REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Laboratoire d'Architecture et d'Analyse des Systèmes (LAAS)

Département d'électronique

Mémoire de projet de fin d'études

en vue de l'obtention du diplôme d'ingénieur d'état en électronique

par :

Lina BEN CHABANE

Intitulé

Modèle de confiance pour la détection d'intrusion dans les réseaux de capteurs sans fil.

Sous la direction de Djamel ADROUCHE (ENP), *du* Pr. Cherif LARBES (ENP)
et du Directeur de Recherche Philippe Owezarski (LAAS).

Présenté et soutenu le 09/09/2020

Membres du jury :

Président	**M. Adel BELOUCHRANI**	**Pr (ENP)**
Examinateur	**M. Mourad ADNANE**	**Pr (ENP)**

ENP 2020

Laboratoire d'Architecture et d'Analyse des Systèmes, 7 Avenue du Colonel Roche BP 54200 31031 Toulouse cedex 4.

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Laboratoire d'Architecture et d'Analyse des Systèmes (LAAS)

Département d'électronique

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en électronique

par :

Lina BEN CHABANE

Intitulé

Modèle de confiance pour la détection d'intrusion dans les réseaux de capteurs sans fil.

Sous la direction de Djamel ADROUCHE (ENP), *du* Pr. Cherif LARBES (ENP)
et du Directeur de Recherche Philippe Owezarski (LAAS).

Présenté et soutenu le 09/09/2020

Membres du jury :

Président	**M. Adel BELOUCHRANI**	**Pr (ENP)**
Examinateur	**M. Mourad ADNANE**	**Pr (ENP)**

ENP 2020

Laboratoire d'Architecture et d'Analyse des Systèmes, 7 Avenue du Colonel Roche BP 54200 31031 Toulouse cedex 4.

ملخص - أصبحت شبكات إنترنت الأشياء حاضرة أكثر فأكثر في حياتنا اليومية بدءًا من الأجهزة الصغيرة وحتى أكثر الأنظمة تعقيدًا. يعد أمان هذه الشبكات أحد أكبر التحديات التي يواجهها المسؤولون الذين يجب عليهم حمايتها من أي اقتحام أو هجوم.

الهدف من هذا العمل هو القدرة على اكتشاف عقدة ضارة عندما تكون موجودة في شبكة إنترنت الأشياء بناءً على الثقة التي يتم إنشاؤها بين العقد المجاورة لنفس الشبكة. هذا من خلال إظهار تأثير العقد الضارة على شبكة إنترنت الأشياء أثناء محاولة توفير طبقة إضافية من الأمان.

الكلمات المفتاحية: شبكات إنترنت الأشياء ، الأمن ، التطفل ، الهجوم ، الثقة ، العقد الخبيثة.

ABSTRACT - IoT networks are becoming more and more present in our daily lives, ranging from small devices to the most complex systems. The security of these networks is one of the greatest challenges for administrators who must protect them from any intrusion or attack.

The goal of this work is to be able to detect a malicious node when it is present in an IoT network based on the trust that is established between neighboring nodes of the same network. This by showing the impact of malicious nodes on an IoT network while trying to provide an additional layer of security.

KEYWORDS: IoT networks, security, intrusion, attack, trust, malicious nodes.

RESUME - Les réseaux IoT deviennent de plus en plus présents dans nos quotidiens allant de petits appareils aux systèmes les plus complexes. La sécurité de ces réseaux est un des plus grands défis des administrateurs qui doivent les protéger de toute intrusion ou attaque.

L'objectif de ce travail est de pouvoir détecter un nœud malveillant quand celui-ci est présent dans un réseau IoT en se basant sur la confiance qui s'instaure entre les nœuds voisins d'un même réseau. Ceci en montrant l'impact des nœuds malveillants sur un réseau IoT tout en essayant d'apporter une couche supplémentaire de sécurité.

MOTS CLES : Réseaux IoT, sécurité, intrusion, attaque, confiance, nœuds malveillants.

Remerciements

Je remercie en premier lieu Dieu, qui m'a guidé tout au long de mon chemin et qui m'a donné la force, la volonté et la patience dont j'avais besoin pour réaliser ce travail.

Je remercie aussi, le directeur du LAAS, *Liviu NICU* pour m'avoir permis d'intégrer le laboratoire pour mes six mois de travail, ainsi que toute l'équipe de LAAS : chercheurs, doctorants et stagiaires pour leur accueil, leurs aides et soutiens tout au long de mon travail.

Je remercie mon responsable de stage et directeur de recherche au sein du laboratoire, *Philippe OWEZARSKI* pour m'avoir accepté parmi ses stagiaires et pour son encadrement et ses conseils. Ainsi que le professeur *Cherif LARBES* pour avoir accepté de m'encadrer durant ce stage.

Je tiens particulièrement à remercier Mr ***Djamel ADROUCHE***, qui a été présent du début à la fin de mon travail, qui m'a accordé son temps et son soutien, ses précieux conseils et son encadrement, et sans qui ce travail n'aurait jamais vu le jour. Mes plus sincères remerciements.

Un grand merci à Mlle *Lynda SAID LHADJ* pour tous ses efforts durant mes cinq années à Polytechnique et pour m'avoir initié à ce stage.

Merci à *Rabah SADOUN* pour son aide.

Je remercie aussi les membres du jury pour avoir accepté d'examiner mon travail.

Merci à tous mes enseignants tout au long de mon parcours scolaire et mon cursus universitaire.

Pour finir, je tiens à remercier tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail. Toutes les aides m'ont été précieuses.

Dédicaces

Je dédie mon travail en premier lieu à mes parents, ces deux personnes qui ont toujours cru en moi et qui m'ont soutenu tout au long de ma vie et dans tout ce que j'entreprends. A mon père qui a su me redonner confiance en moi après chaque épreuve. A ma mère qui m'a toujours accompagnée dans mes décisions.

A mes sœurs, et mon petit frère, Lydia, Ouiza, Amel, Rachida, Katia et Belkacem. Leurs présences, leurs conseils et leurs soutiens m'ont permis de me relever après chaque mauvais coup et continuer d'avancer avec un moral d'acier.

A mon compagnon, une personne précieuse à mon cœur Smail, qui a été la dans les bons mais surtout les mauvais moments de ma vie, qui n'a jamais douté de moi et de mes capacités et qui m'a toujours poussé vers l'avant. Ses précieux conseils ont fait de moi une femme forte, sage et heureuse.

A une amie très chère à mon cœur, Toumia. Une personne que j'ai eu la chance de rencontrer et avec qui j'ai passé de merveilleux moments. Une personne toujours présente dans mes succès mais surtout mes échecs. Une personne qui m'a encouragé à relever le défi pour mon PFE et qui a eu raison de le faire.

A ma famille, et ma grande famille.

A mes amis : Amine, Ryad, Yacine et tous mes camarades.

Table des matières

Liste des tableaux

Liste des Figures

Liste des sigles et acronymes

2G : 2ème Génération

3G : 3ème Génération

AODV : Ad-hoc On-demand Distance Vector

API : Application Programming Interface

BS : Base Station

CBR : Constant Bit Rate

CH : Cluster Head

DSDV : Destination Sequenced Distance Vector Protocol

DSR : Dynamic Source Routing

DTLS : Datagram Transport Layer Security

E2E : End to End

GPS : Global Positioning System

ID : IDentification

IoT : Internet of Things

IP : Internet Protocol

LEACH : Low Energy Adaptive Clustering Hierarchy

LTE : Long Term Evolution

M2M : Machine to Machine

MiM : Man in the Middle

MPR : Multi-Point Relay

NS : Network Simulation

OLSR : Optimized link State Routing

OTcl : Object Tool Command Langage

QdS : Qualité de Service

RAM : Random Access Memory

RCSF : Réseau de Capteurs Sans Fil

RERR : Route ERRor

RF : Radio Frequency

RFID : Radio Frequency IDentification

RREP : Route REPonse

RREQ : Route REQuest

RTOS : Real Time Operating System

SDN : Software-Defined Network

SOA : Services Oriented Architecture

TAODV : Trust AODV

TCP : Transmission Control Protocol

UDP : User Datagram Protocol

WIFI : WIreless FIdelity

WSN : Wireless Sensor Network

ZHLS : Zone-Based Hierarchical Link State

ZRP : Zone Routing Protocol

Introduction générale

Depuis leur apparition, les objets connectés sont exploités dans de nombreux domaines : de la médecine (les nanorobots) aux maisons intelligentes et villes intelligentes. Cette technologie ne cesse d'évoluer et de prendre de l'ampleur et nombreux sont ceux qui migrent vers ce nouveau concept technologique innovant. Nous pouvons même dire que ces petits objets connectés ont bouleversé le cours de nos vies.

En effet, ils font partie de notre quotidien : smartphones, montre intelligente, système de détection de présence, etc. et la liste est encore longue. Ceci a poussé les plus grands constructeurs et éditeurs technologiques et leaders mondiaux, tels qu'Orange, Phillips, Google et Apple à intégrer cette technologie dans la conception de leurs systèmes.

Désormais la perturbation du fonctionnement des réseaux IoT ne reste pas sans conséquences sur les services mis à la disposition des citoyens et l'activité des entreprises. En effet, la technologie IoT fait partie intégrante des plateformes smart cities et safe cities. Des installations industrielles critiques sont surveillées et contrôlées au travers de la technologie IoT, notamment dans le contexte du concept industrie 4.0 où des plateformes industrielles sensibles seront connectées au réseau à risque Internet.

Force est de constater que la sécurité des réseaux connectés devient la préoccupation des gouvernements et des citoyens. Dans le domaine de la recherche plusieurs travaux ont abordés les aspects de sécurité liés aux technologies IoT. Il s'avère que les technologies et solutions de sécurité utilisées pour la protection des systèmes informatiques restent dans la majorité des cas non adaptées aux spécificités des réseaux IoT.

En effet, les réseaux IoT présentent des contraintes de limitation de ressources et de consommation d'énergie que la plupart des solutions de sécurité informatiques sont incapables de satisfaire. Par voie de conséquence, les nouveaux défis auxquels font face les différents axes de recherches est d'adapter les concepts de sécurité aux spécificités susmentionnées des réseaux IoT.

Lorsqu'on parle de sécurité des objets connectés, on parle de la sécurité du réseau face aux attaques externes mais aussi face aux attaques internes. On désigne par attaque externe un nœud (ou plusieurs nœuds) qui essaient de pénétrer (ou attaquer) le réseau de plusieurs façons et altérer son bon fonctionnement. En revanche, une attaque interne est réalisée lorsqu'un nœud du réseau est lui-même infecté et représente un danger aux autres nœuds du réseau. Que ce soit l'une ou l'autre des deux attaques citées, la nuisance causée peut aller de l'écoute simple du trafic, en essayant de modifier le contenu des communications jusqu'à la destruction complète et l'indisponibilité du réseau.

Dans ce mémoire nous proposons une solution de protection des réseaux IoT basée sur le concept de modèle de confiance. Ce modèle permettra la détection des nœuds malveillants notamment ceux qui constituent une menace interne au réseau.

Il s'agira, de mettre en place un modèle qui permettra à chaque nœud d'évaluer ses voisins en termes de degré de confiance attribuée à ce dernier et de détecter sur la base de cette évaluation si un voisin présente ou pas un danger. Nous procéderons à l'implémentation de ce

modèle de confiance et nous analyserons comment les nœuds connaîtront leurs voisins et sur quelles bases se constitue la confiance. Et surtout, à quel moment peut-on dire qu'un nœud représente vraiment un danger.

Les nœuds malveillants provoquent désormais une dégradation des performances des réseaux IoT. Pour évaluer l'impact de ces nœuds malveillants sur ces réseaux, nous définissons des métriques de performances qui nous permettront d'apprécier à tout moment l'état de fonctionnement des réseaux IoT et de détecter par conséquence la présence des nœuds malveillants.

Nous modéliserons un réseau IoT sous l'environnement de simulation NS2 et nous procéderons par la suite à l'implémentation de notre modèle de confiance. Ce dernier nous permettra de juger de la malveillance des nœuds constituant le réseau. Nous allons aussi, grâce à la simulation, évaluer l'impact de la présence de nœuds malveillants au sein de notre réseau et comment ces derniers dégradent les performances du réseau et impactent la qualité de service du réseau dans un contexte de sécurité.

L'analyse de l'impact des nœuds malveillants sur le réseau nous permettra de réorganiser la communication entre les nœuds du réseau dans un souci d'optimisation de la consommation d'énergie et de disponibilité du réseau.

Le mémoire est structuré autour de quatre chapitres organisés comme suit :

Dans *le premier chapitre*, nous présenterons une vue générale de l'Internet des Objets. En quoi sont-ils différents des réseaux classiques. Dans quels domaines d'application sont-ils utilisés. Comment sont déployées leur architecture et topologie. Ensuite, nous nous intéresserons aux contraintes auxquelles font face les technologies IoT et nous identifierons les vulnérabilités de ces technologies ainsi que les menaces auxquelles elles sont exposées. Nous terminerons le chapitre par une introduction des futurs challenges de l'IoT.

Nous nous intéresserons ensuite aux notions de routages et des différents protocoles des réseaux IoT au début du *second chapitre*. Puis, s'en suivra une introduction au simulateur de réseau NS-2 et de ses différents outils d'analyse qui seront mis en œuvre dans notre travail.

Le troisième chapitre sera entièrement consacré à la modélisation du réseau : On commencera par le déploiement d'un réseau IoT auquel on va injecter des nœuds malveillants. On verra ensuite l'impact de ces derniers sur les performances du réseau, notamment la consommation d'énergie. Un travail d'automatisation nous permettra de montrer cet impact sur les autres paramètres tels que le délai, le débit, la surcharge du réseau ainsi que la gigue.

Finalement, *le quatrième chapitre*, traitera la partie sécurisation du réseau. D'abord en implémentant notre modèle de confiance qui nous permettra de détecter tout comportement malveillant des nœuds constituants le réseau. Nous montrerons comment à travers le rajout d'une couche de sécurité et la réorganisation du réseau en clusters, il serait possible de réduire et d'optimiser la consommation d'énergie liée notamment à la fonction de routage.

Nous concluons le mémoire en présentant une évaluation de notre travail et des perspectives futures.

Chapitre 1

Généralités

1.1 Introduction

L'internet des objets (*Internet of Things*) désigne une interconnexion d'objets dans un réseau doté d'un niveau d'intelligence colossale. L'IoT explose en taille et en complexité, connectant un vaste univers de services grand public, industriels et numériques au réseau. [1]

Au cours de ces dernières années, le nombre d'objets connectés a vu une croissance considérable. L'IoT englobe tout : du stationnement au suivi des véhicules, de la saisie des informations des patients à leurs soins, de maisons / villes intelligentes, des procédures de surveillance et de drones, des réseaux cellulaires, des équipements ménager interconnectés … (etc.) [2]

La sécurité face à cette nouvelle technologie devient une préoccupation vitale. La collecte de données se fait en temps réel et les réseaux sont déployés dans des environnements hostiles d'où la nécessité d'une sécurité accrue autre que les mesures conventionnelles, qui, en raison de contraintes de ressources d'énergie et de mémoire, ne conviennent plus. [3]

Les systèmes IoT sont exposés à différents types de menaces. La considération majeure actuelle est la sécurisation de chaque appareil, chaque sous réseau, chaque réseau de transport, chaque système d'analyse, de service ou de stockage. Il faut assurer la sécurité de tous les processus à chaque couche de l'architecture. [4]

La gestion de la sécurité du réseau est un grand défi pour les administrateurs réseau en raison de l'augmentation des vulnérabilités. Les vulnérabilités sont les faiblesses du réseau qui permettent aux attaquants malveillants d'accéder aux ressources.

Dans cette première partie, nous allons d'abord présenter les différentes technologies sous-jacentes et les différentes phases de l'IoT. Ensuite, les attaques les plus répandues auxquelles l'IoT est sujet, et finalement quelques méthodes pour s'en défendre.

1.2 Aspect général

La variété des services de l'IoT est assurée par différentes technologies comme : L'identification par radiofréquence (RFID), les capteurs, les smart technologies et les nanotechnologies. [1]

- RFID : La radio-identification est une méthode pour mémoriser et récupérer des données à distance en utilisant des étiquettes qu'on appelle « radio étiquettes » qui peuvent être collées ou incorporées dans des objets.

- Capteurs : Les capteurs sont des dispositifs permettant de transformer une grandeur physique observée (température, luminosité, mouvement …) en une grandeur digitale utilisable par les logiciels.

- Les technologies intelligentes : Désignent des appareils qui peuvent être programmés via une interface utilisateur. Ils sont contrôlés ou surveillés à distance.

- Les nanotechnologies : Définies comme ensemble de dispositifs et systèmes matériels à l'échelle nanométrique permettant aux nano-objets la connexion et l'interactivité avec l'environnement.

La figure suivante résume les précédents dires

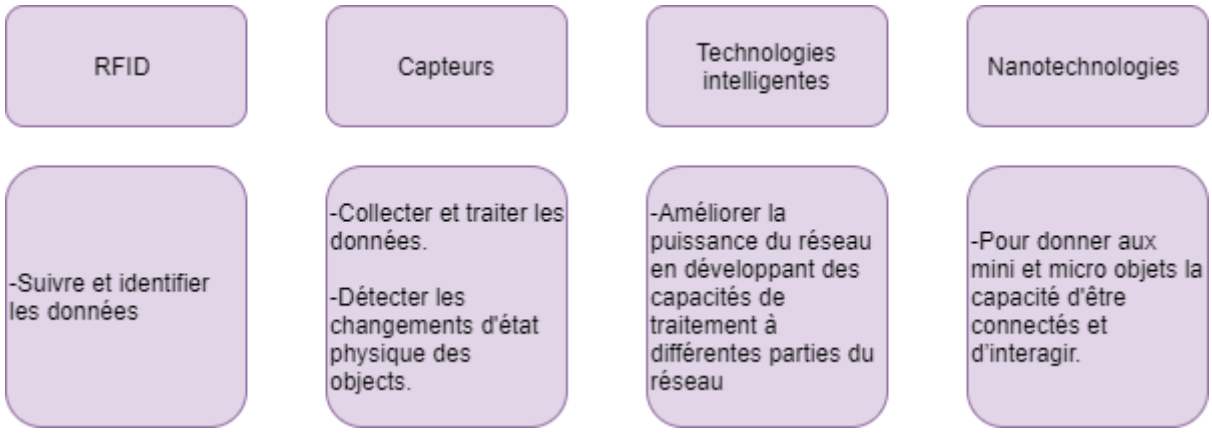

FIGURE1.1 DIFFERENTES TECHNOLOGIES DES IOT.

1.3 Les différentes phases de l'IoT

On définit cinq phases pour l'IoT : La collecte de données, le stockage, le traitement, la transmission de données et finalement la livraison de données.

- Collecte de données : Il s'agit de collecter des données à partir des objets ou appareils. En fonction des caractéristiques des objets, différents types de collecteurs son utilisés : capteurs, étiquettes RFID, puces …

- Stockage : les données collectées seront alors stockées soit dans une mémoire locale ou généralement sur cloud.

- Traitement : Les données stockées seront analysées afin de répondre aux requêtes.

- Transmission : La transmission de données est effectuée durant chaque étape :
  - Des capteurs, étiquettes RFID ou puces aux contrôleurs de domaine.
  - Des contrôleurs de domaines aux unités de traitement.
  - Des processeurs aux contrôleurs, périphériques ou utilisateurs terminaux.

- Livraison : La livraison de données doit se faire à temps, sans erreurs et sans altérations. [1]

Une communication IoT passe principalement par les cinq phases citées précédemment. Chaque phase peut être la cible de plusieurs attaques d'où l'importance de réaliser un système qui peut à la fois détecter mais surtout pouvoir se défendre de ces dernières.

1.4 Topologie

On désigne par topologie la façon dont est organisé un réseau.

- **Topologie en étoile**

Dans cette topologie une station de base envoie ou reçoit un message via un certain nombre de nœuds. Ces nœuds peuvent seulement envoyer ou recevoir un message de l'unique station de base, il ne leur est pas permis de s'échanger des messages.

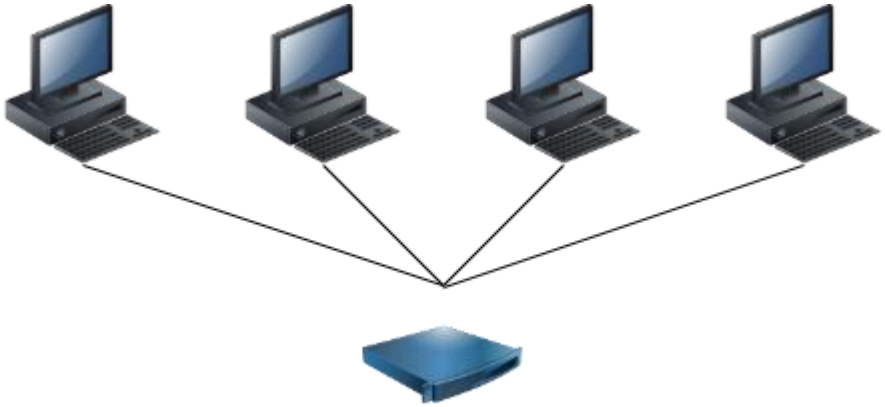

FIGURE1.2 TOPOLOGIE EN ETOILE

Les réseaux en étoiles sont utilisés pour connecter des capteurs isolés capable de transmettre une information dès la mesure du phénomène, ce qui représente la majorité des applications d'avertissement telles que les alarmes (ex : capteur de courant, détecteur de fumée …). Ils ont l'avantage de simplifier les équipements aux niveaux des nœuds et de nécessiter peu d'infrastructure.

Avantage : simplicité et faible consommation d'énergie des nœuds, moindre latence de communication entre les nœuds et la station de base.

Inconvénient : la station de base est vulnérable, car tout le réseau est géré par un seul nœud.

- **Topologie point par point**

Dans ce cas (dit *« communication multi-sauts »*), tout nœud peut échanger avec n'importe quel autre nœud du réseau (s'il est à portée de transmission). Un nœud voulant transmettre un message à un autre nœud hors de sa portée de transmission, peut utiliser un nœud intermédiaire pour envoyer son message au nœud destinataire.

FIGURE1.3 TOPOLOGIE POINT A POINT

Avantage : Possibilité de passer à l'échelle du réseau, avec redondance et tolérance aux fautes.

Inconvénient : Une consommation d'énergie plus importante est induite par la communication multi-sauts. Une latence est créée par le passage des messages des nœuds par plusieurs autres avant d'arriver à la station de base.

- **Topologie maillé**

Les réseaux maillés sont employés dans les réseaux de capteurs plus denses. Ils consistent à interconnecter tous les nœuds ensembles selon le degré d'interconnexion voulu. Ils sont très utilisés dans la domotique ou l'industrie pour la communication entre équipements (M2M) car ils sont très peu sujets aux pannes et extensibles. Ils sont plus complexes à mettre en place vu le nombre de liaisons nécessaires.

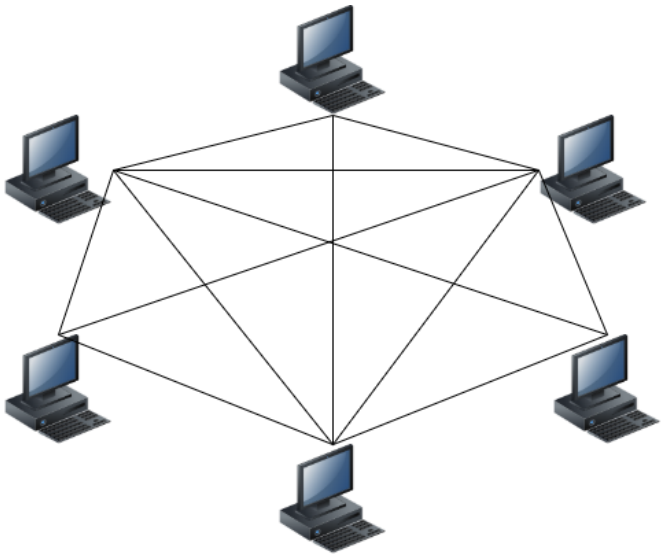

FIGURE1.4 TOPOLOGIE MAILLE

Avantage : Dans un réseau en étoile, un nœud qui se verrait défectueux n'affectera pas le reste du réseau.

Inconvénient : Si l'ordinateur central tombe en panne, l'ensemble du réseau deviendra inutilisable.

1.5 Architecture de l'IoT

L'IoT assure une interconnexion entre différents objets et doit à cet effet assurer une architecture flexible. L'architecture de base est constituée de 3 couches : couche application, couche réseau et couche de perception.

- **Couche de perception**

Aussi appelée 'couche capteurs' ou 'couche physique', elle identifie, rassemble et traite les données pour les envoyer à la couche réseau. Elle utilise principalement les technologies d'identification par RFID, GPS ou capteurs.

- **Couche réseau**

Cette couche est responsable du routage ou du transfert de données vers différents hubs. Elle est constituée de plateformes, de cloud computing, de routeurs et passerelles qui sont des éléments essentiels pour cette couche. Elle utilise certaines technologies telles que 2G / 3G, LTE, WIFI, Bluetooth et ZigBee.

- **Couche application**

Cette couche permet de fournir diverses applications d'environnements intelligents : ville intelligente, transport intelligent ou des applications industrielles telles que les voitures autonomes. [7]

Elle comprend un middleware, un protocole de communication (M2M) et une plateforme de support de service.

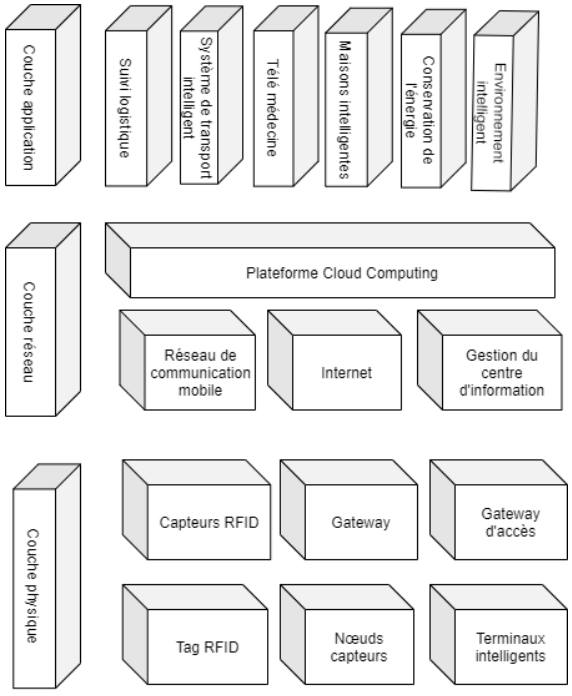

FIGURE1.5 ARCHITECTURE DE L'IOT

1.6 Applications

Les domaines d'application de l'IoT varient : du transport et logistique en passant par les villes/maisons intelligentes et touchant même au domaine des soins et de la santé.

Nous citerons ci-dessous quelques applications :

- **Environnement intelligent**

Dans un environnement intelligent, les données sont échangées avec le monde extérieur. La température, l'humidité et la pression représentent les attributs phares de l'environnement. Les environnements intelligents inclus la surveillance des sols pour l'agriculture, la conservation d'eau pour éviter le gaspillage … etc.

- **Surveillance du trafic**

Les villes intelligentes représentent le grand sujet des IoT et la surveillance du trafic est un élément important pour améliorer l'ensemble du système de congestion sur les routes et ainsi économiser le carburant. Les informations de la congestion sur les routes peuvent être échangées avec les véhicules et ainsi rediriger le trafic. Des mesures de sécurité cruciales doivent être prises pour éviter les attentats terroristes contre les villes.

- **Parking intelligent**

Afin de gagner du temps et du carburant, un système de stationnement intelligent peut renseigner quant aux informations sur les aires de stationnement.

- **Soins de santé**

Grace aux puces RFID, tous les antécédents médicaux peuvent être enregistrés et une alarme peut être envoyée en cas de toute urgence à l'hôpital. Les puces peuvent être implantées ou portées par les personnes et un capteur sera utilisé pour déclencher l'alarme. [5]

L'application des technologies IoT reste très vaste et touche maints domaines hormis ceux cités précédemment.

1.7 Contraintes de l'IoT

- **Contraintes de calcul et d'énergie**

Les appareils IoT sont alimentés par batterie et utilisent des processeurs basse consommation à faible fréquence d'horloge. Par conséquent, les algorithmes cryptographiques coûteux en calcul ne peuvent pas être portés directement sur de tels appareils à faible puissance.

- **Contraintes de mémoire**

Les appareils IoT sont construits avec des mémoires RAM et flash limitées par rapport aux systèmes numériques traditionnels. Ils utilisent un système d'exploitation en temps réel (RTOS) ou une version allégée du système d'exploitation à usage général (GPOS). Par conséquent, les

schémas de sécurité doivent être efficaces en mémoire et les algorithmes de sécurité conventionnels ne peuvent pas être utilisés pour sécuriser les appareils IoT.

- **Contraintes logicielles**

Les systèmes d'exploitation IoT qui sont intégrés aux appareils IoT ont des piles de protocoles réseau minces et peuvent manquer de modules de sécurité. Par conséquent, le module de sécurité conçu pour la pile de protocoles doit être mince, mais robuste et tolérant aux pannes.

- **Mobilité**

La mobilité est l'un des principaux attributs des appareils IoT, où les appareils rejoignent un réseau proximal sans configuration préalable. Cette nature de mobilité soulève la nécessité de développer des algorithmes de sécurité résilients à la mobilité pour les dispositifs IoT.

- **Évolutivité**

Le nombre d'appareils IoT augmente chaque jour et de plus en plus d'appareils se connectent au réseau mondial d'information. Les systèmes de sécurité actuels n'ont pas la propriété d'évolutivité ; par conséquent, de tels schémas ne conviennent pas aux appareils IoT.

- **Multiplicité des appareils**

La diversité des appareils IoT au sein du réseau va des PC à part entiers aux étiquettes RFID bas de gamme. Par conséquent, il est difficile de trouver un schéma de sécurité unique qui accepte l'hétérogénéité de tous les appareils.

- **Multiplicité du support de communication**

Les dispositifs IoT se connectent au réseau local et public via une large gamme de liaisons sans fil. Par conséquent, il est difficile de trouver un protocole de sécurité complet compte tenu des propriétés du support filaire et sans fil.

- **Mise en réseau multi-protocole**

Les appareils IoT peuvent utiliser un protocole réseau propriétaire (par exemple, un protocole non IP) pour la communication dans les réseaux proximaux. Au même temps, ils peuvent communiquer avec un fournisseur de services IoT via le réseau IP. Ces caractéristiques de communication multi-protocole rendent les schémas de sécurité traditionnels inadaptés aux appareils IoT.

- **Topologie de réseau dynamique**

Un appareil IoT peut rejoindre ou quitter un réseau à tout moment et en tout lieu, ce qui décrit la topologie du réseau dynamique. Les modèles de sécurité existant pour les systèmes numériques ne résistent pas à ce type de changements topologiques soudains du réseau. Par conséquent, un tel modèle ne correspond pas à la sécurité des appareils intelligents. [6]

1.8 Analyse de la sécurité

Face aux nombreuses limites auxquelles fait face la technologie de l'IoT, l'aspect sécurité a particulièrement attiré les experts et chercheurs.

La sécurité de l'IoT implique la sécurité : des applications, de l'architecture, de la communication et des données telles montrées dans le diagramme de la **FIGURE1.6** suivante.

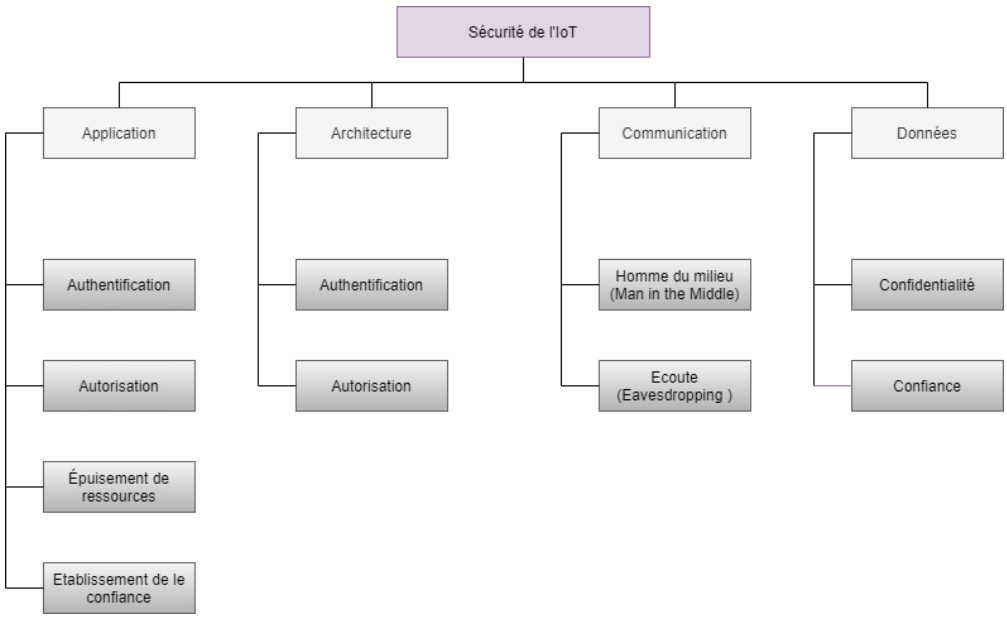

FIGURE1.6 TAXONOMIE DE LA SECURITE DE L'IOT

1.8.1 Applications

Elles sont caractérisées selon : le type d'accessibilité réseau, la portée, la répétabilité, l'hétérogénéité …

Les mesures de sécurité sont : l'authentification, l'autorisation, épuisement de ressource et la confiance.

- **L'authentification**

Elle permet l'intégration de différents appareils IoT et leur déploiement dans différents environnements. Elle implique la validation entre les paires de routage des appareils IoT avant d'échanger les informations.

- **L'autorisation**

Elle consiste à spécifier les droits d'accès aux ressources. Les capteurs, par exemple, ne doivent pas exposer les données collectées à un nœud voisin non autorisé.

- **Epuisement de ressources**

Il s'agit d'une consommation ou allocation de ressources de manière indéfinie ou inutile, ou l'incapacité de les libérer lorsqu'elles ne sont plus nécessaires. Les attaques par épuisement des ressources drainent l'énergie des nœuds IoT cibles en introduisant des boucles de routage et en étendant le chemin pendant la transmission des paquets. Les protocoles de routage sont vulnérables aux attaques d'épuisement des ressources.

- **Etablissement de la confiance**

Un mécanisme de confiance convaincant doit être disponible pour établir la confiance entre les objets physiques de l'IoT et les événements, tels que les WSN interconnectés, les systèmes basés sur la RFID et les téléphones portables. Il existe des mécanismes pour vérifier les périphériques réseau. Cependant, il n'existe pas de mécanisme convaincant pour établir la confiance dans la vérification des applications réseau.

1.8.2 Architecture

Elle doit répondre aux deux contraintes : authentification et autorisation.

- **Authentification**

L'architecture SDN permet d'éliminer la rigidité des réseaux traditionnels, elle permet aux administrateurs d'avoir une perspective globale du système et de contrôler le réseau en fonction des besoins. Cependant, plusieurs vulnérabilités de sécurité existent dans les SDN. L'absence de mécanismes d'authentification et d'autorisation sophistiqués fait des contrôleurs SDN la cible fondamentale des pirates.

L'architecture basée sur SOA prend en charge l'hétérogénéité et l'interopérabilité des appareils IoT, la gestion des informations et la sécurité. Les procédures basées sur SOA fournissent également aux applications IoT une réflexion identique et organisée des services et des conversations avec les appareils IoT. Les méthodes basées sur SOA fournissent une abstraction uniforme et contrôlée des services entre les appareils IoT et garantissent la confidentialité, l'intégrité et la protection des canaux de communication. La fonction principale de SOA est d'empêcher tout accès non autorisé via les fonctionnalités d'authentification, telles que la gestion de la confiance et de l'identité.

- **Autorisation**

L'autorisation dans l'architecture IoT est obtenue en échangeant des données identifiées entre les éléments connectés. Cette procédure est vulnérable aux attaques MitM. OSCAR est une solution pour la sécurité E2E. Ce type d'architecture utilise des serveurs d'autorisation pour accorder l'accès aux utilisateurs. Cependant, l'inconvénient de ce cadre est la latence de l'autorisation, qui affecte largement l'unité de microcontrôleur et les capacités de calcul des dispositifs IoT.

1.8.3 Communication

La communication IoT implique un échange / partage d'informations entre les appareils IoT ou entre différentes couches IoT.

- **MitM**

Les attaques MitM représentent une véritable menace pour la sécurité de l'IoT car elles offrent à un attaquant la possibilité de saisir et de contrôler un canal de communication. Par conséquent, les attaquants peuvent accéder aux données sensibles en communication en temps réel entre les nœuds et obtenir le contrôle du canal. L'attaquant établit alors une connexion avec le nœud réel et sert d'intermédiaire pour lire, rediriger, insérer et modifier le trafic entre l'utilisateur et le nœud authentique. Les protocoles cryptographiques légers sont considérés comme assurant la sécurité des communications pour les appareils IoT sur un réseau informatique dans le cadre du DTLS. Cependant, les attaques MitM profitent des failles dans les protocoles d'authentification utilisés par les parties communicantes.

- **Eavesdropping**

Il s'agit d'une interception d'informations entre deux nœuds communicants. Un programme particulier est utilisé pour renifler et enregistrer des paquets à partir de la couche réseau, qui sont ensuite écoutés ou lus à l'aide d'outils cryptographiques pour l'analyse et le déchiffrement. La confidentialité est utilisée comme méthode pour fournir un contrôle d'accès efficace et une sécurité contre l'écoute lors de la communication de données.

1.8.4 Les données

La confidentialité des données des utilisateurs doit être garantie car les utilisateurs ont besoin d'une protection maximale pour leurs informations personnelles. La confiance implique la préservation de la vie privée des utilisateurs, qui comprend les données personnelles des utilisateurs, par la politique et la perspective des utilisateurs de manière flexible. La transmission et le calcul de la confiance entre différents nœuds dans un IoT hétérogène est un problème difficile car différents nœuds de réseau ont des critères de confiance différents. [7]

1.9 Les attaques les plus répandues

Dû aux vulnérabilités des réseaux IoT en raison des contraintes que ces derniers présentent, une attaque visant un système IoT peut se faire en endommageant ou altérant un nœud, on parle alors de vulnérabilité physique, ou bien en utilisant des erreurs dans les protocoles de routage, c'est-à-dire une vulnérabilité réseau, ou encore en utilisant un programme malveillant qui est une vulnérabilité de chiffrement. [8]

1.9.1 Attaques physiques

Ce sont des attaques qui visent les périphériques physiques du réseau.

- **Altération de nœud**

L'attaquant modifie physiquement le nœud afin d'obtenir des informations sensibles telles que la clé de chiffrement.

- **Interférence RF sur RFID**

L'attaquant effectue une attaque par déni de service en envoyant des signaux de bruit sur des signaux radiofréquence.

- **Brouillage de nœud dans les WSNs**

L'attaquant peut perturber une communication sans fil en utilisant un brouilleur et provoque une attaque par déni de service.

- **Injection de nœud malveillant**

L'attaquant injecte physiquement un nouveau nœud malveillant entre deux nœuds. Il modifie ensuite les données et transmet des informations erronées aux autres nœuds.

- **Injection de code malveillant**

L'attaquant introduit physiquement un code malveillant dans un nœud du réseau et ainsi il peut prendre contrôle de tout le système. [8]

1.9.2 Attaques réseaux

Ce sont des attaques qui visent la couche réseau du système IoT.

- **Attaque Sinkhole**

L'attaquant tente d'attirer tout le trafic des nœuds voisins et écoute toutes les données transmises.

- **Selective Forward**

Le nœud attaquant reçoit les paquets de transmission, mais refuse d'en transmettre certains et abandonne ceux qu'il a refusé de transmettre. L'attaquant doit choisir les paquets à rejeter selon certaines normes telles que la taille, la destination ou l'origine. Dans ce cas, seuls les paquets libérés par le nœud attaquant peuvent être librement transmis. [9]

- **MitM**
- **Déni de service**

L'attaquant à travers un large trafic cause un déni de service. [8]

- **Sybil Attack**

L'attaque Sybil est définie comme un «appareil malveillant prenant illégalement de multiples identités». En utilisant l'attaque Sybil, un adversaire peut se trouver à plusieurs endroits à la fois car un nœud unique présente plusieurs identités aux autres nœuds du réseau. [3]

1.9.3 Attaques logiciel

Utilisant des virus ou des vers pour obtenir des informations ou créer des dénis de service.

- **Virus, vers ou cheval de trois**

L'attaquant utilise des codes malveillants injectés dans des pièces jointes des mails, ou en téléchargeant des contenus par internet. Les vers ont la capacité de se dupliquer d'eux-mêmes sans intervention humaine. [8]

1.9.4 Attaques de chiffrement

Elles consistent à détruire les techniques de cryptage et obtenir les clés privées.

- **Attaque par canal latéral**

L'attaquant utilise le canal latéral qui contient des informations sur la puissance, le temps nécessaire pour effectuer l'opération, la fréquence … et déduit la clé.

- **MitM**

1.10 Challenges de l'IoT

L'IoT est un réseau d'objets hétérogènes et peut interagir avec le milieu extérieur. Le principal défi de l'IoT est de réduire la consommation d'énergie et de minimiser l'utilisation des ressources. Lors de la conception de tout protocole ou architecture pour l'IoT, il faut prendre en considération :

- **Evolutivité massive**

Les appareils intelligents déployés sur le réseau sont nombreux, il faut donc fournir l'authentification, la maintenance, la protection, l'utilisation et la prise en charge d'un si grand nombre d'objets.

- **Architectures et dépendances**

Un grand nombre d'appareils est connecté à internet d'où la nécessité d'avoir une architecture adéquate qui permet une connectivité, un contrôle et des communications faciles.

- **Génération d'un big data**

Une grande quantité de données brutes sera collectée en continue. Il est nécessaire de développer des techniques pour convertir les données brutes en flux utilisable.

- **Robustesse**

Les applications IoT fonctionnent sur la base de plateforme de détection, d'automatisation et de calcul. Lors des déploiements, il est nécessaire pour les appareils de connaître leurs emplacements, d'avoir des horloges synchronisées, de connaître leurs appareils voisins lorsqu'ils coopèrent et d'avoir un ensemble de paramètres cohérents tels que le sommeil, les horaires d'éveil, les niveaux de puissance appropriés pour la communication.

D'autre part, la sécurité est le plus grand challenge de l'IoT. Chaque couche est la cible de nombreuses attaques. Dans ce qui va suivre, nous verrons les mesures de sécurité à satisfaire pour chaque couche.

- **Couche de perception**

Les nœuds des systèmes IoT se trouvent à la périphérie et sont exposés aux attaques physiques, de plus, la faible capacité de stockage avec une capacité de calcul limitée rendent les nœuds plus vulnérables. Pour limiter les attaques, cette couche doit être dotée d'un système de chiffrement, d'un contrôle d'accès et d'authentification afin de confirmer l'identité de l'envoyeur.

- **Couche réseau**

Pour faire face aux attaques liées à cette couche, l'échange de clé doit être doté d'une sécurité accrue. En effet, la communication IoT (machine to machine) est différente de la communication traditionnelle jusque-là connue (machine to human), et les protocoles courants ne couvrent pas la nature hétérogène de l'IoT. La nécessité d'utiliser des protocoles plus adéquats est requise.

- **Couche application**

La couche application assure la gestion du trafic et l'interaction avec les utilisateurs et est la cible des attaques DoS. Cette couche doit, lors de la conception, prendre en considération la quantité de données qu'elle doit gérer et les mécanismes d'authentification liés à l'interaction des utilisateurs.

1.11 Conclusion

Dans ce qui a précédé, on a présenté un aspect général de la technologie IoT, l'architecture de cette dernière ainsi que ses nombreuses applications. Nous avons aussi eu un aperçu des limites de cette technologie et ses vulnérabilités qui peuvent être exploitées à des fins malveillantes causant différentes attaques citées dans ce chapitre.

En vue des contraintes que l'IoT rencontre, différents challenges défient les experts afin d'apporter la sécurité nécessaire à cette technologie qui ne cesse de se répandre dans plusieurs domaines de la vie quotidienne.

Dans les chapitres suivants, nous allons nous pencher vers la sécurité d'un réseau de nœuds capteur. Nous allons surtout aborder l'aspect de la confiance entre les nœuds afin de détecter les intrusions malveillantes. Mais avant, il sera question de définir dans le détail les différents aspects du protocole AODV avant de nous lancer dans la simulation puis l'analyse des résultats pour finir avec la conclusion et les perspectives.

Chapitre 2

Routage et Qualité de Service dans les Réseaux de Capteurs Sans Fil

2.1 Introduction

Dans ce chapitre nous allons commencer par aborder la notion des réseaux sans fil, des protocoles de communication et plus particulièrement du protocole AODV (*Ad-hoc On Demand Vector*) sur lequel se basera notre simulation dans le chapitre suivant. Puis nous procéderons à un état de l'art sur le modèle de confiance entre les nœuds.

Nous finirons par introduire le simulateur NS-2 (*Network Simulation*) avec lequel le travail de simulation se fera dans le prochain chapitre. Pour cela, nous présenterons les différents outils dont nous aurons besoin afin de bien assimiler la simulation lors du chapitre 3.

2.2 Le routage dans les réseaux de capteur sans fil

Comme souligné dans le chapitre précédent, les protocoles de routage pour les réseaux filaires sont inadéquats aux réseaux sans fil à cause de leurs architectures décentralisées, leurs topologies dynamiques et le manque d'infrastructures.

Pour les réseaux sans fil, les protocoles de routage doivent prendre en considération la mobilité des nœuds et les changements de topologies. Pour cela, ils doivent être capables de gérer les mises à jour à tout instant et les changements de route.

Depuis l'émergence des réseaux sans fil, plusieurs protocoles de routage ont été proposés et évalués par rapport à plusieurs aspects tels que l'évolutivité (scalabilité), la sécurité, la performance, la qualité de service et l'adaptabilité.

Nous allons commencer par voir les différentes classifications de routage et par la suite la classification des protocoles pour les réseaux de capteurs sans fil.

2.2.1 Classification des routages dans les RCSF

Les différents protocoles de routage proposés pour les RCSF sont classés en deux catégories :

Selon la structure : on distingue le routage à plat, le routage hiérarchique et le routage basé sur la position des nœuds.

Selon le routage : on distingue le routage orienté-données, le routage multi-chemin et le routage basé sur le QdS (*Qualité de Service*).

2.1.1.1 Routage à plat

Dans ce cas, l'ensemble des nœuds collaborent à l'acquisition et transmission d'information. Le routage peut être proactif ou réactif. Dans le cas d'un routage proactif, les routes sont construites au préalable et restent disponibles. Dans le cas d'un routage réactif, les routes ne sont établies qu'à la demande. [10]

2.1.1.2 Routage hiérarchique

Le routage à plat connait une limite car il est sur un seul niveau et entraîne une surcharge des stations de base. Afin d'y apporter une solution, dans cette approche, les nœuds peuvent avoir des fonctions différentes. Un nœud particulier « le cluster-head » est mis en place pour acheminer les informations vers les stations de base afin de diminuer le nombre de communications et d'éviter la latence du réseau. [10]

2.1.1.3 Routage basé sur la position des nœuds

Dans cette approche, la position des nœuds est exploitée afin d'acheminer les informations suivant le plus court chemin. Ceci est possible en calculant des coordonnées virtuelles ou réelles et en estimant ainsi la position des nœuds. Toutefois, un inconvénient majeur de cette technique est qu'elle n'est pas très précise d'une part et est très consommatrice d'énergie d'autre part. [10]

2.1.1.4 Routage orienté-données

Dans ce paradigme, un nœud n'est pas contraint à posséder qu'un seul identifiant. Les stations de base se chargent de diffuser des requêtes et attendent des réponses de la part des nœuds. Ces dernières comparent les requêtes diffusées avec leurs données pour donner réponse. Si un nœud n'est pas en mesure de satisfaire une requête, il la transfère à son voisin. [10]

2.1.1.5 Routage multi chemin

Il s'agit là d'utiliser plusieurs routes pour la transmission des données. Ceci permettrait d'améliorer les performances et la tolérance aux pannes. Le protocole se doit alors de maintenir en permanence les routes secondaires en cas de défaillance de la route principale, ceci engendre une consommation supplémentaire de l'énergie. [10]

2.1.2 Classification des protocoles

Selon que le routage soit proactif, réactif ou hybride, on classifie les protocoles de routage comme suite.

2.1.2.1 Protocoles proactifs

Nous verrons dans ce cas le protocole DSDV et OLSR.

- DSDV (Destination Sequenced Distance Vector Protocol) est un protocole adapté aux réseaux multi-sauts. Il permet aux nœuds d'évaluer la fraicheur des routes en introduisant un numéro de séquence aux champs de la table de routage. Ce dernier sera incrémenté à chaque changement. Les nœuds s'échangent régulièrement les mises à jour afin de maintenir la cohérence des tables de routage. [10]

- OLSR (Optimized link State Routing) est un protocole adapté aux réseaux ad hoc. Son concept repose sur l'utilisation d'un relai multipoints (MPR) pour réduire la redondance dans la diffusion des paquets. Les MPRs rassemblent tous les voisins à un saut ce qui permet de joindre tous les voisins à deux sauts. Lors de la diffusion d'un message, tous les voisins reçoivent le message mais uniquement les nœuds MPRs peuvent le retransmettre ce qui diminue les transmissions au sein d'un réseau. [10]

2.1.2.2 Protocoles réactifs

- DSR (Dynamic Source Routing) emploie l'approche « routage source ». Le nœud source est chargé de déterminer la séquence des nœuds formant le chemin à travers lequel les données vont être transmises. Il commence par un processus de découverte de route. Une fois trouvé, il s'agira de maintenir les routes tant que les nœuds formant ces dernières sont toujours actifs. En cas de détection d'une erreur, le nœud concerné sera supprimé ainsi que toutes les routes le contenant. [10]
- AODV (Ad hoc On demand Distance Vector) est un protocole de routage à vecteur de distance inspiré des deux protocoles DSDV et DSR. Les routes sont établies à la demande afin de réduire le nombre de messages diffusés et limiter l'impact des modifications topologiques uniquement aux routes actives.
  AODV utilise un champ indiquant la fraicheur de la route appelé numéro de séquence afin de maintenir la cohérence des informations de routage et ainsi prendre toujours les routes les plus récentes.
  Quand un nœud source désire établir une route vers un nœud distant et qu'il ne possède pas de route disponible au niveau de sa table de routage (route expiré ou non existante), il diffuse un message de demande de route RREQ (Route REQuest). La demande de route contient un identifiant (RREQ ID) associé à l'adresse du nœud source qui servira à identifier la demande de route de façon unique. A chaque fois qu'un nœud reçoit une nouvelle demande de route (qui n'a pas été traité auparavant), il sauvegarde l'identifiant de la demande de route (RREQ ID) et l'adresse du nœud source dans son historique pour une durée bien déterminé (Path_ Discovery_Time).
  Lorsqu'un nœud intermédiaire (le cas des nœuds C, B et D dans la **FIGURE2.1**) qui ne possède pas de route valide vers la destination reçoit la demande de route RREQ, il ajoute ou met à jour le voisin émetteur au niveau de sa table de routage. Si la RREQ a été déjà traitée, le nœud récepteur l'abandonne et ne la rediffuse pas. Sinon, il ajoute ou met à jour la route vers le nœud source au niveau de sa table de routage. Il incrémente ensuite le nombre de sauts dans la demande de route et la rediffuse. À la réception de la demande de route, la destination (nœud J) ajoute ou met à jour dans sa table de routage une route vers le nœud voisin a partir duquel il a reçu la RREQ (nœud H) ainsi qu'une route vers le nœud source (nœud A). Le nœud destination (nœud J) génère ensuite une réponse de route RREP qu'il transmet en unicast vers le prochain saut en direction du nœud source.
  En AODV même un nœud intermédiaire peut générer une réponse de route s'il possède une route valide et fraiche vers la destination à condition que le nœud source l'autorise à le faire (bit Destination_only de la RREQ mis à 0).
  Lors de la réception d'une réponse de route par un nœud intermédiaire, celui-ci ajoute ou met à jour la route vers la destination, ensuite retransmet la réponse de route RREP en unicast (après l'incrémentation du nombre de sauts) via la route inverse qui a été créé lors du passage de la RREQ. Lorsque le nœud source reçoit la RREP, une route bidirectionnelle est établie entre lui et le nœud destination, et ainsi la transmission de paquets de données peut commencer.

Pour maintenir les routes, AODV utilise des diffusions périodiques de messages HELLO (une RREP diffusé aux voisins à un saut), et des diffusions (ou des unicasts) des messages d'erreur RERR (Route ERRor) pour signaler les ruptures de liaison et invalider les routes non fiables. [10]

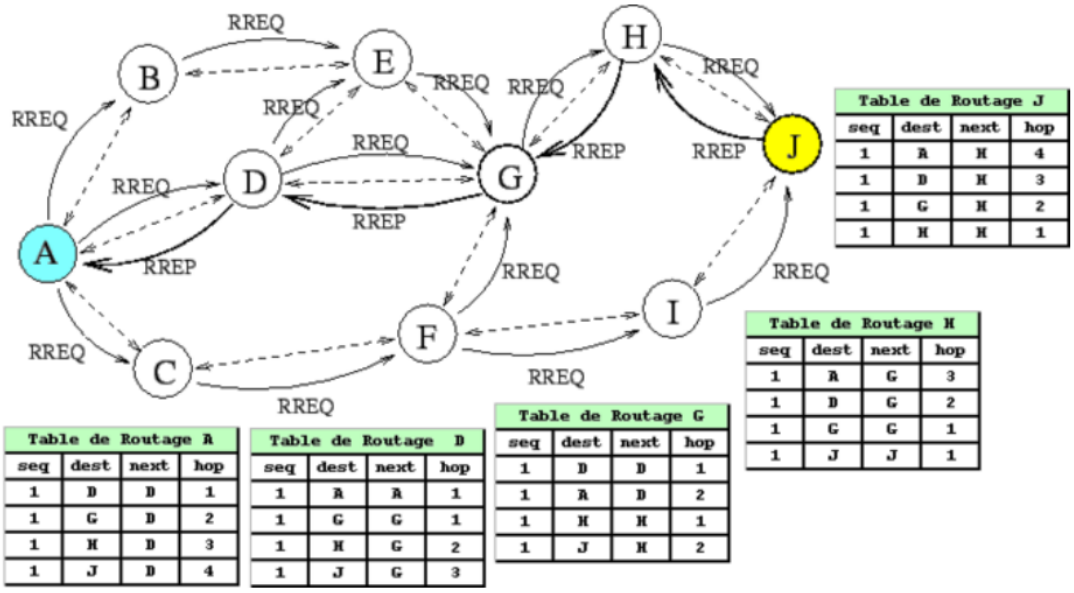

FIGURE2.1 DECOUVERTE DE ROUTE EN AODV

2.1.2.3 Protocoles hybrides

- ZRP (Zone Routing Protocol) est un protocole hybride qui spécifie pour chaque nœud une zone de routage incluant tous les nœuds à K sauts. Il utilise l'approche proactive afin de trouver toutes les routes à k-sauts ; et l'approche réactive afin de trouver toutes les routes à une distance supérieure à K.
  Le protocole construit un arbre multicast afin de fournir les différents chemins atteignant tous les nœuds du réseau. Grâce à cette approche, il réduit le temps de latence et le nombre de messages de contrôle. [10]
- ZHLS (Zone-Based Hierarchical Link State) est un protocole qui divise le réseau en zones géographiques qui ne se chevauchent pas. Chaque nœud est localisé par ses coordonnées GPS. Il emploi l'approche réactive pour trouver la zone du nœud destination. Chaque nœud dispose de deux tables de routage, une table intra-zone qui maintient les liens entre les nœuds appartenant à la même zone et les liens vers des nœuds d'autres zones. Et une table inter-zone pour maintenir une carte des zones qui fournit un plan des connexions entre les zones. [10]

Nous avons à travers cette section dédiée aux protocoles de routage présenté les protocoles les plus adaptés aux RCSF. Nous nous sommes attardé sur le protocole AODV car ce dernier sera utilisé comme protocole de routage lors de la simulation qu'on effectuera dans le chapitre suivant. Désormais, nous allons procéder à la partie état de l'art afin de résumer les études faites en relation avec notre travail.

2.2 Etat de l'art

Dans l'étude qui sera la nôtre, le routage va être au cœur du travail. En effet, suite au transfert de données, chaque nœud tiendra une table de confiance de ses nœuds voisins et ce en se basant sur le comportement de ce dernier dans sa tâche de délivrance des paquets. Mais nous allons d'abord commencer par définir certaines notions importantes.

De nos jours, l'IoT gagne en terrain et évolue comme étant la prochaine génération des réseaux et bien d'autres services, assurer la fiabilité des données et services, la confidentialité des utilisateurs et surtout la sécurité des informations est donc cruciale. La «confiance» représente un concept très complexe, qui tient compte de la fiabilité et des attentes en matière de sécurité, de capacité et d'exactitude. [7]

Un réseau d'objets IoT peut être vu comme un graphe pondéré non orienté G (V,E,w) tq : V est l'ensemble des nœuds, E l'ensemble des arêtes, et w représente le poids. Un graphe peut être sujet à des attaques qui cibleront un nœud mais qui impacteront l'ensemble du réseau. Un nœud malveillant exploitera son degré de confiance afin d'altérer un autre ou un ensemble de nœud et finalement diviser le réseau.

L'attaque éclipse peut être un bon exemple pour illustrer le schéma précédent. Une attaque éclipse est un moyen d'attaquer un réseau par lequel un attaquant cherche à isoler et à attaquer un seul nœud. L'attaquant inonde d'abord la cible de ses propres adresses IP auxquelles la victime va se connecter au redémarrage (qui peut être forcé). La victime aura une vue des nœuds malveillants sans aucune vue possible sur le réseau plus large. Ainsi cette étape constituera une porte pour diviser le réseau et en prendre possession.

Cette attaque est possible car les réseaux IoT sont décentralisés et ne permettent pas à tous les nœuds de se connecter simultanément à tous les autres du réseau. Au lieu de cela, un nœud se connecte à un nombre sélectionné d'autres nœuds, qui à leur tour sont connectés à un groupe … etc. La facilité avec laquelle une attaque éclipse peut se produire dépend d'un nombre de facteurs, y compris la structure de données d'un réseau et le nombre de connexion de chaque nœud.

Afin d'assurer la fiabilité, la confidentialité et la sécurité en terme d'information plusieurs méthodes ont été étudiées notamment le modèle de confiance. L'évaluation de la confiance est généralement un nombre calculé par les appareils IoT pour évaluer la fiabilité d'autres appareils qui ont précédemment interagi avec eux. Le modèle de confiance se base sur plusieurs caractéristiques des nœuds, notamment l'énergie résiduelle, le rang ou le degré de ce dernier.

2.2.1 Modèle de confiance dans les RCSF

En termes de gestion de confiance, la valeur de confiance et la confiance globale sont deux concepts importants. La valeur de confiance est généralement un nombre qui indique la fiabilité d'un nœud. Il s'agit d'une valeur calculée par les autres nœuds en fonction de l'expérience précédente entre eux. D'autre part, la confiance globale est l'accumulation de plusieurs valeurs de confiance à partir de différents nœuds IoT. En d'autres termes, lorsque plusieurs nœuds IoT calculent différentes évaluations de confiance envers un nœud spécifique,

ces différentes valeurs seront agrégées ensemble pour produire l'évaluation finale de confiance (confiance globale) envers ce nœud. [7]

Nous étudierons les solutions apporté dans la littérature sous deux aspects : l'aspect du modèle de confiance entre nœuds afin d'assurer la sécurité de l'information et éventuellement la défense contre les nœuds malveillants et l'aspect QdS qui assurera la bonne gestion de l'énergie du réseau en sa globalité et des nœuds en particulier. Ceci va s'inscrire dans le contexte de la sécurisation d'un réseau IoT et ce en assurant la « Disponibilité » du réseau au même titre que la « Confidentialité » et l'« Intégrité ».

Comme souligné précédemment, les réseaux IoT sont ouverts, anonymes, dynamiques par nature, de sorte qu'un objet malveillant puisse pénétrer dans le réseau et le perturber. Des modèles de confiance ont été proposés pour identifier les objets malveillants et améliorer la fiabilité du réseau. La confiance peut être mesurée comme une valeur continue [0,1] où 0 est la méfiance et 1 est entièrement fiable. Les valeurs discrètes [-1,0,1] peuvent également être utilisées pour mesurer la confiance où -1 est la méfiance, 1 est entièrement fiable et 0 n'est ni la confiance ni la méfiance. Les approches basées sur des seuils utilisent une valeur de seuil pour identifier la fiabilité du nœud.

[4] propose un modèle afin d'atténuer les attaques et supprimer l'impact des recommandations malveillantes dans le calcul de confiance. Chaque nœud évalue les nœuds voisins et sur la base de cette évaluation décide de l'interaction. Les nœuds peuvent partager des informations sur les évaluations de confiance envers les voisins autant que recommandation. Parfois, les nœuds malveillants envoient de fausses recommandations et les nœuds sains obtiennent des valeurs de confiance faibles. Les recommandations doivent être évaluées en fonction de la crédibilité du recommandataire, afin d'atténuer ce type d'attaques.

Plusieurs approches ont été proposées, le modèle CORE utilise la technologie du watchdog pour calculer la valeur de confiance et ne transmet que les recommandations positives. CONFIDENT transmet les bonnes et mauvaises recommandations tout en déclenchant une alarme pour les mauvaises recommandations. SORI élimine les paquets selon une probabilité calculée sur la valeur de confiance.

L'approche proposée dans [4] calcule la crédibilité de la recommandation. Cette dernière est utilisée comme pondération dans le calcul de la confiance indirect et ainsi permet de réduire l'impact des fausses recommandations dans le calcul de la confiance.

Une autre solution a été apportée en tenant compte de la mémoire limitée des objets IoT et leur capacité de stockage limitée. Pour ce faire, dans [11], la gestion de la confiance se base sur des recommandations directes et indirectes, d'un seuil de confiance tout en rajoutant la solution sur la taille de stockage qui sera constante et ce en compressant l'historique du comportement dans un seul enregistrement. Ils rajoutent aussi le fait que les mauvais comportements sont punis davantage tandis que les bonnes opérations influencent lentement le résultat. Cela signifie qu'un utilisateur a besoin de nombreux bons comportements pour gagner en fiabilité, mais quelques activités malhonnêtes pourraient détruire la confiance.

2.2.2 Qualité de service

La QdS est un concept très important dans les réseaux de capteurs sans fil. Ceci se confirme bel et bien par les nombreuses études qu'on trouve dans la littérature. La durée de vie

d'un réseau représente l'une des contraintes à laquelle doit faire face ce dernier. En raison de l'autonomie dont sont dotés les capteurs (nœuds IoT), dès que la batterie d'un noeud arrive à expiration, ce dernier s'éteint.

On peut donc dire que le fait d'augmenter la durée de vie du réseau nous offrirait une meilleure qualité de service.

Sur cette base, plusieurs approches ont vues le jour. Les algorithmes les plus populaires proposent la démarche de la division de l'ensemble des nœuds en « clusters ».

Dans [5] les auteurs ont mis en place une méthode de sélection des Cluster Head en divisant les nœuds en nœuds CH et nœuds non CH, ainsi les nœuds non CH ne sont responsables que de la surveillance et les nœuds CH collectent toutes les informations des nœuds non CH et les transmettent aux BS (Stations de bases) qui sont à un saut. Les techniques basées sur le cluster peuvent être classées en approche de « clustering distribué » et « clustering centralisé » : Le clustering distribué signifie qu'un nœud décide de lui-même de ses actions et de ses tâches ce qui engendre des clusters non uniformes. Le clustering centralisé signifie que toutes les activités concernant le clustering dans le réseau sont communiquées aux BS. Dans les approches centralisées, de meilleures économies d'énergie peuvent être réalisées. Mais lorsqu'il s'agit de réseaux à grande échelle, le problème se présente sous la forme d'une évolutivité. La BS a du mal à surveiller les activités de tous les nœuds déployés. Les techniques centralisées conviennent uniquement aux réseaux à petite échelle avec moins de nœuds de capteur.

Une méthode bien efficace de la sélection des CH est basée sur l'énergie résiduelle. Après avoir divisé la zone de regroupement, les premiers CH sont sélectionnés à l'aide d'un algorithme d'estimation du temps de survie dans lequel la proportion de l'énergie résiduelle par rapport à l'énergie moyenne des nœuds du réseau est prise en compte. Le nœud du capteur situé près du point moyen du cluster est sélectionné comme CH. Afin d'assurer au mieux la continuité de transmission des données, une fois que la mort du premier CH survient, un autre est sélectionné selon l'énergie restante des nœuds, et celui dont l'énergie est la plus grande est sélectionné. [5]

Dans [12], une modification de l'algorithme LEACH (Low Energy Adaptive Clustering Hierarchy) a été apportée. Ce dernier offrait une élection des CH au hasard, de sorte qu'un nœud avec une énergie petite et un autre avec une grande énergie aient les mêmes chances d'être élus et ceci constituait une limitation de l'algorithme car si un nœud avec une petite énergie est sélectionné, il s'éteint très rapidement. La modification visait à sélectionner le CH en tenant compte de paramètres importants comme l'énergie initiale, l'énergie restante du nœud individuel et le nombre optimal de CH dans le réseau. À la fin de chaque tour, l'énergie résiduelle des nœuds non CH est vérifiée, et celui avec le niveau d'énergie le plus élevé par rapport aux autres a une probabilité plus élevée de sélection CH pour le tour en cours.

Les méthodes décrites précédemment sont les plus populaires, il existe encore plusieurs solutions qui n'ont pas été citées.

2.3 Introduction au simulateur NS2

Network Simulator (NS-2) est un simulateur à événements discrets orienté objet, écrit en C++ avec une interface qui utilise le langage OTcl (Object Tool Command Langage). A

travers ces deux langages il est possible de modéliser tout type de réseau et de décrire les conditions de simulation : La topologie réseau, le type du trafic qui circule, les protocoles utilisés, les communications qui ont lieu ...etc. Le langage C++ sert à décrire le fonctionnement interne des composants de la simulation. Quant au langage OTcl (dérivé de Tcl), il fournit un moyen flexible et puissant du contrôle de la simulation comme le déclenchement d'événements, la configuration du réseau, la collecte de statistiques, ...etc.

NS-2 prend en charge les réseaux filaires et sans fil. Toute simulation sous NS-2 se base sur un modèle composé des éléments suivants :

-Nœuds du réseau : Nœuds d'extrémités où le trafic est généré ou consommé et les nœuds de routage (nœuds intermédiaires).
-Liens de communications entre ces nœuds.
-Agents : représentent les protocoles au niveau transport (TCP, UDP), ces agents sont connectés aux nœuds et sont attachées les uns aux autres pour permettre l'échange de données.
-Application : qui génère le trafic des données. [13]

Le processus de simulation via NS-2 suit trois phases comme le montre la **FIGURE2.2** suivante :

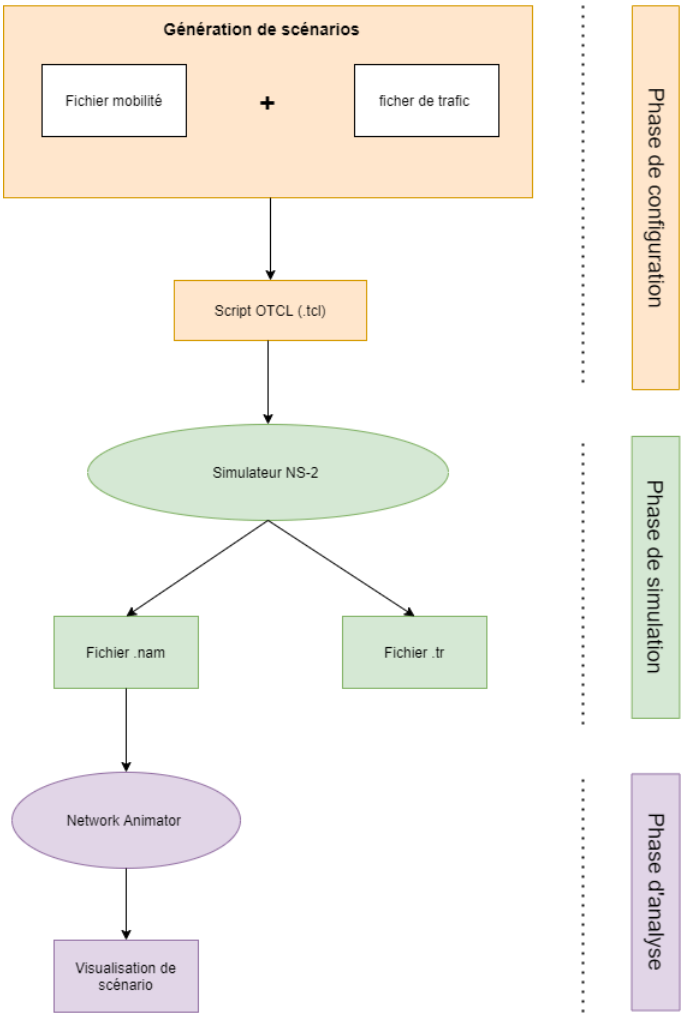

FIGURE2.2 PHASES DE SIMULATION SOUS NS-2

La phase de configuration s'occupe de la génération des fichiers d'entrées. Ils sont souvent classés en deux catégories : Fichier mobilité décrivant le nombre de nœuds, la vitesse … et le fichier de communication qui décrit le trafic dans le réseau.

La phase de simulation exécute le scénario configuré dans la phase précédente.

La phase d'analyse s'occupe quant à elle de la collecte et la compilation des résultats de la simulation. [14] (cf. Annexe. A pour l'installation complète de NS-2)

2.3.1 Le fichier de mobilité

NS-2 nous permet de simuler un réseau de nœuds mobiles comme le sont les capteurs dans un environnement réaliste.

Pour ce faire, un outil de génération de mobilité est mis à notre disposition et ce dans le répertoire suivant :

/opt/ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-gen/setdest

La commande : ./setdest permet d'avoir toutes les informations sur cet outil.

La génération du fichier de mobilité se fait grâce à la ligne de commande suivante :

./setdest –**v** <1/2> **-n** <nombre de nœuds> **-m** <vitesse min> **-M** <vitesse max > **-t** <temps de simulation> **-x** <max X> **-y** <max Y> > <nom de fichier de mobilité>

A l'issu de cette commande, un fichier sera créé. Dans la **FIGURE2.3** suivante, un aperçu du fichier généré où on peut apercevoir dans la première ligne les différents paramètres renseignés et les positions X,Y et Z de chaque nœud.

```
#
# nodes: 20, speed type: 1, min speed: 10.00, max speed: 15.00
# avg speed: 12.33, pause type: 1, pause: 0.00, max x: 800.00, max y: 800.00
#
$node_(0) set X_ 416.455596178482
$node_(0) set Y_ 581.627229828961
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 418.430700300631
$node_(1) set Y_ 527.370811275469
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 380.119411615850
$node_(2) set Y_ 745.284436042226
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 246.441640532543
$node_(3) set Y_ 87.043590402479
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 214.850229371536
$node_(4) set Y_ 325.543458692972
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 313.068894557266
$node_(5) set Y_ 33.396303222149
$node_(5) set Z_ 0.000000000000
```

FIGURE2.3 FICHIER DE MOBILITE

Les nœuds du réseau auront une mobilité aléatoire à chaque instant comme le montre la **FIGURE2.4** suivante :

```
$ns_ at 0.000000000000 "$node_(0) setdest 178.750910512216 42.832403469165 12.835386212627"
$ns_ at 0.000000000000 "$node_(1) setdest 570.092610684052 526.139994113967 12.186073294073"
$ns_ at 0.000000000000 "$node_(2) setdest 72.892255513368 477.045829041286 14.026169347913"
$ns_ at 0.000000000000 "$node_(3) setdest 228.830909262604 498.898184246109 10.633725244974"
$ns_ at 0.000000000000 "$node_(4) setdest 198.916743457011 534.544252042008 13.830980344639"
$ns_ at 0.000000000000 "$node_(5) setdest 396.457176483956 164.604216403718 11.850126089953"
$ns_ at 0.000000000000 "$node_(6) setdest 419.671006149512 146.218397471455 14.669942107598"
$ns_ at 0.000000000000 "$node_(7) setdest 314.253270012485 502.433987638510 10.951627370189"
$ns_ at 0.000000000000 "$node_(8) setdest 292.727034931822 642.986765723035 13.595406137836"
$ns_ at 0.000000000000 "$node_(9) setdest 616.580253524867 83.075664490400 10.132029714869"
$ns_ at 0.000000000000 "$node_(10) setdest 131.644022693382 499.811797766214 11.661994008043"
$ns_ at 0.000000000000 "$node_(11) setdest 225.728449726365 622.837706084885 12.058412424091"
$ns_ at 0.000000000000 "$node_(12) setdest 210.872015511007 703.898544053290 10.197766520114"
$ns_ at 0.000000000000 "$node_(13) setdest 564.567195957056 2.555980843409 11.468346878052"
```

FIGURE2.4 MOBILITE DES NŒUDS

A la fin du fichier, un tableau récapitulatif montrant l'ID de chaque nœud ainsi que tous les changements de lien et de route.

Node	Route Changes	Link Changes
0	25	11
1	23	11
2	40	9
3	69	11
4	44	13
5	39	10
6	30	13
7	24	9
8	38	11
9	55	4
10	24	13
11	76	2
12	23	12
13	28	13
14	41	12
15	24	13
16	37	15
17	33	7
18	27	9
19	52	6

FIGURE2.5 TABLEAU RECAPITULATIF

2.3.2 Le fichier trace

NS-2 génère deux fichiers, un fichier log qu'on appelle « Trace File » et un fichier d'animation qu'on appelle NAM.

Le fichier trace nous permet d'enregistrer tous les évènements qui se produisent dans le réseau : envoie, réception, partage, perte de paquets … Ce qui nous facilitera l'analyse des performances du réseau par la suite.

En analysant une ligne, nous pouvons avoir toutes les informations nécessaires :

```
s –t  0.00000000 –Hs 0 –Hd -2 –Ni 0 –Nx 416.46 –Ny 581.63 –Nz 0.00 –Ne -1.000000 –Nl RTR –Ma 0 –Md 0 –Ms 0 –Mt 0 –Is 0255 –Id -1.255 –It TAODV –Il 44 –If 0 –Ii 0 –Iv 1 –P taodv –Pt 0x1 –Ph 1 –Pd 0 –Pds 2 –Pl 4.000000 –Pc HELLO
```

FIGURE2.6 FORMAT DU FICHIER TRACE

Dans le tableau suivant [15] chaque champ sera expliqué :

Numéro	Signification	Valeurs possibles
1	Type d'évènement	s : send
		r : receive
		d : drop
		f : forward
2 -3	Temps	-t : time
4-7	Informations sur les nœuds	-Hs : id du nœud courant
		-Hd : id du nœud prochain
8-17	Propriété des nœuds	-Ni : id du nœud
		-Nx : coordonnée X du nœud
		-Ny : coordonnée Y du nœud
		-Nz : coordonnée Z du nœud
		-Ne : niveau d'énergie du noeud
18-19	Niveau de trace,	-N1 : niveau de trace (AGT, RTR, MAC)
	Couche dans laquelle se trouve le paquet	AGT : agent
		RTR : routage
		MAC : couche mac
20-27	Informations de paquet au niveau MAC	-Ma : durée
		-Md : adresse ethernet de la destination
		-Ms : adresse ethernet de la source
		-Mt : type d'antenne

27-42	Informations de paquet au niveau d'IP	-Is : adresse IP source et port correspondant
		-Id : adresse IP destination et port correspondant
		-It : type de paquet (AODV, DSR …)
		-Il : taille du paquet
		-If : identifiant du flux du paquet
		-Ii : identifiant du paquet
		-Iv : durée de vie (valeur ttl)
43-56	Champs spécifiques	-P : protocole utilisé
		-Pf : combien de fois le paquet a été expédié
		-Ps : numéro de séquence
		-Pd : adresse destinataire
		-Pds : adresse source
		-Pl : longueur de réponse
		-Po : nombre optimal d'expédie

TABLEAU2.1 SIGNIFICATION DES DIFFERENTS CHAMPS D'UN FICHIER TRACE

Ce fichier nous sera d'une grande utilité dans notre travail de simulation et d'analyse des résultats qu'on verra dans le chapitre suivant.

2.3.3 L'outil de visualisation NAM

L'exécution du fichier « .nam » génère une fenêtre graphique. Une fois apparut, on peut distinguer trois parties : une grande surface pour l'animation, une échelle temps munie d'un curseur qu'on peut faire glisser et barre de commande afin de lancer la lecture, la stopper et la faire avancer.

La **FIGURE2.4** nous montre un aperçu graphique du fichier « .nam » sous l'environnement NS-2.

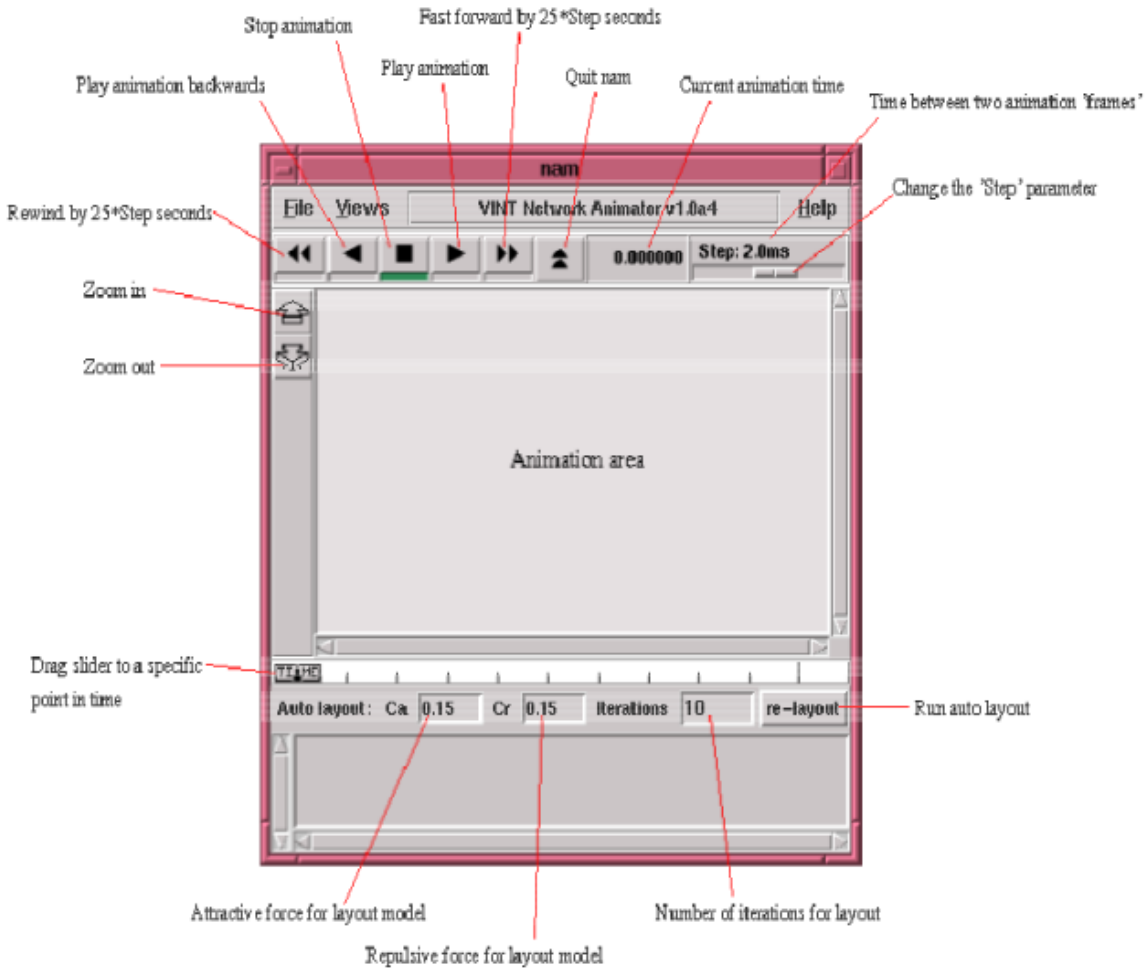

FIGURE2.7 APERÇU DE L'OUTIL DE VISUALISATION NAM

2.3.4 Le langage AWK

AWK est un langage de traitement de lignes utilisé principalement pour la manipulation des fichiers textuels, nommé d'après le nom de ses trois auteurs originaux:

A : Alfred Aho

W : Peter Weinberger

K : Brian K ernighan

Le fichier « .awk » contient 3 sections : Begin, Middle et End. Le bloc *Begin* est exécuté une seule fois au début, avant le traitement des données. Il sert à initialiser les variables et leur affichage.

Le bloc *End* est exécuté une seule fois à la fin, après le traitement des données.

Le bloc du milieu est exécuté plusieurs fois selon le besoin. Les données sont transformées et manipulées avec une ou plusieurs instructions.

Ce langage nous sera d'une grande aide afin de traiter les résultats du fichier trace « .tr ». Nous verrons dans le chapitre suivant comment est définit chaque section du fichier et la puissance de ce langage.

2.4 Conclusion

Dans ce chapitre nous avons d'abord commencé par une partie théorique des protocoles de routage les plus adaptés aux RCSF tout en mettant l'accent sur le protocole AODV. Nous avons aussi vu les études faites dans la littérature concernant le modèle de confiance et la durée de vie du réseau. Ces concepts seront au cœur de notre étude tout au long du chapitre suivant.

Pour finir, nous avons présenté le simulateur NS-2 avec lequel se fera le travail de simulation et les différents outils dont on aura besoin.

Le chapitre suivant traitera principalement de l'utilisation de NS-2 afin de modéliser un réseau de capteurs sans fil dans lequel des nœuds malveillants seront injectés. On étudiera l'impact de ces derniers sur les performances du réseau et nous présenteront notre modèle de confiance qui nous permettra de détecter la présence des nœuds malveillants.

Chapitre 3

Simulation de nœuds malveillants et étude de leur impact.

3.1 Introduction

Afin d'évaluer le comportement d'un système complexe quand il devient couteux de mettre en place tout un réseau à des fins de test, on a souvent recours à des simulations.

La simulation est une technique de modélisation du monde réel. Elle permet de représenter le fonctionnement d'un système que l'on souhaiterait observer. Elle est utilisée pour mesurer les performances d'un réseau et devient nécessaire lorsque l'expérimentation est couteuse et complexe.

Dans ce chapitre, nous allons utiliser le simulateur de réseau NS-2 comme introduit dans le chapitre précédent. Au long de ce chapitre, nous allons commencer par une modélisation d'un réseau de capteurs. Des nœuds malveillants seront ensuite injectés au sein du réseau.

Nous admettons que la simulation, aussi bien réalisée que soit-elle, peut refléter la réalité mais ne jamais la représenter complètement. C'est dans cette perspective que nous allons essayer de nous rapprocher le plus de la réalité dans notre analyse des performances. Pour cela, nous allons automatiser le processus de calcul des paramètres d'analyse afin d'évaluer le degré de dégradation des différents paramètres du réseau.

3.2 Modélisation

3.2.1 Déploiement du réseau

Dans ce qui va suivre, une première modélisation d'un ensemble de nœuds communicants sera faite. Afin d'être fidèle à la réalité, NS-2 nous permet d'introduire différents paramètres qu'on verra dans ce qui va suivre.

Au cours de cette simulation, nous allons utiliser le protocole de routage AODV (Ad-hoc On-demand Distance Vector) comme indiqué précédemment. On va déployer un réseau de 20 nœuds dans une surface de 800 x 800m² sous NS-2 avec 4 nœuds malveillants placés aléatoirement au sein du réseau.

La simulation se fera avec la présence de 20% de nœuds malveillants. Les paramètres de simulation sont donnés dans le tableau Tab3.1 suivant :

Paramètre	Valeur
Temps de simulation	10 s
Nombre de nœuds	20
Surface	800 x 800
Protocole de transport	UDP
Protocole d'application	CBR

TABEAU3.1 PARAMETRES GENERAUX DE SIMULATION

Une fois les paramètres généraux renseignés, nous passons à la génération du fichier de mobilité dont la commande d'exécution fût déjà renseignée.

Dans le répertoire */opt/ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-gen/setdest*, nous allons ouvrir un terminal et taper la commande :

./setdest –v 2 -n 20 -m 10.0 -M 15.0 -t 10 -x 800 -y 800 > mob

Un fichier nommé « mob » sera créé. Ce dernier sera placé dans le même répertoire où sont enregistrés tous les scripts de la simulation. Il sera appelé au cours de l'exécution du script TCL comme le montre la **FIGURE3.1** suivante, afin d'assurer la mobilité des nœuds.

```
for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns_ node]
        $ns_ initial_node_pos $node_($i) 35
}

source mob
```

FIGURE3.1 APPEL DU FICHIER DE MOBILITE

Afin de veiller au bon déroulement de la simulation, nous allons spécifier certains paramètres importants tels que : le type d'antenne, de couche de liaison, de l'interface réseau, du protocole de routage et du nombre de nœuds comme le montre la **FIGURE3.2** suivante :

```
set val(chan)           Channel/WirelessChannel  ;# Channel type
set val(prop)           Propagation/TwoRayGround ;# Radio-propagation model
set val(ant)            Antenna/OmniAntenna      ;# Antenna type
set val(ll)             LL                       ;# Link layer type
set val(ifq)            Queue/DropTail/PriQueue  ;# Interface queue type
set val(ifqlen)         50                       ;# Max packet in ifq
set val(netif)          Phy/WirelessPhy          ;# Network interface type
set val(mac)            Mac/802_11               ;# MAC type
set val(rp)             TAODV                     ;# Routing protocol
set val(x)              800                      ;# X length
set val(y)              800                      ;# Y length
set val(finish)         10                       ;# Finish time
set val(nn)             20                       ;# Number of mobilenodes
```

FIGURE3.2 DIFFERENTS PARAMETRES REQUIS

Nous remarquons que le protocole de routage ainsi définit est « TAODV » au lieu d'« AODV ». Le T fait référence à Trust, et désigne les changements apportés au protocole originel AODV afin d'implémenter notre modèle de confiance. Nous tâcherons d'expliquer dans le chapitre suivant les étapes de ce travail.

Après cela, vint une autre procédure obligatoire afin de bien mener cette simulation. Il s'agit de la déclaration de la simulation, des différents fichiers de sauvegarde, de la topologie, du canal de communication et de l'API de déclaration des nœuds. Ci-dessous les lignes de codes spécifiques.

```
set ns_ [new Simulator]                                    #Déclaration d'une nouvelle simulation

$ns_ use-newtrace                                          #Utilisation du nouveau format trace

set trf [open out.tr w]                                    #Ouverture du fichier trace en mode écriture
$ns_ trace-all $trf

set namtrace [open out.nam w]                              #Ouverture du fichier nam en mode écriture
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

set topo [new Topography]                                  #Déclaration de la topologie du réseau
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
set chan_1 [new $val(chan)]

#Déclaration de l'API des noeuds
$ns_ node-config -adhocRouting $val(rp) \
        -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
        -topoInstance $topo \
        -agentTrace ON \
        -macTrace ON \
        -routerTrace ON \
        -movementTrace ON \
        -channel $chan_1
```

FIGURE3.3 AUTRES PARAMETRES DE SIMULATION

Finalement, une procédure « *finish* » est requise afin de fermer les fichiers de sauvegarde ouverts plus en haut et de lancer la simulation. Voici un aperçu de cette dernière :

```
proc finish {} {
    global ns_ namtrace filename
    $ns_ flush-trace
    close $namtrace
    exec nam out.nam &
    exit 0
}

$ns_ at $val(finish) "finish"
puts "Start of simulation..."
$ns_ run
```

FIGURE3.4 DECLARATION DE LA PROCEDURE FINISH

Une visualisation du fichier « .nam » montre le déploiement d'un réseau de 20 nœuds comme suite.

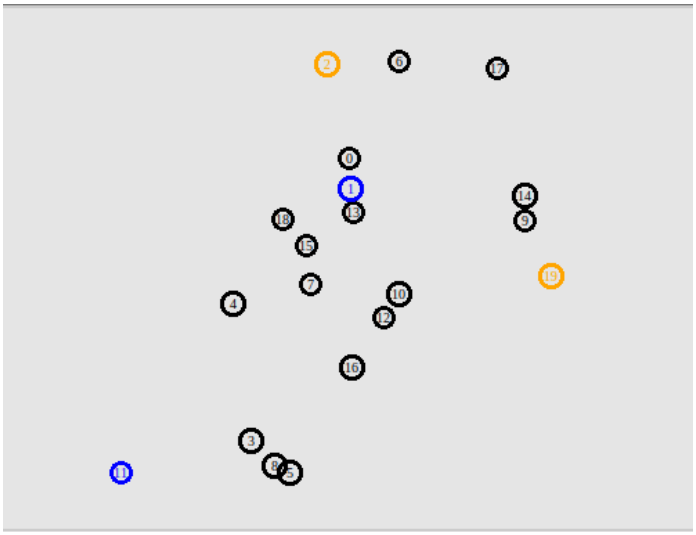

FIGURE3.5 VISUALISATION DU RESEAU

Les nœuds qui apparaissent en couleur bleue sont des nœuds sources et ceux en orange des nœuds destination. Le trafic s'effectue du nœud 1 vers le 2 et du 11 vers le 19. Le protocole de communication étant UDP avec CBR comme application de transfert de paquets.

Notre réseau étant déployé et fonctionnel, notre prochaine étape sera de simuler un comportement malsains par des nœuds malveillants.

3.2.2 Injection des nœuds malveillants

L'injection de nœuds malveillants passe par une procédure fastidieuse. La première étape est d'activer le mode promiscuité (*promiscuous mode* en anglais). Ce dernier se réfère à une configuration de la carte réseau. En l'activant, cette dernière va accepter tous les paquets qui transitent dans le réseau même ceux qui ne lui sont pas destinés. Ce mode provoque :

Au niveau de l'attaque : Il permet d'écouter le réseau, de repérer l'architecture ou même de voler certaines informations.

Au niveau de la défense : Il permet de faire une analyse du trafic, de chercher les problèmes de sécurité, des pannes ou même de détecter les intrusions dans le système. Ce dernier point nous intéresse particulièrement.

Après avoir activé ce mode (cf. Annexe. B pour la procédure d'activation de ce mode). Notre prochaine étape est de simuler le comportement malsain des nœuds, l'algorithme suivant décrit les étapes suivies et les modifications apportées.

Algorithme 1 : Ajout du comportement malveillant des nœuds

```
//Déclaration de la variable malicious

bool malicious

//Ajout de la condition d'ajout d'un nœud malveillant

malicious = false
```

si (strcmp(argv[1], ‘’malicious’’) == 0)

malicious = true ;

return TCL_OK ;

fin si

//Ajout du flag du comportement malveillant

Define DROP_MAL ‘’MAL’’

//Ajout de la condition du comportement malveillant

si (malicious)

drop(packets, DROP_MAL)

fin si

Une fois l’implémentation validée, nous allons au niveau du script TCL activer le comportement malveillant en déclarant les lignes suivantes :

```
$ns_ at 0.0 "$node_(0) color red"
$node_(0) color "red"
$ns_ at 0.0 "[$node_(0) set ragent_] malicious"

$ns_ at 0.0 "$node_(5) color red"
$node_(5) color "red"
$ns_ at 0.0 "[$node_(5) set ragent_] malicious"

$ns_ at 0.0 "$node_(7) color red"
$node_(7) color "red"
$ns_ at 0.0 "[$node_(7) set ragent_] malicious"

$ns_ at 0.0 "$node_(16) color red"
$node_(16) color "red"
$ns_ at 0.0 "[$node_(16) set ragent_] malicious"
```

FIGURE3.6 DECLARATION DES NŒUDS MALVEILLANTS

Une fenêtre NAM de visualisation va apparaître après compilation du script montrant les nœuds malveillants en rouge.

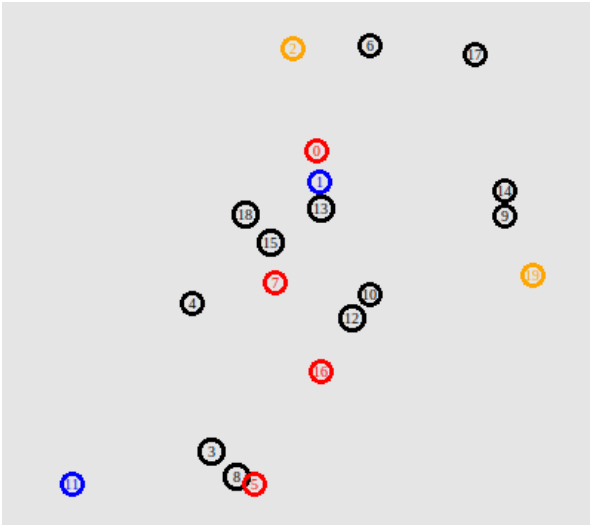

FIGURE3.7 VISUALISATION DU RESEAU EN PRESENCE DE NŒUDS MALVEILLANTS.

En lançant la simulation, les paquets seront envoyés du nœud 11 vers le nœud 19 en prenant des chemins aléatoires et variés. On note au niveau du nœud 5 (**FIGURE3.8**) que la communication est coupée et que les paquets n'atteignent pas le nœud destinataire (nœud 19). Ceci valide notre processus d'injection de nœuds malveillants en simulant un comportement malsain.

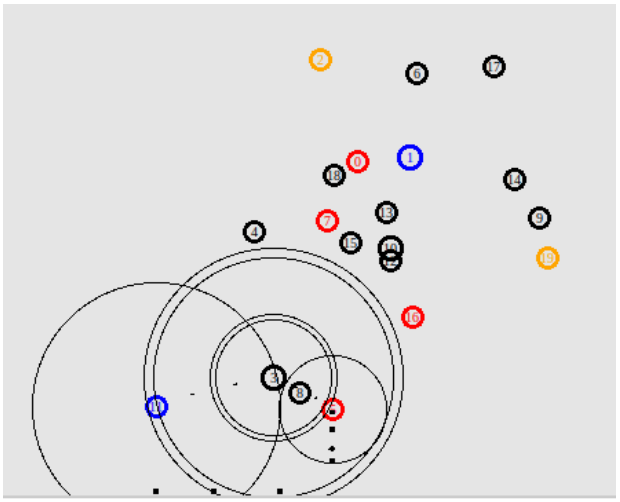

FIGURE3.8 PERTE DE PAQUETS LORS DE LA TRANSMISSION

L'étape de l'injection des nœuds malveillants faite, nous allons maintenant montrer comment ces derniers peuvent altérer les performances du réseau en termes de durée de vie et de consommation d'énergie.

3.3 Impact des nœuds malveillants sur le réseau

La présence des nœuds malveillants dans le réseau peut impacter le bon fonctionnement de ce dernier. Afin d'avoir un aperçu du degré d'impact de ces nœuds sur la dégradation des performances du réseau nous allons suivre l'évolution de chaque nœud, sa vitesse, sa position et son énergie tout au long de la période du transfert des paquets.

Dans ce qui va suivre l'algorithme détaillant la procédure suivie.

Algorithme 2 : Position, Vitesse et énergie des nœuds

```
//Déclaration des variables en mode protégé
double xpos, ypos, zpos, energy_t ;
int n_speed ;
MobileNode *t_node ;
//Initialisation des variables :
xpos = ypos = zpos = 0.0 ;
energy_t = 0.0 ;
n_speed = 0 ;
//Récupération de chaque nœud par son adresse :
t_node = get_node_by_address
//Affectation des variables de position, d'énergie et de vitesse
get_location (&xpos, &ypos, &zpos) ;
n_speed=t_node (speed()) ;
//affichage des valeurs dans un fichier .dat :
fprintf (fp, ''Id of node is %d. Position is %d %f %f %f. speed is %d. energy is %f '',index, xpos, ypos, zpos, n_speed, energy_t)
```

Un fichier .dat va être créé, et on peut visualiser les informations liées à chaque nœud comme le montre la **FIGURE3.12**.

Le premier champ désigne l'ID du nœud, les trois champs suivants désigne les positions x,y et z respectivement du nœud, puis sa vitesse et finalement son énergie.

```
1, 464.868143, 526.993947, 0.000000,1 , 12 , 1.075818
1, 465.432020, 526.989371, 0.000000,1 , 12 , 1.063113
1, 466.058608, 526.984285, 0.000000,1 , 12 , 1.039491
1, 466.451129, 526.981100, 0.000000,1 , 12 , 1.019534
1, 467.271202, 526.974445, 0.000000,1 , 12 , 0.974977
1, 467.741217, 526.970630, 0.000000,1 , 12 , 0.969760
1, 468.416872, 526.965147, 0.000000,1 , 12 , 0.931059
1, 469.210753, 526.958704, 0.000000,1 , 12 , 0.922556
1, 469.813234, 526.953815, 0.000000,1 , 12 , 0.891052
1, 470.560014, 526.947754, 0.000000,1 , 12 , 0.866499
1, 470.945117, 526.944629, 0.000000,1 , 12 , 0.836429
1, 471.297296, 526.941771, 0.000000,1 , 12 , 0.811126
1, 471.422344, 526.940756, 0.000000,1 , 12 , 0.802624
1, 471.703728, 526.938472, 0.000000,1 , 12 , 0.791202
1, 472.276977, 526.933820, 0.000000,1 , 12 , 0.765955
1, 472.746213, 526.930012, 0.000000,1 , 12 , 0.744849
1, 473.055276, 526.927504, 0.000000,1 , 12 , 0.728607
1, 473.666224, 526.922546, 0.000000,1 , 12 , 0.702932
1, 474.431252, 526.916337, 0.000000,1 , 12 , 0.668856
1, 475.144100, 526.910552, 0.000000,1 , 12 , 0.641624
1, 475.823735, 526.905036, 0.000000,1 , 12 , 0.609164
1, 476.572579, 526.898959, 0.000000,1 , 12 , 0.576555
1, 477.442551, 526.891899, 0.000000,1 , 12 , 0.540108
1, 477.890785, 526.888261, 0.000000,1 , 12 , 0.519638
1, 478.295785, 526.884974, 0.000000,1 , 12 , 0.502386
```

FIGURE3.12 EVOLUTION DE LA POSITION, LA VITESSE ET L'ENERGIE DU NŒUD 0

On note que le nœud « 1 » à force de renvoyer les paquets perdues (à cause des nœuds malveillants qui coupent la communication) consomme rapidement son énergie.

Afin de suivre cette consommation dans le temps, nous allons suivre l'énergie résiduelle des nœuds directement dans le fichier trace « .tr ». Pour cela, on va déclarer le modèle énergétique directement dans l'API des nœuds telle le montre la **FIGURE3.13**

```
$ns_ node-config -adhocRouting $val(rp) \
        -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
        -energyModel "EnergyModel" \
        -initialEnergy 3.0 \
        -txPower 0.9 \
        -rxPower 0.5 \
        -idlePower 0.45 \
        -sleepPower 0.05 \
        -topoInstance $topo \
        -agentTrace ON \
        -macTrace ON \
        -routerTrace ON \
        -movementTrace ON \
        -channel $chan_1
```

FIGURE3.13 IMPLEMENTATION DU MODELE ENERGETIQUE

Nous pouvons suivre l'évolution de l'énergie des nœuds dans le fichier trace. Nous aurons à t=0.0000000 la position de chaque nœud et son énergie comme le montre la **FIGURE3.14**.

```
M 0.00000 0 (416.46, 581.63, 0.00), (178.75, 42.83), 12.84
M 0.00000 1 (418.43, 527.37, 0.00), (570.09, 526.14), 12.19
M 0.00000 2 (380.12, 745.28, 0.00), (72.89, 477.05), 14.03
M 0.00000 3 (246.44, 87.04, 0.00), (228.83, 498.90), 10.63
M 0.00000 4 (214.85, 325.54, 0.00), (198.92, 534.54), 13.83
M 0.00000 5 (313.07, 33.40, 0.00), (396.46, 164.60), 11.85
M 0.00000 6 (503.22, 750.91, 0.00), (419.67, 146.22), 14.67
M 0.00000 7 (349.63, 359.91, 0.00), (314.25, 502.43), 10.95
M 0.00000 8 (287.28, 44.46, 0.00), (292.73, 642.99), 13.60
M 0.00000 9 (721.48, 473.41, 0.00), (616.58, 83.08), 10.13
M 0.00000 10 (505.25, 343.57, 0.00), (131.64, 499.81), 11.66
M 0.00000 11 (17.77, 32.83, 0.00), (225.73, 622.84), 12.06
M 0.00000 12 (476.64, 303.03, 0.00), (210.87, 703.90), 10.20
M 0.00000 13 (423.47, 487.53, 0.00), (564.57, 2.56), 11.47
M 0.00000 14 (724.09, 514.34, 0.00), (508.72, 423.74), 12.38
M 0.00000 15 (340.67, 430.43, 0.00), (581.83, 93.29), 12.34
M 0.00000 16 (420.81, 216.91, 0.00), (626.82, 315.29), 13.71
M 0.00000 17 (674.99, 737.37, 0.00), (181.95, 207.56), 13.16
M 0.00000 18 (299.36, 473.93, 0.00), (738.99, 663.55), 10.07
M 0.00000 19 (769.35, 376.21, 0.00), (117.38, 26.41), 10.34
```

FIGURE3.14 VALEURS INITIALES

Au cours de la simulation, les changements des valeurs d'énergie des nœuds seront transcrite au niveau du fichier trace. Dans la **FIGURE3.15** qui va suivre, le premier champ désigne le temps, le second l'Id du nœud et le dernier champ renseigne de l'énergie résiduelle du nœud.

```
-t 0.014874 -n 14 -e 2.991971
-t 0.014874 -n 19 -e 2.992001
-t 0.014875 -n 10 -e 2.991944
-t 0.014875 -n 17 -e 2.992041
-t 0.014875 -n 12 -e 2.991939
-t 0.014875 -n 13 -e 2.991944
-t 0.014875 -n 1 -e 2.991944
-t 0.014875 -n 0 -e 2.991964
-t 0.014875 -n 6 -e 2.991970
-t 0.014875 -n 15 -e 2.991939
-t 0.014875 -n 7 -e 2.991939
-t 0.014875 -n 16 -e 2.991947
-t 0.014876 -n 18 -e 2.991939
-t 0.014876 -n 2 -e 2.991970
-t 0.014876 -n 4 -e 2.991977
```

FIGURE3.15 ENERGIE RESIDUELLE

Il est à noter que dans notre cas, le calcul de l'énergie résiduelle d'un nœud se base sur sa consommation d'énergie à envoyer ou recevoir des paquets et ne comprend pas sa mobilité.

Au niveau de la visualisation graphique, les nœuds qui apparaissent en couleur jaune arrivent à un niveau bas de leurs batteries. Ceci est le cas du nœud 1 et 11 tous deux des nœuds source, mais aussi du nœud 3 qui est un nœud malveillant et qui supprime tous les paquets qu'il reçoit.

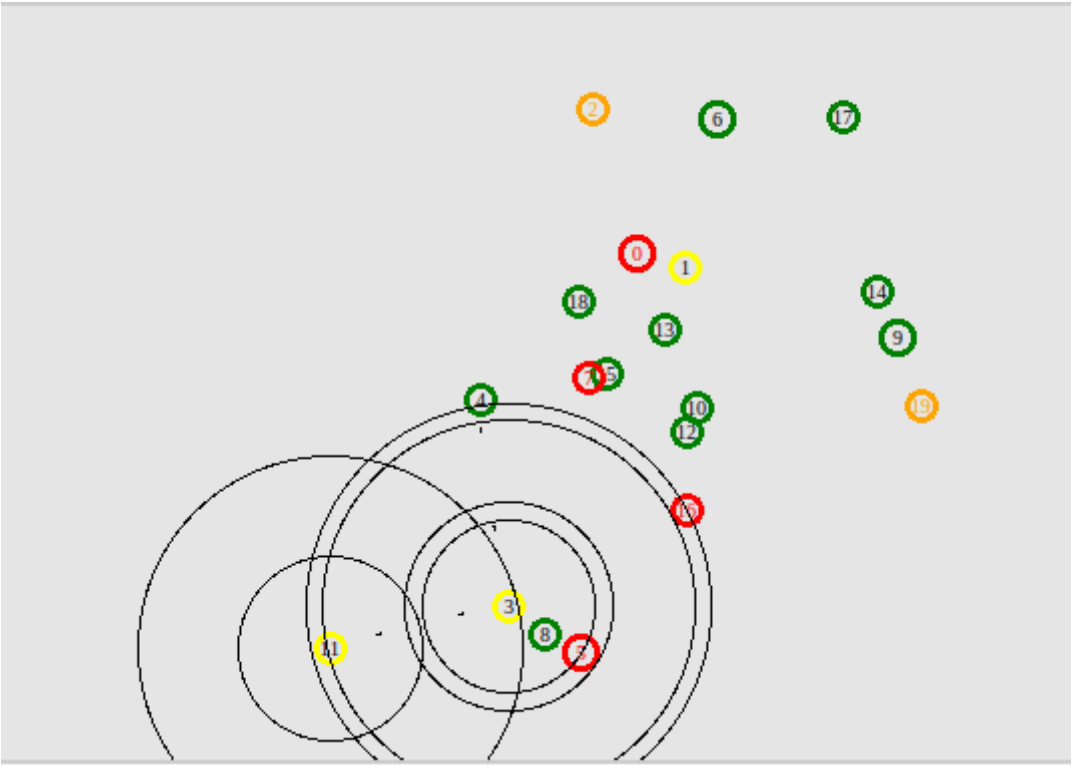

FIGURE3.16 EPUISEMENT DE BATTERIE DES NŒUDS

Nous constatons que la présence d'un nœud malveillant qui empêche la bonne transmission des paquets d'un nœud source vers un nœud destinataire provoque une plus grande consommation d'énergie. Car à chaque coupure de transmission, les paquets non reçus doivent être retransmis et ceci altère la batterie des nœuds.

La forte consommation de l'énergie des nœuds n'est pas le seul impact qu'engendrent les nœuds malveillants. Nous verrons que ces derniers altèrent les performances du réseau en termes de débit, de congestion du réseau et de surcharge de routage et cela sera notre prochaine étape.

3.4 Analyse de performance

Afin de connaître les métriques de performance du réseau, nous allons procéder à une analyse de différent paramètres qui nous permettrons par la suite de juger de la qualité du réseau.

Une bonne analyse nous permet de vérifier notre travail, nous aide à apporter les modifications nécessaires et surtout nous permet de mettre en place les conclusions auxquelles notre travail a abouti.

Au cours de cette analyse, nous allons nous baser précisément sur l'analyse des paramètres suivants :

-Le ratio de livraison de paquets (*Packets delivery ratio*) : qui correspond au nombre de paquets reçus par la destination divisé par le nombre de paquets émis par la source.

-Le ratio de perte de paquets (*Packets dropping ratio*) : qui correspond au nombre de paquets perdus divisé par le nombre de paquets émis.

-Le débit (*Throughput*) : qui correspond à la quantité de données par unité de temps qui est fourni d'un nœud à un autre par l'intermédiaire d'une liaison de communication. Il est mesuré en bits/secondes.

-Surcharge du routage normalisée (*Nomalized routig overhead*) : qui correspond au nombre de paquets en cours de routage divisé par le nombre total de paquet reçus.

-Surcharge de contrôle du réseau (*Control overhead*) : Qui correspond au rapport entre le nombre de paquets reçus par le nœud destinataire et la durée d'observation.

-Le délai (*Delay*) : qui correspond au temps que met un paquet pour arriver à la destination. Un délai plus petit correspond à de meilleures performances.

-La gigue (*Jitter*) : qui correspond à la variation entre les arrivées des paquets dans le temps. Il sert à mesurer la stabilité de la réponse. Il mesure le retard causé par la congestion du réseau, un changement de topologie ou un changement d'itinéraire.

NS-2 à travers la commande suivante, nous permet de visualiser tous ces paramètres et ce après avoir écrit un script en AWK et nous référant toujours au fichier trace :

```
awk –f nom_fichier.awk nom_ficher.tr
```

AWK est un langage de traitement de lignes utilisé principalement pour la manipulation des fichiers textuels.

Le fichier .awk contient 3 sections : Begin, Middle et End. Le bloc *Begin* est exécuté une seule fois au début, avant le traitement des données. Il sert à initialiser les variables et leur affichage.

Le bloc *End* est exécuté une seule fois à la fin, après le traitement des données.

Le bloc du milieu est exécuté plusieurs fois selon le besoin. Les données sont transformées et manipulées avec une ou plusieurs instructions.

Les **FIGURE317 FIGURE3.18** suivantes montrent les deux blocs *Begin* et *End* sous NS-2.

```
BEGIN {
        send = 0;                       #variable for storing number of packets sent
        recv = 0;                       #variable for storing number of packets received
        drp=0;                          #variable for total number of packet dropped
        bytes = 0;                      #variable for storing number of bytes transmitted
        st = 0;                         #variable for start time
        ft = 0;                         #variable for end time
        rtr = 0;                        #variable for number of routing packets
        delay = 0;                      #variable for delay
        jitter=0;                       #variable for jitter
        jitter_count=0;                 #variable for jitter count
        last_pkt_recv_time=0;           #variable for last packet received time
        nbr_of_mal_node=0;              #variable for total number of malicious node
```

FIGURE3.17 BLOC BEGIN

```
END {
        if(recv == 0) recv=1; #Handled issue when recv value is 0 which causes other values to raise a divide by zero error.

        #Printing results
        printf("No_of_Packets_Sent: \t\t%.f\n",send);
        printf("No_of_Packets_Received: \t%.f\n",recv);
        printf("No_of_Packets_dropped: \t\t%.f\n",drp);
        printf("Packet_Delivery_Ratio: \t\t%.2f %%\n",recv/send*100);
        printf("Packet_Dropping_Ratio: \t\t%.2f %%\n",(send-recv)/send*100);
        printf("Throughput: \t\t\t%.2f Kbps\n",bytes*8/(ft-st)/1000);
        printf("Normalized_Routing_Overhead: \t%.2f %%\n",rtr/recv*100);
        printf("Control_Overhead: \t\t%d\n",rtr);
        printf("Delay: \t\t\t\t%.2f Seconds\n",delay/recv);
        printf("Jitter: \t\t\t%.2f\n",jitter/jitter_count);
        if(nbr_of_mal_node!=0) {
                printf("The number of malicious node is:\t\t%d\n",nbr_of_mal_node);
                printf("The ID of malicious node are:\n");
                for(k=0;k<s;k++) {
                        printf("%d\n", Tab[k]);
                }
        }
}
```

FIGURE3.18 BLOC END

Le résultat de l'exécution du script nous permet de dire que les nœuds malveillants causent une dégradation des performances du réseau comme on peut le déduire des **FIGURE3.19** et **FIGURE3.20** suivantes :

```
No_of_Packets_Sent:             223
No_of_Packets_Received:         111
No_of_Packets_dropped:          58
Packet_Delivery_Ratio:          49.78 %
Packet_Dropping_Ratio:          50.22 %
Throughput:                     85.28 Kbps
Normalized_Routing_Overhead:    245.95 %
Control_Overhead:               273
Delay:                          0.01 Seconds
Jitter:                         0.05
The number of malicious node is:                2
The ID of malicious node are:
0
5
```

FIGURE3.19 RESEAU EN PRESENCE NŒUDS MALVEILLANTS

```
No_of_Packets_Sent:             222
No_of_Packets_Received:         165
No_of_Packets_dropped:          0
Packet_Delivery_Ratio:          74.32 %
Packet_Dropping_Ratio:          25.68 %
Throughput:                     125.97 Kbps
Normalized_Routing_Overhead:    185.45 %
Control_Overhead:               306
Delay:                          0.01 Seconds
Jitter:                         0.03
```

FIGURE3.20 RESEAU SANS NŒUDS MALVEILLANTS

Il faut noter que dans le second cas, bien que le nombre de nœuds malveillants soit nul, et que le nombre de paquets perdus est nul aussi, on note quand même une différence entre le nombre de paquets envoyés et ceux reçus. Ceci est dû aux pertes de paquets par cause de saturation du réseau, d'erreurs matérielles ou même des performances des appareils qui parfois sont incapables de traiter les données comme il se doit.

Afin de pouvoir apporter une analyse des paramètres précédemment observés, nous allons, dans le chapitre suivant procéder à une automatisation de la partie d'analyse des performances en procédant à répéter la procédure plusieurs fois et sauvegarder les différentes valeurs de sortie.

Puis, grâce à l'outil GNUplot, nous pourrons avoir une visualisation graphique de ces paramètres. GNUplot est un outil très puissant en termes de présentation graphique, en deux ou trois dimensions, de fonctions numériques ou de données et nous sera d'une grande utilité.

3.5 Automatisation de l'analyse de performances

3.5.1 Script d'automatisation

Nous admettons que le fait de répéter une expérience plusieurs fois et moyenner les résultats nous aide à avoir de meilleurs résultats qui peuvent mieux refléter la réalité.

Nous allons répéter le processus 10 fois pour chacun des deux cas, présence ou absence des nœuds malveillants. Nous sauvegarderons les résultats dans un fichier texte. Nous allons par la suite copier ces résultats vers un fichier .CSV afin de calculer les valeurs moyennes des différents paramètres. Grâce à notre fichier .CSV nous pourrons visualiser les résultats en forme de présentation graphique en utilisant l'outil GNUplot.

Nous allons effectuer cela avec toujours un réseau de 20 nœuds dont 20% seront malveillants lors de l'un des cas cités juste avant.

Ci-dessous l'algorithme de simulation qu'on codera sous forme de script bash.

Algorithme 3 : Automatisation de la simulation

```
//Création des répertoires
mkdir mobility_files
mkdir –p Output/nam
mkdir –p Output/trace
mkdir –p Output/results
chmod 777 –R mobility_files
//Création des variables de stockage  //Création des variables pour stocker les valeurs moyennes
send=0                                avg_send=0
recv=0                                avg_recv=0
```

pdr=0 avg_pdr=0

pdrop=0 avg_pdrop=0

tp=0 avg_tp=0

nro=0 avg_nro=0

co=0 avg_co=0

dl=0 avg_dl=0

jitter=0 avg_jitter=0

//Création de l'entête du fichier CSV

echo '' '' | awk 'BEGIN{ printf

''Simulation_Number,No_of_Packets_Send,No_of_Packets_received,Packet_Delivery_Ratio, Packet_Dropping_Ratio,Throughput,Normalized_Routing_Overhead,Control_Overhead,Dela y,Jitter''}'>Output/Results/Final_Results.csv

//Création de la boucle principale avec n=10

while : ; do

((i++))

Création de 10 fichiers de mobilité.

Exécution du script TCL avec les fichiers de mobilités créés comme paramètre d'entrée

Appel du fichier .awk et redirection des valeurs des paramètres dans tmp_results

Extraction des paramètres du fichier tmp_results vers un fichier 1.txt

Copie des valeurs dans le fichier .csv

Calcul des valeurs moyennes et les sauvegarder au niveau des variables appropriées

si (i=10)

break

fin si

end

Copie des valeurs moyennes dans le fichier .csv

Suppression des fichiers temporaires : 1.txt, tmp_results

Déplacer chaque fichier dans son répertoire

Ecrire : ''Completed successfully''

L'exécution de ce script va d'abord créer 10 fichiers de mobilités qui vont servir comme entrée au script TCL. On aura par la suite 10 fichiers .nam et 10 fichiers .tr.

Chaque fichier trace sera donné en entré au script .awk afin de tirer tous les paramètres d'analyses. Le résultat du script .awk sera transcrit dans un fichier temporaire. Une fois les paramètres renseignés ils seront copiés d'abord dans un fichier texte puis dans un fichier .CSV

Ce dernier va comporter tous les paramètres des 10 simulations et les valeurs moyennes calculées. Ci-dessous un aperçu de ce fichier dans le cas de présence et d'absence des nœuds malveillants.

Simulation_Number	No_of_Packets_Sent	No_of_Packets_Received	No_of_Packets_Dropped	Packet_Delivery_Ratio	Packet_Dropping_Ratio	Throughput	Normalized_Routing_Overhead	Control_Overhead	Delay	Jitter
1.00000	237.00000	108.00000	129.000000	45.57000	54.43000	74.06000	189.81000	205.00000	0.01000	0.02000
2.00000	224.00000	124.00000	100.000000	55.35000	44.64000	122.74000	82.14000	184.00000	0.02000	0.03000
3.00000	232.00000	107.00000	125.000000	46.12000	53.88000	74.66000	203.74000	118.00000	0.01000	0.03000
4.00000	229.00000	114.00000	115.000000	49.78000	50.22000	87.24000	156.14000	178.00000	0.02000	0.03000
5.00000	108.00000	21.00000	87.000000	19.44000	80.56000	33.25000	766.67000	161.00000	0.01000	0.05000
6.00000	56.00000	7.00000	49.000000	12.50000	87.50000	22.22000	228.57000	86.00000	0.01000	0.03000
7.00000	229.00000	106.00000	123.000000	46.29000	53.71000	72.82000	197.17000	180.00000	0.03000	0.03000
8.00000	214.00000	103.00000	111.000000	48.13000	51.87000	81.34000	193.20000	169.00000	0.01000	0.02000
9.00000	226.00000	109.00000	117.000000	48.23000	51.77000	83.74000	166.97000	182.00000	0.01000	0.03000
10.00000	152.00000	73.00000	79.000000	48.03000	51.97000	85.64000	239.73000	175.00000	0.02000	0.03000
Average	190.70000	97.20000	93.50000	46.40900	53.59100	78.77100	342.41400	179.70000	0.01500	0.03500

Simulation_Number	No_of_Packets_Sent	No_of_Packets_Received	No_of_Packets_Dropped	Packet_Delivery_Ratio	Packet_Dropping_Ratio	Throughput	Normalized_Routing_Overhead	Control_Overhead	Delay	Jitter
1.00000	222.00000	221.00000	0.000000	99.55000	0.45000	169.02000	91.40000	202.00000	0.01000	0.02000
2.00000	212.00000	208.00000	0.000000	98.11000	1.89000	168.32000	105.77000	220.00000	0.01000	0.03000
3.00000	207.00000	207.00000	0.000000	100.00000	0.00000	162.82000	88.89000	184.00000	0.01000	0.03000
4.00000	217.00000	195.00000	0.000000	89.86000	10.14000	151.41000	128.72000	251.00000	0.01500	0.03000
5.00000	227.00000	177.00000	0.000000	77.97000	22.03000	132.96000	134.46000	238.00000	0.01000	0.03000
6.00000	219.00000	215.00000	0.000000	98.17000	1.83000	171.34000	92.56000	199.00000	0.01000	0.03000
7.00000	210.00000	104.00000	0.000000	49.52000	50.48000	182.61000	181.73000	189.00000	0.01000	0.03000
8.00000	216.00000	214.00000	0.000000	99.07000	0.93000	166.39000	86.45000	185.00000	0.01000	0.03000
9.00000	208.00000	194.00000	0.000000	93.27000	6.73000	158.50000	130.93000	254.00000	0.01000	0.03000
10.00000	219.00000	217.00000	0.000000	99.09000	0.91000	169.64000	84.33000	183.00000	0.01000	0.03000
Average	215.70000	195.20000	0.00000	90.46100	9.53900	153.30100	112.52400	210.50000	0.01050	0.02800

FIGURE3.21 (A) EN ABSENCE DE NŒUDS MALVEILLANTS FIGURE3.21 (B) EN PRESENCE DE NŒUDS MALVEILLANTS

FIGURE3.21 RESULTAT DU SCRIPT D'AUTOMATISATION

Une première analyse des tableaux obtenus montre qu'en absence des nœuds malveillants, il n'y aucun paquet perdu (colonne 4 **FIGURE3.20(A)**). Cette valeur ne tient pas en compte les paquets perdus pour des causes liées à la saturation du réseau mais seulement ceux perdus à cause de comportements malveillants des nœuds du réseau.

Avant de procéder à une analyse complète, nous allons d'abord transformer les valeurs des tableaux en représentation graphique en utilisant l'outil GNUplot.

3.5.2 Présentation graphique

3.5.2.1 Script et résultats

Grâce à l'outil GNUplot, nous allons pouvoir visualiser les paramètres précédents sous forme de graphes.

Nous commencerons par la présentation de l'algorithme transcris sous forme de script bash afin d'utiliser l'outil GNUplot.

Algorithme 4 : Présentation graphique des paramètres

1. Définir le style des données : ***set style data***
2. Définir comment les données d'entrées sont séparées (exemple : par des espaces, des virgules, des points-virgules … etc) : ***set datafile separator '',''***
3. Définir les bordures de l'affichage des graphes : ***set border 15***
4. Définir les tics sur les axes x et y : ***set xtics nommirror***

   set ytics nomirror

5. Définir la clé (ou légende) décrivant les tracés : set key
6. Définir le titre des axes x et y : ***set xlabel '' ajouter le titre ''***

   set ylabel ''ajouter un titre ''

7. Rediréger l'affichage vers un fichier spécifique : ***set output ''nom_du_graphe.png''***
8. Tracer le graphe : ***plot 'nom_du_fichier_d'entrée' using 'ajouter_la_colonne' title 'ajouter le titre' with linespoints lc rgb 'choisir_la_couleur'***

Nous aurons en sortie le tracer des graphes des paramètres renseignés précédemment comme le montre les figures suivantes.

3.5.2.2 Nombre de paquets envoyés

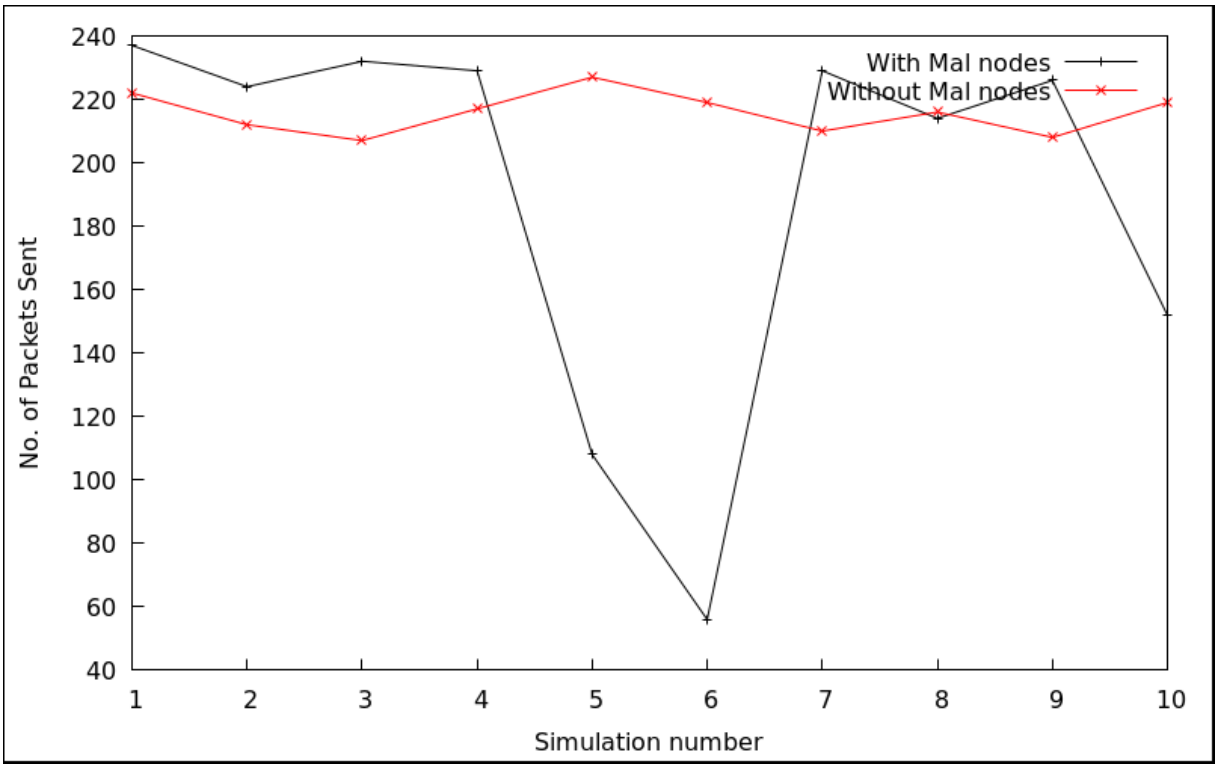

FIGURE3.22 NOMBRE DE PAQUETS ENVOYES

3.5.2.3 Nombre de paquets reçus

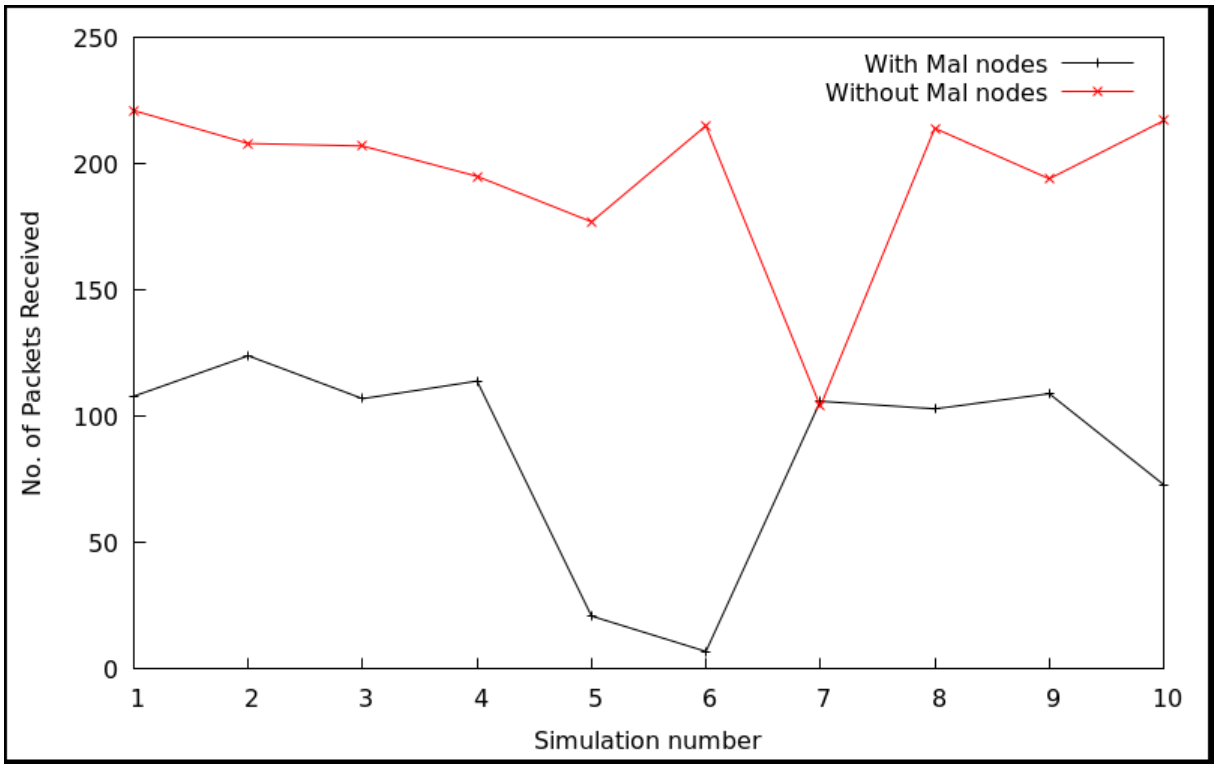

FIGURE3.23 NOMBRE DE PAQUETS REÇUS

3.5.2.4 Nombre de paquets perdus

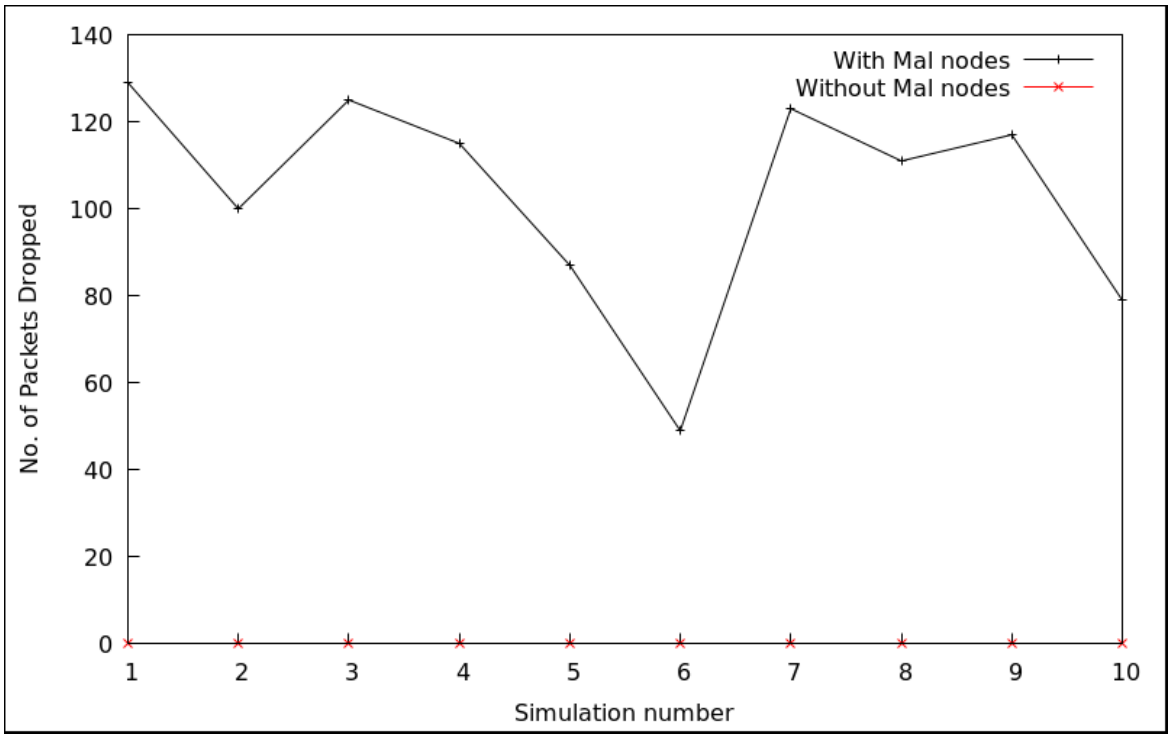

FIGURE3.24 NOMBRE DE PAQUETS PERDUS

3.5.2.5 Ratio de livraison de paquets

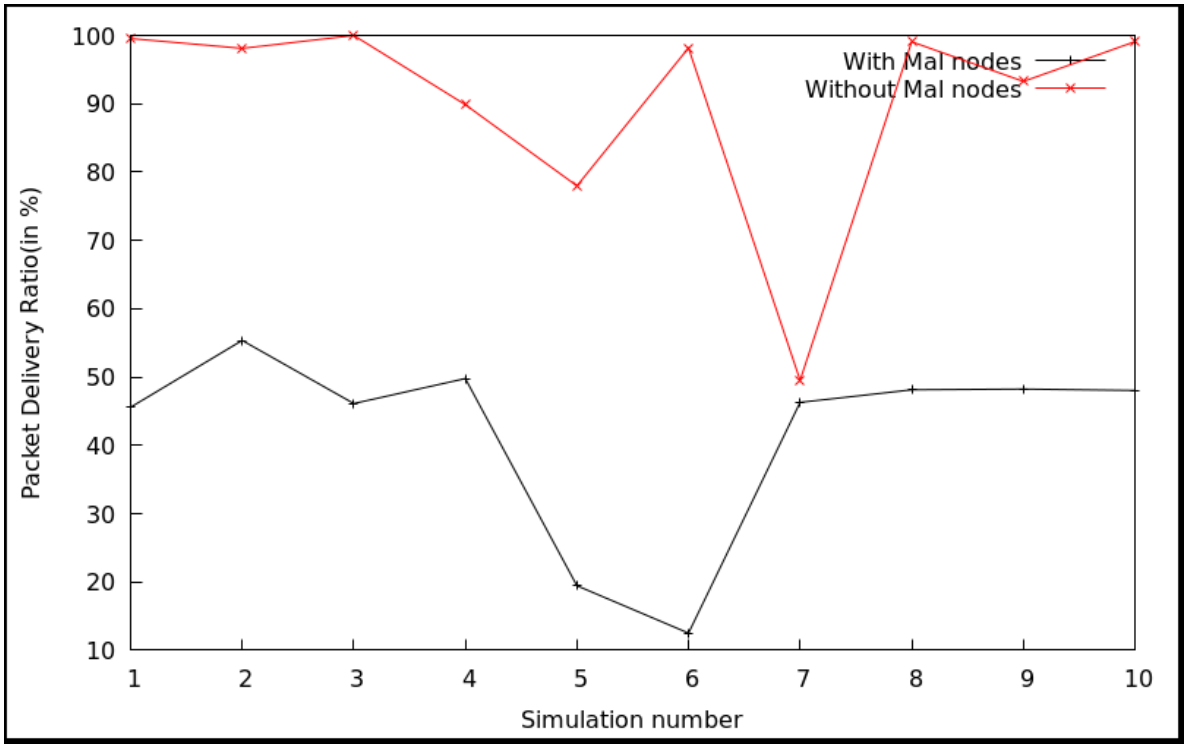

FIGURE3.25 RATIO DE LIVRAISON DE PAQUETS

3.5.2.6 Ratio de perte de paquets

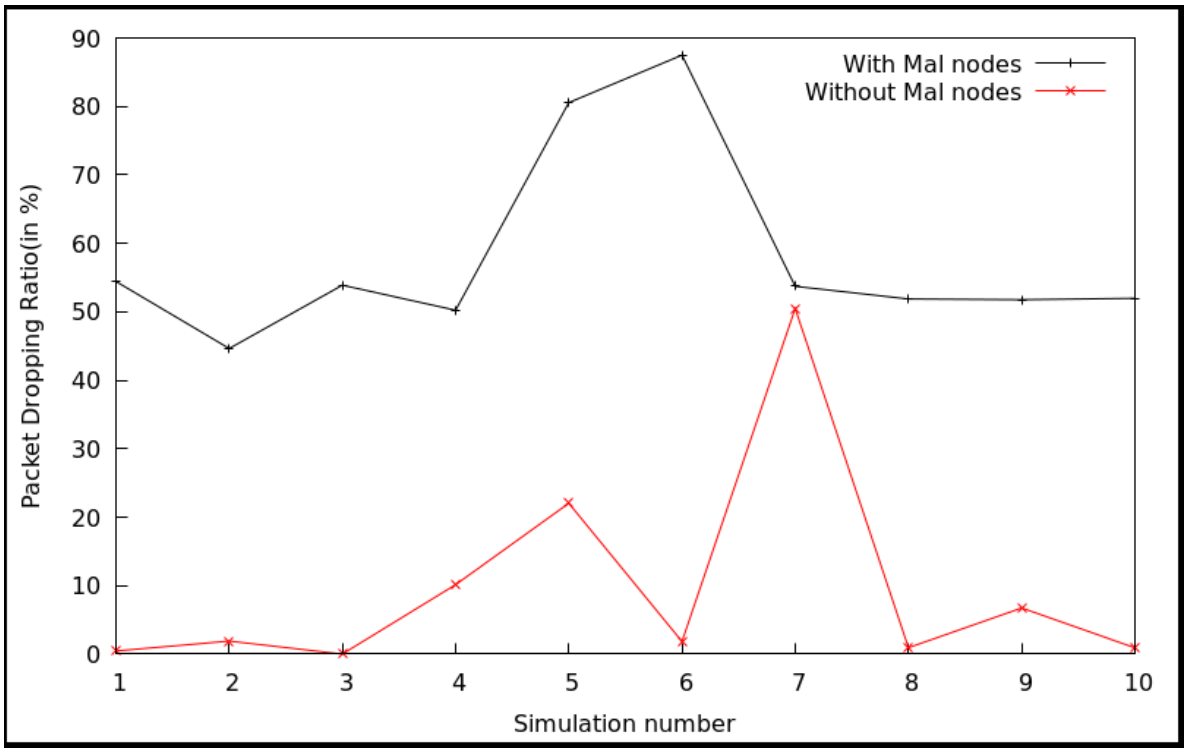

FIGURE3.26 RATIO DE PERTE DE PAQUETS

3.5.2.7 Débit

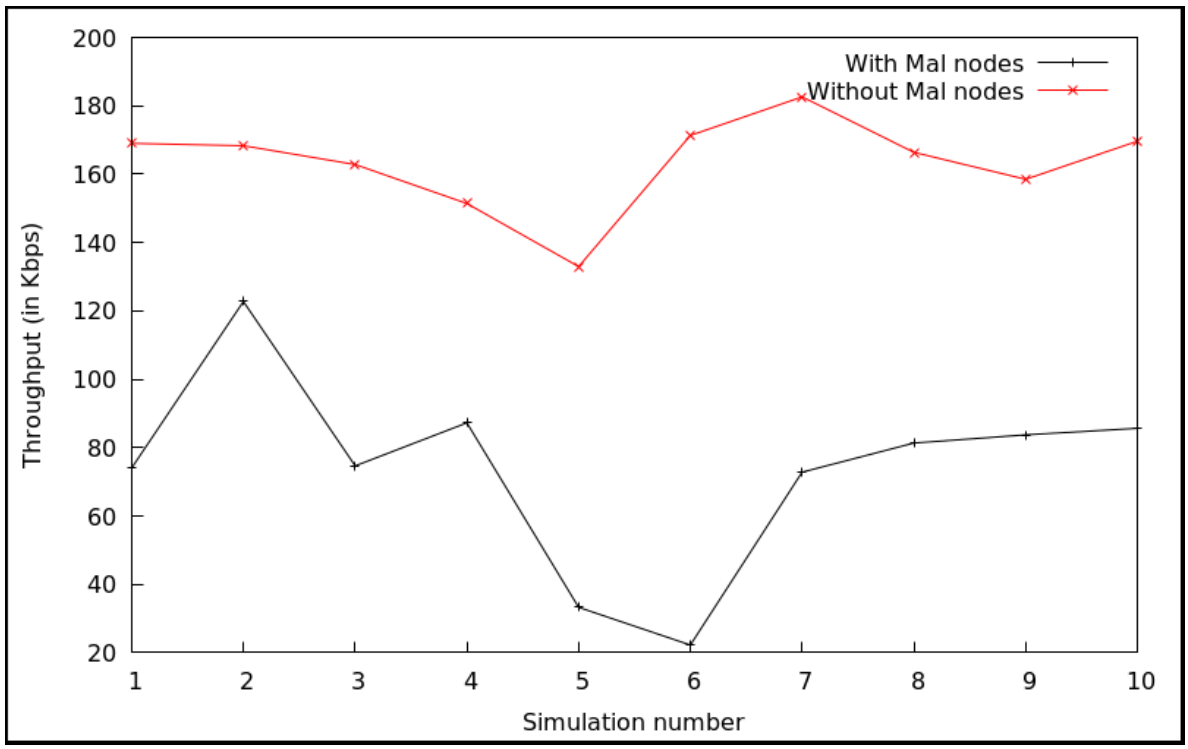

FIGURE3.27 DEBIT

3.5.2.8 Surcharge du routage normalisée

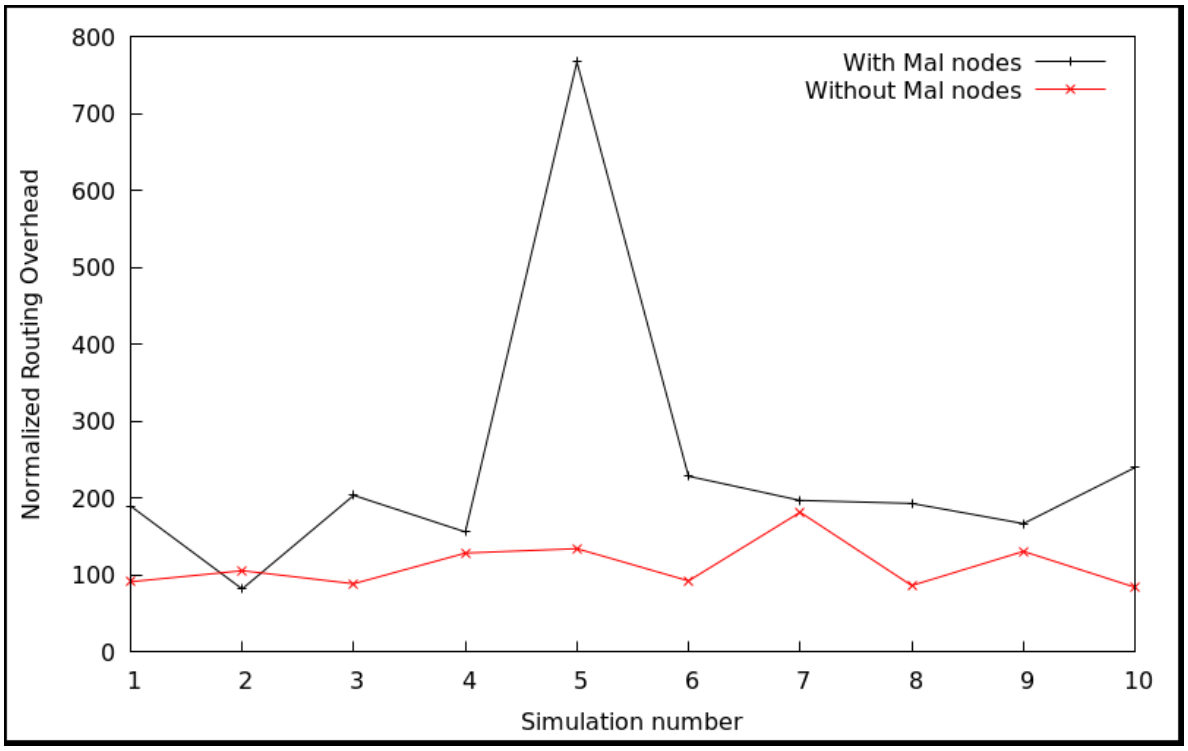

FIGURE3.28 SURCHARGE DE ROUTAGE NORMALISEE

3.5.2.9 Surcharge de control du réseau

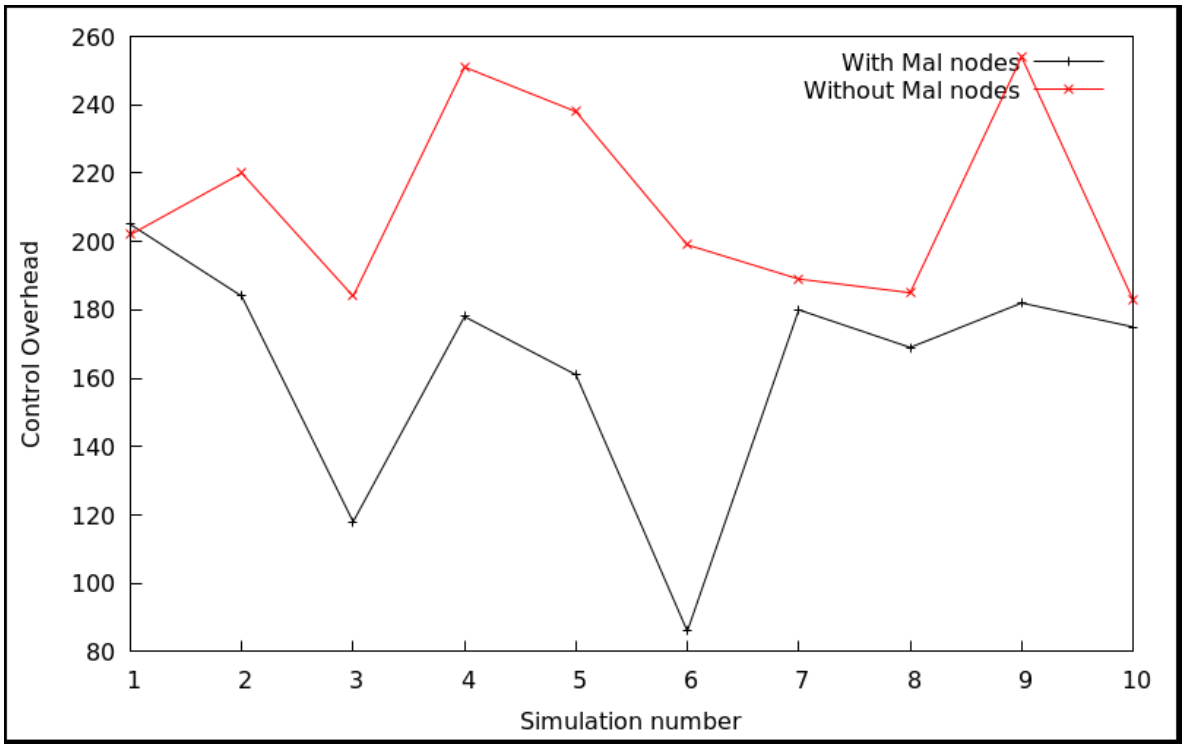

FIGURE3.29 SURCHARGE DE CONTROL DU RESEAU

3.5.2.10 Délai

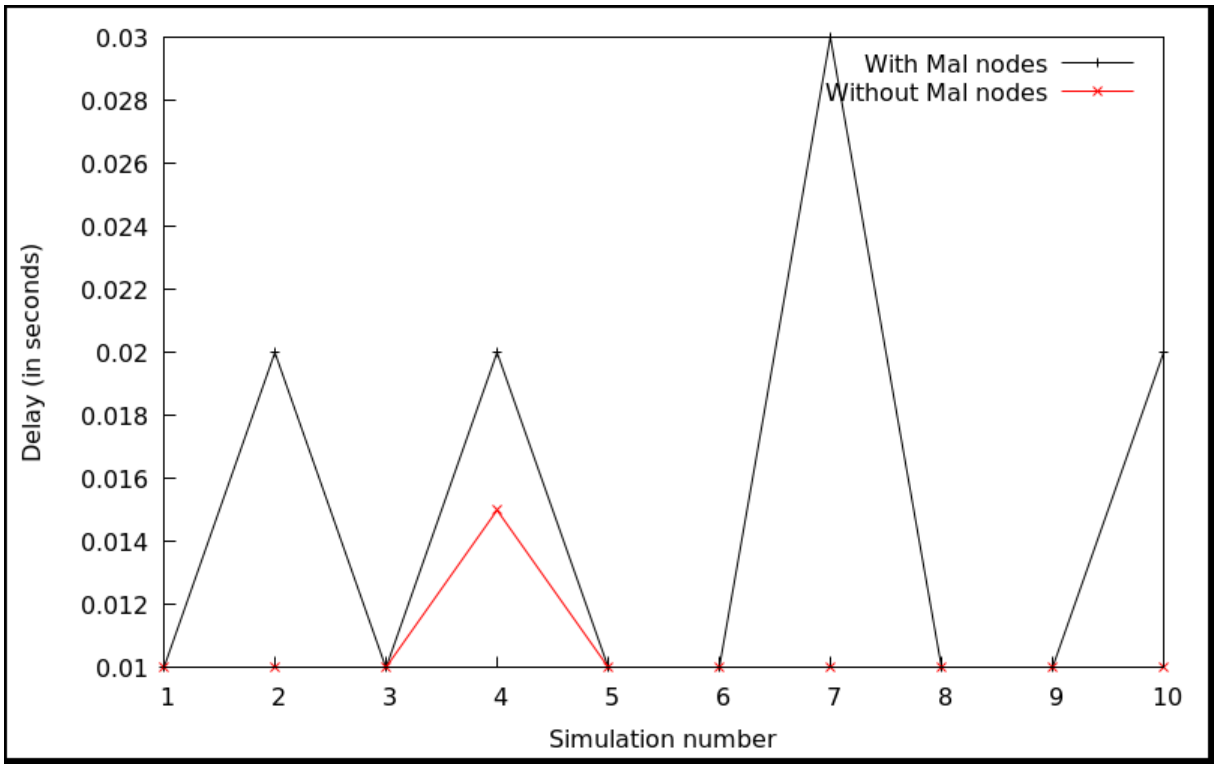

FIGURE3.30 DELAI

3.5.2.11 Gigue

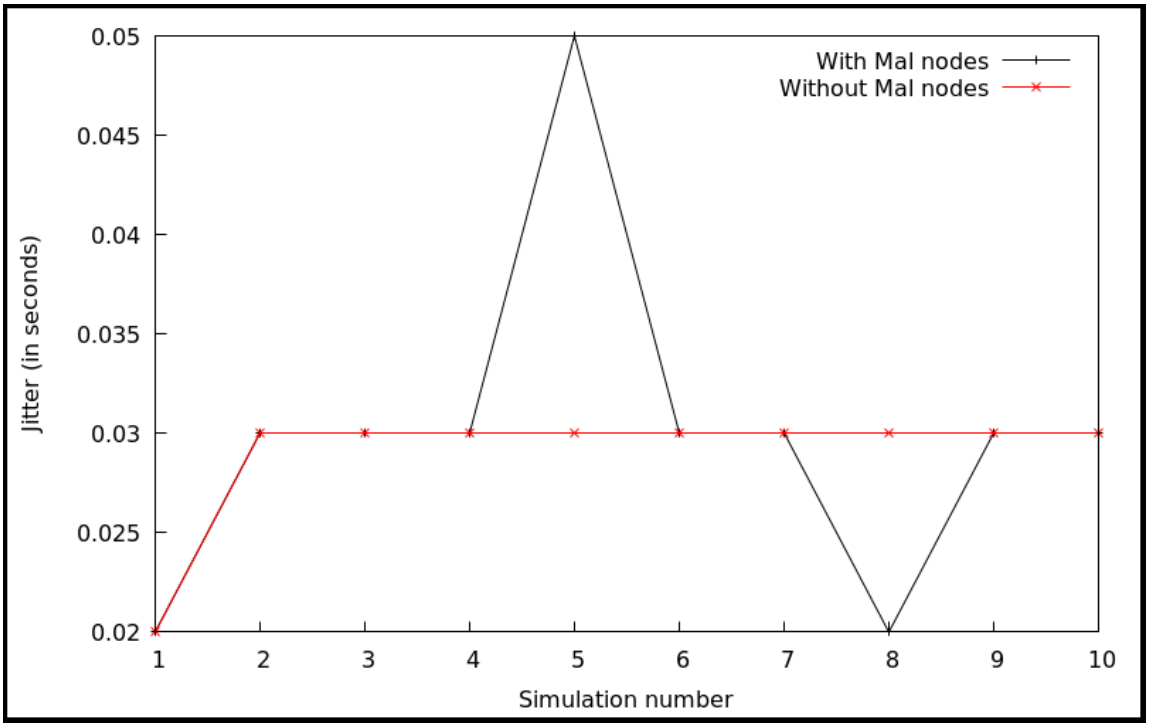

FIGURE3.31 GIGUE

3.5.3 Analyse des résultats

Dans cette partie, nous allons effectuer une analyse des résultats des deux tableaux de la **FIGURE3.21** ainsi que des présentations graphiques précédentes.

On remarque déjà qu'au cours des simulations, le nombre de paquets envoyés en présence de nœuds malveillants est supérieur au cas de la simulation sans nœuds malveillants (cas de simulations n° 1, 2, 3, 4, 7 et 9) **FIGURE3.21(B).** Par contre, le nombre de paquets reçus en présence de nœuds malveillants atteint une valeur maximale de 124 sur un total de 224 (simulation n°2) **FIGURE3.21(B)** contre 221 sur un total de 222 (simulation n°1) **FIGURE3.21(A).**

En traduisant ces valeurs en termes de pourcentage sur les dix simulations faites tout en moyennant les résultats obtenus, nous avons un taux de 46.409 % de livraison de paquets (en présence de nœuds malveillants) contre 90.461% (en absence de nœuds malveillants). En présence de seulement 20% de nœuds malveillants dans un réseau, le taux de livraison de paquet n'atteint même pas les 50%, qui veut dire que la moitié de la communication a été interrompue.

Nous pouvons aussi voir de façon claire que la **FIGURE3.26** montrant le taux de perte de paquets est complémentaire à la **FIGURE3.25** montrant le taux de livraison de paquets. Un résultat tout à fait logique.

Les nœuds malveillants causent la perte des paquets qui sont ensuite renvoyés par la source car cette dernière ne reçoit pas de message informant la réception des paquets par le destinataire. Ceci engendre, comme le montre la **FIGURE 3.28**, une augmentation de la surcharge du routage normalisé en présence des nœuds malveillants. Un maximum de 766.67 contre 128.72, soit une surcharge de 6x plus en présence de seulement 1/5 de nœuds malveillants.

La **FIGURE3.27** nous montre aussi la dégradation du débit en présence des nœuds malveillants. Valeur minimum de 22.22 Kbps en présence de nœuds malveillants contre 132.96Kpbs en leur absence. Un fort débit est signe d'un transfert avec succès de données entre la source et la destination et les résultats de la simulation sont parfaitement logique et prouvent encore l'impact des nœuds malveillants sur le bon fonctionnement du réseau.

La **FIGURE3.30** compare entre le délai mis par les paquets à router en présence et en absence de nœuds malveillants. Les paquets mettent plus de temps à arriver à destination en présence de nœuds malveillants signe de lenteur dans la communication.

La **FIGURE3.31** quant à elle montre les variations de la gigue lors de la simulation. Les résultats de la figure nous laissent constater que le réseau est instable en présence de nœuds malveillants et parce que l'arrivée des paquets varient dans le temps par cause des communications coupées. Chose qu'on ne note pas dans le cas d'un réseau où tous les nœuds sont sains et où la gigue paraît stable.

L'analyse des résultats nous a permis de prouver encore l'impact des nœuds malveillants non seulement en termes de consommation d'énergie mais aussi en termes de débit, de délai, de livraison de paquets et de surcharge de réseau. Les nœuds malveillants présentent un réel danger et leur impact peut être dévastateur pour le réseau.

3.6 Conclusion

Tout au long de ce chapitre, nous nous sommes familiarisés avec le simulateur NS-2. Nous avons d'abord réalisé le déploiement du réseau, puis on a procédé à l'injection de nœuds malveillants au sein de ce dernier.

Nous avons par la suite étudié l'impact des nœuds sur les performances du réseau. Nous avons noté que la présence d'un nœud malveillant impactait la bonne transmission des paquets, dégrade la qualité de service, augmente la consommation d'énergie des nœuds et diminue la durée de vie du réseau.

Un travail d'automatisation d'analyse des paramètres du réseau nous a montré que les nœuds impactaient tout autant le débit, la surcharge du réseau, la gigue ainsi que le délai.

Afin de palier cela, dans le prochain chapitre, nous allons intégrer un processus de détection de nœuds malveillants et une couche de protection se basant sur l'augmentation de la durée de vie du réseau.

Chapitre 4

Détection des intrusions et amélioration de la durée de vie du réseau

4.1 Introduction

Dans ce chapitre, nous allons commencer par présenter notre modèle de détection des nœuds malveillants au sein du réseau modélisé sous l'environnement NS2.

La durée de vie du réseau constitue un des aspects importants qui caractérisent se réseau. Elle renforce la sécurité des réseaux notamment en matière de disponibilité un des critères fondamentaux de la sécurité au même titre que l'intégrité et la confidentialité. Nous allons à la fin de ce chapitre mettre au point une méthode permettant d'augmenter la durée de vie du réseau et assurer une sécurité accrue.

4.2 Processus de détection des nœuds malveillant (Modèle de confiance)

Cette partie va être consacrée à l'implémentation du modèle de confiance. Ce dernier permettrait de détecter si un nœud est malveillant ou non.

Avant de présenter la logique de notre modèle de confiance, nous allons d'abord revenir sur le protocole utilisé lors du déploiement du réseau : le Trust AODV. Comme soulignée, ce dernier est un clonage du protocole AODV avec une couche de protection ajoutée lors du clonage.

Pour réaliser le clonage du protocole et les modifications apportées, nous utiliserons le logiciel « Eclipse », un logiciel open-source qui inclut un environnement de développement et un éditeur de texte pour les langages C et C++. Cette particularité nous permettra d'effectuer les changements nécessaires pour faire notre clonage. (cf. Annexe. C pour les étapes d'installation et de configuration).

Les étapes suivies pour faire fonctionner notre protocole sont décrites dans ce qui va suivre.

1. Dans le répertoire : *ns-allinone-2.35/ns-2.35,* créer le répertoire TAODV.
2. Copier tous les fichiers .h et .cc du répertoire AODV vers TAODV en les renommant.
3. Sur « Eclipse », ouvrir le répertoire TAODV et remplacer les instances : aodv → taodv et AODV → TAODV.
4. Dans le fichier « packet.h », inclure : static_const packet_t PT_TAODV, à la ligne 201.
   type == PT_TAODV, à la ligne 270.
   name_ [PT_TAODV] = ''TAODV'', à la ligne 350.
5. Dans le répertoire : *ns-allinone-2.35/ns-2.35/queue*, ouvrir le fichier « prique.cc », inclure : case PT_TAODV, à la ligne 93.
6. Dans le répertoire : *ns-allinone-2.35/ns-2.35/trace/*
   Dans *cmu-trace.h,* inclure : void format taodv_(Packet *p, int offset), à la ligne 162.
   Dans *cmu-trace.cc,* inclure : #include<taodv/taodv_packet>, à la ligne 53
   Sélectionner les lignes 862 à 965 et 1528-1530. Aller à Edit →Find/Replace, et remplacer : aodv → taodv et AODV → TAODV.
7. Dans le répertoire : *ns-allinone-2.35/ns-2.35/routing/rttable.h,* inclure :

```
classe Neighbor {
        friend class TAODV,
friend class rt_entry,
} à la ligne 75.
```

8. Dans le répertoire : *ns-allinone-2.35/ns-2.35/tcl/lib/ns-agent.tcl*, inclure : Agent/TAODV set port_ 0, à la ligne 192.
9. Dans le répertoire : *ns-allinone-2.35/ns-2.35/tcl/lib/ns-lib.tcl*, inclure :

```
TAODV {
        set ragent [create-taodv-agent $node
        } à la ligne 630.

simulator instproc create-taodv-agent {node} {
        set ragent [newAgent/TAODV [$node node-addr]]
        $self at 0.0 "$ragent start";
        $node set ragent_ $ragent
        return $ragent
        } à la ligne 872.
```

10. Dans le répertoire : *ns-allinone-2.35/tcl/lib/ns-mobilenode.tcl*, inclure :

```
set taodvonly [string first ''TAODV '' [$agent info class]]
if {$taodvonly != -1} {
        $agent if-queue [$self set ifq (0)]
}
```

11. Dans le répertoire : *ns-allinone-2.35/ns-2.35/tcl/lib/ns-packet.tcl*, inclure dans la section « Ad-hoc networks » :

```
TAODV, à la ligne 173.
```

12. Aller au *ns-allinone-2.35/ns-2.35/Makefile* et ajouter les lignes suivantes :

```
taodv/taodv_logs.o taodv/taodv.o\
taodv/taodv_rtable.o taodv/taodv_rqueue.o\    à la ligne 278
```

13. Aller au *ns-allinone-2.35/ns-2.35/Makefile.in* et ajouter les lignes suivantes :

```
taodv/taodv_logs.o taodv/taodv.o\
taodv/taodv_rtable.o taodv/taodv_rqueue.o\    à la ligne 278
```

14. Aller au *ns-allinone-2.35/ns-2.35/Makefile.vc* et ajouter les lignes suivantes :

```
taodv/taodv_logs.o taodv/taodv.o\
taodv/taodv_rtable.o taodv/taodv_rqueue.o\    à la ligne 263
```

15. Enregistrer toutes les modifications et fermer « eclipse ».
16. Aller au répertoire *ns-allinone-2.35/dei80211mr-1.1.4/src/InitTCL.cc*, inclure :

```
PacketHeaderManager set tab_(PacketHeader/TAODV), à la ligne 10
```

17. Aller au répertoire *ns-allinone-2.35/wpan/p802_15_4nam.cc*, inclure :

```
(strcmp(packet_info.nam(PT_TAODV), nam) ==0)PT_TAODV, à la ligne 173
```

18. Dans le répertoire *ns-allinone-2.35/ns-2.35*, ouvrir un terminal, et taper les commandes suivantes :

```
sudo ./configure –with –tcl-ver=8.5
sudo make clean
sudo make
sudo make install
```

Une fois les dernières commandes validées, l'écran affichera si une erreur a été détectée, sinon l'étape de création du protocole sera validée.

La création du protocole TAODV nous permettra, grâce notamment à l'activation du mode promiscuité à son sein :

- De suivre tous les paquets qui transitent au sein du réseau.
- De créer un agent TAODV.
- De créer la classe « neighbor » et maintenir la liste des voisins
- Implémenter les procédures de création de liste et de comparaison de liste qui seront requises pour notre modèle de confiance.

Une fois cette étape effectuée, nous allons entamer la partie du modèle de confiance afin de détecter la présence de nœuds malveillants dans un réseau.

Pour modéliser ce processus, chaque nœud se verra attribué une table. Cette dernière tiendra les identifiants des nœuds voisins avec lesquels chaque nœud échange des informations et la valeur de confiance qu'il attribue à chacun d'eux suite à son comportement.

Pour cela, un nœud doit d'abord connaitre son voisinage et maintenir la liste de ces derniers. Ceci se fera grâce à la classe « neighbor » et l'envoie régulier d'un « Hello Packet » en attendant la réponse. Si le voisin répond avant qu'un certain délai expire le nœud maintien la relation, sinon il supprimera l'ID du nœud de ses nœuds voisins ainsi que tous les chemins qui l'incluent.

Afin d'expliquer la procédure de confiance entre les nœuds, on va s'appuyer sur l'exemple suivant :

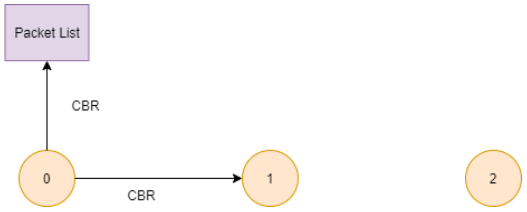

FIGURE4.1 PROCEDURE DE CONFIANCE ENTRE LES NŒUDS

La **FIGURE4.1** montre 3 nœuds. Soit le nœud « 0 » la source et le nœud « 2 » le destinataire. Le nœud « 0 » va commencer par envoyer un paquer 'CBR' (*Constant Bit Rate*) au nœud « 1 » qu'il va aussi ajouter à sa « Packet List ».

Si le nœud « 1 » transfère le paquet alors sa valeur de confiance est augmentée par le nœud « 0 » qui va alors supprimer le paquet CBR de la « Packet List ». Après un certain seuil, si le paquet se trouve toujours dans la Packet List, alors ceci prouve que le nœud « 1 » ne l'a pas transféré.

Grâce à ceci, un premier jugement peut être pris sur la malveillance d'un nœud. Il est à souligner qu'un nœud peut couper la transmission des messages pour d'autres raisons comme un dysfonctionnement ou une panne. Pour palier cela, une procédure « CheckMal » sera implémenté et nous permettra de juger si le nœud présente réellement un danger ou c'est juste un dysfonctionnement momentané.

L'implémentation de ce modèle passe par les étapes suivantes :

1. Maintien de deux listes : Trust List et la Packet List.
2. Définir toutes les variables de stockage.

3. Implémentation de la méthode « tap » : vérifie la correspondance des paquets au niveau de la Trust List et la Packet List et est responsable de la suppression des paquets transférés (ceci correspond au fait qu'un paquet est présent dans les deux listes).
4. Déclaration du timer « MalTimer » : Ce dernier détiendra un compteur qui passer un seuil nous informe qu'un paquet n'a pas été transféré par un nœud donné.
5. Implémentation de la méthode « CheckMal » : Vérifie si un nœud représente un dysfonctionnement, est en panne ou présente un comportement malveillant.
6. Implémentation de la fonction « trust_print » : Va nous permettre l'écriture de toutes les tables de confiance.

La sortie de ce code produira un fichier de tables. Le nombre de tables sera égal au nombre de nœuds du réseau. La **FIGURE4.2** montre un exemple du résultat de l'exécution.

Dans notre cas, la valeur 0.5 est la valeur attribuée à chaque voisin lors de la phase de découverte de voisinage. Cette valeur tend vers 1 pour désigner une confiance totale signe d'un comportement exemplaire du voisin. Par ailleurs, elle diminue jusqu'à atteindre 0 suite à un comportement malveillant de ce dernier.

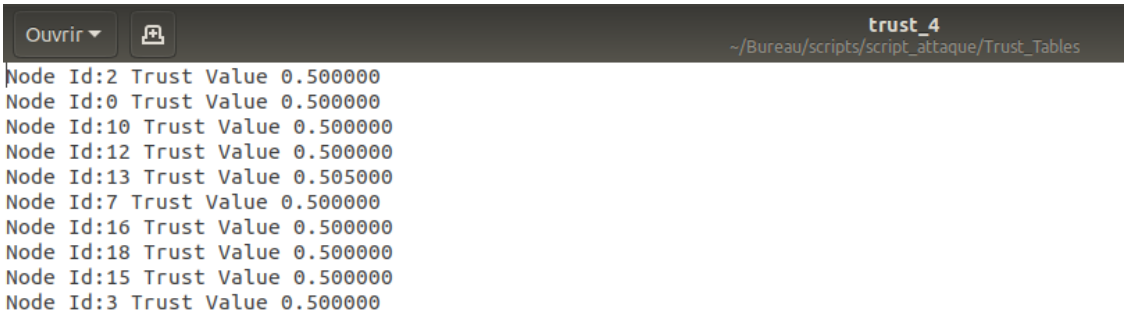

FIGURE4.2 APERÇU D'UNE TABLE DE CONFIANCE

La figure 4.3 montre la valeur attribuée à un nœud dont le comportement est malveillant.

FIGURE4.3 VALEUR DE CONFIANCE D'UN NŒUD MALVEILLANT

Grace au modèle implémenté, chaque nœud dont le comportement est malsain sera détecté par sa valeur de confiance. Ceci représente une première étape de notre travail. La seconde sera consacrée à assurer la disponibilité des nœuds en augmentant leur durée de vie.

4.3 Augmentation de la durée de vie du réseau

4.3.1 Pourquoi ?

La disponibilité des nœuds est un des aspects de la sécurité des réseaux. Les nœuds, particulièrement ceux qui contribuent au routage et par la même occasion à la détection d'intrusions ne doivent pas s'éteindre brusquement au cours de leurs tâches. Aussi, un nœud qui épuise sa batterie et s'éteint bouleverse tous ses voisins car toutes les routes qui l'incluent doivent être effacées et de nouvelles routes doivent être établies. Cette tâche, en plus de consommer davantage d'énergie aux nœuds peut être le moyen pour qu'un nœud malveillant intègre le réseau.

Afin de minimiser la consommation des ressources au sein du réseau et assurer une meilleure sécurité, nous allons opter pour une organisation des nœuds en clusters. La procédure suivie sera expliquée dans ce qui va suivre.

4.3.2 Comment ?

Nous admettons que l'énergie du réseau est la somme des énergies des nœuds le constituant et que le réseau s'éteint complètement lorsque le dernier nœud épuise sa batterie.

La principale tâche des nœuds est le routage des paquets et le maintien des routes, cette dernière représente aussi la tâche qui consomme la plus grande énergie.

L'idée est donc de limiter les routages de longues distances. En effet, ils incluent de nombreux nœuds intermédiaires et une plus grande consommation d'énergie. L'idéal est de transmettre le paquet du nœud source au destinataire directement mais cela n'est pas très pratique. Nous allons donc essayer de réduire les grandes distances tant que possible.

Pour cela, le réseau va être organisé en clusters, ces derniers seront homogènes et incluront un nombre similaire ou assez proche les uns des autres.

Les nœuds du même cluster communiquent entre eux assurant une distance minimale possible.

Si un nœud du cluster (0) veut communiquer avec un nœud du cluster (1) à titre d'exemple, ils devront passer par les CHs (Cluster-Head). Ces derniers se chargeront de router les informations intra-clusters. Pour ne pas retomber dans le précédent scénario où les paquets parcourent de longs chemins vers le destinataire, les CHs seront programmés de façon à être proches les uns des autres minimisant la distance qui les sépare.

4.3.3 Quels avantages ?

L'organisation du réseau en clusters offre une meilleure optimisation des ressources et ce sous différents angles.

Au sein du même cluster, les nœuds sont proches les uns des autres et la majorité des communications ne requièrent au plus qu'un ou deux nœuds intermédiaires.

Pour les communications entre les nœuds de clusters différents, les paquets transitent par les CHs, ces derniers assureront le transfert des paquets directement vers le cluster

approprié. Comme tous les nœuds sont rattachés à un cluster, le paquet aura un chemin comme le suivant :

Nœud (x) ----- CH(a) ------CH(b) ------- Nœud (y) ce qui fait un chemin de 4 nœuds incluant la source et le destinataire.

4.3.4 Simulation

Afin d'atteindre notre objectif, nous allons revenir sur notre précédent réseau et effectuer les changements nécessaires.

Nous allons retravailler sur le même réseau de 20 nœuds, dont 3 seront des nœuds CHs et 17 des nœuds mobiles. Nous allons ajouter un nœud filaire qui va servir de station de gestion.

Le réseau va inclure cette fois-ci des nœuds sans fil (les nœuds mobiles) et des nœuds filaires (les CHs et la station de gestion). Ce choix nous permettra une meilleure optimisation de l'énergie pour les clusters-head qui peuvent être amenés à router plusieurs paquets.

Afin de bien gérer les nœuds mobiles et les nœuds statiques, nous allons déclarer deux domaines. Chaque domaine sera divisé en clusters et chaque cluster comportera un nombre de nœuds mobiles.

Une façon de programmer cela serait d'utiliser l'adressage suivant :

« 0.0.0 » pour déclarer un nœud. Le premier chiffre fait référence au domaine, le second au cluster et le dernier au numéro du nœud. Ainsi, cet exemple sert à déclarer le nœud « 0 » au sein du cluster « 0 » dans le domaine « 0 ».

Le domaine « 0 » est celui des nœuds filaires, le domaine « 1 » est celui des nœuds non filaires.

4.3.4.1 Script de simulation

Algorithme 5 : programmation du réseau en clusters

```
//Déclaration des paramètres de simulation //FIGURE4.4 (A)

Déclaration du type de canal
Déclaration du modèle de propagation radio
Déclaration du type d'antenne
Déclaration du type de la couche de liaison
Déclaration du type d'interface réseau
Déclaration du protocole de routage
Déclaration du nombre de nœuds mobiles

// Création des variables de stockage // FIGURE4.4 (B)
 val start_simulation =0.0
val stop_simulation = 40.0
val nbr_wired_nodes = 2
val nbr_wireless_nodes =17
val nbr_CH = 3
```

Création d'une nouvelle instance de simulation
Ouverture du fichier trace en mode écriture
Ouverture du fichier nam en mode écriture

//Création des nœuds filaires // **FIGURE4.4 (C)**

Adressage des nœuds filaires : (0.0.0 0.1.0)
W(0) <- adresse (0.0.0)
W(1) <- adresse (0.1.0)

//Création CHs // **FIGURE4.4 (D)**

Adressage des CHs et des nœuds mobiles de chaque cluster : (1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 1.0.5 1.0.6 ; 1.1.0 1.1.1 1.1.2 1.1.3 1.1.4 1.1.5 ; 1.2.0 1.2.1 1.2.2 1.2.3 1.2.4 1.2.5 1.2.6)

CH(0) <- adresse (1.0.0)
CH(1) <- adresse (1.1.0)
CH(2) <- adresse (1.2.0)

Fixé les coordonnées des CHs :
BS(0) set X_ 200.0 Y_ 200.0 Z_ 0.0
BS(1) set X_ 250.0 Y_ 250.0 Z_ 0.0
BS(2) set X_ 150.0 Y_ 150.0 Z_ 0.0

//Création des nœuds mobiles : exemple pour un seul nœud, la même procédure sera reprise pour les autres nœuds mobiles // **FIGURE4.4 (E)**

node-config wired-routing OFF
node_ (0) <- adresse (1.0.1)
$node_(0) attach_cluster CH(0)

//Initialiser la position de départ des nœuds // **FIGURE4.4 (F)**
Pour i **de** 1 **à** nbr_noeuds_sans_fil
    initial_node_pos node_(i) 20
    i++
Fin Pour

//Création des liens entre les nœuds filaires et les CHs // **FIGURE4.4 (G)**
set duplex_link $W(0) $W(1) 5Mb 2ms DropTail
set duplex_link $W(1) $CH(0) 5Mb 2ms DropTail
set duplex_link $W(1) $CH(1) 5Mb 2ms DropTail
set duplex_link $W(1) $CH(2) 5Mb 2ms DropTail

Déclaration du fichier de mobilité des nœuds
Création des connexions UDP entre les nœuds mobiles

//Avertir les CHs de la fin de la simulation // **FIGURE4.4 (H)**
At val stop_simulation ‘’CH(0) reset’’
At val stop_simulation ‘’CH(1) reset’’
At val stop_simulation ‘’CH(2) reset’’

//Procédure de fin de simulation // **FIGURE4.4 (I)**
proc (finish)
close trace file
close nam file
ns nam file.nam
fin proc

puts ‘’Start Simulation’’

```
set val(chan)         Channel/WirelessChannel  ;# Channel type
set val(prop)         Propagation/TwoRayGround ;# Radio-propagation model
set val(ant)          Antenna/OmniAntenna      ;# Antenna type
set val(ll)           LL                       ;# Link layer type
set val(ifq)          Queue/DropTail/PriQueue  ;# Interface queue type
set val(ifqlen)       50                       ;# Max packet in ifq
set val(netif)        Phy/WirelessPhy          ;# Network interface type
set val(mac)          Mac/802_11               ;# MAC type
set val(adhocRouting)           TAODV                     ;# Routing protocol
set val(x)            800                      ;# X length
set val(y)            800                      ;# Y length
```

FIGURE4.4(A) DECLARATION DES PARAMETRES DE SIMULATION

```
set val(start)   0.0                            ;# time to start simulation.
set val(stop)   40                             ;# time to stop simulation
set val(nn)           17                        ;# Number of mobilenodes
set num_wired_nodes      2
set num_ch_nodes         3
```

FIGURE4.4(B) CREATION DES VARIABLES DE STOCKAGE

```
#create wired nodes
set temp {0.0.0 0.1.0}        ;# hierarchical addresses for wired domain
for {set i 0} {$i < $num_wired_nodes} {incr i} {
    set W($i) [$ns_ node [lindex $temp $i]]
}
```

FIGURE4.4(C) CREATION DES NŒUDS FILAIRES

```
#create base-station node
set temp {1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 1.0.5 1.0.6
1.1.0 1.1.1 1.1.2 1.1.3 1.1.4 1.1.5
1.2.0 1.2.1 1.2.2 1.2.3 1.2.4 1.2.5 1.2.6 }   ;# hier address to be used for wireless
                                    ;# domain
set CH(0) [$ns_ node 1.0.0]
$BS(0) random-motion 0              ;# disable random motion

set CH(1) [$ns_ node 1.1.0]
$BS(1) random-motion 0

set CH(2) [$ns_ node 1.2.0]
$BS(2) random-motion 0

#provide some co-ord (fixed) to base station node
$CH(0) set X_ 200.0
$CH(0) set Y_ 200.0
$CH(0) set Z_ 0.0

$CH(1) set X_ 250.0
$CH(1) set Y_ 250.0
$CH(1) set Z_ 0.0

$CH(2) set X_ 150.0
$CH(2) set Y_ 150.0
$CH(2) set Z_ 0.0
```

FIGURE4.4(D) CREATION CHS

```
#configure for mobilenodes
$ns_ node-config -wiredRouting OFF

# attaching the mobile nodes to Cluster-Head 0
set node_(0) [$ns_ node 1.0.1]
$node_(0) cluster-head [AddrParams addr2id [$CH(0) node-addr]]
set node_(1) [$ns_ node 1.0.2]
$node_(1) cluster-head [AddrParams addr2id [$CH(0) node-addr]]
set node_(2) [$ns_ node 1.0.3]
$node_(2) cluster-head [AddrParams addr2id [$CH(0) node-addr]]
set node_(3) [$ns_ node 1.0.4]
$node_(3) cluster-head [AddrParams addr2id [$CH(0) node-addr]]
set node_(4) [$ns_ node 1.0.5]
$node_(4) cluster-head [AddrParams addr2id [$CH(0) node-addr]]
set node_(5) [$ns_ node 1.0.6]
$node_(5) cluster-head [AddrParams addr2id [$CH(0) node-addr]]
```

FIGURE4.4(E) CREATION DES NŒUDS MOBILES

```
for {set j 0} {$j < 17} {incr j} {
    $ns_ initial_node_pos $node_($j) 20
}
```

FIGURE4.4(F) INITIALISER LA POSITION DE DEPART DES NŒUDS

```
#create links between wired and CH nodes

$ns_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail
$ns_ duplex-link $W(1) $CH(0) 5Mb 2ms DropTail
$ns_ duplex-link $W(1) $CH(1) 5Mb 2ms DropTail
$ns_ duplex-link $W(1) $CH(2) 5Mb 2ms DropTail

$ns_ duplex-link-op $W(0) $W(1) orient down
$ns_ duplex-link-op $W(1) $CH(0) orient down
$ns_ duplex-link-op $W(1) $CH(1) orient left-down
$ns_ duplex-link-op $W(1) $CH(2) orient right-down
```

FIGURE4.4(G) CREATION DES LIENS ENTRE LES NŒUDS FILAIRES ET LES CHS

```
$ns_ at $val(stop).0 "$CH(0) reset";
$ns_ at $val(stop).0 "$CH(1) reset";
$ns_ at $val(stop).0 "$CH(2) reset";

$ns_ at $val(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $val(stop).0001 "stop"
```

FIGURE4.4(H) AVERTIR LES CHS DE LA FIN DE LA SIMULATION

```
proc stop {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
}

puts "Starting Simulation..."
$ns_ run
```

FIGURE4.4(I) PROCEDURE DE FIN DE SIMULATION

4.3.4.1 Résultats

Le résultat de la compilation du script précédent sous NS-2 est montré dans la **FIGURE4.4(B)** suivante avec un zoom sur les nœuds filaires représentants les Clusters-Head.

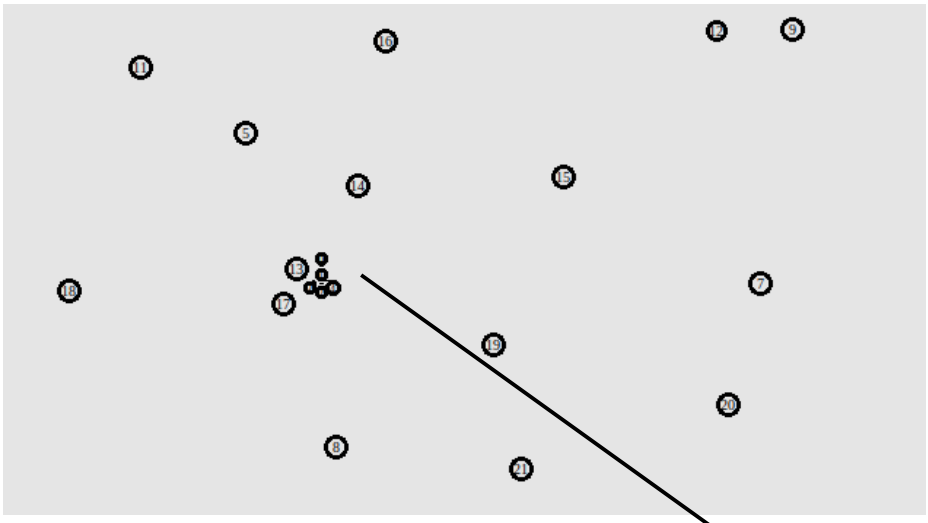

FIGURE4.4 (A) VISUALISATION DU RESEAU

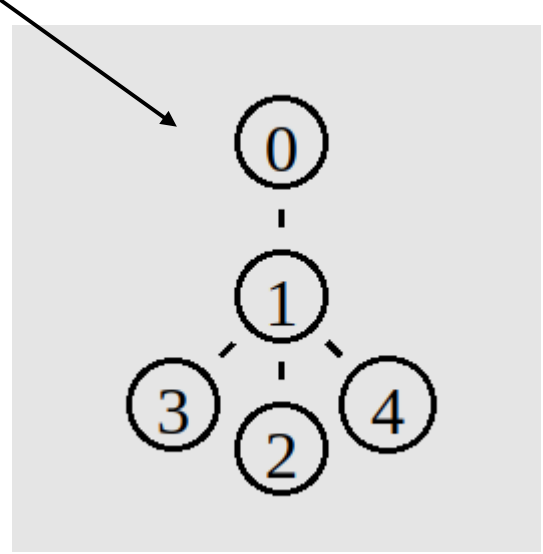

FIGURE4.4 (B) ZOOM SUR LES NŒUDS FILAIRES

FIGURE4.4 REORGANISATION DU RESEAU EN CLUSTERS

Maintenant que le réseau est divisé en domaines et que chaque nœud appartient à un cluster nous allons comparer l'évolution de l'énergie des nœuds constituants ce réseau avec celui étudié précédemment.

Soit le réseau_1 le réseau composé des 20 nœuds mobiles et le réseau_2 le réseau organisé en clusters.

Les mêmes paramètres de simulation ont été renseignés dans le cas des deux réseaux. La simulation des deux réseaux est faite pour t=50 sec.

Pour le réseau_1 la transmission se fait du nœud 1 vers le 2 et du nœud 11 vers le 19.

Pour le réseau_2 la transmission se fait du nœud 1 vers le 2 (au sein du même cluster) et du nœud 10 vers le nœud 12 (clusters différents)

Afin de pouvoir comparer les résultats des deux simulations, nous procédons à l'extraction des données comme suit :

//Création des variables de stockage :

nbr-of-node =0

nbr-of-rout-node=0

id_1[nbr-of-node]= 0

id_2[nbr-of-node]= 0

initial_energy=0

final_energy=0

//Création des fichiers de sauvegarde :

- Ouverture d'un fichier .tmp en mode écriture
- Ouverture d'un fichier .txt en mode écriture

//Extraction des informations du réseau et des nœuds.

- Déclarer le fichier .tr comme entrée au fichier .awk.
- Extraction du nombre de nœud et le stocker dans la variable nbr-of-node.
- Dans le fichier .txt, ecrire le nombre de nœud du résau.
- Extraction de l'énergie initiale de chaque nœud et son ID. Stocker l'ID dans le tableau id1[nbr-of-node] et son énergie dans le fichier .tmp
- Extraction de l'énergie finale de chaque nœud et son ID. Stocker l'ID dans le tableau id2[nbr-of-node] et son énergie dans le fichier .tmp

pour i **de** 0 **à** nbr-of-node

pour j **de** 0 **à** nbr-of-node

si (id1[i]== id2[j])

nbr-of-rout-node ++

Ecrire le nombre de nœud qui contribue au routage dans le fichier .txt

Copier l'ID du nœud dans le fichier .txt

Copier l'énergie initiale du nœud dans le fichier .txt

Copier l'énergie finale du nœud dans le fichier .txt

fin si

fin pour

fin pour

Nous allons appliquer les étapes précédentes sur les deux réseaux dont nous disposons, la **FIGURE4.14** suivante montre les résultats obtenus dans chacun des cas

```
Nom du réseau : réseau_(1)
Nombre de noeuds du réseau : 20
Nombre de noeuds qui contribuent au routage : 5
Energie initiale des noeuds :
Id : 0 , E : 2.773559
Id : 1 , E : 1.075818
Id : 3 , E : 1.615438
Id : 11 , E : 1.125179
Id : 14 , E : 0.210119

Energie Finale des noeuds :
Id : 0 , E : 0.048702
Id : 1 , E : 0.009457
Id : 3 , E : 0.384325
Id : 11 , E : 1.125179
Id : 14 , E : 0.210119
```

FIGURE4.14(A) RESULTAT DU RESEAU 1.

```
Nom du réseau : réseau_(2)
Nombre de noeuds du réseau : 21
Nombre de noeuds qui contribuent au routage : 3
Energie initiale des noeuds :
Id : 1 , E : 1.931227
Id : 10 , E : 1.531018
Id : CH(1) , E : 2.272502

Energie Finale des noeuds :
Id : 1 , E : 1.531018
Id : 10 , E : 0.597322
Id : CH(1) , E : 2.265138
```

FIGURE4.14(B) RESULTAT DU RESEAU 2

FIGURE4.5 EXTRACTION DES DONNEES D'ENERGIES DES NŒUDS

L'analyse des résultats obtenus nous permet d'apporter les conclusions suivantes :

Nous remarquons dans un premier temps, pour le même temps de simulation, et la même configuration d'envoi de paquets :

D'abord, dans le second réseau le nombre de nœuds contribuant au routage est plus petit que dans le premier. Ensuite, nous voyons clairement qu'à la fin de la simulation, les nœuds du premier réseau sont au bas de leur batteries tandis que ceux du second réseau garde plus d'énergie.

Outre les résultats cités ci-dessus, nous pouvons ajouter que le fait d'organiser le réseau en clusters ajoute une couche de sécurité. En effet, chaque nœud est rattaché à un CH par son adresse. Ceci rend la procédure d'intrusion plus difficile. L'attaquant doit, afin d'introduire un nœud malveillant, connaitre la topologie du réseau, l'architecture des nœuds et l'adressage de chaque cluster.

Grâce à ce petit exemple, nous pouvons déjà dire que l'organisation du réseau en clusters permet aux nœuds de consommer moins d'énergie et ceci assure leur disponibilité pour une durée plus longue. Ceci contribue d'une manière indirecte mais efficace à la sécurité des réseaux IoT.

4.4 Conclusion

Dans ce chapitre, nous avons vu d'une part, l'implémentation de notre modèle de confiance. Afin de protéger notre réseau, nous nous sommes basé sur la mise en place d'un environnement de confiance entre les nœuds voisins du réseau. Et grâce au comportement des nœuds, nous avons pu détecter la présence des nœuds malveillants.

D'autre part, nous avons rajouté une couche de protection et ce en augmentant la durée de vie du réseau. Pour ce faire, nous avons divisé le réseau en clusters et associé chaque nœud à

un cluster. Nous avons ensuite comparé l'énergie des nœuds dans le cas d'un réseau sans clusters avec celui organisé en clusters.

Les résultats des simulations montrent que cette méthode permet aux nœuds de consommer moins d'énergie et par conséquent nous augmenterons leur durée de vie et celle du réseau global offrant ainsi une meilleure disponibilité des nœuds.

Conclusion générale

Les réseaux IoT est une technologie qui envahit notre quotidien. Cette technologie est aussi largement exploitée dans le domaine industriel notamment dans le contexte de l'industrie 4.0 donnant naissance aux réseaux IoT industriels IIoT. Les réseaux IoT sont souvent déployés dans des environnements hostiles soumis à d'innombrables perturbations de l'environnement extérieur. Désormais cette situation rend les IoT très vulnérables et les expose à plusieurs types de menaces.

Dans leurs différents domaines d'usages, la collecte des données à partir des objets connectées doit se faire désormais en temps réel. Cet état de fait, nous impose de garantir une disponibilité de ces réseaux et d'assurer la confidentialité et l'intégrité des données traitées, véhiculées et stockées sur ces réseaux.

Dans le cadre de ce mémoire nous nous sommes intéressés aux aspects de sécurité inhérents à ces réseaux. Il s'agissait dans un premier temps d'identifier la présence de nœuds malveillants au sein du réseau. Cette présence de nœuds malveillants sur un réseau IoT peut provoquer plusieurs problèmes de sécurité. Ces derniers peuvent aller d'une simple écoute du trafic jusqu'à la perturbation de la communication et du fonctionnement du réseau.

Une fois ces nœuds malveillants injectés sur le réseau, il fallait étudier leur impact sur les performances et la disponibilité du réseau. Pour se faire nous avons modélisé un réseau IoT sur l'environnement de simulation NS2 ; lequel simulateur est largement utilisé pour la simulation et l'évaluation de performances des réseaux filaires et non filaires.

Nous avons défini des indicateurs de performances qui nous ont permis d'apprécier et d'identifier l'impact réel des nœuds malveillants sur le réseau IoT. Grâce à l'utilisation de certains outils développés du simulateur NS2, notamment l'outil Gnuplot, nous avons pu constater que la présence de nœuds malveillants sur le réseau a une conséquence très négative sur l'énergie.

En effet, un nœud malveillant au sein du réseau dégrade les performances du réseau en termes :

- De diminution de l'énergie des nœuds : suite à la perte de paquets, les nœuds qui contribuent aux différents routages consomment toute leur énergie à renvoyer les paquets.
- Dégradation du débit : les données ne sont pas correctement transférées.
- Augmentation du délai : les données mettent plus de temps à atteindre le destinataire.
- Instabilité de la gigue : l'arrivée des données varient dans le temps causant une instabilité.

Il s'agissait par la suite de proposer des solutions permettant la sécurisation de notre réseau contre les nœuds malveillants, nous avons implémenté deux (02) solutions de sécurité. La première solution repose sur la mise en place d'un modèle de confiance entre les nœuds du réseau. La seconde consiste à réorganiser le réseau IoT en cluster afin d'optimiser la consommation d'énergie.

Notre modèle de confiance repose sur les mécanismes de sécurité suivants :

- L'activation du mode promiscuité qui nous a permis l'écoute de tout le trafic réseau.
- La création du protocole TAODV, un clonage du protocole AODV en plus de l'implémentation de l'aspect « Trust ». Ce dernier inclue le mode promiscuité, le maintien des liens de voisinage, le suivi du routage des paquets.
- L'implémentation de notre modèle de confiance qui se base sur TAODV comme protocole de routage. Grâce au maintien de deux listes pour le transfert et la réception des paquets, une première détection des nœuds malveillants est faite. S'en suit une vérification si le nœud est vraiment malsain ou juste défectueux. Les tables de confiance nous informent de la valeur de confiance attribuée par chaque nœud à tous ses voisins. Cette valeur augmente pour chaque bon comportement et diminue pour chaque mauvais comportement. Ainsi un nœud malveillant est détecté lorsqu'il est présent.

Afin de rajouter une couche de protection aux nœuds et d'augmenter leur disponibilité, nous avons procédé à une réorganisation de notre réseau. Pour cela, des clusters ont été créés avec un Cluster-Head à chaque tête. Ceci nous a permis de diminuer les routages de longues distances, d'éviter d'inclure plusieurs nœuds intermédiaires et de ce fait optimiser la consommation d'énergie globale du réseau.

Notre travail nous a permis d'aboutir à des résultats concrets que nous estimons assujettis à des améliorations. Il serait prometteur dans un premier temps d'utiliser le résultat du modèle de confiance pour la détection et la suppression des nœuds malveillants. Dans un second temps, de faire en sorte que les CHs soient aussi des nœuds mobiles et de procéder à une sélection dynamique de ces derniers selon l'énergie restante des nœuds du réseau et le degré de confiance que ses voisins lui accordent.

Bibliographie

[1] NARANG Sahil, NALWA Tarun, CHOUDHURY Tanupriya, KACHYAP Nirbhay "An efficient method for security measurement in internet of things" (2018), p. 319-323

[2] FEI, Hu. Internet of Things (IoT) as Interconnection of Threats (IoT). In : *Security and Privacy in Internet of Things (IoTs).* New York : Edition Fei Hu, 2016, p.25-26-27.

[3] TURUK Ashok Kumar, BYSANI Leela Krishna. "A Survey On Selective Forwarding Attack in Wireless Sensor Networks" (2011)

[4] MINOLI Daniel, SOHRABY Kazem, KOUNS Jacob, " IoT Security (IoTSec) Considerations, Requirements, and Architectures" (2017)

[5] KAUR Karandeep, "A Survey on Internet of Things – Architecture, Applications, and Future Trends" (2018)

[6] HOSSAIN Md. Mahmud, FOTOUHI Maziar, HASAN Ragib "Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things" (2015), p. 21-28

[7] ASSIRI Abeer, ALMAGWASHI Haya "IoT Security and Privacy Issues" (2018).

[8] ALABA Fadele Ayotunde, OTHMAN Mazliza, HASHEM Ibrahim Abaker Targio, ALOTAIBI Faiz "Security Attacks in IoT: A Survey", p. 1-35

[9] MELLO Ruan de A. C, RIBEIRO Admilson de R. L, MORENO Fernando M. de Almeida, Edward D. " An Architecture for Self-protection in Internet of Things." (2016), p. 40-46

[10] KOBRA Abdelaziz Amara, "*Détection d'Intrusion et Sécurisation du Routage dans les Réseaux Ad hoc*". Thèse de doctorat. Université Badji mokhtar Annaba. (2016). Disponible à l'adresse :

http://biblio.univ-annaba.dz/wp-content/uploads/2016/11/These-Abdelaziz-Amara-Korba.pdf

[11] HELLAOUI Hamed, BOUABDALLAH Abdelmadjid, KOUDIL Mouloud, "TAS-IoT: Trust-based Adaptive Security in the IoT" (2016), p. 599-602

[12] GUPTA KrishnaKanth, SHUKLA Sapna, "Internet of Things: Security Challenges for Next Generation Networks" (2016), p. 315-318

[13] ANIJI John, RAJPUT Anagha, BABU Vinoth, "Dynamic Cluster Head Selection in Wireless Sensor Network for Internet of Things Applications" (2017), p. 45-48

[14] BEHERA Trupti Mayee, MOHAPATRA Sushanta Kumar, SAMAL Umesh Chandra, S.KHAN Mohammad, DANEDHMAND Mahmoud, H.GANDOMI Amir, "Residual Energy Based Cluster-head Selection in WSNs for IoT Application" (2019)

[15] LABENI Yacine, BOULKOUR abdelouahid, "Outil pour l'analyse des traces de simulation des protocoles de routage ad hoc avec NS-2". Mémoire de master. Université Mohamed Sadik Jijel. (2016). Disponible à l'adresse :

http://chettibi.e-monsite.com/medias/files/memoire-1-.pdf

Annexe A

Installation de NS-2 sous Ubuntu

NS-2 est conçu pour fonctionner sous les systèmes d'exploitation Unix et Linux. Il est livré sous forme de paquetage regroupant tous les fichiers nécessaires pour un bon fonctionnement. Dans ce travail, nous utiliserons la version 2.35.

A.1 Prérequis d'installation de ns2

Avant de procéder à l'installation de ns2, nous allons d'abord effectuer les mises à jours et les téléchargements nécessaires pour une parfaite installation de ce dernier.

Sous ubuntu, nous allons ouvrir un terminal et exécuter les commandes suivantes en mode super utilisateur :

- ✓ Sudo apt-get update
- ✓ Sudo apt-get install gcc
- ✓ Sudo apt-get install build-essential autoconf automake
- ✓ Sudo apt-get install tcl8.5-dev tk8.5-dev
- ✓ Sudo apt-get install perl xgraph libxt-dev libxll-dev libxmu-dev

A.2 Installation de ns2

Une fois les commandes précédentes réalisées, nous allons procéder à l'installation de ns2 en srespectant les étapes suivantes :

- ✓ Télécharger NS2 (version 2.35) à partir du lien suivant :

https://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/download?use_mirror=vorboss

- ✓ Extraire le fichier ***ns-allinone-2.35.tar.gz*** grâce à la commande : ***sudo tar –zxvf ns-allinone-2.35.tar.gz***
- ✓ Modifier le fichier ***opt/ns-allinone-2.35/ns-2.35/linkstate/ls.h*** en ajoutant *this->* à la ligne 137 et avoir le résultat suivant, puis sauvegarder la modification.
  void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
- ✓ Taper la commande ***cd ns-allinone-2.35*** puis ***sudo /install*** afin de lancer l'installation de ns2.
- ✓ S'il n y a aucune erreur, à la fin de l'installation, un message s'affichera indiquant que l'installation s'est bien déroulée. Il y aura aussi un message demandant la modification de variables d'environnements.
- ✓ En utilisant un éditeur de texte, modifier les variables d'environnements en tappant la commande : ***sudo gedit ~ /.bashrc*** et ajouter les lignes suivante à la fin du fichier :

export PATH=$PATH:/home/[nom de l'utilisateur]/ns -allinone-2.35/bin:/home/[nom de l'utilisateur]/nsallinone-2.35/tcl8.5.10/unix:/home/[nom de l'utilisateur]/ns-allinone-2.35/tk8.5.10/unix export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/[nom de l'utilisateur]/ns -allinone-2.35/otcl-1.14:/home/[nom de l'utilisateur]/ns-allinone-2.35/lib export TCL_LIBRARY=$TCL_LIBRARY:/home//[nom de l'utilisateur]/ns-allinone-2.35/tcl8.5.10/library export DISPLAY=:1.0

A.3 Validation de l'installation

- ✓ Au niveau de repertoire : ns-allinone-2.35/ns-2.35, taper la commande : *./validate*

  Cette commande va prendre beaucoup de temps afin de vérifier que tout est bien installé, à la fin, un message du succès de l'opération va être affiché et ns2 est désormais prêt à être utilisé.

Annexe B

Activation du mode promiscuité sous NS-2

Ce mode va permettre à chaque nœud d'écouter le trafic de tous ses nœuds voisins une fois activé. Pour ce faire, nous allons suivre les étapes suivantes :

-Aller au répertoire ***ns-allinone-2.35/ns-2.35/taodv/taodv.h*** et ajouter les lignes suivantes :

- ✓ *#include<mac.h>* à la ligne 46
- ✓ class TAODV : *public Tap* , public agent à la ligne 211
- ✓ public :
  *void tap (const packet *p) ;* à la ligne 233
- ✓ protected :
  *Mac *mac_ ;* à la ligne 243

-Aller au répertoire ***ns-allinone-2.35/ns-2.35/taodv/taodv.cc*** et ajouter les lignes suivantes :

- ✓ *else if(strcmp(argv[1], ''install-tap'')==0) {*

  mac_ = (Mac) TclObject::lookup(argv[2]);*

  if(mac_==0) return TCL_ERROR;

  mac_->installTap(this);

  return TCL_OK;

  } à la ligne 143

- ✓ *void*
  *TAODV :: tap(const Packet *p) {*
  printf(" I am node %d and I received packet fron a neighbor node\n" , index);
  } à la ligne 156

-Aller au répertoire ***ns-allinone-2.35/ns-2.35/tcl/lib/ns-mobilenode.tcl*** et ajouter les lignes suivantes :

- ✓ $self instvar dmux_ imep_ toraDebug_ *mac_* à la ligne 170
- ✓ *$agent install_tap $mac_(0)* à la ligne 204

Une fois tous ces ajouts effectués, il ne reste qu'à recompiler NS-2 afin d'enregistrer les modifications faites et ce en utilisant les commandes suivantes :

- ✓ ./configure
- ✓ Make
- ✓ sudo make install

Annexe C

Installation de « eclipse » sous Ubuntu

Eclipse est un projet organisé en un sous-ensemble de projets de développement logiciels. Il inclut le projet CDT (C/C++ Development Tools project) qui est un outil de développement pour les langages C et C++. La dernière version d' « eclipse » : Eclipse Oxygen intègre un éditeur de texte utilisant un Langage Server Potocol.

C.1 Prérequis d'installation

- ✓ Avoir une machine virtuelle avec Java installé et fonctionnel.
- ✓ Bénéficier d'un compilateur ou un interpréteur C/C++.

Sous ubuntu, nous allons ouvrir un terminal et exécuter les commandes suivantes en mode super utilisateur :

- ✓ apt get-install openjdk-8-jdk
- ✓ chmod –R 777

C.2 Installation de « eclipse »

L'installation d' « eclipses » sous environnement Ubuntu passe par les étapes suivantes :

- ✓ Télcharger « eclipses » à partir du lien suivant : https://eclipse.org/downloads/
- ✓ Extraire le fichier ***eclipse-inst-linux64.tar.gz*** grâce à la commande : ***sudo tar –zxvf eclipse-oxygen-linux64.tar.gz***
- ✓ Dans le répertoire ***eclipse***, lancer l'installation grâce à la commande : ./eclipse
- ✓ Une fenêtre *Eclipse Launcher* s'affichera sur l'écran, elle nous permettra de sélectionner le chemin du répertoire ***ns-allinone-2.35***
- ✓ En cliquant sur ''*Launch*'', la fenêtre de la **FIGURE C.1** suivante va apparaître.

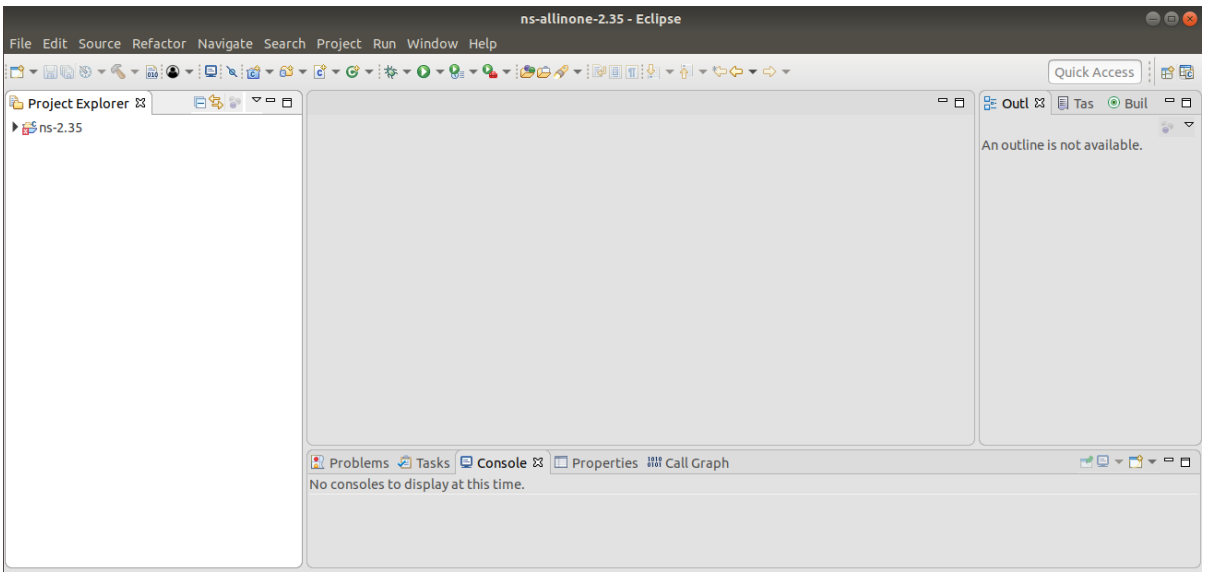

FIGURE C.1 LANCEMENT D'ECLIPSE OXYGEN

On peut apercevoir quatre grandes panneaux sur la figure précédente : en extrême droite un panneau projet, en bas un panneau console affichant les erreurs, en extrême droite un panneau outline et au milieu le panneau principale.

Pour ajouter un projet, il faut cliquer sur : new project / C++ project.