*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 sécurités informatique*
*Thème*

# Une extension de la plateforme Omnet++ pour la gestion de la mobilité dans les VANET-CLOUD

*Présenté Par :*
*DEGAICHIA Adel*

*Devant le jury :*

| | | | |
|---|---|---|---|
| *Mr T.Mekhaznia* | *MCB* | *Univérsité de Larbi Tébessi* | *Président* |
| *Mr R.Mahmoudi* | *MAA* | *Univérsité de Larbi Tébessi* | *Examinateur* |
| *Mr A.Sahraoui* | *MAA* | *Univérsité de Larbi Tébessi* | *Encadreur* |
| *Mr M.Derdour* | *MCA* | *Univérsité de Larbi Tébessi* | *Co-Encadreur* |

*Date de soutenance : Juin 2019*

# Résumé

Les réseaux ad hoc véhiculaire (VANETs) ont fait l'objet des études approfondies en raison de la diversité de leurs applications et services, tels que la sécurité des passagers, l'amélioration de l'efficacité du trafic et l'infotainment. Plusieurs problèmes techniques de déploiement et de gestion sont dus à une flexibilité réduite, une évolutivité réduite, une connectivité médiocre et une intelligence insuffisante. Le Cloud Computing est considéré comme un moyen de satisfaire ces exigences dans les VANET. Dans ce travail, nous nous intéressons à l'amélioration d'un modèle de circulation automobile dans les situations d'encombrement. Alors, ce travail est destiné à ajouter un certain valeur d'intelligence et flexibilité au système de covoiturage a l'aide de l'algorithme de la colonie de fourmi optimisé ce qui aide de trouver le plus court chemin pour les voitures qui servent dans ce système, et pour également résolu le problème d'embouteillage routier par la remplir de tous les sièges de véhicule, sachant que tous les voitures et les passagers dans ce système sont différents au point de départ, mais ils ont la même destination. Les plateformes exploité pour implémenter ce travail sont : Omnet++, INET et iCanCloud.

**Mots-clés** : Cloud Computing véhiculaire, Omnet++, gestion de mobilité, congestion, VANETs.

# Abstract

Vehicular ad hoc networks (VANETs) have been extensively studied due to the diversity of their applications and services, such as passenger safety, improved traffic efficiency and infotainment. Several technical issues of deployment and management are due to reduced flexibility, reduced scalability, poor connectivity and poor intelligence. Cloud computing is seen as a way to meet these requirements in VANETs. In this work, we are interested in the improvement of a vehicular traffic model in congested situations. So, this work is intended to add a certain amount of intelligence and flexibility to the carpooling system using the optimized ant colony algorithm which helps to find the shortest path for the vehicules that serve in this system, and for also solved the problem of road bottling by filling all the vehicle seats, knowing that all vehicle and passengers in this system are different at the starting point, but they have the same destination. The frameworks exploited to implement this work are: Omnet++, INET and iCanCloud.

**Keywords** : Vehicular Cloud Computing, Omnet++, Mobility management, congestion, VANETs.

# ملخص

يتم دراسة شبكات سيارات الأدوك (فانيت) بشكل كبير بسبب تنوع تطبيقاتها و خدماتها، مثل أمن المسافر، تحسين فعالية حركة المرور و التسلية. مختلف المشاكل التقنية للتطبيق و التسيير هو بسبب نقص و التوسع، ضعف الإتصال و الذكاء. الكلاود كامبيوتين هي من أحد وسائل تلبية هذه الإحتياجات في الفانيت. في هذه المذكرة نحن مهتمون بتحسين نموذج حركة السيارات في حالات الإحتقان المروري. إذن هذا العمل موجه لإضافة مقدار من الذكاء و المرونة لأنظمة الكاربولين بإستعمال خوارزمية مستعمرة النمل للتحسين الذي يساعد على إيجاد أقصر طريق للسيارات التي تخدم في هذا النظام، و حل مشكلة إكتظاظ الطريق بملئ جميع مقاعد السيارة، مع العلم أن السيارة و المسافرين في مكان الإنطلاق و لكن لهم نفس الوجهة. أطر العمل المستغلة لإنجاز هذا العمل هي: أومنت بلاس بلاس، أينيت و أيكانكلاود.

**الكلمات المفتاحية:** سيارات الكلاود كامبيوتين، أومنت بلاس بلاس، تسيير الحركة، الإكتظاظ، فانيت.

# Dédicace

*A ma famille,*

*A mon encadreur Dr. Abdelatif Sahraoui,*

*A mon Co-Encadreur Dr. Derdour Makhlouf*

# Remerciement

*Mes remerciements vont tout premièrement à Allah (dieu) tout puissant pour ses grâces qui sont innombrable et incomptable.*

*Je suis très reconnaissant aux membres du jury qui ont accepté de réviser et d'évaluer ma mémoire et de fournir des critiques pertinentes.*

*Je tiens ma spécial gratitude à mon encadreur Dr. Abdelatif Sahraoui, et mon co-encadreur Dr. Derdour Makhlouf pour l'accepte de diriger cette mémoire durant mon parcours d'étude.*

*Spécial gratitude également à mes amis Yahiaoui Med Bachir, Ghozlane Abd elhak et EL Hamza Houssem.*

*Je profite cette occasion pour exprimer ma gratitude à tous mes collègues et mes amis.*

# Table des Matières

# Liste des figures

# Liste des Tableaux

# Introduction Générale

# Introduction générale

La technique de simulation a eu une grande importance, car elle permet de mettre les hypothèses et les théories à l'épreuve, sans tenir compte des conséquences négatives qui peuvent survenir si le teste de ces hypothèses et théories directement sur le terrain.

Généralement, la simulation permet de suivre et prévoir la dynamique d'un processus, d'un système ou d'un modèle. Ce dernier est le sujet de nombreux domaine de recherche. Les concepteurs des applications ont vu comme un processus de simulation intermédiaire entre le monde réel et le monde théorique, ainsi que le teste dans un vrai sens et plus profond. S'il y'a eu un doute sur une théorie particulière et ses conséquences sont inconnues, et qu'on souhaite d'assurer qu'ils étaient corrects, on peut être dérivé un modèle de la théorie et expérimenté ce modèle à l'aide de simulation. Si les résultats souhaités sont obtenus, il est possible de raffiner et exploiter ce modèle en présent, et le perfectionner et développer en future.

Il faut prendre en considération que la situation réelle n'est pas toujours reflétée par la simulation, elle ne peut qu'on lui ait donné la possibilité de le faire et que la reproduire. Le contexte de notre travail à une forte de besoin d'utiliser les techniques de simulation au lieu d'essaie concrètement. En particulier, nous allons proposer un modèle de simulation pour les applications moderne de trafic. En effet, ces applications nécessitent des moyens de concrétisation financière et matériel. Alors que le coût de ces évaluations réelles augmente de façon exponentielle avec le nombre des échecs.

Les applications Cloud Véhiculaires (VC) à attirer dernièrement beaucoup d'intérêt ces dernier années pour être la prochaine génération des réseaux. Dés coup ces plateformes de simulations n'ont pas définie jusqu'à présent dans la littérature, le cadre général de nos propositions est de développer une plateforme de simulation permet aux concepteurs de ces applications d'évaluer le coût associe à une telle solution. En particulier, nous allons focaliser sur la gestion du trafic et la mobilité des véhicules. En effet, nous allons adopter le « *covoiturage* » comme un nouveau modèle de mobilité pour réduire le problème de congestion routière.

Le covoiturage consiste à partager des trajets en voiture de manière à ce que plus d'une personne se déplace en voiture et évite à d'autres personnes de devoir se rendre elles-mêmes en voiture. En faisant en sorte que plus de personnes utilisent un véhicule, le covoiturage réduit les coûts de déplacement de chaque personne, tels que les coûts de carburant, les péages et le stress de la conduite. Le covoiturage est également un moyen de transport plus respectueux de l'environnement et plus durable, car le partage des trajets réduit la pollution de l'air, les émissions de carbone, les embouteillages sur les routes et le besoin d'espaces de stationnement. Les autorités encouragent souvent le covoiturage, en particulier en période de pollution élevée ou de prix élevés du carburant. Le covoiturage est un bon moyen d'utiliser toute la capacité en sièges d'une voiture, qui autrement resterait inutilisée si c'était simplement le conducteur qui utilisait la voiture.

Le développement d'un système de covoiturage et de contrôle de congestion de trafic est un processus non trivial, où il n'est pas raisonnable de tester ce système dans un environnement réel.

Ceci est causer par coût élevé requis pour tester le modèle, les résultats catastrophiques qui peuvent survenir après le test en cas de mauvaise conception de modèle ou en cas des variations de données.

En général, et aussi lors de notre étude des réseaux VANETs-Cloud présenté dans ce mémoire, nous pouvons dire que la simulation est une technique indispensable, elle est un outil de gain de moyens et de temps et aussi d'apprentissage.

Il est difficile d'implémenter un scénario VANETs-Cloud en utilisant les plateformes de simulation existants tell que Omnet++, NS-2 et OpNet en raison de plusieurs enjeux. Notre contribution est de faire une extension sous la plateforme Omnet++ pour la gestion de la mobilité dans les réseaux VANETs-Cloud. La comparaison entre Omnet++, NS-2 et OpNet justifier notre choix (i.e., Omnet++) comme un modèle de simulation pour achever le travail proposé.

La proposition d'un modèle de simulation en tant que plateforme sous Omnet++ aider les concepteurs en ce domaine d'évaluer les solutions aux problèmes de mobilité en cas d'embouteillage en trafic routier.

La structure générale du notre travail est organisé en quatre chapitres comme suivante :

Nous allons introduisant dans le chapitre (01) le paradigme VANETs-Cloud qui est considéré comme un nouveau paradigme par rapport à les VANETs conventionnelles. Le chapitre (02) est consacré en deux partie, la première partie tourner au tour les principes de base de théorie de flux de trafic, la deuxième partie concerne les simulateurs de toutes sortes avec un comparaison entre les simulateurs de réseaux. Le chapitre (03) représente un explication détailler de la partie conceptuelle de travail proposé commençant par une description des outils utilisée pour atteindre les objectifs soulignés, après cela on passera à l'explication de l'algorithme de la colonie de fourmi pour optimiser les solutions de notre scénario de covoiturage proposée avec clarification des formules mathématique utilisé dans cette algorithme, et on le finie par le diagramme de classe qui est considérer comme une vue conceptuelle de notre modèle. Le chapitre (04) représente la partie réalisation de la plateforme proposée.

# Chapitre 01 :

# Introduction au Cloud Computing Véhiculaire

# Introduction :

Les réseaux ad hoc de véhicules (VANET) ont acquis une grande popularité ces dernières années, ces réseaux étant confrontés à de graves problèmes dans le monde tels que les accidents de la route, la congestion routière, la consommation de carburant et la pollution de l'environnement en raison du grand nombre de véhicules sur les routes. Les accidents de la route sont des problèmes persistants entraînant des pertes de vies humaines et matérielles considérables. Pour surmonter ces problèmes et rendre le voyage plus sûr et plus efficace, les systèmes de transport intelligent (STI) a introduit des réseaux de transport intelligents (VANET) afin de créer une infrastructure de transport plus sûre. Un réseau VANET se concentre sur la sécurité routière et la gestion efficace du trafic sur les routes publiques, offrant commodité et divertissement aux conducteurs et aux passagers le long du chemin.

Cette technologie VANET centralisée traditionnelle peut ne pas être efficace pour traiter d'énormes quantités de données de trafic générées par des véhicules intelligents tels que des données vidéo et des capteurs. Afin de collecter et de traiter une grande quantité d'informations de trafic instantanées, des serveurs supplémentaires sont nécessaires dans les zones distribuées. Les VANETS utilisant le paradigme Cloud Computing (CC) peuvent constituer une solution adaptée à ce type de situation. Récemment, le Cloud Computing (CC) a été adopté par divers appareils mobiles, véhicules et infrastructures pour gérer des calculs complexes difficiles à mettre en œuvre localement.

Dans ce chapitre, nous allons tout d'abord mettre en évidence les Systèmes de Transport Intelligents (STI), ces définitions, ces technologies utilisées et ces applications. Ensuite, nous allons présenter les réseaux VANET, ces architectures générales et ces avantages. Enfin, le paradigme du Cloud Computing Vehiclaire (CCV) est également présenté avec ces architectures, ces plates-formes et ces services.

# 1 Les réseaux véhiculaires :

## 1.1 Introduction aux systèmes de transport intelligents (STI) :

Le transport est une base fondamentale pour le fonctionnement quotidien de l'économie et la société. Au cours des dernières années, on a vu l'initiation, le développement et le déploiement du système de transport et effet significatif de ces développements dans notre société et notre vie. Nous pouvons donc redéfinir le système de transport en tant que STI. Aujourd'hui, ne pas uniquement les domaines du génie civil et mécanique qui concernent la recherche et le développement des transports. Les concepts d'ingénierie informatique, tels que l'intelligence artificielle (AI), l'apprentissage automatique, la communication, Internet et de nombreux autres domaines émergents de l'ingénierie et des sciences de l'information, deviennent le cœur des STI [01].

### 1.1.1 Définition de (STI) :

les STI sont définis comme un ensemble des applications avancées qui visent à appliquer des technologies intelligentes de l'information et de la communication afin de fournir des services de transport et de gestion du trafic. Les STI jouent un rôle crucial pour la réduction de nombreux problèmes tels que la pollution de l'air, les voyages à long temps, la consommation de carburant, les embouteillages et les accidents, qui ont augmenté en raison de la croissance démographique. Les organisations STI déploient d'innombrables efforts pour trouver des solutions à ces problèmes critiques en développant la communication sur la circulation et la mise en réseau des véhicules.

Le paradigme du contrôle et de la gestion du trafic est divisé en 5 phases distinctes. Dans la phase initiale, les ordinateurs sont de grande taille et coûteux. Par conséquent, toute la gestion du trafic est effectuée à l'aide d'un modèle centralisé. Dans la deuxième phase, des micro-ordinateurs sont introduits, ce qui conduit également à l’introduction du contrôleur de signalisation routière (TSC, Traffic Signal Controller). Chaque TSC comprend une capacité de stockage et une puissance de calcul permettant de gérer une intersection. Dans la troisième phase, l’introduction de réseaux Ethernet (LAN) et sans fil permet le partage des ressources et des informations. En cette ère d'internet, récupérer de nombreuses données sur des sites distants et les traiter nécessitent beaucoup de bande passante réseau. Pour résoudre ce problème, les agents mobiles et l'informatique à base d'agents sont introduits dans la quatrième phase. Les agents mobiles n’exigent que l’exécution et exécutent des calculs proches des données, ce qui réduit les coûts et les délais de communication. Mais cette informatique à base d’agent nécessite de grandes ressources de calcul. De plus, les équipements embarqués (OBE, On-Board Equipments) et les systèmes de positionnement global (GPS) dans les infrastructures de transport ne cessent d’être améliorés.

En dernière phase, le secteur informatique a été révolutionné par le cloud computing. C'est un nouveau paradigme qui fournit aux utilisateurs des ressources informatiques à la demande payables à l'utilisation. Ces dernières années, les systèmes de transport et de gestion parallèles (PtMS, Parallel transportation and Management System) sont devenus le point chaud des systèmes de gestion du trafic. Les PtMS utilisent un système de transport artificiel (ATS, Artificial Transport System) qui intègre des systèmes artificiels, une exécution en parallèle et des expériences de calcul. ATS permet d’évaluer et d’optimiser diverses stratégies de contrôle du trafic. ATS peut utiliser les

concepts de l'informatique en nuage pour bien coordonner les ressources et également contrôler le travail des stratégies visant à améliorer les performances des systèmes de trafic et à minimiser les exigences en matériel [01].

### 1.1.2 Les technologies utilisées dans les (STI) :

*** Global Positioning System (GPS) :** Les récepteurs GPS intégrés dans les unités embarquées du véhicule (OBU) reçoivent des signaux de plusieurs satellites différents pour calculer la position du dispositif (et donc du véhicule). Cela nécessite une visibilité directe sur les satellites, ce qui peut empêcher l'utilisation du GPS dans les centres-villes en raison des effets de «canyon urbain». L'emplacement peut généralement être déterminé à une dizaine de mètres. Le GPS est la technologie de base de nombreux systèmes de navigation embarquée et de guidage d'itinéraire.

*** Dedicated-Short Range Communications (DSRC) :** Le DSRC est un canal de communication sans fil à courte et moyenne portée, fonctionnant dans le spectre sans fil à 5,8 ou 5,9 GHz, spécialement conçu pour les applications automobiles. De manière cruciale, le DSRC permet des communications sans fil bidirectionnelles entre le véhicule (via des balises ou des capteurs intégrés) et des équipements en bordure de route (RSE, roadside equipment). Le DSRC est une technologie clé pour de nombreux systèmes de transport intelligents, y compris l'intégration de véhicule à infrastructure, la communication de véhicule à véhicule, la synchronisation adaptative des feux de circulation, la collecte de péage électronique, la tarification électronique de la route, la fourniture d'informations, etc. DSRC est un sous-ensemble de la technologie d'identification par radiofréquence (RFID, radio frequency identification). La technologie pour les applications ITS fonctionne sur la bande de fréquences 5.9 GHz (États-Unis) ou sur la bande de fréquences 5.8 GHz (au Japon et en Europe).

*** Wireless Networks :** Semblables à la technologie couramment utilisée pour l'accès Internet sans fil, les réseaux sans fil permettent des communications rapides entre les véhicules et le bord de la route, mais ne couvrent que quelques centaines de mètres. Toutefois, cette plage peut être étendue de manière à ce que chaque véhicule ou nœud de route successif transmette des informations au véhicule ou nœud suivant.

*** Mobile Telephony :** Les applications ITS peuvent transmettre des informations sur des réseaux de téléphonie mobile standard de troisième ou quatrième génération (3G ou 4G). Les avantages des réseaux mobiles incluent une grand disponibilité dans les villes et le long des routes principales. Cependant, une capacité de réseau supplémentaire peut être nécessaire si les véhicules sont équipés de cette technologie, et les opérateurs de réseau devront peut-être couvrir ces coûts. La téléphonie mobile peut ne pas convenir à certaines applications STI critiques pour la sécurité, car elle peut être trop lente.

*** Roadside Camera Recognition :** Les systèmes basés sur une caméra peuvent être utilisés pour des systèmes de taxation de la congestion basés sur des zones ou pour des routes spécifiques. Ces systèmes utilisent des caméras placées sur les routes où les conducteurs entrent et sortent des zones de congestion. Les caméras utilisent la reconnaissance automatique des plaques d'immatriculation (ALPR, Automatic License Plate Recognition), basée sur la technologie de reconnaissance optique des caractères (OCR, Optical Character Recognition), pour identifier les plaques d'immatriculation

des véhicules; ces informations sont transmises numériquement aux serveurs d'arrière-guichet, qui évaluent et comptabilisent les redevances dues aux conducteurs pour leur utilisation des routes dans la zone de congestion [02].

### 1.1.3 Les Applications (STI) :

Les différentes applications ITS sont décrites comme suit:

*** Contrôle de la circulation :** Il se concentre principalement sur la priorisation des modes de transport tels que les autobus, les cyclistes, les piétons et autres véhicules d'urgence afin d'évaluer les performances et d'étudier les causes des émissions de la circulation et de la congestion.

*** Systèmes de gestion des catastrophes :** Diverses technologies sont utilisées à cette fin afin de fluidifier le trafic et de fournir une aide médicale et toute autre assistance connexe dans de tels cas.

*** Systèmes d'information et de navigation du véhicule :** Le système d'information embarqué avertit les conducteurs des conditions climatiques défavorables, de l'état de la chaussée, des embouteillages et des dangers, y compris les accidents. Les systèmes de navigation fournissent des informations de localisation du véhicule en temps réel et permettent au conducteur de suivre son itinéraire de manière optimale.

*** Systèmes d'assistance à la conduite :** Afin de préserver le conducteur des accidents, ces systèmes ont Certaines décisions de conducteur humain ont été remplacées par des décisions de machine, qui contribuent également à un contrôle plus fluide du véhicule.

*** Contrôle de la pollution de l'air :** Le transport routier est la principale source de pollution atmosphérique qui a causé impact sur la santé humaine et la qualité de l'environnement. Différents modèles et protocoles sont utilisés dans les STI pour contrôler la pollution de l'air [01].

## 1.2 L’architecture générale des VANETs :

D’après des statistiques réalisé par des experts dans les derniers années qu’ils indiquent que le nombre de véhicules dans le monde en augmentation terrible et à un rythme rapide, ce qui explique pourquoi cela entraîne les accidents de la route parmi les principales causes de mortalité dans le monde. Pour cette raison, les chercheurs ont pensé à inventer un technologie pour résoudre ce problème, ou au moins pour l'atténuer. Et ainsi les Réseaux ad hoc véhiculaires (VANETs) ont émergé.

Qu'est-ce que les VANET ?, est une sorte de Réseaux ad hoc mobiles (MANETs), Le composant le plus important en eux est le véhicule, ce dernier est équipé par Le (OBU, On-Board Unit) qui rassemble de calculateurs et dispositifs de communications sans fil, alors le VANET est un ensemble de véhicules intelligentes qui se connectent entre eux directement en mode décentralisé (V2V, vehicle to vehicle) basés sur la communication inter-véhicules sans utilisation d’infrastructure. Ou se connectent en mode (V2I, vehicle to infrastructure) à une infrastructure (RSUs, Road Side Units) qui installées au bord des routes, leur responsabilité principale est la gestion du trafic et des véhicules. Ou hybride.

Lorsque nous parlons de l'architecture générale des réseaux ad hoc véhiculaire, nous ne pouvons pas oublier de parler de l'autorité de certification (CA, Certification Authority). Le CA est responsable de la prévention des problèmes de sécurité dans les réseaux de véhicules. Ce composant est conscient des véhicules à proximité et fournit une validation et une évaluation de l'entité sur les sources d'informations lors de la réception de messages. Une autorité de certification est une partie entièrement fiable de l'environnement et est cruciale pour les réseaux véhiculaires. Dans l'architecture, les entités de confiance communes servent d'agents de certification, tels que le service de transport municipal.

Le capacité et l'autonomie d'énergie est l'une des caractéristiques les plus importantes en VANET, parce que ses éléments n'ont pas de limite en terme d'énergie, car les OBUs utilisent les batteries des véhicules, et les RSUs sont déjà alimentés par une source d'énergie illimitée, deux autre caractéristiques pas moins important que le premier, le changement de topologie et le forte mobilité liée à la vitesse de déplacement des véhicules qui peut rapidement rejoindre ou quitter le réseau en un temps très court, ce qui rend les changements de topologie très fréquents, le deuxième caractéristique est le variation des environnements d'un VANET qui peut être une route, une autoroute ou une ville, c'est-à-dire une situation d'embouteillage dans la ville peut mener à l'encombrement du réseau, tandis qu'une route de campagne peut conduire à la disparition des liens du réseau.

Le caractéristique de la forte mobilité et le changement de topologie fait que la question de routage en VANET un grand défi. Donc différentes solutions pour le routage dans les réseaux VANETs ont été proposées :

Les protocoles basés sur la topologie, qui sont divisés en protocoles réactifs adoptent des algorithmes classiques tels que le routage par vecteur de distance, les routes sont établies uniquement sur demande et seules les routes en cours d'utilisation sont maintenues. Les protocoles proactifs sachant que chaque noeud garde une image de la topologie de tout le réseau, cette image est mètre à jour périodiquement ou à chaque modification topologique, et les protocoles hybrides.

Et des protocoles de routage basés sur la localisation géographique sont les plus adaptés pour les réseaux VANETs, puisque le mécanisme de routage se base sur les données géographiques des nœuds.

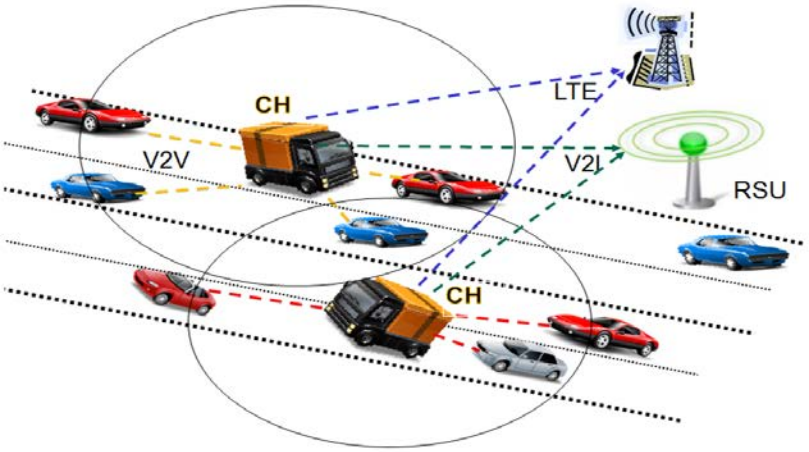



**Figure (1-1): L'architecture des VANETs**

# 2 Le paradigme Cloud Computing :

Le Cloud Computing est un modèle de fourniture ou d'utilisation de services dans le domaine des technologies de l'information et de la communication (TIC) par l'intermédiaire d'un réseau comme celui de l'internet. Les services proposés peuvent comprendre les serveurs, les systèmes d'exploitation, les réseaux, les logiciels, le stockage et les applications. Les services Cloud Computing sont proposés par l'intermédiaire d'un groupe modulable et variable de ressources qui peuvent être partagées et qu'utilisent un matériel installé dans des centres de données conçus à cette fin [03].

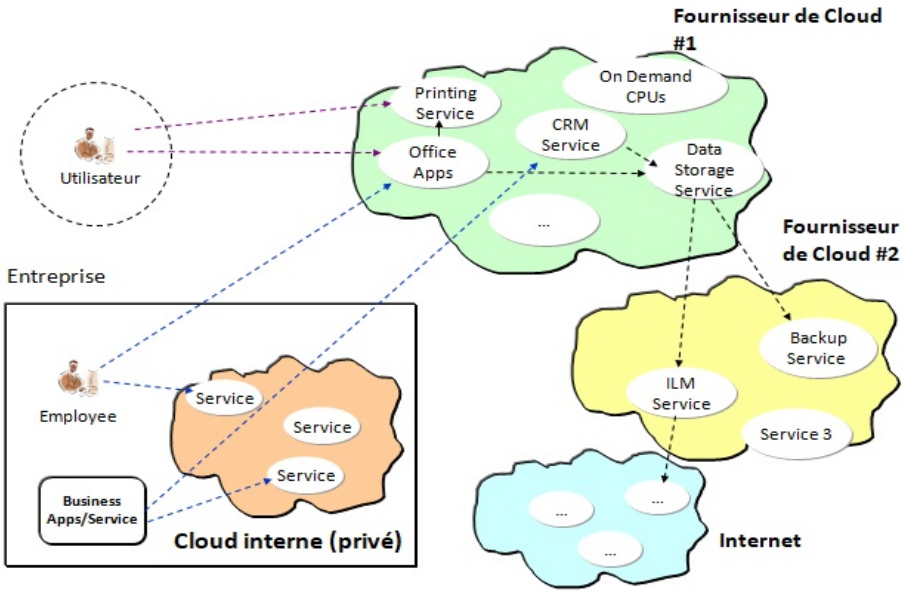



**Figure (1-2):**

**Le paradigme Cloud Computing.**

## 2.1 Les services de Cloud Computing :

Selon le (National Institute of Standards and Technology) il existe trois catégories de services qui peuvent être offerts en cloud computing, IaaS, PaaS et SaaS :

**- Infrastructure as a Service (IaaS):** Ceci est la couche de base du modèle de la pile du Cloud. Il sert de base pour les deux autres couches, pour leur exécution. Il consiste à offrir un accès à un parc informatique virtualisé. Des machines virtuelles sur lesquelles le consommateur peut installer un système d'exploitation et des applications. Le consommateur est ainsi dispensé de l'achat de matériel informatique. Ce service s'apparente aux services d'hébergement classiques des centres de traitement de données et la tendance est en faveur de services de plus haut niveau, qui fonte davantage abstraction de détails techniques [03].

**- Platform as a Service (PaaS):** Maintenant, vous n'avez pas besoin d'investir des millions de dollars pour obtenir cette plateforme de développement prête pour vos développeurs. Le fournisseur PaaS vous livrera cette plateforme sur le web, et dans la plupart des cas, vous pouvez utiliser cette plateforme en utilisant votre navigateur, sans avoir besoin de télécharger un logiciel, cette couche

contient : OS du Cloud, Middleware qui est un logiciel tiers créant un réseau d'échange d'informations entre différentes applications informatiques en Cloud. La différence étant que les systèmes sont mutualisés et offrent une grande élasticité - capacité de s'adapter automatiquement à la demande, alors que dans une offre classique d'hébergement web l'adaptation fait suite à une demande formelle du consommateur [03].

**- Software as a Service (SaaS):** Ceci est le plus haut niveau de la couche de la pile du Cloud directement utilisé ou consommé par l'utilisateur final. Les services de logiciel en tant que service fournissent au client des applications destinées à l'utilisateur final, ainsi que les ressources informatiques nécessaires pour l'exécuter. Dans ce type de service, les applications sont mises à la disposition des consommateurs, et peuvent être manipulées à l'aide d'un navigateur web ou installées de façon locative sur un PC, et le consommateur n'a pas à se soucier d'effectuer des mises à jour, d'ajouter des patches de sécurité et d'assurer la disponibilité du service [03].

**Autres services également disponibles :**

* Data as a Service : correspond à la mise à disposition de données délocalisées quelque part sur le réseau, Ces données sont principalement consommées par ce que l'on appelle des mashups.
* BPaaS : il s'agit du concept de Business Process as a service qui consiste à externaliser une procédure d'entreprise suffisamment industrialisée pour s'adresser directement aux managers d'une organisation, sans nécessiter l'aide de professionnels de l'informatique.
* Desktop as a Service : le Desktop as a Service ; aussi appelé bureau virtuel hébergé, est l'externalisation d'une Virtual Desktop Infrastructure auprès d'un fournisseur de services. Généralement, le Desktop as a Service est proposé avec un abonnement payant.
* Network as a Service (NaaS): le Network as a Service correspond à la fourniture de services réseaux, suivant le concept de Software Defined Networking (SDN).
* STorage as a Service (STaaS): correspond au stockage de fichiers chez des prestataires externes, qui les hébergent pour le compte de leurs clients, le plus souvent à des fins de sauvegarde ou de partage de fichiers.
* Workplace as a Service (WaaS): Espace de travail distribué.
* Communication as a Service (CaaS): correspond à la fourniture de solutions de communication substituant aux matériels et serveurs locaux des ressources partagées sur Internet [03].

## 2.2 Les types du Cloud Computing :

Un nuage (Cloud) peut être public, privé, hydride ou communautaire :

* Un **nuage public** est mis à disposition du grand public. Les services sont typiquement mis à disposition par une entreprise, qui manipule une infrastructure qui lui appartient.
* Un **nuage privé** est destiné exclusivement à une organisation, qui peut le manipuler elle-même, ou faire appel à services fournis par des tiers.
* Dans **un nuage communautaire**, l'infrastructure provient d'un ensemble de membres qui partagent un intérêt commun. Ce type de nuage est semblable à ceux montés par les milieux académiques pour des études de grande envergure. Le déploiement des applications y sera communautaire.

* **Le nuage (Cloud) hybride** (interne et externe) est un environnement composé de multiples prestataires internes et externes. Un exemple, IBM avait conclu un partenariat avec Juniper Networks. Cette association a permit à Big Blue de déployer son offre de **cloud hybride**. Ainsi les entreprises qui utilisent ce service peuvent faire basculer, par un simple glisser-déposer, des applications hébergées dans un nuage privé interne vers un nuage public sécurisé [03].

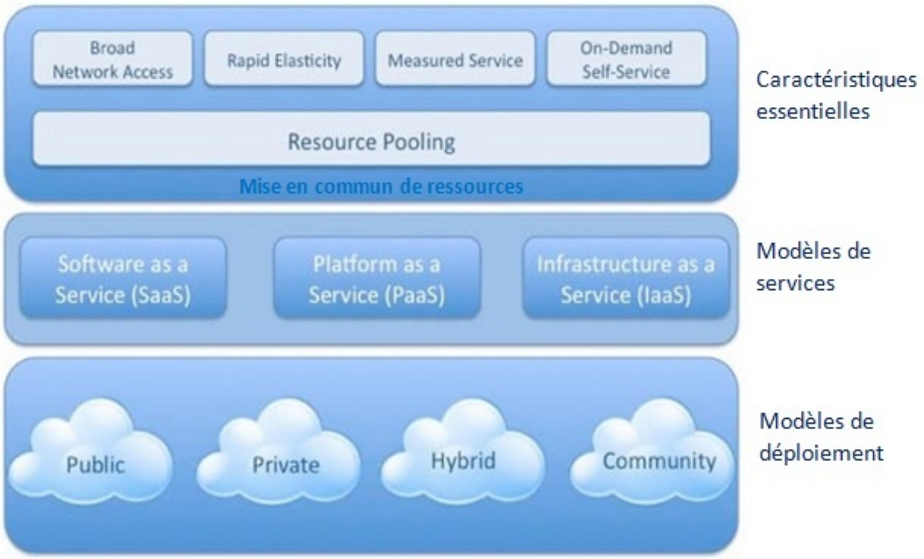



**Figure (1-3): caractéristiques, modéles et types de cloud computing.**

## 2.3 Les avantages du Cloud Computing :

Le Cloud Computing peut apporter des avantages considérables à la fois aux organisations et aux individus. L'avantage le plus évident est sans doute la réduction des coûts. Dans un centre de données exploité par un fournisseur de services d'un Cloud Computing, les différents clients ont besoin de ressources informatiques à des moments différents, ces fournisseurs sont en effet capables de réaffecter rapidement les ressources de ceux qui ont moins de besoins à un instant donné vers ceux qui ont des besoins plus importants [04].

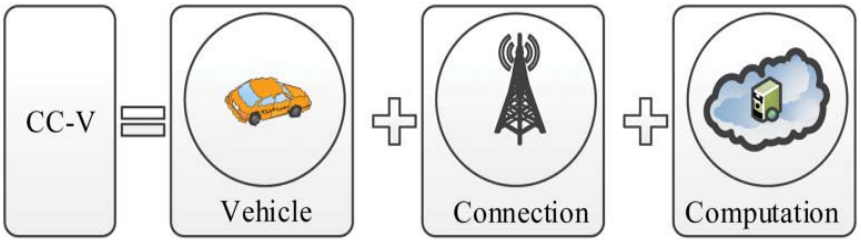
Cependant, le coût n'est pas le seul moteur de l'adoption des services d'un Cloud Computing. Les raisons les plus importantes sont la rapidité d'accès aux ressources dans le cloud et l'élasticité de l'aptitude des fournisseurs de services d'informatique en nuage à répondre à cette demande, cette élasticité dispensant les entreprises d'avoir à planifier des investissements considérables dans des matériels physiques et des logiciels. L'informatique en nuage permet aux entreprises de déployer rapidement leurs applications à grande échelle et de ne payer pour les services informatiques que lorsque la demande est effective. L'informatique en nuage peut également améliorer l'efficacité énergétique et réduire l'empreinte environnementale [04].

# 3 Les Cloud Computing Véhiculaire (CCV) :

## 3.1 Les concepts de base du CCV :

Les trois éléments principaux qui ont formé le CCV sont explorés avec la concentration sur l'aspect de l'architecture en couches proposée. Dans une pointe de vue physique et structurelle, les composants y compris le véhicule, la connexion et la computation représentent les aspects réseau de CCV. Les véhicules : sont des utilisateurs finaux et les services sont donc livrés aux véhicules. Cette composante est liée à la couche perception. Cela est dû à différents types de capteurs de surveillance pour contrôler la vitesse, la direction et la position des véhicules. Le second composant à savoir, la connexion : comprend les dispositifs de réseau ou de communication des réseaux hétérogènes. Ce composant est lié à la couche de coordination. Le dernier composant appelé computation : fait référence aux dispositifs de calcul et de stockage, responsables du traitement efficace des données de trafic volumineux et l'inférant des décisions intelligentes. Cette dernière composante est étroitement liée à la couche d'intelligence artificielle. Cela est dû aux fonctionnalités associées de la couche. La relation entre le véhicule et la connexion détermine l'efficacité de la livraison et de l'acceptation des services fournis. La relation entre la connexion et le computation déterminent la qualité de l'information des services. Chaque élément et son rôle dans CCV sont définis ci-dessous [05]:

**Figure (1-4): Les éléments de CCV.**




### 3.1.1 Le véhicule :

Les véhicules disposent plusieurs technologies et ressources intégrées, qui sont sous-utilisées dans l'architecture de communication de véhicule traditionnelle VANETs. Les technologies et les ressources comprennent les dispositifs télématiques filaires (TWD, Telematics Wire Devices), le GPS, les capteurs, les communications dédiées à courte portée (DSRC, Dedicated Short-Range Communication), les ordinateurs et les smart phones. L'utilisation de ces ressources peut être améliorée grâce à une architecture de réseau coopérative et collaborative. La sous-utilisation des ressources motive le concept de CCV. en raison au fait que les véhicules ne rencontrent pas certains des problèmes des périphériques mobiles dans le MCC, tels que les faible puissances et les moins capacités de calcul, les mémoires de volume moins importante et les batteries d'un durée de vie courte. Les ressources des véhicules peuvent être partagées dans le plateforme de CV à de nombreux endroits tout en voyageant, il comprend un parking, un garage et un feu de circulation. En cas de besoin de calcul important pour une durée plus longue, les technologies intégrées du véhicule peuvent être utilisées pour établir une connexion durable à un cloud conventionnel VuC [05].

### 3.1.2 La connectivité :

La connexion fait référence au réseau ou à la dispositif de communication de l'architecture de réseaux hétérogènes considérée en CCV. Les périphériques réseau sont utilisés pour établir une communication fiable entre les véhicules et le cloud. Dans le cas de VuC, la connexion est soit une communication directe entre les véhicules et l'infrastructure cloud, soit une communication multi-sauts utilisant des unités au bord de la route (RSU) avec les véhicules. La connexion définit également un accord de niveau de service (SLA) entre un client véhiculaire et une infrastructure cloud. Cela est dû au fait que le SLA représente le niveau de qualité de service (QoS) d'une application basée sur l'architecture de CCV. Par conséquent, la connexion détermine la livraison et la réception des services [05].

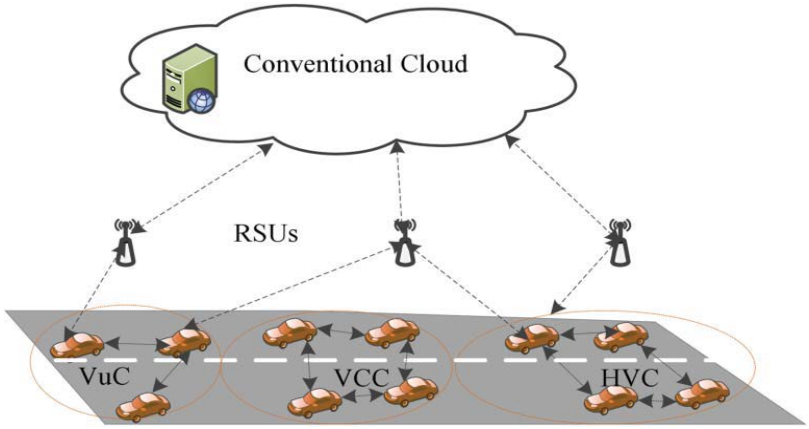
### 3.1.3 Le Calcul (Computation) :

Le calcul fait référence aux dispositifs de calcul et stockage dans lesquels les données de trafic volumineux sont traitées et où les décisions intelligentes sont inférées. Le calcul est divisé en trois modèles: fournisseur de calcul, consommateur de calcul et hybride. Dans le modèle de fournisseur de calcul, les véhicules doivent s'inscrire auprès du fournisseur de services pour utiliser leurs ressources de véhicule dans une architecture de cloud computing utilisant un accord de niveau de service (SLA). Dans le modèle de consommation de calcul, les véhicules doivent s'inscrire pour utilisant les services de cloud computing comme leurs propres ressources. Dans le cas du modèle hybride, les véhicules doivent être enregistrés en tant que fournisseur et consommateur à la fois. Le modèle de fournisseur générerait des revenus en améliorant l'utilisation des ressources. Le modèle consommateur améliorer la capacité de calcul des véhicules en utilisant une infrastructure cloud. Le modèle hybride serait plus complexe en raison de la nécessité de maintenir l'état de fournisseur ou de consommateur pour chaque véhicule [05].

## 3.2 Les plateformes CCV :

Une architecture a été proposé focalisé sur l'utilisation de services traditionnel de cloud computing dans les véhicules et comment améliorer la communication entre le cloud conventionnelle et le cloud véhiculaire. Un concept qui fusionne les réseaux VANETs avec le Cloud Computing (MVCC, Merges VANETs with Cloud Computing) a été suggéré pour remédier la sous-utilisation des dispositifs intégrés en véhicules et le manque d'architecture standard pour les systèmes (CCV). Les dispositifs intégrés ont les capacités suivantes: calcul, communication et stockage d'informations. Une architecture pour le cloud computing dans la communication véhiculaire a été définie et divisée en trois plateformes architecturaux comprenant: Les plateformes Cloud Véhiculaire (CV), Les plateformes VANET-Cloud (VuC) et Les plateformes Cloud Véhiculaires Hybrides (HVC) [05].

**Figure (1-5): Les plateformes CCV**




### 3.2.1 Le plateforme Cloud Véhiculaire (CV) :

En CV, un groupe de véhicules connectés forme un nuage dans le but de partager leurs ressources abondantes. CV augmente l'utilisation des ressources dans la communication entre les véhicules. Les ressources abondantes comprennent le stockage, la capacité de calcul, la détection et la capacité de communication. Les ressources sont devenues disponibles pour le partage lorsque plusieurs véhicules se sont rassemblés sur des places tell que des parcs de stationnement, des garages, des barrages routiers et des feux de signalisation. À ces endroits, l'idée de CV a été réalisée. CV alloue dynamiquement des ressources abondantes aux utilisateurs autorisés [05].

### 3.2.2 Le plateforme VANET-Cloud (VuC, Vehicle using Cloud) :

VuC permet l'accès aux services de Nuage conventionnel aux véhicules via l'internet. VuC aide à la réalisation d'applications ITS intelligentes, y compris les services de prévision de trafic en temps réel et les services Web [05].

### 3.2.3 Le plateforme Cloud Véhiculaires Hybrides (HVC) :

Tout simplement ce plateforme Combiner les caractéristiques de la plateforme Cloud Véhiculaire (CV) et la plateforme VANET-Cloud (VuC), Afin d'obtenir le plus grand bénéfice possible.

## 3.3 L'architecture En Couches de CCV:

Une architecture à quatre couches pour CCV est conçue en termes de représentation, de fonctionnalités et de protocole. Dans une perspective opérationnelle et structurelle, CCV comporte quatre composantes principales : les dispositifs physiques de perception, la coordination, l'intelligence artificielle et les services d'application intelligente. Par conséquent, l'architecture se compose de quatre couches, y compris perception, coordination, intelligence artificielle et application intelligente [05].

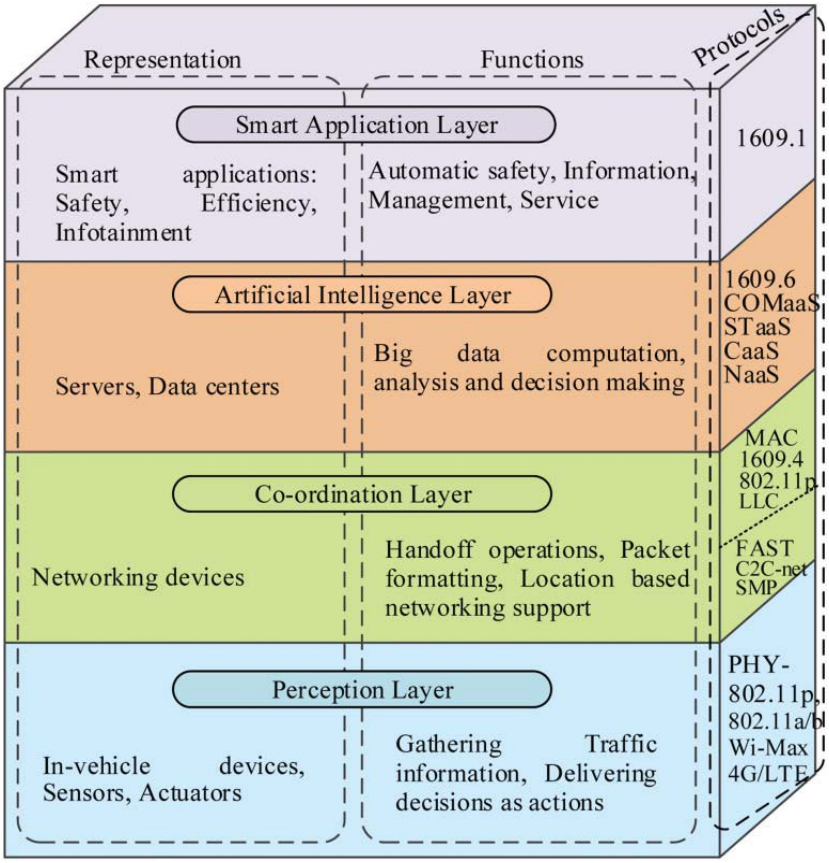



**Figure (1-6): L'architecture En Couches de CCV**

Les couches sont différenciables en termes de fonctions et de représentations. Chaque couche est décrite ci-dessous :

### 3.3.1 Couche de perception :

La couche de perception de l'architecture représente les dispositifs intégrés en véhicule, comprenant : les capteurs, les unités d'affichage, les smart appareils mobiles, les récepteurs du système de positionnement global (GPS), etc. Les deux fonctions principales de la couche incluent la collecte de données sur le trafic par la détection et aussi la livraison des informations à l'utilisateur final. La couche définit les interactions entre le véhicule et le monde physique à travers des dispositifs intelligents. En plus de ces fonctionnalités, un certain nombre de services sont fournis par la couche de perception à la couche suivante en tant que services d'interface. Les services d'interface incluent la détection et la correction des erreurs au niveau des données détectées, et la vérification des résultats des informations inférées, etc. En termes de protocole, la couche physique fait partie des protocoles : IEEE 802.11p (PHY-802.11p) de WAVE, 802.11a / b / g de WLAN, Wi-Max et 4G / LTE, Ceci est en raison de dispositifs hétérogènes intégrés en véhicule contenant différents types de capteurs [05].

### 3.3.2 Couche de coordination :

La couche de coordination représente les périphériques et les technologies d'interconnexion de réseaux. La coordination du réseau est importante en raison de la prise en compte de différents types de réseaux, notamment les réseaux VANET, Wi-Fi et 4G / LTE. Les fonctionnalités principales de la couche incluent le transfert de paquets de données et la livraison fiable entre ces réseaux. Les autres fonctionnalités de la couche incluent la dissémination des données entre les réseaux hétérogènes, le niveau d'authentification en réseau et l'autorisation, l'architecture de réseau hétérogène rend ces fonctionnalités assez compliquées. En termes de protocole, deux sous-couches sont considérées. Dans la première sous-couche, la couche de contrôle d'accès au support (MAC) contenant les protocoles : IEEE 1609.4 de WAVE, 802.11p et LLC. Dans la deuxième sous-couche, comprenant les protocoles de couche réseau et transport, les liaisons de voiture à voiture (C2C) et le protocole de message court (SMP) dans WAVE sont considérés séparément des combinaisons IP et TCP / UDP traditionnelles [05].

### 3.3.3 Couche d'intelligence artificielle :

La couche d'intelligence artificielle représente une infrastructure basée sur le cloud et le cloud computing. Cela inclut les centres de données et les serveurs du cloud conventionnel comme le (VuC). Les ressources de calcul des véhicules sont également incluses dans le cas du (CV). Les fonctionnalités de la couche incluent le calcul, l'analyse et l'inférence des informations à partir des données volumineuses (Big Data) pour des décisions intelligentes à la bénéfice des applications en temps réel. Outre le cloud computing et la prise de décision, des autres fonctionnalités de la couche comprennent : l'exploitation de données de trafic (data traffic mining) pour de nouveaux modèles business, assurer l'efficacité de calcul, la planification de computing et le calcul en parallèle, etc. Les opérations de la couche sont très similaires à celles des services en cloud. La couche améliore considérablement les capacités de calcul des véhicules grâce à l'architecture VuC. Il améliore également l'utilisation des ressources des véhicules grâce à l'architecture CV. Du point de vue de protocole, les protocoles liés au service, à l'analyse de données volumineuses (BDA, big data analysis). Les protocoles incluent IEEE 1609.6 de WAVE, COaaS, STaaS, Image en tant que service (PICaaS), Infrastructure en tant que service (INaaS), CaaS, NaaS et Gateway en tant que service (GaaS) [05].

### 3.3.4 Couche d'application intelligente :

La couche application intelligente représente des applications intelligente basé sur le cloud et a trois catégories: la sécurité, l'efficacité et l'infotainment. La couche est responsable de livraison des services intelligents à l'utilisateur final. Les fonctionnalités de la couche incluent la gestion des applications, la gestion des services, la gestion de données basée sur les services et les applications, l'authentification et l'autorisation basée sur les applications. Bien que les applications de sécurité et d'efficacité soient également implémentées dans les VANET, le fonctionnement en cloud améliore considérablement l'intelligence et l'utilité de ces applications, cela est dû à l'évolution de la puissance de calcul des applications fonctionnant via le cloud. Certaines des applications intelligentes incluent la prévision de trafic en temps réel, la collection intelligent des frais via le

paiement en voiture, le règlement intelligente pour la violation des règlementations de circulation, le partage ad hoc multimédia et les appels d'urgence intelligents. En termes de protocole, le protocole de gestionnaire de ressources IEEE 1609.1 de WAVE est considéré pour la gestion efficace des ressources. Outre, les modèles de business liés à la publicité, la vente, le service et l'assurance sont pris en compte [05].

## 3.4 Les applications CCV:

Les applications CCV sont beaucoup plus essentielles pour la réalisation d'une communication VANET opérationnelle et efficace basée sur le cloud computing. Les applications CCV peuvent être divisées en trois catégories : sécurité, efficacité et infotainment. Les applications de sécurité sont créées pour améliorer la prise de conscience du comportement des véhicules, afin d'éradiquer ou de réduire les accidents de véhicules via la communication V2V. Ses applications incluent l'avertisseur de perte de contrôle, les feux de freinage électroniques de secours, l'avertissement d'angle mort / changement de voie, etc. Les applications de communication V2I incluent l'avertissement de véhicule surdimensionné et d'alerte de vitesse en virage. La communication entre les véhicules et le piéton (V2P) inclut une indication de piéton de transport en commun [05].

### 3.4.1 Sécurité :

Les transports et les communications jouent un rôle essentiel dans la gestion et l'intervention en cas de catastrophe afin de combattre ou de réduire les pertes de vies humaines et de biens. Un modèle de système d'intervention d'urgence en cas de catastrophe (EDRS**,** an Emergency Disaster Response System) a été introduit. Le système ERDS se compose des trois couches principales. Les couches comprennent l'infrastructure en tant que services, l'intelligente et l'interface système. La couche infrastructure en tant que service comprend un plateforme de base et un système de réponse d'urgence intelligent écologique. La couche intelligente fournit un modèle de calcul et des algorithmes. La couche interface système acquiert des données à partir de passerelles telles que l'internet, les mâts en bordure de route, les téléphones intelligents mobiles et les réseaux sociaux [05].

### 3.4.2 Efficacité :

Ont été proposé des réseaux de cloud sécurisés et fiables pour les services de transport intelligent (SCST, **S**ecure and reliable **C**loud networks for **S**mart **T**ransportation) destinés aux préventions des accidents, de surveillance et systèmes de contrôle. Le système de transport intelligent comporte quatre couches fonctionnelles, à savoir la couche application, la couche support, la couche réseau et la couche perceptuelle.

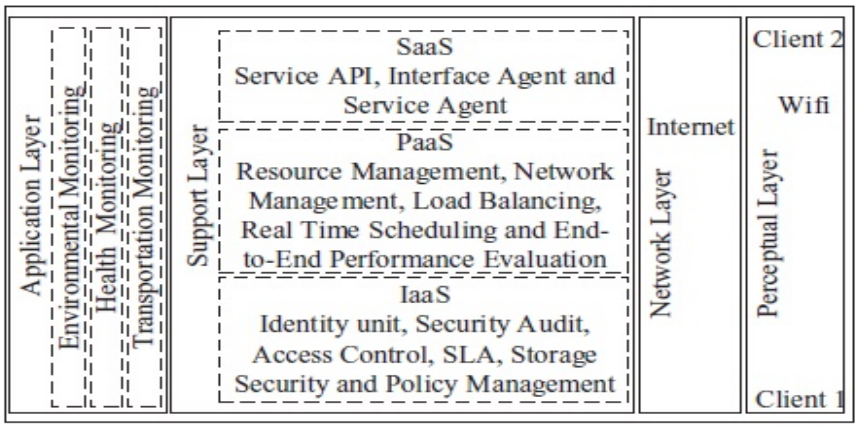



**Figure (1-7) : Les quatre couches fonctionnelles de STI.**

La couche application représente des applications utilisateur diverses. La couche support couvre les services de cloud computing. La couche réseau implique l'internet pour la connexion et la couche perceptuelle est la couche client. Un algorithme de procédure de détournement de véhicule dans un système de transport intelligent a été développé. L'algorithme a été utilisé pour résoudre le problème de routage de cloud computing aux temps d'arrêt. Les fonctions du système de transport intelligent incluent la prévention des accidents, la recherche de la destination et le transfert des informations concernant les accidents aux véhicules utilisant le cloud. Un concept de services en temps réel basé sur le cloud computing pour le bénéfice des réseaux de véhicules (RTCV, A **R**eal **T**ime services concept for future **C**loud computing- enabled **V**ehicle) en futur ont été suggérés pour assurer des performances en temps réel ainsi que pour améliorer le degré de confort des conducteurs. Les services de cloud véhiculaire en temps réel sont introduits sous forme de surveillance du trafic routier et des soins de santé, ainsi que d'autres services personnalisés [05].

### 3.4.3 Infotainment (Infotainment) :

Les STI basés sur le cloud (CITS, **C**loud-based **ITS**) ont été suggérés pour résoudre le problème croissant des transports à l'aide des applications d'infotainment. Un système de nuage de données véhiculaire multicouches a été présenté. Le système utilise les technologies de cloud computing et l'internet des objets (IoT). Le système comporte trois modules, à savoir un service cloud de stationnement intelligent, la communication des réseaux VANET vers le cloud et un cloud véhiculaire de data mining. Le module cloud de stationnement intelligent prend en charge le processus de décision consistant à sélectionner un emplacement de parking disponible pour les véhicules, et l'appareil mobile avec une application de service Android pour la communication avec le cloud. Le système a une interdépendance plus élevée entre les modules, ce qui pourrait dégrader les performances du système [05].

Les services multimédias sont devenus l'un des principaux domaines d'intérêt de la recherche en cloud computing et les VANET en raison de leur pertinence d'infotainment et de sécurité. Ainsi, les services multimédias des réseaux cloud véhiculaires (MSCVN, **M**ultimedia **S**ervices in **C**loud-based **V**ehicular **N**etworks) ont été utilisés pour intégrer le cloud computing et le

stockage avec les véhicules, afin d'accroître l'accessibilité aux services multimédias. Différents systèmes, notamment le système LTE pour l'accès au réseau et le système multimédia de cloud computing, ont été suggérés. Toutefois, la connectivité peut être retardée en raison de la taille importante des fichiers audio et vidéo à transmettre [05].

## 3.5 Les défis et les besoins futurs des applications CCV :

Le CCV est un nouveau paradigme qui combine l'idée du cloud computing et les VANET. De nombreuses questions de recherche doivent être abordées pour réaliser le CCV:

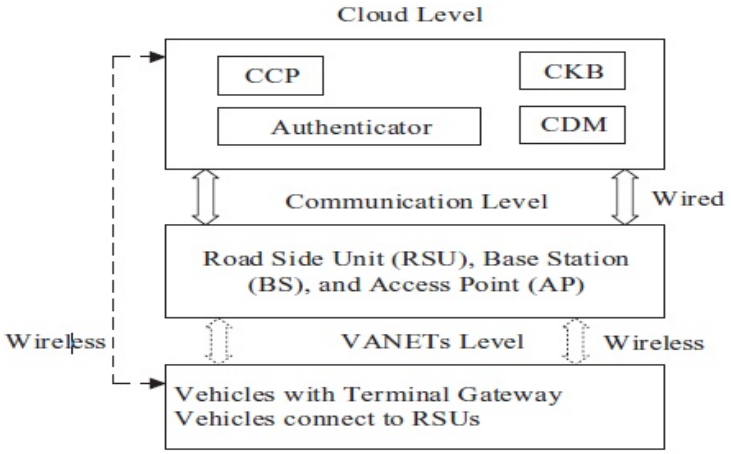
### 3.5.1 Sécurité:

Les principaux défis de la sécurité en CCV incluent la confidentialité, la détection d'intrusion et l'authentification. La confidentialité en CCV c'est la communication avec les véhicules voisins en utilisant les informations de localisation, de vitesse et de direction, sans dévoiler l'identité de l'autre. La détection d'intrusion est par l'identification du véhicule voisin non-coopératif. L'authentification en CCV est la vérification de la base du véhicule voisin sur une certaine réputation. Les défis de sécurité des applications intelligentes sont principalement liés à des problèmes d'authentification, notamment la sécurité automatique, la gestion des informations et l'authentification des services. Pour l'intelligence artificielle, les principaux problèmes en matière de sécurité concernent la détection d'intrusion sur les serveurs et les centres de données ou les véhicules qui forment un nuage lorsqu'une demande de calcule d'un client externe est envoyée aux machines. La couche de coordination peut être utilisée comme un canal pour l'intrusion et aussi pour l'authentification. Par conséquent, les problèmes de détection d'intrusion et d'authentification peuvent être résolus efficacement au niveau de cette couche. La couche de perception constitue les dispositifs intégrés dans le véhicule et les nœuds du véhicule. Le défi majeur de cette couche est donc la détection d'intrusion et la confidentialité. Avec la mise en œuvre de l'architecture en couches CCV, une solution plus sécurisé peut être atteint [05].

#### 3.5.1.1 Confidentialité:

Ont été suggéré un conscient des informations de trafic confidentiels et sécurisées en tant que service (TIaaS, Traffic Information as a service), pour résoudre le problème de la confidentialité d'utilisateur du véhicule, La question de la confidentialité est l'une des préoccupations majeures pour lesquelles de nombreux utilisateurs de véhicules ne souhaitent pas participer au partage leurs informations entre les véhicules. Le modèle de réseau de TIaaS est présenté en figure (1-8).

**Figure (1-8) : Le modèle de réseau de TIaaS**

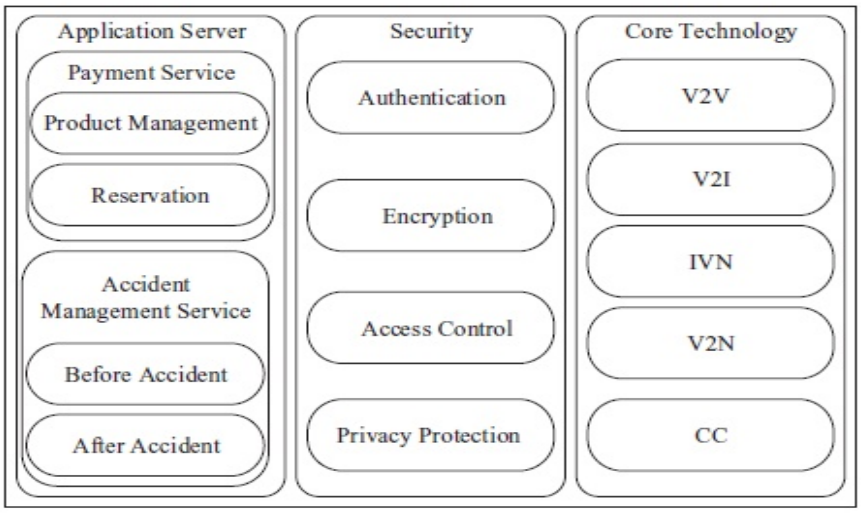



Les problèmes de confidentialité cause une utilisation moindre de l'infrastructure VANET, notamment en matière de communication, de calcul et de stockage. Un mécanisme de révocation, un concept de client léger qui profite de TIaaS pour les véhicules et une plateforme de vecteurs de mobilité efficaces ont été conçus. Le modèle TIaaS a utilisé le plateforme VuC, Il fournissait des informations de trafic raffiné aux véhicules depuis le cloud grâce à la coopération des abonnés avec le cloud de manière sécurisée, Malgré sa force dans la dissimulation d'informations de localisation, le temps de traitement pourrait être plus long en raison du schéma de chiffrement adopté. Le TIaaS est un système de sécurité qui s'applique le mieux aux fonctions de couche de perception et de coordination de l'architecture en couches CCV [05].

### 3.5.1.2 Détection d'intrusion:

Ont été proposé aussi une plateforme de service de sécurité orientée visant à résoudre le problème de l'insuffisance de la sécurité interne ou externe de l'infrastructure des véhicules, et le fuite des informations provenant de capteurs intégrés aux véhicules. La structure est illustrée en figure 1-9 [05].

**Figure (1-9) : La plateforme de service de sécurité**



Le plateforme suggéré a été utilisé pour gérer des services de gestion du paiement et de prévention des accidents orienté aux l'utilisateur finale. En outre, le plateforme a fourni le chiffrement, l'authentification, le contrôle d'accès, la confidentialité, l'intégrité et la protection des informations privée relatives aux utilisateurs et véhicules. La plateforme est composée de deux modèles: un proactive et un autre réactive. Le modèle proactive a utilisé des capteurs fixés à des véhicules pour surveiller l'état et la santé du conducteur et ses capacités de conduite. Toutefois, ce plateforme ne pas être économiquement viable en raison de sont complexité et des ces exigences. Les résultats peuvent être considérés comme très proches des fonctionnalités des couches : applications intelligentes, l'intelligence artificielle et coordination de l'architecture à couches CCV.

#### 3.5.1.3 Authentification:

Il a été suggéré de protéger les cloud véhiculaire contre les nœuds malicieuse (PVCM, Protecting Vehicular Cloud Against Malicious) utilisant les autorités de la surface afin de résoudre le problème de faiblesse de protection des connexions CCV, ainsi que des menaces sur les données, les ressources et les services. Une infrastructure sécurisée utilisant la technique de gestion de clés et de révocation pour sécuriser les connexions CCV contre les nœuds malveillants a été présenté. L'infrastructure est décentralisé et utiliser plusieurs autorités de zone. Chaque autorité de surface gère une surface appelée zone qui composée d'RSUs, les véhicules et les clients de cette zone. Chaque autorité de zone considérée comme une passerelle qui authentifie les actions de cette zone. Il gère les demandes de service et le flux de données, et préserve la confidentialité des entités du cloud, y compris les véhicules et les clients. Dans ce type de protection de nœud, trop d'authentifications sont effectuées lors du déplacement d'une zone à une autre, et cela pourrait dégrader la communication entre les véhicules. Les résultats sont considérés comme proches des fonctionnalités de la couche de coordination et d'application intelligente de l'architecture à couches CCV [05].

### 3.5.2 Des autres défis de recherche en future :

**(1) Conception d'architecture :**

Étant donné que CCV est encore un nouveau domaine de recherche, il n'existe pas une architecture standard générale pour cette idée nouvelle. Bien que de nombreuses architectures initiales pour CCV été suggérées, des architectures standard avec des détails de mise en œuvre ne sont pas disponibles. Par conséquent, la question nécessite une exploration approfondie.

**(2) dissémination des données :**

La dissémination efficace des données en CCV est une question délicate, en raison de la haute dynamique des véhicules en changement de leurs positions. La conception et la manière de transmettre les données dans CCV doivent être abordées de manière critique. Même si certains travaux ont été faits en suggérant des solutions pour traiter divers types de problèmes de diffusion de données à l'aide de la mise en cluster, centrée sur les données et autres approches de routage, mais de nombreuses questions n'ont pas encore été abordées, telles que la vérification de la localisation, la diffusion des données et le routage centré sur la vidéo. Le routage en VANET

conventionnel peut ne pas convenir dans le cas de CCV en raison des problèmes de connectivité liés à la mobilité.

**(3) déchargement des données :**

Les données complexes non raffinées doivent être transférées vers un cloud conventionnel ou un cloud véhiculaire pour le traité. Après le traitement, ces données raffinées sont accessibles par les véhicules et par des autres organisations. Cependant, la question qui doit être examinée ces comment décharger les données non raffinées et accéder aux données affinées dans un scénario de véhicule à grande mobilité. L'agrégation et le calcul des données sont également nécessaires car les véhicules utilisent des capteurs et d'autres dispositifs intégrés pour la collection des données relatives au véhicule, au l'environnement et au trafic. Par conséquent, la question de l'agrégation et du calcul des données est clairement abordée au niveau du véhicule et cloud. Les données doivent être agrégées et raffinées pour l'utilisation finale.

**(4) Conception et déploiement d'applications :**

Certaines applications VANET doivent encore être conçues ou déployés sur les véhicules pour une utilisation efficace, tels que la couverture vidéo et Image en tant que service (PVCaaS) en CCV. PVCaaS est un service permettant aux véhicules sur la route de prendre des photos et de filmer leurs environs. La photo et la vidéo sont automatiquement déchargées vers le cloud pour être stockées et analysées. Ce service est utile pour l'organisation de la sécurité routière. Certains des travaux récents abordent certaines conceptions d'applications pour la gestion des catastrophes, le contrôle du trafic routier et les applications multimédias.

**(5) Normalisation et interopérabilité :**

Les véhicules sont dotés de dispositifs hétérogènes, tels que des capteurs, des GPS et des smart phones. Le faire des dispositif variés fonctionner ensemble efficacement est hautement nécessaire pour la collecte de données. La normalisation est également nécessaire pour ces dispositifs intégrés, en s'intéressant à la compatibilité, la qualité, la mise en œuvre des directives, l'interopérabilité et à la répétabilité des équipements et logiciels intégrés.

**(6) Retard dans le cloud - communication client :**

C'est l'un des problèmes fondamentaux de tout service basé sur le cloud en raison de l'environnement réseau dynamique et de la prise en compte de l'infrastructure en nuage dans un environnement de véhicules à forte mobilité. La distribution clairsemée des véhicules et la nature dynamique de la densité des véhicules dans l'environnement du réseau sont inévitables. CCV nécessite une décision de communication en temps réel dans les applications de sécurité, ce qui est assez difficile compte tenu du problème de retard en cas d'accès au réseau.

**(7) conduite autonome :**

C'est l'un des aspects des STI qui nécessite une intelligence artificielle, des capacités d'apprentissage et du stockage pour prendre une décision de calcul sur la route à suivre pour une navigation rapide et sûre. Le cloud computing doit être appliqué au calcul des données intelligentes et par la suite, à l'analyse, qui devrait être accessible en véhicule autonome.

**(8) Stockage de données basé sur l'apprentissage :**

C’est un autre aspect des STI qui est nécessaire pour obtenir des STI distribués. Il nécessite un stockage initial des informations dans les VANET sans aucune disposition pour la configuration du périphérique. La configuration du périphérique peut être RSU ou un point d'accès externe. Par conséquent, il est nécessaire de renforcer de manière robuste les techniques de stockage de données dynamiques et adaptatives basées sur l’apprentissage, pour les réseaux VANET afin de réaliser des STI distributifs [05].

# Conclusion:

Dans ce chapitre, nous avons présenté, les systèmes de transport intelligents (STI) et les réseaux véhiculaires VANETs, Le paradigme Cloud Computing et les Cloud Computing Véhiculaire (CCV). Le chapitre suivant sera consacré pour les théories de flux de trafic et la simulation des réseaux VANETs.

# Chapitre 02 :

# Les théories de flux de trafic et la simulation des réseaux VANETs

# Introduction *:*

La congestion de la circulation automobile dans les zones urbaines est un problème rencontré dans le monde entier. Il a des effets négatives sur la qualité de vie des personnes en raison des retards, des accidents et de la pollution de l'environnement. Un moyen d'éliminer le problème consiste à augmenter la capacité des routes existantes en ajoutant des voies. Cependant, cela est grandement entravé par le manque d'espace, de ressources ou par des problèmes parfois politiques. Cela laisse aux autorités une option majeure: améliorer l'utilisation des infrastructures existantes en utilisant de meilleures stratégies de gestion et d'exploitation du trafic. Pour une gestion et un contrôle efficaces du trafic, une bonne compréhension de la congestion du trafic est nécessaire. Pour atteindre ce dernier objectif, il convient d'étudier de près le comportement spatio-temporel de schémas empiriques de congestion du trafic. En effet, on observe que les embouteillages se produisent dans l'espace et dans le temps sous la forme de structures de trafic spatio-temporel encombrées qui se propagent sur les routes. Les observations empiriques indiquent que les embouteillages sur un réseau routier sont une conséquence de la rupture du trafic dans un trafic initialement libre. L'arrêt de la circulation est le soudain déclin de la vitesse passant de valeurs élevées en écoulement libre à des valeurs inférieures en cas de trafic congestionné, et se produit normalement aux goulets d'étranglement tels que les sorties de voie et les accidents.

Maintenant que l'origine de la congestion du trafic est connue, il est essentiel de bien décrire les situations à l'origine de la congestion sur les réseaux routiers. Ceci s'appelle les théories et la modélisation de flux de trafic dont l'objectif est de décrire de manière mathématique précise les interactions des véhicules entre eux, et aussi les interactions entre les véhicules et l'infrastructure.

Alors, dans ce chapitre nous allons expliquer les théories de flux de trafic avec certain détail, et nous allons parler aussi de simulateurs de toutes sortes, et comment simuler un réseau VANET.

**PART I :**

# 1 Les caractéristiques du flux de trafic:

On peut diviser les variables du flux de trafic en deux parties distinctes, variables secondaires et variables fondamentales:

## 1.1 Les variables secondaires:

Les variables secondaires décrivent le comportement individuel d'un véhicule ou l'interaction entre deux véhicules en circulation routier, et est représenté en: la vitesse instantanée (v), le spacing (s) et le temps d'avance (h : headway en anglais).

**La vitesse instantanée (v):** La vitesse du véhicule instantanée est également mesurée sur des sections courtes, telles que la distance entre deux points rapprochées (6 m), auquel cas on n'a plus la vitesse instantanée du véhicule, mais une approximation proche de celle-ci (sauf en cas d'accélération ou de décélération rapide).

**Le spacing (s):** Est-ce que la distance entre le véhicule et le véhicule devant elle, y compris la longueur du véhicule. Donc la distance entre le pare-chocs arrière du véhicule et le pare-chocs arrière du véhicule suivant.

**Le temps d'avance (h):** De même, nous pouvons prendre le temps qu'un suiveur arrive à atteindre (avec son pare-chocs avant) la position du pare-chocs arrière, on ajoutons aussi le temps nécessaire pour couvrir la longueur du véhicule.

## 1.2 Les variables fondamentales:

Les variables fondamentales décrivent le comportement d'un flux de trafic de manière globale, et est représenté en: la Vitesse moyenne (u), la densité (k) et le débit (q).

**La Vitesse moyenne (u) :** La vitesse moyenne est basée sur le temps moyen T d'un nombre de véhicules N pris pour traverser une distance D donnée.

$$u = \frac{D}{T} \quad , \quad T = \frac{1}{N}\sum_{i}^{N} t$$

**La densité (k) :** La densité nous permet d'avoir une idée de l'encombrement d'un certain tronçon de route. Il est généralement exprimé en nombre de véhicules par kilomètre (ou mile).

$$k = \frac{n}{L}$$

Notez que le concept de densité ignore totalement les effets de la composition du trafic et de la longueur des véhicules, car il ne considère que la quantité abstraite 'nombre de véhicules'.

**Le débit (q) :** Alors que la densité est généralement une mesure spatiale, le débit peut être considéré comme une mesure temporelle. Le débit, que nous utilisons comme raccourci pour le

flux, s'exprime généralement en un taux horaire, c'est-à-dire en nombre de véhicules dépassant un point de référence par unité de temps. $q = \frac{n}{T}$

Il y a une identité fondamentale d'un flux du trafic représenté par la formule suivante qui contient les trois variables clés (débit, densité, vitesse moyenne) : q = k * u, Connaître deux variables permet d'obtenir la troisième, Donc, ces trois variables constituent une identité fondamentale cohérente.

# 2 Le diagramme fondamental du trafic:

Comme dans de nombreuses disciplines scientifiques, les théories qui en résultent sont souvent précédées par une enquête sur les données expérimentales obtenues, ce qui leur sert de preuve empirique. Dans cette ligne de raisonnement, Greenshields a été l'un des premiers à fournir une base de la plupart des travaux classiques sur ce que l'on appelle le diagramme fondamental empirique [06]. En raison de la relation fondamentale q = k.u que nous avons vue auparavant, une compréhension qualitative sera donnée, après quoi l'effet sera montré pour divers couples de variables clés (débit, densité, vitesse moyenne). Et étudier les valeurs expérimentales de ces couples de notre trois variables et leurs résultats sont les suivants:

## 2.1 Vitesse moyenne vs densité:

Il est un fait observable que les conducteurs réduisent leur vitesse à mesure que le nombre de voitures les entourant augmente. En raison de cette interaction étroite entre la densité et la vitesse, et connaissant à la fois la densité et la vitesse, à partir duquel le flux peut être calculé, il n'est pas surprenant que les enquêteurs aient exploré les relations entre la vitesse et la densité. La plus simple de ces relations est une relation linéaire.

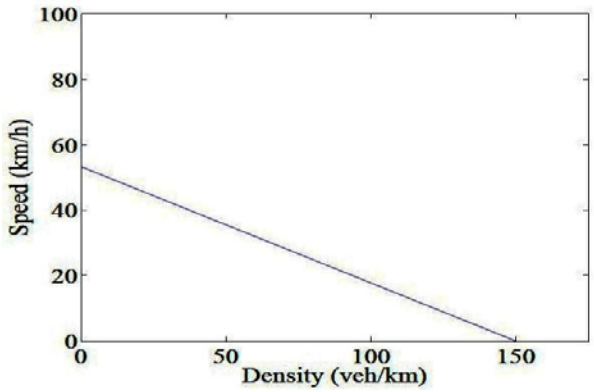
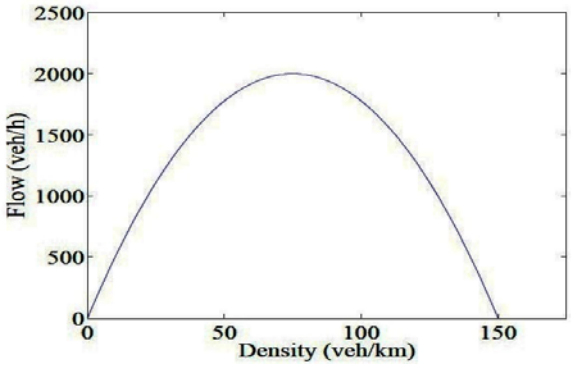



**Figure (2-1)** : **Graphe de (vitesse moyenne/densité)**

## 2.2 Débit vs densité:

Les premières études sur la capacité des autoroutes ont suivi deux approches principales. Certains chercheurs ont examiné les relations vitesse-débit à de faibles densité, d'autres ont discuté des phénomènes de temps d'avance (headway) à fortes densité. Les chercheurs ont proposé d'utiliser la courbe de débit-densité pour unifier ces deux approches. En raison de cette caractéristique unificatrice et de la grande utilité de la courbe de débit-densité dans les situations de contrôle du trafic (telles que le mesure de longueur d'une autoroute), les connaisseurs appelées la courbe densité-débit: "le diagramme de base du trafic".

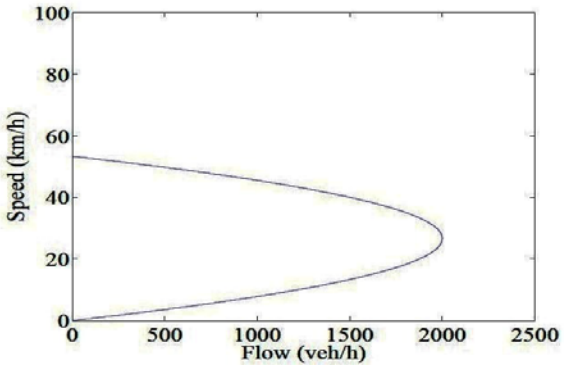
**Figure (2-2)** : **Graphe de (débit/densité)**




## 2.3 Vitesse moyenne vs débit:

Une fois qu'un modèle de vitesse-densité a été déterminé, un modèle de vitesse-débit peut en être déterminé. Dans tous les modèles réalistes de vitesse-densité, la vitesse en débit libre à une densité de zéro est la vitesse maximale pouvant être atteinte. Ainsi, le point le plus élevé de la courbe vitesse-débit sera le point à vitesse de débit libre et à débit nul. Dans la mesure où les valeurs de débit sont les produits des valeurs de vitesse et de densité correspondantes, il y aura un deuxième point de débit nul, correspondant à la vitesse zéro (densité maximale). Ainsi, quelle que soit la forme de la courbe vitesse-densité, la courbe vitesse-débit aura un point à l'origine et un point à la vitesse axe à la valeur maximale de la vitesse. Entre zéro et la vitesse maximale, le diagramme formera un type de boucle vers le débit maximal. Si la courbe vitesse-densité est une ligne droite, comme suggéré par Greenshields, la courbe vitesse-débit résultante est une parabole.

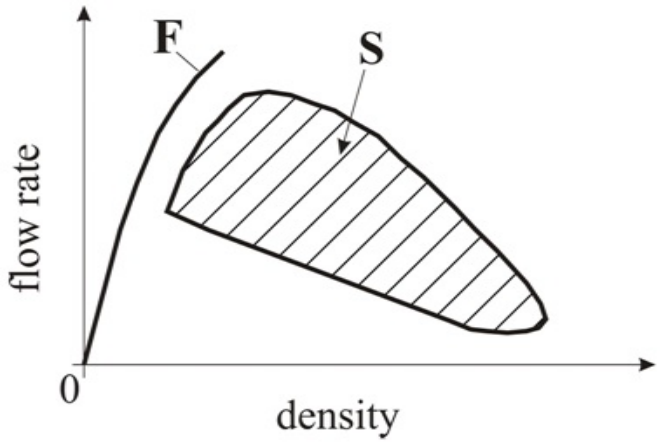
**Figure (2-3)** : **Graphe de (Débit vs densité)**




# 3 Le flux de trafic a trois phases:

Selon débit vs densité diagramme qui s'appelle le diagramme de base du trafic comme nous l'avons dit auparavant. Parmi des nombreuses questions soulevées par Kerner (2004), il y a trois phases par lesquelles le trafic passe (la phase libre, la phase capacité (synchronisé) et la phase congestionnée) plutôt que deux (la phase libre et la phase congestionnée) [07], comme disent d'autres théories. Les trois phases sont les suivantes:

**Figure (2-4) : Le graphe de flux de trafic a trois phases.**




## 3.1 La phase libre:

En circulation libre, les véhicules ne sont pas (beaucoup) influencés les uns par les autres et peuvent se déplacer librement. Dans un trafic à plusieurs voies, cela signifie que les véhicules peuvent dépasser librement. Notez que, par conséquent, le trafic dans la voie gauche est plus rapide que le trafic dans la voie droite [08].

## 3.2 La phase capacité (synchronisé):

Le flux synchronisé se retrouve dans le trafic multi-voies et se caractérise par le fait que les vitesses des véhicules dans les voies deviennent égales - d'où le nom de l'état «synchronisées». Cette phase est l'un des états de la phase congestionnée, la différence qui caractérise les deux est que, dans le flux synchronisé, les véhicules se déplacent à une vitesse éventuellement élevée, alors que dans les larges embouteillages, ils sont presque à l'arrêt. Plusieurs caractéristiques sont attribuées au cette phase: Les vitesses des véhicules dans toutes les voies sont égales, les vitesses peuvent être élevées, Le débit peut être élevé, voire supérieur à la vitesse maximale de débit libre. Notez que dans ce cas, la «capacité» de la route se trouve dans le flux synchronisé, et enfin, il y a une zone, plutôt qu'une ligne, d'états de trafic dans le diagramme de base de trafic [08].

## 3.3 La phase congestionnée:

Un large embouteillage ressemble le plus à ce que l'on appelle le plus souvent une vague (arrête et va). La vague (arrête et avancez) est lié au mouvement effectué par un véhicule individuel, il s'arrête (brièvement, de l'ordre de 1 à 2 minutes), puis avancez. La vitesse dans l'embouteillage est presque égale à zéro, et dans le diagramme de base du trafic l'état du trafic se trouve donc à un débit presque égal à zéro. Par conséquent, la queue se déplace en amont avec chaque véhicule attaché à l'embouteillage. La tête se déplace également en amont, car les véhicules partent de l'avant, par conséquent, le motif de l'embouteillage se déplace en amont. Ce phénomène est purement basé sur le faible débit à l'intérieur de la congestion, ce qui lui permet de voyager en amont sur de longues distances (des dizaines de kilomètres). Il peut traverser des zones d'écoulement synchronisé et faire des passes rapides [08].

# 4 La modélisation du flux de trafic:

On peut trouver un compte rendu de divers modèles mathématiques basés sur le diagramme fondamental du flux de trafic. Ces modèles peuvent être classés en tant que trois niveaux de flux de trafic: microscopique, macroscopique et mésoscopique. Le niveau microscopique implique le suivi de chaque véhicule pour décrire son comportement et ses interactions avec les autres véhicules en circulation. Le niveau macroscopique décrit de manière agrégée la dynamique des flux de trafic en utilisant des caractéristiques telles que la densité, la vitesse moyenne et le débit. Le niveau mésoscopique (modèle cinétique) décrit le flux de trafic de manière moins agrégée que le niveau macroscopique et en termes probabilistes [09].

## 4.1 Le niveau microscopique:

Le niveau microscopique est la plus ancienne famille de modèle qui intègre la dynamique. Ils sont basés sur l'hypothèse que les conducteurs adaptent leur comportement à celui du véhicule leader [10]. Les flux de circulation routière sont composés de conducteurs associés à des véhicules individuels, chacun ayant ses propres caractéristiques. Ces caractéristiques sont appelées microscopiques lorsqu'un flux de trafic est considéré comme composé d'un tel flux de véhicules. Les aspects dynamiques de ces flux de trafic sont formés par les interactions sous-jacentes entre les conducteurs des véhicules. Ceci est largement déterminé par le comportement de chaque conducteur, ainsi que par les caractéristiques physiques des véhicules [06].

## 4.2 Le niveau macroscopique:

Le niveau de flux de trafic macroscopique décrit le flux de trafic comme s’il s’agissait d’un flux continu et est souvent comparé ou dérivé par analogie avec les modèles de continuum pour les fluides. Les véhicules individuels ne sont pas modélisés, mais des variables agrégées telles que la densité, la vitesse moyenne et le débit sont utilisées [10].

## 4.3 Le niveau mésoscopique:

Le niveau de flux de trafic mésoscopique a été développé pour remplir le fossé entre la famille de modèles microscopiques décrivant le comportement de véhicules individuels et la famille de modèles macroscopiques décrivant le trafic en tant que flux continu. Le niveau mésoscopique décrit le comportement des véhicules en termes globaux. Cependant, des règles comportementales sont définies pour chaque véhicule [10].

# 5 Tableau taxonomique de caractéristiques de chaque famille de modèles de mobilité:

| | |
|---|---|
| **Le niveau microscopique** | Le niveau microscopique est un moyen de modélisation du flux dans lequel les caractéristiques locales du véhicule sont attribuées, telles que l'accélération/décélération de la voiture, le comportement du conducteur, la longueur de la voiture, sa vitesse, etc [11]. Il contient les composants suivants:<br>**Le modèle suiveur (**The car-following model **):** Ce modèle évalue le comportement d'un véhicule spécifique sur la base du comportement de conduite du véhicule qui précède.<br>**Le modèle de changement de voie:** Cela concerne la façon dont les véhicules changent de voie en fonction des véhicules les entourant.<br>**Modèle de choix d'itinéraire:** les véhicules doivent trouver l'itinéraire le plus court à travers un réseau d'infrastructures.<br>**Modules complémentaires:** Étant donné que la position, la vitesse et l'accélération de chaque véhicule sont connues à chaque segment de temps (par exemple, toutes les demi-secondes), il est assez facile de calculer les effets dérivés tels que la pollution, la perte de temps et les coûts économiques [12]. |
| **Le niveau macroscopique** | le niveau macroscopique comprend un nombre inférieur de paramètres par rapport aux modèles microscopiques. De plus, ils ont une forme analytique, ce qui permet leur utilisation pour diverses tâches d'ingénierie du trafic importantes par simulation [13]. Parmi les caractéristiques de ce niveau : il décrit les propriétés les plus importantes des flux de trafic, telles que la formation et la dissipation de files d'attente (queues), et les ondes de choc (shock waves). De plus, la demande de calcul n'augmente pas avec les densités de trafic, c'est-à-dire qu'il ne dépend pas du nombre de véhicules dans le réseau. Il permet de déterminer les temps de trajet moyens, la consommation moyenne de carburant et les émissions en relation avec les opérations de trafic [14].<br>Les modèles de ce niveau utilisées pour la prédiction d'état de flux de trafic si est en état congestionnée, libre ou synchronisée.<br>**Observation** : les modèles de ce niveau sont basés sur les informations collectées dans le niveau microscopique. |
| **Le niveau mésoscopique** | Étant donné que le processus de participation à un flux de trafic est fortement basé sur les aspects comportementaux associés aux facteurs humains, le niveau **microscopique** semblerait important d'inclure ces facteurs humains dans les équations de modélisation. Cependant, cela entraîne une augmentation considérable de la complexité, ce qui n'est pas toujours un artefact souhaité [06]. Aussi, le flux de trafic de niveau microscopique est souvent critiqué pour ses nombreux paramètres, dont les valeurs sont difficiles à estimer en raison de la dynamique du système [10].<br>D'autre part, le niveau **macroscopique** a été dérivé par analogie avec un modèle de flux de rivière [10]. Il a été discuté que la différence entre le flux de fluide et le flux de trafic sont trop importantes pour justifier l'approche de continuum, parce que le véhicule et le conducteur se comportent différemment et changent leur comportement au fil du temps [10]. D'autres ont répondu à cela que les conducteurs ont tous des objectifs similaires (même direction, même vitesse désirée) et préfèrent généralement et souvent de ne pas accélérer ou ralentir. Cela implique que l'hypothèse du continuum est raisonnable pour le flux de trafic, si l'on ne cherche pas trop de détails descriptifs [10]. Cela signifie qu'à ce |

| | |
|---|---|
| | niveau, plus il y a de paramètres, moins il est précis. Pour remédier et remplir ces deux failles existe dans les modèles des autres niveaux, le niveau **mésoscopique** est apparu, les détails dans ce niveau est inférieur à un modèle microscopique et supérieur à un modèle de trafic macroscopique [15]. Les modèles mésoscopiques également spécifient le comportement du trafic par groupes de véhicules/conducteurs, tandis que les interactions de ces groupes étaient décrites avec un niveau de détail peu élevé [15]. |

**PART II :**

# 6 Les simulateurs pour les réseaux VANETs:

L'objectif de la simulation est de reproduire les mêmes résultats que ceux que l'on obtiendrait en conditions réelles. Un simulateur de réseau VANET réaliste se compose des éléments indispensables : un simulateur de réseau Ad Hoc regroupant les bases réseaux d'une simulation VANET et un simulateur de mobilité.

Le simulateur de mobilité génère des fichiers qui décrivent la mobilité des nœuds dans un environnement, et qui sont transmis au simulateur de réseau Ad Hoc. Bien que les deux simulateurs soient indispensables a la simulation des réseaux VANETs, il n'y a aucun lien direct entre eux [16].

## 6.1 Simulateurs de réseaux Ad Hoc :

Il n'y a pas de principe de fonctionnement standard pour les simulateurs de réseau Ad Hoc ou les simulateurs de mobilité. Il existe plusieurs stratégies et architectures qui permettent de simuler un réseau Ad Hoc, chacune étant propre a un simulateur. Chaque simulateur répond aux besoins de la simulation de réseau via des architectures variées, en fonction des objectifs que se sont fixes leurs développeurs. Cependant, ils ont tous le même but: reproduire le plus fidèlement possible le fonctionnement d'une pile protocolaire [16].

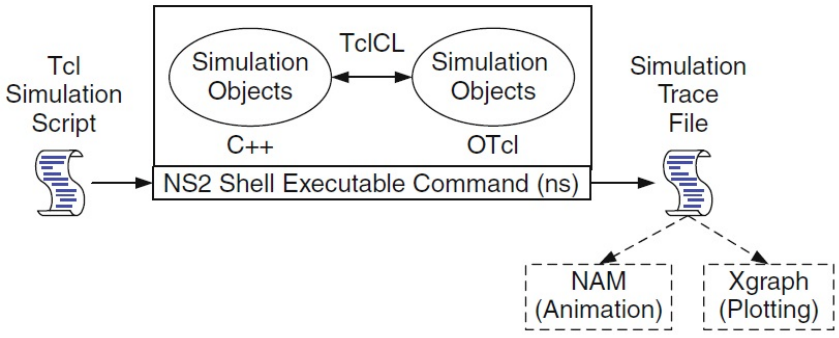
### 6.1.1 Le simulateur NS-2 :

Network Simulator (Version 2), plus connu sous le nom de NS2, est simplement une open source outil de simulation événementiel qui s'est révélé utile pour étudier la nature dynamique des réseaux de communication. NS2 permet de simuler des fonctions et des protocoles réseau câblés et sans fil (algorithmes de routage, TCP, UDP, … etc). En général, NS2 offre aux utilisateurs un moyen de spécifier de tels protocoles réseau et de simuler les comportements correspondants. Grâce à sa flexibilité et sa nature modulaire, NS2 a acquis une popularité constante dans la communauté des chercheurs en réseaux depuis sa création en 1989. Depuis lors, plusieurs révolutions et révisions ont marqué la maturité croissante de l'outil, grâce aux contributions substantielles des acteurs du domaine. Parmi ceux-ci, citons l'Université de Californie et l'Université Cornell, qui ont mis au point le simulateur de réseau REAL, la fondation sur laquelle NS est basée. Depuis 1995, la DARPA (Defence Advanced Research Projects Agency) a soutenu le développement de la SN grâce au projet Virtual InterNetwork Testbed (VINT). la National Science Foundation (NSF) a rejoint le monter dans le développement [17].

**Architecture de base :** La figure 2-5 illustre l'architecture de base de NS2. NS2 fournit aux utilisateurs une commande exécutable ns qui prend l'argument d'entrée, le nom d'un fichier de script de simulation Tcl. Les utilisateurs fournissent le nom d'un script de simulation Tcl (qui configure une simulation) en tant qu'argument d'entrée d'une commande exécutable NS2. Dans la plupart des cas, un fichier de trace de simulation est créé et est utilisé pour tracer un graphique et / ou créer une animation.

NS2 est composé de deux langages clés: C++ et OTcl (Object-Oriented Tool Command Language). Alors que le C++ définit le mécanisme interne 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. Mappées sur un objet C++, les variables des domaines OTcl sont parfois appelées descripteurs. Conceptuellement, un descripteur (par exemple, n en tant que descripteur de nœud) est simplement une chaîne dans le domaine OTcl et ne contient aucune fonctionnalité. Au lieu de cela, la fonctionnalité (par exemple, la réception d'un paquet) est définie dans l'objet C++ mappé. Dans le domaine OTcl, un descripteur agit comme une interface qui interagit avec les utilisateurs et les autres objets OTcl. Il peut définir ses propres procédures et variables pour faciliter l'interaction. Les procédures membres et les variables du domaine OTcl sont appelées procédures d'instance (instprocs) et variables d'instance (instvars). NS2 fournit un grand nombre d'objets C++ intégrés. Il est conseillé d'utilisez ces objets C++ pour configurer une simulation à l'aide d'un script de simulation Tcl. Cependant, les utilisateurs avancés peuvent trouver ces objets insuffisants. Ils doivent développer leurs propres objets C++ et utiliser une interface de configuration OTcl pour assembler ces objets.

**Figure (2-5) : Architecture de base de NS-2.**




Après la simulation, NS2 génère des résultats de simulation sous forme de texte ou d'animation. Pour interpréter ces résultats graphiquement et interactivement, des outils tels que NAM (Network AniMator) et XGraph sont utilisés. Pour analyser un comportement particulier du réseau, les utilisateurs peuvent extraire un sous-ensemble pertinent de données textuelles et le transformer en une présentation plus concevable [17].

### 6.1.2 Le simulateur NS-3 :

Le simulateur NS3 est un simulateur open source de réseau à événements discrets principalement destiné à la recherche et à l'utilisation éducative. Il est sous licence GNU GPLv2, et

disponible pour la recherche et le développement. Il définit un modèle de procédure de travail des réseaux de données par paquets et fournit un moteur de simulation. Plusieurs utilisateurs utilisent NS3 pour modéliser des systèmes non basés sur Internet. NS3 utilise deux langages clés comme son prédécesseur NS2. Le simulateur est entièrement écrit en C++ avec des liaisons python optionnelles. Les scripts de simulation peuvent donc être écrits en C++ ou en Python. Les animateurs ont l'habitude d'afficher visuellement les résultats. NS3 fournit une bibliothèque solide qui permet à l'utilisateur de faire son travail en éditant le NS-3 lui-même. Pour la technologie filaire, le NS-3 fournit un modèle d'appareil d'un simple réseau Ethernet utilisant CSMA / CD comme schéma de protocole avec un délai de décroissance augmentant de manière exponentielle afin de lutter pour le support de transmission partagé. Il fournit également un ensemble de modèles 802.11 qui tentent de fournir une implémentation précise au niveau MAC de la spécification 802.11 et un modèle au niveau PHY de la spécification 802.11a. Il réduit l'encombrement de la mémoire de la simulation et n'alloue aucune mémoire pour zéro octet virtuel. Le modèle de mobilité n'est pas nécessaire, car les périphériques câblés n'ont pas besoin de connaître la position de son nœud. Il prend également en charge l'intégration de logiciels réseau plus ouverts et évite de réécrire les modèles à des fins de simulation [18].

### 6.1.3 Le simulateur OPNET :

OPNET (Optimized Network Engineering Tools) est utilisé avec souplesse pour étudier les réseaux de communication, les périphériques, les protocoles et les applications. Il offre un support graphique relativement puissant aux utilisateurs. L'interface de l'éditeur graphique permet de créer une topologie de réseau et des entités de la couche application à la couche physique. La technique de programmation orientée objet est utilisée pour créer une correspondance entre la conception graphique et la mise en œuvre des systèmes réels. Les résultats de configuration et de simulation de toutes les topologies peuvent être présentés de manière très intuitive et visuelle. Les paramètres peuvent également être ajustés et les expériences peuvent être répétées facilement via l'interface graphique. OPNET est aussi basé sur un système d'événements discrets. La structure hiérarchique est utilisée pour organiser les réseaux. Le simulateur OPNET est très utile lorsque vous travaillez avec des réseaux complexes avec un grand nombre de périphériques et de flux de trafic ou dans des réseaux où un petit changement peut être critique. Avant de mettre en œuvre un changement, il est possible de prédire le comportement et de vérifier les configurations des périphériques. OPNET dispose de différents outils (NetDoctor, ACE et MVI) qui permettent aux administrateurs d'analyser leurs réseaux et les implémentations futures qu'ils souhaitent réaliser. Le langage de programmation principal dans OPNET est C (les versions récentes supportent le développement C++) [18].

OPNET est un produit commercial et sa disponibilité est strictement limitée. Le simulateur est disponible avec le code source complet pour tous les modules de simulation (bien que le code pour le moteur de simulation ne soit pas disponible car ce n'est pas le code source de certains protocoles restreints), des exemples et une documentation exhaustive. Une assistance supplémentaire, disponible via la liste de diffusion OPNET, nécessite une licence de maintenance supplémentaire. En termes simples, «OPNET n'a rien de gratuit» - il existe un certain nombre de packages OPNET supplémentaires qui nécessitent un achat séparé [19].

### 6.1.4 Le simulateur OMNET++ :

L'environnement de simulation d'événements discrets OMNeT++ a été disponible au public depuis 1997. OMNeT++ est un simulateur d'événements discrets basé sur C++ permettant de modéliser des réseaux de communication, des multiprocesseurs et d'autres systèmes distribués ou parallèles. OMNeT++ est une open source et peut être utilisé sous la licence publique académique qui rend le logiciel gratuit pour une utilisation sans but lucratif. Le développement d'OMNeT++ visait à produire un puissant outil de simulation d'événements discrets à code source ouvert pouvant être utilisé par des institutions universitaires, des établissements d'enseignement et des établissements commerciaux axés sur la recherche pour la simulation de réseaux informatiques et de systèmes distribués ou parallèles. OMNeT++ tente de remplir le fossé entre les logiciels de simulation open source, tels que NS-2, et les alternatives commerciales et couteuses, telles que OPNET. OMNeT ++ est disponible sur toutes les plateformes courantes, y compris Linux, Mac OS / X et Windows, à l'aide de la chaîne d'outils GCC ou du compilateur Microsoft Visual C++ [20].

### 6.1.5 Tableau comparative d'OMNeT++ avec les autres simulateurs :

| | |
|---|---|
| **Avec NS-2** | * NS-2 est actuellement le simulateur de réseau le plus utilisé dans les milieux universitaires et de recherche. NS-2 ne suit pas la même séparation claire du noyau de simulation et des modèles que OMNeT++. la distribution NS-2 contient les modèles ainsi que leur infrastructure de support, sous la forme d'une unité indissociable. C'est une différence clé. L'objectif du projet NS-2 est de construire un simulateur de réseau, tandis que OMNeT++ a pour objectif de fournir une plate-forme de simulation sur laquelle différents groupes de recherche peuvent construire leurs propres plateformes de simulation. Cette dernière approche s'appelle l'abondance de modèles de simulation et de plateformes de modélisation basés sur OMNeT++ et a transformé OMNeT++ en une sorte d '«écosystème».<br><br>* NS-2 manque à de nombreux outils et composants d'infrastructure fournis par OMNeT++: prise en charge des modèles hiérarchiques, éditeur graphique, environnement d'exécution basé sur une interface graphique, séparation des modèles des expériences, outils d'analyse graphique, fonctionnalités de bibliothèque de simulation intégrées de manière transparente, etc. C'est parce que le projet NS-2 se concentre sur le développement des modèles de simulation et beaucoup moins sur l'infrastructure de simulation.<br><br>* NS-2 est un simulateur à deux langages, les modèles de simulation sont des scripts Tcl, tandis que le noyau de simulation et divers composants (protocoles, canaux, agents, etc.) sont implémentés en C et sont accessibles depuis le langage Tcl. La topologie du réseau est exprimée dans le script Tcl, qui traite généralement de plusieurs autres éléments, allant de la définition de paramètres à l'ajout de comportements d'application et à l'enregistrement de statistiques. Cette architecture rend pratiquement impossible la création d'éditeurs graphiques pour les modèles NS2 [20]. |
| **Avec NS-3** | * NS-3 est un effort continu pour consolider tous les correctifs et modèles récemment développés dans une nouvelle version de NS. Bien que les travaux incluent également la refactorisation du noyau de simulation, les concepts restent essentiellement inchangés. Les objectifs du projet NS-3 incluent certaines fonctionnalités (par exemple, la simulation en parallèle, l'utilisation d'implémentations de protocoles réels en tant que modèles de simulation) qui se sont déjà révélées utiles avec OMNeT++ [20]. |

| | |
|---|---|
| **Avec OPNET** | * OPNET Modeler est un produit commercial librement disponible dans le monde entier pour les universités éligibles. OPNET possède probablement le plus grand choix de modèles de protocoles prêts à l'emploi (notamment IPv6, MIPv6, WiMAX, QoS, Ethernet, MPLS, OSPFv3 et bien d'autres).<br>* OPNET et OMNeT++ fournissent des bibliothèques de simulation riches en fonctionnalités comparables. La bibliothèque de simulation OPNET est basée sur C, alors que celle d'OMNeT++ est une bibliothèque de classes C ++. L'architecture d'OPNET est similaire à celle d'OMNeT++ dans la mesure où elle autorise des modèles hiérarchiques avec une imbrication arbitrairement profonde (comme OMNeT++), mais avec certaines restrictions (le niveau du "nœud" ne peut pas être hiérarchique). Une différence significative par rapport à OMNeT++ réside dans le fait que les modèles OPNET ont toujours une topologie fixe, tandis que NED d'OMNeT++ et son éditeur graphique autorisent les topologies paramétriques. Dans OPNET, le meilleur moyen de définir la topologie du réseau consiste à utiliser l'éditeur graphique. L'éditeur stocke les modèles dans un format de fichier propriétaire, ce qui signifie en pratique que les modèles OPNET sont généralement difficiles à générer par programme (il est nécessaire d'écrire un programme C utilisant une API OPNET, tandis que les modèles OMNeT++ sont de simples fichiers texte pouvant être générés, par exemple: avec Perl).<br>* OPNET et OMNeT ++ fournissent tous les deux un débogueur graphique et une forme d'animation automatique essentielle au développement facile des modèles [20]. |

## 6.2 Simulateurs de mobilité:

On trouve deux types de simulateurs de mobilité : les simulateurs indépendants et les simulateurs intègres. Les simulateurs indépendants génèrent uniquement la mobilité des réseaux VANETs. Il faut en général leur associer des outils pour pouvoir exploiter cette mobilité dans un simulateur de réseau Ad Hoc. Les simulateurs intègres regroupent a la fois un simulateur de mobilité et un simulateur de réseau Ad Hoc, donc il n'y a pas de problème de compatibilité [16]:

### 6.2.1 Les simulateurs indépendants :

**6.2.1.1 Le simulateur PARAMICS :** PARAMICS est un logiciel de simulation microscopique du trafic urbain et autoroutier utilisé pour modéliser le mouvement et le comportement de véhicules individuels sur des réseaux routiers. Au début des années 1990, Quadstone et SIAS Ltd ont développé une version de PARAMICS pour superordinateur Cray au Centre de calcul parallèle de l'Université d'Édimbourg. Le nom PARAMICS est un acronyme dérivé de PARAllel computer MICropic Simulation. Depuis 1998, Quadstone et SIAS développent et commercialisent des versions indépendantes de PARAMICS [22].

Le progiciel PARAMICS comprend cinq modules: Modeller, Processor, Analyzer, Programmer et Monitor. Un package d'ingénierie de transport typique comprend trois de ces modules: modélisateur, processeur et analyseur. Le module Modeller offre la possibilité de créer, simuler et visualiser le réseau routier à l'aide d'une interface utilisateur graphique. Le module de modélisation permet également la création d'une sortie statistique sous la forme de fichiers texte bruts. Le module Processeur fournit les mêmes fonctionnalités que Modeller, à ceci près que les simulations se produisent sans visualisation, ce qui augmente la vitesse de la simulation. Le module

Analyzer utilise les données de sortie de chaque simulation exécutée par Modeller pour produire des résultats sous forme de tableau et graphique pour une analyse hors ligne [22].

La précision obtenue dans un modèle de simulation de trafic microscopique dépend de la précision du codage des caractéristiques du réseau. Étant donné que les véhicules réagissent individuellement aux contraintes géométriques telles que la largeur des voies et les faibles rayons de braquage, il est essentiel que le réseau modélisé représente les conditions réelles. Le codage de réseau dans PARAMICS est basé sur des «nœuds» et des «liens», chaque lien étant codé comme un connecteur entre deux nœuds. La création du réseau s'effectue dans une interface utilisateur graphique qui permet à l'utilisateur de créer le réseau à l'aide de commandes de dessin sur ordinateur, soit en dessinant à main levée des liens, soit en dessinant sur un fichier de géométrie de route type, tel qu'une photographie aérienne ou un dessin CAD [22].

**6.2.1.2 Le simulateur CORSIM:** CORSIM (Corridor Similation) est un simulateur commercial basé sur un modèle de simulation microscopique pour simuler le modèle de mobilité dans les autoroutes et les routes urbaines. Le développement initial de la logique CORSIM a commencé au début des années 1970. Sous le parrainage de la FHWA (Federal Highway Administration), le programme CORSIM a subi de nombreuses améliorations et mises à niveau majeures, à la fois en logique de simulation et en génie logiciel. Son développement logiciel récent est l'introduction de TSIS (Traffic Software Integrated System), qui fournit une interface et un environnement intégrés et conviviaux pour l'exécution du modèle de simulation de trafic CORSIM. Les chercheurs et les pratiquants utilisent CORSIM en raison de ses bases solides en matière de modélisation et d'analyse de l'ingénierie du trafic. Il offre de fonctionnalités riches de modélisation permettant aux utilisateurs de simuler un flux de trafic étendu pour des applications pratiques ou de recherche. Il a été unanimement accepté que CORSIM possède des atouts spécifiques dans les domaines suivants:

Sa capacité à modéliser des conditions géométriques complexes. CORSIM peut gérer pratiquement toutes les conditions géométriques complexes qui se produisent pratiquement sur le terrain. Les conditions de géométrie acceptées par CORSIM incluent les rues en surface avec différentes combinaisons de voies de circulation et de virages, de segments d'autoroutes à plusieurs voies, de différents types de bretelles d'accès et de sortie.

Sa capacité à simuler différentes conditions de circulation. CORSIM a été calibré pour simuler avec précision un large éventail de conditions de circulation, allant d'une demande modérée à des conditions très encombrées. CORSIM peut également simuler efficacement le flux de trafic lors d'un incident, depuis l'accumulation de la file d'attente jusqu'à la récupération en passant à l'état normale.

Sa capacité à modéliser des conditions de trafic et de contrôle variables dans le temps. CORSIM utilise des types d'enregistrement (RTs) pour organiser les entrées de données pour les géométries, les volumes et les modèles, les dispositifs de surveillance et de détection, le contrôle de trafic, les critères d'ingénierie, le contrôle d'exécution et les exigences de sortie. CORSIM simule les conditions de trafic d'un réseau sur une période donnée. Les entrées prennent en charge des spécifications qui diffèrent non seulement d'un point du réseau à un autre, mais qui peuvent également changer avec le temps. La partie de l'analyse de simulation qui varie dans le temps est exprimée sous forme d'une séquence de «périodes» spécifiées par l'utilisateur. Au cours de chaque

période, CORSIM permet différentes demandes de trafic, ainsi que différentes opérations et contrôles de trafic [23].

**6.2.1.3 Le simulateur CityMob :** Est un simulateur développé à l'Université Polytechnique de Valence, il est conçu pour être utilisé avec le simulateur NS-2 dans la génération de modèle de mobilité pour les réseaux VANET. CityMob 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 les 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. CityMob propose trois modèles de mobilité différents :

**Modèle simple (MS) :** Modélise les modèles de mobilité verticale et horizontale sans changement de direction. Les feux ne sont pas non plus pris en charge.

**Le Modèle Manhattan (MM) :** il modélise la ville sous forme de grille de style Manhattan, avec une taille de bloc uniforme dans toute la zone de simulation. Toutes les rues sont à double sens, avec une voie dans chaque direction. Les mouvements de voiture sont limités par ces voies. La direction de chaque nœud à chaque instant sera aléatoire et ne pourra pas être répétée deux fois de suite. De plus, ce modèle simule des feux à des positions aléatoires (pas seulement en croisement) et avec des retards différents. Lorsqu'un véhicule rencontre un feu, il reste arrêté jusqu'à ce que le feu devienne vert.

**Le modèle de Centre-ville (Downtown) réel (MD) :** Ce modèle ajoute la densité de trafic au modèle de Manhattan. Dans une vraie ville, le trafic n'est pas uniformément réparti, il y a des zones avec une densité de véhicule plus élevée. Ces zones sont généralement situées en centre-ville et les véhicules doivent circuler plus lentement qu'en périphérie [24].

**6.2.1.4 Le simulateur VanetMobisim :** VanetMobiSim est une extension de CanuMobiSim, un simulateur de mobilité d'utilisateur générique. CanuMobiSim fournit une architecture de mobilité efficace et facilement extensible, mais en raison de sa nature d'usage général, souffre d'un niveau de détail réduit dans des scénarios spécifiques. VanetMobiSim vise donc à étendre le support de mobilité de CanuMobiSim en matière de mobilité des véhicules à un degré de réalisme supérieur. Par la suite, pour des raisons d'espace, nous ne listons que les ajouts originaux introduits par VanetMobiSim, mais il convient de noter que l'outil complet intègre toutes les fonctionnalités de CanuMobiSim, offrant un très large éventail de possibilités pour simuler la mobilité des véhicules. VanetMobiSim possède des caractéristiques macro et micro-mobilité :

Les caractéristiques macro-mobilités : la macro-mobilité prend en compte la topologie de la route, la structure de la route (unidirectionnelle ou bidirectionnelle, une ou plusieurs voies), les caractéristiques de la route (limitations de vitesse, restrictions de classes de véhicules) et la présence de panneaux de signalisation (panneaux d'arrêt, feux de circulation, etc). En outre, le concept de macro-mobilité inclut également les effets de la présence de points d'intérêts, qui influencent les schémas de déplacement des véhicules sur la topologie routière.

Les caractéristiques micro-mobilités : Le concept de micro-mobilité véhiculaire englobe tous les aspects liés à la modélisation de la vitesse et de l'accélération d'une voiture individuelle. VanetMobiSim ajoute deux modèles de mobilité microscopiques originaux afin d'inclure la gestion des intersections régies par des panneaux de signalisation et des routes à plusieurs voies [25].

**6.2.1.5 Le simulateur SUMO :** Le Centre aérospatial allemand (DLR) a lancé le développement du logiciel de simulation de trafic open source SUMO (Simulation of Urban MObility) en 2001. SUMO est une plateforme de simulation de flux de trafic microscopique, inter-modale et mult-imodale, continue dans l'espace et discrète dans le temps. SUMO n'est pas seulement une simulation de trafic, mais plutôt une suite d'applications permettant de préparer et de simuler un scénario de trafic. Les applications incluses dans la suite sont présentées ci-dessous, en les divisant par leur objectif: génération de réseau, génération de demande et simulation:

**Génération de réseau routier :** Les réseaux de routes SUMO représentent des réseaux du monde réel sous forme de graphiques, où les nœuds sont des intersections et les routes sont représentées par des arêtes. Les intersections consistent en une règle de position, de forme et de priorité, qui peut être remplacée par un feu de signalisation. Les bords sont des connexions unidirectionnelles entre deux nœuds et contiennent un nombre fixe de voies. Une voie contient la géométrie, les informations sur les classes de véhicules autorisées et la vitesse maximale autorisée. Par conséquent, les modifications du nombre de voies le long d'une route sont représentées à l'aide de plusieurs arêtes [26].

**Véhicules et routes :** SUMO est une simulation de trafic purement microscopique. Chaque véhicule est donné explicitement, défini au moins par un identifiant unique, l'heure de départ et la route du véhicule à travers le réseau. Par «route», nous entendons la liste complète des arêtes connectées entre le départ et la destination du véhicule. Si nécessaire, chaque véhicule peut être décrit plus en détail à l'aide des propriétés de départ et d'arrivée, telles que la voie à utiliser, la vitesse ou la position exacte sur un bord. Chaque type de véhicule peut recevoir un type, décrivant les propriétés physiques du véhicule et les variables du modèle de mouvement utilisé. Chaque véhicule peut également être affecté à l'une des classes de polluants ou d'émissions sonores disponibles. Des variables supplémentaires permettent de définir l'apparence du véhicule dans l'interface utilisateur graphique de la simulation.

Un scénario de simulation d'une grande ville couvre facilement un million de véhicules et leurs itinéraires. Même pour de petites zones, il est difficile de définir manuellement la demande de trafic. La suite SUMO inclut certaines applications, qui utilisent différentes sources d'information pour définir une demande [26].

**Simulation :** L'application «sumo» effectue une simulation discrète dans le temps. La longueur de pas par défaut est 1s, mais peut être choisie pour être inférieure, jusqu'à 1ms. En interne, le temps est représenté en microsecondes et stocké sous forme de valeurs entières. La durée maximale d'un scénario est donc liée à 49 jours. Le modèle de simulation est continu dans l'espace et à l'intérieur, la position de chaque véhicule est décrite par la voie sur laquelle il se trouve et par la distance qui le sépare du début de cette voie. Lorsque vous vous déplacez sur le réseau, la vitesse de chaque véhicule est calculée à l'aide d'un modèle appelé système de suivi de voiture. Les modèles suiveurs

de voiture calculent généralement la vitesse d'un véhicule ayant fait l'objet d'une enquête en examinant la vitesse de ce véhicule, sa distance au véhicule en tête (Leader) et sa vitesse [26].

**Interaction en ligne :** En 2006, la simulation a été étendue à la possibilité d'interagir avec une application externe via une connexion socket. Cette API, appelée «TraCI» pour «Traffic Control Interface», a été mise en œuvre par Axel Wegener et ses collègues de l'Université de Lübeck et a été mise à disposition dans le cadre de la publication officielle de SUMO [26].

### 6.2.2 Les simulateurs intégrés :

**6.2.2.1 Le simulateur TraNs:** TraNS (Traffic and Network Simulation Environment) est le premier simulateur VANET. C'est un outil de simulation qui intègre à la fois un générateur de mobilité et un simulateur de réseau pour réaliser une simulation réaliste de VANET. Il fournit un outil pour construire des simulations VANET réalistes. TraNS fournit également une feedback entre le modèle de mobilité et le comportement du véhicule. Une version séparée appelée Translite a été conçue pour la génération de modèles de mobilité uniquement sans utiliser le simulateur intégré NS-2 pour la simulation de réseau. TraNS peut fonctionner dans deux modes: mode centré sur l'application et mode centré sur le réseau. En mode centré sur le réseau, aucun retour n'est fourni de NS-2 à SUMO. Ainsi, le lien entre NS-2 et SUMO se fait via un analyseur. L'analyseur analyse le fichier de trace de mobilité généré par le simulateur SUMO, puis le convertit dans un format adapté à NS-2. Il analyse le fichier de trace de mobilité généré par le simulateur SUMO, puis le convertit dans un format approprié pour le simulateur NS-2. Dans le cas du mode centré sur l'application, le retour d'informations entre NS-2 et SUMO est fourni par l'intermédiaire d'une interface appelée TraCI. Dans ce mode, SUMO et NS-2 doivent être exécutés simultanément. TraNS a pour principal mérite de prendre en charge la norme 802.11p, hautement évolutive et fonctionne sous les plateformes Linux et Windows [27].

**6.2.2.2 Le simulateur NCTUns:** Le simulateur (National Chiao Tung University Network Simulator) (NCTUns) est un simulateur VANET qui combine le simulateur de trafic et de réseau dans un module unique construit à l'aide du langage de programmation C++ et prenant en charge un niveau élevé d'interface graphique. Il s'agit d'un simulateur de réseau extrêmement extensible et robuste, sans souci de la complexité du code.

Il prend en charge les simulations parallèles pour le réseau fixe sur des machines multi-core. Le principal avantage de NCTUns est qu'il peut simuler les technologies IEEE 802.11a, IEEE 802.11b, IEEE 802.11e, IEEE 802.16d, IEEE 802.11g et IEEE 802.11 et qu'il prend en charge un grand nombre de nœuds. NCTUns inclut la communication bidirectionnelle et unidirectionnelle. Comparé à TraNS, NCTUns combine le simulateur de trafic et de réseau dans un seul module et renvoie les informations en retour au support VANET [27].

# 7 La plateforme Veins et le couplage entre OMNeT++ et SUMO:

L'évaluation des performances des protocoles développés est généralement réalisée à l'aide de techniques de simulation, car des essais sur le terrain réalistes sont encore impossibles ou limités

à quelques centaines de voitures, comme en témoignent plusieurs projets de recherche récents en Europe, aux États-Unis et en Asie. La simulation des protocoles VANET, tels que la communication sans fil, le routage multi-sauts et la diffusion assistée par l'application sont généralement effectués dans des environnements de simulation de réseau tels que OMNeT++. Le principal avantage est la disponibilité de modèles de protocoles de communication précis et éprouvés.

Néanmoins, ces simulateurs ne sont pas suffisamment équipés pour simuler de manière adéquate la mobilité des véhicules. La science de transports et du trafic classe les modèles de trafic en modèles macroscopiques, mésoscopiques et microscopiques, en fonction de la granularité avec laquelle les flux de trafic sont examinés. Les simulations de scénarios VANET concernent la modélisation précise des transmissions à une seule onde radioélectrique entre les nœuds et par conséquent, nécessite des positions exactes des nœuds simulés. Seules les simulations microscopiques, qui modélisent le comportement de véhicules individuels et leurs interactions, peuvent être considérées comme un modèle de mobilité adéquat pour les nœuds VANET simulés [28]. C'est pourquoi nous avons fait appel le SUMO dont on le sait qui est purement microscopique.

Cependant, il est essentiel d'avoir une interaction forte entre le simulateur de mobilité et le simulateur de réseau avec une latence très faible et une précision élevée pour mesurer la performance du comportement du réseau dans tous les scénarios de véhicules. Les simulateurs à couplage bidirectionnel tel que SUMO ont donc été développés dans lesquels le simulateur de mobilité et le simulateur de réseau sont couplés ensemble en tant que deux processus interdépendants s'exécutant simultanément. Les caractéristiques des véhicules comme la vitesse, la direction et l'accélération, et les caractéristiques de la route comme le nombre de voies et les intersections ont été simulées dans le simulateur de trafic et l'impact de la structure de mobilité sur le réseau est évalué par le simulateur de réseau. Pour ces expériences, le Plateforme Veins a été utilisé. Veins est disponible publiquement et fournit un couplage bidirectionnel de la simulation du trafic routier et réseau, et basé sur les plateformes SUMO et OMNeT++ [29].

# Conclusion:

Dans la première partie de ce chapitre, nous avons expliqué les caractéristiques du flux de trafic, le diagramme fondamental du trafic et sont trois phases et on le finie avec les trois niveaux pour modéliser un flux de trafic. Dans la deuxième partie nous avons présenté les simulateurs de réseaux, les simulateurs de mobilités et comment évaluer l'impact de la structure de mobilité sur le réseau par le couplage entre SUMO et OMNeT++. Dans le prochain chapitre nous allons présenter la partie conceptuelle de notre travail proposé.

# Chapitre 03 :

# Implémentation

# Introduction :

Dans le chapitre précédent, nous avons présenté un cadre théorique traite le problème des embouteillages dans les zones urbaines. Nous concéderons dans ce chapitre un nouveau modèle de mouvement appelé « covoiturage » comme une, parmi plusieurs, solutions les plus efficaces en cas de congestion du trafic. En effet, il consiste à augmenter le taux d'occupation des voitures en réduisant efficacement le nombre de sièges vides dans ces véhicules. Les conducteurs partagent leur véhicule avec une ou plusieurs passagers ayant des itinéraires de déplacement commun. En réduisant le nombre de places vides dans ces véhicules, le nombre d'occupants augmente considérablement. Par conséquent, il faudrait moins de véhicules pour transporter le même nombre de passager vers leurs destinations respectives, ce qui réduirait sensiblement le nombre de voitures sur la route. Les autres avantages du covoiturage comprennent la réduction des coûts de déplacement, de la consommation d'énergie et des émissions des véhicules.

Avant la réalisation de toute simulation, celle-ci doit être passé par une phase de conception, afin de tracer les besoins et de mieux comprendre son fonctionnement. Ce chapitre est dédié à la conception et l'implémentation d'un nouveau modèle de mouvement sous Omnet++, dans lequel nous allons présenter une description des nouveaux modules en vue d'améliorer la congestion routière. Ces modules sont conçus pour développer des scénarios Cloud Véhiculaires et simuler le problème de covoiturage en temps réel. Nous proposons également une approche génétique basé sur l'algorithme de la colonie de fourmis en vue d'optimiser les solutions de correspondance optimales tout en réduisant le temps de calcul nécessaire.

# 1 Les plateformes de développement

## 1.1 Le simulateur OMNET++

Le simulateur OMNeT++ représente une approche basée sur un Plateforme. Au lieu de fournir directement des composants de simulation pour des réseaux informatiques ou d'autres domaines, il fournit des mécanismes et des outils de base pour écrire de telles simulations. Des domaines d'application spécifiques sont pris en charge par divers modèles et plateformes de simulation, tels que la plateforme de mobilité ou le plateforme INET. Ces modèles sont développés de manière totalement indépendante d'OMNeT++ et suivent leurs propres cycles de publication [20].

**Structure du modèle d'OMNeT++**

Un modèle de simulation sous OMNeT++ est constitué de plusieurs modules qui communiquent avec les messages. Les modules actifs sont appelés modules « *simples* », ils sont écrits en C++, à l'aide de la bibliothèque de classes de simulation. Les modules simples peuvent être regroupés en modules « *composés* », etc. le nombre de niveaux hiérarchiques n'est pas limité. Les messages peuvent être envoyés via des connexions s'étendant entre des modules ou directement vers leurs modules de destination [20].

Les modules communiquent avec des messages pouvant contenir des données arbitraires. Les modules simples envoient généralement des messages via des portes, mais il est également possible de les envoyer directement à leurs modules de destination. Les portes sont les interfaces d'entrée et de sortie des modules : les messages sont envoyés par les portes de sortie et arrivent par les portes d'entrée. Une porte d'entrée et une porte de sortie peuvent être liées à une connexion. Les connexions sont créées au sein d'un seul niveau de la hiérarchie des modules : au sein d'un module composé, les portes correspondantes de deux sous-modules ou une porte d'un sous-module et une porte du module composé peuvent être connectée [20].

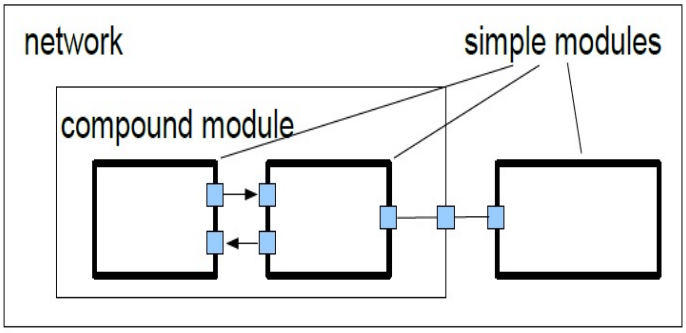



**Figure (3-1) : Structure du modèle d'OMNeT++.**

Dans la construction d'un nouveau modèle de simulation, il y aura à chaque fois des informations chargées dynamiquement telles que la topologie du réseau à partir des fichiers « *.ned* », et les configurations sont disponibles dans les fichiers « *.ini* ». Lors de la simulation, différents fichiers trace seront remplies. On a aussi le « *Plove* » qui est un outil pouvant visualiser

les données enregistrées. Les deux fichiers « *omnet.vec* » et « *omnet.sca* » seront utiles lors du traçage de la courbe et du calcul des statistiques [21].

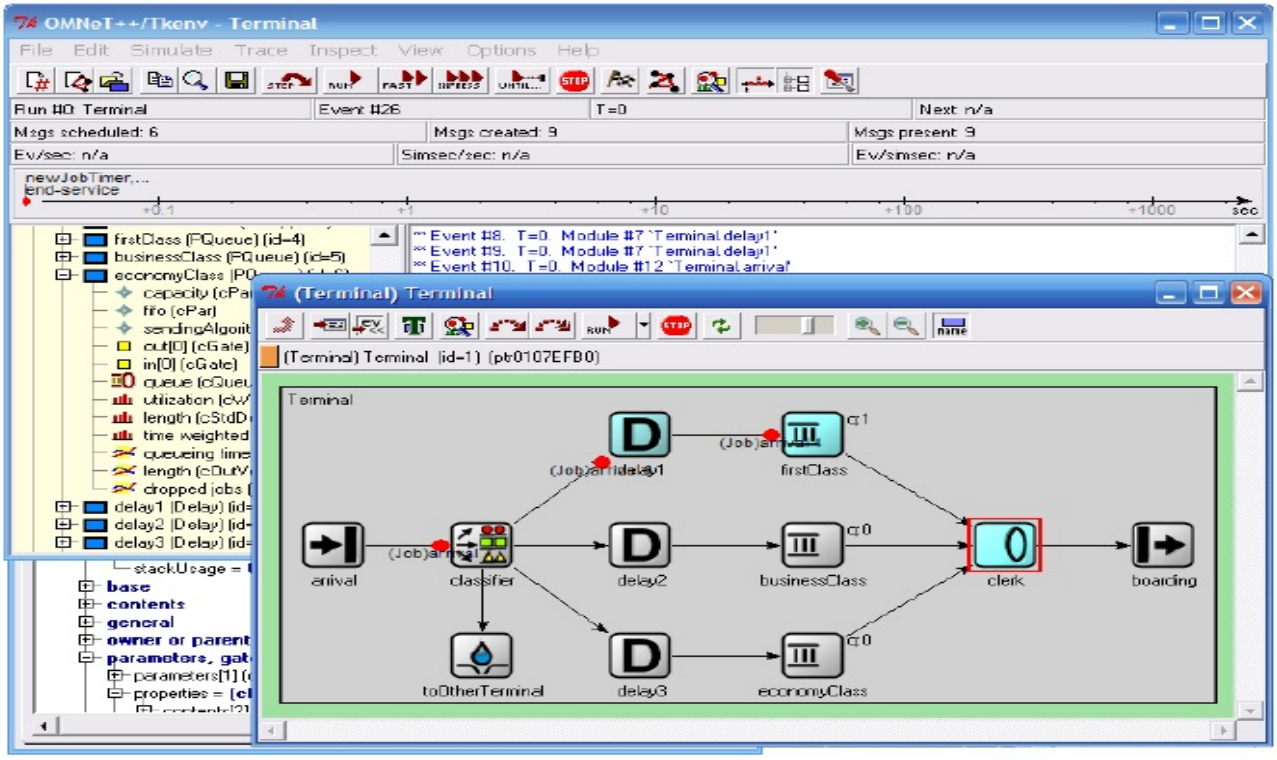



**Figure (3-2) : L'interface graphique d'OMNeT++.**

## 1.2 La plateforme INET :

*INET* est une bibliothèque de modèles à code source ouvert pour l'environnement de simulation *OMNeT ++*. Il fournit des protocoles, des agents et d'autres modèles aux chercheurs et aux étudiants travaillant avec les réseaux de communication. *INET* est particulièrement utile lors de la conception et de la validation de nouveaux protocoles, ou lors de l'exploration de nouveaux ou exotiques scénarios. *INET* contient des modèles pour la pile Internet (*TCP*, *UDP*, *IPv4*, *IPv6*, *OSPF*, *BGP*, etc.), les protocoles de couche de liaison filaire et sans fil (*Ethernet*, *PPP*, *IEEE802.11*, etc.), la prise en charge de la mobilité, les protocoles *MANET*, *DiffServ*, *MPLS* avec signalisation *LDP* et *RSVP-TE*, plusieurs modèles d'application et d'autres nombreux protocoles et composants.

Plusieurs autres structures de simulation utilisent *INET* comme base et l'étendent dans des directions spécifiques, telles que les réseaux véhiculaires, les réseaux *peer-to-peer* ou *LTE*.

*INET* est également conçu pour l'expérimentation et repose sur le concept de modules qui communiquent par la transmission de messages. Les agents et les protocoles réseau sont représentés par des composants, qui peuvent être librement combinés pour former des hôtes, des routeurs, des

commutateurs et d'autres périphériques réseau. Les nouveaux composants peuvent être programmés par l'utilisateur, et les composants existants ont été écrits de manière à être faciles à comprendre et à modifier.

Il bénéficie aussi de l'infrastructure fournie par *OMNeT*. Outre l'utilisation des services fournis par le noyau et la bibliothèque de simulation *OMNeT* (modèle de composant, paramétrage, enregistrement des résultats, etc.), cela signifie également que les modèles peuvent être développés, assemblés, paramétrés, exécutés et leurs résultats évalués dans le confort de l'utilisateur *d'IDE d'OMNeT* ou à partir de la ligne de commande [30].

## 1.3 La plateforme *iCanCloud*:

Les architectures de base d'un système *Cloud* peuvent être très différentes selon les propriétaires du système *Cloud*. Ces différences entravent l'analyse des performances des applications exécutées dans le *Cloud*. En outre, le système de paiement par répartition « *pay as you go* » inhérent pour les entreprises qui est adopté par les fournisseurs de *Cloud*, empêche également d'obtenir le juste équilibre entre les coûts et les performances. Une fois qu'une application est exécutée, les utilisateurs ne savent pas combien de ressources sont nécessaires à leurs applications. Les expériences de planification ont le même problème. Afin de connaître le comportement des stratégies de courtage en gestion et des gestionnaires de cloud, un nombre élevé d'exécutions doit être lancé dans un véritable environnement de Cloud Computing. Ainsi, ces expériences ajoutent des coûts supplémentaires.

Afin de résoudre ces problèmes dans un environnement réel, nous proposons l'utilisation de techniques de simulation. La plate-forme de simulation iCanCloud a été conçue pour modéliser le comportement des environnements de *Cloud Computing* et de leur architecture sous-jacente. *iCanCloud* fournit un ensemble complet de modules : nœuds, réseaux, machines virtuelles, utilisateurs, applications, stratégies de planification de l'environnement cloud et de l'hyperviseur afin de fournir aux utilisateurs un niveau de détail élevé sans perte de précision.

La plateforme de simulation *iCanCloud* est orientée vers la simulation d'un large éventail de systèmes d'informatique en nuage et de leurs architectures sous-jacentes. Ce projet a été lancé en 2010 et est actuellement disponible sous forme de logiciel open source. *iCanCloud* a été conçu pour obtenir un bon compromis entre la flexibilité, la précision, les performances et l'évolutivité, ce qui en fait une plateforme de simulation puissante pour la conception, le test et l'analyse d'architectures existantes et non existantes. En fait, des systèmes informatiques complets de haute performance peuvent être modélisés à l'aide de cette plateforme de simulation. La meilleure caractéristique *d'iCanCloud* est sa capacité à modéliser et à simuler des environnements étendus (des milliers de nœuds) avec un niveau de détail personnalisable.

*iCanCloud* a été développé sur la plateforme *OMNeT++*. Les systèmes de réseau reposent sur la simulation de la plateforme *INET*. L'ensemble de modules fournis par le noyau du simulateur *iCanCloud* simule le comportement de composants spécifiques. Ces modules sont regroupés par fonctionnalité [31].

# 2 Scénario d’étude :

## 2.1 Le problème de covoiturage :

De nos jours, parallèlement à l'augmentation de population et la dispersion des habitants, les services de transport public sont souvent incapables de desservir efficacement les zones où des systèmes de transport rentables ne peuvent être mis en place. En conséquence, de plus en plus de personnes utilisent des véhicules privés pour leurs déplacements quotidiens. Cependant, la forte utilisation de véhicules privés, combinée à une mobilité humaine accrue, alourdit l'environnement et soulève les problèmes de transport tels que la congestion, les problèmes de stationnement et la faible vitesse de transfert. Afin d'atténuer ces problèmes, différents services de mobilité innovants sont en train d'émerger. Le covoiturage est un service de mobilité proposé et organisé par de grandes organisations, telles que les grandes entreprises, les administrations publiques et les universités. Ces organisations encouragent leurs employés ou leurs étudiants à ramasser ou à reprendre des collègues ou des camarades de classe en conduisant vers ou depuis un site commun. Le service tente de réduire le nombre de véhicules privés circulant sur la route en améliorant l'occupation moyenne des voitures. En fait, le covoiturage existe depuis plus de 60 ans. Aux États-Unis, il est devenu une tactique de rationnement pendant la Seconde Guerre mondiale. Il était populaire dans les années 1970 en raison de la crise du pétrole de 1973 et de la crise de l'énergie de 1979. À cette époque, les premiers programmes de covoiturage destinés aux employés étaient organisés chez Chrysler et 3M. Au début du XXe siècle, la popularité d'Internet et des téléphones mobiles a grandement contribué à l'expansion du covoiturage en permettant aux gens de trouver et de contacter plus facilement les membres du covoiturage. Avec de tels antécédents, le service de covoiturage connaît actuellement la période la plus prospère.

La raison pour laquelle les gens adhèrent au système de covoiturage est que le covoiturage réduit les frais de déplacement en partageant ces frais tels que le carburant, les péages et la location de voiture entre les voyageurs. C'est également un moyen de transport plus respectueux de l'environnement et durable, car le partage d'un trajet réduit les émissions de carbone, les embouteillages et les besoins en places de stationnement. Le covoiturage peut également réduire le stress de la conduite, chaque conducteur n'ayant à conduire que pendant un ou deux jours par semaine.

Après plusieurs années de développement rapide, le covoiturage était déjà considéré comme un important service de transport alternatif dans le monde entier. Afin de réduire la circulation et d'encourager le covoiturage, certains pays ont mis en place des voies réservées aux véhicules à occupation multiple (HOV, high occupancy vehicle) réservées aux véhicules à deux passagers ou plus. Dans certains pays, il est également courant de trouver des places de stationnement réservées en particulier aux covoitureurs. De nombreuses entreprises et autorités locales ont mis en place des systèmes de covoiturage, souvent dans le cadre de programmes de transport plus vastes.

Le développement réussi du covoiturage a tendance à être associé principalement aux zones non urbaines telles que les banlieues et plus récemment les universités et autres campus.

Actuellement, la plupart des programmes de covoiturage sont exploités quotidiennement. Un certain nombre d'utilisateurs déclarent être prêts à prendre en charge ou à ramener d'autres utilisateurs à une destination commune un jour donné. Par conséquent, ces utilisateurs sont considérés comme des serveurs et les autres utilisateurs capturés ou ramenés sont considérés comme des clients. Ensuite, le problème consiste à attribuer des clients aux serveurs et à identifier les itinéraires devant être gérés par les serveurs [32].

## 2.2 Description de scénario :

Le scénario selon la figure est que le propriétaire de la voiture doit exploiter tous leurs sièges, et pour que le problème de mobilité routier soit résolu, il ne devrait pas y avoir d'espace vide dans la voiture lorsqu'elle atteint la destination. Dans ce scénario, nous avons donc l'emplacement source et l'emplacement destinataire, les voyageurs dispersant dans la ville, la voiture qui a pour l'objectif de collecter quatre (04) passagers ce qui correspond au nombre de sièges vides, Sachant que la voiture et les passagers voyagent de différents endroits, mais ils se dirigent tous vers la même destination. Comme nous avons déjà indiqué que les objectifs de covoiturage sont de réduit les émissions de carbone, les embouteillages et les besoins en places de stationnement, la voiture doit emprunter le chemin le plus court pour atteindre ces objectifs.


**Figure (3-3) : Scenario d'étude.**

Nous devons encore parler des critères sur lesquels le scénario sera utilisé pour collecter les quatre personnes afin d'atteindre l'emplacement destinataire en prenant l'itinéraire le plus court, qui sera le suivant : le chemin qui contient le moine flux de trafic, la position et le moine temps nécessaire pour atteindre la destination. Alors ce scénario sera multi-objectif.

## 2.3 Travail proposé :

### 2.3.1 Optimisation par colonies de fourmis (ACO, Ant Colony Optimization):

Les fourmis dans la nature gèrent des prouesses incroyables telles que la construction de nids et la recherche de nourriture. L'intelligence en essaim d'agent environnemental naturel appelée optimisation des colonies de fourmis (ACO) a été développée par Dorigo comme une sorte de méthode d'optimisation méta-heuristique. L'ACO est basé sur le système de communication utilisé par les fourmis lorsqu'elles recherchent des sources de nourriture. Les fourmis cherchent de la nourriture à partir de leur nid et empruntent différents chemins pour atteindre la nourriture. Dans cette situation, la seule information nécessaire est le chemin emprunté pour se nourrir. Néanmoins, la topologie complète de l'environnement est celle qui n'est pas claire pour une seule fourmi qui dépose la phéromone sur son trajet. Les fourmis qui suivent utilisent une communication indirecte consistant à retracer la phéromone laissée par les fourmis précédentes. Plus les fourmis passent par un chemin particulier, plus la concentration de phéromone augmente le long de ce chemin. La phéromone agit comme un stimulus important puisque d'autres fourmis peuvent sentir la phéromone déposée l'une par l'autre et prennent généralement le chemin où la concentration de phéromone est maximale. Les fourmis suivent cette méthode pour converger progressivement vers un seul chemin optimal entre leur nid et leur nourriture [33].

Au départ, l'important est de fournir le chemin le plus court et le plus optimal pour un véhicule, de la source S à la destination D. Cela peut être réalisé grâce aux techniques d'intelligence en essaim qui utilisent un comportement similaire aux organismes vivants de la nature.

### 2.3.2 L'application de la colonie de fourmis dans les systèmes de covoiturage :

Pour ajouter une valeur d'intelligence aux systèmes de covoiturage et rendre ces systèmes plus flexible et plus dynamique, et pour également atténuer ou annuler la souffrance de passer un temps supplémentaire dans le processus d'attribution des passagers par les conducteurs. Un algorithme inspiré de la nature avec un temps de réponse rapide et un effort de calcul réduit est particulièrement utile pour notre scénario d'étude.

Ils cherchent toujours une fonction permettant d'offrir une bonne utilisation de la route tout en assurant des embarquements moins couteux. Ils verront le problème d'embarquement une fonction du coût associe a des contraints multiples de l'embouteillage. Pour cela, nous appliquons l'algorithme de la colonie de fourmis comme un méta heuristique pour optimiser le problème d'embarquements a des contraintes multiples de l'embouteillage. Pour trouver un bon itinéraire de compromis contient des points optimaux d'embarquements (ou des points du dépôt), plusieurs colonies de fourmis sont incorporées pour optimiser indépendamment chaque contrainte, duquel la coopération entre eux est favorisée pour trouver une solution Pareto-optimal tout en respectant des contraintes multiples de l'embouteillage.

#### 2.3.2.1 La modélisation du problème :

Formellement, l'itinéraire de compromis a des contraintes multiples (ICMC) est modéliser comme un problème d'optimisation combinatoire multi-objectif (PMO), ce problème est paramétrer par le quadruplet ICMC(V,D,C,F). Duquel, V = {v1,v2,...,vn} décrit l'espace des objectifs du conducteur. L'ensemble D = {d1,d2,...,dn} représente n domaine d'objectifs. C'est un ensemble de contraintes sur V. De plus, on considère S = {x1,x2,...,xp} est l'ensemble des itinéraires de compromis admissibles satisfaisant les contraintes sur D. F = {f(x) : x ∈ S} décrit l'espace d'image de chaque objectif à optimiser. L'itinéraire de compromis est une solution non dominée (Pareto optimale), dans lequel il n'existe pas une solution x ∈ S telle que x domine x', c'est-à-dire :arg min cost(x) = {x'∈ S : ∀ x∈ S,cost(x')<cost(x)}.

#### 2.3.2.2 L'algorithme CarPooling-VC :

L'algorithme CarPooling-VC est inspirer de la colonie de fourmis capable d'identifier l'itinéraire de compromis associe a des contraints multiples de l'embouteillage. Il prend comme entre le nombre de colonie (NbColonies) qui représente le nombre des objectifs à optimiser, le nombre du structures de phéromone (NbPhrmStruct), les seuils $\tau_{min}$ et $\tau_{max}$ qui limitent le niveau de phéromone et le nombre des sièges vides (NbEmptySeats). L'algorithme est base sur l'abstraction du réseau routier en graph abstraite G(V,E), ou chaque intersection du réseau de trafic est un sommet dans le graphe abstraite. Le lien l(i,j) est le segment routier qui relie deux intersections i et j. De plus, les positions des passagers qui ont postule pour le covoiturage seront marquées comme des points d'embarquement sur le Graphe, où un point d'embarquement d'un passager n est note (pn). Ainsi, le cout de la transition entre deux passagers pm et pn selon un objectif i est estime par la fonction costi. A chaque itération de l'algorithme, un ensemble de K fourmis sont assumer d'explorer le graphe G(V, E) en cherchant parmi des points d'embarquement candidats un sous-ensemble de points qui satisfont un objectif j. Dans la phase de découverte, chaque fourmi utilise une fonction de transition probabiliste Psc et la densité de phéromones $\tau$ pour construire une solution pour le problème. Une fois qu'une solution est trouvée, la fourmi appréciera l'importance de la solution trouvée par rapport aux solutions trouvées par les membres de la colonie. Sur la base de cette estimation, la fourmi prépare une étape d'évaporation et définit la quantité de phéromone qui sera déposée sur chaque arc dans leur chemin de retour.

**Algorithme de CarPooling-VC :**

Require: *NbColonies*, *NbPhrmStruct*, *NbEmptySeats*, $\tau_{min}$, $\tau_{max}$ repeat
for each*C* ∈{1,2,...,*m*} do

for each ant *k* ∈ {1,2,...,*NbAnts*} do
*PList*←φ*CardP*
while *CardP*6 = 0 And *PList.Size*() 6 = *NbEmptySeats* do
Choose a boarding point $p_i \in CardP$ with the probability $P_s^c(p_i)$
*PList*←*PList*+{$P_i$}
*CardP*←*CardP*−{$P_i$} end while
end for
end for
for each *j* ∈ {1,2,..., *NbPhrmStruct*} do
for each $(p_m,p_n) \in$ {1,2,...,*PList*} do
Update the pheromone trail $\tau_{m,n}$ of the structure *j* end for

end for
for each *j* ∈ {1,2,..., *NbPhrmStruct*} do
for each $(p_m,p_n) \in G$ do
Evaporate the pheromone trail $\tau_{m,n}$ of the structure *j* if
$\tau_{m,n} < \tau_{min}$ then $\tau_{m,n} \leftarrow \tau_{min}$
end if
if $\tau_{m,n} > \tau_{max}$ then $\tau_{m,n} \leftarrow \tau_{max}$
end if end for
end for
until The maximum number of iterations

### 2.3.2.3 Construction de la solution :

Pour déterminer le prochain point d'embarquement, la fourmi choisir un point d'embarquement candidat ($p_n$) tel que $(p_m,p_n) \in S$, ou chaque point d'embarquement candidat a une probabilité $P_s^c(p_n)$ d'être choisie est donnée par :

$$P_s^c(p_n) = \frac{[\tau_s^c(p_n)]^\alpha [\eta_s^c(p_n)]^\beta}{\sum_{p_j \in Cand} [\tau_s^c(p_j)]^\alpha [\tau_s^c(p_j)]^\beta} \qquad (1)$$

α et β sont deux paramètres strictement positifs permettant de contrôler l'importance $\boldsymbol{\tau}_{cs}$ de la phéromone et de la fonction heuristique $\boldsymbol{\eta}_{cs}$, respectivement.

### 2.3.2.4 La fonction heuristique :

Dans le processus de construction d'une solution, la fourmi construira la solution en plusieurs étapes. A chaque étape, la fourmi définit un facteur heuristique qui évaluera l'importance de la fonction « coût » d'un objectif donne relatif à la sélection des passagers. En particulier, ce facteur est conçu pour évaluer le coût relatif à la piste courant en fonction des passagers candidats avec ceux qui ont sélectionnés, comme montre dans l'équation suivante :

$$\eta_c^i(P_m,P_n)=\frac{Cost(P_m,P_n)^i}{Cost(P_m,P_h)^i+Cost(P_h,P_n)^i} \quad (2)$$

#### 2.3.2.5 Les traces et la mise a jour du phéromone :

Lors de la tournée de la fourmi, une quantité´ e de phéromones doit être déposée sur la piste de retour pour identifie l'expérience passé d'une colonie (c) en ce qui concerne un objectif j. D'abord, une fourmi k utilise un facteur d'évaporation ρ pour éviter les solutions locales optimale et stimuler les autres fourmis d'explorer d'autres passagers, comme le montre l'équation 3.

$$\tau_{m,n}^k(c)=(1-\rho)\tau_{m,n}^k(c) \quad (3)$$

Ainsi, l'équation 4 permet de calculer la quantité des phéromones que doit mettre entre deux passagers, sa valeur se dépend sur le coût de la solution trouvée(Sj) par la fourmi k et la solution optimal (Sbest) explorées par d'autres membres de la colonie (c).

$$\Delta\tau_{m,n}^k(c)=\begin{cases}\frac{1}{1+cost_c^k(S_j)-cost_c^k(S_{best})} & si\\ 0 & sinon\end{cases} \quad (4)$$

La solution de compromis est assurer par la structure de phéromone Cs, ou la quantité des phéromones entre deux passagers (Pm,Pn) dans Cs est égale la somme des quantité de phéromones déposé entre (Pm,Pn) sur les autres structures de phéromones, comme mentionné dans l'équation 5.

$$\tau_{m,n}(c_s)=\sum_{k=1}^{NbColonies}\Delta\tau_{m,n}(k) \quad (5)$$

# 3 Méta-Model de la plateforme

Selon la figure 3.4 qui représente la partie de conception dans la plateforme proposée, les véhicules contient deux interfaces, l'un d'eux est s'appelle Ultran, est conçu pour la communication (V2C) par l'intermédiaire des réseaux cellular, l'autre interface s'appelle DSRC, est conçu pour les communications traditionnelle interne (V2V) et (V2I).Les segments de route qui est localisée entre deux intersection est contrôlé par les détecteurs de flux de trafic qui collecter le données pertinent le trafic routier tell que : la densité, le débit, la vitesse et les points d'embarquements, ces données est utilisées par le service de covoiturage. La plateforme de covoiturage est un service installé sur une machine virtuelle dans le cloud. « L'hyperviseur » est un logiciel installé dans les RSU, qui vise à créer pour chaque segment de la route un système virtuel pratique et véhiculer des données capturées vers une instance de machine virtuelle. La coordination entre ces composants former notre système de covoiturage qui annuler la souffrance des conducteurs des véhicules par trouver le plus court chemin en temps réel, est que les voyageurs ne prennent pas des temps long en attente.

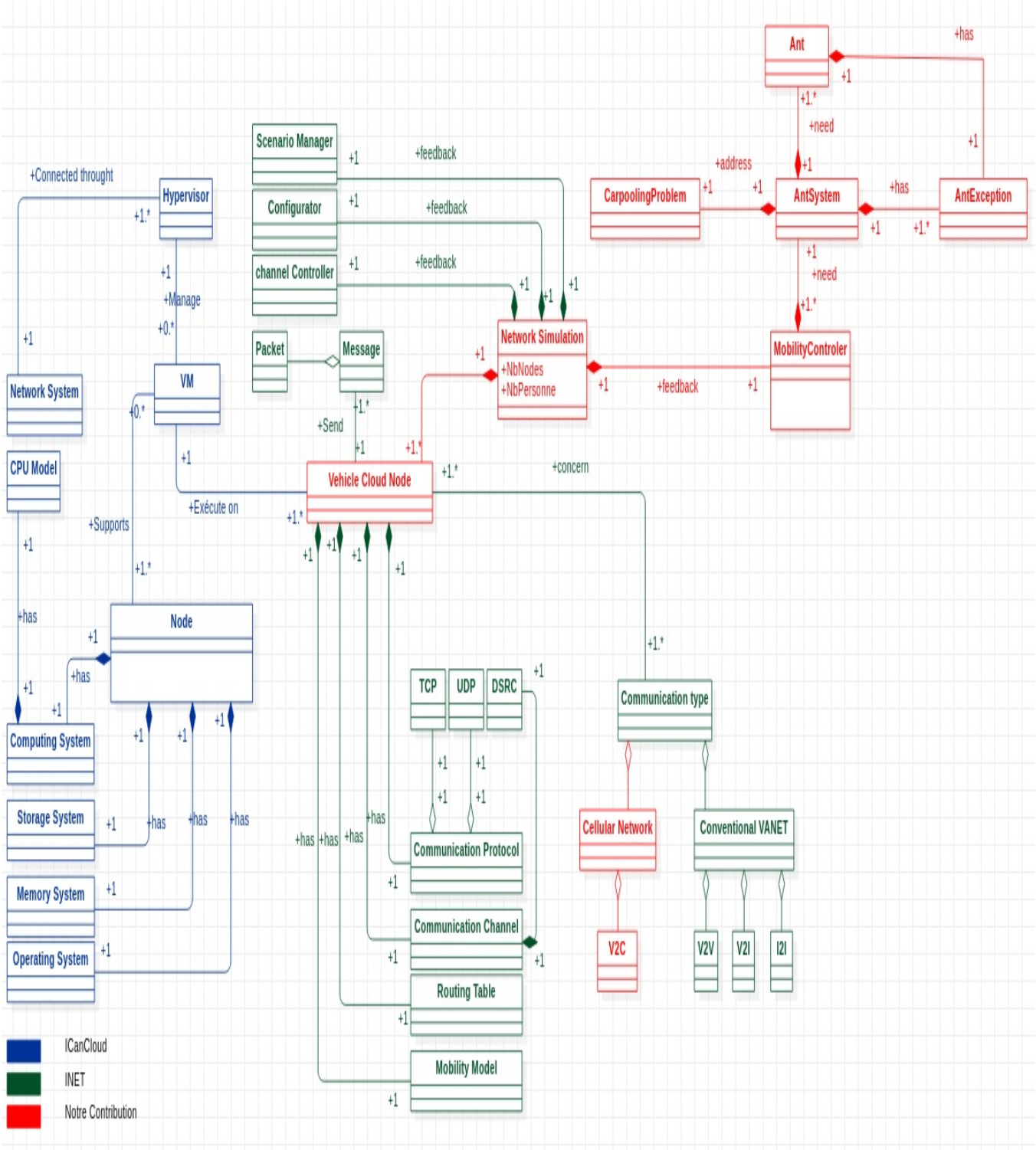



**Figure (3-4) : Diagramme de class de la plateforme.**

Les principales classes de la partie rouge illustré dans la figure (3-4) qui représente notre contribution sont cinq classes que l'on peut expliquer comme le suivant :

**Ant** : décrit les deux principaux comportements d'une fourmi qui sont : comment calculer la quantité de phéromones déposé et comment choisir le point d'embarquement suivante.

**AntSystem** : décrit comment les fourmis traitent entre eux collectivement comme un colonie, pour découvrir le plus court chemin du nid au nourriture.

**AntException** : exclut la fourmi qui n'a pas atteint la destinataire.

**CarpoolingProblem** : décrit le problème de covoiturage, ce problème doit être comprendré par la class AntSystem.cc pour répondre aux besoins de module mobilityCtr.ned.

**mobilityCtroler** : est la base de notre application, dans la réalité on le trouve au niveau d'un agence de covoiturage pour répondre aux demandes de clients qui sont des passagers et les serveurs qui sont des covoitureurs. il traite avec tous les autre modules de notre plateforme pour extraire ses paramètres et faire ses calcules.

# Conclusion :

Ce chapitre est dédié à la conception de notre plateforme, dans lequel on a présenté une description des outils utilisée pour atteindre les objectifs soulignés. Après cela on passera à l'explication de l'algorithme de la colonie de fourmi pour optimiser les solutions de notre scénario de covoiturage proposée avec l'explication des formules mathématique utilisé dans cette algorithme, et on le finie par le diagramme de classe qui est considérer comme une interface conceptuelle de notre application montrant ses principales fonctionnalités. Dans le prochain chapitre nous allons mettre en œuvre cette conception.

# Chapitre 04 :

# Réalisation

# Introduction :

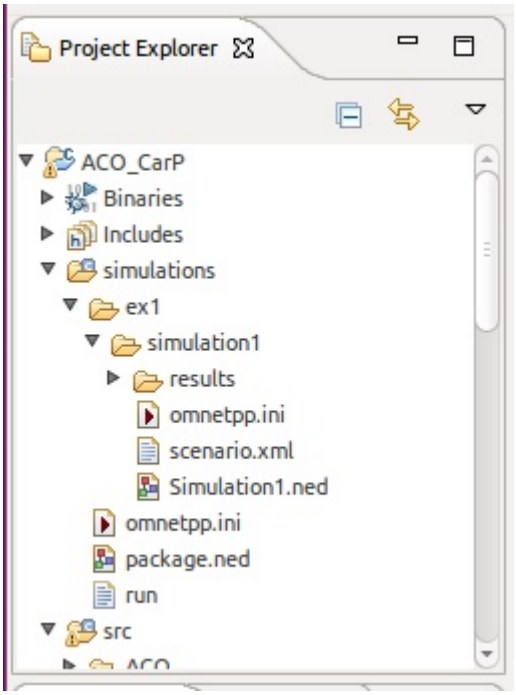
Dans le chapitre précédent, nous avons présenté l'étape de conception de la proposition. Dans ce chapitre, nous allons présenter la réalisation de l'extension proposer sous Omnet++. Le comportement de chaque classe de notre contribution sous le projet « ACO_CarP » dans les extensions« .cc », et la déclaration des paramètres et fonctions dans les extensions« .h ». nous allons démontrer aussi la description de réseaux dans le fichier « .ned » qui contient tous les composant nécessaire pour la fonctionnement de ce simulation, avec des définition concernant les modules prédéfinie de la plateforme INET. Nous allons également illustrer également les paramètres de chaque composant dans fichiers « .ned ».En fin, l'exécution de covoiturage avec la plateforme ACO_CarP.

# 1 Le projet ACO_CarP :

L'architecture de projet **ACO_CarP** se compose de plusieurs modules hiérarchiquement imbriqués qui sont :

- Le module système.
- Modules simples (les feuilles) : programmés en C++ encapsulant le comportement d'un réel d'un système. Pour chaque module simple correspond un fichier **.cc** et un fichier **.h**.
- Modules composés : constitués d'un ou de plusieurs modules simples ou des modules composés reliés entre eux. Les paramètres, les ports et les modules de chaque module sont spécifiés dans un fichier **.ned**.

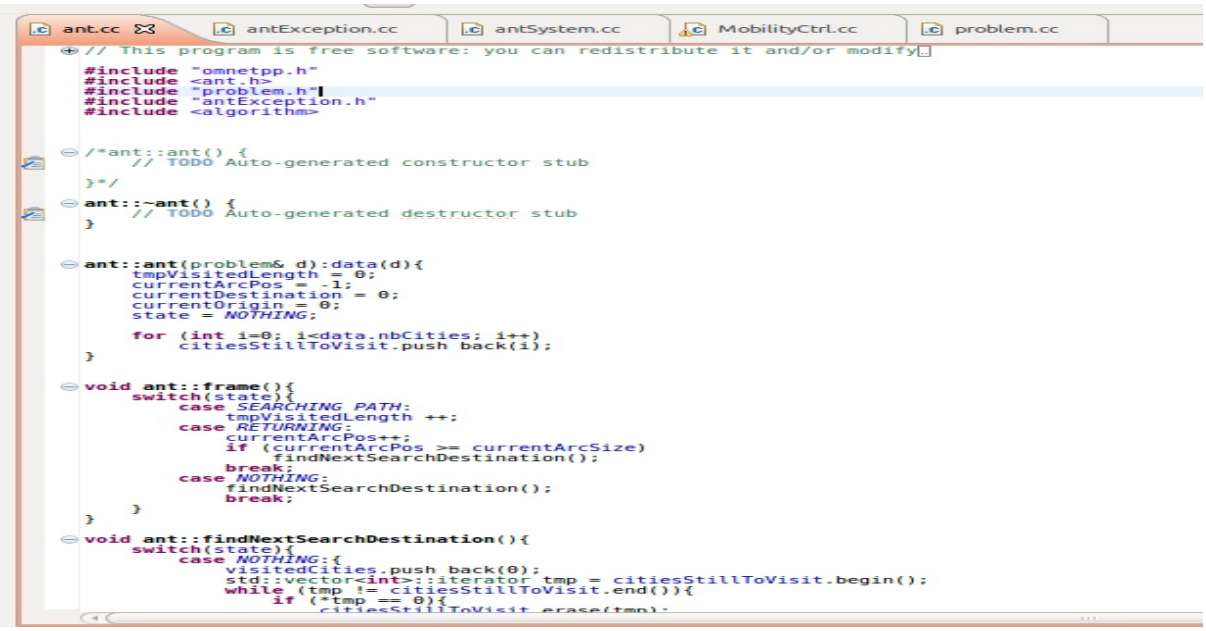
Dans la construction d'un nouveau programme de simulation, il y aura à chaque fois des informations chargées dynamiquement telles que la topologie du réseau à partir des fichiers **.ned**, et les configurations sont disponibles dans le fichier **.ini**.



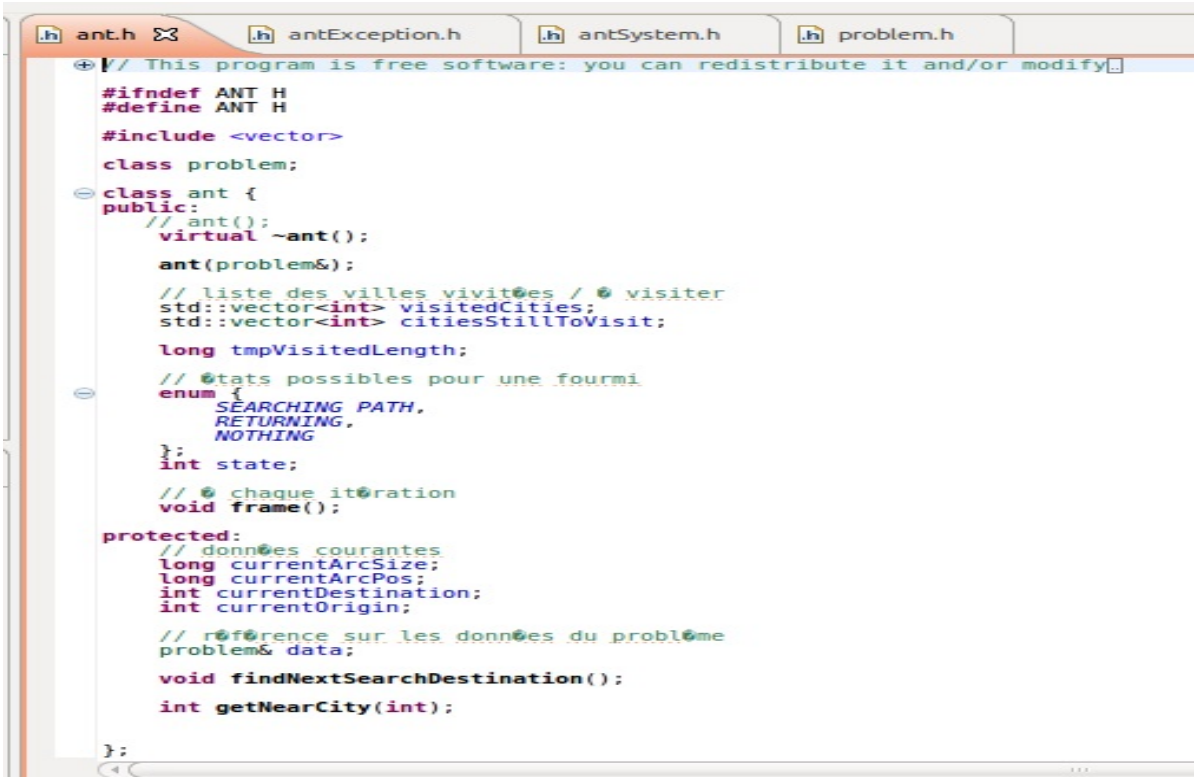
# 2 L'implémentation de l'algorithme :

Le captures d'images suivants démontre l'implémentions de chaque class de l'algorithme de la colonie de fourmi séparément aves ses paramètres:

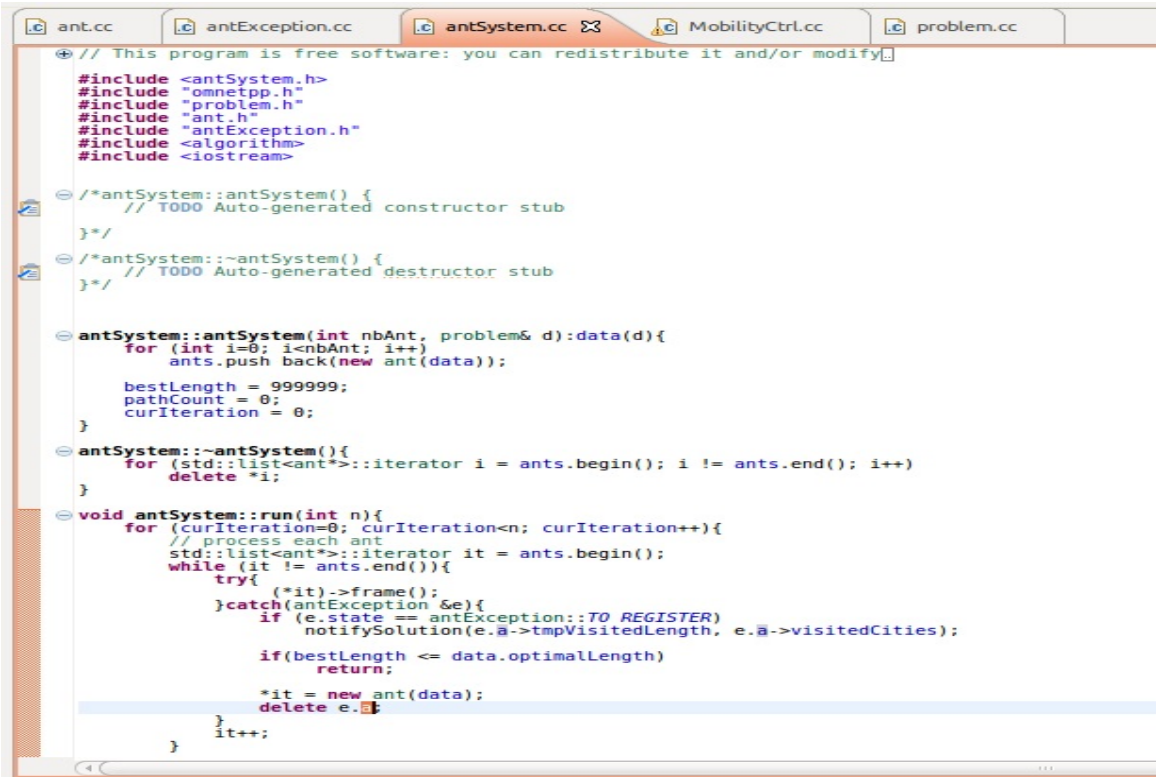
**Ant.cc** : pour la description de comportements de class **Ant**.




**Ant.h** : pour la description de paramètres et fonctions de class **Ant**.

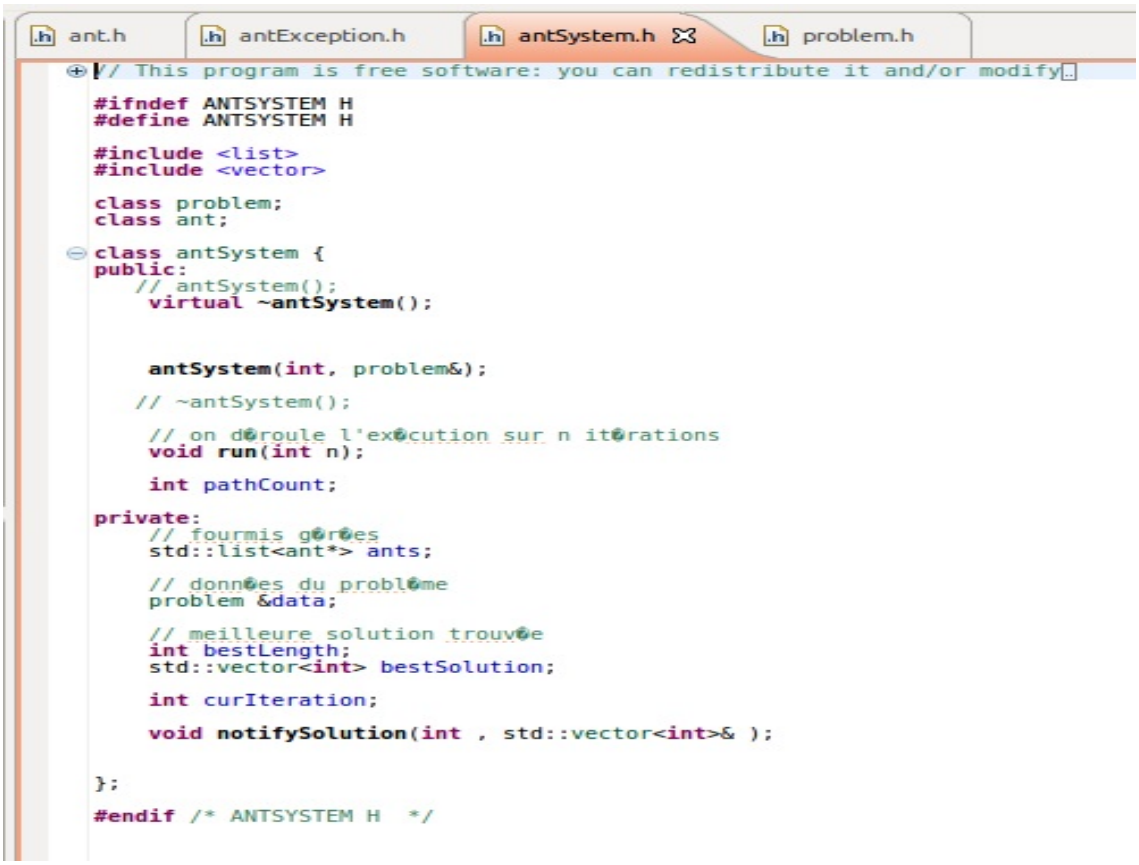


**AntSystem.cc** : pour la description de comportements de class **AntSystem**.



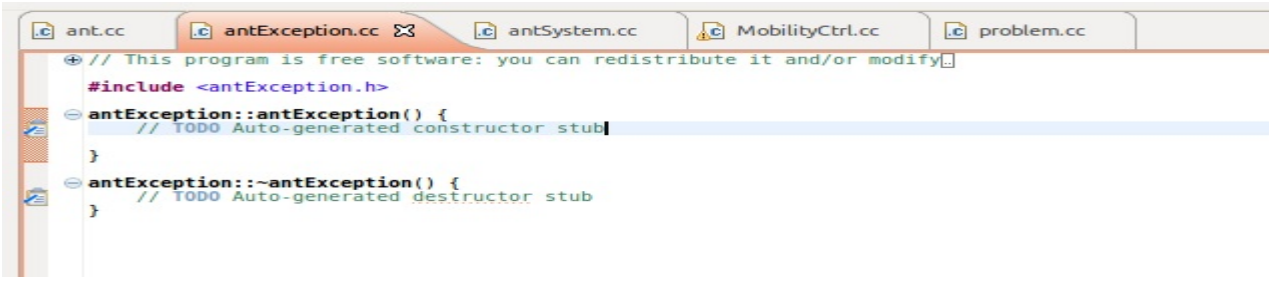
```
// This program is free software: you can redistribute it and/or modify...

#include <antSystem.h>
#include "omnetpp.h"
#include "problem.h"
#include "ant.h"
#include "antException.h"
#include <algorithm>
#include <iostream>

/*antSystem::antSystem() {
    // TODO Auto-generated constructor stub

}*/

/*antSystem::~antSystem() {
    // TODO Auto-generated destructor stub
}*/


antSystem::antSystem(int nbAnt, problem& d):data(d){
    for (int i=0; i<nbAnt; i++)
        ants.push_back(new ant(data));

    bestLength = 999999;
    pathCount = 0;
    curIteration = 0;
}

antSystem::~antSystem(){
    for (std::list<ant*>::iterator i = ants.begin(); i != ants.end(); i++)
        delete *i;
}

void antSystem::run(int n){
    for (curIteration=0; curIteration<n; curIteration++){
        // process each ant
        std::list<ant*>::iterator it = ants.begin();
        while (it != ants.end()){
            try{
                (*it)->frame();
            }catch(antException &e){
                if (e.state == antException::TO_REGISTER)
                    notifySolution(e.a->tmpVisitedLength, e.a->visitedCities);

                if(bestLength <= data.optimalLength)
                    return;

                *it = new ant(data);
                delete e.a;
            }
            it++;
        }
```

**AntSystem.h** : pour la description de paramètres et fonctions de class **AntSystem**.



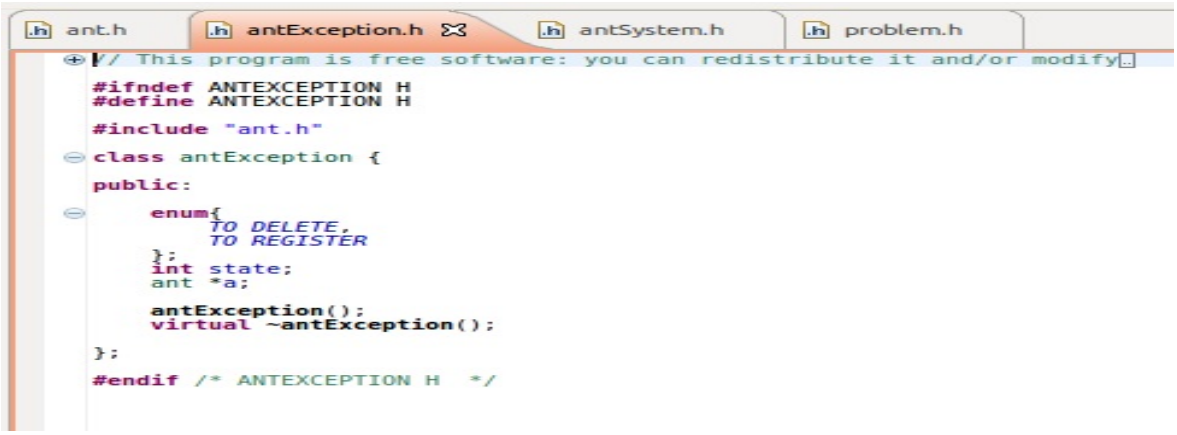
```
// This program is free software: you can redistribute it and/or modify...

#ifndef ANTSYSTEM_H
#define ANTSYSTEM_H

#include <list>
#include <vector>

class problem;
class ant;

class antSystem {
public:
    // antSystem();
    virtual ~antSystem();


    antSystem(int, problem&);

    // ~antSystem();

    // on d@roule l'ex@cution sur n it@rations
    void run(int n);

    int pathCount;

private:
    // fourmis g@r@es
    std::list<ant*> ants;

    // donn@es du probl@me
    problem &data;

    // meilleure solution trouv@e
    int bestLength;
    std::vector<int> bestSolution;

    int curIteration;

    void notifySolution(int , std::vector<int>& );


};

#endif /* ANTSYSTEM_H  */
```

**AntException.cc** : pour la description de comportements de class **AntException**.



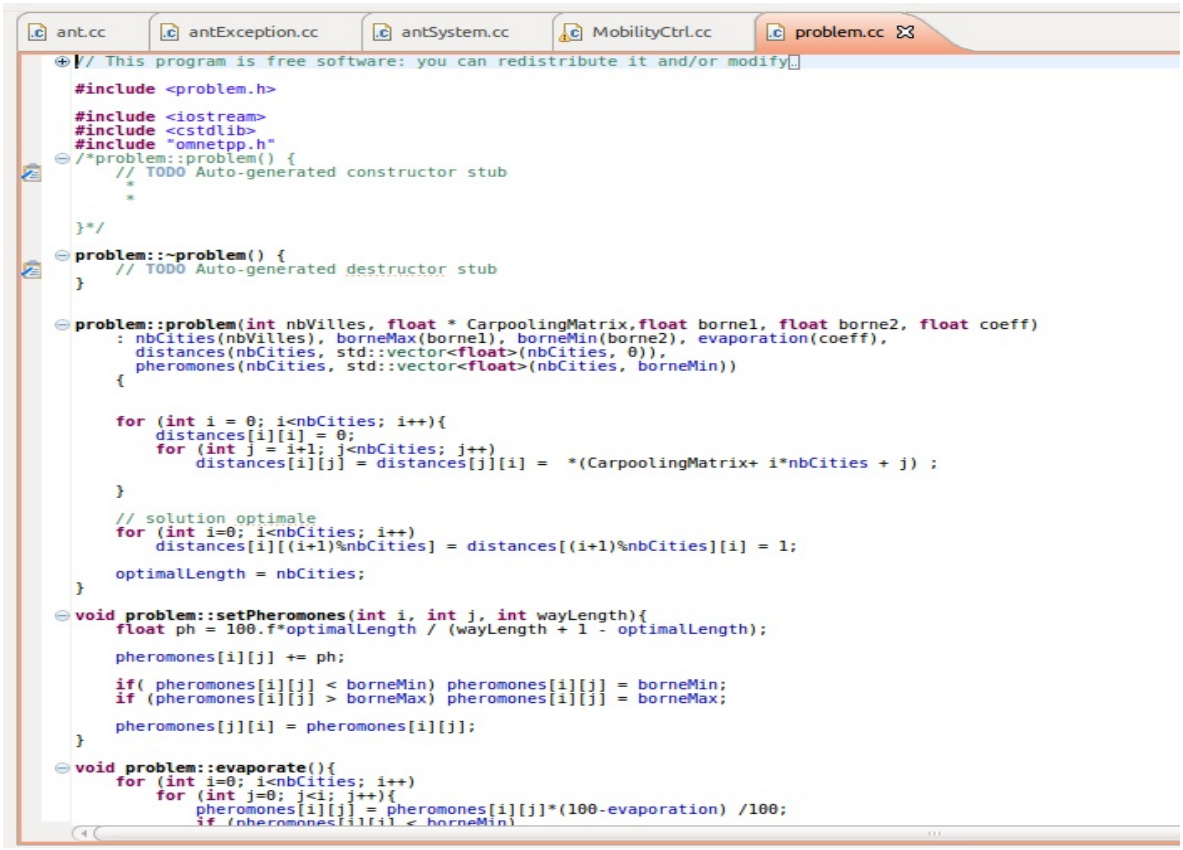
```
// This program is free software: you can redistribute it and/or modify
#include <antException.h>

antException::antException() {
    // TODO Auto-generated constructor stub

}

antException::~antException() {
    // TODO Auto-generated destructor stub
}
```

**AntException.h** : pour la description de paramètres et fonctions de class **AntException**.



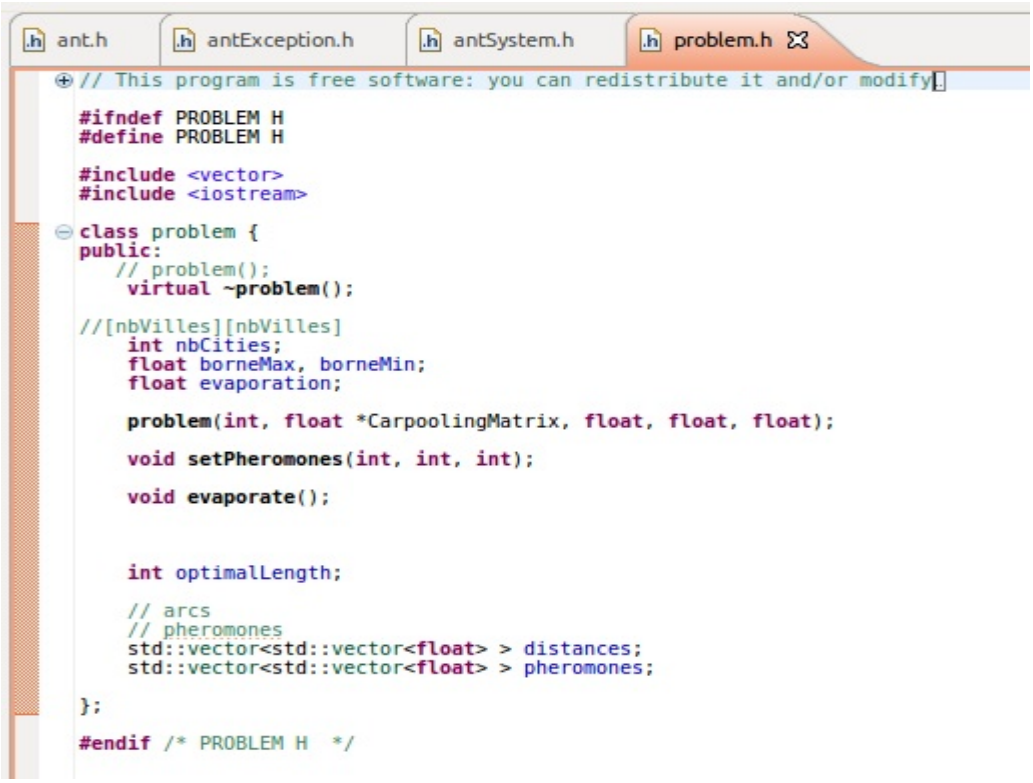
```
// This program is free software: you can redistribute it and/or modify
#ifndef ANTEXCEPTION_H
#define ANTEXCEPTION_H

#include "ant.h"

class antException {

public:

    enum{
        TO_DELETE,
        TO_REGISTER
    };
    int state;
    ant *a;

    antException();
    virtual ~antException();

};

#endif /* ANTEXCEPTION_H  */
```

**Problem.cc** : pour la description de comportements de class **CarpoolingProblem**.



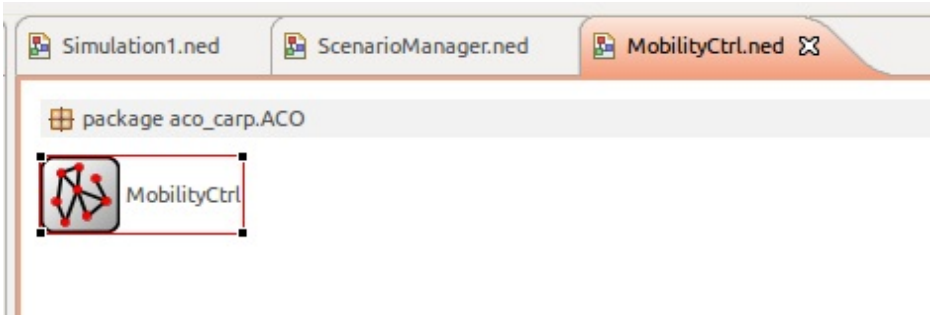
```
// This program is free software: you can redistribute it and/or modify

#include <problem.h>

#include <iostream>
#include <cstdlib>
#include "omnetpp.h"
/*problem::problem() {
    // TODO Auto-generated constructor stub
     *
     *

}*/

problem::~problem() {
    // TODO Auto-generated destructor stub
}


problem::problem(int nbVilles, float * CarpoolingMatrix,float borne1, float borne2, float coeff)
    : nbCities(nbVilles), borneMax(borne1), borneMin(borne2), evaporation(coeff),
      distances(nbCities, std::vector<float>(nbCities, 0)),
      pheromones(nbCities, std::vector<float>(nbCities, borneMin))
    {

    for (int i = 0; i<nbCities; i++){
        distances[i][i] = 0;
        for (int j = i+1; j<nbCities; j++)
            distances[i][j] = distances[j][i] =  *(CarpoolingMatrix+ i*nbCities + j) ;

    }

    // solution optimale
    for (int i=0; i<nbCities; i++)
        distances[i][(i+1)%nbCities] = distances[(i+1)%nbCities][i] = 1;

    optimalLength = nbCities;
}

void problem::setPheromones(int i, int j, int wayLength){
    float ph = 100.f*optimalLength / (wayLength + 1 - optimalLength);

    pheromones[i][j] += ph;

    if( pheromones[i][j] < borneMin) pheromones[i][j] = borneMin;
    if (pheromones[i][j] > borneMax) pheromones[i][j] = borneMax;

    pheromones[j][i] = pheromones[i][j];
}

void problem::evaporate(){
    for (int i=0; i<nbCities; i++)
        for (int j=0; j<i; j++){
            pheromones[i][j] = pheromones[i][j]*(100-evaporation) /100;
            if (pheromones[i][j] < borneMin)
```

**Problem.h** : pour la description de paramètres et fonctions de class **CarpoolingProblem**.




# 3 Module de gestion de mobilité :

Est la base de notre application, on le trouve inséré dans le fichier **Simulation1.ned**.



**MobilityCtrl.cc :** pour la description de comportements de module **MobilityCtrl**.

ant.cc | antException.cc | antSystem.cc | MobilityCtrl.cc | problem.cc

```
// This program is free software: you can redistribute it and/or modify

#include "MobilityCtrl.h"

#include "IMobility.h"

#include "problem.h"
#include "antSystem.h"
#include <iostream>
#include <cstdlib>


Define Module(MobilityCtrl);

void MobilityCtrl::initialize()
{
    fixedDistance = par("fixedDistance");
    this->nbVilles = getParentModule()->par("NbPersons");

    //EV << " le Nombre des noeuds connecté est " << endl;

       cTopology *Net = new cTopology("Net");

              std::vector<std::string> ListNedFiles;
             //  ListNedFiles.push back(getParentModule()->getNedTypeName());
              ListNedFiles.push back(getModuleByPath("Vehicle[]")->getNedTypeName());

              Net->extractByNedTypeName(ListNedFiles);

              EV << " le Nombre des noeuds connecté est " << Net->getNumNodes() << endl;
              for (int i = 0; i < Net->getNumNodes(); i++) {



                   // int j= Net->getNode(i)->getModuleId();

                    cModule *nd = Net->getNode(i)->getModule();

                    EV << " NB PAR  = " << nd->getNumParams() <<endl;

                      for (int s = 0; s < nd->getNumParams(); s++){
                          cPar &p = nd->par(s);
                        // EV << " PARAMETER ->  " << p.getName() <<"  =  " << p.str() <<endl;
                      }

                      IMobility *mobility = check and cast<IMobility *>(nd->getSubmodule("mobility"));
                      Coord senderposition = mobility->getCurrentPosition();
                      Coord senderSpeed = mobility->getCurrentSpeed();

                     // EV << "Speed = "<<senderSpeed<<"( X =  " << senderposition.x <<"  ,Y=  " << senderposition.y <<"  ,Z=  " << senderposition.z  <<")"<<endl;


                      cMessage *Msg = new cMessage("getCurrentPosition");
                        scheduleAt(simTime()+0.1, Msg);
```

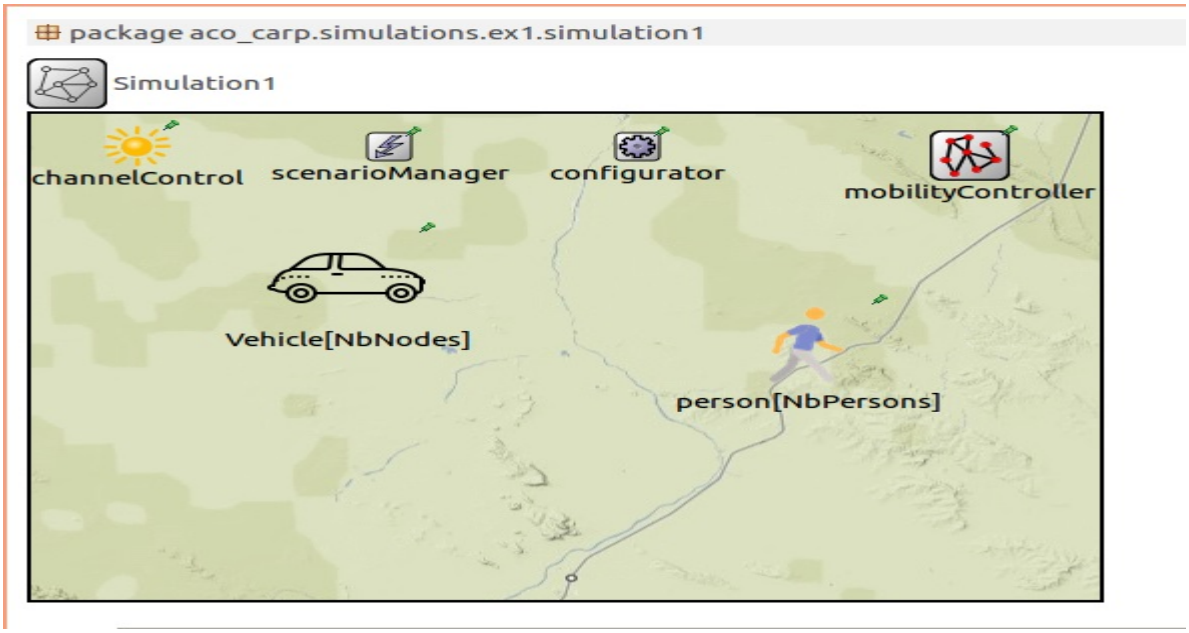
Les paramètres de module **MobilityCtrl** sont les suivants :

Defined parameters

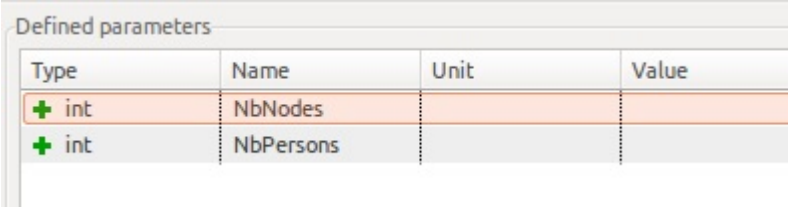
| Type | Name | Unit | Value |
|---|---|---|---|
| int | fixedDistance | | |

# 4 Le scenario :

On trouve la description de scénario dans le fichier **Similation1.ned**.



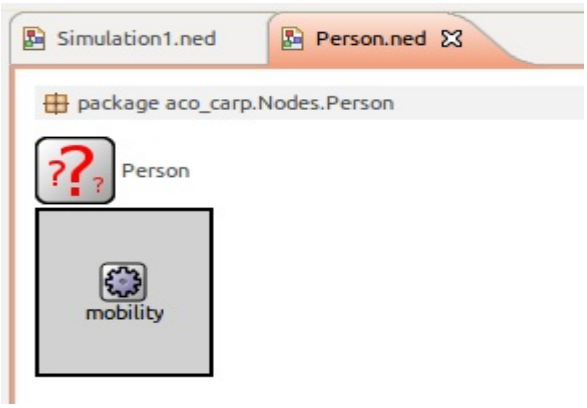

Les paramètres de **Similation1.ned** sont les suivants :




Le fichier **Similation1.ned** contient les modules suivants :

## 4.1 Le module Pesron :

**Person.ned** représente l'interface graphique de ce module :



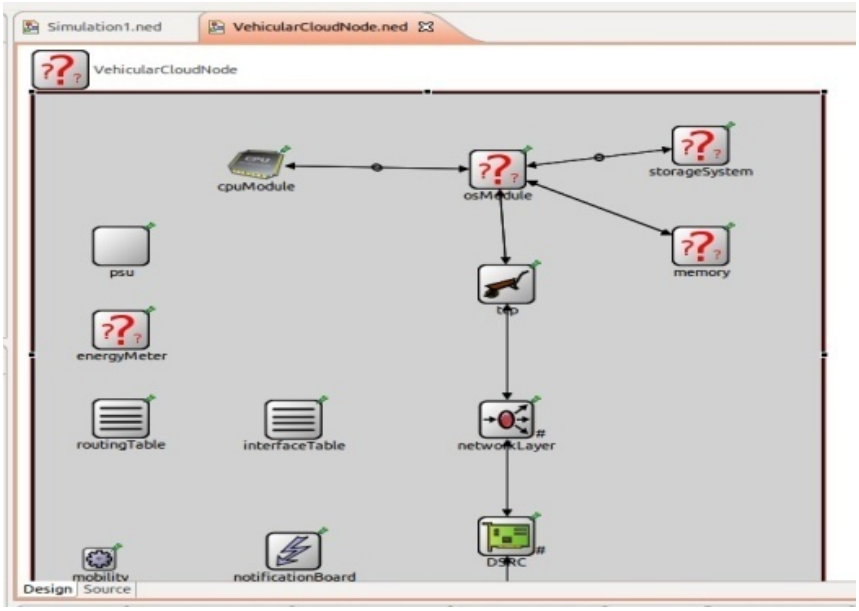
Les paramètres de module **Person** sont les suivants :

Defined parameters

| Type | Name | Unit | Value |
|---|---|---|---|
| string | mobilityType | | default("StationaryMobility") |

## 4.2 Le module VehicularCloudNode :

**VehicularCloudNode.ned** représente l'interface graphique de ce module :



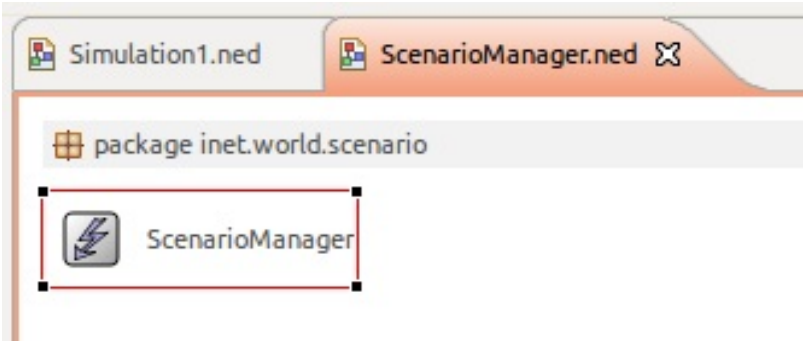

Les paramètres de module **VehicularCloudNode** sont les suivants :

Defined parameters

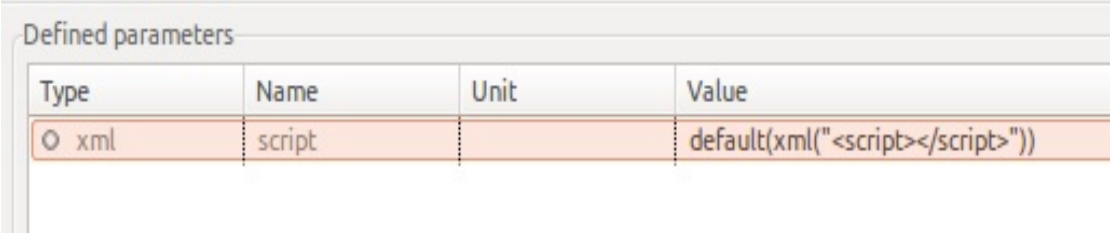
| Type | Name | Unit | Value |
|---|---|---|---|
| string | mobilityType | | default("StationaryMobility") |
| int | numExtInterfaces | | default(0) |
| int | numNetworkInterf | | default(1) |
| string | tcpType | | default(firstAvailable("TCP", "TCP_lwIP", "TCP_NSC", "TCP_None")) |
| bool | IPForward | | default(false) |
| string | ip | | default("") |
| string | routingFile | | default("") |
| int | namid | | default(-1) |
| bool | forwardMulticast | | default(false) |
| bool | storageNode | | default(false) |
| int | storage_local_port | | default(2049) |
| string | hostName | | default("") |

## 4.3 Le module ScenarioManager :

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 comportement transitoire. **ScenarioManager.ned** représente l'interface graphique de ce module :




Les paramètres de module **ScenarioManager** sont les suivants :




## 4.4 Le module ChannelControl :

ChannelControl a exactement une instance dans chaque modèle de réseau qui contient 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. **ChannelControl.ned** représente l'interface graphique de ce module :

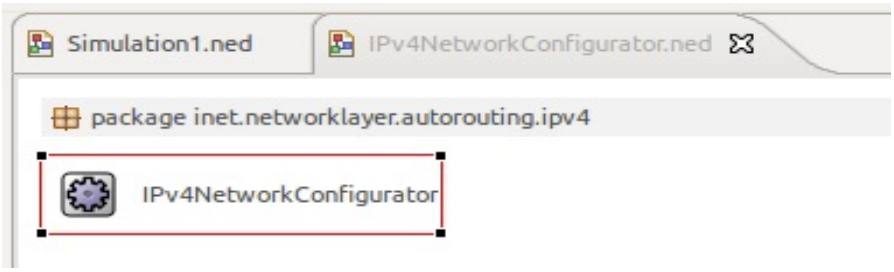
Les paramètres de module **ChannelControl** sont les suivants :

Defined parameters

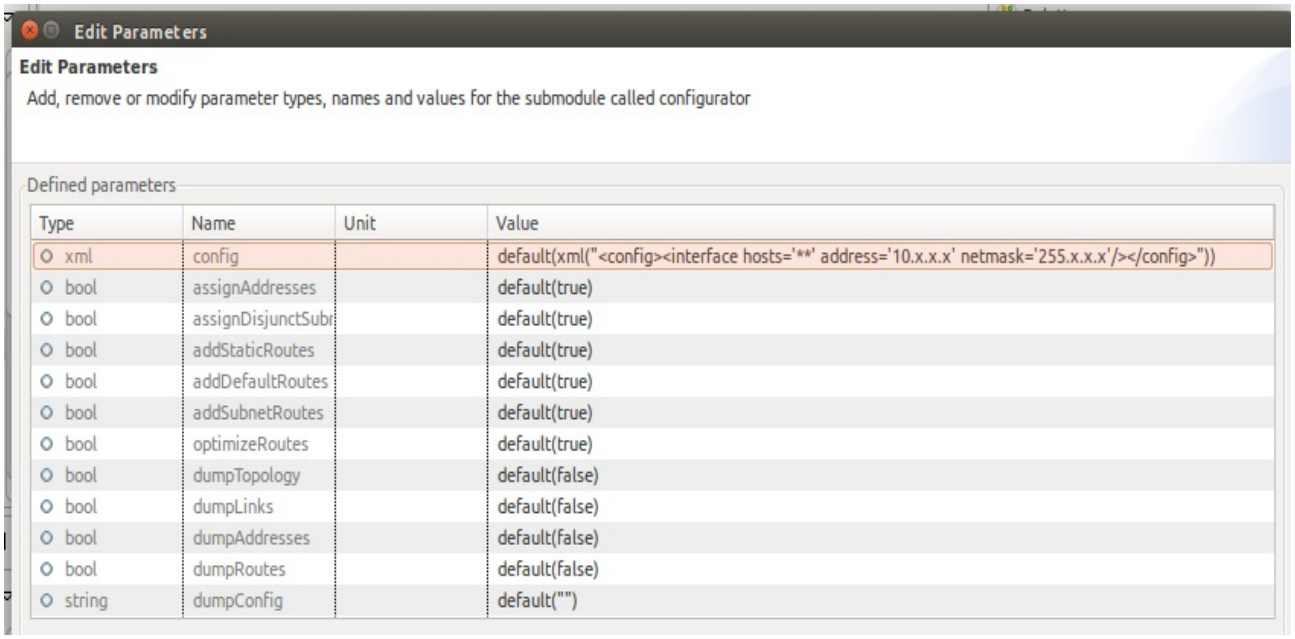
| Type | Name | Unit | Value |
|---|---|---|---|
| bool | coreDebug | | default(false) |
| double | pMax | mW | default(20mW) |
| double | sat | dBm | default(-110dBm) |
| double | alpha | | default(2) |
| double | carrierFrequency | Hz | default(2.4GHz) |
| int | numChannels | | default(1) |
| string | propagationModel | | default("FreeSpaceModel") |

## 4.5 Le module 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. **Configurator.ned** représente son interface graphique.

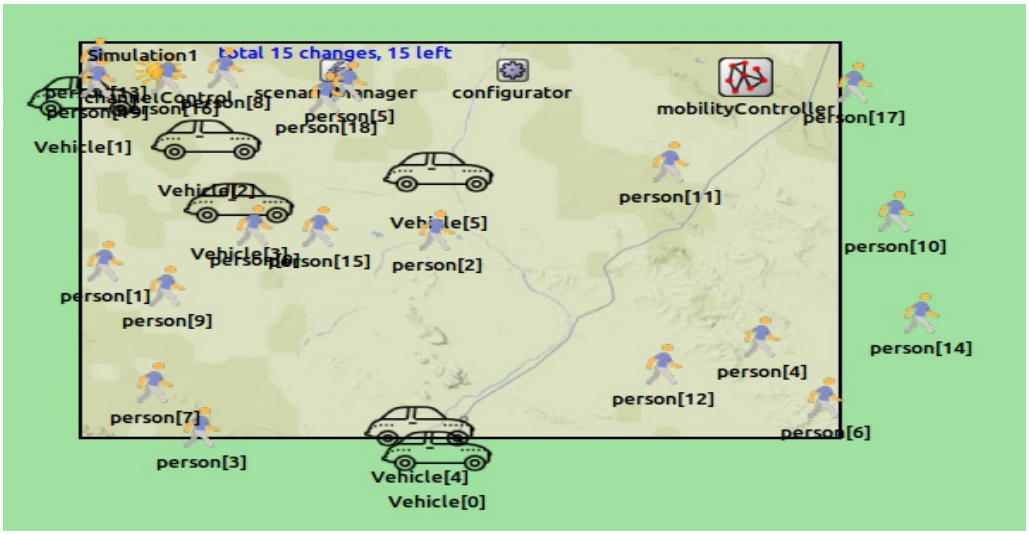



Les paramètres de module **Configurator** sont les suivants :



# 5 la simulation :

Capture d'image de l'application au cours d'exécution :




# 6 la solution :

Capture d'image illustre la solution le plus optimale :

```
The distance between P 1 and P3 = 214.499
The distance between P 1 and P4 = 275.729
The distance between P 1 and P5 = 177.357
The distance between P 2 and P3 = 135.014
The distance between P 2 and P4 = 230.609
The distance between P 2 and P5 = 75.8599
The distance between P 3 and P4 = 101.812
The distance between P 3 and P5 = 117.349
The distance between P 4 and P5 = 218.694
----------------Start CarpoolingMatrix-----------------
0  32.3354  279.206  246.672  307.311  203.406
32.3354  0  253.14  214.499  275.729  177.357
279.206  253.14  0  135.014  230.609  75.8599
246.672  214.499  135.014  0  101.812  117.349
307.311  275.729  230.609  101.812  0  218.694
203.406  177.357  75.8599  117.349  218.694  0
----------------End CarpoolingMatrix-----------------
999999
999999
98 332  La solution :
0,12,5,7,6,
1 chemins testes
```

# Conclusion :

Dans ce chapitre, nous avons présenté la réalisation de projet ACO_CarP dédier pour les VANETs-Cloud qui permet de traiter le problème de la congestion routier avec un system de covoiturage dynamique et intelligente.

# Conclusion Générale et perspectives

# Conclusion générale :

Les systèmes de transport intelligents et les réseaux véhiculaires n'en sont qu'à leurs balbutiements. Malgré toutes les évolutions dans le domaine des systèmes de transport intelligents, et tous les efforts dédiés par les chercheurs et les connaisseurs, la gestion de mobilité dans les réseaux VANETs sont encore un domaine très actif, mais à ce moment, Nous avons encore besoin de plus en plus d'idées pour le faire ressortir afin que les utilisateurs de ces réseaux puissent en bénéficier. Il faut donc encore une fois adapter des mécanismes et des protocoles spéciaux à ce contexte particulier et qui répondent aux exigences et spécificités de ces réseaux. La mise en œuvre réelle des réseaux véhiculaires fait de la simulation des scénarios réel le moyen le plus largement utilisé pour la conception et l'évaluation des solutions proposées. Le sujet abordé dans ce travail concerne la proposition et la modélisation d'une approche de gestion de mobilité dans un environnement VANET-CLOUD, le concept de cette proposition est basé sur l'intégration de l'algorithme de la colonie de fourmi pour l'optimisation avec l'idée de covoiturage qui est un moyen célèbre dans les pays développés depuis des décennies, pour trouver le plus court chemin et l'éventement des état de congestion routière.

Après avoir présenté en première chapitre les principaux concepts des systèmes de transports intelligents (STI) le contexte, le principe de fonctionnement, les services offerts par les STIs (les services liés à la sécurité routière, services liés à la congestion de trafics, services liés au confort du conducteurs) et les domaines d'applications des STIs. Nous avons ensuite étudié les réseaux VANET en termes de leurs caractéristiques et les services offerts. Et nous avons terminé ce chapitre en parlant de VANET-CLOUD qui est considéré comme un nouveau paradigme, en parlant de ces concepts de base, ces plateformes, son architecture en couches et les besoins futurs des applications.

En deuxième chapitre nous avons parlé du premier début d'étudier de près le comportement spatio-temporel de schémas empiriques de congestion du trafic par greenshield et ce quoi les théories de flux de trafic, sons phases et comment le modélisé. Et nous avons finie ce chapitre par une présentation des simulateurs de toutes sortes : les simulateurs de réseaux, les simulateurs de mobilités et les simulateurs intégrés.

Dans les deux dernières chapitres de notre travail, nous avons proposé une nouvelle approche de gestion de mobilité dédié aux réseaux VANET-CLOUD. Ce travail est destiné à ajouter un certain valeur d'intelligence et flexibilité au système de covoiturage a l'aide de l'algorithme de la colonie de fourmi optimisé ce qui aide de trouver le plus court chemin pour les voitures qui servent dans ce système, et pour également résolu le problème d'embouteillage routier par la remplir de tous les sièges de véhicule. La troisième chapitre est dédié a la partie de conception, et la quatrième chapitre a la partie de réalisation.

## **Perspectives** :

Parmi les obstacles rencontrés dans ce travail est :

- Nous n'avons pas la façon dont nous le faisons une agrégation de ressources de véhicules cloud pour appliquer ce scénario dans un environnement Vanet-Cloud.
- Nous n'avons pas le temps pour intégrer le Veins avec notre application et visualiser le comportement de mobilité de trafic routier en SUMO (Simulation Of Urbain Mobility).

# Bibliographie :

[01] A. Zear, P. K. Singh and Y. Singh. Intelligent Transport System: A Progressive Review, (2016).

[02] S. Ezell. Explaining International IT Application Leaderhip: Intelligent Transportation Systems, (2010).

[03] B. Lisan. Cloud Computing: Généralités et enjeux, (2015).

[04] R. Davies. L'informatique en nuage: Une vue d'ensemble des enjeux économiques et politiques, (2016).

[05] A. Aliyu et Al.: Cloud Computing In Vanets: Architecture, Taxonomy, And Challenges, (2017).

[06] S. Maerivoet and B. De Moor. Traffic Flow Theory, (2008).

[07] S. Hoogendoorn and V. Knoop. Traffic flow theory and modelling, (2013).

[08] Victor L. Knoop. Introduction to traffic flow theory, (2017).

[09] Mark Eric Mwiti Kimathi. Mathematical Models for 3-Phase Traffic Flow Theory, (2012).

[10] F. van Wageningen-Kessels et al. Genealogy of traffic flow models, (2014).

[11] Nayana. P .Vaity & Dnyaneshwar. V. Thombre. A survey on vehicular mobility modeling: flow Modeling, (2012).

[12] Prof. L.H. Immers and S. Logghe. Traffic flow theory, (2002).

[13] A. Spiliopoulou et al. Macroscopic traffic flow model calibration using different optimization algorithms, (2015).

[14] TU Delft OpenCourseWare. Chapter 9 : Macroscopic traffic flow models.

[15] A. Jamshidnejad, I. Papamichail, M. Papageorgiou, and B. De Schutter. A mesoscopic integrated urban traffic flow-emission model, pp. 45–83 , (2017).

[16] Jonathan Ledy. Thèse doctorat :Stratégie d'adaptation de liens sur canaux radios dynamiques pour les communications entre véhicules - Optimisation de la qualité de service, (2012).

[17] Teerawat Issariyakul • Ekram Hossain. Introduction to Network Simulator NS2, (2009).

[18] M.H. Kabir et al. Detail Comparison of Network Simulators, (2014).

[19] M. Małowidzki. Network Simulators: A Developer's Perspective, (2004).

[20] A. Varga and R. Hornig. An overview of the omnet++ simulation Environment, (2008).

[21] B. Abdellah et K. Menad. Mémoire de Master: Conception et Réalisation d'un simulateur dédie pour les réseaux véhiculaires, (2016).

[22] R.L. Bertini, R. Lindgren and S. Tantiyanugulchai. Application of PARAMICS Simulation At a Diamond Interchange, (2002).

[23] Owen, Zhang, Rao, and McHale. Traffic Flow Simulation Using Corsim, (2000).

[24] F.J. Martinez et al. CityMob: A Mobility model pattern generator for VANETs, (2008).

[25] J. H¨arri, F. Filali, C. Bonnet and M. Fiore. CityMob: VanetMobiSim: Generating Realistic Mobility Patterns for VANETs, (2006).

[26] D. Krajzewicz et al. Recent Development and Applications of SUMO – Simulation of Urban MObility, (2012).

[27] N.M. Mittal, S. Choudhary. Comparative Study of Simulators for Vehicular Ad-hoc Networks (VANETs), (2014).

[28] C. Sommer and F. Dressler. Progressing toward Realistic Mobility Models in VANET Simulations, (2008).

[29] S. Cloudin1, P. Mohan Kumar. Challenges on Mobility Models Suitable to Vanet, (2016).

[30] https://inet.omnetpp.org/Introduction.

[31] G.G. Castané, A. Nunez and J. Carretero. iCanCloud: A brief architecture overview, (2012).

[32] Y. Guo, G. Goncalves and T. Hsu. A Multi-Destination Daily Carpooling Problem And An Ant Colony Based Resolution Method, (2013).

[33] N. Melaouene and R. Romadi. An enhanced routing algorithm using ant colony optimization and VANET infrastructure, (2019).