N° d'ordre:02/2012-M/IN

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

En: INFORMATIQUE

Spécialité: Informatique Mobile

Par : Mlle. Hania GATI

SUJET :

Dissémination dans les Réseaux de Capteurs Véhiculaires.

Soutenu publiquement le 06/11/2012, devant le jury composé de :

Pr BadacheNadjib	**Professeur à USTHB**	**Président.**
Dr Moussaoui Samira	**Maître de Conférences A à USTHB**	**Directrice de mémoire.**
Dr Nouali Nadia	**Directrice de recherche au CERIST**	**Examinatrice.**

Remerciements

Je tiens tout d'abord à remercier vivement *le Professeur Nadjib BADACHE* et *Docteur Nouali Nadia*, pour m'avoir fait l'honneur d'accepter de participer à mon jury de soutenance. Je les remercie pour le temps qu'ils m'ont accordé et leurs disponibilités.

Je tiens à dire toute ma gratitude au Docteur *Moussaoui Samira* pour ses encouragements, sa disponibilité, ses idées, ses conseils et sa sympathie qui m'ont permis de mener à bien ce mémoire.

J'adresse également mes sincères remerciements à ma famille; mon père, ma mère et mes trois petits frères pour m'avoir aidé à surmonter tous les obstacles et à me forger à travers les difficultés vécues durant toute cette période de travail. C'est donc avec le plus grand plaisir que je remercie ma très chère maman qui n'a jamais cessé de me m'encourager à ne pas abandonner.

Je remercie également Monsieur Loucif Amirouche, Docteur Chawki Sahnine, Professeur Leila Mokhnache et Monsieur Samir Ouilmi pour leurs précieux conseils, soutiens et encouragements.

Merci aussi à tous mes amis Khadidja, Adel, Sukaina, Oussama, Maison, Reham et tant d'autres que je ne nommerai pas par manque de place et de peur de ne pas être exhaustive.

Enfin et avant tout, le grand et le vrai merci à Dieu qui m'a donné la force et la vie pour m'affranchir de tous les obstacles rencontrés.

À toutes les personnes qui m'ont aidé de
près ou de loin a présenté ce modeste travail,
Merci.

Résumé

Les récentes avancés dans le domaine des communications véhiculaires préparent le terrain pour de nouvelles applications de surveillance urbaine où des véhicules détectent et capturent continuellement des événements, traitent des données et acheminent des messages vers des stations de base sur la route (V2I) ou vers d'autres véhicules voisins (V2V). Ils suscitent un intérêt certain aussi bien en Europe qu'au Japon et en Amérique du Nord, dans le but de fournir de nouvelles technologies capables d'améliorer la sécurité et l'efficacité des transports routiers.

Cependant, l'environnement des réseaux de capteurs véhiculaires posent des problématiques nouvelles liées à leurs spécificités. Ainsi peu de travaux ont été réalisés pour répondre aux problèmes de ces environnements émergents. Parmi ces protocoles, nous décrivons ceux qui ont été développés pour le contrôle urbain proactif dans les VSNs et qui exploitent la mobilité des véhicules afin de permettre la détection, le traitement et le filtrage des données capturées.

Se basant sur une approche de protocole décentralisé, nous nous intéressons dans cette thèse à la dissémination de données dans les réseaux de capteurs véhiculaires dans un environnement urbain. Notre objectif est de proposer une solution ad hoc répondant à la fois aux exigences et besoins technologiques des cas d'utilisation envisagés (principalement des services d'information et de confort), mais aussi et surtout aux contraintes liés aux communications inter-véhiculaires ad hoc tel que la surcharge réseau, connectivite intermittente, etc.

Notre démarche consiste à prendre en compte un paramètre clé qui influence le bon fonctionnement du réseau de capteurs véhiculaires, à savoir la communication optimisée entre les nœuds véhicules. Notre solution a pour objectif principalement d'assurer une plus grande distribution des traitements et une réduction de la surcharge réseau. Notre proposition se base sur l'exploitation des connexions entre des véhicules de contrôle pour assurer l'élimination au maximum du nombre d'opérations collectes et leurs traitements récurrent. Elle tire partie du nombre des rencontres opportunistes lors du déplacement des nœuds véhicules ce qui permet de la diffuse/collecte des données de façon épidémique vers une population bien défini dans le réseau ad hoc urbain.

Les mesures de simulation effectuées ont montré que notre protocole permet un meilleur temps de collecte, avec un coût réduit en overhead et une bonne gestion des ressources réseaux dans un environnement urbain décentralisé.

Mots clés: Réseaux de capteurs véhiculaires, Collecte de données, dissémination de données, accès à l'information.

Table des matières

Table des figures

Table des tableaux

Introduction

Les réseaux de capteurs véhiculaires présentent un nouveau paradigme pour les réseaux de capteurs conventionnels. Ils peuvent être construits sur des réseaux véhiculaires en équipant les véhicules avec des capteurs ou en considérant des capteurs déployés tout au long des routes. Contrairement au WSNs (Wireless Sensor Networks), ce type de réseau n'a pas de contraintes sur les ressources matérielles. Ils ne sont pas limités en termes de puissance de calcul, de capacité de stockage mémoire et d'énergie. Les nœuds d'un réseau VSN (Vehicular Sensor Networ) peuvent collecter, entreposer, et traiter des quantités importantes de données capturées. Ces dernières seront par la suite envoyées à un nœud puits dans des conditions prédéterminées afin d'assurer des services tels que l'aide à la prise de décision pour les conducteurs.

La dissémination de données concerne la délivrance d'information à des entités intéressées ou concernées par ces informations. Ces informations portent sur des évènements observés tels que le taux de pollution dans l'air, la présence de produits chimiques, etc. Toutefois, cette dissémination doit assurer certains facteurs de qualités de service. Ces exigences par rapport au service de dissémination varient selon le type d'applications visées. En effet, les applications critiques ont, souvent, pour facteurs prioritaires la fiabilité et les délais de livraison. Alors que des applications dites de confort favorisent souvent le facteur coût en communication.

L'évolution des technologies de communication sans fil et de l'équipement des véhicules actuels permettent la réalisation de nouvelles applications de contrôle, de surveillance et, d'aide aux conducteurs en environnement urbain. En effet, de nos jours, les véhicules ont la possibilité d'observer, de détecter et de capturer des évènements. Ces véhicules jouent aussi le rôle de moyens efficaces pour la collecte et le transport des données. Les paquets de données peuvent être acheminés des véhicules vers des infrastructures routières ou station de base (communication V2I). Ils peuvent aussi être transportées de proche en proche d'un véhicule à l'autre (communication V2V) jusqu'à atteindre la destination. Cependant, ce type d'environnement suscite de nouveaux défis à relever notamment dans la prise en compte des connexions volatiles, de la répartition

du réseau et du risque de congestion important suite à une densité véhiculaire en zone urbaine.

Après l'étude des caractéristiques des réseaux de capteurs véhiculaires, nous avons défini les différents aspects de la problématique de dissémination/collecte de données sur un tel réseau. Nous avons, par la suite, présenté les quelques protocoles existant dans la littérature dont le souci majeur consiste à répondre aux caractéristiques fondamentales des VSNs. Peu de travaux ont été réalisés pour répondre au problème. Parmi ces propositions, nous citons le système MobEyes qui a été développé pour le contrôle urbain proactif dans les VSNs. MobEyes exploite la mobilité des véhicules équipés de caméras vidéo et d'une variété de capteurs afin de permettre la détection de certains événements, le traitement et le filtrage des données capturées. Pour éviter de diffuser directement aux autorités toutes les données capturées. MobEyes propose que les données capturées restent sous le contrôle des nœuds mobiles (i.e., mémoire distribuée) pour les transmettre, au besoin, suite à la réception d'une requête.

L'approche de collecte/dissémination du système MobEyes est une approche opportuniste qui consiste à propager les données à travers les nœuds véhiculaires mobiles qui diffusent de façon épidémique les données aux autres véhicules jusqu'à atteindre les infrastructures de contrôle. Cette diffusion de données est faite, par un nœud, à travers la dissémination d'un rapport ou résumé des données locales dit ORD (Opportunistic Rapport Dissémination). Un nœud mobile diffuse ses rapports contenus dans sa mémoire locale aux nœuds rencontrés et obtient de manière épidémique de nouveaux rapports en échange. D'où un enrichissement de la base de données locale. Les protocoles utilisés par des nœuds MobEyes pour la diffusion/collecte de résumés exploitent la mobilité intrinsèque des véhicules et des communications à un seul-saut entre les nœuds pour propager à moindre coût les données.

Ce système considère des véhicules usagers et un type particulier de véhicules représentés par les véhicules des agents de police qui se chargent de la collecte de données. Ainsi, un résumé qui a déjà été collecté par un véhicule agent peut être collecté une seconde fois, par un autre agent pouvant favoriser un ''broadcast storm problem'' dans le cas d'une forte densité réseau par exemple. C'est en se basant sur ces constations que nous avons tenté de proposer une solution améliorée destinée à mieux exploiter les

possibilités de ces environnements et à réduire de manière significative la surcharge du réseau. En effet, l'idée consiste à établir un échange proactif des résumés entre véhicules agents et ce de façon opportuniste.

Les études de simulation et d'évaluation qui ont été effectuées à l'aide du simulateur OMNET++ ont montré l'intérêt du schéma proposé. Les mesures obtenues montrent que cette solution permet d'avoir un temps de collecte optimisé -des données existantes- par rapport à la solution initiale. Les mesures démontrent aussi que la solution induit un overhead réduit. La solution offre en outre une organisation plus tolérante aux répartitions des réseaux et aux ruptures de connexions vu qu'elle offre une plus grande décentralisation. L'utilisation de notre protocole n'est pas limitée à la collecte de données pour une autorité, mais peut aussi servir pour un large spectre d'applications: par exemple, les véhicules peuvent mesurer des niveaux de pollution et rassembler des données concernant le trafic, comme la congestion et les conditions de route.

Le document est structuré en quatre chapitres. Le chapitre 1 fournit un aperçu sur les réseaux de capteurs ou WSNs, les réseaux ad hoc véhiculaires ou VANETs, et particulièrement, les réseaux de capteurs véhiculaires VSNs qui relient les technologies des capteurs, des véhicules et des communications sans fil. Le chapitre 2 détaille les spécificités et les caractéristiques des VSNs, la dissémination et l'accès aux données dans ces réseaux ainsi que les solutions qui ont été proposées dans cet environnement. Le chapitre 3 présente notre protocole pour une dissémination de données optimisée basée sur la collaboration entre des véhicules de contrôle de la sécurité urbaine tels que les véhicules agents de la police par exemple. Enfin, le chapitre 4 propose une évaluation des performances par simulation de notre solution.

Chapitre I: Les réseaux de capteurs véhiculaires.

1. Introduction:

L'évolution de la technologie dans le domaine de la microélectronique et des communications radiophoniques sans fil, ont permis la fabrication d'unités de captage de très petite taille capables de prélever des évènements, de les traiter et, de transmettre et recevoir des paquets de données grâce à des connexions établies sur un réseau ad hoc. La vulgarisation de ces composants est surtout due au coût raisonnable de leur production [ABT04].

D'autre part, les constructeurs automobiles produisent des véhicules de plus en plus sophistiqués et équipés de divers capteurs dans un souci d'offrir plus de confort et de sécurité aux conducteurs. L'association des capteurs, des véhicules et des connexions sans fil offrent des perspectives très attrayantes pour la supervision de la circulation sur les routes et pour la proposition de nouveaux services aux conducteurs. Les réseaux de capteurs véhiculaires sont constitués de nœuds mobiles véhiculaires et de nœuds capteurs d'observation et de captage des évènements. Ils peuvent, donc, aussi participer à des applications d'observation de l'environnement comme, par exemple, le prélèvement du degré de pollution dans l'air ou l'état de congestion de la circulation.

2. Les capteurs :

Avant de définir les réseaux de capteurs véhiculaires nous nous intéressons à définir dans ce qui suit : les capteurs.

2.1. Méthode de construction

Deux approches sont proposées dans la construction d'un capteur:

- **Capteurs intégrés**: Ils intègrent l'unité de captage directement dans le capteur. Cette méthode est plus économique et plus robuste, elle se traduit aussi par une diminution visible de la taille du capteur (Figure 1.1).

Figure1.1: Capteurs intégrés [ZIN12]

- **Capteurs de carte**: L'unité de captage est connectable au capteur à travers des bus d'extensions (Figure 1.2).Cette méthode est plus dédiée à la demande de l'utilisateur qui peut connecter des capteurs de différents types suivant le domaine d'application du même capteur.

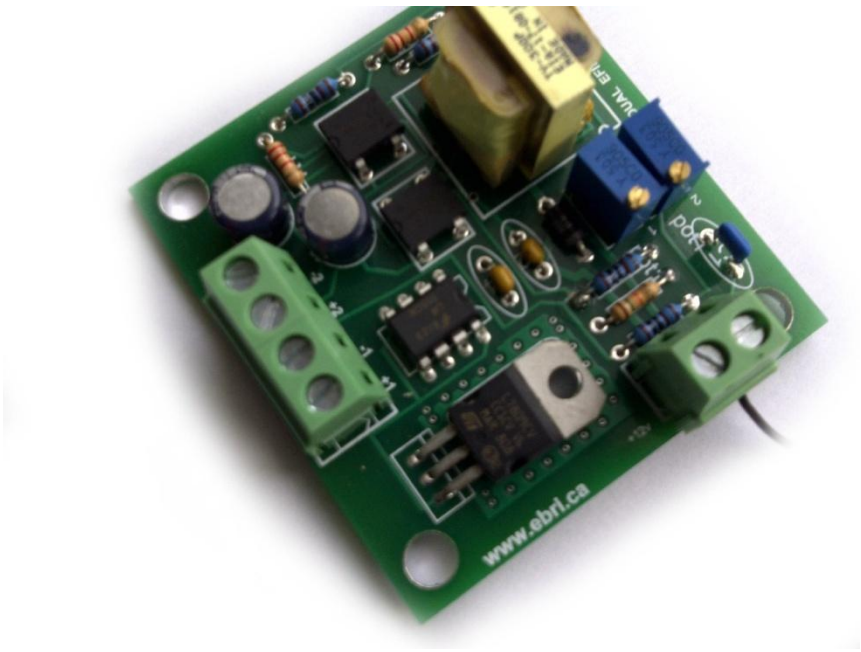

Figure 1.2: capteurs carte [HEA12]

2.2. Caractéristiques physiques d'un capteur:

2.2.1. *Petite taille:*

La taille d'un capteur varie selon la composition entre $1cm^3$ dans le cas d'un capteur intégré (Figure1.1), et 4cm*4cm dans le cas d'un capteur à carte avec des extensions possibles. Sa taille relativement petite lui permet d'être placé dans beaucoup d'endroits avec discrétion.

2.2.2. *Taux de transfert limité:*

Un faible débit de données n'est pas toujours handicapant pour un réseau de capteurs où les fréquences de transmission ne sont pas importantes. Le taux de transfert dans les capteurs est limité à 250 Kbps, taux théorique pour le protocole ZigBee sur une fréquence de 2.4 GH (Tableau 1.1).

Protocole	**Zigbee**	**Bluetooth**	**Wi-Fi**
IEEE	802.15.4	802.15.1	802.11a/b/g/p
Besoins mémoire	4-32 Kb	250 Kb +	1 Mb +
Autonomie avec pile	Années	Jours	Heures
Nombre de nœuds	65 000+	7	32
Vitesse de transfert	250 Kb/s	1 Mb/s	11-54-108 Mb/s
Portée	100 m	10-100 m	300

Tableau 1.1: Tableau comparatif des standards IEEE

2.2.3. *Faible coût de fabrication:*

Le faible coût de production constitue un avantage considérable pour les capteurs. Cet avantage est la raison de leur généralisation et diversification.

2.2.4. *Autonome et adaptative:*

Un capteur intégré est complètement autonome. Il peut effectuer toutes les tâches qui lui sont affectées (capter, communiquer…) sans avoir besoin d'un composant tiers. En revanche, il est moins adaptatif qu'un capteur à carte qui

est très adaptatif et peut recevoir plusieurs capteurs pour différentes grandeurs physiques.

2.3. Architecture matérielle d'un capteur:

Un nœud capteur contient quatre unités:

- l'unité de captage
- l'unité de traitement
- l'unité de transmission
- l'unité de contrôle d'énergie [AkK04].

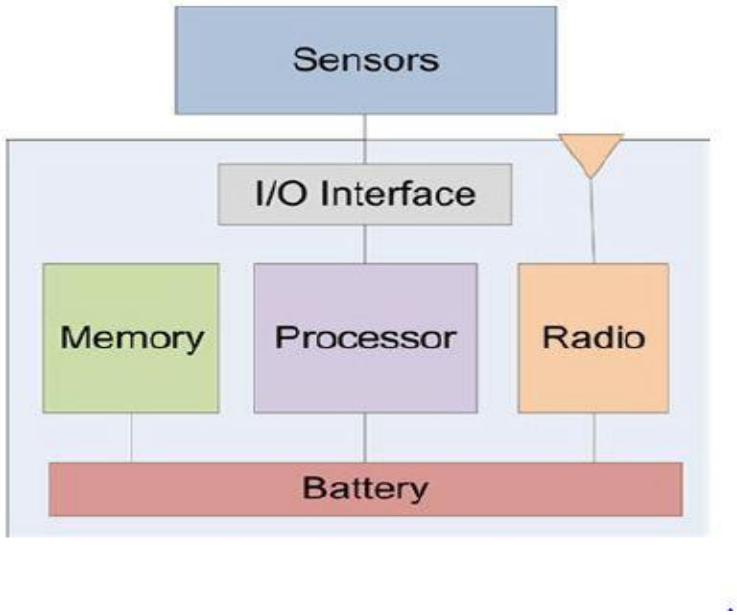

Figure 1.3: Architecture matérielle d'un capteur.

Un capteur peut, également, contenir, suivant son domaine d'application, des modules supplémentaires tels qu'un système de localisation (GPS), ou bien un système générateur d'énergie (cellule solaire).Il peut aussi à bord d'un véhicule être alimenté par la batterie de celui-ci. On peut même trouver des capteurs, un peu plus volumineux, dotés d'un système mobilisateur chargé de déplacer le capteur en cas de nécessité (Figure 1.3).Dans les VSNs Les capteurs de véhicules sont généralement alimentés par la batterie de démarrage du véhicule. Par conséquent, l'économie d'énergie est importante, mais n'est pas indispensable lorsque les capteurs sont déployés dans les véhicules.

2.4. Types de capteur:

Différents types de capteurs peuvent être ajoutés sur un capteur pour étendre ses capacités tels:

- Capteurs de distance: ultrason, micro-onde, triangulation…
- Capteurs de lumière: photodiode, capteur photographique…
- Capteurs MultiMedia: microphone, camera, vidéo, hydrophone…
- Capteurs chimiques: pour polluants atmosphériques…

Des capteurs de température, de pression, de débit ou encore d'inertie (pour le calcul de l'accélération) et d'autres peuvent collaborer en réseau dans divers domaines d'applications.

3. Les réseaux de capteurs sans fil (Wireless Sensor Network):

Les réseaux de capteurs sans fil ou Wireless Sensor Networks (WSN) sont considérés comme un type spécial de réseaux ad hoc mobiles (MANET). Les nœuds de ce type de réseaux consistent en un grand nombre de capteurs capables de collecter et de transmettre des données environnementales d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils sont dispersés aléatoirement à travers une zone géographique, appelée champ de captage, qui définit le terrain d'intérêt pour le phénomène capté. Les données captées sont acheminées grâce à un routage multi-saut (multi-hop) à un nœud considéré comme un "point de collecte", appelé nœud puits (ou sink). Ce dernier peut être connecté à l'utilisateur du réseau via Internet ou un satellite.

Ainsi, l'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et collecter les données environnementales captées par le biais du nœud puits [POK00].

Les récentes avancées technologiques dans la miniaturisation et la communication ont permis la création de Réseaux de Capteur Sans fil (WSN) composés d'un grand nombre de dispositifs numériques peu couteux, qui intègrent des capteurs,

des processeurs et des émetteurs/récepteurs radios sur une portée de quelques centimètres.

Le but principal des WSNs est de livrer rapidement aux utilisateurs finaux l'information surveillée du monde physique, en atteignant une haute précision tout en respectant les limitations spécifiques aux nœuds WSN, car ces derniers fonctionnent généralement dans des environnements difficiles empêchant le déploiement d'une quelconque infrastructure supportant la communication sans fil.

Pour former une base de détection adéquate et fiable, les WSN incluent généralement un grand nombre des dispositifs. En fait, dans la littérature les résultats théoriques montrent que le comportement coordonné des centaines de nœuds de WSN reparties près du phénomène (intéressant à surveiller) peut améliorer fortement la précision de détection comparé aux macro-capteurs centralisés.

Cependant, l'exploitation d'un grand nombre de dispositifs écarte certainement la possibilité que des opérateurs humains puissent configurer manuellement et gérer un WSN: il y a là un besoin de solutions d'auto-configuration et auto- organisation (WSN devraient avoir le principale but d'être omniprésent et de disparaître), aussi capable d'adapter le comportement d'un WSN dans le cas des variations de l'environnement contrôlé et des conditions de contrôle. Également, et en raison de leur déploiement dans des sites distants et hostiles, les dispositifs de WSN sont limités dans leurs ressources à bord. Tout d'abord, ils sont équipés des batteries qui une fois épuisées, ne peuvent être remplacées facilement. Par conséquent, la réduction de consommation d'énergie est cruciale dans les solutions et les applications des WSNs. Comme dans le cas des MANETs, les évaluations expérimentales récentes ont prouvé que, avec des technologies courantes, la communication est la tâche qui consomme le plus d'énergie: pour prolonger la vie des WSNs, il est nécessaire de réduire soigneusement le taux de transmissions et la quantité de données émises.

Dans ce contexte, alléger la coordination des nœuds apparait comme une question importante. Par exemple, via le traitement collaborative dans le réseau pour permettre la fusion automatisée des données capturées et la transmission uniquement d'indicateurs concis et agrégés, et ce, seulement en cas de besoin en fonction des

objectifs de la détection. De plus, les limitations d'énergie et la taille de capteur réduite imposent de nouvelles contraintes, tel qu'une puissance CPU et une mémoire limitées.

Les solutions WSN peuvent s'appliquer à différents domaines. En effet, le contrôle de l'environnement offre une riche collection de scénarios d'exploration possibles: les domaines d'application bien reconnus incluent l'analyse chimique de l'air/l'eau/ du sol et le support/validation/retour de l'information (*feedback*) du développement des modèles complexes d'écosystème, Le déploiement des équipements de capteurs dissimulées dans le secteur militaire, tel que celui pour détecter des objets qui violent un périmètre fixe ou traquer les véhicules mobiles ou encore l'automatisation à l'intérieure des maisons (*smart houses*) des systèmes d'éclairage et de chauffage peuvent de manière significative tirer bénéfice du contrôle basé WSN. Plus encore, envisager des applications critiques, par exemple, surveillance de la santé d'un patient, posent des contraintes additionnelles concernant les propriétés WSN, telle que la fiabilité et la sécurité, suggérant, de ce fait, encore d'autres activités continues de recherche et d'investigation.

La recherche dans les WSNs a abordé des questions au niveau de chaque couche de protocole: nous passons en revue rapidement plusieurs résultats intéressants. Les protocoles WSNs pour la couche MAC visent à créer des topologies de couche de liaison parmi les nœuds et à supporter l'accès au canal de communication. Le protocole de SMACS [SGA00] exerce en même temps la découverte et la création de topologie de couche liaison, et l'organisation d'accès au canal. Il est essentiellement basé sur l'algorithme TDMA, et exploite un system distribué et local pour programmer l'accès des nœuds au canal. Les solutions de contrôle de topologie visent à identifier les nœuds redondants qui peuvent être mis hors tension sans affecter la capacité du système à répondre aux exigences de l'utilisateur, par exemple, alternativement des nœuds voisins supportant les mêmes chemins de routage [STG02]. De la même façon que dans le cas des MANETs, la couche de routage a canalisé la plupart des efforts de recherche sur la conception de protocoles de dissémination pour le contrôle des informations captées.

3.1. Approches pour dissémination dans les WSN :

On peut, en gros, classer les approches, en ce qui concerne le type de tâche qu'ils supportent en proactif, réactif et l'hybride.

3.1.1. *Les protocoles proactifs:*

Les protocoles proactifs visent périodiquement à retourner les valeurs d'attribut captés, et à mettre les plateformes hors service le temps restant afin d'économiser l'énergie. En ajustant correctement le nombre d'intervalles de rapport, il est possible de négocier entre une solution rapide (des intervalles de rapport courts) et une plus conservatrice (des intervalles de rapport longs). La première a l'avantage de transmettre les données critique avec de faibles délais. Malheureusement, elle s'avère inefficace pour le problème de l'énergie parce qu'elle rapporte probablement aussi des données non pertinentes. La dernière solution conserve l'énergie, mais exige de grands délais de livraison pour les données critiques [HCB00, LRS02].

3.1.2. *Les protocoles réactifs:*

Les protocoles réactifs fournissent des données seulement si un attribut capté dépasse un seuil de valeur indiqué par l'utilisateur. D'une part, cela améliore la latence de livraison; D'autre part, si le seuil n'est jamais atteint, l'utilisateur ne peut déterminer si le réseau est toujours actif ou si tous les nœuds sont morts [MAA01].

3.1.3. *Des protocoles hybrides:*

Des protocoles hybrides représentent une approche bien plus équilibrée. En fait, les utilisateurs peuvent complètement contrôler le comportement de réseaux en accordant un intervalle de rapport et une valeur-seuil: le réseau peut imiter un *proactif* en augmentant le taux de rapport; et peut imiter un *réactif* en diminuant le même paramètre [MAA02, IGE00].

Les réseaux proactifs (connue pour leurs lourdes surcharges réseau) peuvent, en adoptant cette solution, c'est-à-dire, de courts intervalles de rapport, diminuer considérablement la charge du système. En même temps, le réseau est rapide pour

réagir aux changements environnementaux les plus importants car les nœuds transmettent de nouvelles données toutes les fois qu'un attribut dépasse son seuil.

Une approche hybride méritant d'être mentionné est la diffusion dirigée [IGE00] (*Directed Diffusion*), visant à établir des chemins entre les points d'émission et les Sinks en exploitant seulement des interactions localisées. Dans la diffusion dirigée, les intérêts (en précisant les valeurs seuil de données ainsi que les délais de rapport périodiques) sont diffusés sur tout le réseau, établissant des itinéraires de chemin de retour (*backpath*) sur des nœuds intermédiaires appelés des gradients. Ceux-ci sont exploités pour rendre des données appropriées au créateur d'intérêt.

Au niveau de la couche d'application, de nouveaux scénarios de WSN ouvrent de toutes nouvelles perspectives et des défis techniques; les secteurs d'application les plus examinés touchent au contrôle environnemental, par exemple, le projet ARGO pour la surveillance de la continuelle augmentation du niveau de mer [ARG09].

4. Les réseaux Ad Hoc véhiculaires (Vehicular Ad Hoc Network)

Le déploiement des nœuds MANET sur des véhicules représente un nouveau scénario pertinent et stimulant. En effets beaucoup de constructeurs automobiles planifient d'installer des équipements de connexion sans fil dans leurs véhicules pour permettre des communications opportunistes entre les véhicules. Une caractéristique distincte est que les véhicules sont fortement mobiles, avec une vitesse allant jusqu'à 30m/s, bien que leurs modèles de mobilité soient plus prévisibles que ceux des nœuds MANET en raison des contraintes imposées par la route, des limitations de vitesse et des habituels trajets quotidiens. C'est pourquoi, ces réseaux exigent des solutions spécifiques et identifient un nouveau secteur de recherche dans le domaine des MANET, c'est-à-dire, des Réseaux Adhoc Automobiles (VANET).

La recherche récente prévoit un grand nombre d'applications spécifiquement conçues pour les VANETS tels que:

(i) Assurer une conduite coopérative sûre où des informations de secours sont répandues aux véhicules proches et des systèmes de réponse en temps réel qui manœuvrent automatiquement pour éviter des accidents [XMS04].

(ii) Permettre le partage de fichiers [NDP05], la diffusion de publicités commerciales [NTD06] et le commerce (marketing) P2P [LPA06].

(iii) La collecte de données distribuées, par exemple, avec l'objectif de fournir aux conducteurs des informations sur des parkings disponibles [CGM06] ou des embouteillages sur route [STC06].

Dans ce cadre des solutions de routage et de communication ont été proposées et évaluées pour les VANETs: unicast [NBG06, SAJ04], broadcast [TJH04, XMS04, KEO04] et carry-and-forward [CKV01, NBG06, BGJ06, FGH04, ZHC06].

4.1.1. Les stratégies Unicast: sont d'habitude applicables seulement quand il est possible d'assurer la disponibilité et la validité d'un chemin ininterrompu entre la source et la destination.

4.1.2. Les stratégies broadcast: augmentent la fiabilité de livraison des messages dans des environnements de déploiement plus généraux.

4.1.3. La stratégie carry-and-forward: permet d'avoir la fiabilité de livraison semblable à celle obtenue dans les *broadcast* avec une légère tolérance au retard dans des scénarios d'application, même avec une faible densité de nœud.

En raison des difficultés intrinsèques au déploiement en grande taille des VANETs, la plupart des protocoles ont été validés et évalués via des simulations, qui exploitent des nouveaux modèles de mobilité basés soit sur des cartes réelles, soit sur des comportements réalistes des véhicules afin de se rapprocher le plus possible des conditions d'environnements ciblés [SAJ04, TJH04].

En particulier, les efforts de recherche pour identifier les protocoles de routage unicast appropriés pour VANET qui proviennent de l'adaptation des protocoles traditionnels de MANET, qui prennent souvent en considération les solutions « à la

demande » (on demand) et les solutions basées données géographiques (geographic-based solutions) compte tenu de la forte mobilité des nœuds VANET. Beaucoup d'auteurs évaluent AODV [PBR99] et GPSR [KAK00] avec des modèles de micro-mobilité de trafic précis. Constatant que les deux protocoles souffrent de faibles proportions de livraison de paquet une fois appliqués au VANETs, des systèmes ont été conçus basés sur l'identification des meilleurs expéditeurs, c'est-à-dire, des nœuds placés pas trop loin de l'émetteur (conduisant à une plus grande probabilité en cas de rupture lien) ni trop proche (augmentant le nombre de sauts de routage).

Plusieurs applications de VANET, par exemple, liées à la sûreté ou au trafic/aux publicités commerciales, font appel à la livraison des messages à tous les nœuds situés près de l'émetteur, avec un taux de livraison élevé, et un court délai [KEO04]. Nous pouvons évaluer l'efficacité des solutions de broadcast basées sur la priorité, par exemple, celles incluses dans les spécifications 802.11e [DRA03], portent sur l'augmentation de la probabilité de réception de messages importants. Dans des conditions de haut trafic, même la priorité n'est pas décisive pour garantir une bonne émission, puisqu'on n'est pas capable de contrecarrer le problème du terminal caché [UMA11].

Des études récentes se sont attelées sur ce point en proposant de nouvelles stratégies de diffusion. Exemple dans [XMS04] on propose des solutions simples qui identifient les éléments utiles pour l'élaboration d'un protocole d'émission fiable, comme la détection d'une porteuse, des systèmes a répétition fixes, c'est-à-dire, où les nœuds répètent des messages pour un nombre de fois fixe, et dans des créneaux horaires en synchronisation avec l'horloge globale.

Cependant, l'émission en *single-hop* ne fournit pas un support complet aux applications (exemple: advertising applications); ainsi, des solutions de dissémination en *multi-hop* efficaces sont examinées dans [KEO04]. En plus des questions communes affectant l'émission en *single-hop*, la dissémination en *multi-hop* souffre aussi de la problématique: tempête de diffusion [1][NTC99]. [KAK00] Propose l'émission *Urban*

[1]Le broadcast storm (ou tempête de diffusion) est une panne réseau lors de laquelle les messages (trames) transmis en diffusion (souvent des messages d'information, de control ou d'erreur) donnent lieu à une réponse des hôtes sur le même domaine de diffusion. Les trames n'ayant pas de durée de vie, elles peuvent tourner indéfiniment. Cette situation est souvent provoquée par une redondance des liens et une absence .Dans certains cas, cette panne provoque une situation de déni de service (DoS).

Multi-hop Broadcast (UMB), où le récepteur (destinataire) le plus éloigné est le seul nœud responsable de la reconnaissance et la retransmission du message diffusé.

Les questions sur la livraison de paquet dans des zones à faible fréquentation véhiculaires ont encouragé les récentes contributions de recherche à examiner les stratégies *carry-and-forward*. Dans [CHV01], les auteurs simulent un scénario sur autoroute pour comparer deux stratégies idéales: **pessimiste** (c'est-à-dire, synchrone), où les sources envoient des paquets aux destinations dès qu'un chemin d'accès multi-hop est disponible, et **optimiste** (c'est-à-dire, *carry-and-forward*), où des nœuds intermédiaires tiennent des paquets jusqu'à ce qu'un voisin tout proche de la destination soit détecté. Conformément aux suppositions implicites i) de tampons messages et largeur de bande illimités et ii) de modèles de mobilité facilement prévisibles quant aux véhicules sur une autoroute, ce dernier a œuvré pour réaliser un plus court délai de livraison. Cependant, dans des situations plus réalistes, les protocoles *carry-and-forward* doivent être soigneusement ajustés.

4.1. Protocoles de dissemination pour les VANETs:

Dans la littérature, plusieurs protocoles de disséminations ont déjà vue le jour, dans ce qui suit nous allons citer les plus connus :

4.2.1. MaxProp: la partie du projet l'UMASS *Diesel Net* [UMA11], est une stratégie de classement afin de déterminer l'ordre de livraison de paquets lorsque des rencontres de nœud surviennent occasionnellement. Ainsi, la priorité baisse dans le cas où les tampons sont pleins. La priorité est donnée aux paquets destinés à l'autre partie, puis aux informations de routage, aux accusés de réception, aux paquets avec petit nombre de saut(*hop-count*), et enfin aux paquets avec une haute probabilité d'être livrés par l'intermédiaire de l'autre partie [BGJ06].

4.2.2. VADD: prend en considération que la plupart des rencontres de nœud auront lieu dans des espaces d'intersection. Dans [ZHC06] des stratégies efficaces de décision sont proposées, réduisant fortement défaillance et délais de la livraison de paquets. Les applications de collecte de données distribuées dans VANET font appelle aux stratégies de dissémination géographique qui livrent des paquets à tous les nœuds

appartenant à la zone distante de la cible, malgré des chemins probablement interrompus [FGH04, STC06].

4.2.3. MDDV [FGH04] exploite la dissémination géographique vers la région destinée. Les messages sont portés par des véhicules en tête, c'est-à-dire, les mieux placés vers la destination par rapport à leurs voisins. Au lieu de cela, [STC06] propose plusieurs stratégies basées sur des champs potentiels virtuels produits par des fonctions de propagation: n'importe quel nœud estime sa position dans le champ et retransmet des paquets jusqu'à ce que les nœuds placés dans des endroits à plus faibles valeurs de potentielles soient trouvés. Cette procédure est répétée jusqu'à ce que les zones cible minimales soient détectées.

5. Les réseaux de capteurs véhiculaires (Vehicular Sensor Networks):

Les réseaux de Capteur véhiculaires (Automobiles) (VSNs) peuvent être construits sur des VANETs en équipant des véhicules avec des dispositifs dotés de capteurs comme indiqué dans la figure 1.4. Les VSNs apparaissent comme un nouveau paradigme de réseau pour contrôler le monde physique efficacement. En particulier, dans les zones urbaines où une forte population de véhicules (travaillant comme des capteurs mobiles) devrait toujours être présente [LMZ06]. Les véhicules ne sont pas limitées par de strictes contraintes d'énergie et peuvent être facilement équipées de puissantes unités de traitement, d'émetteurs sans fil, dispositifs de détection et même ceux d'une certaine complexité, coût et poids (détecteurs de déversement de produits chimiques, appareils photo/vidéo.

Notons que les VSNs représentent un scenario de déploiement stimulant, nouveau et intéressant, qui est différent des environnements traditionnels de réseau de capteurs sans fil, ce qui exige alors des solutions innovatrices spécifiques. En fait, différemment des nœuds de capteurs sans fil, les véhicules montrent habituellement des modèles de mobilité contraints dus aux formes des rues, aux carrefours, et aux limitations de vitesse. En outre, ils n'ont habituellement aucune limitation stricte sur la capacité de traitement et les capacités de stockage mémoire. Le plus important, est qu'ils peuvent héberger des capteurs qui peuvent produire des quantités de données

importantes (comme les flux multimédia enregistrés par des caméras). Ceci implique que les solutions de reporting de données (enregistrement, signalement) qui sont déjà connues dans la littérature des WSNs seront inapplicables.

L'échelle typique d'un VSN sur de vastes zones géographiques (par exemple, des milliers de nœuds), le volume de données produites (par exemple, flux de données 'Stream'), et la mobilité des véhicules rendent les solutions de réseau traditionnelles WSNs irréalisable à adopter où les données capturées tendent à être systématiquement livrées aux nœuds puits (*sinks)* moyennant les protocoles data-centric [MAA02].

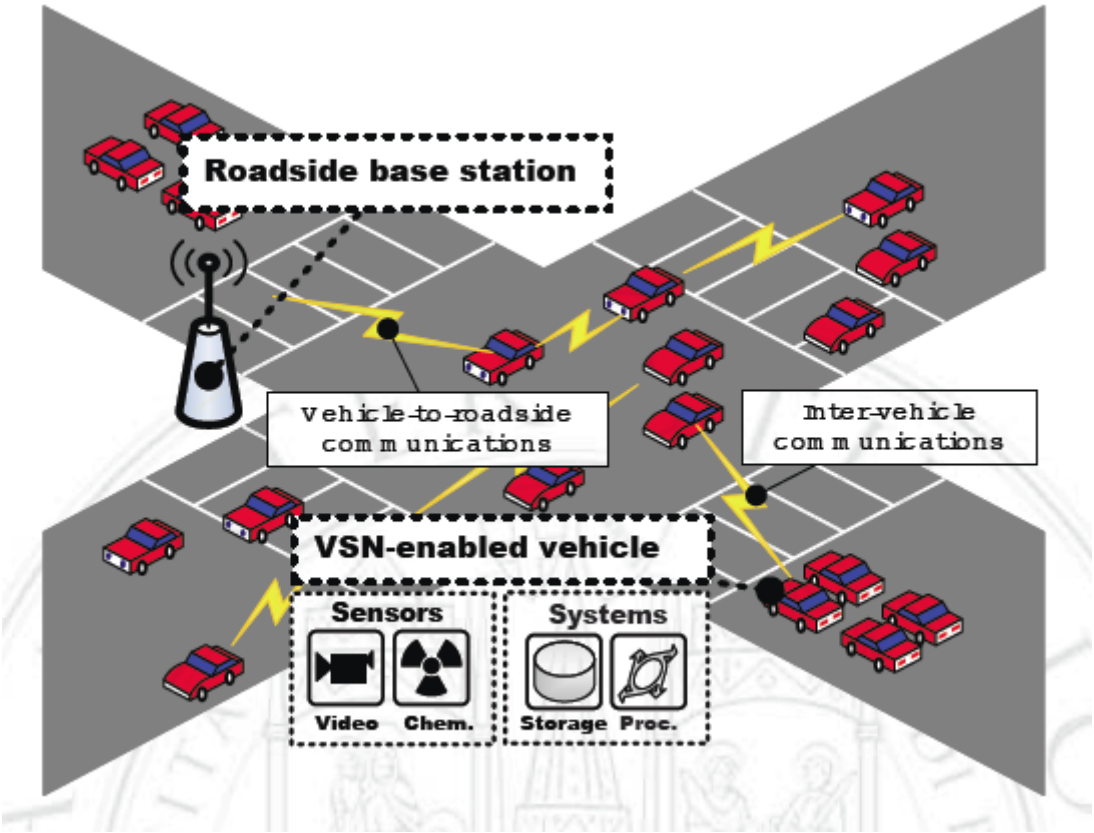

Figure 1.4: Réseaux de capteurs véhiculaires VSN [LMZ06].

Etant considéré comme un nouveau model des réseaux VANETs, les VSNs ont souvent pour buts: la collecte et la diffusion de l'information en des délais optimisée ou même en temps réel. Dans [GUI04], l'auteur a utilisé un VSN pour mieux comprendre la formation des embouteillages. Ils ont souligné le fait que les réseaux de capteurs véhiculaires est l'une des solutions la moins coûteuse qui tend à réduire les embouteillages, les émissions de CO_2 et la consommation de carburant. Dans [LMG06],

les auteurs ont proposé d'utiliser les VSN pour des raisons de sécurité où les nœuds agent peuvent chercher une voiture volée, par exemple, en envoyant une requête à tous les nœuds qui ont croisé ce véhicule. Une autre application de VSN est celle proposée dans [LMZ06] où le réseau fournit aux usagers de la route une conduite plus sûre en envoyant des messages d'alerte en cas d'urgence.

5.1. Caracteristiques des VSNs:

Ce type de réseau est considéré comme une application dédiée et spécifique des réseaux mobiles [C2C10]. Cependant, les travaux de recherche étudiés et réalisés dans les domaines des MANETs ne peuvent être directement appliqués dans le contexte des réseaux de capteurs véhiculaires à cause de leurs spécificités. Nous présentons quelques une des propriétés:

- Pas de contrainte énergétique.
- Pas de limitation mémoire.
- Dotée de dispositif de détection (caméras, capteur de produits chimiques…)
- Unités de traitement puissantes.
- Bande passante élevée.
- Plus longue portée pour la dissémination des données
- Etendu sur une vaste zone géographique.
- Forte vélocité.

5.2. Modes de circulation ou modèles de mobilité :

Les réseaux de capteurs véhiculaires sont un paradigme spécial des réseaux de capteurs mobiles, en effet dans les VSNs (comme pour les VANETs) la mobilité est contrôlée, nous citons ici les différents cas de circulations qui existent :

- Libre circulation: chaque voiture se rapproche de sa propre vitesse désirée et voitures n'interagissent pas beaucoup les unes avec les autres [NEJ08].
- Circulation synchronisée: voitures se déplacent avec à peu près la même vitesse dans toutes les voies de l'autoroute.
- Les embouteillages: la vitesse est faible et peut fluctuer beaucoup alors que le débit reste relativement élevé et ne varie pas de façon significative.

- Bouchon: les véhicules presque ne bouge pas et le débit est très faible.

5.3. Les domaines d'applications:

Les réseaux de capteurs véhiculaires sont des réseaux hétérogènes, qui peuvent être considérés comme l'union entre un réseau de capteur simple WSNs et un réseau ad-hoc véhiculaires VANETs avec ou non une infrastructure centralisées. Les capteurs, peuvent être embarqués sur les véhicules comme ils peuvent être déployés sur la route, les véhicules eux ont pour charge de collecter les informations transmises par les capteurs, les traiter et lancer les actions appropriées pour répondre à des phénomènes physiques suivant le domaine d'application [NEK05].

- Prévention des collisions aux intersections (y compris les piétons, les cyclistes).
- Prévention des collisions entre les intersections et dans les zones rurales.
- Disponibilité de signalisation à l'intérieur voiture.
- Etat de la route et sa situation.
- Etat ou statu des feux de circulation.
- Informations sur les embouteillages.
- Informations Coopérative sur un ensemble de véhicules.
- Informations détaillé d'un accident pour les premiers intervenants.
- Avertissements des occasions de covoiturage.
- Approche d'ambulance et itinéraire.
- Informations Parking.
- Informations du compteur de stationnement et réservations.
- Publicité en bordure de route.
- Messagerie voiture à voiture ou de la voix.
- Détection conducteur ivre.
- Télématique – suivi de camions, les flottes de taxis, etc.
- Péages (les zones de péage urbain ou usage de la route sur la base de péage).
- Des jeux sur route (pour les passagers, bien sûr).
- Mise à jour de cartes GPS.
- Des informations de zone locale.

- Diagnostics de la population des véhicules.

5.4. Architecture des VSNs:

Comme nous le montrons dans la Figure 1.5, l'architecture des réseaux de capteurs véhiculaires (VSN), peut être basée sur les trois types d'architectures suivants [GRG08]:

5.1.1. Vehicule to Infrastructure V2I:

Considérant ici l'infrastructure aux liens sans fil (GSM, UMTS, WiMax, etc) sont utilisés pour collecter les données provenant des nœuds VSN. Les points d'infrastructures offrent des services grâce à des portails Internet communs comme: l'accès à internet, échange de données de 'voiture à domestique', communications de 'voiture à garage' pour le diagnostic distant, etc. Ce qui profite aux clients et peuvent motiver des conducteurs à investir dans l'équipement sans fil supplémentaire pour leurs véhicules.

5.1.2. Vehicule to Vehicule V2V:

Où la collecte et la restitution de l'information sont réalisées au sein du réseau véhiculaire. Cette solution peut être utilisée pour la diffusion rapide des messages d'alerte par exemple: Freinage d'urgence, collision, ralentissement, etc. Ou pour la conduite coopérative. Dans ce cas le modèle de mobilité n'est pas aléatoire, mais prévisible (sous couche du réseau routier) avec une très forte mobilité. En effet, dans le cadre des applications de sécurité routière, les réseaux à infrastructure montrent leurs limites, surtout en termes de délais. Et il est clair qu'une communication ad hoc avec saut est plus performante qu'une communication passant par un réseau d'operateur.

5.1.3. Architecture Hybride:

Cette architecture combine à la fois l'architecture V2V et V2I, En effet, les portées des infrastructures étant limitées, l'utilisation de véhicules comme relais permet d'étendre cette distance. Dans un but économique et en limitant le nombre de bornes à chaque coin de rue, l'utilisation de communication multi sauts sur les véhicules prend toute son importance.

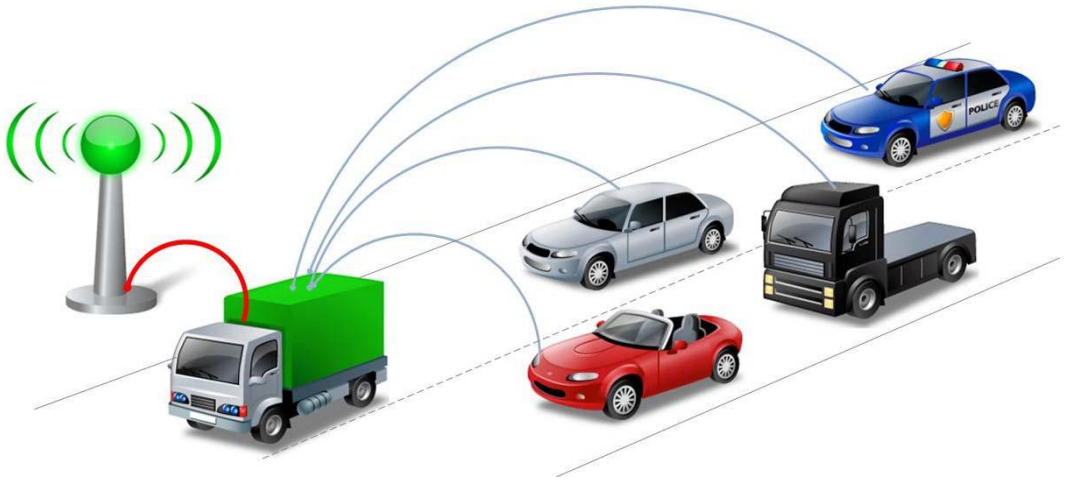

Figure 1.5: Communication dans les VSNs.

5.5. Classification des VSNs:

D'après la littérature actuelle sur les VSNs et suivant des implémentations spécifiques, nous pouvons classer les VSNs en deux groupes:

5.5.1. *VSN Intra-vehicule:*

Un seul véhicule VSN utilisé pour donner des diagnostiques au conducteur. Les intra-véhicule VSN sont introduits à des fins de suivi, de contrôle et la communication entre les composants et les sous-systèmes à l'intérieur d'un véhicule. Ils sont motivés par la complexité croissante, le poids, et le coût du câblage. Dans le véhicule, la connectivité intra-véhicule VSN est généralement régie par les normes sans fil pour les communications ad hoc telles que Bluetooth [BGG02], ZigBee [NEJ08] et UWB [GMK07].

5.5.2. *VSN Inter-vehicule*

Cette architecture est basée sur les différentes normes comme le DSRC (Dedicated Short Range Communication) de la couche liaison / physique des données [BIL07]. Ils peuvent soit être totalement distribuée ou centralisée. En communication V2V, un véhicule peut communiquer avec ses voisins véhicules même en l'absence d'une entité centrale (par exemple, une station de base). Le concept de cette communication directe est d'envoyer des messages de sécurité aux véhicules 'one-to-one' ou 'one-to-many', via une connexion sans fil. Ces messages sont généralement de petite taille et ils ont une durée de vie très courte dans laquelle ils doivent atteindre la destination. Le

système de communication inter-véhicule permet d'augmenter la sécurité, et de rendre la gestion du trafic plus efficace sur les routes [SUS07].

5.6. La norme 802.11 pour les communications inter-vehicules:

La norme IEEE 802.11 est un standard international décrivant les caractéristiques d'un réseau local sans fil (WLAN). La norme IEEE 802.11 est en réalité la norme initiale offrant des débits de 1 ou 2 Mbps. Des révisions ont été apportées à la norme originale afin d'optimiser le débit (c'est le cas des normes 802.11a, 802.11b, 802.11g et 802.11p, appelées normes 802.11physiques) ou bien préciser des éléments afin d'assurer une meilleure sécurité ou une meilleure interopérabilité .Par rapport au modèle OSI, l'IEEE 802.11 ne concerne qu'une partie de la couche de liaison de données et la couche physique et reste donc entièrement compatible avec les couches supérieures.

Nous nous intéressons pour la communication de véhicule à véhicule à la norme sans-fil nommée **DSRC** (Dedicated Short Range Communication)qui a été défini en 2003, comme un nouveau standard dédié aux communications inter -véhicules, appelé WAVE (Wireless Ability in Vehicular Environments) et aussi connu sous le nom de IEEE 802.11p [FOS2011]. Cette norme utilise le concept de multicanaux afin d'assurer les communications pour les applications de sécurité et les autres services du Transport Intelligent.

DSRC œuvre dans la bande de fréquence des 5.9GHz (Europe et Etats-Unis) ou 5.8GHz (Japon). Cette bande de fréquence est définie en Europe et aux Etats-Unis respectivement par l'ETSI (European Telecommunications Standards Institute) et le FCC (Federal Communication Commission). Elle est généralement segmentée en 7 canaux de 10 MHz chacun, l'ensemble des canaux se répartissant fonctionnellement en 1 canal de contrôle et 6 canaux de service. Le canal de contrôle est réservé à la transmission des messages de gestion du réseau et des messages de très haute priorité à l'instar des certains messages critiques liés à la sécurité routière. Les 6 autres canaux sont quant à eux dédiés à la transmission des données des différents services annoncés sur le canal de contrôle.

Plus largement, DSRC regroupe une série de standards et protocoles dédiés aux communications véhiculaires. Ainsi, DSRC se base sur une couche physique et une couche MAC définies dans le standard IEEE 802.11p (**WAVE**). Cette couche physique est dérivée de l'IEEE 802.11a. Elle est capable d'offrir un débit **entre 6 et 27 Mbps** (pour des distances jusqu'à 1000 mètres) avec une modulation de type OFDM (Orthogonal Frequency Division Multiplexing). De même, la couche MAC du 802.11p reprend le principe du CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) développé dans le protocole MAC de l'IEEE 802.11, avec un complément apportant la gestion de la qualité de service et le support du protocole de marquage de priorité. En effet, la couche MAC du WAVE est gérée en utilisant des priorités d'accès comme la norme IEEE 802.11e.

5.7. Challenges dans les VSNs

Nous allons détailler dans ce qui suit les différents challenges à relever pour les VSNs [BEH04]:

- La sécurité est un défi majeur ayant un grand impact sur le futur déploiement des réseaux véhiculaires ainsi que leurs applications. Dans ce contexte, le développent des mécanismes de sécurité instituant les relations de confiance entre les entités communicantes et garantissant le contrôle d'accès aux services de même que la sécurité des transferts de données, s'avère d'une importance capitale.
- L'auto-organisation est un sujet critique pour les réseaux de véhicules. Elle vise à tirer parti des propriétés des véhicules pour dégager une structure permettant l'aménagement du réseau. Cette structure se doit d'être suffisamment autonome et dynamique pour supporter tout changement local au niveau d'un ou plusieurs véhicules singuliers.
- L'accès au canal est un sujet récurrent car les réseaux de véhicules utilisent des communications radio. Par conséquent, il est important de concevoir des solutions MAC spécifiques aux réseaux de véhicules. Des protocoles qui permettent d'apporter de la qualité de service et de gérer les priorités en résolvant les problèmes d'interférences radio, des problèmes de multi-trajets des

ondes, les irrégularités électromagnétiques, de l'allocation de ressource distribuée dans une topologie dynamique, etc.

- Le routage et la dissémination sont aussi un challenge dans les VSNs puisque pour pouvoir communiquer entre eux, les véhicules doivent définir un protocole de routage. En effet, quand les terminaux ne sont pas à une portée de transmission radio directe, le routage unicast est exigé pour établir la communication entre deux véhicules ou entre un véhicule et un relais fixe. Chaque véhicule peut donc prendre le rôle d'un émetteur, récepteur ou routeur. La dissémination d'information quant à elle, consiste à acheminer une information d'une source vers une ou plusieurs destinations, en assurant un délai d'acheminement réduit, une grande fiabilité et une meilleure utilisation des ressources.
- L'utilisation multiple des données dans des contextes différents – exemple: Chercher un stationnement dans un parking et à long terme: quelles sont les chances de trouver une place? Courte portée, qui est la plus proche?
- Besoin d'approches de communication multiples - Infrastructure pour de longues distances de région à région de données - FM ou IR liaison pour les données globales - V2I autour des intersections - V2V pour les données entre véhicule et agent – réseaux cellulaire ou le WiMax pour le client-serveur ou point-à-point.

Le test de ces systèmes est un défi gigantesque car une simulation réaliste de la densité de véhicule et la surcharge du réseau est la clé mais le seul moyen de savoir ce qui va se passer avec l'adoption généralisée sachant que par exemple les USA ont investi dans cette industrie prêt de 4 milliards de dollars de la bande passante libre, mais nous n'avons aucune idée sur ce que qui va se passer lorsque les utilisateurs commenceront à se manifester en grand nombre[BEH04].

6. Conclusion:

Les réseaux de Capteur peuvent être intégrés sur des VANETs en équipant les véhicules avec des dispositifs dotés de capteurs, ce qui a donnée naissance au nouveau paradigme de réseau: les réseaux de capteurs véhiculaires. Les VSNs apparaissent dans le but d'assurer le contrôler du monde physique efficacement en zones urbaines, où nous avons une forte population de véhicules ; ces réseaux ne sont pas concernés par les contraintes énergétiques et limitations en capacités de stockage.

Le besoin des VSNs s'est fait sentir dans divers domaines d'applications, tels que des applications de sécurité routière ou encore des applications de confort. Dans toutes ces applications une problématique majeure a fait surface, à savoir, la dissémination d'information dans un milieu fortement mobile, et la réorganisation autonome d'un réseau décentralisée de véhicules.

Dans le chapitre suivant, nous allons présenter différentes solutions proposées dans la littérature pour la dissémination et l'accès à l'information dans les VSNs.

Chapitre II: La dissémination de données dans les VSNs

1. *Introduction:*

Se basant sur des tragédies telles que le 11 septembre, les attentats à la bombe à Londres et les attentats terroristes en Algérie, on peut prévoir que les foules intelligentes peuvent effectivement aider à soulager les pertes sur les routes. Par exemple, à Londres lors de l'attentat à la bombe, des agents de police étaient capables de suivre à la trace certains des suspects dans le passage souterrain en utilisant des caméras de surveillance en circuit fermé (closed-circuit TV cameras). Mais, ils avaient à des moments des difficultés de perception à cause des autobus à deux étages. Cet incident a convaincu la police britannique d'installer plus de caméras pour lire les plaques d'immatriculation et traquer les véhicules. Pourtant, dans un tel scénario, une approche foules intelligentes serait préférable. Car, avec une coopération opportuniste complètement distribuée, il serait très difficile pour les attaquants potentiels de désactiver la surveillance [DCT10].

La reconstitution d'un crime et, plus généralement, l'enquête postérieure à un événement potentiellement contrôlé par des capteurs mobiles et distribués, exige la collecte, le stockage et la récupération de quantités massives de données capturées. Il s'agit d'un écart important par rapport aux réseaux de capteurs classiques où les données sont généralement recueillies, examinées, et envoyées à un sink. Ces données sont alors transmises à des conditions prédéfinies, tels que des seuils d'alarme. Par exemple, il est impossible de délivrer toutes les données détectées par des capteurs vidéo sur les voitures au sink des autorités de police en raison de la taille du flux de données. De plus, généralement, les nœuds de capteurs ne peuvent pas déterminées à priori si leurs données seront d'une utilité quelconque pour de futures enquêtes. Cela implique, par la suite, le problème de la recherche dans un espace de stockage qui est massif, mobile et totalement décentralisé. Cette approche doit se baser une coopération complètement distribuée des nœuds mobiles.

Les propriétés des VSNs doivent donc lever des verrous technologiques importants tels le routage et la dissémination des données, l'auto-organisation et la sécurité. Dans ce mémoire, on s'intéresse à un de ces verrous technologiques à savoir la dissémination des informations dans ce type de réseaux. La dissémination de données concerne le transport d'information aux récepteurs prévus tout en répondant à certains

objectifs de conception que l'on considère incluent un court délai (retard), fiabilité élevée, courte occupation mémoire, et un overhead minimal. Les récepteurs prévus - véhicules agents- sont ceux qui spécifient l'événement physique surveillé (i.e. ; taux de pollution, présence de produit chimiques…). Les utilisateurs peuvent aussi définir des intérêts arbitraires [STC06].

Dans ce qui suit, nous allons présenter quelque uns des travaux de recherche qui ont été présentés dans le domaine des réseaux de capteurs véhiculaires et plus précisément pour la dissémination dans un environnement urbain.

2. La dissémination des données dans les VSNs:

Il existe différentes approches pour la dissémination dans les VSNs

2.1. Dissémination opportuniste:

En raison de la répartition inhérente des réseaux VSN, certains travaux, tel que [FGH04], recommande l'utilisation de la diffusion opportunistes de données. Dans ce cas, les messages sont stockés dans chaque nœud intermédiaire et transmis à chaque nœud rencontrés jusqu'à ce que la destination soit atteinte. Ainsi, le taux de livraison est amélioré. Cependant, ce type de mécanisme n'est pas adapté aux applications non tolérantes au retard.

2.2. Dissémination géographique:

Les chemins de bout en bout ne sont pas constamment présents dans un VSN, une diffusion géographique est utilisée dans [LMZ06] en envoyant le message au nœud le plus proche vers la destination jusqu'à ce qu'elle soit atteinte. Une autre façon de réaliser la diffusion géographique est donnée dans [FGH04], où les auteurs montrent comment utiliser le géo-casting (sélection géographique) pour délivrer des messages à plusieurs nœuds dans une zone géographique.

2.3. Dissémination Peer to Peer:

Dans une solution peer to peer (P2P), le nœud stocke les données sources dans sa mémoire de stockage et ne les transmet qu'à la suite d'une demande. Dans [LMZ06], une telle architecture est proposée pour des applications tolérantes au retard.

2.4. Dissémination basée cluster:

Pour un meilleur taux de livraison et afin de réduire le *broadcast storm problem,* un message doit être rediffusé par un minimum de nœuds intermédiaires vers la destination. Les nœuds sont alors organisés en un ensemble de clusters, dans lesquels le Cluster Head collecte les données de son cluster. Et, il les transmet alors à un autre cluster. Les solutions à base de clusters fournissent un délai réduit pour la propagation et un taux élevé de livraison. Dans [BOF07], les auteurs utilisent un algorithme de clustering distribué pour créer un backbone virtuel qui ne permet qu'à certains nœuds de diffuser des messages. Ainsi, la tempête de diffusion est réduite de manière significative.

3. Projets existant dans les VSNs:

Un large éventail de nouvelles applications de surveillance en milieu urbain est une preuve évidente de l'intérêt croissant dans le domaine des réseaux de capteurs véhiculaires:

3.1. IrisNet:

Un exemple d'application qui fournit un contrôle à grande échelle basé sur des PC câblés équipés de caméras et de microphones standards. Des véhicules particuliers (véhicules de l'autorité gouvernementale) peuvent recueillir et traiter des données brutes pour répondre de façon appropriée aux requêtes de certaines applications. Une de ces applications est celle qui permet de suivre, à la trace, les places disponibles pour le stationnement dans une région métropolitaine [GKK03].

3.2. Gunshot Detection System:

Un autre exemple est le Système de Détection de Coup de feu (Gunshot Detection System) (GDS) qui utilise un réseau sans fil de capteurs acoustiques, stratégiquement et statiquement déployés afin de déterminer les emplacements d'où pourrait provenir les tirs [LIB05]. Les capteurs GDS traitent localement les données acoustiques pour identifier le numéro du tir et le type d'arme à feu. Dans ce type de système, la connectivité est utilisée pour avertir des agents de police [GDS05].

3.3. SafeSpot:

Le projet européen SafeSpot [SAF11] a pour objectif principal l'amélioration de la sécurité routière. Des études ont montré que le conducteur est, à 90%, responsable des accidents, principalement pour des causes d'inattention ou des erreurs de jugement. Aussi, le projet SafeSpot propose de développer un ensemble d'assistance fournissant au conducteur une indication de sa marge de sécurité. Ceci est effectué suffisamment à l'avance à l'approche d'une difficulté.

3.4. CVIS:

Le projet CVIS (Coopérative. Véhiculé-Infrastructure Systems) [CVI11] vise à concevoir, mettre en œuvre et tester les nouvelles technologies nécessaires pour permettre aux véhicules de disséminer de l'information entre eux ainsi qu'avec les infrastructures routières situées à proximité. Il a pour ambition d'amorcer une révolution dans la mobilité des voyageurs et des marchandises en concevant de toutes nouvelles modalités d'interaction entre les conducteurs, leurs véhicules, les marchandises qu'ils transportent et les infrastructures routières. Ainsi, le projet CVIS vise à améliorer la sécurité et l'efficacité du transport routier et à réduire son impact sur l'environnement.

3.5. MobEyes:

Prend en charge la formation des foules intelligentes (smart mobs)[NET10] de véhicules équipés de capteurs. Comme il est généralement impossible de transmettre directement l'énorme quantité de données détectées à un sink centralisé, par exemple,

l'autorité de police, MobEyes propose que les données détectées rester avec les nœuds de contrôles mobiles. Dans le véhicule un traitement en local est exploité pour extraire les caractéristiques d'intérêt, par exemple, les plaques d'immatriculation à partir d'images de la circulation.

Les deux premières solutions brièvement décrits ci-dessus, s'appuient sur le déploiement de capteurs statiques et des infrastructures de communication et logiciel prédéterminées. Ces aspects les rendent difficilement évolutives et vulnérables aux attaques contrairement à la dernière solution qui opte pour une approche décentralisée, dans le chapitre suivant nous allons détailler MobEyes, car nous nous sommes inspirés de son protocole dans nos travaux.

4. Schéma de déploiement:

Sur le terrain, un réseau de capteurs véhiculaires peut se décliner en deux schémas de déploiement. Le premier est un modèle proposé dans le projet Cartel du MIT [CAR06]. Ce modèle considère des capteurs déployés sur des voitures. Ce qui donne naissance à un système mobile de capture d'informations. Ce système est conçu pour collecter, traiter, livrer, et même visualiser les données provenant de capteurs situés sur les unités mobiles (Figure2.1). Cet environnement peut fournir une interface de programmation simple orienté requêtes. Il peut permettre le traitement de grandes quantités de données hétérogènes provenant des capteurs. Il devra, cependant, gérer la connectivité réseau intermittente et variable.

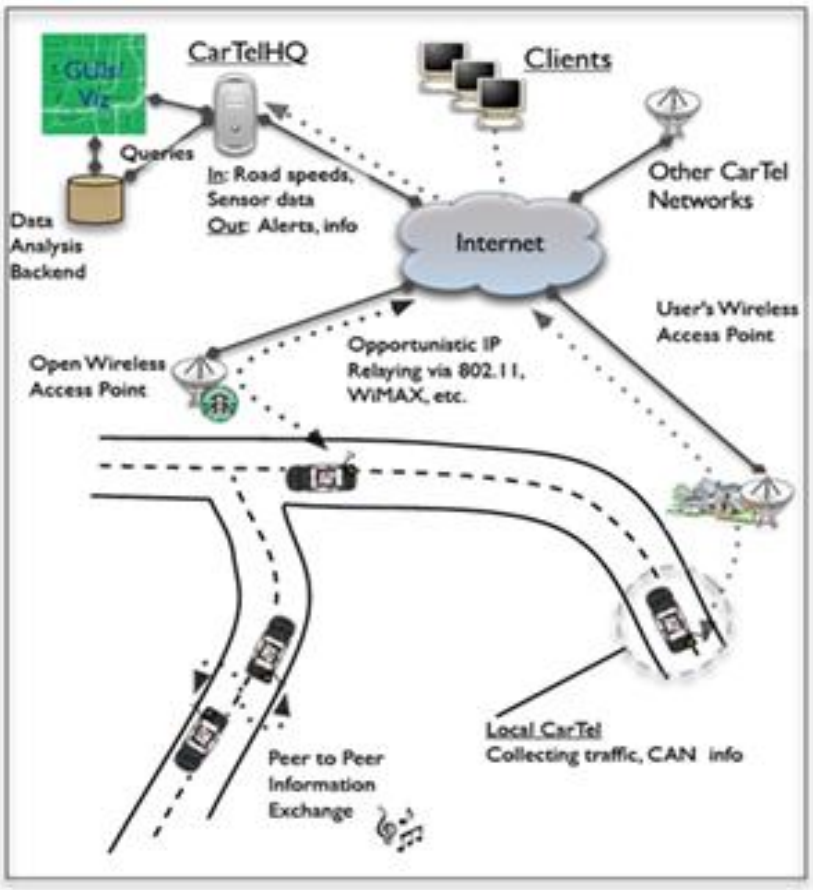

Figure 2.1: Schéma de déploiement du projet Cartel [CAR11]

Le deuxième schéma est proposé dans le projet MobEyes [NET10], comme un support efficace pour la surveillance urbaine proactive en exploitant la mobilité intrinsèque des véhicules. MobEyes offre un système d'observation mobile des évènements et peut collecter des résumés de données.

4.1. Schéma de déploiement Capteurs au bord des routes:

Le modèle capteurs au bord des routes ou RSS ''**Road side sensors"** est le modèle le plus simple et le plus utilisé. Quand nous parlons de VSNs, nous visualisons directement et plus facilement un ensemble de nœuds capteurs interconnectés.

Un capteur en bord de route est plus ou moins une petite unité de calcul capable de détecter des évènements, de traiter et de stocker des données. Ce type de capteurs peut contrôler non seulement la fluidité du trafic, mais ils peuvent agir en tant que points d'accès aux données. Ils peuvent obtenir et restaurer les données à partir de

n'importe quelle voiture dans leur champ de communication. Ils peuvent aussi fournir des services aux voitures.

Par exemple, il est plus probable pour les voitures d'obtenir des informations sur l'état des routes à partir des capteurs en bord de route. La portée de monitoring virtuel de ces capteurs est complétée par les voitures mobiles qui parcourent toute la ville. Un capteur de bord de la route comporte quatre composants comme l'illustre la figure 2.2:

• Noyau: Il est responsable du traitement des informations brutes capturées ou des nouveaux paquets reçus issues des composants de communication.

• Base de données: elle sauvegarde à la fois: les données brutes capturées, périodiquement, mais aussi les résumés produit après le traitement fais en locale et aussi les résumés collectés de façon épidémique.

• Carte et position: Il fournit la carte de la région. Il peut faire une correspondance entre la carte et les coordonnées géographiques.

• Communication: Il est utilisé pour communiquer avec d'autres voitures mobiles. Norme IEEE 802.11 est le standard de communication utilisé dans les réseaux véhiculaires. [JDE08]

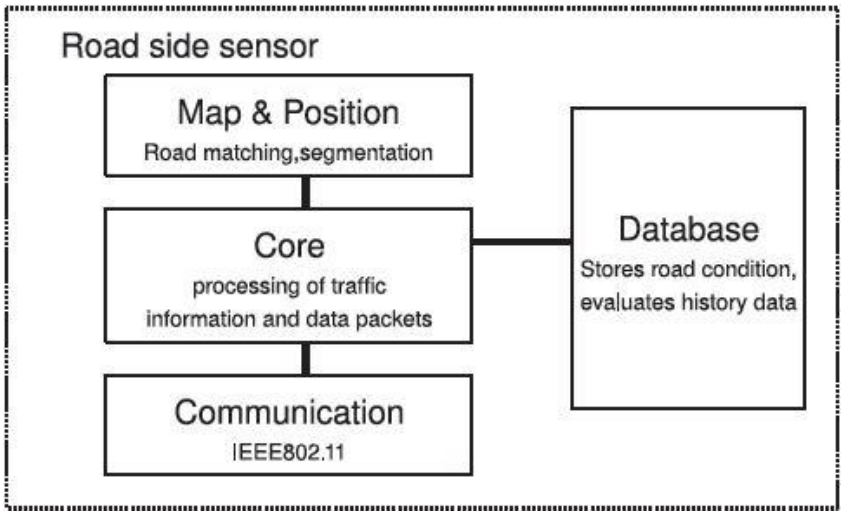

Figure 2.2: Diagramme en block d'un capteur en bord de route(RSS)

4.3. Schéma de déploiement Capteurs sur les véhicules:

Dans ce modèle, les capteurs mobiles sont situés sur des véhicules **''Mobile Vehicular Sensor''** (MVS) en mouvement dotés des dispositifs GPS. Ces capteurs peuvent surveiller la vitesse de la voiture et donner, en même temps, la position GPS exacte du véhicule. Ils peuvent aussi acquérir l'état des routes à travers le déplacement des unités mobiles.

Le conducteur doit indiquer sa destination ainsi le capteur sera plus attentif à l'état de la route de destination, c'est à dire lorsque l'échange de données se produit, les données relatives à la destination ou le long de la route vers la destination aura une priorité plus élevée à transmettre. Les capteurs mobiles sont capables de communiquer avec les deux capteurs latéraux statiques de la route et capteurs mobiles. Pour les capteurs statiques, ils peuvent télécharger leurs propres donnés au niveau du point d'accès et obtenir les informations dont ils ont besoin, pour les capteurs mobiles, ils peuvent obtenir des données à partir des autres voitures à proximité d'eux. La figure 2.3 montre les composants de capteurs mobiles.

- le GPS: qui fournit l'information de position précise.
- le capteur: qui permet d'observer et de capter des évènements ou des situations sur la route.
- le composant de Communication: qui permet de communiquer avec, à la fois, d'autres capteurs mobiles et des capteurs statiques.
- le composant d'affichage: ce composant offre la visualisation de l'information disponible, la position du véhicule et la destination. Par exemple, l'information routière est réalisée et visualisée à partir de la base de données locale.

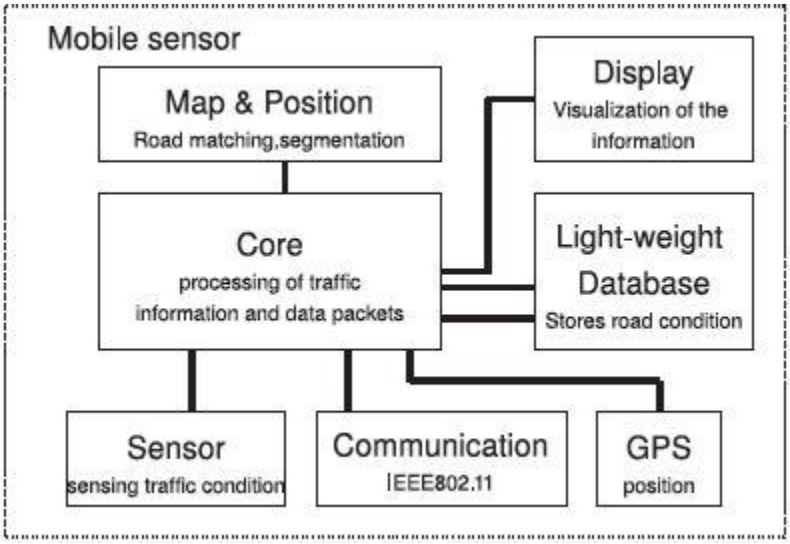

Figure 2.3: Diagram en block d'un capteur sur vehicule(MVS)

5. Conclusion

Durant notre recherche nous avons constaté qu'il existe très peu de travaux sur la dissémination de données dans les réseaux de capteurs véhiculaires malgré l'importance stratégique de ce domaine bien particulier.

MobEyes a retenu notre attention car il démontre la faisabilité des VSNs autonomes pour assurer la sureté dans les villes urbaines par une procédure proactive avec des protocoles opportunistes pour la diffusion/collecte de résumés.

Dans ce qui va suivre nous allons étudier en détails le protocole de Mobeyes, appelé MDHP, puis nous en inspirer et proposer une nouvelle technique de dissémination qui s'appuie sur la mise en place d'une forme collaborative de communication entre les nœuds de contrôles, ici ce sont les véhicules d'agents de polices. Cette technique nous permettra d'optimiser l'utilisation des ressources réseau.

Chapitre III: Protocole de Dissémination Collaboratif CDP

1. Introduction:

L'état de l'art réalisé nous a permis de constater que peu de travaux ont été dédiés aux réseaux VSNs, malgré l'importance croissante que prennent ces environnements. Parmi ces propositions nous citons le système MobEyes [NET10] qui a particulièrement accaparé notre attention par sa vue globale des problématiques sur les VSNs. MobEyes a été développé pour le contrôle urbain proactif. Il exploite la mobilité des véhicules équipés de caméras vidéo et une variété de capteurs afin de permettre la détection, le traitement et le filtrage des données capturées. Il propose aussi un routage ad hoc des messages d'un véhicule vers d'autres véhicules. Puisqu'il est impossible de diffuser directement aux autorités toutes les données capturées, MobEyes propose que: les données capturées restent sous le contrôle des nœuds mobiles (i.e., mémoire distribuée).

Cependant, dans la solution adoptée par MobEyes, un résumé collecté au préalable par un véhicule agent peut être re-collecté une seconde fois par un autre agent provoquant une redondance des résumés transmis et par la suite des paquets de transmission en double. Ces transmissions peuvent générer un important overhead dans le trafic du réseau VSN et mener à une congestion des communications. Elles peuvent aussi causer une perte de données due aux collisions de paquets.Notre objectif est de proposer une amélioration au protocole de collecte/dissémination de MobEyes. Notre proposition consiste à réduire l'overhead des communications, et d'offrir une solution décentralisée, collaborative et optimisée qui permettra aux véhicules agents de collaborer avec d'autres véhicules agents, en répondant aux requêtes émises lors des rencontres opportunistes via des connexions ad hoc, elle permet aussi de lancer une collecte de données à la demande d'une autorité.

Dans ce qui suit, nous allons voir les avantages et les inconvénients du protocole MDHP ("MobEyes Diffusion/ Harvesting Processor ") [NET10]. Nous essayerons de nous en inspirer afin de proposer des ou une méthode pour optimiser la diffusion/collecte de résumés dans le réseau. Dans ce chapitre, nous suggérons notre schéma de déploiement et nous établissons les hypothèses sur lesquelles nous nous sommes appuyés. Et, nous décrivons, par la suite, notre algorithme. Pour finir par une conclusion pour clore ce chapitre.

2. Principe de MobEyes:

Le protocole MobEyes pour la diffusion/collecte de données propose, en fait, l'échange de résumés de ces données. Pour cela, cet échange utilise la mobilité des véhicules et exploite les communications à un saut (single–hop). Mobeyes est appliqué pour des applications telles que la mesurer du degré de pollution, ou la collecte d'informations sur le trafic comme les conditions routières et les embouteillages.

MobEyes exploite la mobilité des véhicules pour un acheminement à moindre coup des résumés. Un scenario d'application possible: la collecte de données capturées par des véhicules à proximité d'un véhicule de criminels en fuite après avoir répandu des produits chimiques toxiques.

Si les véhicules sont équipés de caméras et/ou de capteurs de détection chimiques, ils produisent des résumés périodiquement ou alors suite à la détection d'un seuil déclencheur. Un résumé est constitué d'un ensemble de MetaDatas décrivant les principaux évènements observés. Ces résumés sont disséminés d'une façon opportuniste aux véhicules voisins. Par conséquent, des véhicules de contrôle, tels que les véhicules des agents de police, peuvent construire un index à posteriori des métadonnées distribuées pour la reconstruction de scènes suspectes passées.

Nous rappelons que Le système MobEyes a été développé suivant une architecture basée composants comme le montre la figure3.1. Le composant clé étant le MDHP - MobEyes Diffusion/ Harvesting Processor-.

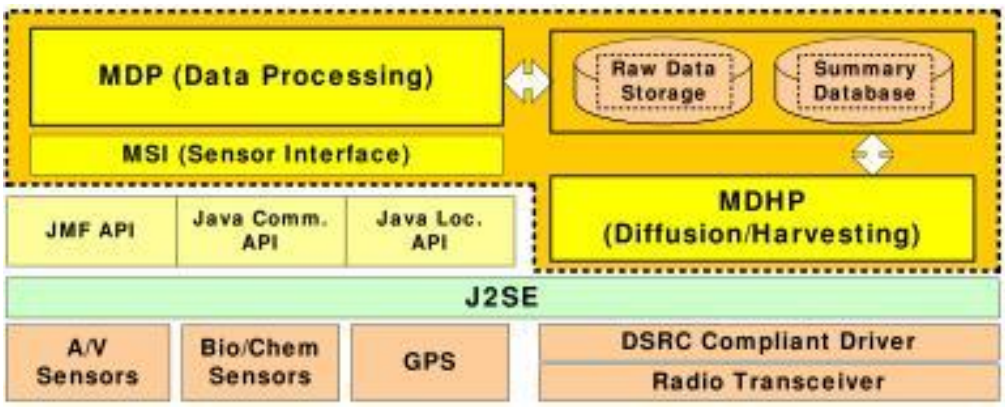

Figure 3.1: Architecture d'un nœud capteur de Mobeyes

MDHP œuvre à disséminer et collecter des résumés produits de façon opportuniste par le MDP ou -MobEyes Data Processor-. MDP a un accès aux données des capteurs via une interface uniforme MSI -MobEyes Sensor Interface-. Le MDP se charge de lire les données brutes capturées via le MSI. Il effectue les traitements et produit des résumés. Ces résumés comprennent des MetaDatas liées au contexte (localisation, horodatage,…) comme par exemple, le filtre de plaque d'immatriculation, ainsi que d'autres caractéristiques extraites par des filtres locaux.

Pour plus de portabilité, MSI permet au MDHP d'avoir accès aux données brutes indépendamment de l'implémentation des capteurs. Ainsi l'intégration de différents types de capteurs hétérogènes devient possible. Actuellement, MSI offre des méthodes pour accéder aux sorties streaming des caméras, au port série I/O stream et au données GPS. Pour l'interface avec les capteurs, MSI exploite des standards connus comme Java Media Framework (JMF), Java Communications et JSR179 location API [NET10].

2.1. Le Protocole MDHP

Le protocole original de diffusion des résumés consiste à faire en sorte que des véhicules privés (nœuds réguliers) propagent de façon opportuniste et autonome des résumés tout en se déplaçant vers les véhicules voisins. La version plus récente de ce protocole de collecte des résumés propose que des agents de police collectent ces résumés afin de construire de façon proactive par rapport aux événements sentis, un index partiel de la mémoire distribuée des nœuds du VSN (mobile VSN Storage).

2.1.1. *La diffusion des résumés*

Tout nœud ordinaire annonce ou publie périodiquement un paquet avec des résumés nouvellement crées, à ses voisins directs. Chaque paquet est identifié de manière unique (identification par un ID et un numéro de séquence local unique). Cette diffusion aux voisins directs offre plus de possibilités aux agents de collecter les résumés générés. La durée de la publication périodique des résumés doit être fixée correctement pour répondre aux exigences en matière de latence[2] voulue.

[2] La latence désigne le délai entre le moment où une information est envoyée et celui où elle est reçue. De façon plus générale, la latence peut aussi désigner l'intervalle entre la fin d'un événement et le début de la réaction à celui-ci.

Les voisins recevant un paquet, le stockent dans leur Bases de données locales de résumés. Selon la mobilité des nœuds et des rencontres entre véhicules, les paquets sont répandus dans le réseau d'une façon opportuniste (échange mutuelle des données).

MobEyes offre une diffusion des paquets vers les voisins à un saut. Un nœud diffuse périodiquement tous les paquets (émis et reçus) de sa base de données locale au détriment d'un *overhead*.

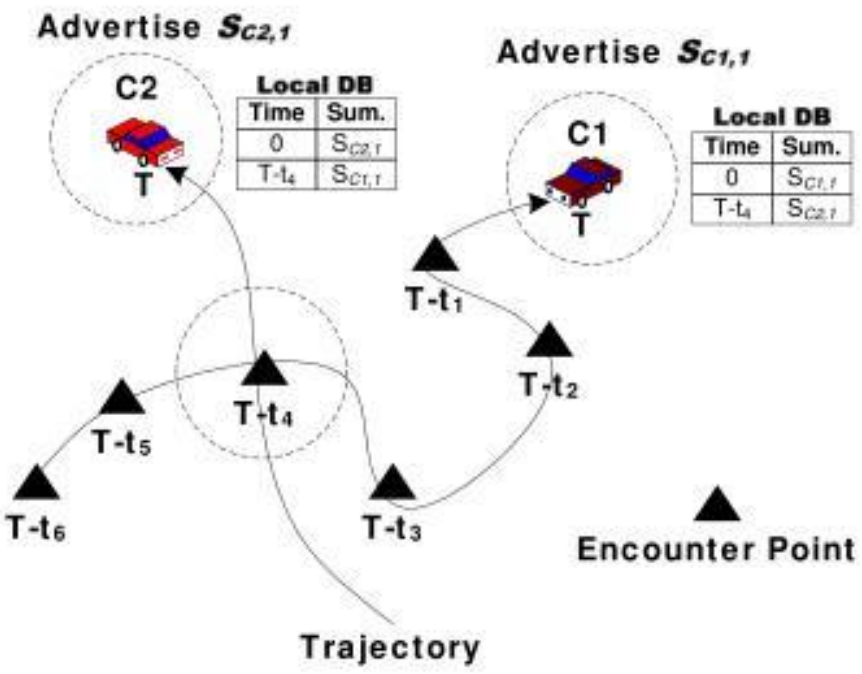

Figure 3.2: Diffusion passive en single-hop dans MobEyes

La figure 3.2 illustre le cas d'un nœud VSN C_1 qui rencontre d'autres nœuds VSN tout en se déplaçant. Pour des raisons de lisibilité, seuls C_2 est explicitement représenté. Les échanges ont lieu lorsque deux nœuds sont dans la portée l'un de l'autre, et lorsqu'ils ont un nouveau paquet de résumé à diffuser. Dans la figure 3.2, les cercles en pointillé et les triangles datés (timestamp) représentent respectivement la portée et les points de rencontres de C_1.En particulier, la figure 3.2 montre que C_1 (lors de la publication de $S_{C1,1}$) rencontre C_2 (qui publie $S_{C2,1}$) au temps T-t_4. En conséquence, après T-t_4, C_1 comprend $S_{C2,1}$ dans son espace de stockage, et C_2 comprend $S_{C1,1}$.

2.3. La collecte des résumés

La collecte des résumés s'effectue parallèlement à la diffusion des données locales. Un agent de police MobEyes peut demander aux nœuds réguliers voisins, par des requêtes proactives, la collecte de résumés dispersés. Le but final étant de rassembler tous les résumés produit dans un secteur donné.

Évidemment, un agent de police est intéressé par la collecte de résumés qu'il n'a pas recueillie jusqu'ici. Un agent MobEyes compare la liste des paquets déjà collectés à la liste des paquets de chaque voisin. Cela se fait par l'utilisation d'une structure de données destinée à optimiser l'espace afin de vérifier l'appartenance, i.e. un filtre de Bloom[3] [BMP06].

Un filtre de Bloom pour représenter un ensemble de n éléments est constitué de m bits, initialement mis à 0. Le filtre applique k fonctions de hachage indépendantes et aléatoires $h_1,...,h_k$, aux identifiants de paquets MobEyes et enregistre la présence de chaque élément dans leur k bits correspondants. Pour vérifier l'appartenance de l'élément x, il suffit de vérifier si toutes les $h_i(x)$ sont définies (mis à 1).

Par conséquent, la procédure de collecte comprend les étapes suivantes:

1) L'agent de police émet une demande « collecte» avec son filtre de Bloom,
2) Chaque voisin prépare une liste des paquets recherché à partir du filtre de Bloom reçus.
3) Un des voisins alors fournir à l'agent des paquets recherchés.
4) Dans ce cas, l'agent renvoie un accusé de réception avec une liste de relevé des paquets tout juste reçus. En détectant cela, les voisins mettent à jour leurs listes de paquets manquants pour l'agent.
5) Les étapes (3) et (4) sont répétées jusqu'à ce qu'il n'y ait plus de paquets recherchés.

Néanmoins, dans MobEyes, l'agent peut obtenir un paquet manquant avec une forte probabilité. Car avec le temps, il est fort probable que les autres nœuds aient récupéré les paquets, et que la procédure de collecte soit répétée à mesure que l'agent se déplace.

[3]Le filtre de Bloom conçu par Burton H. Bloom en 1970, est une structure de données probabiliste qui optimise l'espace utilisé. Cette structure est utilisée pour tester si un élément fait partie d'un ensemble.
Il permet notamment d'optimiser les flux entre pairs dans un réseau informatiques pairs à pairs (Gnutella).

3. CDP Collaborative Dissemination Protocol:

3.1. Problématique et contribution:

Le laboratoire qui a proposé MobEyes et son protocole MDHP est l'un des laboratoires pionniers dans le domaine des VSNs. Les chercheurs attachés à ce laboratoire ont réussi, dans un premier temps, à répondre aux besoins de ces réseaux, en assurant une latence convenable dans la collecte et une surcharge réseau acceptable. Néanmoins, il reste encore d'autres problématiques. Par exemple, le protocole MDHP ne prend pas en charge:

- La sécurité des paquets lors des échanges entre les véhicules,
- Le partitionnement dû à la forte mobilité du réseau et,
- Le risque de collision lors des échanges de données,
- Une collecte répétée de données déjà récupérées, ce qui constitue une source de trafic inutile, pouvant conduire à un overhead et une tempête de diffusion.

C'est en se basant essentiellement sur ce dernier point, que nous avons pensé à apporter une amélioration au protocole MDHP. En effet, l'idée consiste à éviter qu'il y'ait plusieurs collectes et des diffusions inutiles de données (ici, il s'agit de résumés). Ceci est possible en permettant l'échange de ces résumés entre les agents qui se retrouvent dans la même portée de transmission. Ainsi, un résumé qui a déjà été collecté par un véhicule agent, ne sera pas collecté une seconde fois par un autre véhicule agent. Les agents échangent leurs informations au fur et à mesure qu'ils se rencontrent. Ils bénéficient ainsi des collectes effectuées lors de leurs déplacements. Ce qui permet de mettre à jour leurs bases de données respectives et de préparer les réponses aux requêtes de manière plus efficace. Cette approche assure une meilleure gestion des ressources de notre réseau et une plus grande adaptation à la topologie dynamique considérée. Dans ce qui suit, nous allons détailler notre solution.

3.2. Modèle d'environnement:

Dans le chapitre précédent, nous avons décrit les deux modèles existants dans les réseaux VSNs .Il faut savoir que le modèle **RSS 'Road Side Sensor'** s'adapte bien au réseau peu dense puisque il permet d'assurer la surveillance et le contrôle d'une région

donnée même si celle-ci n'est pas très fréquentée par un ensemble de véhicules à un moment de la journée.

Cependant sa mise en place est bien plus couteuse et son déploiement est physiquement contraignant. C'est pour cela que le modèle **MVS 'Mobile Vehicule Sensor'** sera notre modèle de déploiement principal pour la communication et la collaboration entre les véhicules. Ce modèle est plus flexible et il présente un déploient moins coûteux et plus accessible que le premier. Toute fois, il ne sera pas exclu d'utiliser, dans le future, une infrastructure sur la route, puisque dans la conception de notre protocole, nous pouvons considérer le modèle RSS pour soutenir et améliorer l'efficacité du protocole.

Dans l'adoption du modèle MVS, nous nous sommes inspirés de la solution Mobeyes, et nous avons optés pour qu'il y ait deux types de véhicules:

3.2.1. *Véhicule privé:*

Ce sont tous les véhicules particuliers équipés de capteurs vidéo, de capteurs chimiques, etc. (figure 3.3). À leurs bords différents type de capteurs, ils seront responsables de captures des données physiques sur l'état de la route au fur et à mesure qu'ils se déplacent. En plus, ils devront diffuser l'information aux autres véhicules à chaque fois qu'ils se rencontrent de façon épidémique.

Les données brutes capturées sont traitées localement, puis encapsulés sous forme de résumé, nous verront les détails de ce processus plus loin.

3.2.2. *Véhicule agent:*

Ce sont tout les véhicules appartenant à la police, ils ont comme mission d'assurer la sûreté de la ville. Ils sont responsables de la collecte des données à partir des véhicules privés ou d'autre véhicules agents.

Ils envoient des requêtes de collecte, stockent les résumés à leurs niveaux, construisent un indexe distribué à postériori et peuvent exécuter un traitement spécifique à un besoin ou alors suite a une demande envoyer de la part d'une autorité (centrale de police).

3.3. Hypothèses

Dans l'optique d'assurer un control urbain proactive nous considérons dans cet environnement de réseau de capteurs véhiculaires les hypothèses suivantes:

- les nœuds mobiles sont composés de deux sortes de véhicules:
  - Véhicules ordinaires ou particuliers
  - Véhicules agents ou de police
- Les véhicules communiquent entre eux en utilisant une communication V2V et sans recours aux points d'infrastructure. Ce choix est dicté par la quête d'une plus grande scalability (passage à l'échelle) et d'un faible coût de déploiement.
- Tous les véhicules sont dotés d'un récepteur GPS afin de les localisés géographiquement.
- Une interface de communication régie par le standard 802.11p, assure la communication entre les véhicules. Ce standard appelé aussi norme WAVE (**Wireless Access in the Vehicular Environment**) offre un soutient au transport intelligent (ITS) en prenant en compte l'échange de données entre les véhicules à grande vitesse et entre les véhicules et l'infrastructure routière sur une fréquence de 5,9 GHz [JDE08].
- Notre protocole est désigné pour agir au niveau applicatif.
- Tous les véhicules sont équipés d'un ordinateur de bord permettant le traitement des données ainsi que leurs stockages en local.
- Chaque véhicule a un identifiant unique sur le réseau (V_ID). Cet identifiant pourrait être délivré par une autorité de certification.
- Nous synchronisons le réseau au départ afin de pouvoir inclure la périodicité au réseau.

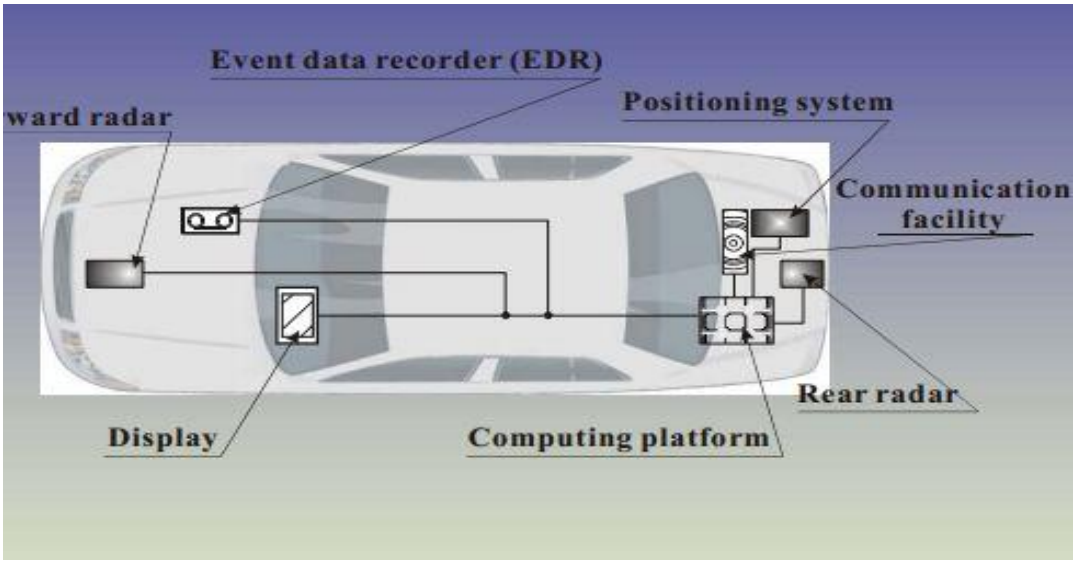

Figure 3.3: Equipement disponible sur les véhicules.

3.4. Principe de fonctionnement du protocole CDP:

Nous avons proposé un protocole qui s'exécute en trois (03) phases: la découverte, la communication et la fermeture, en permettant de diviser un réseau de capteurs véhiculaires en groupuscules (clusters) suivant les rencontres opportunistes. L'algorithme de CDP permet d'optimiser MDHP, il a pour but de minimiser la surcharge réseau avec un bon passage à l'échelle, ce qui permet d'éviter Le problème de la tempête de diffusion, il limite le risque de collision lors du transfère d'information, et il assure une auto-organisation du réseau de capteurs véhiculaires. Dans ce qui suit nous allons détailler les phases de CDP:

3.4.1. *Phase découverte:*

Cette phase débute périodiquement suivant la période transmise par une autorité ou encore suivant une période paramétrable, relative aux besoins d'une application, mais en pratique cette période de découverte est égale au temps au maximum probable qui s'écoule entre deux rencontres.

Lors d'une rencontre, un véhicule qui initie une collecte construit son dernier filtre de bloom suivant sa base de données locale et envoi un message découverte **MSG_DISCOVERY** contenant: son identifiant et son filtre en broadcaste vers tous les véhicules voisins dans sa portée à un seul saut.

Chaque véhicule qui est à l'écoute et qui reçoit cette information construit à son tour son filtre bloom suivant sa base de donnée locale, puis envoi une réponse au message de découverte vers tous ces véhicules voisins, y compris le véhicule initiateur en broadcaste. Voir la Figure 3.4

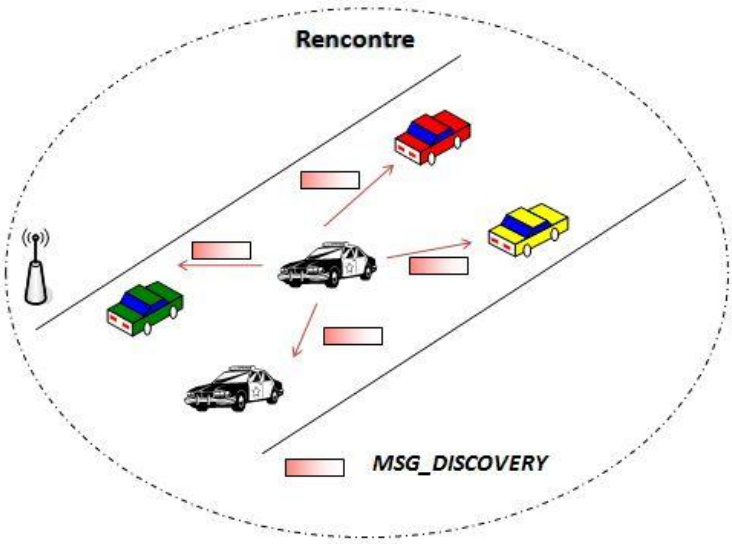

Figure 3.4: Diffusion du message de découverte

Le message **Msg_Discovery** aura la forme suivante:

ID_Ms	ID_Ap	V_ID	V_Bloom	V_Timestamp	V_Speed

ID_ Msg: Identificateur du Msg_Discovery.

ID_Apt: Identificateur de la rencontre actuelle.

V_ID: Identificateur du véhicule initiateur du Msg_Discovery.

V_Bloom: le filtre de bloom du véhicule initiateur du Msg_Discovery.

V_Timestamp: L'heure et la position à laquelle le Msg_Discovery a été envoyé, cette information est récupérée du GPS.

V_Speed: La vitesse à laquelle roule le véhicule initiateur lors de l'envoi du Msg_Discovery, cette information est récupérée grâce à un capteur de vitesse véhicule.

L'envoi de la réponse au message découverte, par un message **REP_DISCOVERY** se fait à des instants aléatoires et en différé. Grâce a une fonction d'attente, cette fonction utilise un ΔT, un lapse de temps de l'ordre de millisecondes, qui permettra une temporisation d'un temps prédéfini et nécessaire à la transmission d'une réponse, ceci

afin d'empêcher la collision entre les paquets. Par exemple le temps d'attente pour envoyer une réponse:

$$\mathbf{T = \Delta T \times Random(N)}$$

Où N est le nombre moyen des véhicules voisins, et la fonction **Random** génère un nombre aléatoire compris entre 1 et N (exemple: pour un ΔT = 1ms. Et un nombre max de véhicule N = 1000, le temps d'envoi d'un ***Rep_ Discovery*** est de: ΔT *Random(N),la découverte peut prendre au max 1000 ms=1s).

Ainsi nous évitons qu'il y ait des collisions lors de l'envoi de message de réponse à la découverte, et nous assurons la réception de tous les paquets émis dont nous aurons besoin pour la suite de notre protocole.

Chaque véhicule qui a reçu son premier message **REP_DISCOVERY** attend un temps nécessaire et paramétrable pour la réception de toutes les réponses des véhicules voisins (comme le montre la figure3.5) ce qui permet à la fin de la phase découverte, à chaque véhicule faisant partie d'une même rencontre de savoir quel véhicule détient telle ou telle information, pour le lancement de la deuxième phase de protocole.

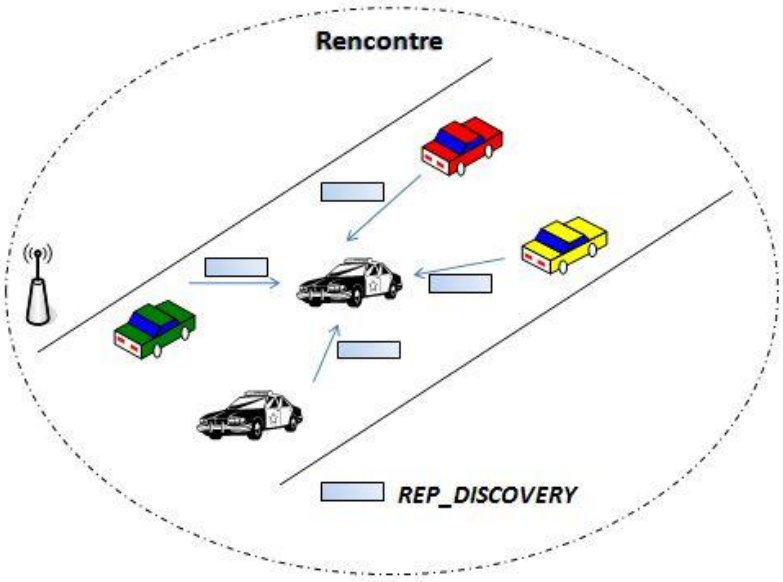

Figure 3.5: Réception de réponses.

Un véhicule ne peut faire partie que d'une seule rencontre à un instant donnée. Donc s'il reçoit un message de découverte d'un autre véhicule, ce message sera ignoré. Le véhicule sera à nouveau apte à participer à une nouvelle rencontre que si la rencontre dans laquelle il prend fin, nous le verrons plus en détails dans la phase fermeture.

Si un véhicule qui fait partie d'une rencontre ne reçoit plus de message durant un temps ***TTL*** (Time to Leave), nous considérerons qu'il aura quitté la rencontre et sera donc à nouveau disposé à faire partie d'une rencontre future.

Le message **REP_DISCOVERY** aura la forme suivante:

ID_Msg	**ID_Apt**	**V_ID**	**V_Bloom**	**V_Timestamp**	**V_Speed**

ID_ Msg: Identificateur du REP_Discovery.

ID_Apt: Identificateur de la rencontre actuelle.

V_ID: Identificateur du véhicule qui envoi le REP_Discovery, cet identificateur est récupéré grâce à une puce RFID.

V_Bloom: le filtre de bloom du véhicule qui envoi le REP_Discovery.

V_Timestamp: L'heure et la position à laquelle le REP_Discovery a été envoyé, cette information est récupérée du GPS.

V_Speed: La vitesse à laquelle roule le véhicule lors de l'envoi du REP_Discovery, cette information est récupérée grâce à un capteur de vitesse véhicule.

3.4.2. ***Phase Transmission:***

Nous considérons comme cluster l'ensemble des véhicules privés ou agents qui sont à l'écoute et dans la même portée que le véhicule initiateur. Comme le montre la figure 3.6, dans une même zone par exemple de 1 km^2, nous pouvons avoir un ensemble de réseaux indépendants les un des autres, exemple le cluster D, en noir c'est le véhicule agent qui communique avec les autres véhicules en gris et en blanc, chaque cluster représente une rencontre.

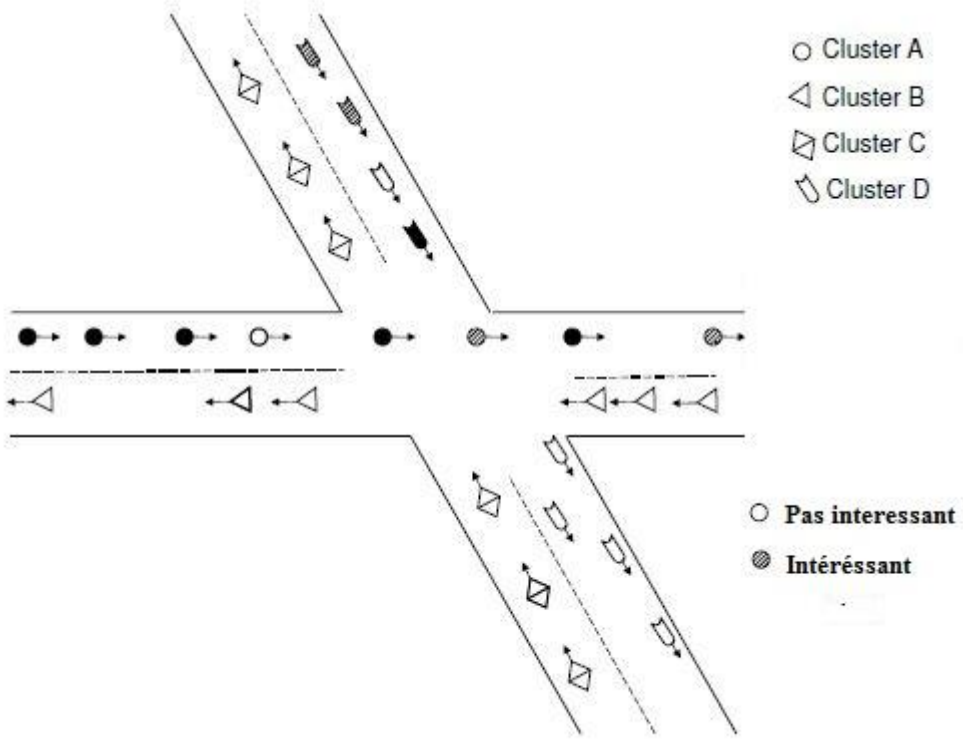

Figure 3.6: Organisation en cluster du réseau.

Après avoir reçus un **REP_DISCOVERY** de tous ces véhicules voisins directs, les identifiants des véhicules participants à la rencontre et leurs filtres de bloom sont connus au niveau de chacun de ces véhicules, chaque véhicule appartenant au même cluster, se sert de cette information et procède à son niveau à l'élection du clusterhead, c'est ainsi qu'à un moment donnée tout les véhicules connaissent l'identité du **VAC** (véhicule agent clusterhead)[4].

Nous définissons le clusterhead d'une même rencontre comme étant le véhicule agent ayant la base de données la plus complète, comparé aux autres véhicules voisins directs. Cela nous permet d'optimiser le temps de la collecte d'information et de réduire la communication (en nombre de message), puisque un véhicule agent (qui a la base de données la plus complète) aura moins de résumés à collecter pour combler les résumés

[4]**Remarque**: Nous considérons comme cluster l'ensemble des véhicules privés ou agents qui sont à l'écoute et dans la même portée que le véhicule initiateur.

manquants de sa BD. Ainsi **VAC** collectera l'information plus efficacement que ces voisins directs.

En optimisant le temps de collecte de résumés par le **VAC**, nous donnons plus de temps aux véhicules voisins de répondre et transmettre les résumés manquants à **VAC** avant que l'un des véhicules ne quitte la rencontre physiquement, c'est-à-dire n'étant plus dans la portée du **VAC**. A la fin du processus d'élection, seul le véhicule **VAC** aura la responsabilité d'envoyer une requête de collecte à ces voisins.

L'élection du VAC permet aussi de situer dans quel type de rencontre nous nous trouvons, en effet si dans notre groupe de véhicule il n'existe aucun véhicule agent ceci implique que nous sommes dans le cas d'une rencontre de véhicules privés, sinon nous sommes dans le cas d'une rencontre avec au moins un agent, dans ce qui suit nous allons détailler ces le fonctionnement de ces deux types de rencontres:

- **Rencontre Véhicules Privés:**

Dans ce type de rencontre**,** chaque véhicule privé qui participe à la rencontre prépare un ou plusieurs résumés à transmettre. Évidement on ne transmet pas tout les résumés existant dans la base de donnés locale du véhicule, mais uniquement les derniers qui ont été générés. De plus, le nombre **N** de résumés envoyés durant un rencontre est relatif au type d'application véhiculaire.

Le nombre de résumés **N** peut être paramétrable suivant la vitesse, le débit et le nombre de véhicules participants à une rencontre. Si par exemple notre véhicule se déplace à grande vitesse, un ou peu de résumés pourraient être transmis en raison des délais nécessaires aux envois. Comme il est alors possible de restreindre la taille des résumés en privilégiant les données prioritaires. Les résumés seraient alors de petites tailles.

La taille du résumé est importante car, dans un réseau à forte mobilité, il est impératif de pouvoir transmettre tout le résumé d'un véhicule à un autre avant qu'il ne quitte sa portée de communication. Le nombre de métadonnées constituants un résumé est aussi variable car il dépend du type d'application dans le réseau.

Connaissant **le débit et la portée de transmission** qu'offre la norme P, nous pouvons suivant la vitesse calculer la durée d'une rencontre. Puis suivant le nombre de

véhicules présents, nous pouvons adapter la valeur de **N**, puisque nous avons la taille maximale que peut avoir la donnée transmise (un ou plusieurs résumés).

La génération de résumés se fait périodiquement, c'est un processus qui s'exécute indépendamment du reste (Interruption Construire Résumé (TypeData**)**), c'est-à-dire que chaque véhicule privé qui se déplace, capture des données grâce à ces capteurs, ces données brutes sont ensuite filtrées grâce à des applications de filtrage -on prendre comme exemple: la plaque d'immatriculation, une information extraite d'un flux vidéo récupérée à l'aide d'un capteur vidéo-. Nous pouvons spécifier bien évidement quel type de donnée **TypeData** nous souhaitons récupérer et ceci dépends des besoins de l'application.

A l'information **Data** extraite, nous ajoutons une **Meta_data**(métadonnée) correspondante, qui n'est autre que: la position, l'heure et la date à laquelle fut capturée la **Data.** Nous ajoutons l'identifiant du véhicule, pour au finale encapsuler toutes ces informations dans un paquet, que nous avons appelé résumé **Sum** dont la structure est comme ceci:

Num_Sum	V_ID	Timestamp	Location	Data

Num_Sum: Identificateur du Summary (résumé) Sum.
V_ID: Identificateur du véhicule générateur du Sum.
Timestamp: L'heure et la date à laquelle le résumé a été généré,
Location: la position où ce résumé a été généré, cette information est récupérée du GPS.

Les données brutes restent au niveau du véhicule qui les a capturé, enregistré dans sa base de données locale, puisque nous n'avons besoin que de l'information extraite **Data** qui représentes l'information utile, ceci permet à la fois d'avoir des paquets **Sum** d'une taille réduite, mais aussi permet de diminuer le temps de transmission de façon considérable.

Comme les **Datas,** les Sums aussi seront sauvegardés dans la BD en locale mais seront par la suite disséminées (figure 3.7) au fur et à mesure que le véhicule privé se déplace, et rencontre d'autres véhicules.

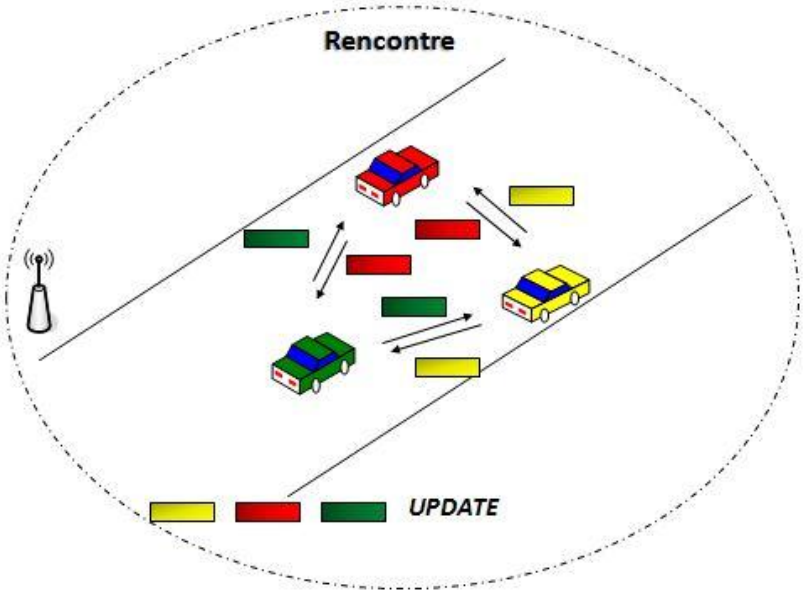

Figure 3.7: Echange de résumés dans une rencontre privé

La période à laquelle nous générons des résumés dépend directement du type d'application, ainsi si nous avons une application non tolérante au délai, il est nécessaire que la période soit très courte. Dans le cas contraire, la génération de résumés pourra se faire avec une période plus longue. Cependant, la période dépend aussi de la nature de la donnée capturée.

Le résumé récemment généré qui est sauvegardé en local, sera encapsulé dans un message **UPDATE** avec les informations de contrôles relatives. Ce message à la structure suivante:

Id_Msg	ID_Apt	V_ID	Timestamp	Sum

ID_ Msg: Identificateur du message UPDATE.
ID_Apt: Identificateur de la rencontre actuelle.
V_ID: Identificateur du véhicule générateur de l'UPDATE.
Timestamp: L'heure et la date à laquelle le UPDATE a été crée.
Sum: le résumé.

Une fois le message **UPDATE** envoyé, Notre protocole CDP passe à la dernière phase qu'est la fermeture, nous verrons plus en détails cette phase par la suite.

- **Rencontre Véhicule Agent:**

Ce que nous appelons rencontre véhicule agent c'est une rencontre où il existerait au moins un véhicule agent parmi l'ensemble des véhicules voisins directs. Ainsi lorsque CDP lance le processus d'élection de **VAC** comme nous l'avons vu précédemment, si l'identifiant local du véhicule qui exécute l'algorithme est le même que l'identifiant du VAC alors CDP lance une procédure de tri.

La procédure **Tri_List()** permet de retourner la liste des véhicules voisins qui ne font parti d'aucune rencontre à cet instant et qui sera triée suivant le filtre le bloom le plus différent par rapport au filtre de bloom de VAC, car c'est avec cette liste que le véhicule agent élu clusterhead **VAC** lancera sa collecte de résumés (figure 3.8). L'avantage dans cette méthode de tri réside dans le fait qu'on aura une collecte de résumés optimisée puisque nous privilégions la communication entre les véhicules, qui ont un contenu différent, et qui sont donc susceptible de rentabiliser notre première collecte.

Vu que seul le VAC aura la main pour faire de la collecte dans une rencontre véhicule agent, celui-ci va construire la première requête de collecte, qu'il va envoyer en unicast au premier élément de la liste des véhicules voisins déjà triée **List_VV**. Cette requête contiendra principalement le filtre de bloom de VAC le plus récent, dans le but d'informer le véhicule destinataire (agent ou privé) des résumés manquants. La requête ***REQ*** aura la structure suivante:

Id_Req	Bloom_Req	Time_start	Time_stop

ID_ Req: Identificateur de la requête de collecte ***REQ***.

Bloom_Req:filtre de bloom du VAC.

Time_start, Time_stop: intervalle de temps de collecte l'information à collecter.

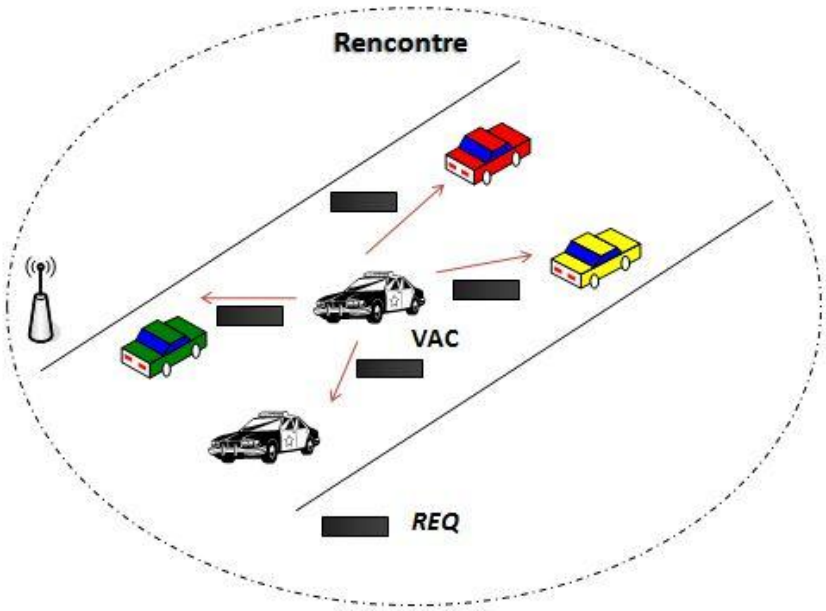

Figure 3.8: Envoi de la Requête de collecte par le VAC

Le véhicule qui reçoit ***REQ*** la requête de collecte prépare la liste des résumés demandés **List_Sum** à partir de sa base de données locale, grâce au **Bloom_Req** reçu dans la requête, mais aussi grâce à l'intervalle de temps, entre le **Time_start** et le **Time_stop**(Nous verrons par la suite dans la partie simulation), qui nous permettent d'avoir tout les résumés générés ou collecté de façon épidémique dans les rencontres privés dans un lapse de temps définis par une autorité ou paramètre au préalable, il faut savoir que plus le lapse de temps est grand plus la taille de l'information (**List_Sum**) augmente ce qui implique une réponse plus longue et donc un temps de transmission plus long vers le VAC ;**List_Sum** sera encapsulée avec des informations de contrôle,et sera envoyé en unicast vers VAC, la structure de notre réponse ***REP*** sera comme suit:

ID_Rep	ID_Apt	List_Sum

ID_ Rep: Identificateur de la réponse à collecte.

ID_Apt: Identificateur de la rencontre actuelle.

List_Sum: Liste des résumés générés ou reçus.

A la réception de du message ***REP,*** le VAC va fractionner l'information et si elle n'est pas vide, il met à jour sa base de données, D'où la nécessité de reconstruire un nouveau filtre de bloom qui sera envoyer dans notre prochaine requête de collecte ***REQ,*** VAC refait un tri grâce à la procédure **Tri_List()**décrite précédemment**,** ceci dans le but d'interroger toujours le véhicule susceptible de nous apprendre le plus dans la rencontre actuelle[5].

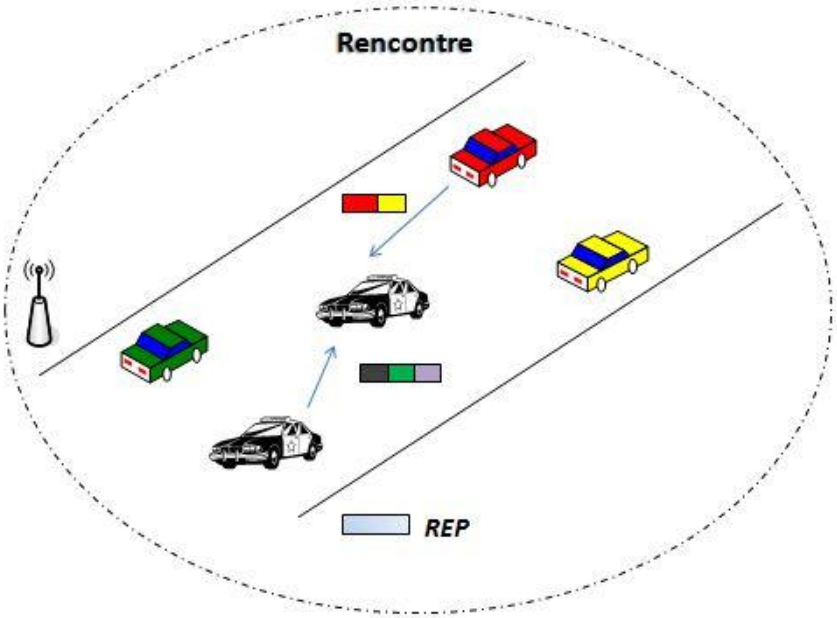

Figure 3.9: Envoi des réponses REP à VAC

Tant que la **List_VV** n'est pas vide, ces étapes seront répéter autant de fois qu'il en restera des véhicules à interroger et des résumés à collecter, Sinon le VAC procédera à la construction de ce que nous avons nommé un filtre de bloom rencontre **Bloom_Rencontre** comme résultant à partir de tout les filtres déjà reçus au tout début, dans la phase de découverte, cela va nous servir dans l'extraction de résumés manquant aux véhicules agents.

[5]**Remarque**: en considérant la forte mobilité dans les VSNs, Nous admettons la possibilité qu'un véhicule quitte la rencontre sans avoir transmit ces résumes, mais tôt ou tard ils finiront par être collectées par un autre véhicule agent lors d'autre rencontre.

Lorsqu'un **VAC** a fini de collecter les résumés de ces voisins directes (figure3.9), il doit mettre a jour les autres véhicules agents dans le but de leurs faire bénéficier de son traitement et afin d'éviter que ces mêmes agents ne réinterroge les mêmes véhicules privés pour collecter les mêmes résumés Ce qui optimise les communications et les délais de latence ; CDP permet donc de diminuer le nombre de messages qui seront échangés par rapport à MDHP.

Pour cela, le VAC dans CDP va extraire les résumés qui manquent aux véhicules agents en utilisant le **Bloom_Rencontre** généré. Ces résumés seront encapsulés dans un message que nous avons nommé ***SYNCHRO*** qui a la structure suivante:

ID _Msg	ID_Apt	List_Sum

ID_ Msg: Identificateur de la réponse à requête de collecte.
ID_Apt: Identificateur de la rencontre actuelle.
List_Sum: Liste des résumés manquants.

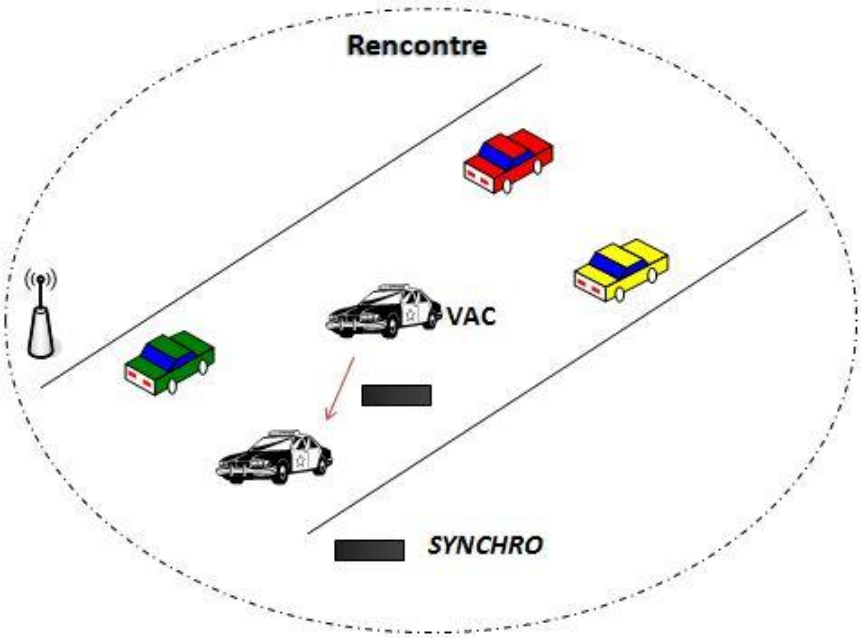

Figure 3.10: Envoi du SYNCHRO aux véhicules agents

SYNCHRO sera envoyé en broadcaste à tout les véhicules agents ayant fait parti de la même rencontre (c'est-à-dire dans la portée), ce qui va servir à synchroniser leurs

contenu (figure 3.10) puisque chaque agent qui reçoit ce message met-à jour sa base de donnée locale.

Une fois cette opération achevée, CDP appelle la dernière procédure fermeture que nous présentons ci-dessous.

3.4.3. ***Phase Fermeture:***

Cette phase permet de mettre fin à une rencontre. Elle peut être déclenchée soit à la fin d'une rencontre véhicule privé ou à la fin d'une rencontre agent ou alors suite à une interruption TTL (écoulement du **TTL**).

Le **TTL** ou *Time To Leave*, si à la fin du TTL le véhicule en attente ne reçoit aucun message alors il ne fait plus parti de la rencontre est doit terminer celle-ci.

La procédure de fermeture dans CDP, permet de mettre la variable **Bool_Rencontre** à Faux comme montrée dans la figure 3.11, ce qui signifie que ce véhicule est disposé à faire parti d'une prochaine rencontre s'il reçoit un ***MSG_DISCOVERY*****,** elle permet aussi d'armer un Timer nommé **Period_Discovery** comptabilisant le temps entre deux rencontres. Car le véhicule dont le temps est écoulé pourra à son tour prendre l'initiative d'être l'initiateur d'une nouvelle rencontre, ce qui nous permet d'avoir une organisation du réseau totalement indépendante d'une quelconque infrastructure.

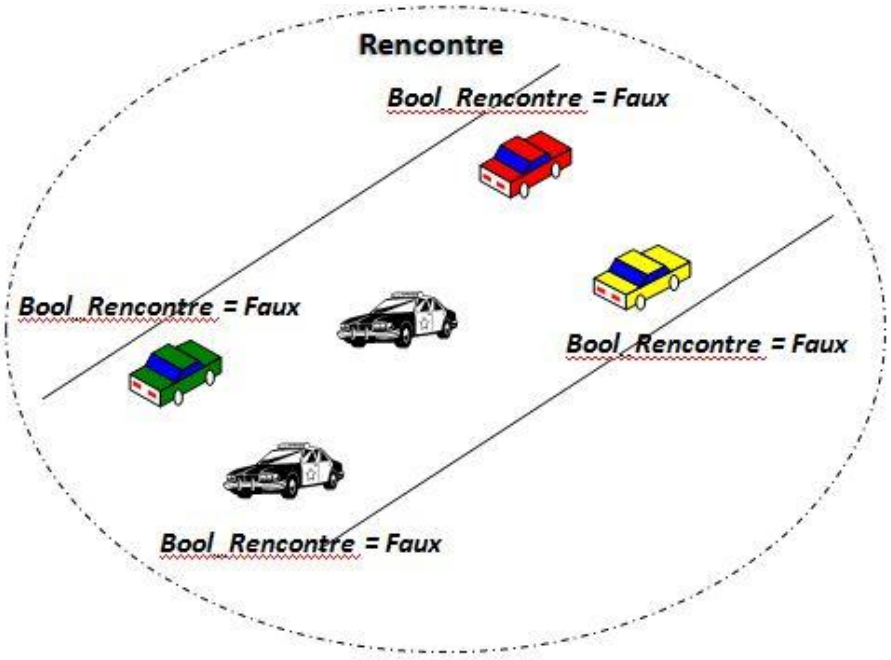

Figure 3.11: Fermeture de la communication et fin de rencontre

3.5. Structure de données

Chaque nœud possède les constantes et les variables suivantes:

Bool_Rencontre: Booléen pour dire si un véhicule est déjà dans une rencontre ou pas, initialisé à Faux.

List_VV: Liste des véhicules voisins d'un véhicule donné à un instant t, dans une rencontre, initialisé à vide.

List_VA: Liste des véhicules agents voisins d'un véhicule donné à un instant t, dans une rencontre, initialisé à vide.

TTL: « time to leave », délai au delà du quel le véhicule peut quitter la rencontre si il ne reçoit pas de message.

Period_Discovery: Timer ou temps entre deux rencontre 40 s, Durée maximale d'une rencontre.

Period_Summery: Période à laquelle, les véhicules privés génèrent des résumés,

Range: Rayon ou portée de communication.

N: Nombre de résumés émis dans une rencontre entre deux véhicules particuliers

V: Structure pour définir chaque véhicule {V_Id, V_Type, V_Bloom}, constituée d'un identifient unique, son type à savoir agent ou particulier, et un filtre de bloom local.

VAC: Véhicule agent cluster Head initialement à vide.

Sums: Liste des derniers résumés d'un véhicule donné, initialement à vide

List_Sum: Liste des résumés manquant à un véhicule donné, initialement à vide.

Bloom: Structure de données représentant le filtre de Bloom, ici nous avons utilisé une matrice d'éléments booléens binaires qui indique ou non l'existence d'un résumé,

Si B [*i,j*] =1 alors le résumé du véhicule *i* au temps *j* existe dans la base de donnée locale,

Sinon B [*i,j*] =0 alors le résumé du véhicule *i* au temps *j* n'existe pas.

3.6. Algorithme CDP:

3.6.1. ***Phase Découverte***

Interruption Découverte ()

```
//Se déclenche lorsque Period_Discovery arrive à ZERO
{
Debut
    Construire_Bloom(V_Bloom) ;
    Msg_Discovery = {Id_Msg, ID_Apt,  V_id, V_bloom}
    Envoyer (Msg_Discovery) // en broadcast sur un rayon R
Fin
}
```

Procédure lors de la réception du Message Découverte (Msg_Discovery)

```
// Se déclenche suite à la réception d'un message de découverte
{
Debut
    Si Bool_Rencontre == Vrai ;
        Alors ignorer  // véhicule appartient à  une rencontre
        Sinon  Bool_Rencontre = vrai
            Armer (Period_Discovery)
            Armer (TTL) //  à chaque fois que je reçois un message
            Construire_Bloom(Bloom)
            REP_ Discovery = {Id_Msg, ID_Apt, V_id, V_bloom}
```

Attendre (ΔT*Random(N)) // Attente un temps variable pour éviter les Collisions en plus du Mécanisme Wave

Envoyer (REP_ Discovery)

Fsi

Fin

}

Procédure lors de la réception de la réponse (REP_ Discovery)

{

Attendre (T)// Attendre toutes les réponses de tout les véhicules voisins à un seul saut

Rencontre (List_VV)// Appelle la procédure de rencontredans la phase transmission

}

3.6.2. *Phase transmission:*

Procédure Rencontre (List_VV)

{VAC = vide ; //véhicule agent clusterhead

List_VV = {V_ID,

V_Type,

Bloom}

List_VV = vide

List_VA = vide

Debut

VAC = Elire_Clusterhead (List_VV)// Elire l'agent qui a la BD la plus rempli

Si VAC == Vide

Alors Rencontre_VP(List_VV) // lancer la procédure Rencontre_VP

Sinon

Si Local.V_ID = VAC.V_ID // si je suis le véhicule clusterhead

Alors List_VV = Tri_List (List_VV, VAC) // liste d'agent et de client triée suivant la différence de Bloom de VAC

Fsi

Rencontre_VA(List_VV,List_VA)//lancer la procédure Rencontre_VP

Fsi;

Fin

}

Fonction Elire_Clusterhead (List_VV)

{

Debut

i=0;

List_VA = Filtrer_Agent (List_VV) // crée une liste d'agent participant à l'interaction

grâce au type agent

Pour chaque List_VA [i]

Faire

// Fonction qui compare le contenu BD entre chaque deux véhicule agent de la liste

VAC = Compare (List_VA [i].Bloom, List_VA [i+1].Bloom*)*

Fait

Retourner (VAC, List_VA) ;

Fin

}

Tri_List (List_VV, VAC.Bloom)

{

List_VV = Tri (List_VV, VAC) // Tri-liste retourne une liste de véhicules triée suivant la différence de leur filtre de bloom avec VAC.bloom

Retourner (List_VV) ;

}

Procédure Rencontre_VP (List_VP)

{

Debut

Sums = Local_ BD.Last[N] ;// récupérer (N) derniers enregistrements dans la BD

Update = {Id_Msg, ID_Apt, V_id, V_timestamp, Sums}

Attendre (ΔT*Random(N)) // Attente un temps variable pour éviter les Collisions en plus du Mécanisme Wave

Envoyer (Update) // envoyer mon sum en déférer à tout les véhicules particuliers de la rencontre en broadcast

Fermeture ()// fonction fermeture de la phase communication

Fin

}

Procédure Recontre_VA(List_VV,List_VA) // exécuter uniquement par le VAC

```
{
Debut
    Si List_VV != vide
    Alors
        REQ_Collecte = Construire-Requete (Time_Start, Time_Stop)
        Envoyer (REQ_Collecte, List_VV[1]); // envoyer requête au premier de la
                                            liste trié en Unicast
    Sinon // ya il n y a plus de véhicule different de VAC
        Bloom_Rencontre = Construire_Bloom_Rencontre(List_VV)
        List_Sum = Extraire_ Data (Bloom_Rencontre) // permet d'extraire les sum
                                                    Manquants des autres
                                                    Agents
        Synchro = {Id_Msg, ID_Apt, List_Sum}
        i=1
        Tantque List_VA[i] =! VAC
        Faire
            Envoyer (Synchro //update en broadcaste vers tous les agents
        Fait
    Fsi
Fermeture ()// fonction de fermeture fin de la rencontre
Fin
}
```

Fonction Construire_Requete (Time_Start, Time_Stop)

```
{
 Debut
    REQ = Vide;
    Bloom_Req = Construire_bloom(Bloom)
    REQ = {Id_Req, Bloom_Req, Time_start, Time_stop}
    Retourner(REQ)
Fin
```

```
}
```

Interruption ou procedure lors de la réception d'une Requête (REQ)

```
{
Debut
        REP = Construire_Reponse(REP)
        Envoyer (REP)// en Unicast List_ sum reçu et emis construit à partir du
                        filtre de bloom reçu dans la requete
Fin
}
```

Fonction Construire_Reponse(REQ)

```
{
Debut
        REP = Vide
        REP= {Id_Rep, Id_Apt, List_Sum }
        Retourner (REP)
Fin
}
```

Interruption ou procédure lors de la réception d'une Reponse (REP)

```
{
Debut
        Si REP != vide Alors  Sauvegarder_Reponse (REP) //sauvegarde BD
        Contruire_bloom (Local Bloom)//update le filtre de bloom local avec les
                                        Nouveaux sum recus
        Rencontre (List_VV)
Fin
}
```

Fonction de Sauvegarde Reponse(REP)

```
{
Debut
        Fractionner (REP)
        Enregistrer (REP); // localement
Fin
```

}

Fonction Interruption Construire Résumé (TypeData)//à chaque Period_Summery

{

Debut

Data = Filtrer (Local_data, TypeData) //Application de filtragesuivantTypeData

Meta_data = { V_id, Timestamp, Location }

Sum = Sauvegarder (Num_Sum, Meta_data, Data)

Fin

}

Interruption ou procedure lors de la réception de SYNCHRO / UPDATE:

{

Debut

Fractionner (MESSAGE)

Enregistrer (SUM) ; // localement

Fin

}

Construire bloom(Bloom)

{

Debut

Pour tout element de la Base de données

Si sum existe

alors B [*i,j*] =1

Sinon B [*i,j*] =0

Fsi

Fin

}

3.6.3. ***Phase fermeture:***

Interruption Fermeture_TTL()

// S'exécute à chaque fois que le TTL arrive à zéro

{

Debut

Fermeture ()

Fin

}

Procédure Fermeture_()

{

Debut

Armer (Period_Discovery) ; // Timer

Bool_Rencontre = Faux //véhicule disposé à recevoir des demandes de collecte du réseau

Fin

}

4. Conclusion:

Nous avons proposé un protocole de dissémination qui améliore une proposition adoptée à la dissémination données dans les VSNs, en nous inspirant du système MobEyes. Cette solution améliorée permet aux véhicules agents de collaborer dans un réseau VSN afin d'assurer le contrôle d'une zone urbaine.

Dans le chapitre suivant, nous allons simuler notre protocole sous OMNET++, pour démontrer son efficacité en matière de latence dans la collecte et en matière de gain en surcharge réseau par rapport au protocole MDHP qui traite les mêmes problématiques dans les VSNs.

Chapitre IV:Evaluation des performances

1. Introduction:

Avant sa mise en place, le déploiement d'un VSN nécessite une phase de simulation afin de s'assurer du bon fonctionnement de tous les protocoles de communication qu'il utilise. En effet, pour de grands réseaux, le nombre de véhicules dotés de capteurs peut atteindre plusieurs milliers et peut entrainer donc un coût financier relativement important. Ainsi, il faut réduire au maximum les erreurs de conception.

Pour évaluer les performances du protocole que nous avons proposé dans le chapitre précédent, et de le comparer avec le protocole du système MobEyes, nous avons effectué des simulations pour évaluer les avantages apportés par notre protocole. Notre choix s'est fixé sur l'environnement de simulation offert par OMNET++ [OMN12] qui est un simulateur modulaire, développé en langage C++.

Dans ce chapitre, nous allons expliquer les raisons du choix de ce simulateur. Nous présentons aussi la plateforme de simulation OMNET++ ainsi que le modèle de mobilité très utilisé dans le milieu Ad hoc pour sa gratuité et aussi parce qu'il offre des modèles réalistes, appelé SUMO.

Après cela, nous détaillons le choix des critères d'évaluation, et les scénarii adoptés pour les simulations. Nous terminerons par l'interprétation des résultats de simulation obtenus.

2. Environnement de simulation:

Dans une simulation d'événement discret, le travail d'un système est représenté par le déroulement chronologique de séquences d'événements. Chaque événement se produit à un instant bien déterminé et marque un changement d'état dans le système.

Une simulation d'événement discret a pour but de visualiser l'état du système à n'importe quel instant. De telles simulations obéissent à une logique de conception qui permet de valoriser certains aspects du système, comme par exemple, dans notre cas, l'efficacité du protocole à échanger les données entre nœuds mobiles (véhicules). Pour cela, nous présentons le processus de simulation que nous avons adopté pour l'élaboration de notre projet (figure 4.1).

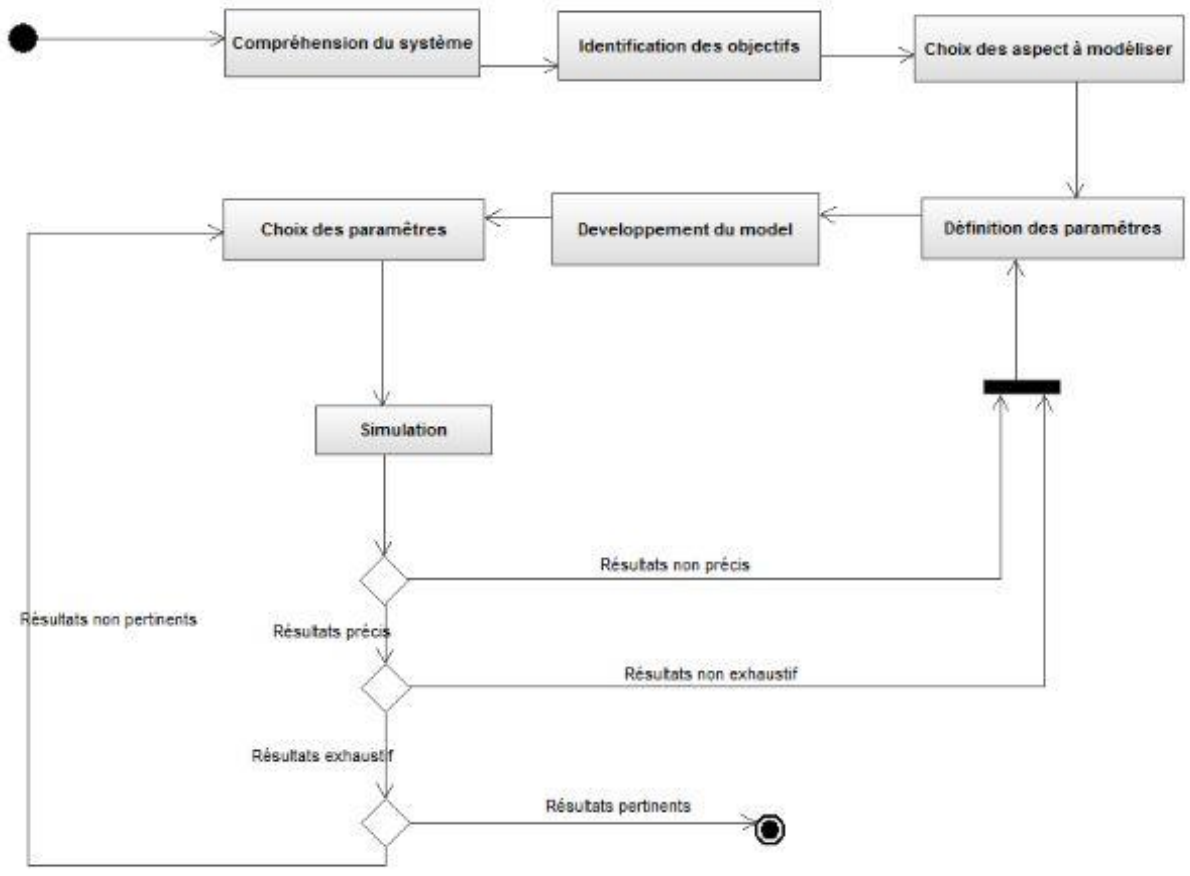

Figure 4.1: Processus de la simulation

Les besoins croissants de tester les nouvelles technologies et les nouveaux protocoles, avant leur déploiement, a conduit à la prolifération des simulateurs. On peut les classer en deux types: les logiciels libres et gratuits tels qu'OMNet++, J-Sim et NS2... et les logiciels commerciaux tels qu'OPNET et Qualnet.

2.1. Le simulateur NS2:

NS2 est un outil de simulation de réseaux [NS212]. Il est bâti autour d'un langage de programmation appelé TCL (Tool Command Language). Du point de vue de l'utilisateur, la mise en œuvre de ce simulateur se fait via une étape de programmation qui décrit la topologie du réseau et le comportement de ses composants, puis vient l'étape de simulation proprement dite et enfin l'interprétation des résultats. Cette dernière étape peut être prise en charge par un outil annexe, appelé NAM [NS212].qui permet une visualisation et une analyse des éléments simulés. NS2 est, en réalité, un programme relativement complexe écrit en C++ et interfacé via TCL. Pour modifier le comportement d'objets existants, il est donc nécessaire de modifier le code C++ qui en réalise l'implantation. TCL est un langage de commandes qui sert à contrôler les applications. C'est essentiellement un langage de scripts offrant des structures de programmation telles que les boucles, les procédures ou les notions de variables.

2.1.1. *Composants de NS2*

Le tableau 4.1 ci-dessous, énumère les principaux composants de NS2.

Application	Web, FTP, Telnet, générateur de trafic (CBR)
Transport	TCP, UDP, RTP, SRM
Routage	Statique, dynamique (vecteur distance) et routage multipoint (DVMRP, PIM)
Gestion des files d'attente	RED, Drop Tail, Token bucket
Discipline de service	CBQ, SFQ, DRR, Fair queueing
Système de transmission	CSMA/CD, CSMA/CA, lien point à point

Tableau 4.1. Principaux composants NS2

2.1.2. ***Modèles de mobilité:***

Parmis les modèles de mobilités implémentés par NS2, nous trouvons:

- *Le Random Waypoint Mobility Model:*

  Ce modèle génère un mouvement aléatoire du noeud de façon que ça soit le nœud qui choisi sa prochaine destination d'une manière aléatoire en se déplaçant avec une vitesse aléatoire constante.

- *Le Trajectory Based Mobility Model:*

  C'est un mouvement généré par un scénario qui consiste à ce que l'utilisateur donne une destination bien précise et une vitesse de déplacement constante.

2.1.3. ***Structure d'un noeud mobile dans NS2:***

Dans NS2, un noeud mobile a une structure qui peut être schématisé par la figure 4. 2.

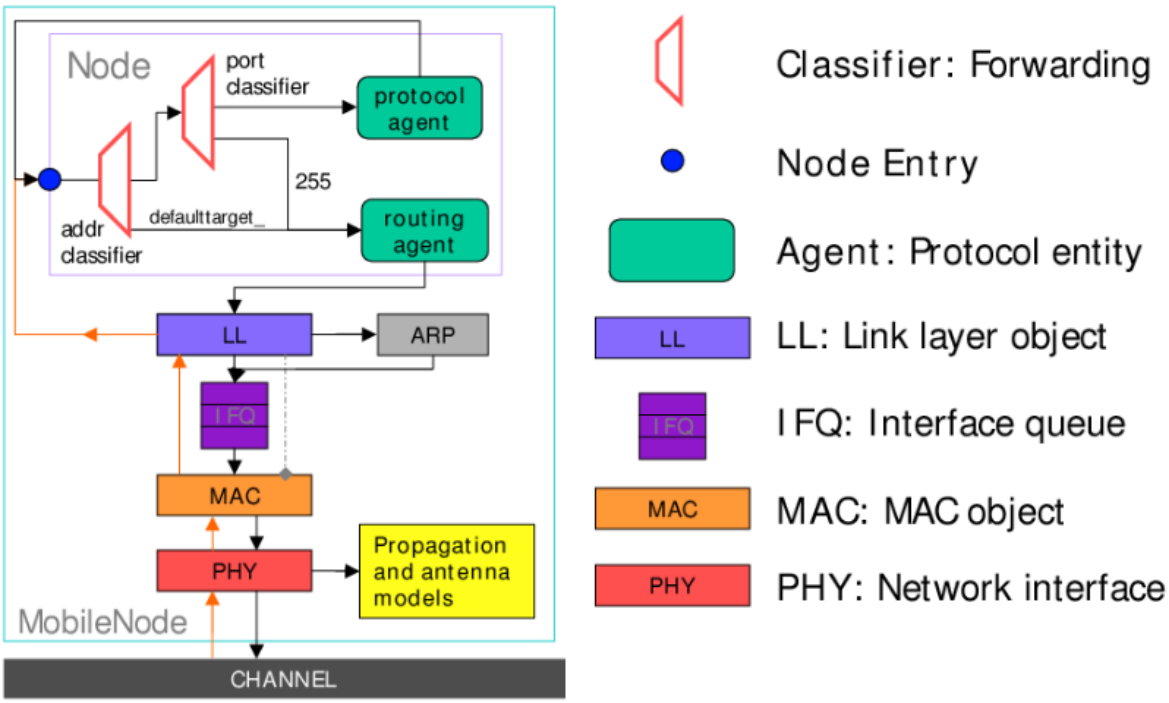

Figure 4.2: Structure d'un nœud mobile dans NS2

2.2. Les simulateur OMNet++:

OMNET++ est un simulateur à évènements discrets orienté objet et basé sur C++. Il a été conçu pour simuler les systèmes réseaux de communication, les systèmes multi processeurs, et d'autres systèmes distribués [OMN12].OMNET++ est un projet open source développé à l'université de Budapest. [DSH03]. Actuellement, Ce simulateur est utilisé par des dizaines d'université pour la validation de nouveaux matériels et logiciels, ainsi que pour l'analyse de performance et l'évaluation de protocoles de communication. [KSW08].Les avantages de OMNET ++ ce sont principalement sa facilité d'apprentissage, d'intégration de nouveaux modules et la modification de ceux déjà implémentés.

2.2.1. *Architecture d'OMNET++:*

L'architecture d'OMNET***++*** est hiérarchique composée de modules. Un module peut être soit un module simple soit un module composé. Les feuilles de cette architecture sont les modules simples qui représentent les classes C++. Pour chaque module simple correspond un ***fichier.cc*** et un ***fichier.h***. Un module composé comporte de simples modules ou d'autres modules composés connectés entre eux. Les paramètres, les sous modules et les ports de chaque module sont spécifiés dans un ***fichier.ned.***

La communication entre les différents modules se fait à travers les échanges de messages. Les messages peuvent représenter des paquets, des trames d'un réseau informatique, des clients dans une file d'attente ou bien d'autres types d'entités en attente d'un service. Les messages sont envoyés et reçus à travers des ports qui représentent les interfaces d'entrer et de sortie pour chaque module.

La conception d'un réseau se fait dans un ***fichier.ned*** et les différents paramètres de chaque module sont spécifiés dans un fichier de configuration (**.ini**). OMNET++ génère, à la fin de chaque simulation, deux nouveaux fichiers ***omnet.vec*** et ***omnet.sca*** qui permettent de tracer les courbes et calculer des statistiques.

2.2.2. ***Composants***

Les principaux composants d'OMNET++ sont présentés par le tableau 4.2.

Application:	FTP, Telnet, générateur de trafic (IPTrfGen..), Ethernet, Ping App, UDPApp, TCPApp
Transport:	TCP, UDP, RTP
Réseau:	IPv4, IPv6, ARP, OSPF, LDP, MPLS, ICMP, TED...
Liaison:	Mgmt, MAC, Radio
Node:	Ad Hoc, Wireless, MPLS...

Tableau 4.2. Principaux composants OMNET++

2.2.3. ***Structure d'un nœud mobile dans OMNET++***

Dans OMNET++, un nœud mobile a une structure représenté par la figure 4.3

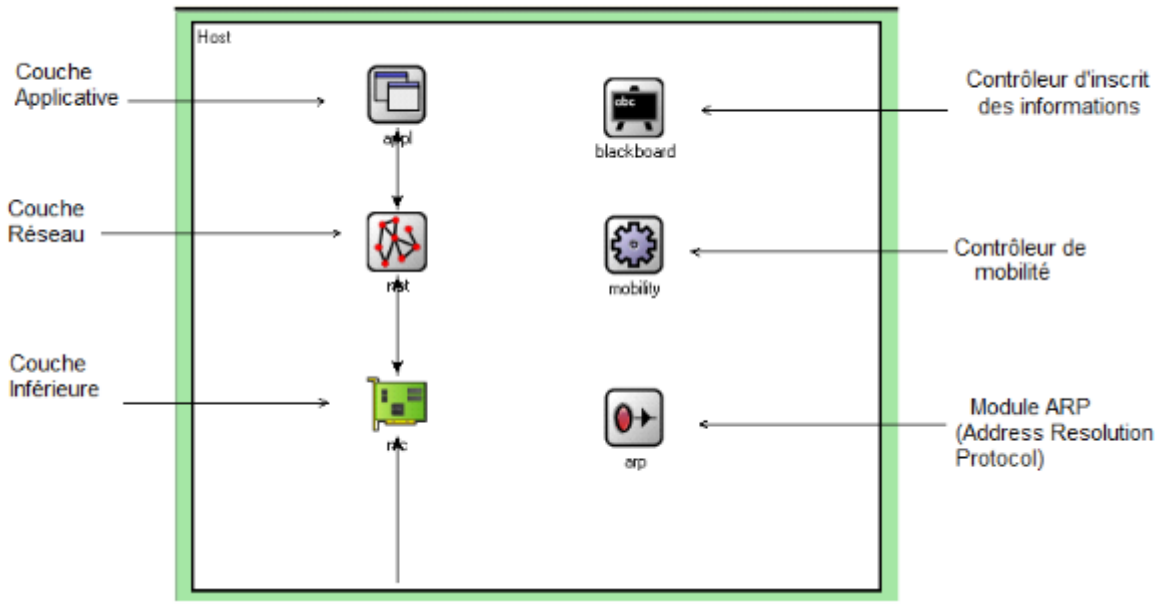

Figure 4.3. Structure d'un nœud mobile dans OMNET++

2.3. Choix du simulateur:

Notre choix s'est porté sur OMNeT++ qui est de plus en plus utilisé par la communauté scientifique et les chercheurs font de plus en plus confiance aux résultats obtenu par ce simulateur, OMNeT++ permet une simulation des réseaux de capteurs véhiculaires. Ces principales possibilités sont:

- Mobilité: OMNeT++ permet de prendre plus facilement en compte la mobilité des nœuds dans le réseau, grâce à la librairie Mobility framework **[MFR]**.
- Topologie dynamique: Contrairement à d'autres simulateurs, OMNeT++ et grâce à son éditeur graphique NED, permet de créer des topologies entièrement paramétrables. De plus, sa structure hiérarchique donne une flexibilité dans l'architecture du modèle.
- Facilité de débogage: OMNeT++ propose trois outils de débogage différents comme une fenêtre « module output », un inspecteur et l'animation automatique.
- Accès au code source: OMNeT++ est totalement open-source où il met ses sources à la disposition des utilisateurs.
- Facilités d'utilisation et d'exploitation des résultats grâce à des outils permettant de créer des fichiers facilement récupérables pour d'autres applications.

Pour toutes ces raisons et plus, nous avons développé notre simulation basé sur OMNet++ pour l'étude des performances de notre algorithme. Dans ce qui suit nous allons voir en détail les frameworks qu'offre OMNet++, et ces modèles de mobilité.

2.4. Frameworks OMNeT++:

OMNeT++est conçus à base de composants, ce qui signifie que de nouvelles fonctionnalités et protocoles peuvent être pris en charge par des modules. Il peut prendre en charge des modèles de réseaux et la mobilité à travers des modules développés indépendamment: Mobility Framework[6] et INET Framework. Réalisation de la simulation dans OMNeT se fait sur une interface graphique.

[6]Mobility framework est incluse dans l'API Mixim.

2.4.1. ***MiXim:***

La librairie MiXim [MIX12]est une extension du simulateur OMNET++. C'est une fusion de plusieurs frameworks OMNeT + +, on cite ChSim, Mac Simulator, Mobility Framework [MFR10], Positif Framework. Sa dernière version a été proposée en Octobre 2011. Cette librairie est destinée à soutenir les réseaux sans fil et mobiles au sein du simulateur OMNET++. En effet, elle permet une bonne manipulation des nœuds mobiles et la gestion des connexions dynamiques. Ceci dans le but d'avoir un modèle de mobilité de réseau sans fils qui fournie des résultats les plus proches possible du monde réel. En outre, MiXim prévoit des modules de base qui peuvent être utilisés pour former de nouveaux modules. Avec ce concept, un programmeur peut facilement développer ses propres protocoles avec l'interface nécessaire. MiXim peut être utilisée pour simuler:

- les réseaux sans fil fixes,
- les réseaux sans fil mobiles,
- les réseaux distribués (ad-hoc) et les réseaux centralisés,
- les réseaux de capteurs,
- les multi-réseaux sans fil. [KSW11]

Aujourd'hui, il existe une bibliothèque de protocoles normalisés pour la MiXim (802,11P, AODV, ...) qui permettent de disposer d'une bibliothèque riche de ces protocoles afin de permettre facilement des simulations de différents types de protocoles largement utilisés.

a) L'interface réseau IEEE 802.11 de la librairie MiXim [KSW11]

La base pour toutes les classes des modules de la couche physique, est celle de l'accès au canal *ChannelAccess* qui est à son tour dérivée de *BasicModule*. La seule fonctionnalité que fournit *ChannelAccess* est la connectivité au *Channel* (i.e. aux autres nœuds). La fonction *sendToChannel ()* doit être utilisée pour passer des messages au Channel. Elle envoi le message pour chaque ***gate*** connecté du module de l'hôte.

- **PhyLayer: Nic80211a**

Il existe deux versions de PhyLayer. La première version est appelée P2PPhyLayer, elle assume les connexions point à point entres les hôtes. C'est la plus simple couche physique que l'on puisse utiliser. Pour la seconde version, elle est appelée Nic80211. Dans cette version, les fonctionnalités de la couche physique ont étés divisées en trois modules simples qui sont: Decider80211a, SnrEcal80211a et Mac80211a. Les calculs de SNR ont été laissés séparés du Decider à cause des bits errors. Ce concept rend la création de modules Decider (qui utilisent le même SnrEval module) plus facile, et vis versa.

On peut, par exemple, définir un module Decider qui compare les calculs de SNR avec un qui utilise le forwarderror correction et les deux modules peuvent utiliser le même SnrEval module.

- **Mac80211a:**

La classe Mac80211a est une implémentation de la norme IEEE 802.11a. Nous nous sommes basés sur pour extraire les différents paramètres comme le calcul de la durée de transmission packetDuration(), la durée des périodes.

- **SnrEval:**

Le module SnrEval simule un délai de transmission pour tous les messages reçus et calcule les informations SNR. Le *BasicSnrEval* ne tient pas compte des retards de propagation. Les SNR informations sont stockées dans un *SnrList*. Chaque entrée SnrList contient un *timestamp* et un SNR de la valeur de ce *timestamp*. *HandleLowerMsg()* est subdivisé en *handleLowerMsgStart()* et *handleLowerMsgEnd().*

Un message est tamponné (fonction *bufferMsg ()*) pour simuler un délai de transmission. Entre temps, il y a d'autres messages qui peuvent arriver mais, ils interférent avec la mémoire tampon du message. Par conséquent, il en résulte de nouvelles entrées *SnrList* supplémentaires pour indiquer un changement de SNR pour ce message.

Une fois la transmission achevée (c'est-à-dire, que le message soit complètement reçu), *handleLowerMsgEnd()* est appelée juste avant que le message soit transmis au *Decider* module. Ici, le message devrait être supprimé du tampon de réception et de la *SnrList* contenant les informations SNR calculés. Ainsi, les valeurs devraient être passée en argument à la fonction *sendUp ()*. La fonction *sendUp()* est celle qui assure la transmission du paquet vers la couche supérieure .

- **Decider**

Le module *Decider* ne traite que les messages provenant de la chaîne (c'est-à-dire à partir de la couche inférieure). Les messages provenant des couches supérieures, ayant contourné le module *Decider*, sont directement remis au module *SnrEval*. Les décisions concernant la perte ou les erreurs sur les bits messages ne sont faites que pour les messages provenant de la chaîne.

- **Blackbaord:**

Pour évaluer les performances d'un protocole, nous pouvons utiliser un *Blackbaord*. L'état des changements. IL permet aussi d'échanger des informations entre les couches, sans passage de pointeurs vers d'autres modules.

b) Les modules de mobilités: [KSW11]

- **L'architecture de la mobilité**

Cette section décrit l'architecture de la mobilité utilisée dans MiXim. Il ya deux questions à examiner concernant une architecture de mobilité dans un cadre de simulation. La première question est de savoir où traiter les informations sur la mobilité et le deuxième est: comment gérer les mouvements des hôtes, c'est-à-dire Où et comment gérer dynamiquement les connexions ? La gestion de la connexion est gérée par un contrôleur central.

La composante de base de l'architecture de la MIXIM le module *ChannelControl* avec un hôte *Mobility Module*. *Channel Control* s'occupe de tous les paramètres liés à la connexion alors que les *Mobility Modules* ont deux tâches principales: la première tâche est de gérer les mouvements de l'Hôte qui peut être fait en utilisant différents modèles

de la mobilité (comme le modèle de mobilité SUMO), alors que le second rôle des *Mobility Modules* est de communiquer pour contrôler l'évolution de l'hôte.

- **Le ChannelControl:**

Le module *Channel Control* est chargé d'établir des canaux de communication entre les modules d'accueil (qui sont à distance de communication) ainsi que de rompre ces liens une fois qu'ils perdent la connectivité à nouveau. La perte de la connectivité peut être due à la mobilité (c'est-à-dire les hôtes se déplacent trop loin) ou en raison d'un changement dans la transmission de puissance ou d'un hôte.

2.4.2. ***INET:***

Est une librairie open source pour la simulation des réseaux informatiques dans l'environnement OMNeT++. Elle contient IPv4, IPv6, TCP, UDP, des protocoles implémentés, et plusieurs modèles d'application. Elle comprend également un modèle avec MPLS RSVP-TE et de signalisation LDP. La couche liaison contient des modèles de PPP, Ethernet et 802.11. Le routage statique peut être configuré à l'aide du réseau auto configurateur, ou on peut utiliser le protocole de routage mise en œuvre.

INET prend en charge les simulations des réseaux sans fil et mobiles, ainsi les réseaux ad-hoc. Récemment les modèles MPLS, RSVP-TE et LDP sont révisés et réécrit, sans oublier le routage dynamique (RIP et OSPFv2).

Dans l'annexe nous avons présenté une étude de l'existant de la librairie INET et plus précisément l'implémentation des couches PHY, MAC, IP, RTP, UDP et TCP dans INET.

2.4.3. ***SUMO:***

SUMO (Simulation of Urban MObility) [SUM12] est un simulateur de mobilité open source, programmé en C++, hautement portable, il permet de générer des scénarios de mouvement au niveau microscopique. C'est-à-dire qu'il peut générer des routes, des véhicules, ainsi que des feux de signalisation.

Ses principales caractéristiques permettent le choix: du type de véhicules, du type de route multivoies, la jonction base de droit de passage, la hiérarchie dans les types de jonction, une Interface graphique utilisateur OpenGL (GUI), et le routage dynamique.

SUMO peut gérer les environnements de grande taille, et il peut importer des formats de nombreux modèles réseaux tel que du XML.

SUMO peut communiquer avec OMNeT + + via une connexion TCP. Afin de créer une communication bidirectionnelle entre les deux simulateurs, OMNeT + + a également été étendue par l'ajout d'un module qui permet à tous les nœuds participants (véhicules) d'envoyer des commandes via la connexion TCP établie au niveau de SUMO. Dans ce cas, les deux extensions constituent l'interface entre le simulateur de réseau et le simulateur mobilité. Ainsi, OMNeT + + peut réagir à la trace de mobilité reçue de SUMO, et ceci en introduisant de nouveaux nœuds, en supprimant les nœuds qui ont atteint leur destination, et en déplaçant les nœuds, suivant les instructions de SUMO.

Il y a un module de gestion qui est responsable de la synchronisation des deux simulateurs. A intervalles réguliers, le module de gestion déclenche l'exécution d'un pas de temps de la trace de simulation. Il reçoit la trace de mobilité qui en résulte, et déclenche une mise à jour de la position pour tous les modules qu'il avait instanciés.

MOVE: Move est un logiciel programmé en java pour générer des simulations du trafic routier, il génère rapidement des modèles de mobilité réalistes pour des simulations de VSN et VANET. MOVE est construit au-dessus de SUMO. En sortie il offre un fichier de trace de mobilité qui contient des informations sur les mouvements réalistes des véhicules qui peuvent être immédiatement utilisées par les outils de simulation de réseaux OMNET++. En outre, MOVE fournit une interface graphique qui permet à l'utilisateur de générer rapidement des scénarios de simulation, sans les tracas de l'écriture de scripts de simulation. Ces principaux composants sont les suivants: **Map Editor** et **Vehicule Mouvement Editor**: [VEI12]

a) Map Editor:

C'est un éditeur de carte, dédier a la création d'une topologie de route. Cette création peut être faite de deux manières différentes: Elle est soit crée manuellement par un utilisateur, ou alors générée aléatoirement. (Figure 4.4).

b) Vehicule Mouvement Editor:

Ce composant permet d'éditer les mouvement, il génère automatiquement ou manuellement le mouvement du véhicules, dans le but de spécifier plusieurs propriétés concernant les itinéraires des véhicules incluant le nombre de véhicules dans une route particulière, l'heure de départ des véhicules, l'origine et la destination des véhicules , la durée du voyage, la vitesse (accélération, décélération , vitesse maximale …), Aussi, l'utilisateur peut définir la probabilité qu'un véhicule prenne une direction donnée lorsqu'il est dans une jonction (exemple: 0.5 pour tourner à gauche, 0.3 pour aller directement). Voir la Figure 4.4:

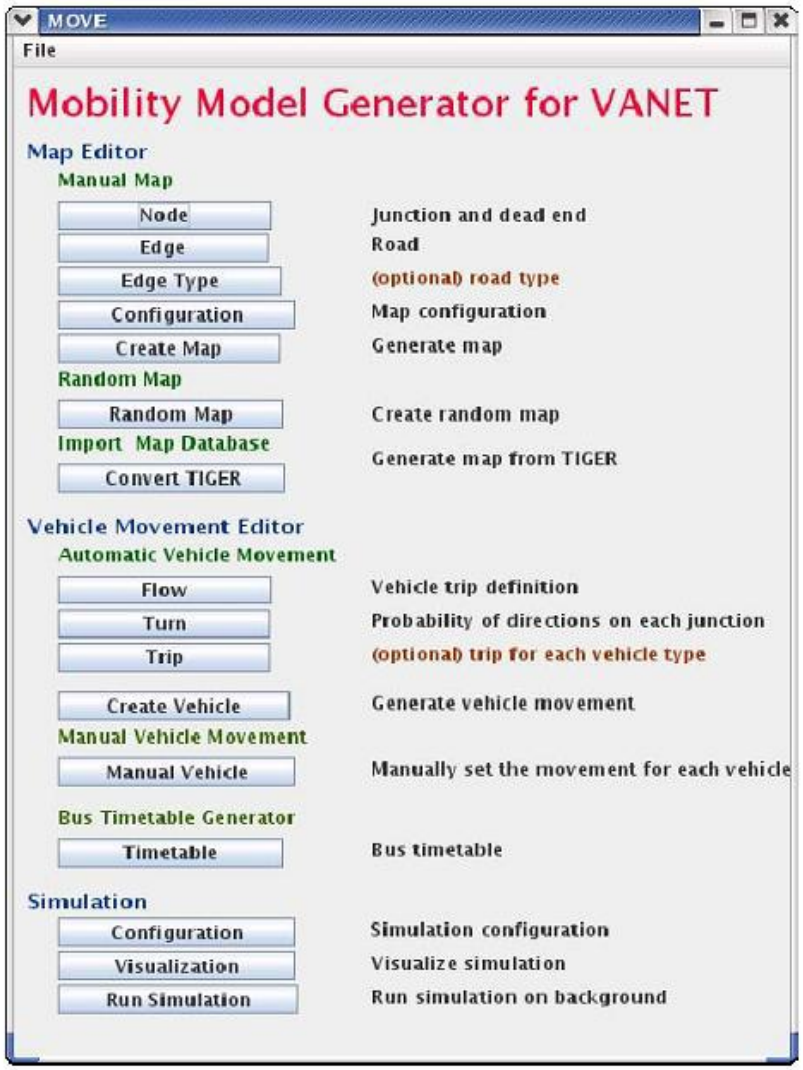

Figure 4.4. GUI MOVE [VEI12]

La figure 4.5 suivante résume les étapes à suivre pour la génération du fichier de mobilité avec MOVE, En effet la création de fichier de mobilité se fait en 3 étapes:

i. **Map module**: cette partie permet de créer la topologie de la carte routière (nombre de route, nombre de voies par route, feux de signalisation, longueur de routes, vitesse maximale autorisée sur chaque route, etc.)
ii. **Move module:** elle permet de créer les véhicules qui vont se déplacer sur la carte crée précédemment. Elle permet de les initialiser, c'est-à-dire spécifier leur point de départ, leur point d'arrivée, la vitesse maximale qu'ils peuvent atteindre, ainsi que d'autres propriétés, comme le choix de la prochaine route à prendre au niveau d'une intersection donnée en utilisant les probabilités.

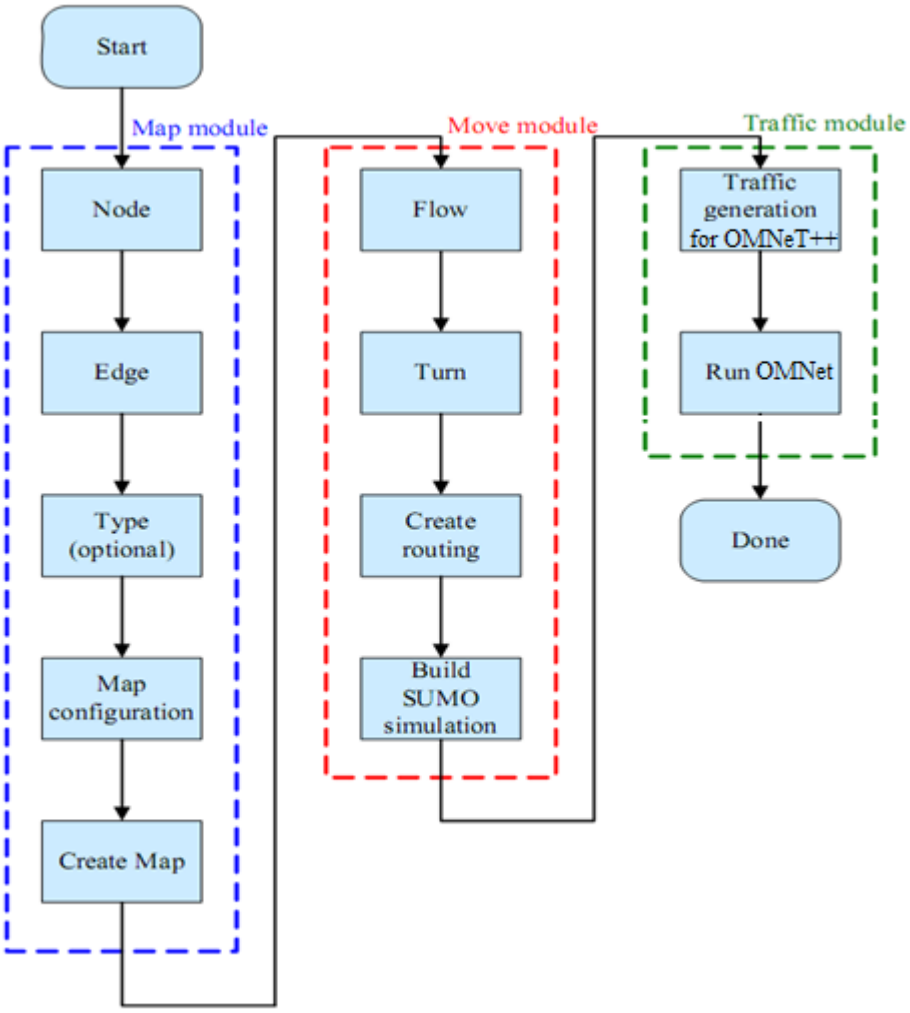

Figure 4.5. Etape de génération de fichier trace de trafic avec MOVE.

iii. **Trafic module:** cette partie permet de regrouper les informations des deux précédents modules, et de générer le fichier de mobilité qui sera utilisé par la suite par OMNET++. Et grâce à cela, on peut voir le déplacement des véhicules suivant les propriétés spécifiées au par avant. Le fichier de mobilité généré reste à l'intégrer au script de l'OMNeT++ avant de lancer la simulation.

3. Mise en œuvre de la simulation:

3.1. Besoins généraux:

Pour pouvoir choisir le meilleur simulateur, il faut spécifier nos besoins. Nous devons disposer de:

i. Module ARP:

Ce module du noyau implémente le protocole de résolution d'adresse ARP. Il sert à la conversion entre les adresses matérielles de niveau 2 et les adresses du protocole IPv4 sur les réseaux connectés en direct. L'utilisateur n'a normalement pas d'interactions avec ce module sauf pour le configurer. En fait, ce module fournit des services aux autres protocoles du noyau. Le module ARP maintient un cache des correspondances entre les adresses matérielles et les adresses logiques. Le cache à une taille limitée, ainsi les entrées anciennes et utilisées moins fréquemment sont récupérées. Les entrées qui sont marquées comme permanentes ne sont jamais effacées.

ii. Module NIC802.11:

Ce module permet de fournir l'adressage MAC et encapsuler les paquets en trames 802.11. Il est composé de 3 modules simples qui sont Decider80211a, SnrEcal80211a et Mac80211a. Comme le montre la figure 4.6, ce module possède deux interfaces de sortie avec la couche IP et avec le canal.

iii. Module de mobilité:

Ce module fournit un modèle de mobilité qui permet de faire déplacer les nœuds, mais permet aussi d'être lié à un autre modèle de mobilité, dans notre cas nous avons préféré utiliser celui de SUMO car c'est le plus utilisé dans les VSNs/VANETs .

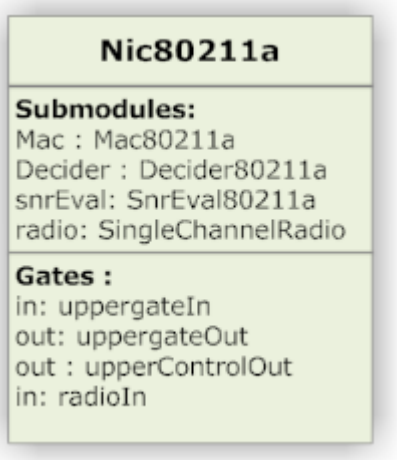

Figure 4.6 Le module Nic80211a

3.2. Environnement logiciel de simulation:

Notre simulation a été réalisée dans l'environnement logiciel suivant:

- Système d'exploitation: Microsoft Windows Seven
- Le simulateur OMNet++ 4.2.
- L'IDE Microsoft Visual Studio 2010: environnement de développement intégré supportant le développement en C++.
- La librairie Mobility Framework dans MiXim: C'est une librairie qui étend OMNet++ pour pouvoir y simuler les réseaux sans fils mobiles, les réseaux Ad Hoc, et les réseaux de capteurs.
- Inet Framework pour assurer la communication en mode 802.11p.
- SUMO 0.15.0 simulation de trafic routier grâce à MOVE qui adopte un modèle de mobilité microscopique, qui est couplé avec OMNeT++ via une connexion TCP qui est utilisée pour transférer des instructions.

4. Simulation:

4.1. Etapes de simulations avec OMNet++:

1) Implémenter les nœuds: Modifier la structure de nœuds déjà existants dans OMNet++ pour y intégrer une couche applicative implémentée selon le CDP.

2) Définir une topologie: Regrouper les nœuds dans un réseau selon une topologie bien donnée grâce aux fichiers .NED d'OMNet++.

3) Définir un modèle de mobilité: Choisir l'un des modèles de mobilité offert ici. Nous avons opté pour SUMO.

4) Créer un scénario: regrouper les éléments suivants pour créer un scénario plausible et voir l'échange des données.

5) Simuler un protocole: Utiliser le scénario créé pour simuler un protocole.

6) Évaluer les performances: Recueillir les données fournies par la simulation et les comparer avec ceux d'autres protocoles (Ici MDHP de Mobeyes).

4.2. *Choix des critères d'évaluation:*

Pour notre simulation nous avons opté pour deux routes bidirectionnelles. Chaque route est composée de deux voies, de 10 km de longueur, et les deux routes se croisent dans une intersection au milieu, les véhicules sont distribuées uniformément sur la route avec un taux N véhicules/Km/Voie que l'on fixe et on varie suivant nos scénarios, un véhicule peut dépasser un autre, et la distance de sécurité est de 20 mètres.

Les véhicules démarrent à 0 mètre, nous avons deux type de véhicules: les véhicules agents et les véhicules privés. La vitesse moyenne de 15 m/s, nous avons choisi une portée de transmission R de 200 mètres, pour se rapprocher de l'environnement de simulation de MDHP.

Vingt cinq 25 résumés sont générés au niveau de chaque véhicule particulier avec les filtres de bloom correspondants. La durée de simulation est 1000 s.la taille des paquets est fixée à valeur élevée 1MO afin d'émuler le paquet volumineux comme c'est le

cas souvent dans les VSNs (ce qui peut être le cas pour les applications de sécurit ou Multimédia). En outre, nous supposons que la taille du tampon(Buffer) est illimitée sur chaque nœud véhicule.

Nous avons opté pour une connexion IEEE 802.11p au niveau de la couche MAC, 20 Mbps de bande passante. La capture d'information se fait à: 0 mètres, le tableau suivant résume nos paramètres de simulation.

Nombre d'itérations (simulations)	10
Rayon de transmission	200 m
Durée de connectivité dans une rencontre	1min à moyenne vitesse, 5min à petite vitesse
Capacité de la bande passante	20 Mbps (27 Mbps théorique)
Taille du paquet	1 Mo
Vitesse moyenne	15 m/s
Variation de Vitesse	5 m/s
Distance de sécurité	20 m
Couche MAC	IEEE 802.11 P
Durée de simulation	1000s
TTL	5 s
Durée rencontre	30s

Tableau 4.3: Paramètres de simulation

Chaque simulation est répétée 10 fois (10 exécutions pour chaque simulation) dans le but d'avoir des résultats fiables. Nous verrons dans ce qui va suivre les mesures de performance les plus pertinents. La figure 4.7 et 4.8 Représentent un aperçu de la route de simulation utilisée, prise avec SUMO[7].

[7] Les véhicules en mauve sont les agents, ceux en jeune sont les véhicules particuliers.

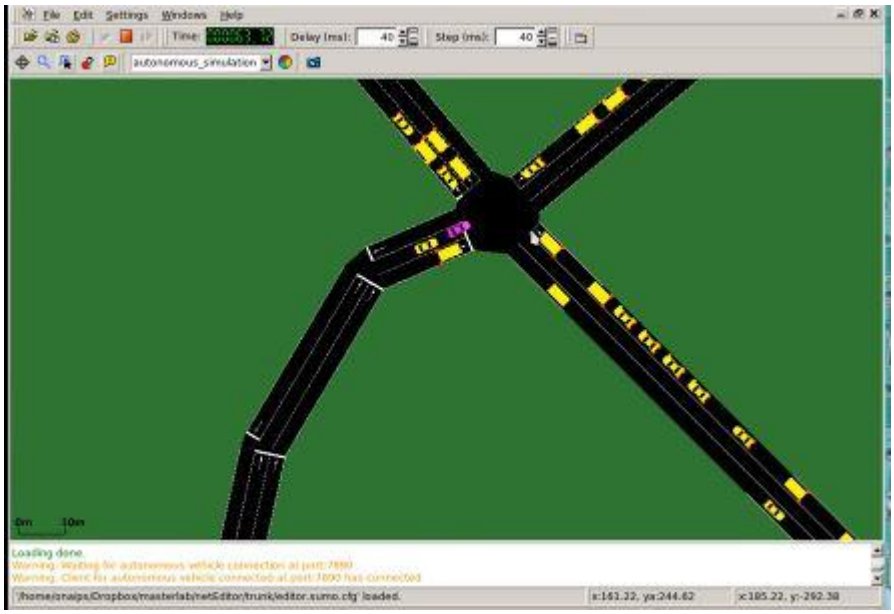

Figure 4.7: Aperçu zoomé de la route de simulation

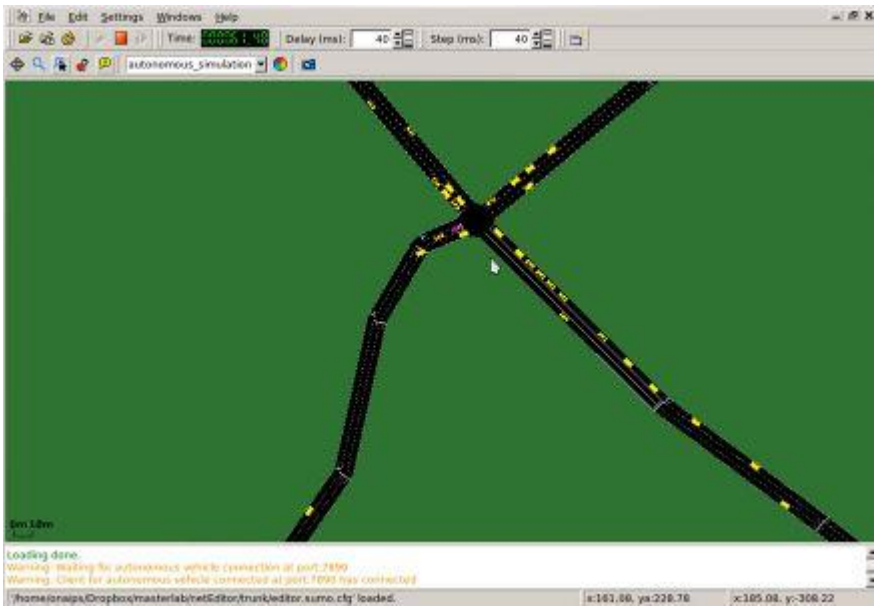

Figure 4.8: Aperçu de la route de simulation

5. Mesures de simulation:

Le but de ces mesures est de valider les performances de notre protocole qui tire son avantage dans l'utilisation du mécanisme de collaboration entre les véhicules agents dans la dissémination et l'accès à la donnée. Il est intéressant d'évaluer CDP suivant les critères de performances suivants:

- Mesure le temps de collecte suivant le nombre de véhicules.
- Mesure du taux de résumé collecté par rapport à la densité des agents.
- Mesure du nombre de message diffusé dans le réseau.

Nous comparons notre protocole avec le protocole de diffusion /collecte des données MDHP de MobEyes que nous avons amélioré. Ce protocole offre un bon taux de livraison de messages, au prix d'une surcharge élevée [LMG06].

5.1. Temps de collecte suivant le nombre de vehicule:

Nous avons fait varier le nombre de véhicules présents dans le réseau, sachant que ici le nombre de véhicules agents représente 5% du nombre des véhicules, exemple pour 100 véhicule présents dans le réseau, 95 sont privés et 5 sont des agents, le but est de mesurer le délai de collecte d'au moins 98% des résumés par au moins un agent.

Dans la figure 4.9. On remarque que le temps de collecte diminue au fur et à mesure que le nombre d'agents augmente ce qui est expliqué par la stratégie épidémique via la dissémination passive, plus la population augmente plus la stratégie épidémique augmente plus nos agents ont de chance de récupérer 98% de l'information, en moins de temps, comparé à MDHP, CDP affiche des temps de collecte meilleur, pour la simple raison que les rencontres se basent sur la technique de tri qui remplit la base de donnée des agents plus rapidement que la technique de collecte non optimisée chez MobEyes.

La stratégie épidémique entre agents joue un grand rôle dans ce cadre vue qu'un agent qui aurait collecté une bonne partie de l'information pourra se synchroniser avec un autre agent dés le début de la rencontre. Et, de ce fait le premier à recevoir le paquet de données, a la possibilité de compléter ses 98% plus rapidement.

Sachant que le premier véhicule agent qui atteint les 98% est considéré comme celui qui réussit à réunir tous les résumés dans sa mémoire locale, et pourra répondre à la requête d'une source (exemple autorité).

La synchronisation entre les agents va de paires. En effet, en augmentant le nombre d'agents, on augmente le nombre de chances que l'épidémie atteint les agents plus rapidement, ceci dit, un nombre très important d'agents (à partir d'un certain nombre) ne rapporte pas énormément de gain dans le délai et les délais tendent à se stabiliser puisque la densité des nœuds fait en sorte que dans chaque rencontre il y ait au moins un à deux agents avec et un nombre plus important de véhicules qui servira à compléter la base de donnée du VAC.

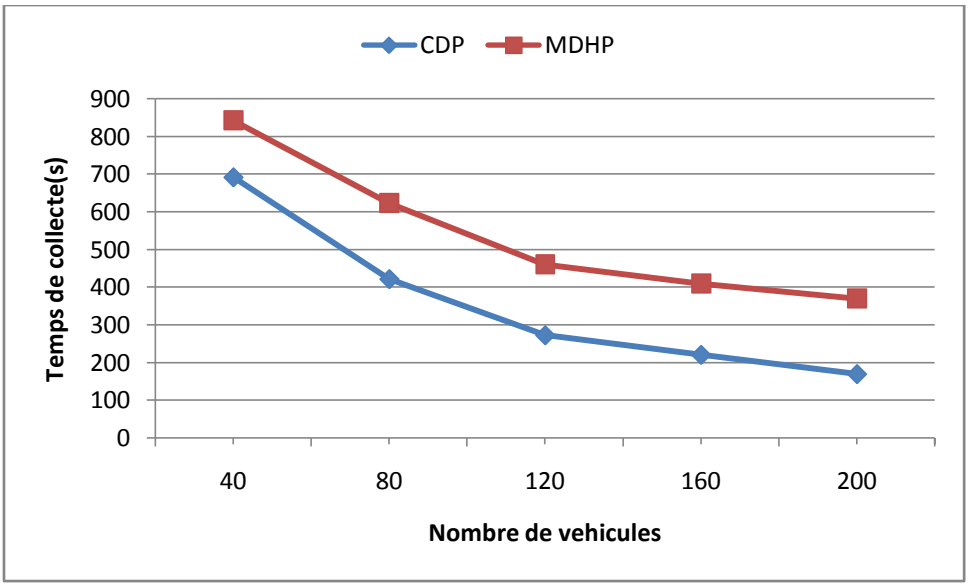

Figure 4.9: Temps de collecte de résumés en fonction du nombre de véhicules.

5.2. Taux de résumé collecté par rapport à la densité des agents:

En fixant le temps de collecte pour l'exécution de cette simulation dans l'algorithme CDP à T= 1000s, pour une population de véhicules privés N= 100, le taux de collecte de résumés est proportionnel au nombre d'agents. Ce taux est calculé par rapport au nombre de la somme des résumés collectés par les véhicules agents / le nombre de résumés produits par tous les nœuds durant cette période. Il claire d'après le graphe montré dans la figure 4.10 que CDP a un rendement meilleur lorsque ne nombre d'agents augmente, nous expliquons ceci par le fait que plus le nombre de véhicules agents augmente, plus il ya de possibilité de rencontre, ce qui implique que ces agents échangeront leurs résumés grâce aux messages de synchronisation plus rapidement, contrairement à MDHP où il ya aucun échange ou communication entre les agents. C'est dû aussià la notion de priorité que nous offre la technique du tri, car le premier véhicule qui nous répond est toujours celui qui nous apprend le plus ; c'est-à-dire, qu'il enrichie le plus la base de données locale.

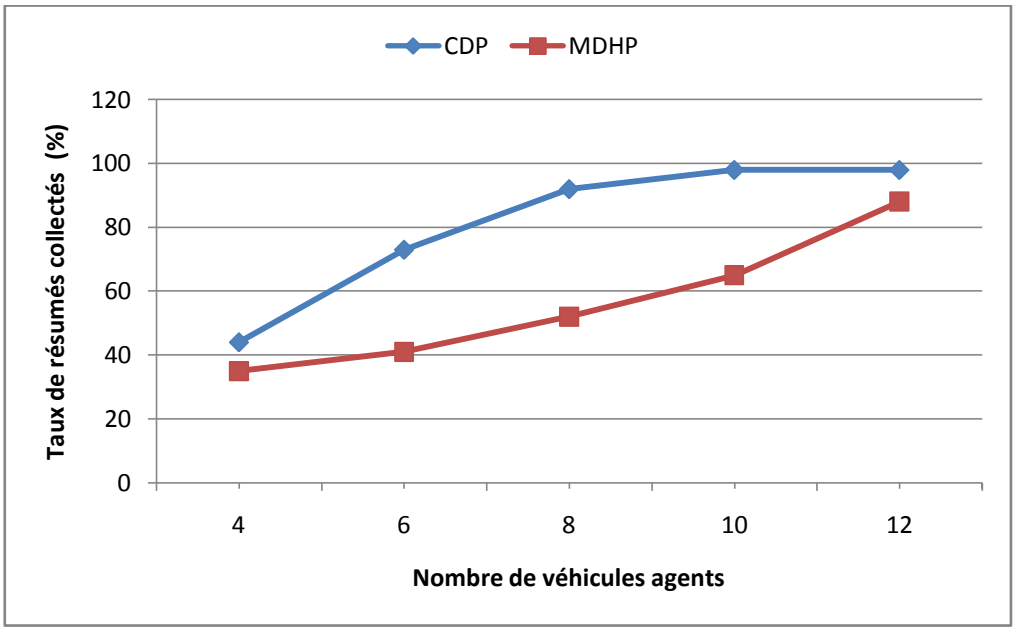

Figure 4.10: Taux de collecte de résumés en fonction du nombre de véhicules agents.

5.3. Nombre de messages diffusés dans le réseau:

La figure 4.11 présente le nombre de messages diffusés dans le réseau pendant toute la durée de simulation c'est-à-dire 1000 s, et pour une vitesse moyenne de 15m/s (c'est-à-dire presque 55km/h). Nous faisons varier le nombre de véhicules du réseau, sachant qu'ici le nombre de véhicules agents représente 5% du nombre de véhicules, ceci dans le but de mesurer la surcharge réseau que provoque CDP et de comparer les résultats avec ceux MDHP, la figure 4.11 montre que plus la densité de véhicule augmente plus la trafic overhead augmente dans les deux cas, sauf que dans le cas de MDHP il monte en exponentiel comparant à CDP, nous expliquons cela par l'optimisation introduite au niveau de la collecte, puisque un agent fait d'abord un tri (par différence de filtres de bloom) avant d'envoyer sa requête de collecte. Et, comme la premier véhicule à répondre est celui qui complétera le plus les résumes manquant a notre agent VAC, l'agent qui a déjà la base de donnée la plus complète et met à jour plus rapidement sa base de donnée et arrête la diffusion de sa requête de collecte vers les autres véhicules présents dans la rencontre. Donc moins de paquet à envoyer. Nous remarquons d'ailleurs une certaine stabilité vers la fin du graphe (figure 4.11). Dans ce cas, le nombre de paquets est nettement réduit avec une taille de paquet légèrement plus grande.

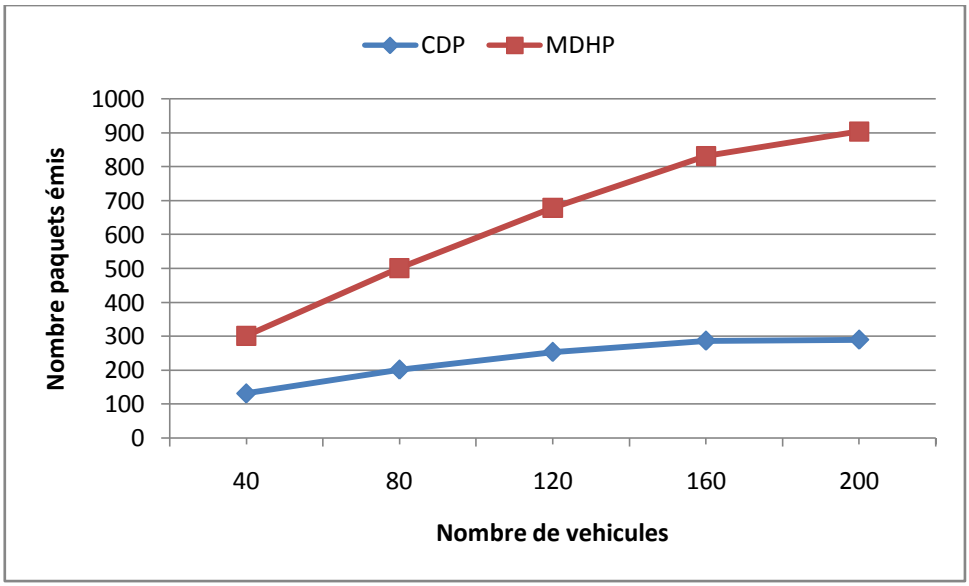

Figure 4. 11: Nombre de paquets émis en fonction du nombre de Véhicules agents

5.4. Nombre de résumés collectés par rapport a la vitesse des vehicules:

Dans cette figure 4.12 nous avons mesuré le nombre de résumés collectés en faisant varier la vitesse des véhicules dans le temps, le nombre de véhicules privés qui diffusent des résumés est de 100 pour 5 véhicules agents. Les résultats sont très concluant puisque, il est claire que plus nous augmentons la vitesse plus le nombre de résumés collectés augmente. Car plus les véhicules se déplacent rapidement plus ils ont de chance de rentrer dans de nouvelles rencontres, donc plus de possibilité de diffusion et de collecte d'informations. Mais aussi, plus la vitesse des véhicules agents augmentent plus vite ils échangeront les résumes manquants en accélérant le processus de synchronisation.

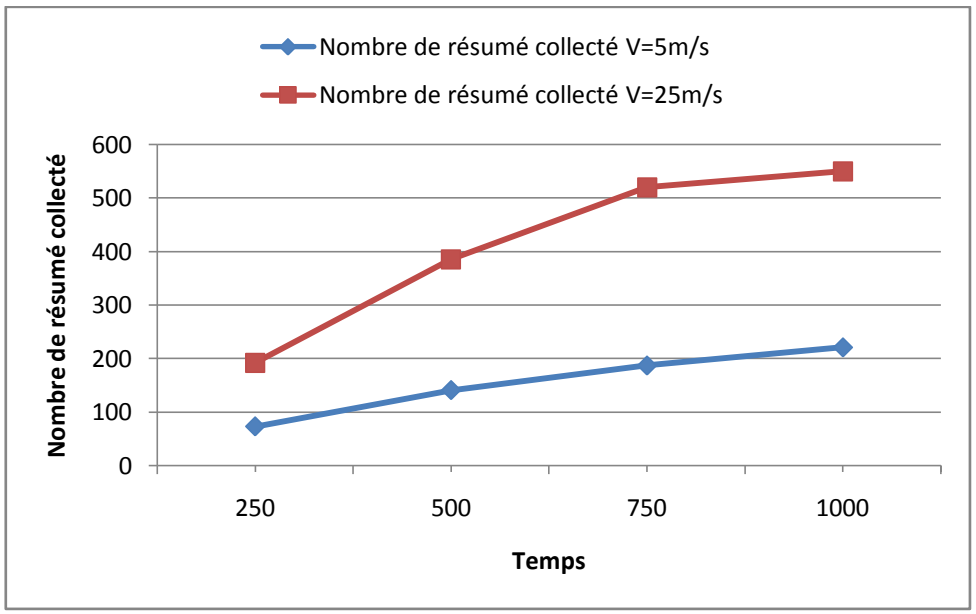

Figure 4.12: Nombre de résumés émis en fonction du nombre de véhicules agents

6. Conclusion

OMNET++ se trouve être un simulateur de réseau de capteurs véhiculaires très puisant. Il est le mieux adapté pour simuler un VSNs même s'il nécessite l'installation de beaucoup de Framework. Nous avons pu utiliser OMNET++ dans sa version 4.2 pour réaliser nos simulations.

Les résultats obtenus ont démontré l'efficacité de notre algorithme par rapport au protocole MDHP. Puisque que CDP améliore considérablement le temps de collecte et diminue la surcharge du réseau.

La comparaison de notre protocole avec le protocole MobEyes montre les avantages de la collaboration entre les agents de polices. En effet, cette dernière permet effectivement d'optimiser les performances du protocole.

Un autre aspect très important est bien le tri des véhicule par ordre d'importance des informations qu'ils détiennent. Plus les informations contenues par deux véhicules sont différents plus ils ont une chance d'échanger leurs informations et de cette manière nous évitons l'envoi multiple de la même information et nous diminuons l'overhead.

Conclusion

Les réseaux de capteurs véhiculaires présentent un nouveau type de réseau qui allie la technologie des réseaux de capteurs sans fil et celle des réseaux véhiculaires. En considérant le nombre d'applications possibles sur chacun de ces types de réseau, nous pouvons imaginer de large éventail d'applications réalisables sur ces environnements.

La dissémination de données a pour objectif de propager une donnée vers tous les nœuds présentant un intérêt pour ces informations. Cette diffusion intelligente vise à atteindre principalement les nœuds concernés en prêtant beaucoup d'attention à plusieurs critères de qualité de service tels que la fiabilité, la réduction de l'overhead et le temps de propagation.

Notre état de l'art sur la dissémination dans les réseaux de capteurs véhiculaires nous a permis de mettre en évidence les principales caractéristiques de ces réseaux. Par la suite, nous avons étudié les principaux concepts liés aux VSNs. Notre choix s'est porté sur un système pionnier qui est MobEyes car il offre une solution décentralisée Ad hoc pour les VSNs avec une latence acceptable mais avec une surcharge réseau plus élevée.

Le protocole MDHP de MobEyes pour la Diffusion/Collecte de données exploite la mobilité des véhicules équipés de caméras vidéo et une variété de capteurs. Il permet la détection des événements, le traitement et le filtrage des données capturées.

Néanmoins, la solution de MobEyes, ne prend pas en compte la collecte inutile et récurrente des mêmes résumés produits par les véhicules particuliers et collectés par les véhicules agents, causant une hausse de la surcharge du réseau pouvant mener à une tempête de diffusion.

Notre contribution a permis de proposer un protocole qui améliore les performances MobEyes pour la dissémination et accès à la donnée. En effet, CDP ou « Collaborative Dissémination Protocole » exploite les connexions occasionnelles entre véhicules agents de police au même titre que les connexions entre les véhicules privés.

Notre protocole CDP améliore les performances de MobEyes grâce à une technique de clustering appelé *rencontre,* qui se base sur le fait de trier les nœuds par ordre de pertinence de leur information avant de lancer la phase de communication.

Cette technique nous a permis d'éviter les transmissions redondantes et d'assurer une rapidité dans le processus de diffusion.

Les évaluations effectuées à l'aide du simulateur Omnet++ démontrent l'efficacité du schéma proposé. Les mesures montrent l'avantage obtenu en temps de collecte, en taux de réception des données et en stabilisation de la surcharge réseau.

Ceci dit, l'aspect sécurité dans les VSNs n'a pas encore été traité. Or, dans le cas d'applications critiques, il est impératif qu'un protocole puisse assurer un échange d'informations sécurisées en introduisant des techniques de sécurité. Cette problématique est une perspective à considérer. D'autres perspectives plus immédiates peuvent être envisagées telles que l'extension du protocole par la prise en compte des environnements avec capteurs intégrés sur la route, et l'intégration de la notion de pertinence lors de la collecte d'information.

Bibliographies

[ABT04] Adam Dunkels, Björn Grönvall, Thiemo Voigt, « Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors », 29th Annual IEEE International Conference on Local Computer Networks, 2004.

[AkK04] Akyildiz, I. F. and Kasimoglu, I. H., "Wireless sensor and actor networks: Research challenges," Ad Hoc Networks (Elsevier), vol. 2 N:4, October 2004.

[STG02] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. B. Srivastava. Optimizi sensor networks in the energy-latency-density design space. IEEE Transactions on Mobile Computing, 1, Jan.-Mar. 2002.

[HCB00] W. Heinzelmann, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *IEEE HICSS*, Jan. 2000.

[LRS02] S. Lindsey, C. Raghavendra, and K. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13, Sept. 2002.

[MAA01] A. Manjeshwar and D. Agrawal. Teen: A routing protocol for enhanced efficiency in wireless sensor networks. In *ACM/IEEE Workshop Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Apr. 2001.

[MAA02] A. Manjeshwar and D. P. Agrawal. An efficient sensor network routing protocol (apteen) with comprehensive information retrieval. In *ACM/IEEE Workshop Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Apr. 2002.

[IGE00] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM MOBICOM'00*, 2000.

[SGA00] K. Sohrabi, J. Gao, V. Ailawadhi, G. J., and Pottie. Protocols for self-organization of a wireless sensor network.*IEEE Personal Communications*, 7(5), Oct. 2000.

[POK00] G. PottieandW. Kaiser. "Wireless integrated networks sensors (wins): principles and approach". *Communication of the ACM*, 43(5), May 2000.

[XMS04] Q. Xu, T. Mak, J. Ko, and R. Sengupta. Vehicle-to-vehicle safety messaging in DSRC.In *ACM VANET*, Oct. 2004.

[NDP05] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Y. Sanadidi. Co-operative Downloading in Vehicular Ad-Hoc Wireless Networks. In *IEEE WONS*, Jan. 2005.

[NTD06] A. Nandan, S. Tewari, S. Das, G. Pau, M. Gerla, and L. Kleinrock. AdTorrent: Delivering Location Cognizant Advertisements to Car Networks. In *IFIP WONS*, Jan. 2006.

[LPA06] U. Lee, J.-S. Park, E. Amir, and M. Gerla. FleaNet: A Virtual Market Place on Vehicular Networks. In *IEEE V2VCOM*, Jul. 2006.

[CGM06] M. Caliskan, D. Graupner, and M. Mauve. Decentralized discovery of free parking places .In *ACM VANET*, Sept. 2006.

[STC06] D. Sormani, G. Turconi, P. Costa, D. Frey, M. Migliavacca, and L. Mottola. Towards lightweight information dissemination in inter-vehicular networks .In *ACM VANET*, Sept. 2006.

[BEH04] J. Blum, A. Eskandarian and L. Hoffmman, Challenges of intervehicle ad hoc networks, *IEEE Trans. Intelligent Transportation Systems*, December 2004.

[NBG06] V. Naumov, R. Baumann, and T. Gross. An Evaluation of Inter-Vehicle Ad Hoc Networks Based on Realistic Vehicular Traces. In *ACM MOBIHOC*, May 2006.

[SAJ04] A. K. Saha and D. B. Johnson.Modeling Mobility for Vehicular Ad Hoc Networks.In *ACM VANET'04*, October 2004.

[BMP06] F.Bonomi, M.Mitzenmacher, R.Panigrahy, S.Singh, G.Varghese.Beyond bloom filters: from approximate membership checks to approximate state machines. Protocols for Computer Communications, Pisa, Italy, September 11-15, 2006.

[TJH04] M. Torrent-Moreno, D. Jiang, and H. Hartenstein. Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks. In *ACM VANET*, Oct. 2004.

[XMS04] Q. Xu, T. Mak, J. Ko, and R. Sengupta. Vehicle-to-vehicle safety messaging in DSRC.In *ACM VANET*, Oct. 2004.

[KEO04] G. Korkmaz, E. Ekici, F. Ozguner, and U. Ozguner. Urban Multi-Hop Broadcast Protocols for Inter-Vehicle Communication Systems. In *ACM VANET*, Oct. 2004.

[CKV01] Z. D. Chen, H. Kung, and D. Vlah.Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In *ACM MOBIHOC*, Oct. 2001.

[BGJ06] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *IEEE INFOCOM*, Apr. 2006.

[FGH04] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter. MDDV: a Mobility-centric Data Dissemination Algorithm for Vehicular Networks. In *ACM VANET*, Oct. 2004.

[ZHC06] J. Zhao and G. Cao. VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks. In *IEEE INFOCOM*, Apr. 2006.

[PBR99] C. E. Perkins and E. M. Belding-Royer. Ad-hoc On-Demand Distance Vector Routing. In *ACM WMCSA*, Feb. 1999.

[KAK00] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. InACMMOBICOM'00, Aug.2000.

[DRA03] IEEE 802.11e/D4.4, Draft Supplement to Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), June 2003.

[NTC99] S.-Y. Ni, Y.-C. Tseng, Y.-S.Chen, and J.-P. Sheu.The broadcast storm problem in a mobile ad hoc network.In *ACM MOBICOM*, Aug. 1999.

[CHV01] Z. D. Chen, H. Kung, and D. Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In *ACM MOBIHOC*, Oct. 2001.

[LMZ06] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi. Efficient Data Harvesting in Mobile Sensor Platforms.In *IEEE PerSeNS*, Mar. 2006.

[BOF07] L. Bononi,M. Di Felice, "A Cross Layered MAC and Clustering Scheme for Efficient Broadcast in VANETs", IEEE MASS'07, October 2007,

[GUI04] E. Guizzo. "Network of traffic spies built into cars in Atlanta". IEEE Spectrum, April 2004.

[LMG06] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, A. Corradi, "Dissemination and Harvesting of Urban Data using Vehicular Sensing Platforms", submitted for publication,Juin 2006.

[GRG08] PIERRE-LUC GREGOIRE-GIRARD Communication inter-Véhicules et route-a-véhicule, apprentissage de la communication inter- véhicule, June2008.

[NEJ08] Mario Naval Escartín, Jiri Novak, INTRA VEHICULAR SENSOR ZIGBEE NETWORK December 2008.

[BGG02] Bakker D, Gilster DM, Gilster R 2002. Bluetooth end to end. Wiley, 2002.

[GMK07] Ghavami M, Michael L, Kohno R. Ultra wideband signals and systems in communication engineering. Wiley 2007.

[BIL07] Bilstrup K (2007) Vehicular communication standards.Telematics valley conference, Göteborg, Sweden, Nov 2007.

[SUS07] Strandén L, Uhlemann E, Strom EG (2008) State of the art survey of wireless vehicular communication projects. 15th world congress on Intelligent Transportation Systems (ITS), New York, Nov 2008.

[NEK05] Nekovee M Sensor networks on the road: the promises and challenges of vehicular adhoc networks and vehicular grids. In: Proceedings of the workshop on ubiquitous computing and e-Research, Edinburgh, UK, May 2005.

[DCT10] Denso Corporation Technology. Pre-crash Safety System. January 2010.

[STC06] D. Sormani, G. Turconi, P. Costa, D. Frey, M. Migliavacca, and L. Mottola. Towards lightweight information dissemination in inter-vehicular networks . In ACM VANET, Sept. 2006.

[GKK03] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, et S. Seshan, "IrisNet: An Architecture for a Worldwide Sensor Web", 2003.

[GDS05] S.Libhart "GDS: gunshots detection system ," 2005.

[DLB05] L. Dlagnekovet S. Belongie, "Recognizing Cars", 2005.

[NET10] http://netlab.cs.ucla.edu/cgi-bin/usemod10/wiki.cgi?MobEyes, 2010.

[JDE08] Jiang, D. AL, Delgrossi. 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments, Vehicular Technology Conference, 2008. May 2008.

[KSW08] A .KOPKE, M.SWIGULSKI, K.WESSEL , « Simulating wireless and mobile networks in omnet++ the mixim vision », *Proc. Int. Workshop on OMNeT++ collocated with SIMUTools*, Mars 2008.

[DSH03] W.DRYTKIEWICZ, S.SROKA, V.HANDZISKI, « A mobility framework for OMNET++ », *In 3rd International OMNeT++ Workshop, at Budapest University of Technology and Economics, Department of Telecommunications Budapest, Hungary*, 2003.

[KSW11] A. Köpke, M. Swigulski, K. Wessel, D.Willkomm,"Simulating Wireless and Mobile Networks in OMNeT++ the MiXiM Vision", May 2011

[FOS011] IEEE 1609 Family of Standards for Wireless Access in Vehicular Environments; de IEEE Standards.

Références URL:

[OMN12] "OMNeT++: discrete event simulation environment"http://omnetpp.org

[NS212] "Ns2: discrete event simulator targeted at networking research"http://www.isi.edu/nsnam/ns/

[MFR10] Mobility Framework for OMNeT++" http://mobility-fw.sourceforge.net/hp/index.html

[MIX12] "MiXiM mixed simulator" http://sourceforge.net/apps/trac/mixim/

[SUM12] "SUMO Simulation of Urban Mobility". http://sumo.sourceforge.net

[CAR11] MIT'sCarTel Central. http://cartel.csail.mit.edu/.

[VEI12] "Vehicles in Network Simulation". http://veins.car2x.org/tutorial/

[SAF11] "SafeSpot" (Cooperative vehicles and road infrastructure for road safety), http://www.safespot-eu.org/

[WOV10] "Watch-Over", http://www.watchover-eu.org/

[CVI11] CVIS (CooperativeVehicle-Infrastructure Systems), http://www.cvisproject.org/

[ARG09] "ARGO - Global Ocean Sensor Network". www.argo.ucsd.edu.

[CRO10] "Crossbow MICA2 Mote Specifications". www.xbow.com.

[UMA11] "UMass Diesel Net". http://prisms.cs.umass.edu/dome/.

[C2C10] "Car-2-Car Consortium". http://www.car-to-car.org/

[ZIN12] "Zone industrie capteur CO2 " http://www.zoneindustrie.com/

[HEA12] "Hydrogen EFIE module pour automobile" http://ebri.ca/

ANNEXES

Installation OMNET++:

D'abord il faut télécharger la version adéquate d'OMNet++ *à* partir du site *www.omnetpp.org*. Après l'installation du Visual C++ dans la machine on procède comme suit: Double clic sur le programme d'installation *omnetpp-4.2.exe,* cette fenêtre va apparaître:

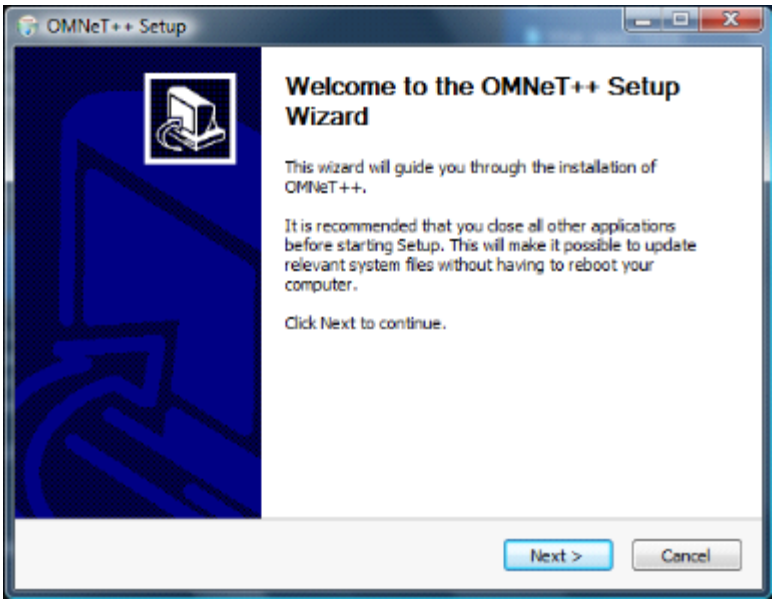

Figure4.3.: *Fenêtre d'installation*

Après l'installation un nouveau dossier sera créé avec le nom omnetpp-*4.2*. Ensuite, il faut ajouter le chemin d'installation *C:\OMNeT++\bin* au variable d'environnement

Installation Mobility Framework:

1. Télécharger le Mobility Framework (*mobility-fw2.0p3.zip*) code source code et décompresser l'archive dans C:\

2. modifier le variable OMNETPP_ROOT dans mkmk.cmd pour pointer sur le chemin d'installation OMNeT++, dans notre cas C:\.

3. Exécuter le script mkmk.cmd dans l'invite de commande: ./mkmk.cmd

4. Compiler les librairies *core, contrib et networks*:

nmake /f Makefile.vc core

nmake /f Makefile.vc contrib

nmake /f Makefile.vc networks

INET:

- **La couche PHY**

La couche physique est la partie essentielle d'un nœud sans fil. Il est responsable de l'envoi et la réception de message, détection de collision, et calcul d'erreur sur les bits. La couche physique est divisée en trois parties, qui sont décrites en détail dans les sous-sections suivantes:

- PhyLayer fournit les interfaces de la couche MAC et de la couche physique des autres noeuds.
- AnalogueModels sont responsables pour la simulation de l'atténuation d'un signal reçu.
- Le Decider est responsable de l'évaluation et de démodulation des messages reçus.

En ce qui concerne notre projet, on a besoin de la couche physique IEEE 802.11p qui est déjà implémentée dans OMNET++.

- **La couche MAC**

La protocole MAC IEEE 802.11p est implémentée dans la librairie INET, elle supporte le mécanisme RTS/CTS. Il n'y a que l'algorithme DCF implémenté, le PCF n'est pas implémenté. Le débit de transmission physique est constant pendant la simulation.Le simulateur OMNeT++ contient deux classes de bases pour former la couche MAC:

- **BaseMACLayer** pour encapsuler et décapsuler les paquets seulement.
- **EyesMACLayer** fournie un ensemble de fonction dont l'importante est de renvoyer les informations concernant la couche MAC d'un noeud.

Dans notre cas la couche MAC IEEE 802.11p est implémentée dans la librairie INET en utilisant RTS/CTS. Il n'y a que l'algorithme DCF implémenté, le PCF n'est pas implémenté. Le débit de transmission physique est constant pendant la simulation. Aucun des algorithmes d'adaptation du débit de transmission physique n'est implémenté. Les paquets multipoints au niveau MAC sont envoyés de la même manière que les paquets broadcast. Le filtrage de ces paquets se fait au niveau IP.

- **La couche IP**

L'Internet Protocol Suite (IPSuite) fournit les protocoles IPv4, TCP et UDP dans les modèles de simulations OMNeT + +. l'ensemble IPsuite a été repris par les concepteurs. Avec également plus de documentation pour le paquet, ce qui rend plus facile à utiliser et faire des ajustements pour le modèle. Dans le même temps, un modèle l'IPv6 a été créé. Mais le routage IP est implémenté de manière statique. Alors, l'utilisateur doit configurer les adresses IP et les adresses de groupes multicast dans les tables de routages avant la simulation. C'est au niveau IP que se passe le filtrage des paquets multicast. Le routage multipoint au niveau IP est déjà implémenté mais de manière statique. C'est à dire que l'utilisateur doit configurer les adresses IP et les adresses de groupes unicast dans les tables de routages avant la simulation. C'est au niveau IP que se passe le filtrage des paquets multicast.

- **Le protocole UDP**

Le protocole UDP est implémenté dans la librairie INET. Elle est présente au niveau transport et elle est très utilisable dans la simulation car son implémentation est simple et elle est rapide par rapport aux autres couches de même niveau.

- **Le protocole TCP**

Dans la librairie INET, ce protocole est présent dans la couche de transport tout comme UDP et RTP et elle est implémentée à l'aide des sockets.

- **La couche Application**

Elle présente la couche supérieure. De nombreuses applications sont implémentées dans la librairie INET comme TCPApp, PingApp, UDPApp.