khuller. « Approximation algorithms for connected dominating sets ». In : *Algorithmica* 20.4 (1998), p. 374-387.

[27] Bevan Das, Raghupathy Sivakumar et Vaduvur Bharghavan. « Routing in ad hoc networks using a spine ». In : *Proceedings of Sixth International Conference on Computer Communications and Networks*. IEEE. 1997, p. 34-39.

[28] Dongxiao Yu et al. « Distributed dominating set and connected dominating set construction under the dynamic SINR model ». In : *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2019, p. 835-844.

[29] Dongxiao Yu et al. « Distributed multiple-message broadcast in wireless ad hoc networks under the SINR model ». In : *Theoretical Computer Science* 610 (2016), p. 182-191.

[30] Mihaela Cardei et al. « Connected Domination in Multihop Ad Hoc Wireless Networks. » In : *JCIS*. Citeseer. 2002, p. 251-255.

[31] Fabrice Popineau. « Introduction à la conception Orientée Objet et à UML ». In : *Supélec, École Supérieur dÉlectricité, Septembre* (2000).

[32] Ivar Jacobson, Grady Booch et James Rumbaugh. « The Unified Modeling Language ». In : *University Video Communications* (1996).

[33] *Présentation UML*. `http://www.iro.umontreal.ca/~dift6825/UML.htm`. Accessed : 2021-05-26.