République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Tébessi - Tébessa
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département : Mathématiques et Informatique

Mémoire de fin d'étude
Pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Réseaux Et Sécurité Informatique

Thème

Une plateforme de simulation sous omnet++ pour diffuser les informations sur le trafic dans les architectures VANETs Cloud

Présenté Par :
Ghozlane Abdelhak

Devant le jury :

Mr T. Mekhaznia	*MAA*	*Univérsité Larbi Tébessa*	*Président*
Mr S. Dhifallah	*MAA*	*Univérsité Larbi Tébessa*	*Examinateur*
Mr A. Sahraoui	*MCB*	*Univérsité Larbi Tébessa*	*Encadreur*
Mr M. Derdour	*MCA*	*Univérsité Larbi Tébessa*	*Co-Encadreur*

Une Plateforme De Simulation Sous OMNET++ Pour Diffuser Les Informations Sur Le Trafic Dans Les Architectures VANETs

Ghozlane Abdelhak

Université de Larbi Tébessi

Résumé

Les réseaux véhiculaires (VANETs) sont devenus un domaine de recherche crucial, grâce à un changement technologique, au cours des dernières années. Ces réseaux composent généralement via des entités fixes (RSUs) et des entités mobiles (véhicules). L'objectif globale est d'assurer un niveau fiable de sécurité routière et le confort lors le déplacement. Avec l'évolution des capacités d'internet, les véhicules sont devenus plus intelligent et une nouvelle gamme d'applications moderne ont apparus comme auto-conduite, le stationnement intelligent et les services Cloud pour la gestion du trafic.

Étant donné les évaluations de ces applications sur le terrain pouvant être coûteux comme elles risquant d'aggraver des conditions de circulation et le rendre dangereuses. Donc, il ne s'agit souvent pas d'une expérimentation réelle. Les chercheurs en ce domaine cherchent évaluer les applications véhiculaires moderne en développant des plateformes de simulation. Avec ces plateformes, les concepts des simulations et du trafic réseau doivent être combinés afin de mettre en places des stratégies permettant d'évaluer les coûts associés à l'infrastructure et la congestion.

Parmi les applications qui ont attiré beaucoup d'intérêt des chercheurs ces dernières années, ceux qui échangent des informations en vue d'accroître la sécurité routière, atteindre plus de fiabilité en diffusant des informations sur le réseau. Et parmi les problèmes se posent dans les VANETs est de fournir des outils et de simulations fiables. Étant donné que ce type de simulation n'était pas encore défini.

Dans ce travail, nous avons développé une plateforme de simulation pour les applications VANETs Cloud moderne permettent de simuler les scénarios de dissémination des informations sur le trafic.

Mots Clés : Cloud Computing, VANET, communication V2V, communication V2I, sécurité routière, dissémination des informations.

Abstract

Vehicular networks (VANETs) have become a crucial area of research, thanks to technological change in recent years. These networks typically consist of fixed entities (RSUs) and mobile entities (vehicles). The overall goal is to ensure a reliable level of road safety and comfort when traveling. With the evolution of internet capabilities, vehicles have become smarter and a new range of modern applications have emerged as self-driving, intelligent parking and cloud services for traffic management.

Given the evaluations of these applications in the field can be expensive as they may aggravate traffic conditions and make it dangerous. So, it is often not a real experiment. Researchers in this field seek to evaluate modern vehicular applications by developing simulation platforms. With these platforms, the concepts of simulations and network traffic need to be combined to put in place strategies for assessing the costs associated with infrastructure and congestion.

Among the applications that have attracted a lot of interest from researchers in recent years, those who exchange information to increase road safety, achieve more reliability by broadcasting information on the network. And among the problems arise in VANETs is to provide reliable tools and simulations. Since this type of simulation was not yet defined.

In this work, we have developed a simulation platform for modern VANETs Cloud applications to simulate the dissemination scenarios of traffic information.

Keywords : Cloud Computing, Réseau Véhiculaires, V2V Communication, V2I Communication, Data Dissemination.

ملخص

أصبحت شبكات المركبات (VANETs) مجالًا مهمًا للبحث، وذلك بفضل التغير التكنولوجي في السنوات الأخيرة. تتكون هذه الشبكات عادةً من كيانات ثابتة وكيانات متنقلة (مركبات). الهدف العام هو ضمان مستوى موثوق من السلامة على الطرق والراحة عند السفر. مع تطور إمكانات الإنترنت، أصبحت السيارات أكثر ذكاءً وظهرت مجموعة جديدة من التطبيقات الحديثة كقيادة ذاتية ووقوف ذكي وخدمات سحابية لإدارة حركة المرور.

بالنظر إلى تطبيق هذه التطبيقات في الواقع، فقد تكون باهظة الثمن لأنها قد تؤدي إلى تفاقم ظروف حركة المرور وجعلها خطيرة. لذلك، في كثير من الأحيان لا يمكن اجراء تجربة حقيقية. يسعى الباحثون في هذا المجال إلى تقييم تطبيقات المركبات الحديثة من خلال تطوير منصات المحاكاة. مع هذه المنصات، يجب دمج مفاهيم المحاكاة وحركة مرور الشبكة لوضع استراتيجيات لتقييم التكاليف المرتبطة بالبنية التحتية والازدحام.

من بين التطبيقات التي جذبت الكثير من الاهتمام من الباحثين في السنوات الأخيرة، التطبيقات التي تسمح بتبادل المعلومات لزيادة السلامة على الطرق. ومن بين المشاكل التي تنشأ في VANETs هو توفير أدوات موثوقة والمحاكاة. منذ هذا النوع من المحاكاة ليس معروفا بشكل جيد بعد.

في هذا العمل، قمنا بتطوير منصة محاكاة لتطبيقات VANETs Cloud الحديثة لمحاكاة سيناريوهات نشر معلومات المرور.

كلمات البحث: الحوسبة السحابية، الشبكة الخاصة بالمركبات، والاتصالات ، V2Vوالاتصالات ، V2I و السلامة على الطرق، نقل البيانات .

Dédicace

A mon père, A ma Mere

A Mes frères, A Mes Sœurs

A Mon Encadreur Dr. Sahraoui Abdellatif

Remerciement

Tout d'abord, louange à Dieu qui, avec sa grâce, fait de bonnes œuvres.

Mes remerciements vont tout premièrement à Dieu tout puissant pour la volonté, la santé et la patience qui m'a donné durant la période de travail.

A mon cher Père

À mon cher père, Dieu vous a sauvé et fait de vous un atout et une fierté pour nous. Rien ne peut suffire pour vous. Merci mon Père.

A ma très chère maman

Aucun acte ou expression ne pourra exprimer mes sentiments envers vous, que dieux vous garde pour nous maman. Merci Maman

A mes frères, Toufik, Med elhadi, hussein, gassou.

A ma famille Ghoul, Vous êtes Des Parents et Des frères, Un mot de remerciement ne suffit pas. Et ne peut pas être suffit. Merci.

Je voudrai exprimer mes sincères remerciements à mon encadreur, le Dr. Sahraoui Abdellatif, pour avoir acceptée de diriger ma thèse. Au cours de ma thèse, Dr. Sahraoui Abdellatif m'a orienté en proposant de nombreuses idées et perspectives. Grâce à son pensé logique dans ce domaine, mes idées et mes propositions initiales ont été améliorés et développés, Merci. Tu étais pas un frère seulement un encadreur.

A tous qui j'aime……

A tous qui m'aiment ……

TABLEAU DES MATIERES

Résumé **1**

Abstract **2**

Dédicace **4**

Remerciement **5**

Introduction Générale **11**

CHAPITRE 01 Introduction Au VANET Cloud Page 12

1.1. Introduction 14

1.2. Les réseaux véhiculaire VANET 14

1.2.1. Définition d'un Réseau VANET 14

1.2.2. Les composants d'un réseaux véhiculaires 15

1.2.3. Caractéristiques de VANET 16

1.2.4. Les Applications offerts par les réseaux VANETs 17

1.2.4.1. Services liés aux gestions de trafic routier 17

1.2.4.2. Les services liés à la sécurité routière 18

1.2.4.3. Services liés au confort 19

1.2.5. Les architectures de communication 20

1.2.5.1. Communications de Véhicule à Véhicule (V2V) 20

1.2.5.2. Communications de Véhicule à Infrastructure (V2I) 20

1.2.5.3. Communication hybride 20

.1.2.6 Les techniques de communications 21

.1.2.6.1 Les communications radio 21

1.2.6.2. Les communications sans fil 22

1.3. Introduction à la dissémination de messages 22

1.3.1. La dissémination 22

1.3.2. Types de message 22

1.3.3. Type de dissémination 23

1.3.4. Stratégies de dissémination 23

.1.4 La convergence des applications VANET vers les applications VANET Cloud 25

1.4.1. Cloud Computing 25

1.4.2. Combinaison de VANET et du Cloud Computing 25

1.4.3. Utiliser les Clouds dans les VANETs 25

1.4.4. Les architectures VANET Cloud 26
1.4.4.1. Vehicular Clouds (VC) 26
1.4.4.2. VANET using Clouds (VuC) 27
1.4.4.3. Hybrid Clouds 28
1.4.5. Les Applications VANET Cloud 28
1.4.5.1. Remote Configuration and Car Performance Checking 28
1.4.5.2. Big traffic data analysis 29
1.4.5.3. Smart location-based advertisements 29
1.4.5.4. Vehicle Witnesses 29
1.5. Conclusion 29

CHAPITRE 02 Les Modèles De Simulation Et Ees Techniques De Dissémination Des Informations Pour les VANETs Cloud Page 31

2.1. Introduction 32
2.2. Les architectures cloud véhiculaire pour la dissémination des informations 32
2.2.1 Les architectures VANET Cloud 33
2.2.2 Les architectures hétérogènes 33
2.2.3 Les architectures véhiculaires sociales 34
2.3. Les données de dissémination d'information 35
2.3.1 Les données Uplink 35
2.3.2 Les données downlink 35
2.4. Les techniques de dissémination d'information pour les Vanet Cloud 35
2.4.1 Le routage 35
2.4.1.1 Types de routage 36
2.4.1.2 Stratégies de routage 36
2.4.2 La migration de la Virtual machine (VM) 38
2.4.2.1 Scénarios de migration VM 38
2.4.3 Clustering 39
2.4.3.1 Procédure de formation du cluster 40
2.4.3.2 Procédure de sélection du Tête de Groupe (Cluster Head) 41
2.4.3.3 Procédure de maintenance du cluster 42
2.5. Les Environnements de Simulation de VANET Cloud 43
2.5.1 Le Simulateur OMNET++ 43
2.5.1.1 Les Plateformes de simulation 43
2.5.2 Le Simulateur NS-2 45
2.5.3 Le simulateur NS-3 46
2.5.3.1 Comparaison entre OMNET++ et NS-2 47
2.5.4 Les Simulateurs de mobilité et de trafic routier 48

2.6. Conclusion 50

CHAPITRE 03 Une Plateforme Vanet-Cloud Pour la Dissémination Des Informations Page 51

3.1 Introduction 52
3.2 Environnement de simulation 53
3.3 Travail proposé 53
3.3.1 Les problèmes de dissémination des informations dans les VANETs 53
3.3.2 L'intégration du Cluster dans les réseaux VANETs Cloud 54
3.3.3 Les applications mobiles adaptatives 54
3.4 Le scenario d'étude 55
3.5 La vue conceptuelle de l'intégration 56
3.5.1 ICanCloud 58
3.5.2 INET 61
3.5.3 Notre Contribution 64
3.5.4 Les Algorithmes 66
3.5.4.1 Dominating Set Algortithm 66
3.5.4.2 Intermittent Sets Algorithm 67
3.6 Conclusion 68

CHAPITRE 04 Implémentation et Simulation Page 69

4.1 Introduction 70
4.2 La Simulation 70
4.2.1 Le projet Vehicular Clouds 70
4.2.2 Data Dissemination in VANET Cloud (DDVC) 70
4.2.3 Construire et exécuter des simulations 71
4.3 L'implémentation 74
4.3.1 Dominating Sets algorithms 74
4.3.2 Intermittent Network Algorithmes 76
4.3.3 Cluster Controller 77
4.4 Un scenario de simulation 79
4.5 La Simulation 80
4.5.1 Lancer le programme de simulation 80
4.6 Conclusion 84

Conclusion Générale 86

Bibliographie 88

LISTE DE FIGURES

Page

Figure 1-1 Présentation de réseau Vanet 14
Figure 1-2 Déférent élément constituant le véhicule 15
Figure 1-3 Exemple de réseau VANET 16
Figure 1-4 Types de communication dans les VANETs [7] 21
Figure 1-5 Vehicular Clouds [16] 27
Figure 1-6 VANET using Clouds [16] 27
Figure 1-7 Hybrid Clouds [16] 28

Figure 2-1 Routage dans les VANETs [38] 35
Figure 2-2 Scénarios de migration de machine virtuelle [45] 39
Figure 2-3 Architecture de base de NS-2 [19] 46
Figure 2-4 Architecture en couches du simulateur NS-3 [23] 47

Figure 3- 1 L'architecture du cluster 56
Figure 3- 2 Diagramme de Classe du Modèle 57
Figure 3- 3 Une Machine Vertuelle avec ces parametre 58
Figure 3- 4 Un Module de calcul Avec ces paramètres 59
Figure 3- 5 Un Module de stockage avec ces paramètres 59
Figure 3- 6 Une Mémoire avec ces paramètres 60
Figure 3- 7 Un Systeme d'Exploitation avec ces paramètres 60
Figure 3- 8 L'hiperviseur 61
Figure 3- 9 Channel Control avec ces paramètres 62
Figure 3- 10 Scenario Manager avec ces paramètres 62
Figure 3- 11 Configurateur avec ces paramètres 63
Figure 3- 12 Interface mobilité avec ces paramètres 63
Figure 3- 13 Table de routage avec ces paramètres 64
Figure 3- 14 un véhicule avec ces paramètres 64
Figure 3- 15 le module qui gère le cluster avec ces paramètres 65

Liste De Figures

Figure 4- 1 Démarage d'un projet 70
Figure 4- 2 Le Dossier Simulation 70
Figure 4- 3 les dossier Src 70
Figure 4- 4 le fichier Omnetpp.ini 71
Figure 4- 5 Fichier. NED 72
Figure 4- 6 fichier .h/.c++ 72
Figure 4- 7 L'interface Source et Design 73
Figure 4- 8 le fichier omnetpp.ini 73
Figure 4- 9 Palette OMNET++ 79
Figure 4- 10 La topologie de notre projet 79
Figure 4- 11 Un noeud véhiculaire 80
Figure 4- 12 Lancer une simulation 80
Figure 4- 13 Lancement de simulation 81
Figure 4- 14 L'échangement de message entre les nœuds 82
Figure 4- 15 Colorisation des nœuds 82
Figure 4- 16 Le module Cluster Controller 83
Figure 4- 17 Résultat 2 84
Figure 4- 18 Résultat 1 84

Introduction Générale

Ces dernières années, l'industrie automobile a connu un grand développement et les véhicules ne sont plus considérés comme de simples systèmes mécaniques contrôlés par certains composants électroniques. Les véhicules ont été rendus "*intelligents*" en ajoutant de nouvelles fonctionnalités pour comprendre leur environnement à travers des capteurs, pour communiquer avec d'autres véhicules via des interfaces radio sans fil ou 3G. L'objectif était de traiter ces informations et de prendre des décisions en temps réel concernant le comportement du véhicule grâce à un ordinateur de bord. Les interfaces radio permettent aux véhicules de communiquer entre eux via la communication entre véhicules (V2V) ou avec les unités situées au bord de la route via les communications V2I (entre véhicules et infrastructures).

Parmi les applications qui ont attiré beaucoup d'intérêt des chercheurs ces dernières années, ceux qui échangent des informations en vue d'accroître la sécurité routière, atteindre plus de fiabilité en diffusant des informations sur le réseau. Ces informations concernent les conditions de congestion, les événements, les accidents, les informations météorologiques. etc. D'autre part, fournir un ensemble de fonctions pour assurer un niveau de confort lors le déplacement, à savoir, accès Internet, paiement en ligne, jeux, etc. Pour pouvoir prendre en charge de telles applications, un réseau entre ces véhicules doit être créé. Dans lequel les intérêts majeurs de ces applications, l'amélioration de la sécurité routière, la diffusion d'informations, l'évitement du trafic. Pour atteindre un haut niveau de sécurité, ces réseaux sont devenus un domaine de recherche actif et caractériser par l'apparition de normes de communication et de travaux de développement ne cessent de s'augmenter.

Jour après jour, des réseaux de véhicules ad hoc se développent et peuvent fournir un système fiable et sécurisé pour un contrôle efficace du trafic. Compte tenu de la nature des médias en communication, le routage, les approches de communication et de la courte durée des sessions de communication entre véhicules, l'amélioration de VANET pour les besoins modernes devient cruciale. Parmi les caractéristiques du réseau ad hoc, il y a le s'auto-organisation des nœuds pour créer un réseau sans infrastructure. Aujourd'hui, le Cloud Computing devrait être la prochaine génération des réseaux en raison de son évolutivité, ces services (i.e., PaaS, IaaS, SaaS) et d'autres fonctionnalités importantes. Des architectures de Cloud ad hoc sont proposées pour combiner le

concept de véhicule et de le paradigmes Cloud Computing afin de fournir un meilleur contrôle du trafic et une sécurité sur la route avec les avantages du Cloud Véhiculaire.

Les chercheurs ne peuvent pas réaliser les expériences de centaines véhicules mobiles situés sur de grandes surfaces. Un autre problème clé se pose dans les VANET est de fournir des niveaux d'évaluation et de simulation fiables. Etant donné que ce type de simulation n'était pas encore défini, nous allons donc proposer de travailler à la réalisation d'un module sous le simulateur OMNET ++ pour la diffusion des informations de trafic dans un environnement de Cloud Véhiculaire (CV).

L'objet clé de ce mémoire est de développer une plateforme de simulation pour les applications VANETs Cloud moderne permettent de simuler les scénarios de dissémination des informations du trafic.

La structure générale de notre travail est organisée en quatre chapitres comme suit :

- Dans le premier chapitre, nous allons présenter les réseaux véhiculaires, la convergence de VANET vers les réseaux VANETs Cloud. Ainsi, nous allons introduire le concept de dissémination des informations.
- Le deuxième chapitre est consacré par les différentes approches et les techniques de dissémination des informations dans un environnement VANET Cloud. Ensuite, étude de performances entre les simulateurs réseaux et mobiles.
- Le troisième chapitre, une vue conceptuelle de notre modèle de simulation sera proposée et un scenario d'étude avec l'implémentation des déférents algorithmes.
- Dans le quatrième chapitre, une extraction du résultat de simulation est mise en évidence. Ainsi que les déférents étape de simulation.

CHAPITRE 01
INTROUDCTION AU VANET CLOUD

1. Les Réseaux Véhiculaire VANET
2. Introduction A La Dissémination De Messages
3. La Convergence Des Applications VANET Vers Les Applications VANET Cloud

1.1. Introduction

Le but de ce chapitre est d'apprendre le concept de réseau de véhicules et le probléme de dissémination d'informations dans ces réseaux. De manière générale, nous présentons d'abord les concepts de base d'un réseau de véhicules en mentionnant leur définition, ces composants, ces caractéristiques, les architectures et les applications offertes par les réseaux de véhicules. Ensuite, nous présentons la convergence des applications VANET vers les applications VANET Clouds.

1.2. Les réseaux véhiculaire VANET

1.2.1. Définition d'un Réseau VANET

Un réseau VANET (réseau ad hoc) est un réseau de communication entre véhicules intelligents équipés d'ordinateurs, de périphériques réseau et de différents types de capteurs. Les réseaux VANET font partie de la famille de réseaux MANET (réseau mobile ad-hoc) qui fonctionnent dans des réseaux point à point sans infrastructure, c'est-à-dire que tout nœud constituant le réseau est un point d'accès. Dans un réseau VANET, les nœuds sont les véhicules intelligents appartenant au réseau. [1]

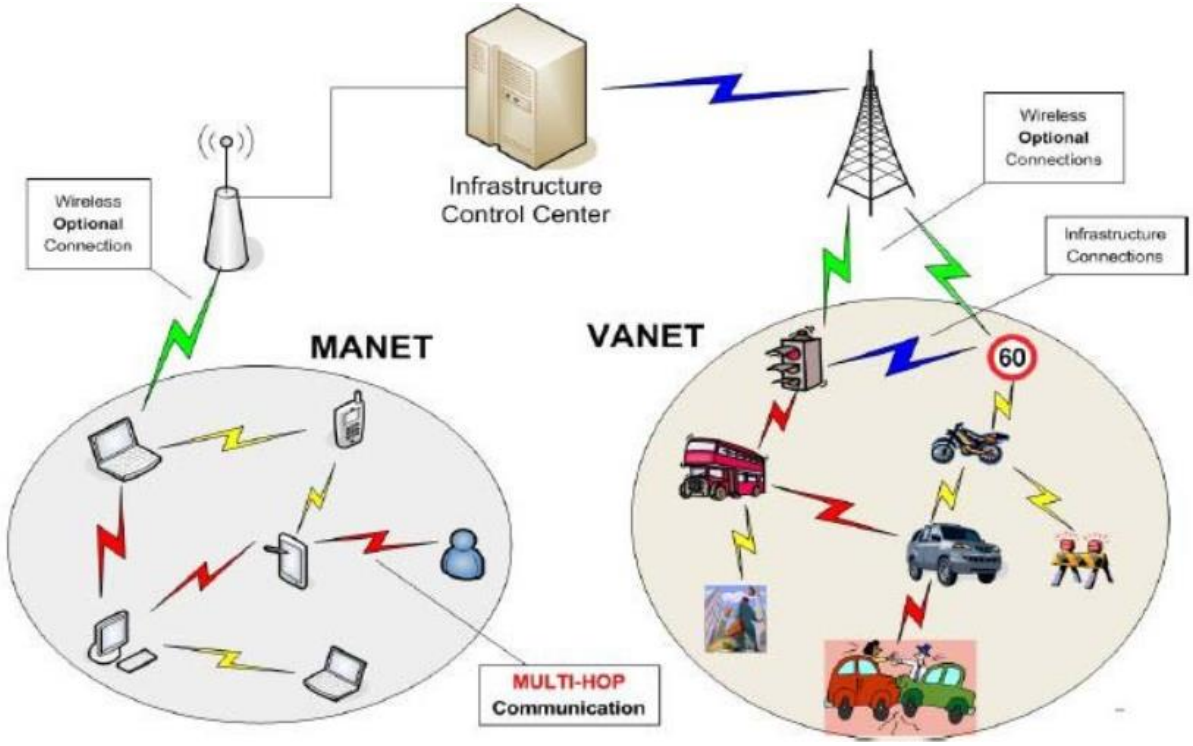

Figure 1-1 Présentation de réseau Vanet

1.2.2. Les composants d'un réseaux véhiculaires

Les services fournis dans les réseaux VANET permettent la communication entre les différents composants du réseau :

RSU (Road Side Unit)

Les RSU sont des infrastructures installées au bord de la route, principalement des feux de signalisation, des lampadaires, elles assurent la transmission, la réception et le traitement de l'information. L'objectif principal de ces RSU est la gestion du trafic et des la mobilité des véhicules. Ainsi que des points d'accès au réseau et toutes les informations sur le réseau.

OBU (On-Board Unit)

Les OBU sont des périphériques logiciels ou matériels de haute technologie impliqués dans des véhicules partageant un réseau de véhicules, tels que (GPS, radar, caméras, capteurs, etc.). Ils ont pour but d'assurer la localisation, l'envoi, la réception, le calcul et le stockage de données déférentes. Les OBU ont des interfaces de communication avec d'autres véhicules et d'autres pour la communication avec l'infrastructure (RSU)

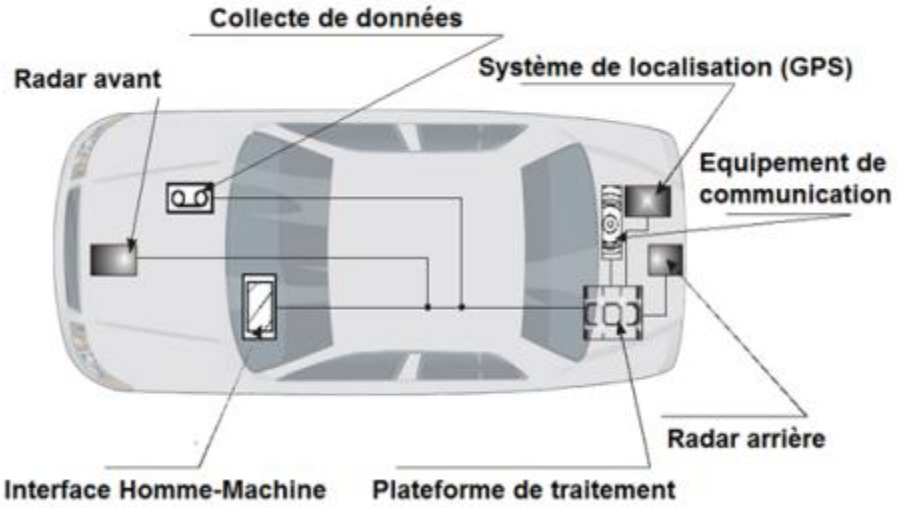

Figure 1-2 Déférent élément constituant le véhicule

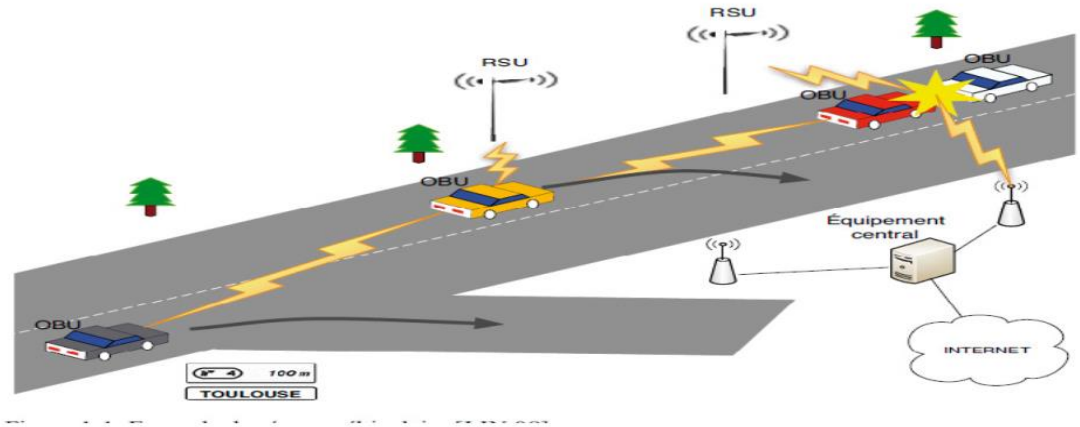

Figure 1-3 Exemple de réseau VANET

1.2.3. Caractéristiques de VANET

Les VANETs font partie de MANET (Mobile AdHoc Network) mais avec quelques options spécifiques tels que : [2]

Capacité de traitement, d'énergie et de communication

Le problème rencontrer dans les MANET est le problème énergétique en raison de la capacité limite en termes d'énergie, généralement c'est la batterie. Contrairement aux VANETS, les nœuds n'ont pas le problème d'énergie, car il utilise des hydrocarbures, ainsi que les capacités des unités de traitement (Wifi, Bluetooth, Caméra, etc.) sont élevés comparés à ceux des autres types de MANET et ce sont des unités de traitement spécialisées (capteur de pollution, capteur de pluie, radar, GPS ...)

Forte mobilité et topologie du réseau

La mobilité est un facteur majeur qui différencie les réseaux VANET à des autres réseaux MANET. La vitesse des nœuds est différente de celle des environnements, de sorte que la topologie du réseau change immédiatement. Par exemple sur une autoroute, la vitesse de la voiture dépasse 100 km / h, ce qui a un effet important sur la durée et les performances des communications suite au changement rapide de la topologie du réseau.

Collecte d'informations

Les différents capteurs (caméra, capteurs de pollution, capteur d'état de la route, radar, etc.) dans la voiture permettent de collecter les différents types d'informations de trafic permettant

au conducteur de visualiser les modifications de son environnement à proximité de manière adéquate.

L'environnement de déplacement et modèle de mobilité

Dans les réseaux MANET, les nœuds se déplacent de manière aléatoire, mais dans les VANET, le déplacement des véhicules dépend de l'architecture de la route (rond-point, autoroute, double voie, etc.). En outre, un véhicule peut quitter un environnement urbain caractérisés par des obstacles au transfert de données et de signaux vers un autre environnement périurbain ou autoroutier. Les VANET disposent de modèles de mobilité, dont le plus évident est l'importance de réduire le temps de réponse entre les véhicules.

1.2.4. Les Applications offerts par les réseaux VANETs

1.2.4.1. Services liés aux gestions de trafic routier

Les applications liées à la gestion du trafic routier consistent à fournir aux conducteurs des informations leur permettant d'adapter leur itinéraire à la situation du trafic. Par exemple, la programmation des feux de circulation et de la surveillance du trafic, la prévention des sorties de voie en ligne ou dans les virages, la conduite coopérative. L'efficacité du trafic coopératif comprend deux types d'applications [2].
La gestion de la vitesse coopérative (CSM) comprend deux services.

Les applications de la gestion de vitesse coopérative (CSM)

- **CSM de La notification de la vitesse limite** : Elle est responsable de fourni au conducteur des notifications concernant la vitesse limite, par exemple la limite de vitesse règlementaire actuelle, la vitesse recommandée contextuels.
- **CSM de vitesse optimale de feux de circulation consultative :** Elle permet d'adapter la vitesse optimale au niveau des feux de signalisation. Pour cela, une station de l'infrastructure fournit des informations sur les phases actuelles des feux de signalisation, le temps restant avant le changement de phase et la durée de chacune d'elles. [3].

Les applications de la navigation coopérative (CoNa)

Elle fournit des informations sur le meilleur chemin et aide à la navigation.

1.2.4.2. Les services liés à la sécurité routière

Chaque année, le nombre d'accidents augmente, car la sécurité routière est une priorité absolue. Pour minimiser le nombre d'accidents, la communication entre les véhicules donne la possibilité d'informer les conducteurs qu'il y a un problème sur la route (verglas, pluie, neige, précurseurs graves, travaux, etc.) en envoyant des messages d'avertissement qui doivent être très courte pour être transmise en temps réel, informer les conducteurs en cas de problème leur permet de prendre la bonne décision, de ce fait, les applications liées à la sécurité routière ont une grande importance car elles concernent directement la réduction du nombre possible d'accidents, La classe de sécurité routière offre plusieurs fonctionnalités qui fournissent des informations de type différent pendant la route, parmi lesquelles on trouve cette fonctionnalité: [4]

Les applications de sensibilisation coopérative (CA)

Au cours de la conduite, l'environnement qui entoure le véhicule change constamment. Pour que le conducteur s'adapte aux conditions de la route, il doit être informé de l'environnement environnant. Dans cette catégorie, plusieurs applications offrent des informations différentes, parmi lesquelles : indication du véhicule à proximité ou motocyclette, véhicule lent, véhicule d'urgence et autres, véhicules diffusant des informations à d'autres véhicules à proximité, informations diffusées aidant les conducteurs à prendre de bonnes décisions.

Les applications d'assistance et d'aide à la conduite coopérative (CDA)

Parmi les services offerts par ce type d'application on mentionne :

Systèmes de conduite coopérative (CDS) : ce type de service consiste à effectuer le changement d'information capturé par les différents capteurs entre les véhicules, il aide les conducteurs à connaître l'état des véhicules proches pour laisser une distance de sécurité permettant de s'assurer que l'action des autres véhicules (i.e., freinage, doublage, arrêtez) ne causera aucun dommage. Le système calcule les progrès en tenant compte des nouvelles conditions, des conditions de sécurité et de la dynamique du véhicule.

Application d'avertissement de collision et risque de la route (RHCW)

Les capteurs du véhicule (capteur de pluie, radar avant et arrière, etc.) collectent diverses informations sur l'état dangereux de la route (pluie, obstacle, etc.). Les systèmes de détection d'accident (CD) sont basés sur ces informations pour détecter la collision imminente pour informer le conducteur d'être alerte à la collision, parmi les services de cette classe :

Avertissement Coopératif de Collision

Le véhicule surveille rapidement les messages d'alerte concernant l'état des véhicules à proximité afin d'éviter des accidents potentiels.

Émergence électronique des feux de stop

Un message d'avertissement doit être envoyé aux conducteurs menacés d'un véhicule afin de le maîtriser fortement afin de l'informer d'une situation critique dans les meilleurs délais.

Notification des risques de la route

Si un risque est détecté sur la route (vent, liquide, glace, etc.), le véhicule informe les autres riverains de son problème. Ou en détectant une caractéristique de la route (ralentisseur, virage, etc.), le véhicule en informe son voisin.

1.2.4.3. Services liés au confort

Plus que des services liés à la sécurité, les réseaux VANET offrent également d'autres types d'applications permettant d'assurer le confort des véhicules pendant le voyage. Parmi ces services, on trouve les jeux en réseau, la messagerie instantanée, l'accès à Internet, l'espace libres dans le parking pour le stationnement et le paiement. La possibilité de développer des applications dans ce domaine et très vaste grâce à la diversité des services qui permettent aux développeurs et aux opérateurs de télécommunication de bénéficier de revenus supplémentaires.

La localisation par carte

Parmi les applications les plus utiles en matière de confort, la localisation joue un rôle important pendant le trajet, car elle aide le conducteur à trouver le bon moyen d'atteindre sa destination dans des villes inconnues. Par exemple, il trace le chemin entre deux points de la ville, les stations d'essence, les hôpitaux, etc.

Parking Intelligent

Informer les conducteurs de la disponibilité des places de stationnement dans les parkings en indiquant aux conducteurs les places libres.

1.2.5. Les architectures de communication

Les principales composantes du réseau VANET sont les RSU (les stations de base) et les entités mobiles (véhicule), la communication entre ces entités permettra l'échange de données liées à la sécurité ou au confort de la route. Il existe trois modes de communication : [5]

1.2.5.1. Communications de Véhicule à Véhicule (V2V)

Dans cette architecture, un véhicule communique directement avec un autre véhicule. Il est vu comme un cas spécialisé du réseaux MANETs (Mobile AdHoc Network) ou les capacités d'énergie, de mémoire et de traitement est élevées. De plus, on peut prévisible la façon de mobilité. Cette proche peut être utilisé pour diffuser les messages d'alerte (Freinage, ralentissement, etc) ou dans la conduite coopérative. En effet, dans le cadre de sécurité routière la communication V2V est plus fiable que la communication V2I [5].

1.2.5.2. Communications de Véhicule à Infrastructure (V2I)

L'approche V2I ou communication en mode infrastructure utilise principalement les RSUs, généralement cette architecture est utilisée pour accéder aux applications de sécurité et de confort. La communication V2I est une communication entre un véhicule et l'infrastructure, par exemple pour informer le service routier d'un accident ou obstacle dangereux dans la route. Tandis que la communication I2V est une communication entre l'infrastructure et le véhicules, par exemple un panneau de signalisation informe les véhicules de feux rouge ou vert. En générale, le terme V2I englobe toutes les communications dans les deux sens [5].

1.2.5.3. Communication hybride

L'architecture hybride est combinée des deux premières approches. La combinaison de ces deux types d'architecture de communication permet d'obtenir une architecture hybride intéressante. En effet, les portées des infrastructures étant limitées, l'utilisation de véhicules comme relais permet d'étendre cette distance. Dans un but économique et en évitant de multiplier les bornes à chaque coin de rue, l'utilisation de sauts par véhicules intermédiaires prend toute son importance. Néanmoins, les communications inter-véhiculaires souffrent de problèmes de routage lors de transmission longue distance. Dans de telles situations, l'accès à une infrastructure peut améliorer les performances réseau. Nous comprenons donc la complémentarité des deux types de communication et l'intérêt d'une architecture hybride [6].

Un cas particulier de l'architecture hybride est le réseau VSN (*Vehicular Sensor Network*). En effet, ce type de réseau émerge en tant que nouvelle architecture de réseaux de véhicules. Le VSN a pour objectif la collecte et la diffusion proactive en temps réel des

données relatives à l'environnement dans lequel évoluent les véhicules. En effet, les voitures sont munies de plus en plus de capteurs de toutes catégories (cameras, capteurs de pollution, capteurs de pluie, capteurs d'état des pneumatiques, ESP, ABS, géolocalisation satellite, etc.). Les informations délivrées par ces équipements peuvent être utiles pour l'obtention d'états sur le trafic routier (embouteillages, ralentissements, vitesse moyenne du trafic, etc.), sur les places de parking disponibles, pour des informations plus générales telles que la consommation moyenne de carburant et le taux de pollution, ou encore pour des applications de surveillance (grâce aux caméras embarquées sur des véhicules).

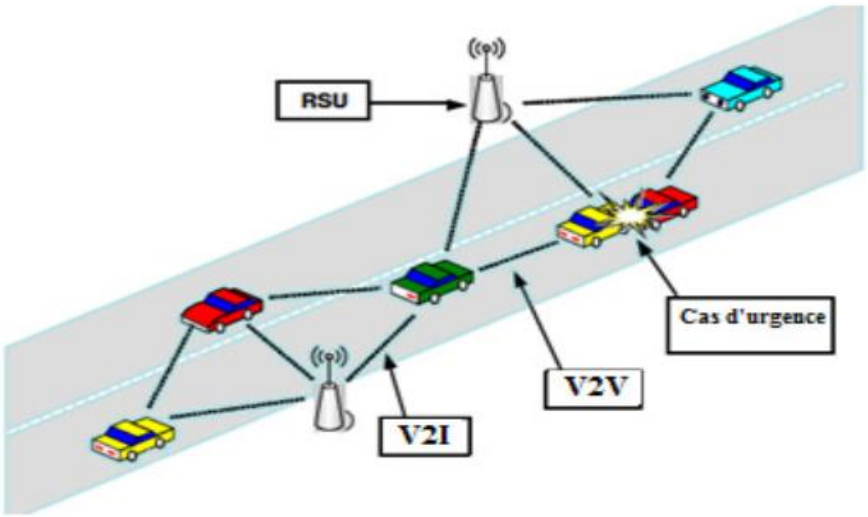

Figure 1-4 Types de communication dans les VANETs [7]

1.2.6. Les techniques de communications

En distingue deux types de communications :

1.2.6.1. Les communications radio

Un système de communication comprend tous les éléments capables de véhiculer de l'information (son, données informatiques, vidéo.) d'une source vers une ou plusieurs destinations.

Depuis la naissance des réseaux sans fil, les communications sont passées de la communication analogique filaire vers une communication numérique sans fil. De nombreuses technologies de communication peuvent être utilisées pour assurer l'échange de données entre véhicules, ces techniques en liaison décrites avec les communications radio. [8]

- Le système RDS (Radio Data System)

- Le système DAB (Digital Audio Broadcasting)
- Le système DVB (Digital Vidéo Broadcasting)

1.2.6.2. Les communications sans fil

Les technologies de communications sans fil sont en plein développement. Parmi les technologies utilisées dans les communications inter véhicules, nous pouvons citer : [6]

- Les systèmes qui réutilisent des infrastructures existantes, systèmes cellulaires de 2G vers 3G et d'autres évolutions (GSM, GPSR, 3GPP, etc).
- Les réseaux locaux sans fil (WLAN) qui sont principalement constitués des normes IEEE 802.11 (Wifi), IEEE 802.16 (Wi-MAX) et 802.11p (DSRC).

La communication sans fil dans l'environnement véhiculaire est basée sur deux entités d'équipements : le premier est un dispositif installé sur des sections critiques autour de l'infrastructure (stop, intersections, etc.), et le deuxième est celui qu'on trouve embarqué sur le véhicule. [10].

1.3. Introduction à la dissémination de messages

1.3.1. La dissémination

La dissémination de messages dans les réseaux véhiculaire consiste à router les messages via le réseau véhiculaire prenant en compte les caractéristiques de réseau ; la taille de réseau, la vitesse de nœud, le cas (message d'alerte ou contrôle), le message sera envoyé d'un véhicule à un autre ou au plusieurs, d'une façon fiable et en temp court. [11]

1.3.2. Types de message

Au cours de déplacement, les nœud de réseaux veut échanger des messages, les applications Vanet offert ce service, la diversité de cas de changement de message exige la diversité de type messages, pour ça, on distingue trois types de messages. [10]

Message de contrôle

Ce message contient des informations sur la position, la vitesse, la direction et l'itinéraire du véhicule émetteur, il est généré d'une façon périodique, toutes les (100 ms), ce type de message est généré pour informer les voisins de véhicules de son état actuel, les autres véhicules par le traitement de ces données peut anticiper les cas grave possible.

Message d'alerte

Le message d'alerte est généré pour informer les autres véhicules voisins d'une cas dangereux est survient, un accident, obstacle, ou quand 'il reçois un autre message d'alerte, en absence de cas dangereux. Les messages d'alerte ne sera diffusé pas, ce type de message comporte la position, l'heure et le type d'événement, il a une grande priorité et l'émetteur doit être sûr que tous les autres récepteurs reçois le message

Autres messages

Les applications de confort offrent la possibilité de changement de message, messagerie, transaction financière, et autres messages, ce type de message pas liés au temps. Tous les messages seront sauvegardés dans une cache pour une période, puis seront supprimés automatiquement.

1.3.3. Type de dissémination

On distingue deux types de dissémination : [13]

Dissémination pour les applications de confort

Les applications de confort consomme une grande quantité de bande passante à cause de types d'applications fournit. Par exemple, la messagerie, le téléchargement de fichiers, les jeux en ligne, etc. Donc il faut optimiser la bande passante, aussi les contenus de messages peut être dynamique et modifiable au cours de transmission de nœud à un autre.

Dissémination pour les messages d'urgence

Les applications de sécurité routière rendre la route plus sécurisée, les protocoles de dissémination doit prendre en compte les contraintes de l'envoi de messages d'urgence. La dissémination doit assurer les contraintes spatiales et temporelles, il faut que tous les véhicules informent les véhicules proches du cas urgent (accident, brouillard, neige, etc)

1.3.4. Stratégies de dissémination

Diffusion

Un message envoyé par une véhicule sera transmis à tous les voisins, puis ces voisins retransmettre le message, jusqu'à s'arriver à sa destination(s). Cette approche est parmi les plus utilisé pour la diffusion des informations dans les réseaux VANETs, car elle nécessite aucune information sur les véhicules qui entoure l'émetteur, chacun véhicule destinataire

reçois plusieurs messages pour cela elle ignore la fiabilité des informations et augmente le taux et la vitesse de diffusion, mais aussi il exige une concurrence d'accès au canal de communication et de consommation de bande passante. [12]

Probabiliste

Les véhicules utilisant cette approche utilisent ses connaissances, historique, et les données collectées sur la localisation et la mobilité des autres nœuds de réseau pour communiquer avec les autres véhicules. L'objectif de cette approche est de minimiser le nombre de message diffusé entre deux véhicules avant de choisir le chemin de dissémination de l'information.

Géographique

Chaque message contenant des informations nécessaires sur le véhicule émetteur par exemple message et les données de localisation. Cette dernière sera utilisée dans cette approche pour diffuser le message. Ces messages sont diffusés périodiquement. Si un véhicule proche est proactif, sinon sera diffusées à la demande dans l'approche réactive. Chaque nœud sauvegarde l'historique d'acheminement vers les autres nœuds voisin dans une table qui sera mis à jour régulièrement, puis ils utilisent cette table pour déterminer le plus court chemin pour atteindre à la destination. Par conséquent, cela permet de réduire le temp d'envoyé. Cette approche peut avertir notamment les véhicule qui sont menacées d'un accident probable dans un chemin à l'aide de ces coordonnées géographique.

Orientée ressources du canal

Malgré les caractéristiques de nœuds VANETs sont mieux par rapport les autres nœuds de réseaux MANETs (énergie, capacité de traitement et stockage, etc.). Il reste le problème d'accès au canal de communication, parce que les ressources de communications sont limitées, il y a plusieurs solutions dans cette approche, parmi ces solutions en mentionnes cette-là, l'amélioration de taux de réceptions de messages d'urgences par l'allocation d'une partie de la bande passante, chaque véhicule envoie un signal d'impulsion avant d'envoyer le message réel.

Orientée priorité des messages

Pour assurer la qualité de service des applications fournées dans les VANETs, il y a des solutions qui proposent une adaptation à la dissémination des informations qui se base sur

l'importance d'information transportée dans le message échangé. Pour éviter la suppression automatique de messages en cas de collision.

1.4. La convergence des applications VANET vers les applications VANET Cloud

1.4.1. Cloud Computing

Le Cloud Computing est un modèle d'accès réseau vise à partager de manière transparente et omniprésente un grand nombre de ressources informatiques. Celles-ci sont louées par un fournisseur de services à des clients numériques, généralement via Internet. [13] Le cloud computing est considéré comme un modèle économique plutôt que comme une technologie, les auteurs ont mené une enquête sur l'état de la technique en répondant à la question de savoir si le cloud computing va rester ou si c'est l'un des sujets passionnés qui sera inévitablement oublié dans les prochaines années. La plupart des acteurs du marché technologique tels que Google, Amazon et Microsoft accélèrent leur rythme dans le cloud computing en fournissant des services à leurs utilisateurs. [14]

1.4.2. Combinaison de VANET et du Cloud Computing

La architectures des réseaux véhiculaire VANETs, soit V2V ou V2I est le système le plus fiable et sécurisé pour assurer une communication et un contrôle efficace du trafic. Considérant le caractère de diffusion de la donnée, plusieurs paradigmes et sessions de communication, la création d'un réseau VANET selon les besoins moderne peut être difficile. Dans les MANETs, les nœuds s'autoorganisent pour créer un réseau sans le soutien d'une infrastructure par exemple (RSU). Donc le CC est censé d'être une nouvelle solution en vue de traiter ces problèmes. En effet, sa force réside dans son évolutivité, PaaS (plateforme as a Service), IaaS (Infrastructure as a Service), SaaS (Storage as a Service) et plusieurs autres caractéristiques importants. La notion de CC est née de la prise de conscience du fait qu'au lieu d'investir dans l'infrastructure, les entreprises peuvent trouver utile de louer l'infrastructure et parfois le logiciel nécessaire pour exécuter leurs applications. L'un des principaux avantages du cloud computing est son accès évolutif aux ressources et services informatiques [15]

1.4.3. Utiliser les Clouds dans les VANETs

Dans le réseau VANET, le CC peut être utilisé comme un réseau en tant que service (NaaS). Toutes les voitures sur la route n'ont pas accès à Internet. Dans NaaS, la véhicules avec accès à Internet peut offrir sa capacité excédentaire, sur demande, à d'autres véhicules. Il est clair

que de nombreux conducteurs disposeront d'une connectivité permanente à Internet via des réseaux cellulaires et d'autres points d'accès fixes sur la route en conduisant. Bien que certaines voitures n'aient pas accès à Internet, elles devront utiliser Internet. Dans telles circonstances, chaque conducteur disposant d'une connexion Internet qui souhaite partager cette ressource annoncera cette information à tous les véhicules qui l'entourent autour de lui. Ou le stockage en tant que service (SaaS), en mode SaaS, les véhicules ayant une grande capacité de stockage partagent le stockage avec d'autres véhicules ayant besoin d'une capacité de stockage pour une application temporaire. [15]

1.4.4. Les architectures VANET Cloud

Les VANET Cloud sont divisées en trois architectures principales, à savoir les Cloud véhiculaires (CV), les véhicules utilisant les Clouds (VuC) et les Clouds véhiculaires hybrides (HVC), chaque section sera développée dans les sous sections suivant : [16]

1.4.4.1. Vehicular Clouds (VC)

VC est formé de la manière suivante. Premièrement, les véhicules initient un protocole pour sélectionner un ou plusieurs courtiers parmi eux et identifier les limites des Clouds en sélectionnant une Entité Autorisée (EA) parmi les courtiers pour demander une autorisation afin de créer un Cloud. Une fois les nœuds et AE choisis. AE invite les nœuds véhiculaire situés dans les locaux de la limite du Cloud à participer au cloud. Les véhicules intéressés répondront avec un accusé de réception. Si le nombre de véhicules intéressés dépasse un certain seuil, AE demandera aux autorités supérieures l'autorisation de créer un Clouds et de fournir les ressources potentielles. Une fois l'autorisation obtenue, les participants du Cloud vont mettre leurs ressources en commun pour former un environnement virtuel riche. AE envoie le calendrier aux autorités supérieures et obtient l'autorisation de mise en œuvre. Notez que le travail en cours peut être confié au Cloud par les autorités supérieures en échange de certaines incitations offertes aux participants. AE dissout le cloud une fois le travail terminé.

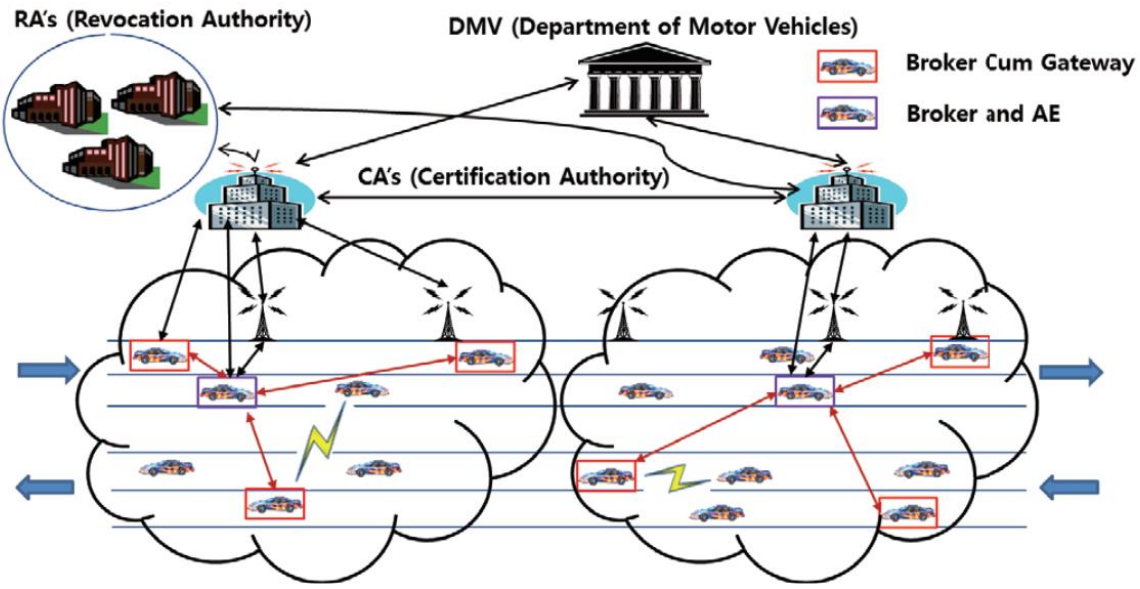

Figure 1-5 Vehicular Clouds [16]

1.4.4.2. VANET using Clouds (VuC)

La Figure 1.6 décrit l'architecture de VuC où VANET utilise des services cloud en déplacement. La couche de virtualisation est fournie par les passerelles. Notez que les RSU agissent comme des passerelles pour les véhicules vers les services cloud. La communication filaire à haute vitesse peut être utilisée depuis les UAR jusqu'aux

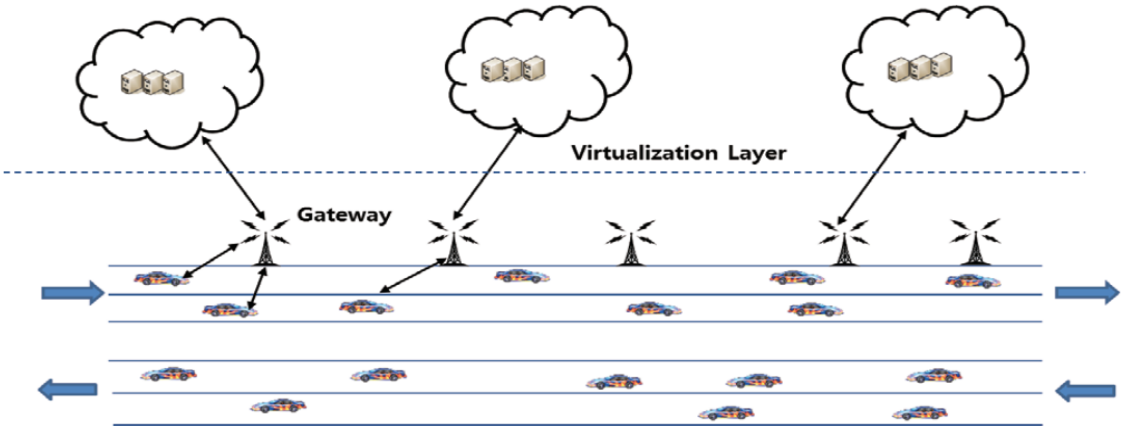

Figure 1-6 VANET using Clouds [16]

services en nuage. Comme décrit dans la taxonomie des nuages VANET, les services offerts par VuC incluent des informations de trafic en temps réel et un système d'information multimédia.

1.4.4.3. Hybrid Clouds

HC est la combinaison de VC et de VuC, où VC sert à la fois de fournisseur de services et de consommateur. L'architecture de HC est illustrée à la Figure 1.7 . HC est motivé par le fait que les véhicules circulant sur la route risquent de louer leurs ressources et de vouloir utiliser les services de cloud computing en même temps. NaaS et P2P sont les exemples Les plus appropriés pour de tels scénarios. Néanmoins, en raison de la nature éphémère de VANET, la connexion entre les nœuds véhiculaires est très intermittente. Cependant, on peut affirmer que, généralement, pour les applications P2P, la taille des fichiers est relativement petite, ce qui le rend approprié pour une connexion de courte durée. Les autres applications potentielles de cette architecture incluent IaaS dans le cas de VC.

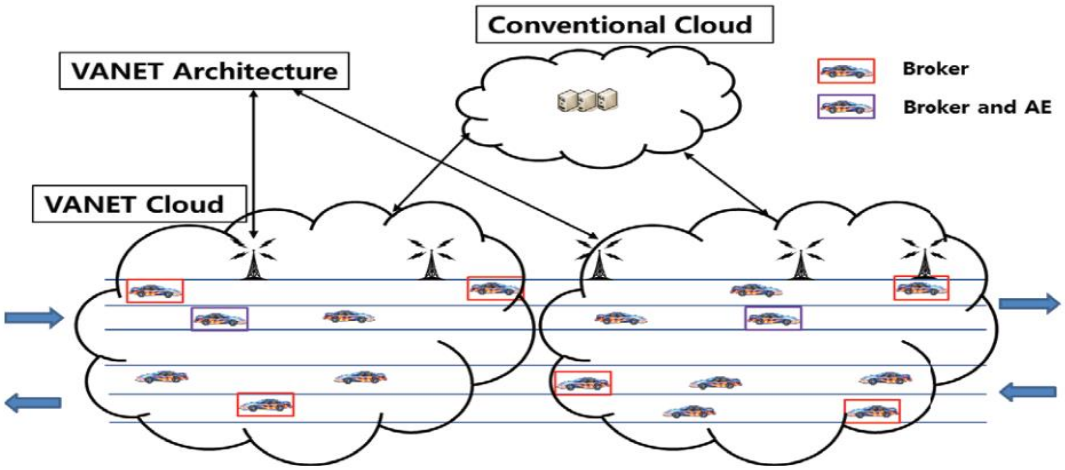

Figure 1-7 Hybrid Clouds [16]

1.4.5. Les Applications VANET Cloud

1.4.5.1. Remote Configuration and Car Performance Checking

En utilisant la technologie cloud, une voiture peut être surveillée et déboguée à distance. Cette technologie a déjà été mise en œuvre par une société automobile bien connue, Hyundai, qui surveille ses voitures dès la chaîne de montage. Les performances de la voiture sont vérifiées à distance en recevant constamment des données de la voiture et sont utilisées pour fournir des services de qualité aux clients. Néanmoins, la confidentialité des utilisateurs et des lieux est une préoccupation majeure pour une telle approche. [14]

1.4.5.2. Big traffic data analysis

Les véhicules produisent une grande quantité de données sur le trafic. Par exemple, les messages de balise produisent des données à l'échelle de millisecondes, qui sont précieuses pour les services fournis par VANET. Le stockage et le traitement des données nécessiteraient une grande quantité de ressources de stockage et de calcul. Ces données pourraient être utilisées à diverses fins, allant des informations routières au divertissement. [14]

1.4.5.3. Smart location-based advertisements

Avec l'avènement des véhicules intelligents, les panneaux d'affichage situés en bordure de route risquent d'être remplacés par un système de publicité dans la voiture, dans lequel les conducteurs peuvent obtenir les annonces de leur intérêt en fonction de leur choix. Comme indiqué ci-dessus, les données de localisation, le modèle de conduite et les requêtes d'emplacement des pilotes permettraient aux serveurs décisionnaires dans le cloud d'annoncer les événements et / ou les lieux de manière intelligente aux utilisateurs, en fonction de leurs intérêts. [14]

1.4.5.4. Vehicle Witnesses

Récemment, un chercheur a proposé une solution de véhicule témoin (VWaaS) qui exploite l'infrastructure en nuage pour sauvegarder les données judiciaires originales en cas d'incident sur la route. Lors de la détection d'un événement ou de l'instruction des autorités, les véhicules prennent des photos du site d'intérêt et l'envoient dans le cloud. L'infrastructure en nuage enregistre ces images en tant que preuves médico-légales et fournit plus tard les informations médico-légales aux organismes chargés de l'application de la loi, à la justice et / ou aux agences d'assurance. [14]

1.5. Conclusion

Dans ce chapitre nous décrivons les réseaux VANETs Cloud d'une manière générale, tout d'abord, nous présentons les réseaux VANET, les composant, les caractéristiques et les déférents applications développées dans ce contexte, puis nous mentionnant les techniques de dissémination des messages dans un réseau VANET et enfin, l'intégration des services Cloud Computing dans les VANETs et la convergence de VANET vers VANET Cloud.

Notre objectif est de réaliser une plateforme de simulation qui permettent de simuler les scenarios de disséminations des informations dans les VANETs Cloud, alors, Dans le chapitre

suivant, nous essayons décrire les techniques et les approches de dissémination des informations dans un environnement VC.

CHAPITRE 02

Les Modèles De Simulation Et Les Techniques De Dissémination Des Informations Pour Les VANETs Cloud

1. Les Architectures Cloud Véhiculaire Pour La Dissémination Des Informations
2. Les Données De Dissémination D'Information
3. Les Techniques De Dissémination D'Information Pour Les Vanet Cloud
4. Les Environnements de Simulation de VANET Cloud

2.1. Introduction

Les modèles de simulation pour les environnements VANET sont devenus importants pour fournir une vue d'ensemble du réseau. Ces modèles visent essentiellement à évaluer les performances d'un tel réseau, la conception de protocoles, la dissémination des information, l'étude de la mobilité et la congestion routière. Dans les cas où, par exemple, des accidents de la route ou tout autre type d'information du trafic disséminer sur le réseau véhiculaire peuvent influencer la dynamique du trafic et dégrader les performances réseau.

Étant donné les essais sur le terrain visant à évaluer les applications véhiculaires innovantes pouvant être coûteux et risquant de créer des conditions de circulation dangereuses, il ne s'agit souvent pas d'une procédure d'essai concrète. Les chercheurs du domaine cherchent à développer des plateformes de simulation appropriées pour tester et évaluer les fonctions VANET avancées. À partir de là, les concepts de simulation du trafic réseau et de trafic doivent être combinés afin de mettre en places des stratégies permettant de déminer les coûts associés à la congestion.

Ce chapitre décrit dans un premier temps les déférents approches et les déférent techniques de dissémination des informations dans les environnements VANET Cloud. Ensuite, nous allons présenter les simulateurs réseautiques ou les simulateurs de gestion de mobilité et de trafic routier.

2.2. Les architectures cloud véhiculaire pour la dissémination des informations

Dans les réseaux de véhicules ad hoc (VANET), une diffusion efficace est essentielle à la sécurité routière et à l'efficacité du trafic. De nombreux systèmes basés sur VANET souffrent de temps de transmission élevés et de redondance des données. Les chercheurs devaient trouver des solutions pour assurer une diffusion fiable dans un court délai.

Avec le développement technologique, plusieurs inventions deviennent très utiles avec ces problèmes, parmi lesquelles nous trouvons le concept Cloud, Cluster, La nouvelle technologie LTE, etc., en fonction de la technologie utilisée, l'architecture du réseau s'échange également. Parmi ces architectures, on citera quelques :

2.2.1 Les architectures VANET Cloud

Le manque de flexibilité, la faible connectivité et d'autres problèmes de communication dans VANET peuvent désormais être résolus en intégrant le *Cloud Computing. M Talib et al*, ont défini le « *Vehicular Cloud Computing* » (VCC), comme une nouvelle technique qui exploite les avantages du CC afin d'aider les réseaux VANET avec divers services de calcul. Ces services peuvent améliorer le contrôle de la circulation en diminuant les accidents de la route, les embouteillages et le temps de trajet [32]. *T hadjji et H Bergaoui* proposent la conception d'un projet d'essai VANET basé sur le Cloud Computing et utilisant son implémentation réelle Car2Car. Ils démontrent comment la technologie de Cloud Computing peut constituer une nouvelle plateforme pour tester la mise en œuvre réelle à grande échelle de protocoles de réseau ad-hoc pour véhicules, il trouve que le principal avantage de l'utilisation du Cloud Computing est son évolutivité pour les applications système volumineuses, ce qui permet de réaffecter les ressources de manière dynamique à la demande [33]. *R Hussain et al*, divisent VANET Cloud en trois architectures principales ; Cloud Véhiculaire (VC), Véhicules utilisant Cloud (VuC) et Cloud Hybride (HC), VC divisé en deux scénarios du point de vue du mouvement, Statique et Dynamique, sont des véhicules fournissant des services de cloud. Par exemple, un super ordinateur virtuel formé par la collaboration de véhicules, VuC connecte le réseau VANET aux Clouds traditionnels et dans les HC les Clouds véhiculaires interagissent avec le Cloud traditionnel ou les véhicules et les RSU jouent le rôle de passerelle [16].

2.2.2 Les architectures hétérogènes

Bien que ces systèmes basés sur VANET soient viables, il est toujours difficile de transmettre des messages rapides et fiables à une zone ciblée dans un environnement de réseau de véhicules intermittent. En effet, en raison de la portée de transmission limitée de DSRC « *Communication dédiée à courte portée* » ainsi qu'une autre méthode de transmission dans le protocole IEEE802.11p. Ce problème est encore aggravé dans les scénarios urbains. Ce problème peut être résolu en utilisant des réseaux cellulaires hétérogènes intégrés dans les VANETs. Dans ces cellules, les véhicules équipés avec des interfaces 3G, 4G ou IEEE802.11p. Ces interfaces sont choisies comme candidats de passerelle. *Bingyi Liu et al*, propose un système de dissémination de messages d'alerte et du sécurité qui se compose à la fois d'une infrastructure Cloud et d'un réseau hétérogène VANET-cellulaire dans laquelle les bus agissent comme des passerelles mobiles, un réseau basé sur un bus est une solution pratique dans un scénario urbain pour les raisons suivantes :

1) la plupart de bus aujourd'hui peut accéder au internet

2) un système de transport public donne accès à un grand nombre d'utilisateurs (par exemple, les passagers eux-mêmes)

3) les temps de déplacement peuvent être prédits à partir de la grille horaire du système de transport.[34].

M S Rayeni et al, présentent un service à base multidiffusion, qui n'a pas encore été bien étudiée dans les HetVNets (heterogeneous vehicular networks), des solutions naïves pour fournir un service est la monodiffusion entre le fournisseur de services et chaque client. La diffusion individuelle consomme une bande passante considérable, auquel *Rayeni et al* pensent qu'il s'agit du premier travail fournissant un service de multidiffusion fondé sur la qualité de service dans HetVNets avec une utilisation minimale de la bande passante V2V sur tous les segments de rue. Dans ce cas, deux approches ont été proposer pour modéliser l'utilisation de la bande passante totale d'un arbre de multidiffusion. Des simulations appliquées montrent que les approches proposées, comparées aux approches existantes, minimisent de l'utilisation de la multidiffusion dans la bande passante dans le réseau VANET [35].

Abdennour Zekri et Weijia Jia Ont étudié et comparé l'intégration des réseaux sans fil hétérogènes VANET. Les articles étudiés ont été choisis en fonction de différentes classifications. De plus, le transfert vertical, la distribution et la collecte de données, la sélection de passerelle et d'autres sont pris en compte. Ils conclurent que L'intégration des réseaux sans fil hétérogènes VANET joue un rôle essentiel dans la réussite des projets de véhicules sans conducteur. D'autre part, ces travaux aideront les futurs chercheurs à se faire une idée de la collaboration entre réseaux hétérogènes de véhicules pour développer des réseaux VANET.[36]

2.2.3 Les architectures véhiculaires sociales

A.Rahim et al, ont été présenter le réseau social véhiculaire comme un concept d'intégration du comportement social possible de voyageurs dans des environnements véhiculaires. Les VSN intègrent des réseaux sociaux pour communiquer en fonction d'intérêts, de comportements et d'objectifs sociaux dans la communauté virtuelle de véhicules, d'RSU, de dispositifs intelligents pour les conducteurs et les passagers. Les réseaux sociaux et leurs applications sont actuellement examinés par la communauté des chercheurs et ont montré que les connaissances sociales acquises par l'interaction des nœuds pouvaient contribuer à améliorer les performances des systèmes de communication mobile. Il est supposé que les comportements sociaux des entités de réseau pourraient être exploités pour un large éventail d'applications [50].

2.3. Les données de dissémination d'information

2.3.1 Les données Uplink

Les données uplink indique les données envoyées de véhicules vers le serveur cloud [34].

2.3.2 Les données downlink

Les données downlink indique les données envoyées de serveur cloud vers le/les véhicules [34].

2.4. Les techniques de dissémination d'information pour les Vanet Cloud

2.4.1 Le routage

Le réseau ad hoc de véhicules fournit des applications telles que le partage de contenu multimédia, les services Internet, le transfert de fichiers, les jeux, etc. Le routage joue un rôle important dans la fourniture de meilleurs services aux utilisateurs de VANET. *I.Wahid et al*, adoptent une étude comparative entre quelques protocoles de routages proposés récemment, ils conclurent que les performances du protocole de routage dans le réseau ad hoc dépendent largement du scénario opérationnel et les changements dans un environnement opérationnel ont un effet positif ou négatif sur les performances de routage. Aussi Les articles de recherche sur les performances des protocoles de routage montrent qu'un protocole de routage particulier surpasse les autres dans un scénario de mobilité particulier et pour une mesure de performance particulière. De même, un autre protocole de routage particulier peut surpasser les autres dans un autre scénario. Le modèle de mobilité joue un rôle important dans l'étude des performances des réseaux ad hoc de véhicules [37]. Dans ce qui suit, nous présentons quelques classifications possibles dans VANETs donnée par *Venkatesh et all*. [61] (voir Figure 2.1).

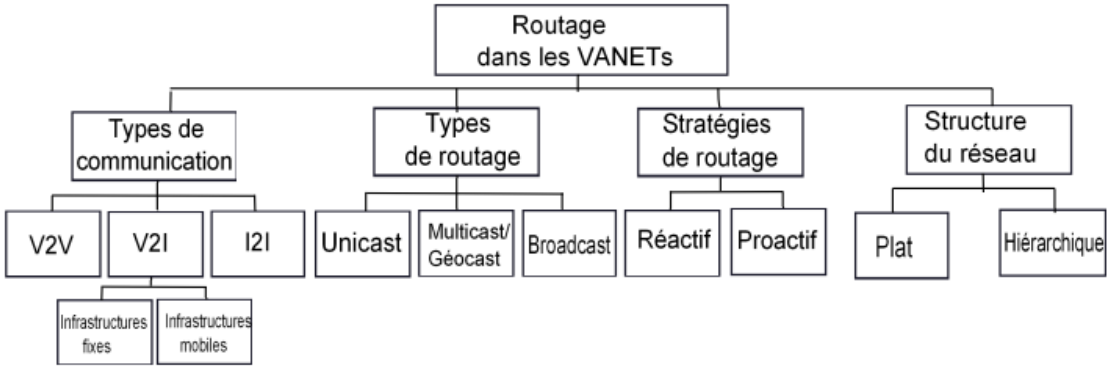

Figure 2-1 Routage dans les VANETs [38]

2.4.1.1 Types de routage

Wahid [37] a mentionné les types de routage selon le nombre d'émetteurs et de récepteurs :

1- **Routage Unicast :** l'information transmettre d'un nœud vers un autre soit avec un seul saut soit avec multi-sauts.
2- **Routage Multicast/Géocast :** l'information transmettre d'un nœud vers un ensemble des nœuds ou vers une zone précisée.
3- **Routage broadcast :** l'information transmettre d'un nœud vers tous les nœuds connectés.

2.4.1.2 Stratégies de routage

Sharef et al.[39] a expliqué les deux stratégies principales qui utilisée dans la plupart de protocole de routage d'information :

Routage Proactif (Table-Driven)

Les informations de tous les nœuds associés sont stockées sous la forme de tables, ces protocoles étant basés sur des tables. Ces tables sont également distribuées avec leurs voisins et les nœuds renouvellent leurs tables de routage lorsque la topologie du réseau change.
Les protocoles proactifs n'ont pas de délai de découverte de route initial mais consomment une bande passante importante pour les mises à jour périodiques de la topologie. Parmi ces protocoles, on mentionne :

Optimized Link State Routing Protocol (OLSR) : Il utilise un échange périodique de messages pour conserver les informations de topologie du réseau sur chaque nœud. OLSR est une optimisation par rapport à un protocole d'état de lien pur, car il calcule la taille des informations envoyées dans les messages et réduit le nombre de retransmissions à diffuser ces messages dans tout le réseau. À cette fin, le protocole utilise une technologie de relais multipoint pour inonder de manière efficace et économique ses messages de contrôle.[40]

Dynamic destination-Sequenced Distance Vector (DSDV) : Les paquets sont transmis entre les stations du réseau en utilisant des tables de routage qui sont stockées dans chaque station du réseau. Le protocole DSDV exige que chaque station mobile annonce à chacun de ses voisins actuels sa propre table de routage. Les informations de routage sont annoncées par diffusion multidiffuser les paquets qui sont transmis périodiquement et progressivement quand des changements topologiques sont détectés.[41]

Routage Reactif (On-Demand)

Les protocoles de routage réactifs sont également appelés " On-Demand Routing". Les protocoles de routage à la demande signifient qu'une route est établie exclusivement lorsque les paquets de données doivent être livrés à la destination. Les protocoles de routage à la demande adoptent une approche de routage différente de celle des protocoles basés sur les tables, tels que *Temporally Ordered Routing Algorithm* (TORA), *Ad hoc On-Demand Distance Vector Routing* (AODV) et *Dynamic Source Routing* (DSR). Les routes vers la destination sont découvertes lorsqu'ils sont réellement nécessaires. Lorsqu'un nœud source doit envoyer des paquets de données à la destination, il commence à vérifier la table de routage pour déterminer s'il dispose ou non d'un chemin. S'il n'y a pas de route existante, le nœud source lance la procédure de découverte de route afin de trouver un chemin d'accès à la destination.

***Ad hoc On-Demand Distance Vector Routing* (AODV) :** Dans le cas du routage vectoriel à distance sur demande ad hoc, pour rechercher un chemin vers la destination, la source diffuse un paquet de demande d'itinéraire. Ce message de diffusion se propage sur le réseau jusqu'à atteindre un nœud intermédiaire disposant d'informations de routage récentes concernant la destination ou jusqu'à ce qu'il atteigne la destination. Lorsque les nœuds intermédiaires transmettent le paquet de requête de route, il enregistre dans ses propres tables le nœud d'où provient la requête de route. Ces informations sont utilisées pour former le chemin de réponse pour le paquet de réponse de route, car AODV utilise uniquement des liens symétriques. Lorsque le paquet de réponse d'itinéraire retourne à la source, les nœuds situés le long du chemin inverse saisissent les informations de routage dans leurs tables. Chaque fois qu'une défaillance de liaison se produit, la source est notifiée et une découverte de route peut être à nouveau requise si nécessaire.[42]

***Dynamic Source Routing (DSR)* :** Chaque hôte mobile participant au réseau ad hoc maintient un cache de route dans lequel il met en cache les routes source qu'il a apprise. Lorsqu'un hôte envoie un paquet à un autre hôte, l'expéditeur vérifie d'abord dans son cache de route un chemin source vers la destination. Si une route est trouvée, l'expéditeur utilise cette route pour transmettre le paquet. Si aucun chemin n'est trouvé, l'expéditeur peut tenter d'en découvrir un en utilisant le protocole de découverte de route. En attendant la fin de la découverte de route, l'hôte peut poursuivre le traitement normal et envoyer et recevoir des paquets avec d'autres hôtes. L'hôte peut mettre en mémoire tampon le paquet d'origine afin de le transmettre une fois que la route est apprise grâce à la découverte de route, ou le rejeter, en faisant appel à un logiciel de protocole de couche supérieure pour retransmettre le paquet si nécessaire. Chaque

entrée du cache de routage est associée à une période d'expiration, après, l'entrée est supprimée du cache.[43]

2.4.2 La migration de la Virtual machine (VM)

Les ressources matérielles des véhicules et des infrastructures routières sont strictement limitées dans les réseaux de véhicules. L'application de la technologie en cloud mobile améliorera considérablement l'utilisation de ressources physiques intensives. Dans le nouveau paradigme des réseaux de véhicules assistés par le cloud, la mobilité des véhicules constitue un défi majeur pour la continuité des services dans le cloud. Les mécanismes de migration dynamique de la machine virtuelle (VM) traitent le problème.[44]

Une machine virtuelle permet à un utilisateur de conFigurer un environnement et d'exécuter ses processus uniques sans se soucier des ressources physiques impliquées. Les ressources sont gérées par des hyperviseurs qui, à leur tour, isolent les ordinateurs virtuels, bien que certains puissent partager la même ressource physique, ce qui rend chaque ordinateur virtuel (et son utilisateur correspondant) inconscient de ses voisins. Le déplacement de la machine virtuelle d'un emplacement physique à un autre est simple et potentiellement transparent. De telles migrations de machines virtuelles (VMM) peuvent être déclenchées pour gérer des points chauds ou pour équilibrer la charge des ressources.[45]

2.4.2.1 Scénarios de migration VM

La migration en direct d'une machine virtuelle fait référence au processus de transfert d'une machine virtuelle en exploitation avec ses applications sur différentes machines physiques. Dans les réseaux de véhicules assistés par le cloud, les véhicules disposent de services de cloud depuis des infrastructures en bordure de route, des serveurs centraux ITS et d'autres véhicules. Lorsqu'un véhicule se déplace sur la route, il doit transférer différentes infrastructures. Dans le même temps, le service cloud doit passer d'un site cloud à un autre. Puisque VM est l'entité fondamentale chargée des services de cloud computing, la migration directe de VM est un moyen privilégié de garantir la continuité du service. Dans une migration de machine virtuelle, l'image de la machine virtuelle doit être copiée de la source sur le site cloud. Dans les réseaux véhiculaires assistés par le cloud, la migration en direct de VM repose sur plusieurs scénarios.[45]

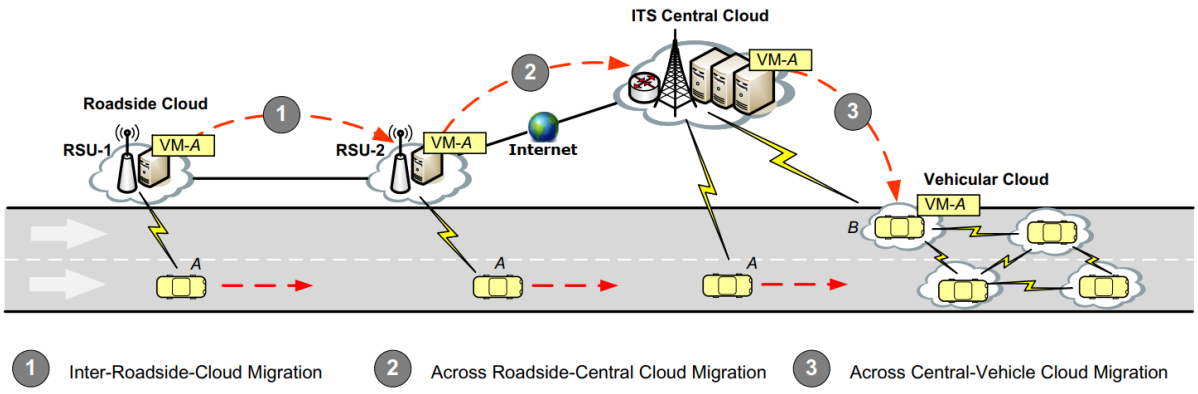

Figure 2-2 Scénarios de migration de machine virtuelle [45]

- ***La migration Inter-Roadside-Cloud :*** Sur la Figure 2.2 (voir le cas 1), lorsque le véhicule A passé de la plage radio de RSU-1 à celle de RSU-2, une migration de machine virtuelle est nécessaire. L'invité VM-A doit être transféré entre les deux cloudlets en bordure de route pour reprendre son service.
- ***La migration Across Roadside-Central Cloud :*** Sur la Figure 2.2, le cas 2, lorsque le véhicule A sort de la portée radio de RSU-2. Plus de nuage en bordure de route, mais nuage central disponible. Dans ce cas, l'invité VM-A doit être migré du nuage en bordure de route vers le nuage central. Après cela, le véhicule A reprendra son service en accédant au nuage central via des communications sans fil.
- ***La migration Across Central-vehicular*** **:** Sur la Figure 2.2 le cas 3, lorsque le véhicule A passe dans le rayon radio d'autres véhicules, par exemple le véhicule B, le véhicule A la possibilité de transférer sa VM sur le véhicule B. Après la migration, le véhicule A reprendra son service en accédant au nuage véhiculaire en utilisant des communications V2V.
- ***La migration Across Roadside-Vehicular Cloud :*** Ce scénario, qui n'est pas illustré à la Figure 2.2, est similaire à celui existant dans « *Across Roadside-Central Cloud* », à l'exception du fait que le véhicule de destination doit connecter au RSU source afin qu'il existe une liaison de données pour la migration de VM.

2.4.3 Clustering

Les VANET présentent plusieurs défis, tels que la mobilité élevée des nœuds, le problème des terminaux cachés, le nombre limité de canaux, les déconnexions fréquentes etc. Les chercheurs ont proposé plusieurs solutions à ces problèmes. Le clustering est l'une des solutions proposées. Dans le Clustering, les véhicules présentant les mêmes caractéristiques (c'est-à-dire vitesse et accélération) forment un groupe appelé Cluster. Une tête de cluster CH

(Cluster Head) est sélectionnée parmi les véhicules de cluster pour effectuer des activités de contrôle au sein de cluster. Le clustering réduit les déconnexions fréquentes et les effets d'une grande mobilité, améliorant ainsi les performances globales du réseau [46].

Les clusters virtuels représentent notamment une solution afin d'optimiser la dissémination des informations du trafic dans les réseaux cloud véhiculaires. L'avantage est que chaque cluster déplace dynamiquement par le changement des membres de groupe, chaque cluster besoin d'un CH (Cluster Head) qui assure la communication et l'échange d'informations (vitesse, nombre de membre, etc.) avec le monde extérieur, par contre la communication inter-cluster se fait par la communication V2V.

En particulier, les services cloud, stockage, calcule, sécurité, peuvent être différencier et invoquer en utilisant les clusters virtuels. Les clusters virtuels peuvent être former pour généraliser l'estimation du trafic en incluant tous les types des nœuds. H R Arkian et al, proposent une approche de clustering en trois-étapes pour l'estimation du flux du trafic. L'approche est un mécanisme évolutif qui exploite efficacement le nombre de véhicules équipés circulant sur un segment de route pour estimer de manière optimale la densité de trafic totale. Pour cette raison, ils ont fourni une nouvelle méthode de clustering, une technique de chaînage de cluster et une méthode généraliser pour l'estimation du trafic. Les résultats de la simulation décrivent la précision de leur approche par rapport au schéma « *online learning weighted support-vector regression* » (OLWSVR) [47].

2.4.3.1 Procédure de formation du cluster

Les véhicules équipés forment des groupes dynamiques et ceux qui sont plus adéquats deviennent des têtes du groupes (Cluster Head). Les groupes dynamiques sont eux-mêmes mobiles et se déplacent avec les véhicules à grande vitesse. Ainsi, même avec les véhicules à grande vitesse, l'architecture en cluster en mouvement produit une topologie relativement stable, tant que la vitesse des véhicules reste à peu près la même. Pour la formation du cluster, *R.Arkian et al*, proposent une approche, lorsqu'un véhicule entrant dans un nouveau segment de la route rechercherait tout cluster disponible en diffusant un message de demande de participation au cluster, ou en communiquant avec un RSU lorsqu'il se trouvait dans son rayon de communication. Lorsque le véhicule initial attend un certain temps et qu'il ne reçoit aucune réponse, il lance le processus de formation du cluster pour identifier les membres du cluster en diffusant un message initial. Ensuite, pour former un cluster, le véhicule initial besoin d'informations du véhicule voisin, en général, les véhicules qui construisent leur relation de voisinage transmettent leurs données de vitesse et de position actuelles intégrées dans les messages HELLO à d'autres véhicules se trouvant dans leur rayon de communication et le

véhicule initial, en conséquence. Les véhicules se déplaçant dans la même direction et à proximité les uns des autres appartiennent à un groupe primitif. Cependant, les niveaux de vitesse dans certaines zones sont différents et cette variation peut être très importante ; donc, tous les véhicules voisins ne sont pas appropriés pour être inclus dans un cluster.

Finalement, Pour sélectionner des membres de cluster avec le même niveau de vitesse, le véhicule initial compare d'abord la différence de vitesse de tous ses voisins avec un seuil. Si la différence de vitesse du voisin du véhicule initiale est inférieure au seuil, le véhicule voisin est considéré comme un membre du groupe primitif. Cette comparaison permet de supposer que les membres du groupe se déplacent presque à la même vitesse. Ensuite, le véhicule initial calcule le nombre de membres du groupe primitif. Si le nombre de membres est supérieur à membre primitive du cluster, le véhicule initial broadcast un message permet de notifier son ID aux membres du cluster. Sinon, le véhicule initial rejette le processus de formation du cluster et attend à nouveau un autre temp, [47], dont le processus de sélection du CH (Cluster Head) de cette méthode sera présentée dans la section suivante.

I.Ahmed et al, présentent un algorithme de formation de grappes comprend trois phases principales, la vérification de l'état des grappes, la formation de grappes et la maintenance des grappes. Le processus de formation de cluster est intégré à l'algorithme de sélection CH. La fusion de la formation de cluster avec la sélection CH est unique et essentielle. La fusion comprend un mécanisme dans lequel chaque véhicule se voit offrir le choix de participer ou non à un cluster. Si un véhicule choisit de faire partie d'un groupe, il peut alors être choisi comme CH et coopérer avec d'autres CM. Ils ont utilisé des paramètres tels que les valeurs de destination ainsi que la direction et la vitesse moyenne [48].

2.4.3.2 Procédure de sélection du Tête de Groupe (Cluster Head)

H.Arkian et al, modélisent un algorithme de sélection de têtes de cluster. Les informations de sélection de tête de cluster pour tout nœud sont limitées aux nœuds situés dans une distance de transmission par rapport au nœud lui-même. La priorité d'un nœud de devenir un CH est déterminée par la valeur de son facteur Befit. Ainsi, les nœuds commencent d'abord à calculer leur facteur Befit pour qu'il devienne un CH et à diffuser des messages contenant leur BF.

Ensuite, chaque nœud vote pour son voisin ayant le maximum local BF. Un nœud peut aussi voter pour lui-même, s'il a le maximum BF. Les nœuds utilisent leurs messages de vote spéciaux pour diffuser localement leurs votes. Une fois la procédure d'élection terminée, le nœud élu reconnaît sa sélection en tant que responsable de cluster en modifiant son état en CH et en informant les autres nœuds. Par la suite, les véhicules voisins modifient leur identifiant de cluster en identifiant du nouveau CH [47]. Une autre procédure complètement

déférente est proposée par R S Bali et N Kumar, Une fois que chaque véhicule a déterminé sa position future, la décision concernant le prochain CH à sélectionner est prise à l'aide d'un environnement informatique centralisé tel que RSU et le cloud. Les nœuds élisent le CH en transmettant leur position prévue au cloud. Cependant, au lieu d'utiliser des métriques traditionnelles, le schéma proposé utilise un paramètre de formation de cluster (σ). La valeur de σ est basée sur le nombre de nœuds voisins. Le nombre de nœuds voisins est calculé en fonction de la portée de transmission des véhicules. Tous les véhicules dans la zone de transmission sont supposés être les voisins de l'autre véhicule. Initialement, le nœud avec les valeurs σ les plus élevées est élu en tant que CH et les nœuds situés dans leur voisinage à un saut jouent alors le rôle de membres du cluster [49].

2.4.3.3 Procédure de maintenance du cluster

Outre l'algorithme de formation de cluster, nous avons également besoin d'un algorithme de maintenance de cluster pour faire face aux changements de topologie causés par les véhicules qui se joignent et qui quittent fréquemment. I Ahmed et al, présente cet algorithme de maintenance. Si un cluster est formulé et qu'un CH est sélectionné avec succès à la fin de la phase de formation du cluster, la maintenance du cluster commence, sinon, le processus de formation de cluster est répété. Un cluster formé a besoin de maintenance dans le sens où de nouveaux membres rejoignent ou des membres existants quittent le cluster. Pour quitter le cluster avant d'atteindre la destination. Si tous les véhicules quittent le CH ou que le CH quitte le cluster, alors tout véhicule qui souhaite une formation de cluster démarre un nouveau processus de cluster [48]. H R Arkian et al, utilisent un algorithme de maintenance contenant trois scénarios différents comme suit :[47]

- ***Cluster Joining*** **:** Lorsqu'un véhicule non-cluster envoie un message de demande d'adhésion à une tête de cluster, le chef de cluster vérifie si sa vitesse relative est dans les limites du seuil de cluster ; Si oui, le responsable du cluster acceptera le véhicule en ajoutant son ID à la liste des membres du cluster.
- ***Cluster Leaving*** **:** Lorsqu'un véhicule quitte le groupe, la tête du cluster perd le contact avec celui-ci. Par conséquent, la tête de cluster supprime ce véhicule de la liste des membres du cluster.
- ***Cluster Emergin*** **:** Lorsque deux têtes de grappes entrent dans les plages de transmission et que leurs propriétés sont identiques (par exemple, vitesse relative), la tête de grappe avec une valeur de pertinence inférieure abandonne son rôle de tête de grappe et devient membre de la grappe.

2.5.Les Environnements de Simulation de VANET Cloud

2.5.1 Le Simulateur OMNET++

OMNeT ++ est une bibliothèque et une infrastructure de simulation C ++ à base de composants, extensibles et modulaires, qui incluent également un développement intégré et un environnement d'exécution graphique. Les fonctionnalités spécifiques à un domaine (prise en charge de la simulation des réseaux de communication, des réseaux de mise en file d'attente, de l'évaluation des performances, etc.) sont fournies par des cadres de modèle développés en tant que projets indépendants. Il existe des extensions pour la simulation en temps réel, l'émulation de réseau, la prise en charge de langages de programmation alternatifs (Java, C #), l'intégration de base de données et plusieurs autres fonctions. [22]

La conception d'OMNeT ++

OMNeT ++ a été principalement conçu pour prendre en charge la simulation réseau à grande échelle. Cet objectif conduit aux principales exigences de conception suivantes :

- Activer la simulation à grande échelle. Les modèles de simulation doivent être hiérarchisés et construits autant que possible en composants réutilisables.
- Le logiciel de simulation devrait faciliter la visualisation et le débogage des modèles de simulation afin de réduire le temps de débogage qui occupe généralement une grande partie des projets de simulation.
- Le logiciel de simulation doit être modulaire, personnalisable et permettre l'intégration de simulations dans des applications plus vastes telles que les logiciels de planification de réseau.
- Les interfaces de données doivent être ouvertes. Il devrait être possible de générer et de traiter des fichiers d'entrée et de sortie avec des outils logiciels couramment disponibles.
- Fournir un environnement de développement intégré qui facilite grandement le développement du modèle et l'analyse des résultats.[21]

2.5.1.1 Les Plateformes de simulation

INET

INET Framework est une bibliothèque de modèles open source pour l'environnement de simulation OMNET ++. Il fournit des protocoles, des agents et d'autres modèles aux chercheurs et aux étudiants travaillant avec des réseaux de communication. INET est particulièrement utile lors de la conception et de la validation de nouveaux protocoles, ou lors

de l'exploration de scénarios nouveaux ou exotiques. La plateforme INET fournis plusieurs fonctions pour les projets du VANETs spécialement, parmi ces avantages :

- Le positionnement et la mobilité des nœuds sont importants dans les scénarios de simulation du VANETs. Dans INET, la mobilité est ajoutée aux nœuds sous la forme de modules représentant des modèles de mobilité, tels qu'un mouvement linéaire ou un point de cheminement aléatoire. Un grand nombre de modèles de mobilité ont été fournis avec INET et peuvent également être combinés pour obtenir des mouvements plus complexes.
- La variété des protocoles de routage pour les MANETs en générale, les protocoles de routage peuvent être classés comme réactifs ou proactifs. INET contient divers protocoles de routage pour les VANETs des deux catégories, parmi les protocoles de routage réactif (AODV) et proactif (DSDV).
- Dans les simulations inter-véhicules, le terrain a des effets profonds sur la propagation du signal. Par exemple, les véhicules situés de part et d'autre d'une montagne ne peuvent pas communiquer directement les uns avec les autres. Pour cela, la plateforme INET décrit un modèle au sol pour décrire la surface 3D du terrain.

Plusieurs plateformes sont développées dans OMNET++, par exemple CASTALIA, aussi peut simuler les réseaux sans files mobiles, mais seulement les périphériques intégrés de faible puissance, par exemple les réseaux de capteurs sans fil (WSN), pour cela, INET est la plateforme utilisée dans les scénarios de simulation VANETs.[51]

ICanCloud

ICanCloud est une plate-forme de simulation destinée à modéliser et à simuler des systèmes de cloud computing, qui est destinée aux utilisateurs qui traitent étroitement avec ce type de systèmes. L'objectif principal de cette plateforme est de prévoir les compromis entre coût et performances d'un ensemble d'applications exécuté dans un matériel spécifique, puis de fournir aux utilisateurs des informations utiles sur ces coûts. Cependant, elle peut être utilisé par un grand nombre d'utilisateurs, des utilisateurs actifs de base aux développeurs d'applications distribuées volumineuses.

Parmi ces avantages, l'un des principes de ces conception comme la possibilité de mener de grandes expériences. C'est pourquoi iCanCloud est support le language C ++. Grâce à cela, iCanCloud peut utiliser toute la mémoire disponible sur les machines exécutant les expériences, pour les machines 32 et 64 bits, parce qu'autre simulateur CloudSim par exemple ne peut gérer plus de 2 Go de mémoire dans des systèmes 32 bits à cause au fait qu'il est écrit en Java. Une extension du simulateur de réseau NS2, GreenCloud se concentre sur la

simulation des communications entre des processus s'exécutant dans un cloud au niveau des paquets. De la même manière que NS2, GreenCloud est écrit en C ++ et OTcl, ce qui est un inconvénient, vous devez utiliser deux langages différents pour mettre en œuvre une seule expérience, par contre ICanCloud est écrit seulement avec C++.

Parmi les problèmes du cloud computing est la consommation d'énergie des centres de données, même iCanCloud, il ne fournit pas de modèles de consommation d'énergie, bien que cela soit inclus dans les travaux futurs.

Aucun des outils de simulation de cloud existants (CloudSim, MDCSim et GreenCloud) possède une interface graphique complète, par contre, iCanCloud dispose d'une interface graphique simple qui permet aux utilisateurs de créer facilement des expériences.

L'inconvénient principale du plateformes existant (CloudSim, MDCSim et GreenCloud) est un principe de conception dans ICanCloud, les outils de simulation existes ne peuvent simulent qu'une expérience sur une seule machine à la fois, par contre, ICanCloud peut simuler une expérience sur un groupe de machine.[52]

L'intégration d'ICanCloud avec l'INET

Dans iCanCloud, le système réseau permet aux applications exécutées dans des machines virtuelles d'échanger des données via un réseau de communication. Afin de remplir cette tâche, la plateforme INET a été utilisé. Cette plateforme contient des modules permettant de simuler complètement un système réseau, il contient des protocoles réseau tels que TCP et UDP. Le principal avantage de cette méthode est le haut niveau de précision obtenu, car tous les éléments composant un réseau sont simulés. Par contre, le principal inconvénient est la performance, car ce niveau de détail élevé nécessite un calcul considérable de la puissance du processeur. Aussi, la structure INET fournit un ensemble de modules, tels que des routeurs, des commutateurs et des protocoles de réseau, permettant de créer un large éventail de réseaux. [52]

2.5.2 Le Simulateur NS-2

NS-2 (Network Simulator 2) est un simulateur d'événements discrets développé à ISI (Information Sciences Institute), en Californie. Le développement de NS2 commencé en 1989 en tant que variante du simulateur de réseau réel.

Le NS2 est un simulateur standard expérimentant des applications de réseaux câblés et sans fil, puisqu'il s'agit d'un outil open source et puissant pour la simulation de réseaux. Bien que ses premières versions s'adressent aux réseaux câblés, le support réseau sans fil est ajouté

ultérieurement par diverses extensions. Il suit de près le modèle OSI. C'est un simulateur orienté objet écrit en C ++ avec un interpréteur OTcl comme interface. NS2 utilise deux langages, C ++ et OTcl. Les utilisateurs implémentent leurs applications en C ++ et les scénarios de simulation et la configuration sont écrits en OTcl.[18]

NS-2 est composé de deux langages clés : C ++ et OTcl (Object-Oriented Tool Command Language). Alors que le C ++ définit le mécanisme interne (c'est-à-dire un backend) des objets de simulation, l'OTcl met en place la simulation en assemblant et en configurant les objets ainsi qu'en planifiant des événements discrets (c'est-à-dire une interface). Le C ++ et l'OTcl sont liés ensemble à l'aide de TclCL.[19]

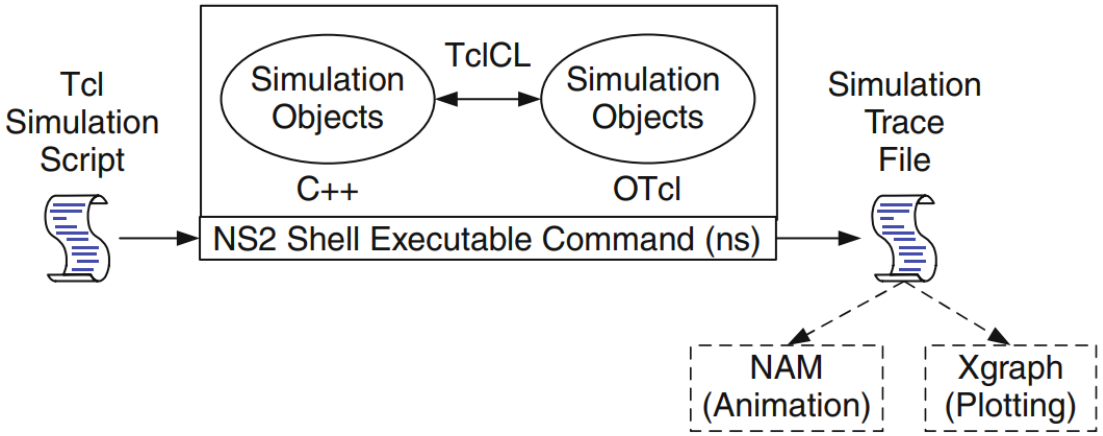

Figure 2-3 Architecture de base de NS-2 [19]

2.5.3 Le simulateur NS-3

NS-3 est un simulateur de réseau à événements discrets ciblant la recherche et l'utilisation à des fins éducatives (ns-3 Consortium 2015). NS-3 est un logiciel sous licence libre qui est publiquement disponible pour la recherche, le développement et l'utilisation. Le projet ns-3 a pour objectif de développer un environnement de simulation ouvert préféré pour la recherche en réseau ; il est conforme aux besoins de simulation de la recherche en réseau moderne et motive la contribution de la communauté.

Le développement de ns-3 a débuté en juillet 2006, il est entièrement écrit en utilisant le langage de programmation C ++. [21]

Un grand avantage a été mis sur la facilité de débogage et un meilleur alignement sur les langages actuels. Sur le plan architectural, cela a conduit l'équipe ns – 3 à s'éloigner du mélange de Tcl orienté objet et C ++, qui était difficile à déboguer et qui était inconnu de la plupart des étudiants. Au lieu de cela, la conception choisie visait à mettre l'accent sur les modèles purement C ++ pour la performance et la facilité de débogage, et à fournir une API

de script basée sur Python permettant l'intégration de ns – 3 avec d'autres environnements ou modèles de programmation basés sur Python. Les utilisateurs de ns–3 sont libres d'écrire leurs simulations en tant que programmes main C ++ ou programmes Python. L'API de bas niveau de ns – 3 est orientée vers l'utilisateur expérimenté.[22]

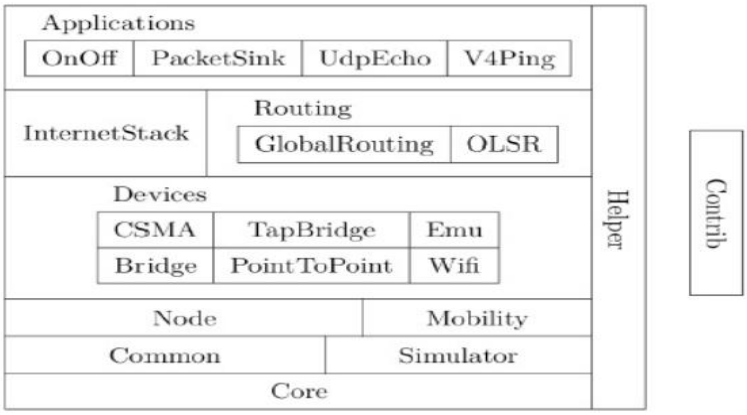

Figure 2-48 Architecture en couches du simulateur NS-3 [23]

2.5.3.1 Comparaison entre OMNET++ et NS-2

Après avoir décrit quelques simulateurs réseaux, on va faire une étude comparative entre quelques simulateurs les plus connus ; OMNET++, NS2 et OPNET.

- Premièrement, OMNET++ et NS2 sont des simulateurs open source, par contre, OPNET est commerciale.
- Aussi, NS2 et OPNET fournis seulement une interface pour le code source, mais OMNET++ comporte une IDE de simulation avec une autre interface de programmation.

Fonctionnalités	OMNET++ [1]	NS2 [2]	OPNET [3]
License	Open Source	Open Source	Commercialisé
Langage	C++	C++/OTCL	C++/Java
Interface Graphique	Bien	Faible	Excellent
Evolutivité	Moindre coût	Très coûteux	Coût moyen
Mobilité	Excellent	Faible	/
Modeles de simulation	Inet, Veins, MixMi, iCanCloud	Miracle	WIMAX-RBDS-Sim
popularité	Haute	Faible	Moyenne

Tableau 2. 1 Tableau comparative

2.5.4 Les Simulateurs de mobilité et de trafic routier

Comme il n'existe pas de modèle exact de flux de trafic en raison de sa grande complexité, les chercheurs tentent principalement de prédire le trafic à l'aide de simulations. Dans ce domaine, de nombreux logiciels de simulation existent et diffèrent par leur architecture logicielle ainsi que par les modèles décrivant le trafic lui-même. Parmi ces simulateurs, on distingue :

VanetMobiSim est une extension de **CanuMobiSim**, un simulateur de mobilité d'utilisateur générique. CanuMobiSim est un logiciel indépendant de la plate-forme et du simulateur, codé en SUN Java et qui génère des traces de mobilité pour les simulateurs de réseau suivants : ns-2, GloMoSim et QualNet. Il est capable d'intégrer des topologies de carte GDF (Fichier de données géographiques), contient une variété de modèles de mobilité et fournit une architecture de mobilité facilement extensible. CanuMobiSim, cependant, souffre d'un niveau de détail limité, ce qui le rend inapproprié pour la modélisation de la mobilité des véhicules. VanetMobiSim vise donc à étendre le support de mobilité des véhicules de CanuMobiSim à un niveau de réalisme supérieur. En étendant CanuMobiSim, VanetMobiSim hérite notamment de toutes ses fonctionnalités, mais contient également les nouvelles fonctionnalités suivantes : intégration des cartes TIGER, une caractérisation complète de la topologie routière, la modélisation des intersections, les capacités de dépassement, la gestion des feux de circulation. VanetMobiSim diffère également de CanuMobiSim par sa structure plus serrée autour d'un modèle spatial. [24]

En 2001, Le Centre aérospatial allemand (DLR) a lancé le développement du progiciel de simulation de trafic open source SUMO. SUMO est devenu une suite complète de services de modélisation du trafic, comprenant un réseau routier capable de lire différents formats de source, de générer des utilitaires de routage à partir de diverses sources d'entrée (matrices de destination d'origine, compteurs de trafic, etc.), une simulation hautes performances utilisable pour des jonctions uniques ainsi que pour des villes entières.

SUMO a commencé à être mis en œuvre en 2001, avec une première version open source en 2002. Il existe deux raisons pour rendre le travail disponible en open source. Le premier est le souhait de soutenir la communauté de simulation de trafic avec un outil gratuit dans lequel ses propres algorithmes peuvent être implémentés. Bien que certaines simulations de trafic open source soient disponibles, la plupart d'entre elles ont été implémentées dans une thèse de l'étudiant et ont été ensuite non prises en charge. La deuxième raison de rendre la simulation open source était le souhait d'obtenir le soutien d'autres institutions. [25]

SUMO est conçu pour simuler un réseau routier de la taille d'une ville. Comme la simulation est multimodale, ce qui signifie que non seulement les mouvements de voitures dans la ville

sont modélisés, mais également les systèmes de transport public sur le réseau routier, y compris les réseaux de trains alternatifs, la partie atomique de la simulation est un seul être humain. Cet être humain est décrit par une heure de départ et la route qu'il prend, qui est à nouveau constituée de subroutes décrivant une modalité de trafic unique.

Ainsi, une personne simulée peut prendre son véhicule jusqu'à la station du système de transport en commun la plus proche et poursuivre son voyage par un autre moyen de transport. Outre les mouvements utilisant des véhicules motorisés, une personne peut également marcher. La marche n'est pas simulée du tout, elle est modélisée en estimant le temps nécessaire à la personne pour atteindre la destination. SUMO contient les fonctionnalités suivantes :[26]

- Mouvement du véhicule sans collision
- Différents types de véhicules
- Rues à plusieurs voies avec changement de voie
- Règles de priorité de jonction (jonctions avec des rues ayant des priorités égales / différentes, par exemple, juste avant la gauche)
- Connexions voie par voie
- Une sortie XML brute contenant des informations sur l'état du réseau pour chaque pas de temps
- Détecteurs avec GnuPlot ou CSV (valeur séparée par des virgules) indépendants
- Entrée de fichiers XML pouvant être répartis sur plusieurs fichiers pour une meilleure gestion

Un autre simulateur de mobilité, **CORSIM** (*Corridor Similation*) qui consiste sur un ensemble intégré de deux modèles de simulation représentant l'ensemble de l'environnement de trafic. NETSIM représente le trafic dans les rues urbaines et FRESIM le trafic sur les autoroutes. Les modèles de simulation représentent les mouvements de véhicules individuels, qui incluent les influences du comportement du conducteur, Les effets de stratégies très détaillées, telles que le déplacement des gares routières ou la modification des restrictions de stationnement.[27]

PARAMICS ou (PARAllel computer MICroscopic Simulation) est une suite logicielle microscopique de simulation de trafic urbain et autoroutier utilisée pour modéliser le mouvement et le comportement de véhicules individuels sur des réseaux routiers. Au début des années 1990, Quadstone et SIAS Ltd., un cabinet d'ingénierie privé, ont mis au point une version de PARAMICS pour superordinateur Cray au Centre de calcul parallèle de l'Université d'Édimbourg. [28]

Paramics simule les intentions, les décisions et les actions ultérieures de chaque conducteur du réseau routier alors qu'il se dirige vers sa destination. Chaque conducteur choisit son meilleur itinéraire en fonction des caractéristiques de base du réseau et de sa connaissance des

embouteillages potentiels. À mesure qu'ils se déplacent sur le réseau, chaque conducteur donne la priorité à un ensemble de décisions sur la voie à utiliser, la vitesse à sélectionner, le moment opportun pour changer de voie et le moment opportun pour traverser ou fusionner avec le trafic dans une autre voie.[29]

Finalement, **CityMob**, qui est un générateur de modèle de mobilité spécialement conçu pour étudier différents modèles de mobilité dans les VANET et leur impact sur les performances de communication entre véhicules. **CityMob** crée des scénarios de mobilité urbaine et simule des voitures endommagées en utilisant le réseau pour envoyer des informations à d'autres véhicules, en essayant de prévenir les accidents ou les embouteillages.[30]

2.6. Conclusion

Dans ce chapitre nous avons montré quelques stratégies de dissémination des informations dans les VANETs Cloud, et aussi l'objectif de la simulation et les différents simulateurs dédies pour les réseaux Ad hoc et spécialement les VANETs. Ainsi, nous avons présenté quelques simulateurs de mobilité, pour la simulation de réseaux on va utiliser OMNET++ et pour la mobilité on va utiliser SUMO, et le simulateur Veins qui nous permettre de faire l'intégration entre SUMO et OMNET++.

Dans le chapitre qui suit, nous allons présenter la partie conception et réalisation de notre extension sous la plateforme iCanCloud.

CHAPITRE 03

Une plateforme Vanet-Cloud pour la Dissémination des Informations

1. L'Environnement De Simulation
2. Le Travail Proposée
3. Le Scenario D'Etude
4. La Vue Conceptuelle

3.1 Introduction

Nous avons présenté dans le chapitre précédent les stratégies de dissémination des informations dans les VANETs Cloud, ainsi, une étude de comparaison entre les simulateurs dédies pour ces environnements tel que Omnet++, Ns2, Opnet. En tenant compte du coût élevé du déploiement et de la mise en œuvre d'un tel scénario véhiculaire empirique, nous avons constaté que dans la plupart des travaux de recherche qui ont été proposé s'appuient sur des plateformes de simulations pour l'évaluation. Nous avons constaté également qu'avec le simulateur Omnet++, les chercheurs en ce domaine supportent ce modèle de simulation afin de juger la fiabilité de leurs contributions, telles que le test des nouveaux protocoles, de nouvelles techniques et de nouveaux algorithmes. Comme nous l'avons dit, cela est principalement dû au coût élevé du déploiement de ces systèmes dans des scénarios réels.

Ce chapitre traite avec l'un des composants clés des simulations véhiculaires, est un modèle réaliste de dissémination des informations conçu pour les Cloud Véhiculaire (VC). Ce modèle garantissant que les conclusions tirées des expériences de simulation seront appliquées à des déploiements réels.

Dans ce chapitre, un cadre conceptuel étendue sous Omnet++ est proposé pour modéliser et simuler les scénarios de dissémination des informations dans les architectures Cloud Véhiculaire (VC). En particulier, l'architecture que nous proposons est étendue par les composants clés de l'architecture ICanCloud [52] et la plateforme INET [51]. Les types de mobilité « *Stationnary Mobility* » et « *TraCI Mobility* » de la plateforme Veins [31] sont également impliqués. Ces modèles donnent l'accès à une simulation de circulation routière en cours d'exécution, comme ils permettent aussi de récupérer les valeurs des objets simulés et de manipuler leur comportement. Et pour la dissémination des informations dans scenarios CV, la proposition en enrichie par un modèle de simulation contrôlant le clustering, la création des réseaux virtuel, l'adaptation en cas des réseaux intermittent qui pouvant êtres se produit.

3.2 Environnement de simulation

La simulation des réseaux Cloud Véhiculaires (VC) nécessite un ensemble des extensions ou chaque extension fournis un service, dans notre contribution nous avons intégrant l'OMNET++ avec SUMO via l'extension Veins.

- L'objectif de **l'OMNET++** est d'assurer la simulation réseau, pour les services Cloud Computing nous avons utilisé la plateforme **ICanCloud** qui se base sur la plateforme INET.[53]
- **SUMO (Simulation of Urban Mobility)** est un outil de simulation de la circulation routière, conçue pour gérer et contrôler les trançon routiers. Il permet de modéliser les systèmes de trafic, y compris les véhicules ordinaires, le transport en commun, le passage des piétons, les tunnels, les zones ferroviaires, etc. il inclut également un ensemble d'outils qui gèrent les itinéraires, la visualisation, la digitalisation des cartes géographiques. SUMO a été amélioré avec des modèles personnalisés et englobe plusieurs API interactive avec d'autre simulateur, comme Omnet++ et Ns2, en contrôlant les comportements à distance. [26]
- Veins version 2.0 pour intégrer OMNET++ avec SUMO

Omnet++	La version 4.6
INET	La version 2.5
ICanCloud	La version 1.0
SUMO	La version 0.19.0
Veins	La version 2.0

Tableau 3. 1 Les versions d'outils utilisées

3.3 Travail proposé

3.3.1 Les problèmes de dissémination des informations dans les VANETs

Il existe aujourd'hui une quantité très importante d'informations utiles pour informer les conducteurs lorsqu'un événement survient (les accidents, les embouteillages, les travaux, les freinages d'urgence, les places de stationnement disponibles, la présence de radar de police, les véhicules d'intervention d'urgence). Ces événements peuvent être observées directement et en temps réel par le véhicule. Il est par ailleurs intéressant que toutes ces données soient

partagées entre les véhicules dans le cadre d'une « conduite collaborative » et autre de ces informations soient sauvegardées.

Pour ajouter une valeur importante à ces technologies et rendre ces réseaux plus utiles et plus serviable, des services cloud sont proposées ou les véhicules devient être capable de faire des calcule à l'aide d'une unité de calcule, sauvegarder des données et des informations dans un espace de stockage installé sur les véhicules, tous ces services rend les véhicules simples des véhicules Cloud, et pour assurer une communication fiable et efficace entre les véhicules, une interface DSRC est ajouté, cette interface assurer les communications RADIO, Alors, avec l'intégration du services Cloud, les réseaux VANETs simple devient des réseaux VANETs Cloud.

3.3.2 L'intégration du Cluster dans les réseaux VANETs Cloud

Dans les VANETs, les véhicules proches les uns des autres forment un cluster, chaque cluster a un Cluster Head, dans notre cas, nous utilisons les cluster heads pour les opérations de calcule, et au lieu d'avoir un seul cluster head, nous choisissons plusieurs membres parmi le cluster pour faire les calculs, le choix de ces membres se faite par les algorithme du groupes dominants (Dominating Sets Algorithm) , Etant donné que les réseaux VANET sont connus par leur mobilité très dynamique, la formation des clusters est une solution efficace pour maintenir les liens entre les véhicules. Mais en même temps, il sera difficile de garder la forme du cluster à cause de la vitesse élevé du nœuds, alors, l'application responsable de création du cluster doit être une application adaptative et tenez compte que le nœud qui sort d'un groupe doit entrer dans un autre s'il remplit les conditions.

3.3.3 Les applications mobiles adaptatives

Les communications sans fil et les appareils portables permettent aux utilisateurs mobiles d'avoir accès aux informations partout et à tout moment. Concevoir, mettre en œuvre et déployer des applications qui fonctionnent bien sur tous les périphériques portables et sur un large éventail de réseaux d'accès sans fil n'est pas une mince affaire, parce que les connexions réseaux dans les scénarios mobiles est caractérisée par une bande passante limitée, un taux d'erreur élevé, un coût plus élevé, par contre dans les VANETs, Les contraintes et les limites auxquelles sont confrontées les applications véhiculaires ne sont pas un produit de la technologie actuelle, mais elles sont naturellement liées à la mobilité, elle complique la conception d'applications et nécessite de repenser les approches traditionnelles de la conception d'applications, en d'autres termes, les applications doivent être adaptatives. Pour rendre notre application adaptative, un message personnel (Self Message) périodique sera

envoyée, après la réception du message, le module qui contrôle le réseau reformule les clusters.[54]

3.4 Le scenario d'étude

La diffusion d'informations du trafic dans un réseau VANET est un grand problème, plusieurs projet et études sont faites dans ce sujet, Les informations de trafic ont généralement un caractère orienté diffusion, En d'autres termes, les informations sur le trafic sont d'intérêt public et profitent généralement à un groupe d'utilisateurs plutôt qu'à un individu spécifique.
Par conséquent, Le réseau de radio cognitif est apparu comme la solution clé pour résoudre le problème de partage des informations. L'idée principale des réseaux de radio cognitive est de permettre aux utilisateurs de partager la même bande passante sans causer de brouillage, L'avantage principale d'un système de diffusion est qu'un véhicule n'a pas besoin de connaître une adresse de destination ni un itinéraire vers une destination spécifique [1].
Avec l'évolution du réseaux VANET et la différentiation des services fournis, l'intégration du « *Cloud Computing* » avec les réseaux véhiculaire conventionnel est devient primordiale pour offrir des services de stockage ou de calcul. Durant le déplacement, quelques informations de trafic doivent être échanger et sauvegarder entre les nœuds. Pour sélectionnés les nœuds d'échange ou de sauvegarde, nous essayons dans ce chapitre d'utiliser, tout d'abord, la technique du cluster afin de prévoir les groupes des véhicules partageants les mêmes informations du trafic ou celle qui forment les réseaux intermittents. Ces informations sont des informations de contrôles de la circulation comme les « *coordonnés* », la « *vitesse* », la « *densité* », le « *volume de trafic* », etc.
Après la formation des clusters, il faut choisir les ensembles dominant via l'application d'un « *algorithme dominant* » pour créer des réseaux véhiculaires virtuels. Les « *ensembles dominants* » assurent la sauvegarde des messages et exécutent les taches de calcule. La connexion entre les membres d'ensemble dominant produit un « *cluster virtuelle* » ou chaque nœud de ce cluster prend une fonction parmi les fonctions du cloud par exemple (un nœud de Storage, nœud de calcule, etc).

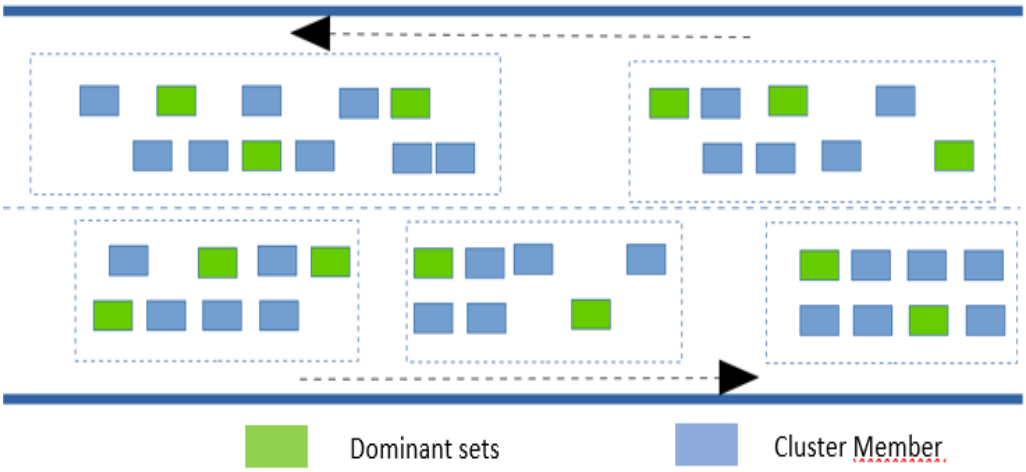

Figure 3- 1 L'architecture du cluster

3.5 La vue conceptuelle de l'intégration

L'idée de base de l'architecture proposée est de fournir aux véhicules des données de haute qualité, qui leur permettent et leur permettent d'adopter les réactions appropriées. Pour atteindre cet objectif, l'architecture du réseau Cloud Véhiculaire (CV) a donc été conçue sur la base de ce principe. La Figure 3.2 montre ainsi le diagramme de classe UML de l'architecture VANET-Cloud Le modèle repose sur deux types de nœuds : le « nœud véhiculaire » et « les stations de base » de la route (RSU). Ces deux appareils sont également équipés de divers systèmes intégrés, tels que le système de calcul et le système de stockage, et un réseau de capteur intégrer.

De cette manière, l'hyperviseur « hypvisor » présente le logiciel installé dans le RSU intelligent, qui vise à gérer pour chaque système d'unité « guest-on-board unit » une « Machine Virtuelle » pratique et / ou un « Station Virtuel »; et également destiné à exécuter les tâches entrants dans l'instance de machine virtuelle spécifique. Le système virtuel peut initier une machine virtuelle ou des stations virtuels, ce qui lui permet de communiquer avec les nœuds véhiculaires et les RSU intelligentes. D'autre part, le cloud central lance le plan de sécurité. En outre, chaque plan de sécurité peut conFigurer les tâches correspondantes qui seront chargées sur l'hyperviseur spécifique.

En outre, le noyau de l'architecture proposé est le Cloud Computing, qui consiste essentiellement sur la définition des entités de base telles que le véhicule, le RSU, la VM et l'hyperviseur, etc.

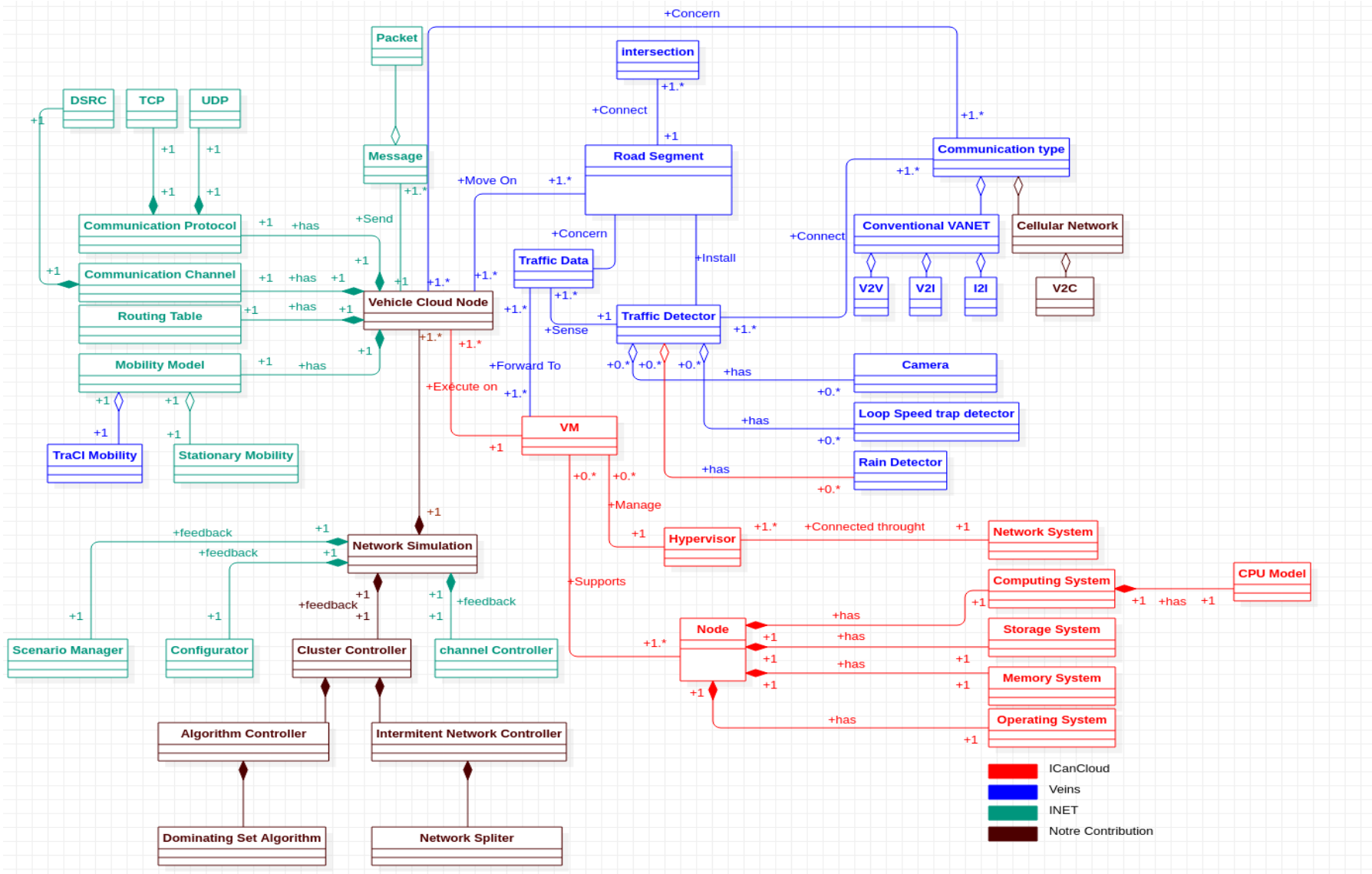

Figure 3- 2 Diagramme de Classe du Modèle

3.5.1 ICanCloud

Comme mentionné dans le chapitre précédent, iCanCloud [52] est une plateforme de simulation basé sur la plateforme INET pour la simulation d'architectures de Cloud Computing évolutives. L'un des objectifs principaux est de prédire le coût et les performances d'une application donnée dans le service cloud. En outre, l'idée de base de l'architecture proposée est de fournir aux véhicules des données précises pour adopter une réaction appropriée en temps réel. Pour modéliser et simuler les performances d'architectures Cloud Véhiculaires (CV), nous proposons d'étendre la plateforme de simulation iCanCloud [52] par les composants suivants :

- VM (Virtual Machine) : la *machine virtuelle* (VM) est définie dans le simulateur *iCanCloud* en tant que type de machine sans ressources physiques. Les ressources physiques sont gérées par l'hyperviseur et la machine virtuelle est liée à l'hyperviseur afin d'effectuer ces tâches. Les principaux *paramètres de configuration* définissant une machine virtuelle sont les suivants :

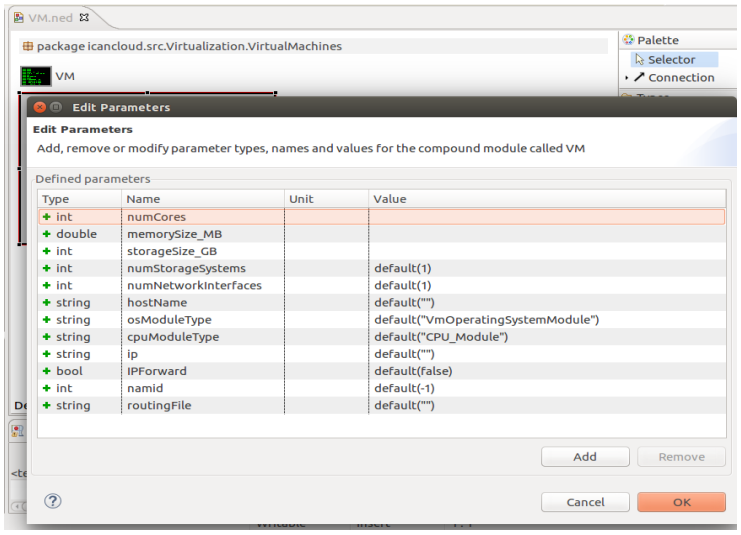

Figure 3- 3 Une Machine Vertuelle avec ces parametre

- Computing System : Module qui implémente un module de *CPU*, est module de base, complet et fonctionnel, comprenant tous les composants nécessaires pour le rendre fonctionnel pour un scénario de simulation. Il recevoir les entrées à partir d'autres modules, traiter les données et, avec les données traitées, collabore avec *l'hyperviseur* en vue de créer des informations pour le stockage et/ou pour la sortie.

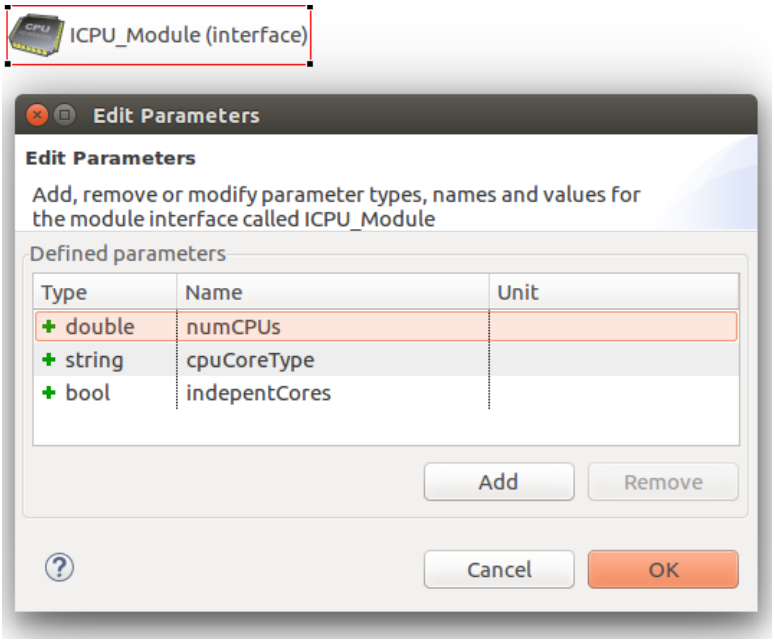

Figure 3- 4 Un Module de calcul Avec ces paramètres

- Storage System : Module qui implémente un système de stockage, est un module de simulation dans lequel les données sont stockées à distants et accessibles sur le réseau. Il est géré par l'hyperviseur en se basant sur les techniques de virtualisation pour exploiter les services de stockage en cloud. Ce type de stockage notamment appelé le stockage utilitaire, est un concept susceptible à la différenciation basé sur la mise en œuvre réelle et la prestation de service cloud.

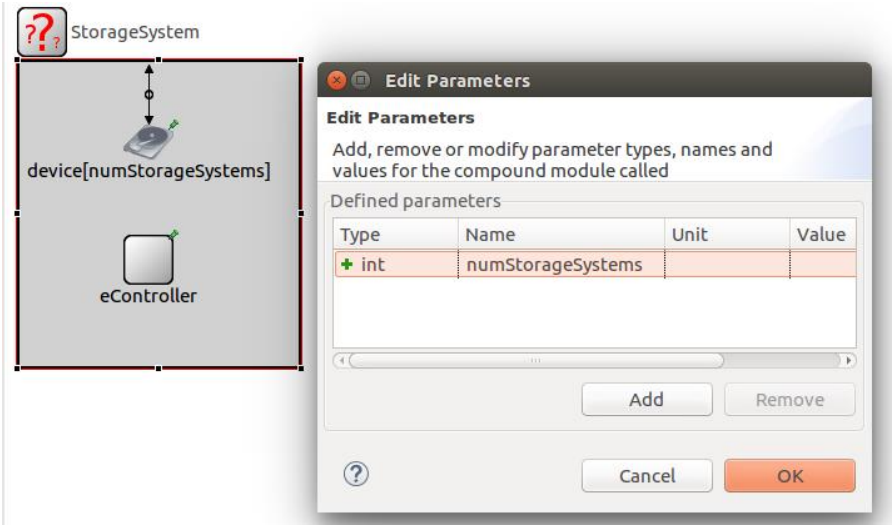

Figure 3- 5 Un Module de stockage avec ces paramètres

- Memory System : Module représentant une mémoire principale

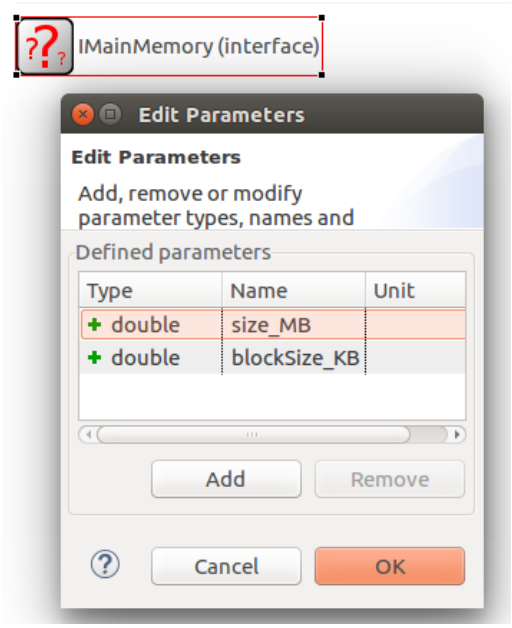

Figure 3- 6 Une Mémoire avec ces paramètres

- Operating System : Module qui implémente un module de système d'exploitation pour une machine physique ou virtuel.

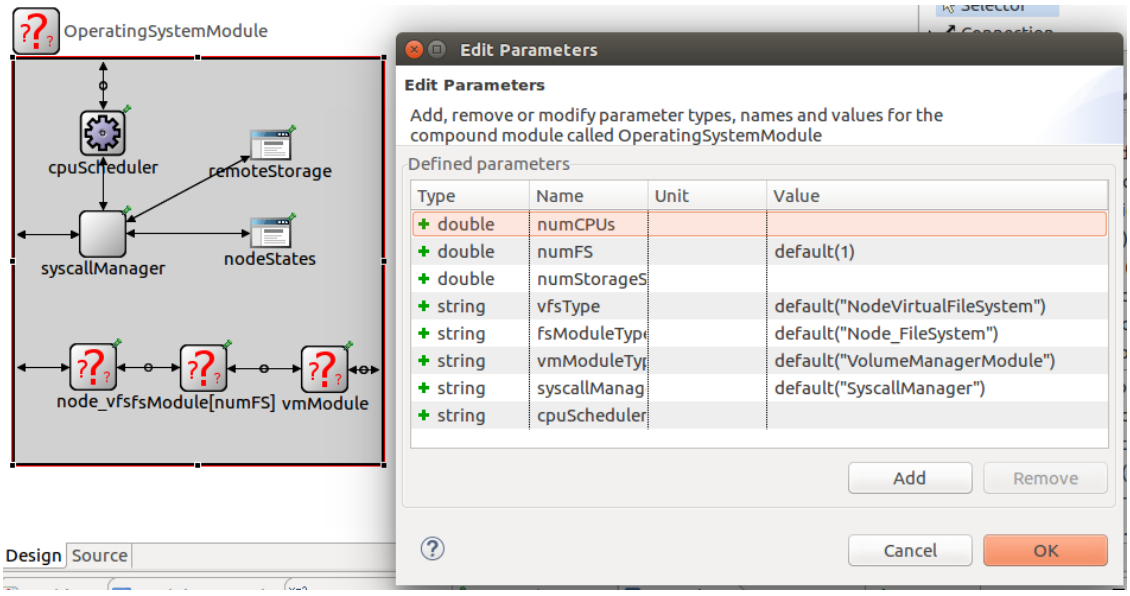

Figure 3- 7 Un Systeme d'Exploitation avec ces paramètres

- Hypervisor : Chaque machine virtuelle est liée dans ce module à tous les contrôleurs.

L'hyperviseur a quatre contrôleurs principaux. Chaque contrôleur est responsable d'une ressource physique : CPU, mémoire, réseau et stockage.

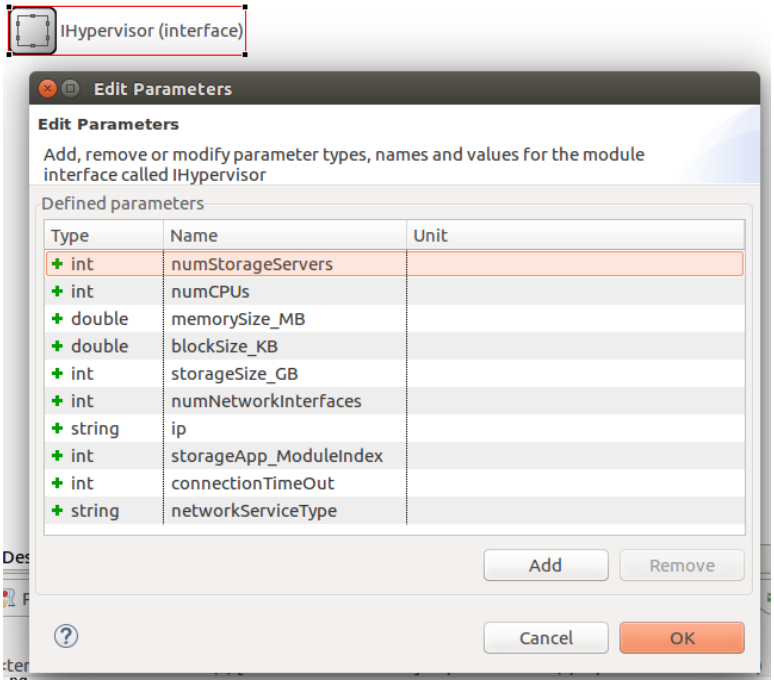

Figure 3- 8 L'hiperviseur

3.5.2 INET

- Channel Controller : ChannelControl a exactement une instance dans chaque modèle de réseau contenant des nœuds mobiles ou sans fil. Ce module est informé de l'emplacement et du mouvement des nœuds et détermine quels nœuds se trouvent dans la distance de communication ou d'interférence. Cette information est ensuite utilisée par les interfaces radio des nœuds lors des transmissions.

Doit être nommé comme "channelControl" à l'intérieur du réseau.

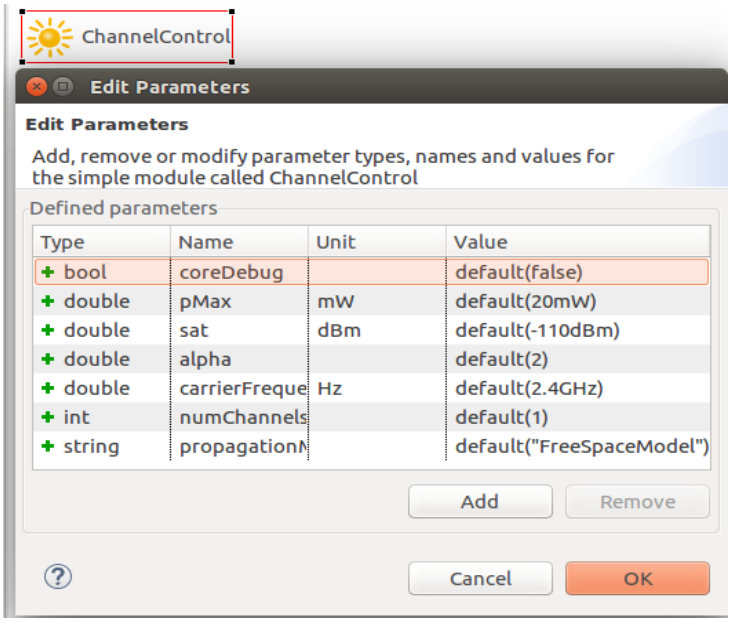

Figure 3- 9 Channel Control avec ces paramètres

- Scenarion Manager : ScenarioManager sert à conFigurer et à contrôler des expériences de simulation. Vous pouvez planifier la survenance de certains événements à des moments spécifiques, tels que la modification d'une valeur de paramètre, le taux d'erreur sur les bits d'une connexion,

la suppression ou l'ajout de connexions, la suppression ou l'ajout d'itinéraires dans une table de routage, etc., afin d'observer le transitoire. Comportement.

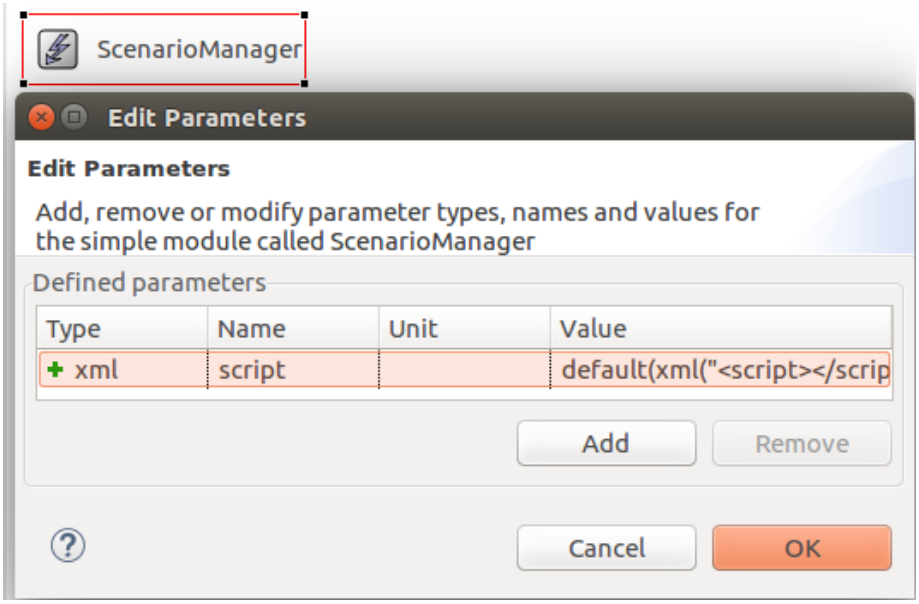

Figure 3- 10 Scenario Manager avec ces paramètres

- Configurator : Ce module attribue des adresses IP et conFigure le routage statique pour un réseau IPv4.

Il attribue des adresses IP par interface, s'efforce de prendre en compte les sous-réseaux et peut également optimiser les tables de routage générées en fusionnant les entrées de routage.

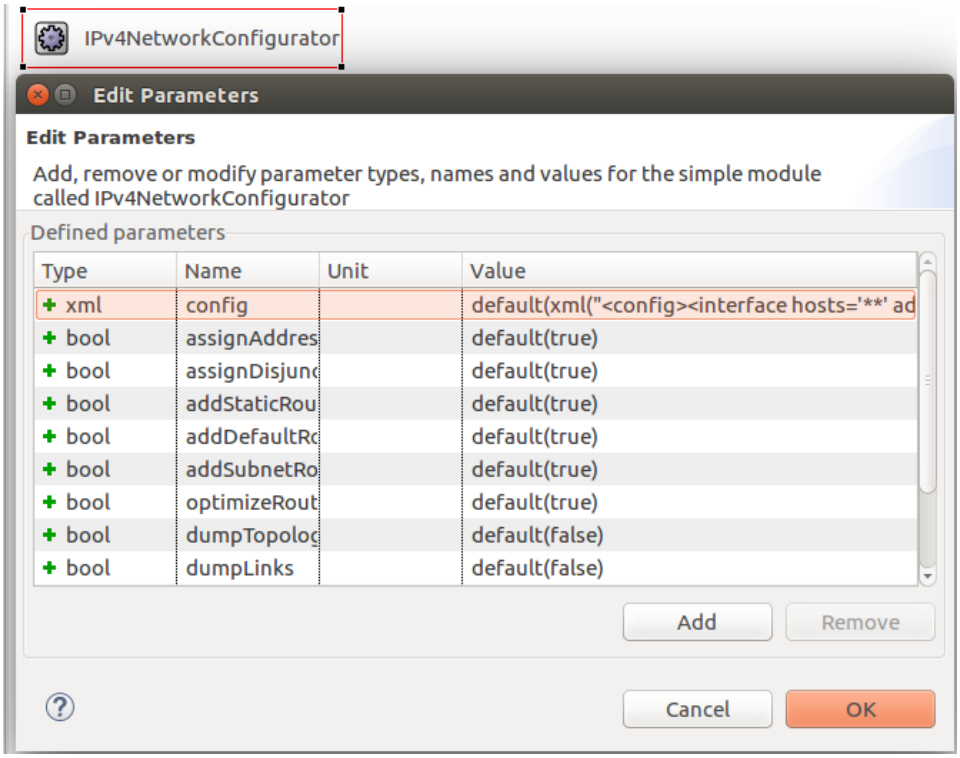

Figure 3- 11 Configurateur avec ces paramètres

- Mobility : Un modèle de mobilité décrit la position et l'orientation dans le temps dans un système de coordonnées euclidien 3D. Son objectif principal est de fournir des données de position, de vitesse et d'accélération, ainsi que de données de position angulaire, de vitesse angulaire et d'accélération angulaire, en tant que quantités tridimensionnelles au moment de la simulation en cours.

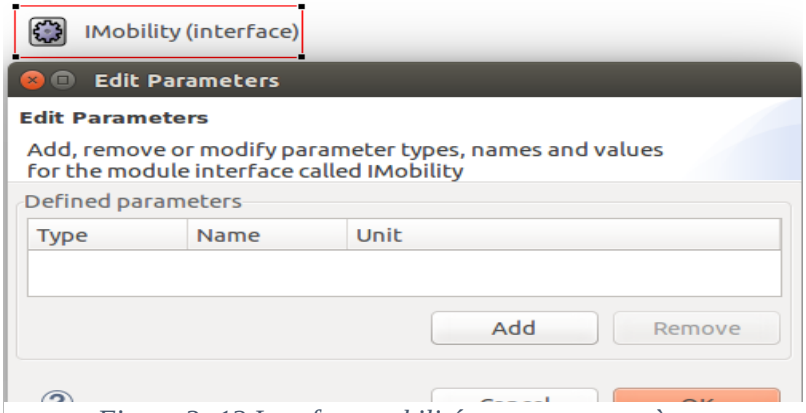

Figure 3- 12 Interface mobilité avec ces paramètres

- Routing Table : Ce module n'a pas de portes ; toutes les fonctionnalités sont accessibles via les fonctions de la classe de module C ++.

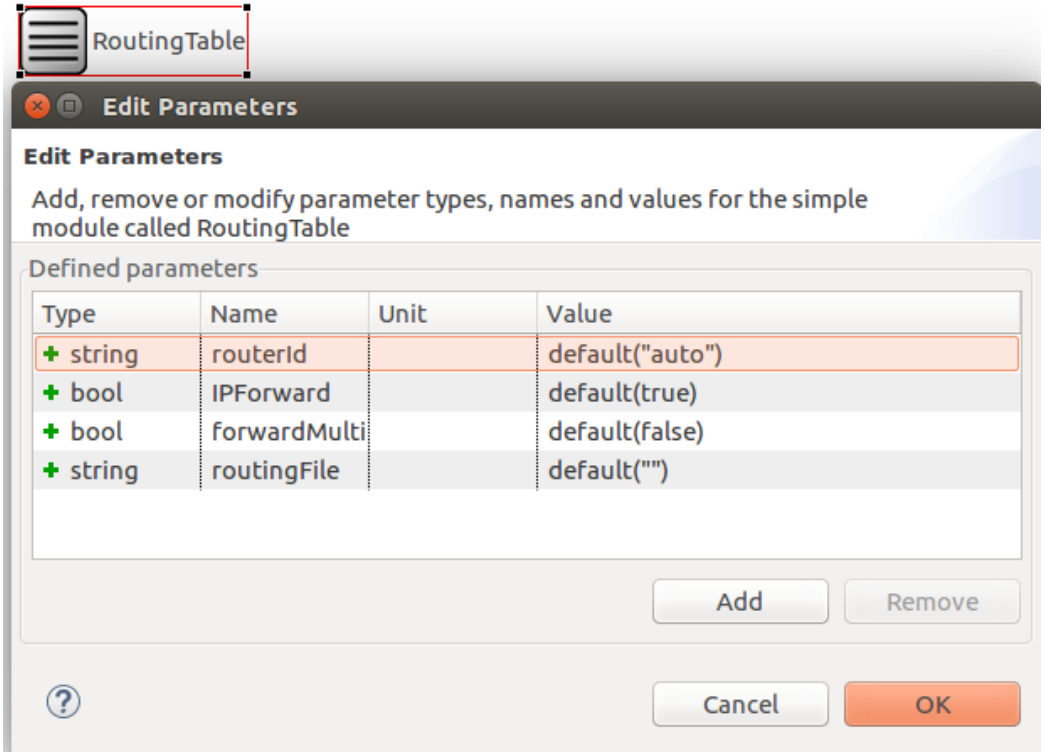

Figure 3- 13 Table de routage avec ces paramètres

3.5.3 Notre Contribution

- Vehicular Cloud Node : un nœud véhiculaire contient des services Cloud

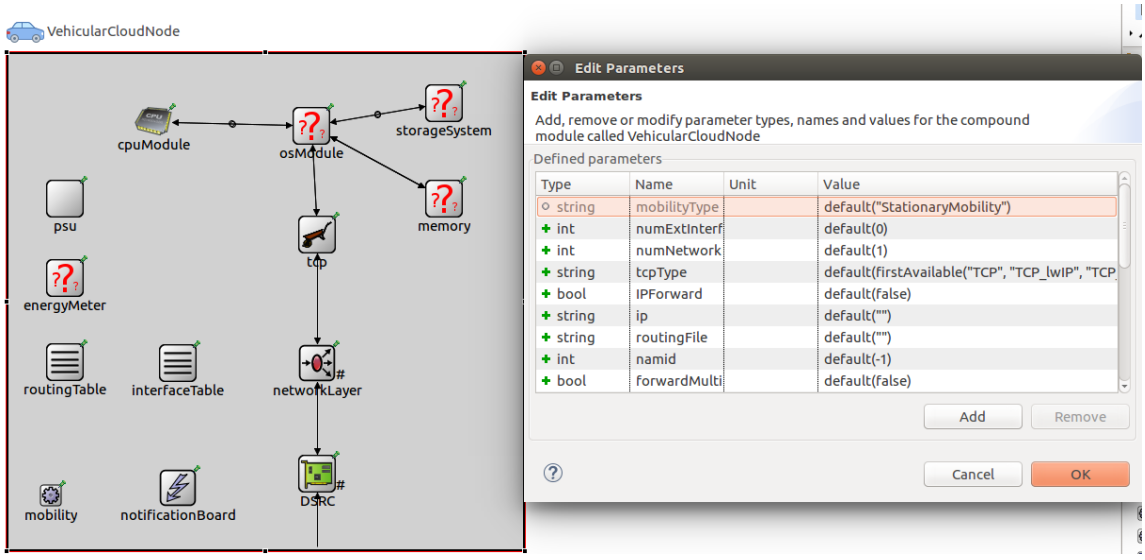

Figure 3- 14 un véhicule avec ces paramètres

- Cluster Controller : une class C++ qui décrivent le comportement du réseau

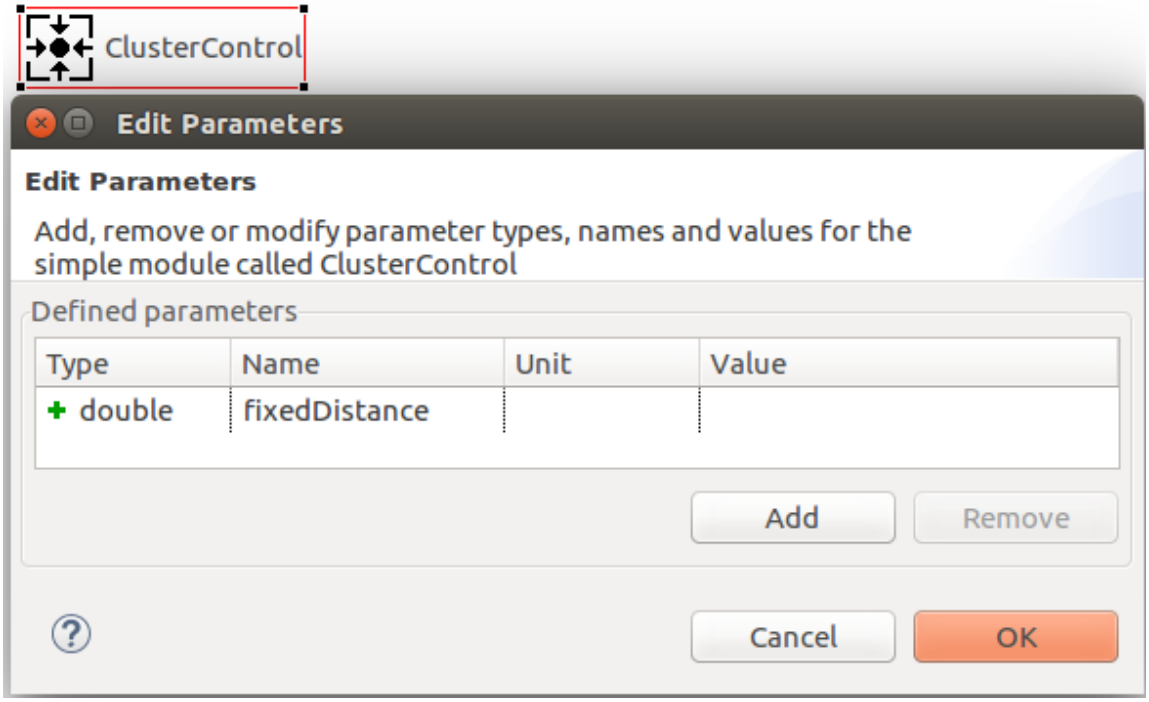

Figure 3- 15 le module qui gère le cluster avec ces paramètres

- Network Splitter : La fonction qui divise le réseau
- Dominating Sets Algorithm : L'algorithme de choix de groupe dominant

3.5.4 Les Algorithmes

3.5.4.1 Dominating Set Algortithm

```
variables i,j : entier
  vis[100011] : booléens
      graph : Tableau[Tableau [entier]]


        Tableau(entier) solve_dominant(nœud entier,edge entier)

          bébut
              Tableau(entier) S;
              pour i allant de 0 à nœud faire
              début
                    Si (vis[i]= faux)
                    début Si
                      S←S+i
                      vis[i]←vrais;
                      pour j allant de 0 à graph[i].longueur() faire
                          début
                           Si (vis[graph[i][j]]= faux)
                             début
                                                          vis[graph[i][j]]←vrais
                                    sortie de boucle
                             fin Si
                                       j←j+1
                       fin pour
                     fin Si
                   i←i+1
              fin pour
              renvoyer S;
          Fin
```

3.5.4.2 Intermittent Sets Algorithm

```
tableau[conteneur [int,int]] Splliter( arcs conteneur[int,int])
début
Variables v1,v2,NbNetworks←0,com←0 : entier
                                        ListNet : conteneur [entier,entier]
                                itr,itr2,it : itérateur de conteneur [int, int]
                                        NetVec : tableau[conteneur [int,int]]
                                exist←false : booléens
ListNet←arcs

Tantque(ListNet.longueur<>0)
début
        com←0;
        variable subNet : conteneur[entier,entier]
        NbNetworks← NbNetworks+1

        pour itr allant de LisNet.Premier à ListNet.Fin faire
        début
                v1←itr.premier;
                v2←itr.deuxieme;
                Si(com == 0){
                        début
                        subNet.ajouter(v1,v2)
                        ListNet.supprimer(itr)
                        com←com+1;
                        sortie de boucle

                        sinon
                                début
                                variables
                                pour it allant de  subNet.premier à  subNet.fin faire
                                        début
                                        Si((it.Premier )=v1 ou (it.deuxieme)=v2 ou
                                                        (it.Premier=v2) ou (it.deuxieme)=v1)
                                        début
                                                exist←vrais;
                                                Si(exist=vrais)
                                                debut
                                                        subNet.Ajouter(v1,v2)
                                                        ListNet.Supprimer(itr)
                                                        sortie de boucle
                                                fin Si
                                        fin Si
                                fin pour
                        fin Sinon
                fin Si
        itr←itr+1
        fin pour
        pour itr2 allant de  subNet.premier à  subNet.fin faire
        NetVec.ajouter(subNet);

fin Tantque
renvoyer NetVec;
Fin
```

3.6 Conclusion

Dans ce chapitre, nous avons proposé un scénario de dissémination des informations dans un environnement VANET Cloud on se basant sur la diffusion des informations via une canal radio et en diffusant les informations du trafic en temps réel. La diffusion faites d'un nœud vers tous les autres nœuds de réseau, l'extension proposé fournis la possibilité de simuler deux types de services du trafic routière ; la clusterisation et les nœuds véhiculaires équipées par des services cloud, les clusters sont définis par les Dominating Sets Algorithm et les services cloud sont héritées depuis la plateforme ICanCloud.

Dans le chapitre suivant nous allons présenter le projet OMNNET++, ainsi l'implémentation des différents algorithmes et la simulation.

CHAPITRE 04

Implémentation Et Simulation

1. Créer Une Simulation
2. L'Implémentation
3. Le Scenario De Simulation
4. La Simulation

4.1 Introduction

Le chapitre précédent à été consacré par la conception de la plateforme proposé en présentant leur caractéristiques, ces techniques de communication, ainsi l'implémentation des scénazroi VANET Cloud. Dans ce chapitre, nous allons mettre en évidence la simulation de la proposition en décrivant un scénario spécifique pour la dissémination des informations. Ensuite, nous allons discuter les résultats obtenus.

4.2 La Simulation

4.2.1 Le projet Vehicular Clouds

Démarrez l'IDE OMNeT ++ en tapant omnetpp dans votre terminal. Une fois dans l'IDE, *choisissez : New -> OMNeT ++ Project* dans le menu.

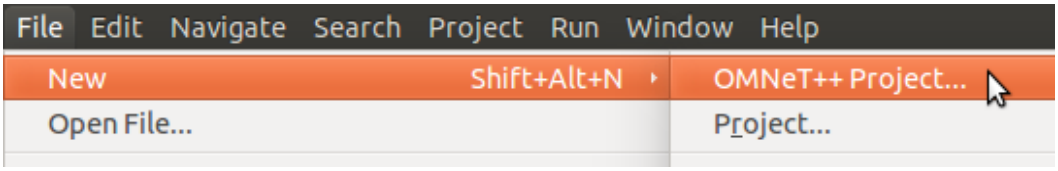

Figure 4- 1 Démarage d'un projet

4.2.2 Data Dissemination in VANET Cloud (DDVC)

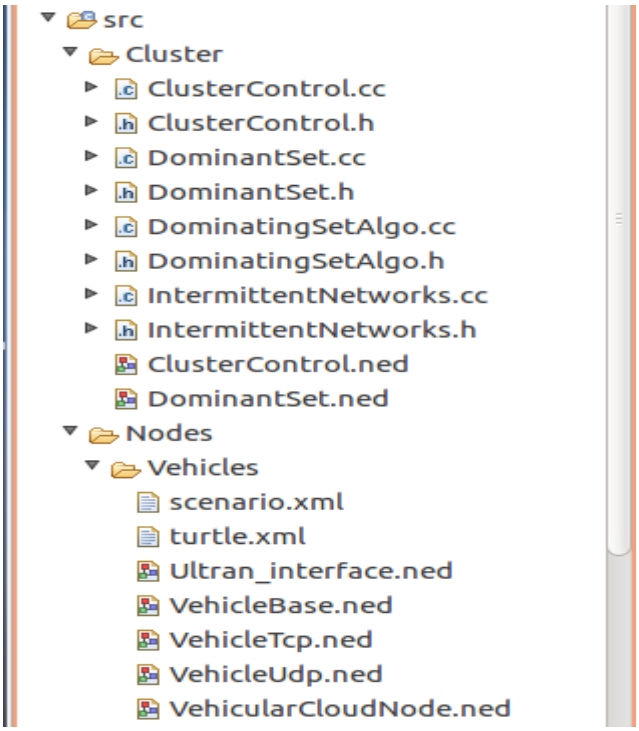

Figure 4- 3 les dossier Src

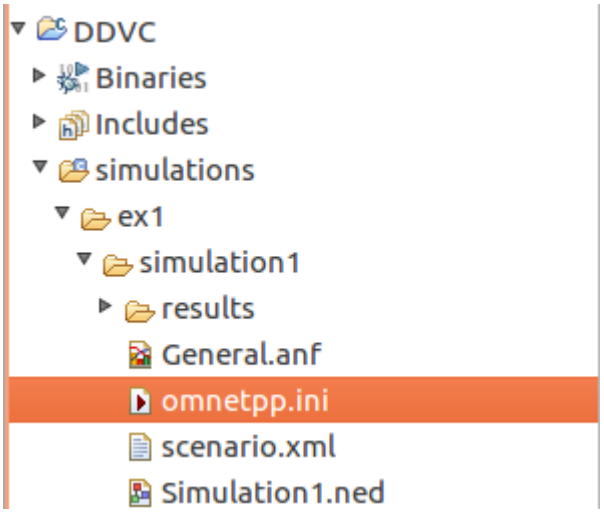

Figure 4- 2 Le Dossier Simulation

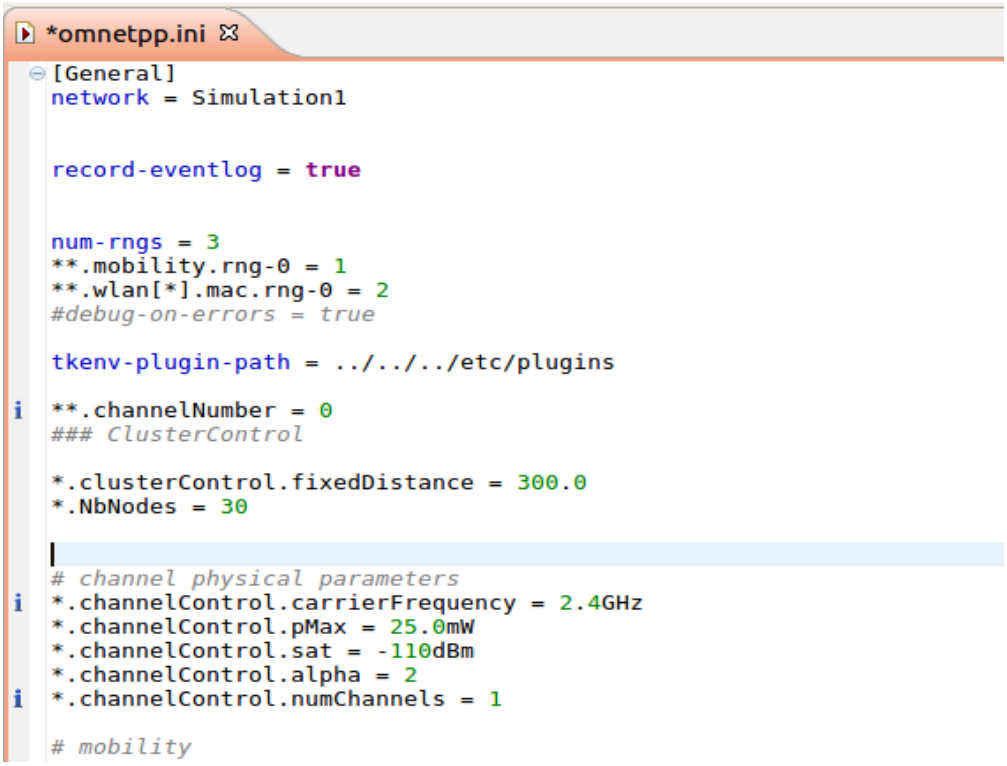

Figure 4- 4 le fichier Omnetpp.ini

Notre projet contient deux conteneurs essentiels seront ouvrir, *src* et *simulation*, le dossier *simulation* contient un fichier de configuration omnetpp.ini, ce dernier prenant la configuration du topologie (package.ned) voir (Figure 4-4), la configuration est concernant les valeurs soient par défaut soient les valeurs après un temps prédéfini d'exécution, voir (Figure 4-2). Le dossier *src* comprenant deux dossiers aussi ; Cluster et Node voir (Figure 4-3), dans le dossier cluster nous avons définis les modules de calculs *ClusterControl.ned* et son comportement. c++/.h, le dossier Nodes prend tous les nœuds véhiculaire à partir du nœud de base jusqu'à le nœud VehicularCloudNode.ned.

4.2.3 Construire et exécuter des simulations

Un modèle OMNeT ++ comprend les parties suivantes :

- Description de la topologie NED (fichiers. ned) : décrivant la structure du module avec paramètres, portes, etc. Les fichiers NED peuvent être écrits à l'aide de n'importe quel éditeur de texte, mais OMNeT ++ IDE fournit un excellent support pour l'édition graphique et l'édition de texte bidirectionnelle.

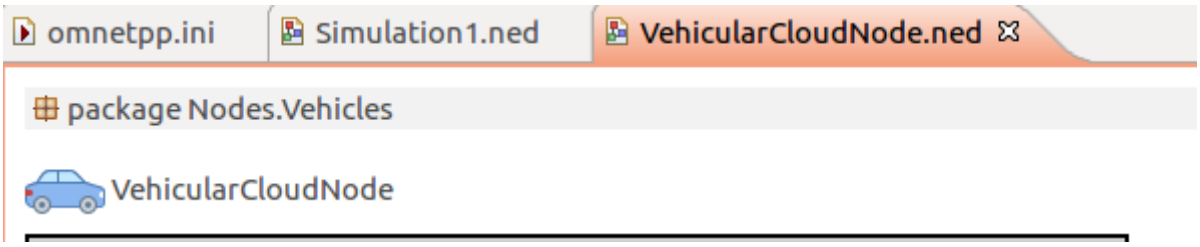

Figure 4- 5 Fichier. NED

- Définitions de message (fichiers .msg) : permettant de définir des types de message

Figure 4- 6 fichier .h/.c++

et d'y ajouter des champs de données. OMNeT ++ traduira les définitions de message en classes C ++ complètes.

-Sources de module simples : Ce sont des fichiers C ++, avec le suffixe .h / .cc.

- Noyau de simulation : Celui-ci contient le code qui gère la simulation et la bibliothèque de classes de simulation. Il est écrit en C ++.

- Les interfaces des utilisateurs : Les interfaces utilisateur OMNeT ++ sont utilisées dans l'exécution de la simulation pour faciliter le débogage, la démonstration ou l'exécution par lots de simulations. Ils sont écrits en C ++.

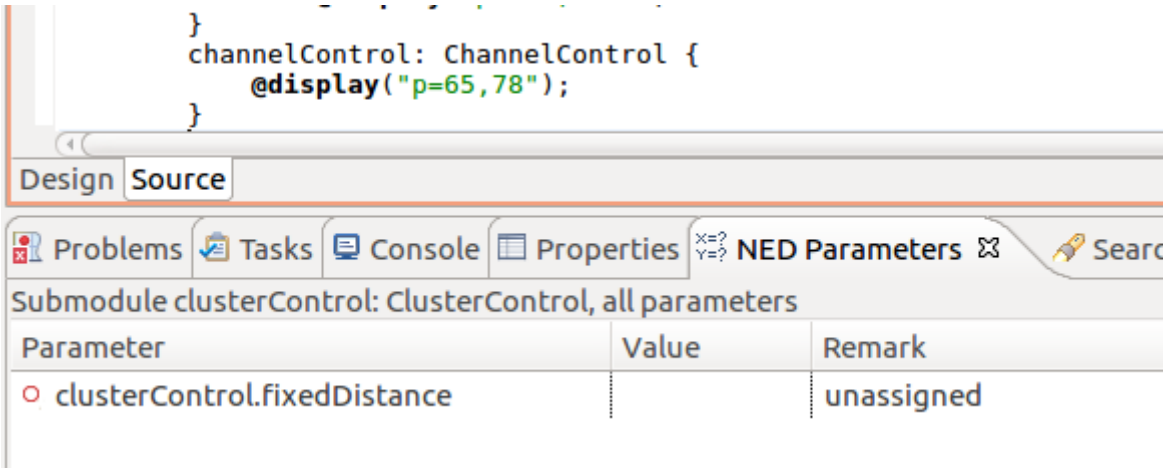

Figure 4- 7 L'interface Source et Design

- Omnet.ini : Pour pouvoir exécuter la simulation, il faut créer un fichier omnetpp.ini. omnetpp.ini contient des paramètres qui contrôlent l'exécution de la simulation, des valeurs pour les paramètres du modèle.

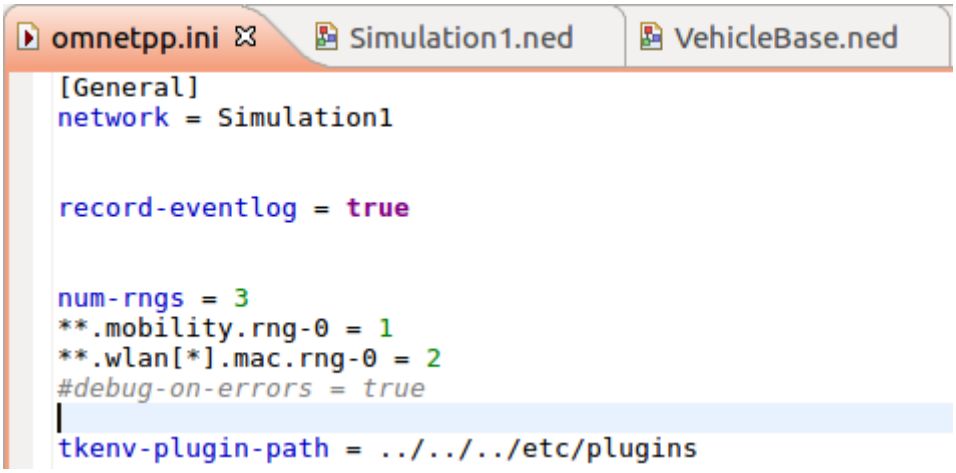

Figure 4- 8 le fichier omnetpp.ini

4.3 L'implémentation

4.3.1 Dominating Sets algorithms

```
std::vector<int> DominatingSetAlgo::solve_dominant(int n,int e){

    std::vector<int> S;
      for(i=0;i<n;i++)
      {
          if(!vis[i])
          {
              S.push_back(i);
              vis[i]=true;
              for(j=0;j<(int)graph[i].size();j++)
              {
                  if(!vis[graph[i][j]])
                  {
                      vis[graph[i][j]]=true;
                      break;
                  }
              }
          }
      }
      return S;

}
std::vector<int> DominatingSetAlgo::MainAlgo(std::map <int,int> setArcs){
      std::vector <int> VerVec = VerticesVector(setArcs);
      std::map <int,int> NewSubNet = NodesTransformation(setArcs);
      n = NbVertices(setArcs);
      e = setArcs.size();
      graph.resize(n);
      memset(vis,0,sizeof(vis));
      int tab1[e],tab2[e],j=0;
      std::map <int,int>::iterator itr; int v1,v2;
      for (itr = NewSubNet.begin(); itr != NewSubNet.end(); )
      {
            v1=  itr->first;     tab1[j] = v1;
            v2=  itr->second;   tab2[j] = v2;
            j++;itr++;
      }
      for(i=0;i<e;i++)
      {
            x=tab1[i];y=tab2[i];
            graph[x].push_back(y);
            graph[y].push_back(x);
      }
      std::vector<int> S = solve_dominant(n,e);
      std::vector<int> D;
      EV<<"The required Dominant Set is as follows:\n";
      for(i=0;i<(int)S.size();i++)
      {
            EV<<VerVec[S[i]]<<" "; D.push_back(VerVec[S[i]]);
      }
      EV<<endl;
      return D;
}

int DominatingSetAlgo::NbVertices(std::map <int,int> net){
      std::map <int,int> net1 = net; int v1,v2;
      std::vector <int> vec;
```

```
	std::map <int, int>::iterator itr;
	for (itr = net1.begin(); itr != net1.end(); itr++)
	{
		v1 = itr->first ; v2 = itr->second;
		if( std::find( vec.begin() ,vec.end(), v1) ==vec.end())
				vec.push_back(v1);
		if( std::find( vec.begin() ,vec.end(), v2) ==vec.end())
				vec.push_back(v2);
	}
	return vec.size();
}
std::vector <int> DominatingSetAlgo::VerticesVector(std::map <int,int>
net){
	std::map <int,int> net1 = net; int v1,v2;
	std::vector <int> vec;
	std::map <int, int>::iterator itr;
	for (itr = net1.begin(); itr != net1.end(); itr++){
		v1 = itr->first ; v2 = itr->second;
		if( std::find( vec.begin() ,vec.end(), v1) ==vec.end())
				vec.push_back(v1);
		if( std::find( vec.begin() ,vec.end(), v2) ==vec.end())
				vec.push_back(v2);
	}
	return vec;
}
std::map<int,int> DominatingSetAlgo::NodesTransformation(std::map <int,int>
subNet){
	int nbVertices = NbVertices(subNet);
	std::map <int,int> oldMap,newMap;
	std::map <int,int>::iterator itr;
	std::vector <int> VerVec = VerticesVector(subNet);
	oldMap = subNet;
	int v1,v2;
	for (itr = oldMap.begin(); itr != oldMap.end(); itr++){
		v1 = itr->first ; v2 = itr->second;
		v1 = getVerticeIndex(VerVec,v1);
		v2 = getVerticeIndex(VerVec,v2);
		newMap.insert({ v1, v2 });
	}
	return newMap;
}
int DominatingSetAlgo::getVerticeIndex(std::vector <int> VerticesVec, int
element){
	int j = -1;
	for(int i=0; i<=VerticesVec.size()-1; ++i)
	{
		if(VerticesVec[i]==element)
		{
			return i;
		}
		else if(VerticesVec[i]!=element && i==VerticesVec.size()-1)
		{
			EV<<"That name is not an element in this vector"<<endl;
		}
		else
		{
			continue;
		}
	}
	return j; }
```

4.3.2 Intermittent Network Algorithmes

```
std::vector<std::map <int,int>> IntermittentNetworks::Splliter(std::map
<int,int> arcs){
	std::map <int,int> ListNet = arcs;
	std::map <int, int>::iterator itr; int v1,v2,com=0;
	std::map <int,int>::iterator it;
	while(ListNet.size()!=0){
		com=0;
		std::map <int,int> subNet;
		this->NbNetworks= this->NbNetworks+1;
		for (itr = ListNet.begin(); itr != ListNet.end(); itr++) {
			v1=  itr->first;
			v2=  itr->second;
			if(com == 0)
			{
				subNet.insert({ v1, v2 });
				ListNet.erase(itr);
				com=com+1;
			}
			else
			{
				bool exist = false;
				for (it = subNet.begin(); it != subNet.end(); it++)
				{
					if((it->first)==v1 || (it->second) == v2 ||
					(it->first==v2) || (it->second) == v1)
					exist = true;

					if(exist==true)
					{
						subNet.insert({v1,v2});
						ListNet.erase(itr);break;
					}
				}
			}
		}
		std::map <int, int>::iterator itr2;
		EV<<"subNet:  "<<endl;
		for (itr2 = subNet.begin(); itr2 != subNet.end(); itr2++)
			EV<<" "<<itr2->first<<" , "<<itr2->second <<endl;

		NetVec.push_back(subNet);
		}
	return NetVec;
}
```

4.3.3 Cluster Controller

```
void ClusterControl::handleMessage(cMessage *msg)
{
	if (strcmp("getCurrentPosition", msg->getName())==0)
	{
		cTopology *Net = new cTopology("Simulation1");
		std::vector<std::string> ListNedFiles;
		ListNedFiles.push_back(getModuleByPath("Vehicle[]")
		->getNedTypeName());
		Net->extractByNedTypeName(ListNedFiles);
		std::vector <Coord> PosVec(Net->getNumNodes());
		float DistanceMatrix[Net->getNumNodes()][Net->getNumNodes()];
		for (int i = 0; i < Net->getNumNodes(); i++)
		{
			cModule *nd = Net->getNode(i)->getModule();
			if (ev.isGUI()) nd->getDisplayString().setTagArg("i",1,"");
				for (int s = 0; s < nd->getNumParams(); s++)
				{
					cPar &p = nd->par(s);
				}
			IMobility *mobility = check_and_cast<IMobility *>(
			nd->getSubmodule("mobility"));
			PosVec[i] = mobility->getCurrentPosition();
		}
		for (int a = 0; a < Net->getNumNodes(); a++)
		for (int b = 0; b < Net->getNumNodes(); b++)
			DistanceMatrix[a][b]=0.0;
		for (int a = 0; a < Net->getNumNodes(); a++)
		for (int b = a+1; b < Net->getNumNodes(); b++)
		{
			EV<<"The distance between V"<<a<<" and V"<<b<<" = "<<
					PosVec[a].distance(PosVec[b])<<endl;
			float dist =PosVec[a].distance(PosVec[b]);
			DistanceMatrix[a][b]=dist;
			DistanceMatrix[b][a]=DistanceMatrix[a][b];
		}
		EV << "----------------Start MAtix-----------------"<<endl;
		int NbMatrixNodes = Net->getNumNodes();
		for (int a = 0; a < NbMatrixNodes; a++)
		{
			for (int b = 0; b < NbMatrixNodes; b++)
			EV << DistanceMatrix[a][b] <<"  ";
		EV <<endl;
		}
		EV << "----------------End MAtix-----------------"<<endl;
		// constuire le graph
		int NbNodes = Net->getNumNodes();
		std::map <int,int> arc;
		EV<<"---------- MAP ---------------"<<endl;
		int NbArc = 0;
		for (int a = 0; a < NbNodes; a++)
		for (int b = a+1; b < NbNodes; b++)
		{
			if(DistanceMatrix[a][b]<=fixedDistance)
			{
				std::map <int, int>::iterator it;bool
				exist = false;
				for (it = arc.begin(); it != arc.end(); ++it)
				if(it->first==a && it->second == b)
				exist = true;
				if(!exist)
				{
					arc.insert({ a, b });
					NbArc = NbArc+1;
					EV<<"(V"<<a<<" , V"<<b<<")"<<endl;
				}
```

```
                }

            }
        // intermittent networks
        IntermittentNetworks in;
        std::vector<std::map <int,int> > NetVec = in.Splliter(arc);
        EV<<"Nb networks in arc = "<<NetVec.size()<<" =
                        "<<in.NbNetworks<<endl;
        for(int i=0;i<Net→getNumNodes();i++)
        {
                cModule *nd = Net->getNode(i)->getModule();
                if (ev.isGUI())
                {
                        nd→getDisplayString().setTagArg("i",1,"");
                }
        }
                std::string NodeColor1[] = {"#FFFF00","red","blue",
                                "green",
        "#4D4D4D","#FF8500","maroon","black"};
        std::string NodeColor2[] ={"#FFFFB3","#FF9999","#9999FF"
                        ,"#99FF99","#BFBFBF","#FFC280","maroon","black"};

        for(int i=0;i<NetVec.size();i++)
        {
                std::map <int,int> IntermittentNet = NetVec[i];
                if(IntermittentNet.size()>0)
                {
                        DominatingSetAlgo ds;
                        std::vector <int> S = ds.MainAlgo(IntermittentNet);
                        std::vector <int> subNet =
                                        ds.VerticesVector(IntermittentNet);
                        // change color
                        for(int j=0;j<(int)S.size();j++)
                        {
                                cModule *nd = Net->getNode(S[j])->getModule();
                                if (ev.isGUI())
                                {
                                        nd->getDisplayString().setTagArg
                                        ("i",1,NodeColor1[i].c_str());
                                }
                        }
                        for(int j=0;j<(int)subNet.size();j++)
                        {
                                if( std::find( S.begin(), S.end(),subNet[j]) ==
                                                        S.end())
                                {
                                        cModule *nd = Net->getNode(subNet[j])
                                        ->getModule();
                                        if (ev.isGUI())
                                        {
                                                nd->getDisplayString().setTagArg

                                                ("i",1,NodeColor2[i].c_str());
                                        }
                                }
                        }
                }
        }
        cMessage *Msgz = new cMessage("getCurrentPosition");
        scheduleAt(simTime()+0.2, Msgz);
    }
}
```

4.4 Un scenario de simulation

Après avoir ouvert OMNET++, Vous pouvez construire une simulation par l'utilisation du palette d'OMNET, la palette OMNET contient les déférents module du plateforme ICanCloud et INET et les autres plateformes référencé (MiXiM, Castalia, etc),

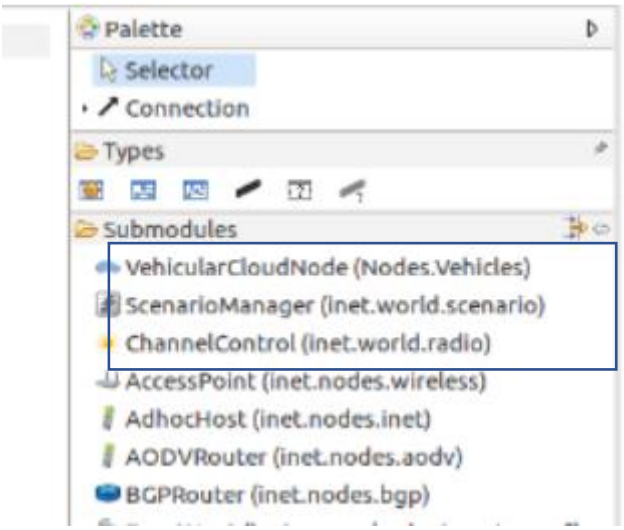

Figure 4- 9 Palette OMNET++

La topologie (Simulation1) est composée de nombreux modules simple et composé, les nœuds véhiculaires, ClusterControl, et autres nœuds de la plateforme INET, channelControl, configurator et scenarioManager.

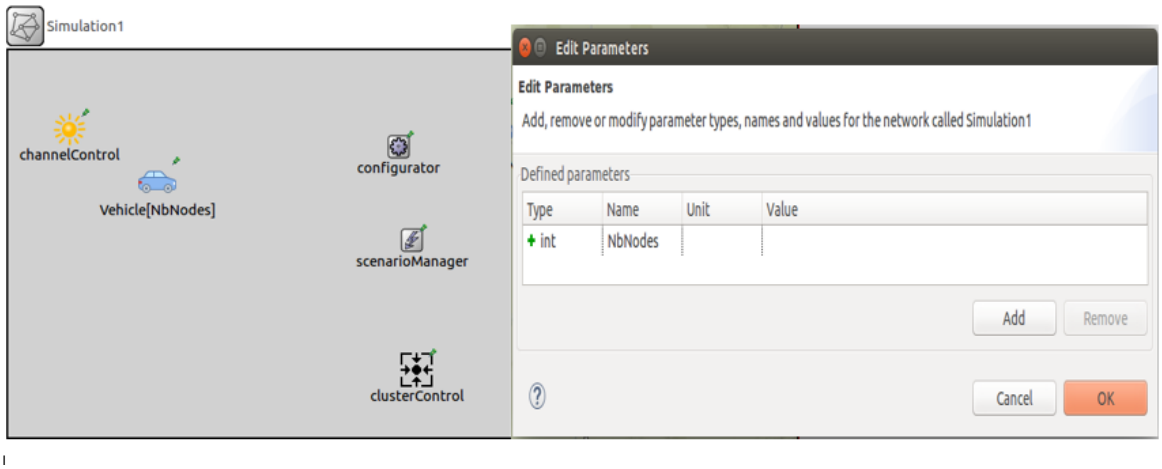

Figure 4- 10 La topologie de notre projet

Le véhicule est un module composé d'un ensemble de modules soient simple ou composé, une partie du Cloud Computing extrait d'iCanCloud ; osModule, StorageSystem, memory, et cpuModule, et autre module d'INET ; tcp, networkLayer, DSRC, mobility, routingTable, interfaceTable et energyMeter.

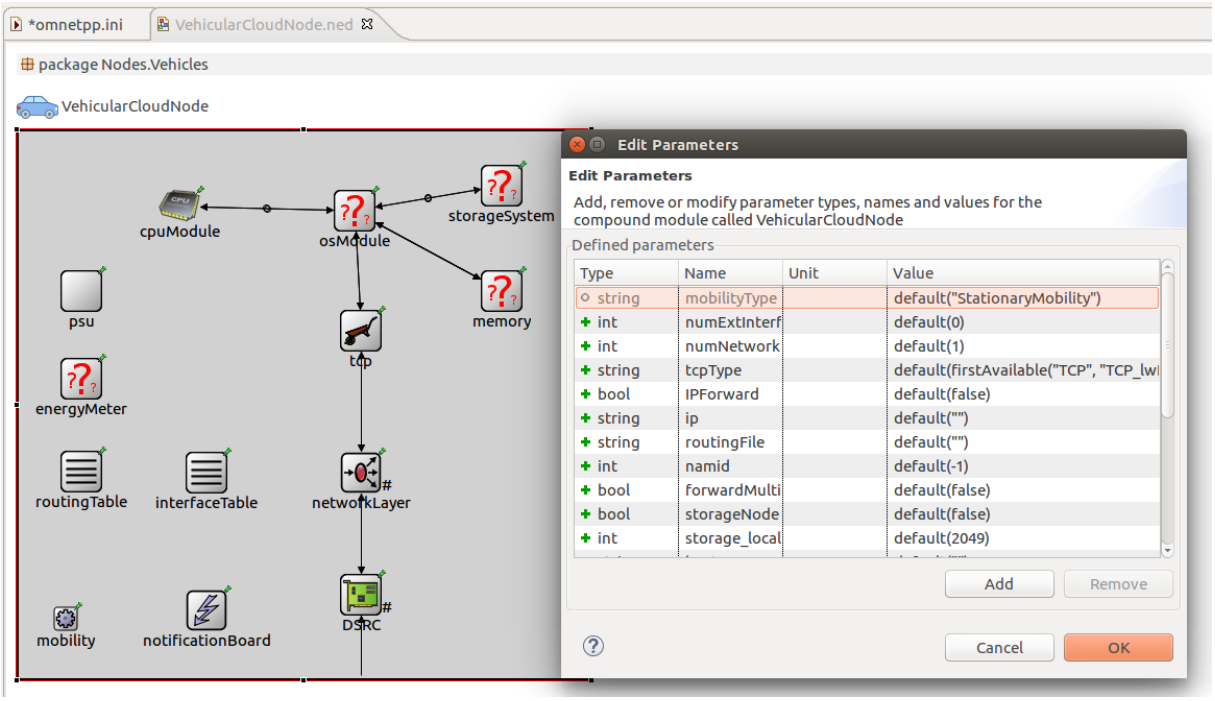

Figure 4- 11 Un noeud véhiculaire

4.5 La Simulation

4.5.1 Lancer le programme de simulation

Une fois que vous avez terminé les étapes précédentes, vous pouvez lancer la simulation en sélectionnant omnetpp.ini dans l'explorateur de projets, puis en appuyant sur le bouton Exécuter.

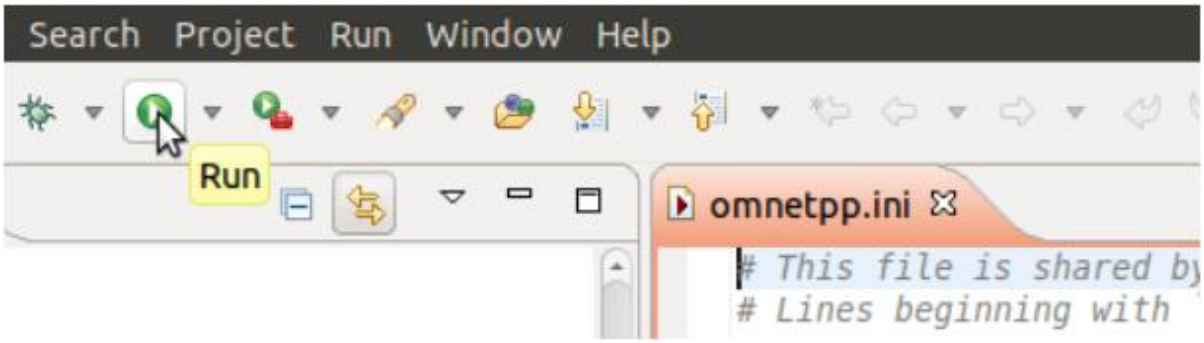

Figure 4- 12 Lancer une simulation

L'IDE *Building* votre projet automatiquement. S'il y a des erreurs de compilation, vous devez les corriger jusqu'à obtenir une compilation et une liaison sans erreur. Vous pouvez déclencher manuellement un *Build* en sélectionnant *Project -> Build* tout dans le menu ou en appuyant sur Ctrl+B.

Après avoir *build* et lancé votre simulation avec succès, vous devriez voir apparaître une nouvelle fenêtre graphique, similaire à celle de la capture d'écran ci-dessous. Vous devriez également voir le réseau contenant les nœuds affichés graphiquement dans la zone principale.

Appuyez sur le bouton Exécuter de la barre d'outils pour lancer la simulation. Ce que vous devriez voir, c'est que les véhicules échangent des messages l'un avec les autres.

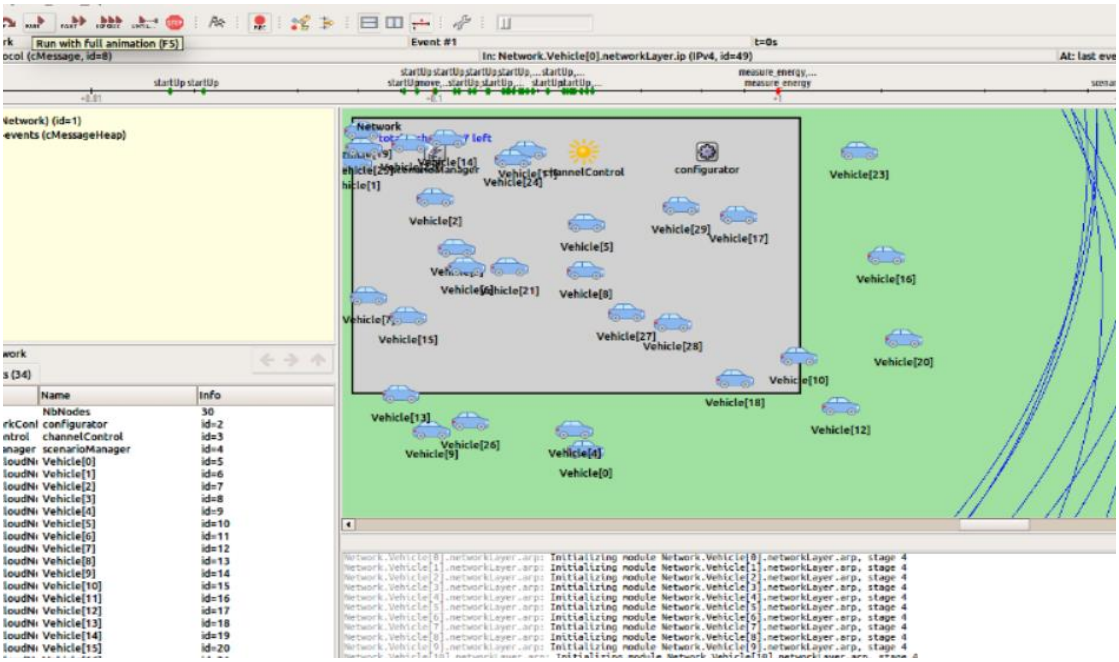

Figure 4- 13 Lancement de simulation

Après le *Run* de simulation, les nœuds véhiculaires seront initialisés, après, un parmi les nœuds envoie un message via une interface radio,

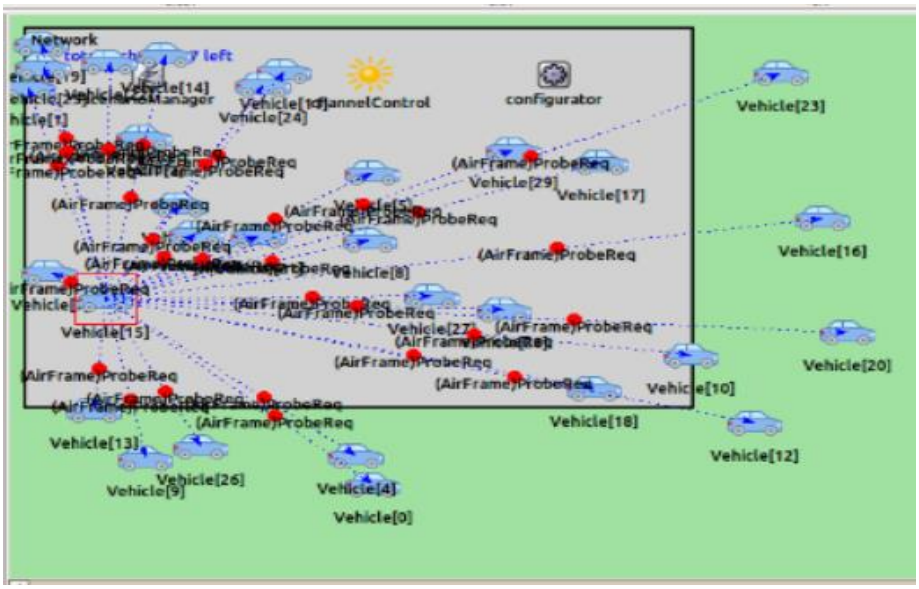

Figure 4- 14 L'échangement de message entre les nœuds

Le premier message envoyé active le module *Cluster Controller*, ce module est responsable de plusieurs taches, premièrement, il extrait tous les nœuds dans la topologie et les déférents liens entre ces nœuds, après, il crée les déférents clusters possibles et les groupes dominants de chaque cluster ou chaque cluster sera coloré avec une couleur et son groupe dominant avec un autre couleur, la Figure 4-15 représente les déférentes clusters et leur groupes dominants.

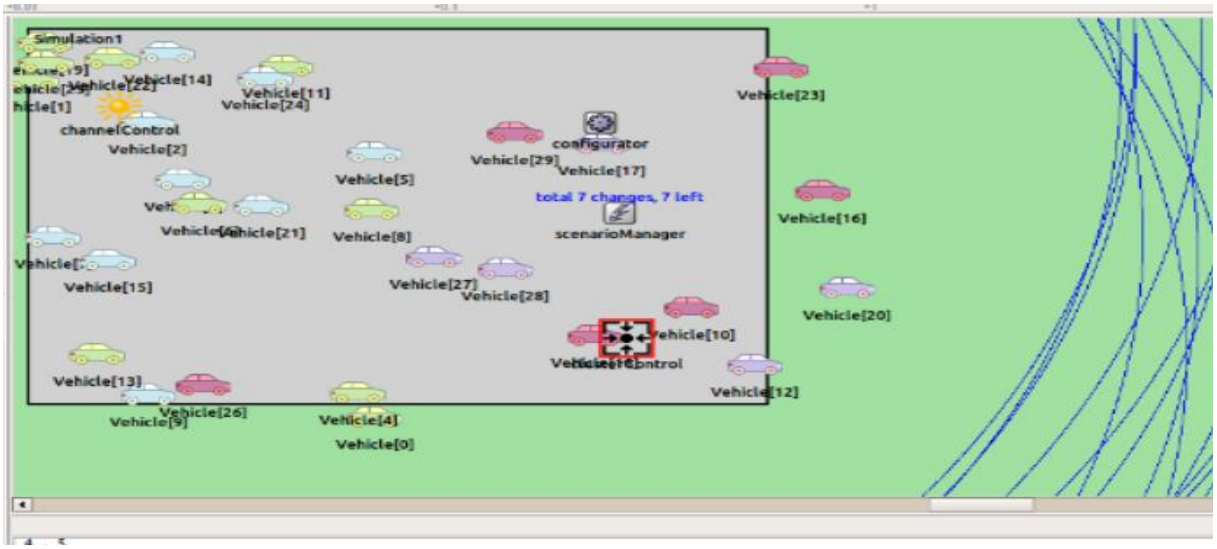

Figure 4- 15 Colorisation des nœuds

Au cours de simulations, chaque composant fait une action, pour voir le comportement de ce module il sufit de faire double click sur lui. Après avoir lancer la simulation, le module *Cluster Controller* extrait les déférents sous réseaux (SubNet), le nombre de sous réseaux et les groupes dominants (Dominant Set) comme il illustré dans la Figure 4-16.

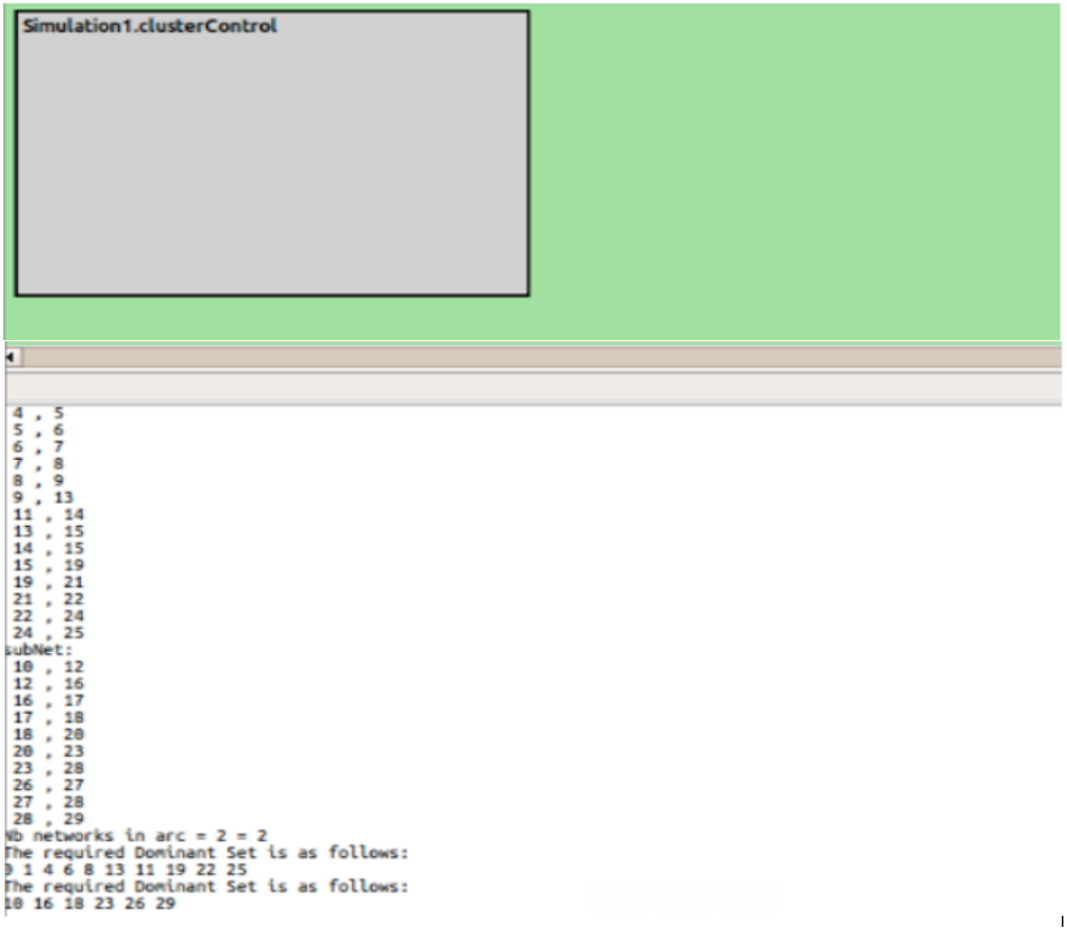

Figure 4- 16 Le module Cluster Controller

Selon la topologie initiale, le C*lusterControl* donne un résultat (Figure 4-17), quatre clusters, chaque un avec ces membres, et à cause de la mobilité du nœuds, la topologie rechange, donc les places des nœuds change qui implique des modifications dans les clusters, alors, le *ClusterControl* redéfini cinq nouveaux clusters (Figure 4-18).

```
subNet:
 0 , 4
 4 , 8
 8 , 11
 11 , 14
 14 , 19
 19 , 22
 22 , 24
 24 , 25
subNet:
 1 , 2
 2 , 3
 3 , 5
 5 , 6
 6 , 7
 7 , 9
 9 , 13
 13 , 15
 15 , 21
 21 , 22
subNet:
 10 , 12
 12 , 16
 16 , 17
 17 , 18
 18 , 20
subNet:
 27 , 28
 28 , 29
Nb networks            = 4
The required Dominant Set is as follows:
0 8 14 22 25
The required Dominant Set is as follows:
1 3 6 9 15 22
The required Dominant Set is as follows:
10 16 18
The required Dominant Set is as follows:
27 29
```

Figure 4- 18 Résultat 1

```
subNet:
 0 , 4
 4 , 8
 8 , 11
 11 , 14
 14 , 19
 19 , 22
 22 , 24
 24 , 25
subNet:
 1 , 2
 2 , 3
 3 , 5
 5 , 6
 6 , 7
 7 , 9
 9 , 13
 13 , 15
 15 , 21
 21 , 24
subNet:
 10 , 12
 12 , 16
 16 , 17
 17 , 23
subNet:
 18 , 20
subNet:
 27 , 28
 28 , 29
Nb networks in arc = 5 = 5
The required Dominant Set is as follows:
0 8 14 22 25
The required Dominant Set is as follows:
1 3 6 9 15 24
The required Dominant Set is as follows:
10 16 23
The required Dominant Set is as follows:
18
The required Dominant Set is as follows:
27 29
```

Figure 4- 17 Résultat 2

4.6 Conclusion

Dans ce chapitre, nous avons présenté la réalisation de notre plateforme dédiée à la simulation des applications Cloud véhiculaires et la dissémination des informations du trafic.

Après l'exécution du scenario proposé nous avons vu que les modules créés aient fonctionnent bien, et son principe est basé sur quatre étapes de base qui sont :

- Extraction des sous réseaux.
- Extraction des liens.
- Le calcule de groupe dominant.
- La colorisation des nœuds.

Le but de cette approche est de mettre en place la différence entre les déférent nœud de calcule et de stockage, diminuer également la charge sur les véhicules cloud et assurer la réception des paquets.

Conclusion Générale

Nous rappelons que l'objectif principale de cette thèse est de réaliser une extension omnet++ pour aider les utilisateurs simples d'omnet++ de faire des simulations des environnement VANETs Cloud, cette extension fournis essentiellement les déférents nœuds véhiculaires de base ou cloud, ces derniers sont équipés par une interface DSRC qui assure la dissémination des informations sur le trafic, puis nous implémentons un module qui nous permettons d'appliquer la technique du cluster.

Au début, nous avons présenté en générale les réseaux VANETs ; les composants, les caractéristiques et les déférents applications véhiculaires, en suite nous décrivons les déférents types et stratégies de disséminations des informations en VANET, et finalement, nous présentons l'intégration du Cloud Computing dans les réseaux VANETs.

Ensuite, nous avons cité les architecture VANET Cloud destiné à la dissémination des informations et les déférents techniques de dissémination comme la migration d'une machine vertuelle, l'agrégation du canal, le Clustering etc. Mais tous ces techniques et approches sont irréalisable à cause de type des nœuds, la nature de topologie qui a besoin d'un environnement spécial ; caractérisé par des centaines de véhicules occupées par des capacités cloud, des RSUs, de déférents types de capteurs sans file et embarquées. Tous ces exigences mettent les projets VANETs irréalisable. Pour cela nous avons eu recours à la simulation.

Dans notre contribution, nous avons mentionné les déférent problèmes de dissémination des informations, aussi nous décrivons comment nous utilisons les

clusters et les clusters vertuelle dans les VANETs Cloud ou les clusters vertuelle joue le rôle de data centre et comment ces clusters adaptatifs, ensuite nous présentons le méta modèle de notre projet, le diagramme de classe, et les déférents composants ICanCloud et INET, et en fin de chapitre nous présentons les algorithmes implémentés.

Finalement, dans le dernier chapitre, nous présentons notre projet final, l'implémentation des déférents algorithmes et fonctions et l'implémentation du module *Cluster Control*, puis, nous décrivons le scenario de simulation et les déférents étapes et résultats de simulation, en fin, nous extrairons les statistiques de simulation.

Les perspectives et les travaux potentiels dans le futur sont

- Parmi les inconvénients du communication radio est l'envoie des paquets à tous les nœuds, nous essayons d'acheminer les paquets vers les nœuds spécifique.
- Gérer les exclusions mutuelles et les accès aux nœuds de calcul et de stockage selon l'importance de message.
- Les véhicules créés sont basés sur le protocole TCP, en va ajouter le protocole UDP

Bibliographie

[1] R. Mazot, W. Meslem, M. Layouni, and A. Tran, *Communication inter véhiculaire*. Thèse de Doctorat, Arles Avignon, 2013.

[2] M. Erritali. *Contribution à la sécurisation des réseaux ad hoc véhiculaires*. Thèse de Doctorat, Université Mohammed Vagdal Rebat, 2014.

[3] N. Chaib. *La sécurité des communications dans les réseaux VANET*. Thèse de Doctorat, Universite Elhadj Lakhdar, Batna, 2013.

[4] K. Ait Ali. *Modelling and performance study in VANET networks*. Thèse de Doctorat, Université de Technologie de Belfort-Montbeliard, Nov. 2012.

[5] K. Moghraoui, *Gestion de l'anonymat des communications dans les réseaux véhiculaires AD HOC sans fil (VANETs)*. Thèse de Doctorat, l'université du Québec à Trois-Rivières, 2015.

[6] D. Bektache, *Application et Modélisation d'un protocole de communication pour la sécurité routière*. Thèse de Doctorat, l'université de Badji Mokhtar Annaba, 2014.

[7] M. JERBI, *Protocoles pour les communications dans les réseaux de véhicules en environnement urbain : Routage et Géocast basés sur les intersections*. Thèse de Doctorat, l'université d'Evry Val d'Essonne, 2008.

[8] P. Plainchault, *sécurisation de la conduite par communication véhicule infrastructure a base de transpondeurs*. Thèse de Doctorat, institut national polytechnique de Toulouse, 2005.

[9] F. Belarbi, *les systèmes de communication entre les véhicules et l'infrastructure : leur contribution aux pratiques d'exploitation de la route*. Thèse de Doctorat, l'École Nationale des Ponts et Chaussées, 2004.

[10] J. Petit, *Surcoût de l'authentification et du consensus dans la sécurité des réseaux sans fils véhiculaires*. Thèse de Doctorat, l'université de Toulouse, 2011.

[11] D. ZEKRI, Agrégation et extraction des connaissances dans les réseaux inter-véhicules. Thèse de Doctorat, université de valenciennes et du Hainaut Cambrésis, 2013.

[12] N. HADDADOU, *Réseaux ad hoc véhiculaires : vers une dissémination de données efficace, coopérative et fiable* Thèse de Doctorat, École Doctorale MSTIC, 2014.

[13] A. BENAIDJA, *échange d'informations en temp réel dans les réseaux véhiculaires,* Thèse de Doctorat, université de USTHB, 2016.

[14] S. BITAM et Al, *vanet-cloud : à generic cloud computing model for vehicular ad hoc networks*, IEEE Wireless Communications, February 2015, pp.96-107.

[15] Med. Al Mamun et Al, *Deployment of Cloud Computing into VANET to Create Ad Hoc Cloud Network Architecture*, Proceedings of the World Congress on Engineering and Computer Science 2012 Vol I WCECS 2012, Octobre 24-26, 2012, San Francisco, USA.

[16] R. Hussain et Al, *Rethinking Vehicular Communications : Merging VANET with Cloud Computing*, 2012 IEEE 4th International Conference on Cloud Computing Technology and Science, pp.606-609.

[17] R. Kim et Al, Prefetching-Based Data Dissemination in Vehicular Cloud Systems ,

IEEE Transactions on Vehicular Technology , Jan. 2016, pp. 292 – 306.

[18] K. Erciyes et Al, *Modeling and Simulation of Mobile Ad hoc Networks*, Research Gate

November 2011.

[19 Teerawat Issariyaku , Ekram Hossain,(2009), *Introduction to Network Simulator NS2*, Springer Science+Business Media, 438p.

[20] The Network Simulator – ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/.

[21] H. Fahmy, *Wireless Sensor Networks Concepts, Applications, Experimentation and Analysis*, Springer Science+Business Media Singapore 2016, 635p.

[22] K. Wehrle, *Modeling and Tools for Network Simulation*, Springer-Verlag Berlin Heidelberg 2010, 566p.

[23] Benzouaoua A, Kebbi M, *Conception et Réalisation d'un simulateur dédie pour les réseaux véhiculaires*, Thèse de Master, université de bejaia, 2016.

[24] The Network Simulator – ns-3. [Online]. Available: https://www.nsnam.org/.

[25] Klügl, Franziska et Bazzan, Ana. (2012). *Agent-Based Modeling and Simulation*.

[26] Daniel K, *SUMO (Simulation of Urban MObility) An open-source traffic simulation*, Conference paper, January 2002.

[27] Davide K. Hale,*CORSIM simulation overview*, Research Gate, January 2010.

[28] Robert L. Bertini et AL, *Application of PARAMICS Simulation At a Diamond Interchange*, Research Gate, April 30, 2002.

[29] Sykes P. (2010), *Traffic Simulation with Paramics Fundamentals of Traffic Simulation*, International Series in Operations Research & Management Science, vol 145. Springer.

[30] Francisco J et Al, *CityMob: a mobility model pattern generator for VANETs*, ICC Workshops - 2008 IEEE International Conference on Communications Workshops, Beijing, China.

[31] vehicular network simulation framework . [Online]. Available: https://veins.car2x.org/documentation/

[32] M. Talib et Al, *Converging VANET with Vehicular Cloud Networks to reduce the Traffic Congestions : A review*, International Journal of Applied Engineering Research Volume 12, Number 21 (2017) pp. 10646-10654

[33] T. Hajji et Al, *Design of a VANET Testbed based on Cloud Computing*, European Conference on Networks and Communications, 2015.

[34] B. Liu et Al, *Cloud-Assisted Safety Message Dissemination in VANET-Cellular Heterogeneous Wireless Network.* IEEE SYSTEM JOURNAL, 2015.

[35] M.Rayeni et Al, *Quality of service aware multicasting in heterogeneous vehicular networks*, Vehicular Communications, Volume 13, July 2018, Pages 38-55, Elsevier.

[36] A. Zekri et Al, *Heterogeneous vehicular communications : A comprehensive study,* Ad Hoc Networks, Volumes 75–76, June 2018, Pages 52-79, Elsevier.

[37] I. Wahid et Al, *State of the Art Routing Protocols in VANETs*, Procedia Computer Science, Volume 130, 2018, Pages 689-694, Elsevier.

[38] Venkatesh et Al, *Routing Protocols for Vehicular Adhoc Networks (VANETs)*, Journal of Emerging Trends in Computing and Information Sciences, Vol. 5, No. 1 January 2014.

[39] B.T. Sharef, R.A. Alsaqour, and M. Ismail. Vehicular communication ad hoc routing protocols. A survey. Journal of Network and Computer Applications, pp. 363-396, *(2014).*

[40] P. Jacquet et Al, *Optimized Link State Routing Protocol for Ad Ho Networks*, Proceedings. IEEE International Multi Topic Conference, 2001, Lahore, Pakistan.

[41] Charles E. Perkins et Al, Highly Dynamic Destination-Sequenced Distance- Vector Routing (DSDV) for Mobile Computers, protocols and applications, Computer Communication Review, Pages 234-244.

[42] M. Singh et Al, *A Survey : Ad-hoc on Demand Distance Vector (AODV) Protocol, (2017).* A Survey : Ad-hoc on Demand Distance Vector (AODV) Protocol. International Journal of Computer Applications.

[43] David b. Johnson et Al, *Dynamic Source Routingin Ad Hoc Wireless Networks*, (1996), Mobile Computing. The Kluwer International Series in Engineering and Computer Science, vol 353. Springer, Boston, MA

[44] Rong Y et Al, *Virtual Machine Live Migration for Pervasive Services in Cloud-Assisted Vehicular Networks*, 8th International Conference on Communications and Networking in China, 2013.

[45] Tarek K. Refaat et Al, *Virtual machine migration and management for vehicular clouds*, Vehicular Communications, Volume 4, April 2016, Pages 47-56, Elsevier.

[46] R. Pal et Al, *Analytical model for clustered vehicular ad hoc network analysis*, ICT ExpressVolume 4, Issue 3, September 2018, Pages 160-164

[47] Hamid R. Arkian et Al, *Cluster-based traffic information generalization in Vehicular Ad-hoc Networks*, Vehicular Communications, Volume 1, Issue 4, October 2014, Pages 197-207.

[48] Iftikhar A et Al, *VANET–LTE based heterogeneous vehicular clustering for driving assistance and route planning applications*, Computer Networks, Volume 145, 9 November 2018, Pages 128-140.

[49] Rasmeet S. Bali, *Learning Automata-assisted Predictive Clustering approach for Vehicular Cyber-Physical System*, Computers & Electrical Engineering, Volume 52, May 2016, Pages 82-97.

[50] A. Rahim et Al, *Vehicular Social Networks: A survey*, Pervasive and Mobile Computing,Volume 43, January 2018, Pages 96-113.

[51] INET User's Guide, [Online]. Available: inet.omnetpp.org/docs/users-guide/

[52] A. Nunez et Al, *ICanCloud : A Flexible and Scalable Cloud Infrastructure Simulator*, Journal of Grid Computing, 2012.

[53] OM NET++ User's Guide, [Online]. Available ; omnetpp.org/omnetpp/UserGuide.pdf

[54] L. Augustin, a software architecture for adaptive and distributed mobile applications, Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications, Taormina-Giardini Naxos, Italy, 1-4 July 2002.