N° d'ordre : ....

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENNE
FACULTÉ D'ÉLECTRONIQUE ET D'INFORMATIQUE

جامعة هواري بومدين للعلوم والتكنولوجيا
USTHB

THESE DE DOCTORAT EN SCIENCES

Présentée pour l'obtention du diplôme de :

DOCTEUR

En **INFORMATIQUE**

Option : Informatique Mobile

Par :

Ahmed ALIOUA

Thème

Intégration du Software-Defined Networking (SDN) dans les réseaux de véhicules (VANETs)

Soutenue publiquement, le 21 Février 2019, devant le jury composé de :

M.	Youcef Aklouf	Professeur à l'USTHB	Président
Mme.	Samira Moussaoui	Professeur, à l'USTHB	Directrice de thèse
M.	Sidi-Mohammed Senouci	Professeur, à l'Univ de Bourgogne	Co-Directeur de thèse
M.	Walid-Khaled Hidouci	Professeur, à l'ESI	Examinateur
Mme.	Nadia Nouali	Directrice de recherche, à CERIST	Examinatrice
M.	Mohamed Guerroumi	Maître de conférences A, à l'USTHB	Examinateur

À mes chers parents
À mon épouse Amina
À tous mes frères et sœurs
À ma petite princesse Rinad

Remerciements

Je commence par exprimer mes remerciements à ma directrice de thèse, Madame Samira Moussaoui, Professeur à l'USTHB, pour toutes ces années de soutien, pour ses précieux conseils scientifiques et humains, ainsi que pour ses encouragements, sa patience et la motivation qu'elle m'a toujours apportée.

Je tiens à exprimer ma plus vive gratitude à mon co-directeur de thèse, Monsieur Sidi-Mohammed Senouci, professeur à l'ISAT – université de Bourgogne Franche Comté, pour ses encouragements, sa pédagogie et ses précieux conseils qu'ils m'ont permis de mener à bien ce travail. Je le remercie également de m'avoir accueilli au sein de son laboratoire DRIVE, de m'avoir consacré tout ce temps et toute cette énergie ainsi que pour toute la disponibilité dont il a su faire preuve à mon égard. J'exprime ici ma profonde reconnaissance à son égard et l'estime respectueuse que je lui porte.

Je souhaite aussi exprimer ma gratitude à Monsieur Youcef Aklouf, Professeur à l'USTHB, pour l'honneur qu'il me fait en acceptant de présider le jury de ma thèse. Mes remerciements vont aussi à Monsieur Walid-Khaled Hidouci, Professeur à l'ESI, pour avoir accepté d'examiner ce travail et participer au jury. Je tiens également à remercier Madame Nadia Nouali, Directrice de recherche au CERIST, pour avoir accepté d'examiner ce travail et faire partie du jury de ma thèse. Enfin, je voudrais remercier Monsieur Mohamed Guerroumi, Maître de conférences A à l'USTHB, pour avoir accepté d'examiner ce travail et participer au jury.

Au cours de cette thèse, j'ai bénéficié d'un stage de 8 mois dans le cadre du programme national exceptionnel (PNE) octroyée par le ministère d'enseignement supérieur et de la recherche scientifique de l'Algérie. J'ai également bénéficié d'un financement octroyé par le laboratoire Drive de l'ISAT, université de Bourgogne Franche Comté pour présenter mes travaux de conférence. Enfin, la faculté NTIC de l'université Abdelhamid Mehri - Constantine 2 m'a accordé un financement pour un stage d'une durée d'un mois au sein de laboratoire DRIVE pour mener à bien ma thèse. Pour tous ces organismes, je tiens à exprimer mes sincères remerciements.

Tout au long de ces années de thèse, j'ai eu l'occasion de rencontrer des personnes toutes aussi intéressantes les unes que les autres. À leur façon, ils ont tous soutenu et contribué à mon apprentissage. Je suis reconnaissant envers chacun de ces personnes.

Je termine par les personnes que je ne saurais jamais remercier assez. À mon père, ma mère, et tous mes frères et sœurs qui m'ont soutenue et encouragée à aller de l'avant. À tous mes amis. Enfin, aucun mot ne peut exprimer toute la reconnaissance et la gratitude que je dois à mon bien aimée épouse Amina et mon adorable petit ange Rinad. Je n'aurais jamais tenu le coup sans leur aide.

Résumé

Propulsés par les longs embouteillages et les nombreux accidents de la route, les réseaux de véhicules (*Vehicular Ad hoc NETwork*, VANET) ont émergé dans le but de rendre le voyage plus agréable, la route plus sûre et le système de transport plus efficace. De nos jours, les architectures des réseaux de véhicules souffrent de problèmes d'évolutivité, car il est difficile de déployer des services à une grande échelle. Ces architectures sont rigides, difficiles à gérer et souffrent d'un manque de flexibilité et d'adaptabilité à cause de l'hétérogénéité des technologies véhiculaires. Ces contraintes limitent les fonctionnalités du système, ralentissent la créativité et conduisent souvent à une sous-exploitation des ressources de réseau.

Au cours des dernières années, le paradigme émergent de l'architecture des réseaux de Software-Defined Networking (SDN), est devenu l'une des technologies les plus importantes pour la gestion des réseaux à grande échelle tels que les réseaux de véhicules. Le SDN est principalement basé sur une séparation physique entre le plan de contrôle et le plan de données et un contrôle et une intelligence logiquement centralisés dans un contrôleur logiciel. Récemment, plusieurs travaux ont montré que l'intégration du paradigme de SDN dans les VANETs permet d'apporter la flexibilité, la programmabilité et mieux supporter l'évolutivité.

Cette thèse étudie les manières permettant de profiter des points forts du SDN pour faciliter la gestion et améliorer les performances des prochaines générations des réseaux de véhicules. Dans une première contribution, nous proposons une nouvelle architecture semi-centralisée qui tire profit des avantages de différents concepts de réseaux à l'instar des réseaux cellulaires, le concept de *Cloud/fog computing* et la technique de *clustering* pour faciliter une intégration efficace du paradigme de SDN dans les réseaux de véhicules hétérogènes, appelée *Cluster-based Software-Defined Heterogeneous Vehicular Networks* (CSDHVN). Le mode de contrôle dans cette architecture est semi-centralisé et il est basé sur une hiérarchie de multiples contrôleurs déployés à la périphérie de réseau. Le principal but est de combler le manque d'architecture basée sur SDN complète qui fonctionne non seulement dans les zones couvertes par l'infrastructure fixe véhiculaire, mais aussi dans les zones non couvertes. CSDHVN utilise un mécanisme de récupération de secours efficace pour lutter contre les éventuelles pannes des contrôleurs SDN. Entre autres, la distance lointaine de déploiement de contrôleurs SDN dans de tels larges réseaux peut générer un important effet négatif sur les performances du système, en particulier sur le délai.

Dans une seconde contribution, nous étendons la version distribuée multi-sauts de CSDHVN pour l'adapter à un cas d'utilisation relative à une mission de secours suite à un incident sur la route dans les environnements véhiculaires sans infrastructure. Nous équipons les véhicules de secours par des mini-drones afin de les assister dans l'exploration des tronçons inaccessibles de routes. Ensuite, nous concentrons nos efforts sur l'étude d'un traitement efficace des données collectées par le drone. Nous formulons le problème en utilisant un jeu séquentiel comme un problème de prise de décision en matière de délestage (*offloading*) de calcul des tâches du drone à son véhicule de secours et/ou le partage

du calcul des tâches entre le véhicule de secours et ses voisins. Nous proposons deux algorithmes distribués pour résoudre le problème d'optimisation qui consiste à trouver le meilleur compromis entre le délai et l'énergie de calcul. Notre politique de traitement des données basée sur SDN assure des performances de calcul efficace avec le meilleur équilibre entre le délai et l'énergie de calcul.

Enfin, nous projetons l'architecture CSDHVN sur un autre cas d'utilisation concernant l'incitation à la mise en cache dans le réseau Internet de véhicules. Nous considérons un scénario de mise en cache concurrentiel composé d'un seul fournisseur de service (*Content Provider*, CP) et de plusieurs opérateurs mobiles (*Mobile Network Operators*, MNOs). Le CP encourage les MNOs à stocker son contenu populaire dans leurs caches sur les stations de bases avec la plus grande popularité, tandis que les MNOs sont en concurrence permanente sur la quantité limitée de contenus populaires du CP. Nous utilisons un jeu de type *Stackelebrg* pour modéliser l'interaction entre le CP et les MNOs avec un sous-jeu non-coopératif pour formuler le conflit entre les MNOs. Ensuite, nous utilisons l'optimisation convexe pour étudier deux problèmes d'optimisation lorsque le CP vise à maximiser la satisfaction de ses utilisateurs, quand chaque MNO vise à maximiser son gain monétaire. La prise en considération d'un système de mise en cache concurrentiel peut améliorer de manière significative la qualité d'expériences des utilisateurs et contribuer à la diminution du coût de mise en cache.

En guise de conclusion et étant donné les apports futuristes que promet le paradigme de SDN, nous croyons profondément que SDN est un choix prometteur pour combler les limitations des réseaux de véhicules et faciliter leurs gestion, specialement avec l'arrivée de la 5G dans les VANETs et le rapide développement de véhicule autonome.

Mots-clés *: Réseaux de véhicules hétérogènes, Internet de véhicules, Véhicules assistés par des drones, Software-Defined Networking, Théorie des jeux, Mise en cache, Traitement de données, Délestage de calcul.*

Abstract

Propelled by the long traffic congestions and high road accidents, Vehicular Ad-hoc Networks (VANETs) have emerged to make the traveling experience more pleasant, the road safer and transportation system more efficient. Nowadays, VANET architectures suffer from scalability issues since it is difficult to deploy services in a large-scale. These architectures are rigid, difficult to manage and suffer from a lack of flexibility and adaptability in control. These constraints limit system functionalities, slow down creativity and often lead to under-exploitation of network resources.

In the last few years, the emerging network architecture paradigm of Software-Defined Networking (SDN) has become one of the most important technologies to manage large-scale networks such as VANETs. SDN is mainly based on a physical separation between the control plane and the data plane and a logically centralized control and intelligence in a software controller. Several works have shown interest in the use of the promising SDN paradigm to address the undermentioned challenges of VANETs. SDN can be used to bring flexibility, programmability and scalability to nowadays vehicular networks.

In this thesis, we study how to benefit from the advantages of SDN to facilitate and enhance the performance of the next generations of vehicular networks. In a first contribution, we propose a new semi-centralized architecture which benefits from different network concepts like cellular networks, the concept of cloud/fog computing and the clustering technique to facilitate an efficient integration of the paradigm of SDN in the heterogeneous vehicle networks, called *Cluster-based Software-Defined Heterogeneous Vehicular Networks (CSDHVN)*. The control mode in this architecture is semi-centralized based on a hierarchy of multiple SDN controllers. The main purpose is to fill the gap that no SDN-based architecture that works in both the covered areas by the fixed infrastructure and the uncovered areas is proposed in the literature. Our CSDHVN architecture uses an efficient fallback recovery mechanism. Moreover, the far deployment of SDN controllers represents a significant negative impact on the system performance (delay).

In a second contribution, we extended the multi-hops distributed version of CSDHVN to a use case relating to a rescue mission resulted of road incident in vehicular infrastructure-less environments. We equipped rescue vehicles with mini-drones to assist them in the exploration of inaccessible road zones. Then, we focused our efforts on an efficient processing of the data collected by the drone. We have formulated the problem by using a sequential game as a decision-making problem of the offloading computation tasks from the drone to its rescue vehicle and/or the sharing computation tasks between the rescue vehicle and his neighbor's vehicles. We proposed two distributed algorithms to solve the optimization problem which consists of finding the best tradeoff between the computation delay and the drone energy. Our SDN-based data processing policy ensures efficient computing performance with the best balance between the delay and the energy.

Finally, we project the CSDHVN architecture on an incentive caching use case in the context of the Internet of vehicles. We considered a competitive caching scenario which

consists of a single content (CP) and multiple mobile network operators (MNO). The CP encourages the MNOs to store its popular content in their caches on small base stations with the highest popularity. We used a Stackelberg game to model the interaction between the CP and the MNOs with a non-cooperative sub-game to resolve the conflict between MNOs. We then used convex optimization to study two optimization problems when the CP aims to maximize its user's satisfaction when each MNO aims to maximize its monetary profit. Considering a competitive caching system can significantly improve the user quality of experience and help to lower the caching cost.

As a conclusion and given the futuristic benefits that offer the paradigm of SDN, we deeply believe that SDN is a promising choice to bridge the limitations of next generation vehicular networks and facilitate their management, especially with the soon arrival of 5G in VANETs and the fast development of autonomous vehicles.

Keywords : *Heterogeneous Vehicular Networks, Internet of Vehicles, Drone assisted vehicular networks, Software-Defined Networking, Game Theory, Caching, Data Processing, Offloading computation.*

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTES DES ACRONYMES

VANET	Vehicular Ad hoc NETwork
HetVNet	Heterogeneous Vehicular NETwork
IoV	Internet of Vehicles
SDN	Software-Defined Networking
UAV	Unnamed Aerial Vehicle
CP	Content Provider
MNO	Mobile Network Operator
CSDHVN	Cluster-based Software-Defined Heterogeneous Vehicular Networks
TIC	Technologies d'Information et de Communication
ITS	Intelligent transportation Systems
V2V	Vehicle to Vehicle communication
V2I	Vehicle to Infrastructure communication
OBU	On Board Units
RSU	Road Side Units
MANET	Mobile Ad-hoc NETwork
DSRC	Dedicated Short-Range Communication
WAVE	Wireless Access in Vehicular Environment
OFDM	Orthogonal Frequency Division Multiplexing
CALM	Communication Access for Land Mobiles
FCC	Federal Communication Commission
ETSI	European Telecommunications Standards Institute
CCH	Control Channel
SCH	Service Channels
MAC	Medium Acces Control
IEEE	Institute of Electrical and Electronics Engineers
QoS	Quality of Service
IoT	Internet of Thing
IP	Internet Protocol
V2H	Véhicule à Humain
V2C	Véhicule à Capteurs
WAN	Wide Area Networks
ONF	Open Networking Foundation
SDVN	Software-Defined Vehicular Network

I2Cloud	Infrastructure vers Cloud
dSDiVN	distributed Software Defined infrastructure-less Vehicular Network
SD-zone	Software-Defined zone
SD-domain	Software-Defined domain
TLS	Transport Layer Security
LDM	Local Dynamique Map
GDM	Global Dynamique Map
NS3	Network Simulator 3
SUMO	Simulation of Urban MObility
EV	Emergency Vehicle
CV	Controller Vehicle
QoL	Quality of Link
CPU	Central Processing Unit
EN	Equilibre de Nash
DOSC	Distributed Offloading Sharing Computation
LC	Local Computation
OC	Offloading Computation
SC	Shared Computation
SBS	Small Base Station
D2D	Device to Device
VR	Video Retailer
HSD-IoV	Hierarchical Software-Defined Internet of Vehicle
QoE	Quality of Experience
ES	Equilibre de Stackelberg

LISTE DES PUBLICATIONS

- **Revues internationales (2)**

  - **Ahmed Alioua**, Sidi-Mohammed Senouci, Hichem Sedjelmaci, Mohamed Ayoub Messous, Samira Moussaoui, "Efficient Data Processing in Software-Defined UAV-assisted Vehicular Networks : A Sequential Game Approach", International Journal of Wireless Personal Communications (Springer), vol. 101(1), Mai 2018. pp. 2255-2286 (Factor Impact : 1.2).
  - **Ahmed Alioua**, Sidi-Mohammed Senouci, Hichem Sedjelmaci, Samira Moussaoui, "Incentive Edge Caching in Software Defined Internet of Vehicules : A stackelberg Approch", International Journal of Communication Systems (Wiley), July 2018. (Factor Impact : 1.717).

- **Conférences internationales (5)**

  - **Ahmed Alioua**, Sidi-Mohammed Senouci, Samira Moussaoui, Hichem Sedjelmaci, Abdelwahab Boualouache, "Software-Defined Heterogeneous Vehicular Networks : Taxonomy and Architecture", IEEE GIIS'2017, Saint Pierre, Reunion Island, France, 25-27 October, 2017.
  - **Ahmed Alioua**, Sidi-Mohammed Senouci, Samira Moussaoui, Esubalew Alemneh, Med-Ahmed-Amine Derradji, Fella Benaziza, "A Distributed Multi-hop Clustering Algorithm for Infrastructure-less Vehicular Ad-Hoc Networks", ICT4DA'2017 (Springer), Bahir Dar, Ethiopia, 25-27 September, 2017.
  - **Ahmed Alioua**, Sidi-Mohammed Senouci, Samira Moussaoui, "dSDiVN : a distributed Software-Defined Networking architecture for Infrastructure-less Vehicular Networks", I4CS'2017 (Springer), Darmstadt, Germany, 26-28 June, 2017.
  - Mohamed Ayoub Messous, Amel Arfaoui, **Ahmed Alioua**, Sidi-Mohammed Senouci "A Sequential Game Approach for Computation-Offloading in an UAV Network", IEEE Globecom'2017, Singapore, 4-8 December, 2017.
  - **Ahmed Alioua**, Samira Moussaoui, Ihcen BenM'hamed, "An Accurate Computational Algorithm for Call Admission Control in Small Cell Networks", IEEE SaCoNet'2018, El-Oued, Algeria, 29-31 October, 2018.

CHAPITRE 1

INTRODUCTION

1.1 Contexte et motivation

De nos jours, les gens passent de plus en plus de temps dans les transports, ainsi, les véhicules sont devenus une partie importante de l'expérience de voyage. Les réseaux ad hoc de véhicules (*Vehicular Ad hoc NETwork*, VANET) ont émergé dans ce contexte afin d'améliorer la sécurité sur les routes, le confort et l'efficacité des déplacements. En permettant la communication entre les véhicules et/ou entre les véhicules et l'infrastructure au bord des routes, le développement des VANETs a contribué massivement dans le succès des systèmes de transports intelligents (STIs). Depuis quelques années, les réseaux de véhicules hétérogènes (*Heterogeneous Vehicular NETworks*, HetVNets) sont apparus comme une évolution importante qui enrichit les VANETs traditionnels avec différentes technologies sans fil (DSRC, WiFi, WiMax, etc.) et principalement avec la technologie cellulaire (4G/5G) [1, 2]. Plus récemment, le développement vigoureux des technologies de communications sans fil, l'explosion de l'utilisation de l'Internet mobile à l'intérieur des véhicules, et l'apparition de la ville intelligente, ont propulsé l'émergence des réseaux d'Internet de véhicules (*Internet of Vehicles*, IoV) comme une évolution innovante pour améliorer les performances des nouvelles générations des réseaux de véhicules hétérogènes et permettre des véhicules de plus en plus connectés avec des capacités commerciales et techniques plus évoluées [3, 4].

Toutefois, malgré le développement et la rapide émergence des réseaux de véhicules avec des capacités futuristes, de nombreux défis architecturaux restent à relever avant d'envisager un déploiement efficace de tels réseaux. Les architectures des VANETs souffrent principalement d'un manque de l'évolutivité dans le déploiement des services à une grande échelle, un manque d'intelligence à cause de l'aspect fermé des équipements et un manque de flexibilité et d'adaptabilité dans la gestion face à la grande hétérogénéité des technologies de communication sans fil (4G/5G, DSRC, WiFi, etc.). Toutes ces contraintes limitent les fonctionnalités du système, ralentissent la créativité et entraînent souvent à la sous-exploitation des ressources du réseau. Par conséquence, le besoin de nouvelles architectures plus flexibles et évolutives devient une exigence absolue pour faire face aux nouveaux défis croissants des prochaines générations des réseaux de véhicules surtout avec

l'arrivée très attendue de la 5G et du véhicule autonome.

Le nouveau paradigme émergent d'architecture de réseau, *Software-Defined Networking* (SDN) ou réseau défini par logiciel en français, a été proposé comme l'une des technologies les plus prometteuses pour surmonter les défis des réseaux de véhicules et faciliter leur gestion. SDN est principalement basé sur une séparation physique entre le plan de contrôle qui assure la gestion de réseau et le plan de données qui s'occupe de la diffusion et du transfert des informations. De plus, SDN s'appuie sur un contrôle et une intelligence logiquement centralisés dans un contrôleur logiciel qui détient une vue globale sur tout l'état du réseau et contrôle les autres équipements de plan de données du réseau dit de diffusion. Ces derniers sont considérés comme de simple transmetteurs/récepteurs de données avec une intelligence minimale [5]. Ainsi, SDN a connu un succès fulgurant dans le domaine industriel et académique. Depuis, les chercheurs explorent de plus en plus comment tirer profit des avantages du SDN pour améliorer les performances et apporter de nouvelles solutions innovantes aux défis des architectures des réseaux de véhicules. Récemment, plusieurs architectures basées sur SDN (voir [6–34]) ont été proposées dans la littérature pour les réseaux de véhicules. Ces architectures ont démontré que l'utilisation de SDN peut apporter la flexibilité, l'évolutivité et la programmabilité. SDN permet aussi d'utiliser plus efficacement les ressources de réseau et d'introduire de nouveaux services. Néanmoins, la quasi-majorité des solutions disponibles sont basées sur l'appui de l'infrastructure véhiculaire comme un endroit privilégié pour déployer le/les contrôleur(s) SDN. Autrement dit, une telle architecture ne peut fonctionner que dans les zones couvertes par l'infrastructure. Une hypothèse peu réaliste surtout avec les premiers pas de déploiement des réseaux de véhicules, où la couverture totale de réseau par l'infrastructure fixe est loin d'être assurée.

Dernièrement, afin d'assister les véhicules dans les environnements véhiculaires sans ou avec très peu de couverture en infrastructure, les drones (*Unnamed Aerial Vehicle*, UAV) ont été utilisés comme une solution flexible pour faciliter de nombreuses applications véhiculaires telles que les missions de recherche et de secours [35–38]. Dans de telles missions, les drones peuvent collecter divers types de données, comme des photos et des vidéos aériennes. Ces données doivent être traitées le plus rapidement que possible pour épurer des informations essentielles sur le nombre et l'état des victimes, le nombre des véhicules impliqués, etc. Le traitement de ce type d'informations fait appel généralement pour des techniques de la reconnaissance des formes et de traitement des vidéos qui sont des tâches complexes à calcul intensif. De plus, le calcul de telles tâches par des équipements à ressources limités comme ceux des drones risque de durer longtemps et consommer beaucoup d'énergie. Face à une telle situation, le drone peut délester (*offloading*) le calcul de ces tâches aux véhicules au sol qui sont beaucoup plus puissants et ont peu de contrainte d'énergie. Cependant, l'envoi de grands volumes de données via l'interface sans fil du drone peut parfois consommer plus d'énergie et de délai que le calcul local de ces mêmes données. Face à ce dilemme, un système de décision pour le délestage de calcul doit être mis au point pour mieux optimiser le traitement des données.

Au cours des dernières années, il y a eu une avalanche de trafic de données mobiles principalement causée par une demande explosive des applications gourmandes en bande passante telles que la vidéo à la demande, la vidéo en streaming, etc. Ces consommateurs de contenus sont de plus en plus mobiles et souhaitent accéder à ces contenus même en voiture. Cependant et très souvent, un groupe de véhicules voisins est susceptible de manifester des intérêts semblables et demander des contenus similaires, conduisant à des transmissions redondantes du même contenu pouvant être parfois de grande taille [39]. La technique de mise en cache (*caching*) qui consiste à stocker des contenus populaires localement sur des nœuds, physiquement proches de l'utilisateur final, a été proposée comme une technologie innovante pour supporter cet accroissement du trafic de données. Toutefois, une opération réussie de mise en cache exige la coopération de différents acteurs [40], [41] à l'instar des fournisseurs de contenu (*Content Provider*, CP), comme YouTube et Netflix et les gestionnaires des caches, tels que les opérateurs de réseau mobile (*Mobile Network Operator*, MNO). À cette fin, des mécanismes d'incitation devraient être développés pour encourager ces acteurs à collaborer afin d'assurer une mise en cache plus efficace [40]. Dans la littérature, quelques travaux [39–46] ont déjà traité le problème de mise en cache d'une perspective commerciale en se concentrant sur la façon d'encourager les acteurs à participer dans l'amélioration du processus de la mise en cache. Cependant, la plupart de ces travaux étudient le système d'incitation à la mise en cache dans un marché cellulaire monopolistique avec un MNO comme unique gestionnaire des caches. C'est une hypothèse peu réaliste dans les marchés cellulaires concurrentiels de nos jours où il existe plusieurs MNOs qui sont en concurrence permanente sur n'importe quelle source de revenus.

1.2 Contributions

Dans ce contexte, nous nous intéressons dans cette thèse dans un premier temps à répondre à la problématique d'une intégration efficace et flexible du nouveau paradigme de SDN dans les réseaux de véhicules. Ensuite, dans un second temps, nous cherchons, à travers deux cas d'application, à prouver sa faisabilité et de démontrer les avantages que peut apporter l'intégration d'un tel paradigme innovant pour améliorer les performances des nouvelles générations des réseaux de véhicules. Les principales contributions de cette thèse sont résumées dans ce qui suit :

- Nous commençons par présenter, une nouvelle architecture semi-centralisée qui tire profit des avantages de la technique de *clustering*, les réseaux cellulaires et le concept de *cloud/fog computing* pour faciliter une intégration efficace du paradigme de SDN dans les réseaux de véhicules hétérogènes, appelée *Cluster-based Software-Defined Heterogeneous Vehicular Networks (CSDHVN)*. CSDHVN est une architecture flexible avec une hiérarchie de trois types de contrôleurs SDN déployés à la périphérie de réseaux, afin de mieux répondre aux exigences en termes de faible délai des services de sécurité routière et les applications en temps réel. Le principal but de CSDHVN est de combler le manque d'une architecture complète dans la littérature basée sur SDN qui fonctionne non seulement dans les zones couvertes par l'infrastructure fixe, mais aussi dans les zones non couvertes. Le mode de contrôle dans CSDHVN est logiquement centralisé, mais physiquement distribué. Il est centralisé dans les zones couvertes en s'appuyant sur une hiérarchie de trois contrôleurs SDN

et multi-sauts distribué dans les zones non couvertes grâce à des contrôleurs installés sur les véhicules mobiles. De plus, nous proposons un mécanisme de récupération de secours efficace qui exploite la hiérarchie des contrôleurs et l'auto-structuration de la technique de *clustering* pour assurer la continuité de service en cas de panne d'un des contrôleurs SDN. Les résultats numériques ont démontré la faisabilité de l'architecture proposée et la fiabilité du mécanisme de récupération de secours.

– Ensuite, nous étendons la version distribuée multi-sauts de CSDHVN afin de faciliter la gestion des réseaux de véhicules sans infrastructure assistés par des drones. Nous projetons cette architecture sur un cas d'utilisation relatif à un service de sécurité dans les missions de secours sur la route, dans lequel nous proposons d'équiper les véhicules de secours par des mini-drones pour les aider à explorer et à enquêter sur les tronçons de la route inaccessibles à cause d'un quelconque incident. Puis, nous concentrons nos efforts sur l'établissement d'une politique de traitement efficace des données collectées par les drones. Nous formulons ce problème comme un problème de prise de décision en matière de délestage (*offloading*) de calcul des tâches à partir du drone à son véhicule de secours et/ou de partage du calcul des tâches entre le véhicule de secours et ses véhicules voisins. Le principal but consiste à trouver le meilleur équilibre entre le délai et l'énergie de calcul. Nous modélisons le problème de décision à l'aide d'un jeu séquentiel avec deux joueurs et nous concevons deux algorithmes distribués pour résoudre les problèmes d'optimisation des résultats. Notre politique de traitement des données basée sur SDN assure des performances de calcul efficace avec le meilleur équilibre entre le délai et l'énergie de calcul.
– Enfin, nous proposons une nouvelle politique pour l'incitation à la mise en cache comme un cas d'application de CSDHVN dans le contexte des réseaux d'IoVs. Nous considérons un scénario de mise en cache concurrentiel composé d'un seul CP et plusieurs MNOs. Les MNOs sont en concurrence permanente pour le stockage de la quantité limitée de contenu populaire de CP. Le CP encourage les MNOs à stocker son contenu populaire dans leurs caches sur les stations de base cellulaires avec la probabilité d'accès la plus élevée. Nous formulons l'interaction entre le CP et les MNOs à l'aide d'un jeu de type *Stackelberg* avec un sous jeu non-coopératif pour modéliser le conflit entre les MNOs. Nous explorons deux problèmes d'optimisation lorsque le CP vise à maximiser la satisfaction de ses utilisateurs en termes de délai d'accès/téléchargement, quand chaque MNO vise à maximiser son gain monétaire. Les résultats ont montré que la considération de plusieurs prioritaires de caches compétitifs dans les scénarios de la mise en cache peut améliorer de manière significative la qualité de l'expérience des utilisateurs et contribuer à la diminution du coût de mise en cache.

1.3 Organisation de la thèse

En plus de l'introduction et de la conclusion, cette thèse est structurée en quatre chapitres. Dans ce qui suit nous détaillons le contenu des différents chapitres.

Nous fournissons dans le chapitre 2, une vue d'ensemble sur les concepts de base des réseaux de véhicules et le paradigme de SDN. Nous énumérons les défis et les limitations

des architectures de réseaux de véhicules avant de présenter les avantages du paradigme SDN comme une solution innovante pour surmonter les défis des VANETs. Une discussion avec une taxonomie des efforts d'intégration de SDN dans les réseaux de véhicules est ensuite fournie et les principaux challenges face au déploiement de SDN dans les VANETs sont énumérés.

Nous présentons dans le chapitre 3, une proposition d'une nouvelle architecture basée sur le paradigme SDN pour les réseaux de véhicules hétérogènes. Nous commençons par décrire la motivation et la problématique avant de détailler l'architecture basée sur SDN pour les HetVNets. Ensuite, nons développons le mécanisme de récupération de secours et nous citons quelques cas d'applications pour l'architecture proposée. Le contenu de ce chapitre est publié dans [47–49].

Dans le chapitre 4, nous présentons un cas d'utilisation de l'architecture basée sur le paradigme SDN proposée dans le chapitre 3 pour un traitement efficace des données dans un scénario de secours avec un réseau de véhicules sans infrastructure assisté par des drones. Nous commençons par présenter une nouvelle architecture basée sur le paradigme de SDN pour les réseaux de véhicules assistés par des drones, nous décrirons le cas d'utilisation relatif à une mission de secours sur la route et nous discutons la motivation pour l'étude du problème de traitement des données. Ensuite, nous modélisons le problème sous forme d'un jeu séquentiel avant de proposer deux algorithmes distribués pour résoudre le jeu proposé et trouver les solutions optimales. Le contenu de ce chapitre est publié dans [50].

Nous présentons dans le chapitre 5, un autre cas d'utilisation de l'architecture basée sur le paradigme de SDN proposée dans le chapitre 3 pour le problème d'incitation à la mise en cache dans les réseaux IoVs. Nous commençons par décrire le modèle de système et nous discutons la motivation de la politique d'incitation à la mise en cache compétitive. Ensuite, nous modélisons le problème de l'incitation à la mise en cache sous forme d'un jeu de type *Stackelberg* avant de détailler la méthode pour le résoudre et obtenir les paramètres optimaux du système. Le contenu de ce chapitre est publié dans [51].

CHAPITRE 2

VUE D'ENSEMBLE SUR LES RÉSEAUX DE VÉHICULES ET LE PARADIGME DE SOFTWARE-DEFINED NETWORKING.

2.1 Introduction

De nos jours, nous passons de plus en plus de temps dans les transports, que ce soit dans des véhicules personnels ou dans des transports en commun. De plus, nos usages des moyens de communication sont devenus de plus en plus nomades surtout avec les grandes avancées des technologies d'information et de communication (TIC). Le concept de réseaux ad hoc de véhicules a donc vu le jour suite à cette évolution dans le but d'offrir une large variété de services, allant de l'amélioration de la sécurité routière à l'optimisation du trafic, en passant par le divertissement du conducteur et des passagers. En effet, ce type de réseau se caractérise principalement par la haute mobilité des nœuds, la topologie extrêmement dynamique et la grande échelle de réseau. Les réseaux de véhicules représentent une projection des systèmes de transports intelligents (*Intelligent transportation Systems*, ITS), dans lesquels les véhicules sont capables de communiquer les uns avec les autres et avec aussi les infrastructures le long des routes. Dans ce chapitre, nous présentons les concepts de base des réseaux de véhicules avant de décrire l'état de l'art du paradigme de software-defined networking appliqué aux réseaux de véhicules.

Le suite de ce chapitre est structuré comme suit. Initialement, nous présentons les concepts de base des réseaux de véhicules dans la section 2.2. Ensuite, nous abordons l'évolution des réseaux de véhicules, au début vers les réseaux de véhicules hétérogènes et plus récemment vers l'Internet de véhicules dans la section 2.3. Dans la section 2.4, nous citons les principaux défis des architectures des réseaux de véhicules de nos jours. Enfin dans la section 2.5, nous présentons un bref survol sur le paradigme de software-defined networking dans les réseaux de véhicules et nous terminons le chapitre par une conclusion dans la section 2.6.

2.2 Réseau ad hoc de véhicules

Dans cette section, nous présentons une vue d'ensemble sur les concepts de base des réseaux de véhicules.

2.2.1 Définition

Les réseaux ad hoc de véhicules souvent appelés VANETs, pour *Vehicle Ad-hoc NETworks*, sont une technologie émergente déployée pour permettre la communication entre les véhicules mobiles les uns avec les autres par l'intermédiaire d'une communication véhicule à véhicule (*Vehicle to Vehicle*, V2V) aussi bien qu'avec l'infrastructure fixe au bord de la route par l'intermédiaire d'une communication véhicule à infrastructure (*Vehicle to Infrastructure*, V2I), voir la figure 2.1. Les VANETs sont considérés comme la composante majeure des futurs systèmes de transport intelligents, dont le but principal est d'améliorer la sécurité sur les routes, le confort des voyageurs et l'efficacité de la gestion du trafic routier [14].

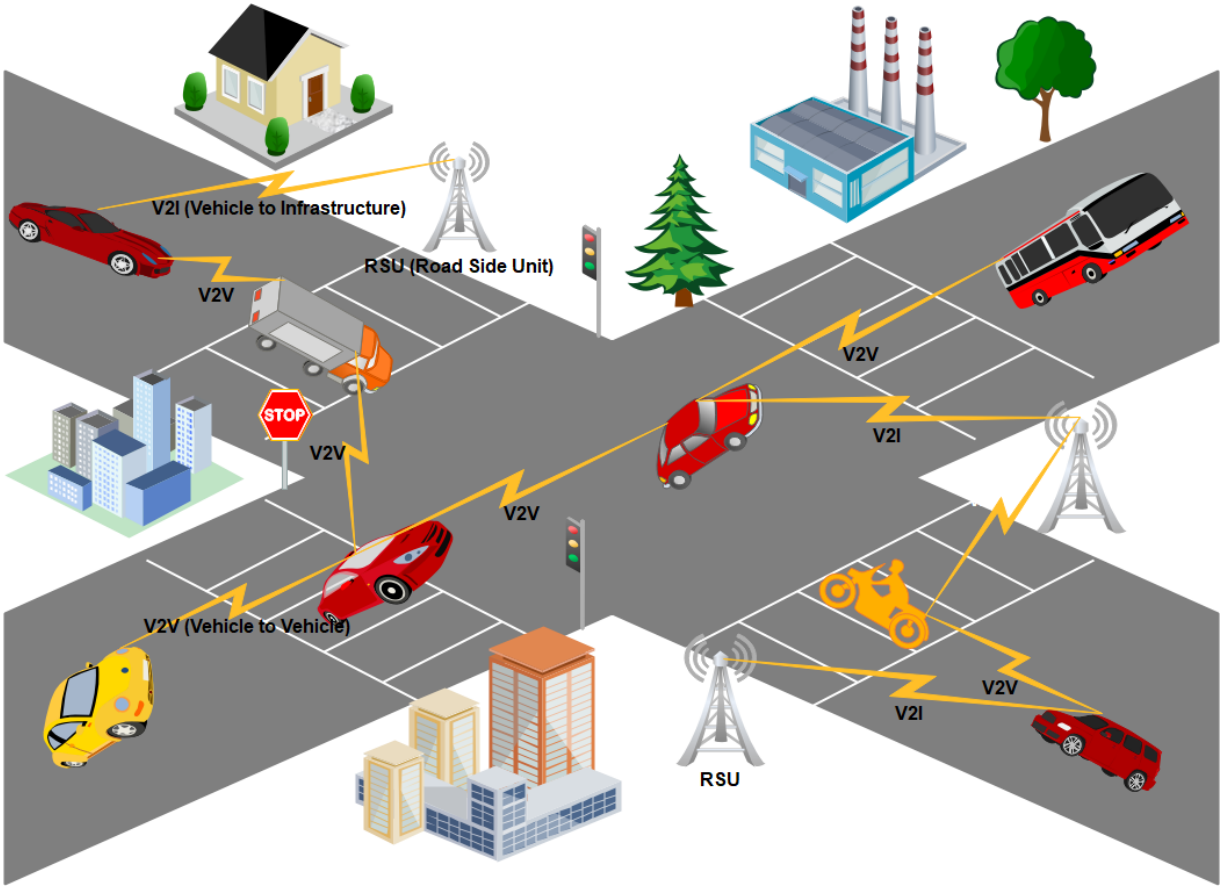


FIGURE 2.1 – Réseau ad hoc de véhicules.

Un véhicule dans les VANETs est un nœud mobile intelligent équipé par des nouvelles générations des technologies de communication sans fil afin de permettre une communication efficace avec les autres véhicules ainsi qu'avec l'infrastructure au bord des routes.

2.2.2 Composants et mode de communication

Un réseau de véhicules repose principalement sur trois éléments de base : les véhicules intelligents, les équipements de bord de route et la communication entre les véhicules.

2.2.2.1 Équipements (OBU et RSU)

L'architecture générale d'un réseau de véhicules définit deux grandes catégories des équipements ou de composants : les équipements internes au véhicule (*On Board Units*, OBUs) et les équipements de bord de la route externe aux véhicules (*Road Side Units*, RSUs) [52], [53]. Les OBUs sont des équipements mobiles (*e.g.*, radars, capteurs, unité de calcul, etc.) installés sur les véhicules intelligents qui peuvent enregistrer, calculer, localiser et envoyer des messages, comme ils peuvent collecter et traiter des informations utiles. Les RSUs sont les équipements placés au bord des routes et qui constituent l'infrastructure fixe du réseau de véhicules. Ces unités peuvent informer les véhicules à proximité en diffusant des informations sur les conditions du trafic, météorologiques ou spécifiques à la route (*e.g.*, la vitesse maximale, l'autorisation de dépassement, etc.). Les RSUs peuvent aussi jouer le rôle de station de base en permettant la communication entre deux véhicules éloignés ou en relayant tout simplement l'information envoyée par un véhicule.

2.2.2.2 Modes de communication

Dans les réseaux de véhicules, on peut distinguer trois modes de communication [54], [55] : le mode avec infrastructure, le mode ad hoc et le mode hybride.

- **Mode avec infrastructure :** Dans ce mode, les véhicules communiquent avec l'infrastructure fixe du réseau, *i.e.*, les RSUs, par l'intermédiaire d'une communication V2I, voir la sous-figure 2.2(a). Ce mode de communication permet une meilleure utilisation des ressources partagées et il démultiplie les services fournis aux passagers concernant le trafic. L'inconvénient majeur de ce mode de communication est que le déploiement des RSUs le long des routes est une tâche coûteuse et prend beaucoup de temps, sans oublier les coûts relatifs à leur maintenance.
- **Mode ad hoc :** Ce mode représente une architecture ad hoc où les véhicules communiquent les uns avec les autres par l'intermédiaire d'une communication V2V, voir la sous-figure 2.2(b). En effet, un véhicule peut communiquer directement avec un autre véhicule s'il se situe dans sa zone de couverture radio, ou bien par le biais d'un protocole multi-sauts qui se charge de transmettre les messages en passant par les véhicules voisins comme des relais. Ce mode de communication fonctionne suivant une architecture décentralisée et garantit une communication moins coûteuse et plus flexible avec une petite latence [56]. En effet, les communications V2V sont très efficaces pour le transfert des alertes relatives aux services liés à la sécurité routière. Toutefois, elles ne garantissent pas une connectivité permanente entre les véhicules et souffrent de fréquentes déconnexions dues à la forte mobilité des véhicules.
- **Mode hybride :** Ce mode combine les deux premiers modes. Il utilise les communications V2V pour étendre la zone de couverture limitée des infrastructures au bord de la route, voir la sous-figure 2.2(c). Ce mode à un grand intérêt économique puisqu'il permet de diminuer les coûts de déploiement des unités tout au long des routes. Il permet aussi de bénéficier des faibles délais des communications V2V.

Par conséquent, ce type de communication est le plus utilisé, parce qu'il permet de bénéficier des avantages des deux modes précédents.

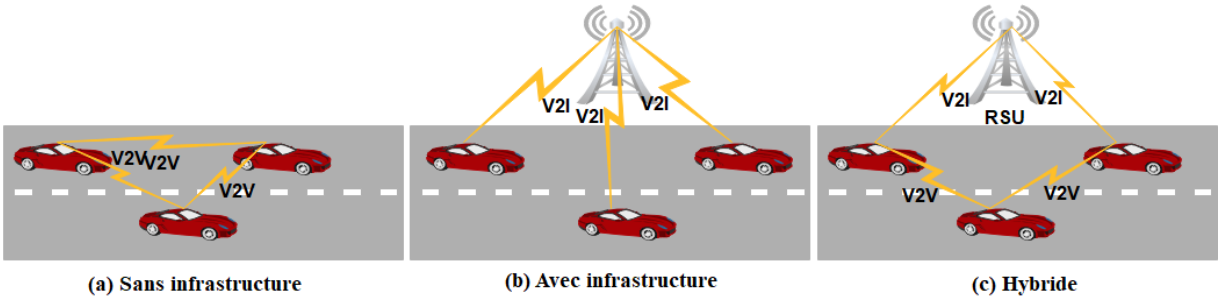


FIGURE 2.2 – Mode de communication dans un réseau de véhicules.

2.2.2.3 Caractéristiques

Les réseaux de véhicules étant un type spécifique des réseaux mobiles ad hoc (*Mobile Ad-hoc NETwork*, MANET), ils héritent de la plupart de leurs caractéristiques. Cependant, les VANETs ont des caractéristiques spécifiques qui les distinguent. Parmi les caractéristiques les plus importantes, nous citons :

- Forte mobilité et topologie hautement dynamique du réseau : les VANETs sont caractérisés par une forte mobilité des nœuds liée à l'importante vitesse des voitures qui varie selon l'environnent, en moyenne de 50km/h en zones urbaines et peut atteindre 120km/h sur les autoroutes. Par conséquent, un véhicule peut rapidement rejoindre ou quitter le réseau en un temps très court, ce qui rend les changements de la topologie très fréquents.
- Capacité de traitement, d'énergie et de communication : les éléments des VANETs disposent de capacités énergétiques suffisantes qu'elles puisent du système d'alimentation des véhicules. De plus, les véhicules sont censés être équipés par des OBUs avec une grande capacité de traitement, de stockage et peuvent posséder de multiples interfaces de communication sans fil [57].
- Modèle de mobilité et prédictibilité de mouvement des véhicules : dans les VANETs, les véhicules ont tendance à avoir des trajectoires très prévisibles et des degrés de liberté limités dans leurs habitudes de déplacement qui sont généralement restreintes aux infrastructures routières (routes prédéfinies, ainsi que les directions et les nombres de voies de circulation) [58]. Par ailleurs, plusieurs facteurs peuvent affecter le modèle de la mobilité des véhicules tels que l'environnement et les contraintes qui l'imposent (*e.g.*, les limitations de vitesse, les sens interdis, etc.). De plus, la mobilité est directement liée au comportement des conducteurs [59].
- Évolutivité et diversité de la densité : les VANETs ont également le potentiel de croissance à une très grande échelle, spécialement dans les zones urbaines où les intersections et les routes à voies multiples sont fréquentes. En outre, l'évolution de la densité des véhicules n'est pas uniforme, mais à variation spatio-temporelle et peut varier d'un réseau très dense (cas urbain) à une distribution de véhicules clairsemée (cas rural). D'un point de vue temporel, la densité est différente selon qu'on considère la nuit ou la journée, les heures de pointe ou les heures creuses [57].

– Sécurité et anonymat : le risque de la falsification des messages échangés entre les véhicules pourrait augmenter les fausses alertes et provoquer parfois même des accidents. De plus, la communication sans fil utilisée dans les VANETs est très vulnérable. En effet, la sécurité des données et la vie privée se dressent comme un problème majeur face à la démocratisation des VANETs.

2.2.2.4 Applications

Les applications des VANETs peuvent être classées en trois grandes catégories : les applications de sécurité, les applications de gestion du trafic routier et les applications du confort.

– **Applications de sécurité :** les applications de la sécurité routière ont comme principal objectif, l'amélioration de la sécurité des conducteurs et/ou des passagers sur les routes. Elles visent à sauver des vies humaines en avisant les véhicules de toute situation dangereuse et en réduisant ainsi le nombre des accidents sur les routes. Cette catégorie d'applications comporte tous les services ayant un impact direct sur la sécurité des personnes et des biens. Parmi les applications de sécurité routière, on trouve les applications d'aide aux dépassements de véhicules, des alertes de freinage d'urgence, des avertissements de collision et d'accidents, de communication d'informations sur l'état de la chaussée, etc. [52]
– **Applications de gestion de trafic routier :** les applications de gestion du trafic routier visent à améliorer la fluidité de la circulation des véhicules, prévenir la congestion et garantir une utilisation efficace des capacités des routes dans le but de réduire les embouteillages. Plusieurs services sont proposés pour cette catégorie, parmi lesquels on peut citer : la surveillance de l'état du trafic, l'ordonnancement des feux de signalisation, la prédiction des embouteillages, la proposition des itinéraires alternatifs, etc. [52], [60]
– **Applications de confort :** les applications de confort ou de divertissement ont comme principal objectif d'assurer aux usagers de la route un voyage convivial et agréable grâce à l'utilisation de nouveaux services tels que les services de la communication à vocation de divertissement comme par exemple les jeux en réseau et le partage de la musique et de la vidéo. À toutes ces applications s'ajoutent les annonces d'ordre commercial comme les offres des restaurants ou d'ordre culturel comme les informations touristiques. Il y a aussi des services télématiques qui visent à faciliter la vie des usagers de la route comme le péage à distance sur les autoroutes [52].

2.2.2.5 Technologies et standard de communication

Pour satisfaire les exigences des applications véhiculaires, de nombreuses normes ont été développées telles que la technologie DSRC (*Dedicated Short-Range Communication*) et le standard WAVE (*Wireless Access in Vehicular Environment*).

A. Technologie DSRC

La technologie DSRC a été proposée comme un standard pour les communications de courtes portées dans le but de supporter les communications avec une petite latence (V2V et V2I). Le DSRC permet une communication dans la plage allant de 300 m à 1 km

et supporte une vitesse de véhicule pouvant atteindre 200 km/h. Cette norme est semi-duplex avec un taux de transfert de données de 6-27 Mbps. DSRC utilise la technique de multiplexage OFDM (*Orthogonal Frequency Division Multiplexing*). De plus, l'utilisation du spectre DSRC est gratuite [1].

Le DSRC a été lancé en 2001 avec la normalisation ISO CALM (*Communication Access for Land Mobiles*), puis finalisé en 2010 dans le travail sur IEEE 802.11p. En 1999, la commission fédérale des communications (*Federal Communication Commission*, FCC) des Etats-Unis a attribué 75 MHz du spectre de communication au DSRC et les ITSs à la bande 5,9 GHz entre la plage 5,850 et 5,925 GHz. De même, en 2008, l'institut européen des normes de télécommunication (*European Telecommunications Standards Institute*, ETSI) a attribué 70 MHz de spectre à la bande 5,8 GHz pour les applications DSRC [1].

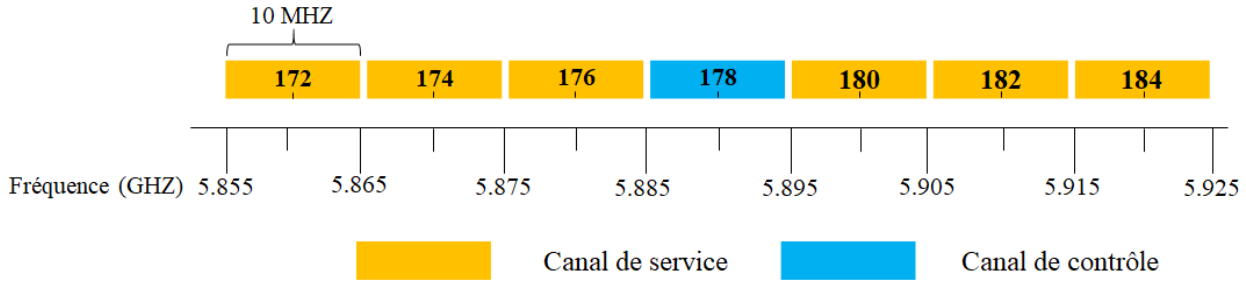


FIGURE 2.3 – Spectre de DSRC alloué par FCC [1].

Comme le montre la figure 2.3, le spectre DSRC est divisé en sept canaux de 10 MHz chacun, un canal de contrôle (*Control Channel*, CCH) et six canaux de service (*Service Channels*, SCHs). Le canal 178 est le canal de contrôle, utilisé exclusivement pour diffuser les messages d'urgences des applications de sécurité. De manière analogue, les canaux 172 et 184 sont réservés aux applications de sécurité, tandis que les autres canaux de service sont destinés à la fois à l'usage des applications liées et non liées à la sécurité. Selon l'ETSI, le spectre DSRC est divisé en intervalles de temps de 50 ms et les messages ont deux priorités : faible pour les messages de dissémination de données ou élevée pour les messages de sécurité [1].

B. Standards de communications (IEEE 1609.x) pour WAVE

L'organisation des standards IEEE (*Institute of Electrical and Electronics Engineers*) a enrichi sa famille de protocoles 802.11 par le protocole 802.11p spécifique aux communications inter-véhicules. Ce protocole est principalement basé sur le protocole 802.11a. En effet, 802.11p modifie les deux couches physique et MAC (*Medium Acces Control*) du protocole 802.11a pour les adapter aux exigences des réseaux de véhicules et à la norme DSRC. En plus, IEEE a défini la famille des protocoles 1609.x, souvent appelés WAVE, pour accéder au médium sans fil dans les VANETs [61]. Cette famille de protocoles est structurée en quatre standards, qui sont brièvement décrits dans ce qui suit [61] :

- IEEE 1609.1 : c'est le standard qui spécifie les services (de données et de gestion) et les interfaces de l'application de gestion des ressources dans l'architecture WAVE. Il définit aussi les formats des messages et la manière de stocker les données.

- IEEE1609.2 : c'est le standard qui spécifie les services de sécurité de système WAVE et décrit le format et le traitement des messages de sécurité.
- IEEE 1609.3 : c'est le standard qui spécifie les services de réseau et offre les services de routage et d'adressage.
- IEEE 1609.4 : c'est le standard qui définit l'organisation, l'ordonnancement et l'utilisation des canaux DSRC. Il définit le mécanisme qui permet à plusieurs nœuds de communiquer sur le même canal au même temps.

Dans ce qui suit, nous présentons un bref survol sur l'évolution des VANETs.

2.3 Évolution des réseaux de véhicules

À cause de la haute mobilité des véhicules et la topologie hautement dynamique des VANETs, il est difficile de fournir une qualité de service (*Quality of Service*, QoS) satisfaisante pour les services ITSs en s'appuyant uniquement sur un seul réseau d'accès sans fil à l'instar de DSRC. D'un côté, les communications V2V et V2I souffrent des problèmes d'évolutivité à cause de la couverture radio limitée de DSRC, le coûteux déploiement de l'infrastructure (RSUs) ainsi que les longs délais de bout en bout face au grand nombre de véhicules. De l'autre côté, bien que l'intégration des réseaux cellulaires, déjà existants et largement déployés dans les VANETs peut assurer une large couverture aux utilisateurs de réseau de véhicules avec un débit élevé. Cependant, ils ne peuvent pas toujours garantir les besoins des services de sécurités exigeant des communications en temps réels stricts. En combinant entre plusieurs technologies d'accès sans fil, plus particulièrement le DSRC et les réseaux cellulaires (4G/5G), les réseaux de véhicules dits hétérogènes, constituent une solution prometteuse pour répondre aux exigences des communications ITSs. Ils ont non seulement la capacité de fournir une couverture de large portée, mais aussi supporter les faibles délais des messages de sécurité [1], [2].

2.3.1 Réseaux de véhicules hétérogènes

Les réseaux de véhicules hétérogènes (*Heterogeneous Vehicular NETworks*, HetVNets), sont une technologie qui enrichit les VANETs traditionnels avec différentes technologies sans fil (DSRC, WiFi, WiMax, etc.) et principalement avec la technologie cellulaire (4G/5G), voir la figure 2.4. Grâce à cette évolution, les véhicules peuvent communiquer en utilisant différents réseaux et bénéficier de l'abondance des ressources hétérogènes avec des capacités variables de différentes technologies pour répondre plus efficacement aux exigences des communications véhiculaires avec une meilleure QoS. Désormais, les véhicules peuvent donc profiter de cette hétérogénéité pour jouir d'une couverture de communication plus étendue avec une faible latence, un débit plus élevé et une bande passante plus large. De plus, la complémentarité entre différentes technologies d'accès sans fil va largement aider les futurs véhicules autonomes à fonctionner dans la réalité [1], [2].

Cependant, l'intégration de plusieurs réseaux sans fil dans les environnements véhiculaires peut augmenter considérablement la complexité de la conception et de la gestion des nouvelles générations des réseaux de véhicules. Par exemple, un handover vertical doit être

assuré en permanence pour garantir la continuité de la connectivité lors du basculement de l'utilisation d'une technologie à une autre [1].

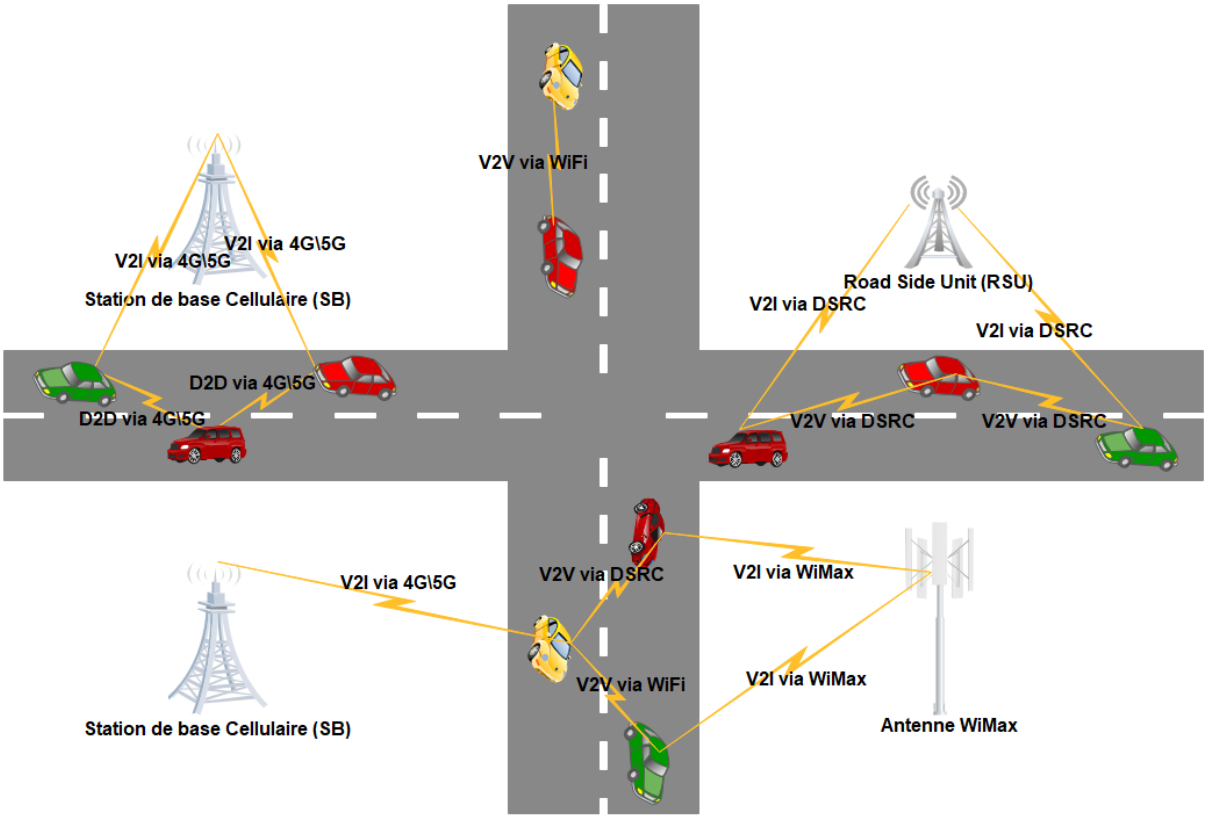


FIGURE 2.4 – Réseau de véhicules hétérogènes.

Récemment, la rapide émergence des villes intelligentes, l'apparition des véhicules autonomes, la démocratisation de l'Internet mobile et l'arrivée très attendue de la 5G, ont propulsé l'apparition des applications véhiculaires de plus en plus orientées multimédia, qui ont besoin d'accéder à Internet à tout moment et à partir de n'importe quel endroit soit à travers des systèmes embarqués dans des smart-phones ou à travers des équipements connectés intégrés aux véhicules. Dans ce cadre, les réseaux d'Internet de véhicules ont émergé comme une évolution importante des réseaux de véhicules.

2.3.2 Internet de véhicules

Durant les dernières années, le développement vigoureux des technologies de communications sans fil, l'explosion de l'utilisation de l'Internet mobile à l'intérieur des véhicules, et l'apparition de la ville intelligente ont propulsé une rapide émergence de la nouvelle génération des réseaux de véhicules avec des capacités commerciales et techniques plus avancées, autrement dit, l'Internet de véhicules (*Internet of Vehicles*, IoV).

L'Internet de véhicules a émergé en tant que technologie innovante qui vise l'amélioration des performances des nouvelles générations des réseaux de véhicules par des

technologies innovantes comme le cloud [1]/fog [2] computing et la 5G. L'IoV est considéré comme l'instanciation du paradigme de l'Internet des objets (*Internet of Thing*, IoT), dans le contexte véhiculaire afin de permettre des véhicules intelligents de plus en plus connectés avec de nouvelles applications basées sur IP (*Internet Protocol*), voir la figure 2.5 [3].

L'objectif de l'IoV est d'intégrer de multiples utilisateurs, plusieurs véhicules, différents objets et divers réseaux, afin de toujours fournir la meilleure capacité de communication possible aux véhicules connectés [4]. Contrairement aux VANETs traditionnels, IoV suppose que les véhicules sont connectés à Internet via différents objets intelligents. De plus, IoV utilise les communications V2V, V2I, V2H (véhicule à humain) et V2C (véhicule à capteurs) pour communiquer entre le réseau public et les véhicules [62]. IoV peut être aussi considéré comme une superclasse des réseaux de véhicules traditionnels avec une meilleure évolutivité et de nouvelles applications orientées IP plus évoluées [62]. Ces applications vont améliorer considérablement la sécurité sur les routes et offrir de nouveaux services de confort plus connectés, comme les réseaux sociaux, le streaming, la navigation en ligne, l'acquisition du trafic en temps réel, etc.

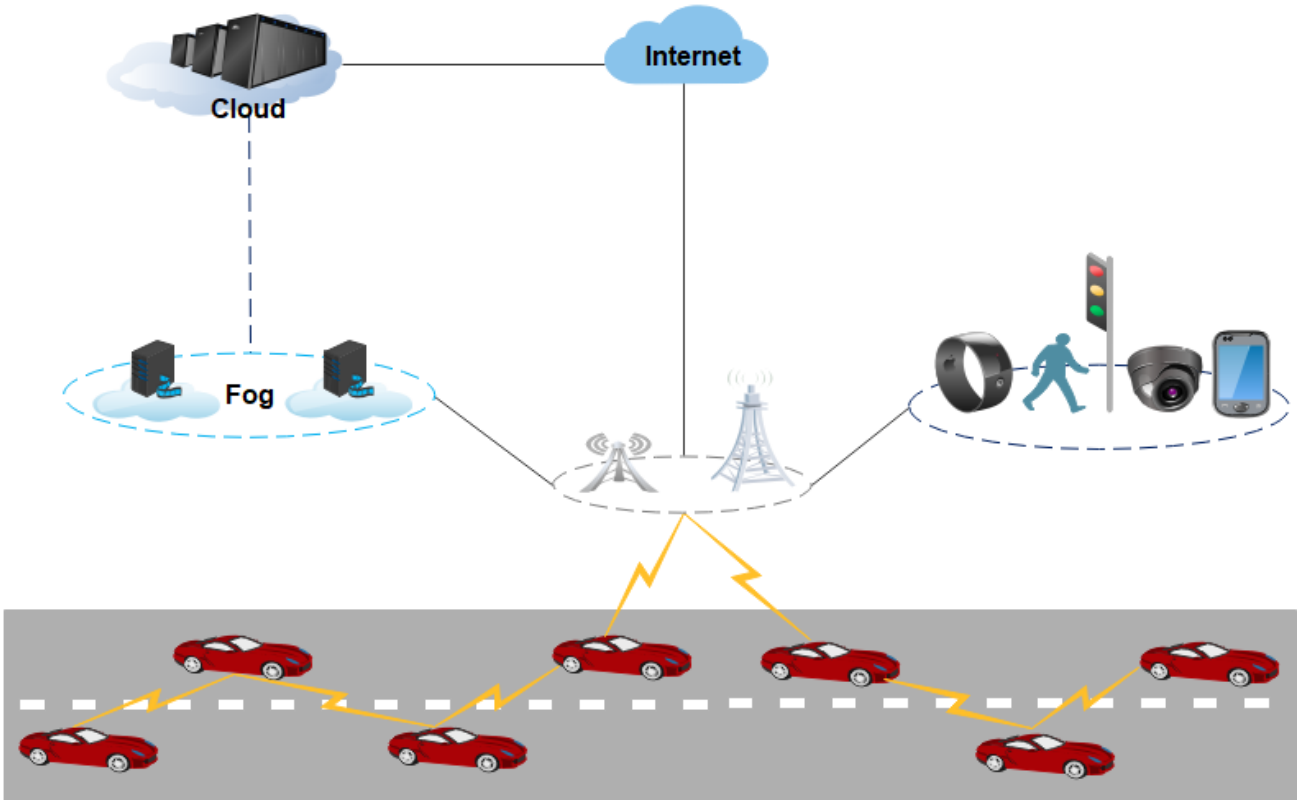


FIGURE 2.5 – Architecture d'Internet de véhicules.

Dans la section suivante, nous citons les principaux défis et challenges des architectures des réseaux de véhicules actuels.

1. **Cloud computing :** consiste à fournir de puissantes ressources informatiques de calcul et/ou de stockage à la demande pour des utilisateurs distants via Internet.
2. **Fog computing :** consiste à distribuer les ressources de calcul et de stockage dans les emplacements les plus efficaces entre l'utilisateur final et le Cloud afin d'améliorer les délais de communications.

2.4 Challenges et défis des architectures des réseaux de véhicules

Malgré le développement et la rapide émergence des réseaux de véhicules avec des capacités futuristes, leurs architectures souffrent de plusieurs lacunes. Outre le problème de l'hétérogénéité des équipements de réseau qui engendre de grandes difficultés de gestion et d'intégration [20], on peut citer par exemple *(i)* le manque de l'évolutivité dans le déploiement des services à une grande échelle dans une topologie aussi dense et dynamique telle que celle des réseaux de véhicules [6], *(ii)* le manque d'intelligence, principalement dû à l'aspect fermé des équipements véhiculaires et leurs caractéristiques inhérentes telles que l'absence de la programmabilité et leur dépendance au développement vis-à-vis aux fournisseurs. Ce qui implique des architectures rigides et difficiles à gérer, *(iii)* le manque de flexibilité et d'adaptabilité, induit par la grande diversité des environnements de déploiement et la large hétérogénéité des technologies de communication sans fil (4G/5G, DSRC, WiFi, etc.). Ce qui rend difficile de sélectionner la technologie d'accès la mieux adaptée en fonction du contexte actuel et de l'évolution rapide des paramètres du réseau, afin de répondre plus efficacement aux exigences en termes de QoS des diverses applications véhiculaires. L'augmentation exponentielle de nombre des véhicules et les difficultés qui peuvent introduire dans la gestion de tels larges réseaux, plus particulièrement pour appliquer des services à grande échelle. Il s'ajoute à tout ça, la grande diversité des applications et l'importante augmentation de la demande de services connectés plus avancés.

Toutes ces contraintes limitent les fonctionnalités du système, ralentissent la créativité et entraînent souvent la sous-exploitation des ressources du réseau. En effet, les solutions offertes par ces architectures ne sont généralement adaptées que dans un contexte spécifique ou des situations particulières. Par conséquent, le besoin de nouvelles architectures pour les réseaux de véhicules de nos jours, plus flexibles et évolutives devient une exigence absolue pour permettre une utilisation efficace des ressources et faire face aux nouveaux défis croissants des prochaines générations des réseaux de véhicules surtout avec l'arrivée très attendue de la 5G et le rapide développement du véhicule autonome.

Le paradigme d'architecture réseau émergent de *software-defined networking* (SDN) a été proposé comme une solution innovante et prometteuse pour surmonter les défis des réseaux de véhicules. Dans ce qui suit nous allons donner un bref aperçu sur l'état de l'art de paradigme de SDN appliqué aux réseaux de véhicules.

2.5 Software-defined networking et les réseaux de véhicules

Initialement conçu pour les réseaux filaires et en particulier les centres de données, SDN a connu un succès fulgurant dans le domaine industriel et académique. En 2012, les ingénieurs de Google ont annoncé qu'ils ont basculé à l'utilisation de SDN pour connecter leurs réseaux WANs (*Wide Area Networks*) des centres de données [63]. Actuellement, il existe même des équipements de réseau compatibles avec SDN et disponibles sur le

marché tels que le contrôleur *OpenStack* [64] ou *OpenDaylight* [65]. Depuis, les chercheurs explorent toutes les possibilités permettant de tirer profit des avantages de SDN pour améliorer les performances et faciliter la gestion des architectures des réseaux de véhicules actuelles.

Dans le reste de cette section, nous commençons par présenter le principe et les caractéristiques de paradigme de SDN dans la sous-section 2.5.1, avant de décrire son architecture logique et les interfaces de communication en particulier celle d'*OpenFlow* dans les sous-sections 2.5.2, 2.5.3 et 2.5.4, respectivement. Ensuite, nous citons dans la sous-section 2.5.5, les apports de l'application d'un tel paradigme dans les réseaux de véhicules. Enfin, nous abordons dans la sous-section 2.5.6, une discussion taxonomique sur les efforts d'intégration de SDN dans les réseaux de véhicules et dans la sous-section 2.5.7, nous citons les principaux défis qu'il faut surmonter pour assurer une application efficace de SDN dans les VANETs.

2.5.1 Principe et caractéristique

Le concept de réseau défini par logiciel, largement connu sous le nom de *Software-Defined Networking* (SDN), est un nouveau paradigme émergent d'architecture réseau principalement basé sur : *(i)* une séparation physique entre le plan de contrôle (*i.e.*, les fonctionnalités qui assurent la gestion du réseau) et le plan de données (*i.e.*, les fonctionnalités qui assurent le transfert des données), et *(ii)* un contrôle et une intelligence logiquement centralisés dans un ou plusieurs contrôleurs logiciels. Dans SDN, les contrôleurs détiennent une vue globale sur tout l'état de réseau et gèrent les autres équipements de plan de données de réseau. Ces derniers deviennent de simple transmetteur/récepteur des données avec une intelligence minimale. SDN promet d'apporter la flexibilité, l'évolutivité et la programmabilité aux architectures des réseaux de véhicules de nos jours. Ils facilitent également la gestion de réseau et introduisent de nouveaux services [5].

2.5.2 Architecture

SDN est basé sur une architecture hiérarchique de trois couches, voir la figure 2.6 :

1) Couche du plan de données : le plan de données représente tous les équipements du réseau, souvent appelés, les équipements de diffusion, qui assurent uniquement l'envoi et la réception des données avec une intelligence minimale. Les équipements de diffusion exécutent aussi les actions du contrôleur.

2) Couche du plan de contrôle : le plan de contrôle représente tous les équipements du réseau, souvent appelés, contrôleurs SDN, qui centralisent l'intelligence du réseau et assurent la gestion des autres équipements de diffusion de plan de données.

3) Couche d'applications : elle regroupe tous les services et les applications des systèmes installés sur le contrôleur SDN.

2.5.3 Interface de communication

Pour permettre la communication entre les trois plans, SDN définit plusieurs interfaces de communication unifiées [66], voir la figure 2.6 :

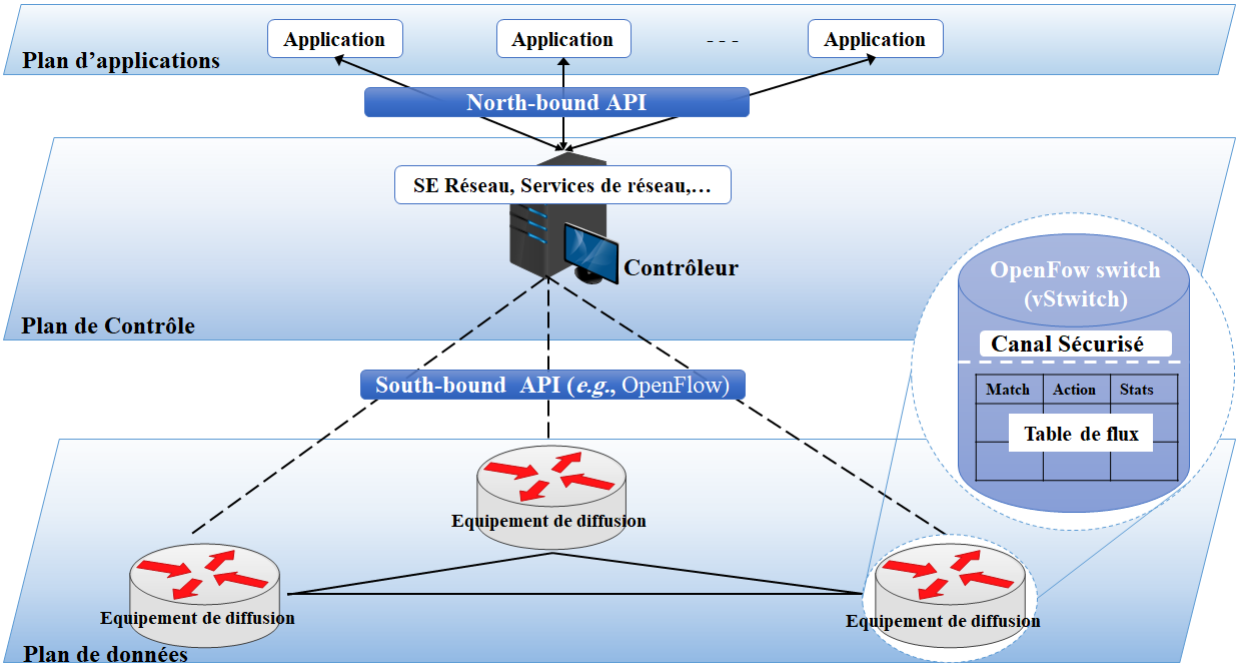


FIGURE 2.6 – Architecture du paradigme de SDN/OpenFlow.

- North-bound API : elle permet la communication et l'échange des données entre le contrôleur SDN sur le plan de contrôle et les applications du réseau. Le type d'informations échangées ainsi que leurs formes et fréquences dépendent de chaque application du réseau. Il n'y a pas de standardisation pour cette interface.
- South-bound API : elle fait référence aux différentes APIs qui permettent la communication entre les équipements de plan de contrôle et ceux du plan de données. *OpenFlow* [67] est le standard le plus utilisé pour cette interface.

2.5.4 OpenFlow

OpenFlow est un standard de communication pour SDN géré par l'ONF (*Open Networking Foundation*) [68], une fondation d'industriels dont la principale mission est de concevoir et promouvoir des équipements de réseau compatibles avec SDN afin de faciliter sa commercialisation. OpenFlow définit deux types d'équipements du réseau : les contrôleurs OpenFlow, un logiciel qui centralise toutes les fonctions de contrôle du réseau et OpenFlow vStwichs, qui sont des commutateurs virtuels qui n'effectuent que des fonctions de transfert des paquets de données. Chaque vSwitch possède une table de flux contenant des entrées de flux du contrôleur. En effet, le contrôleur gère les vStwichs en installant des règles de flux dans la table de flux. Comme le montre la figure 2.6, un canal sécurisé est utilisé pour connecter le contrôleur et les commutateurs virtuels correspondants.

Dans la sous-section suivante, nous citons les principaux avantages qui peut offrir l'intégration de SDN aux architectures des réseaux de véhicules.

2.5.5 Apports de SDN pour les réseaux de véhicules

Parmi les principaux apports et avantages qui peuvent apporter l'intégration de SDN dans les réseaux de véhicules, nous citons les suivants :

- Faciliter une gestion efficace du réseau : grâce à la séparation des fonctionnalités de contrôle et celles de diffusion des données et le mode de contrôle logiquement centralisé, l'intégration de SDN dans les réseaux de véhicules permet de simplifier l'architecture de réseau et offrir par conséquent une gestion plus flexible et une configuration plus rapide du réseau. En effet, au lieu de configurer et gérer chaque véhicule individuellement, toute la gestion de réseau est centralisée dans le contrôleur SDN. De plus, la vue globale du contrôleur SDN sur tout l'état du réseau offre une gestion flexible en temps réel des ressources du réseau ce qui permet l'adaptation dynamique et rapide aux changements fréquents des conditions de réseau [17], [22].
- Exploiter plus efficacement les ressources du réseau : avec la vue globale en temps réel du contrôleur SDN sur toute la topologie du réseau et le mode de contrôle centralisé, il devient facile d'identifier l'état actuel du trafic, ce qui rend l'allocation, de tous types de ressources du réseau, plus efficace (*i.e.*, bande passante, spectre, puissance de transmission, etc.). De plus, il serait plus facile de prendre de meilleures décisions plus informées selon l'état des ressources du système en se basant sur une combinaison de différentes sources. Par conséquent, les performances des réseaux de véhicules traditionnels peuvent être améliorées et les protocoles existants mieux exploités. Cela, en adaptant le déploiement des solutions les plus appropriées par l'ajustement de leurs paramètres en fonction de chaque situation spécifique et par rapport au contexte actuel [6], [20], [22].
- Faciliter le passage à l'échelle et réduire les coûts : le mode de contrôle logiquement centralisé de SDN permet de faciliter le déploiement des services et la proposition de nouvelles fonctionnalités à une grande échelle avec un coût minimal. Cela, en intervenant tout simplement au niveau du contrôleur SDN au lieu d'agir au niveau de chaque véhicule séparément. Par exemple, le contrôleur SDN peut facilement gérer une mise à jour logiciel à grande échelle en installant simplement des entrées dans les tables de flux des véhicules à distance [69], [22].
- Accélérer l'innovation et introduire de nouveaux services plus avancés : avec l'aide de la programmabilité et les interfaces de communication unifiées et reconfigurables de SDN, les équipements du réseau deviennent ouverts et configurables avec des logiciels indépendants des fournisseurs. Ce qui va encourager l'innovation et accélérer le développement de nouvelles solutions plus évoluées [20], [17].
- Mieux supporter l'abstraction et le découpage de réseau : avec la virtualisation et l'abstraction de SDN, le contrôleur SDN peut créer plusieurs réseaux virtuels dédiés de bout en bout sur la même infrastructure physique partagée de réseau de véhicules. Ces réseaux virtuels sont ensuite personnalisés pour répondre aux besoins spécifiques des services, des véhicules, ou des passagers. Ainsi, de nouveaux protocoles peuvent être facilement proposés et testés dans un environnement déjà opérationnel [20].
- Mieux gérer l'hétérogénéité : avec la virtualisation des fonctionnalités de réseau et l'abstraction des interfaces de communications promis par SDN, tous les véhicules, les RSUs et les infrastructures sans fil peuvent être considérés comme des commutateurs SDN virtuels gérés via une interface standardisée et unifiée. Ce qui peut

simplifier la gestion, la configuration et l'intégration de nouvelles technologies hétérogènes dans les réseaux de véhicules hétérogènes [20], [70].
– Améliorer les services de sécurité et réagir plus rapidement aux situations d'urgence : la flexibilité de SDN permet de réagir plus rapidement aux situations d'urgence et aux évènements soudains. Cette caractéristique est très appropriée pour gérer les services de sécurité routière [6]. De plus, le contrôleur SDN peut réserver d'une manière plus informée et dynamique des ressources (canaux et fréquences) pour l'utilisation exclusive des services de sécurité en se basant sur l'état actuel du trafic et sur les exigences des applications. Ceci est effectué en appliquant différents niveaux de priorités avec une plus grande priorité pour les services de sécurité.

Dans ce qui suit, nous discutons une taxonomie des efforts d'intégration de SDN dans les réseaux de véhicules.

2.5.6 Réseaux de véhicules définis par logiciel : taxonomie et discussion

Récemment, les chercheurs explorent les manières de tirer profit des avantages de SDN pour améliorer les performances des réseaux de véhicules actuels. Plusieurs architectures basées sur SDN ont été proposées dans la littérature pour les réseaux de véhicules communément appelées, *Software-Defined Vehicular Network (SDVN)*.

Sur la base du nombre de contrôleurs SDN et le mode de contrôle utilisé, nous pouvons classer les architectures SDVNs en deux grandes catégories :

i. Architectures SDVNs complètement centralisées (voir [6–16]) : le mode de contrôle dans ce type d'architectures est complètement centralisé autour d'un seul contrôleur SDN souvent installé quelque part sur l'infrastructure fixe pour contrôler tout le réseau, et
ii. Architectures SDVN distribuées (voir [17–34]) : le mode de contrôle dans ce type d'architectures est distribué et basé sur une hiérarchie de multi-contrôleurs SDN souvent installés sur l'infrastructure fixe, tel que chaque contrôleur SDN gère une partie du réseau.

Cependant, l'utilisation d'un mode de contrôle complètement centralisé autour d'un seul contrôleur SDN dans des réseaux aussi larges, dynamiques et denses que les réseaux de véhicules pose de sérieux problèmes pour plusieurs raisons. Un mode de contrôle centralisé risque de générer un long délai d'installation des règles de flux, en particulier lorsque la distance entre les véhicules et le contrôleur SDN dépasse les centaines de kilomètres. Il y a aussi un risque de goulot d'étranglement du contrôleur SDN face au grand nombre de requêtes des véhicules. De plus, il présente un risque omniprésent de fiabilité et de sécurité, en cas de défaillance du contrôleur surtout avec la nature intermittente et instable des connexions sans fil. Par conséquent, il semble bien évident que l'utilisation d'un mode de contrôle distribué avec plusieurs contrôleurs SDN est plus appropriée pour dealer avec les caractéristiques inhérentes des réseaux de véhicules.

De plus, certaines architectures dans les travaux [17–19] [25] [25–34] proposent d'installer les contrôleurs SDN à la périphérie de réseau aussi proches que possible des véhicules en bénéficiant des avantages du concept de *fog computing* afin d'assurer un délai de bout en bout acceptable et mieux répondre aux exigences des applications de sécurité et les services en temps réels sensibles aux délais. Certains travaux, dans [6], [17], [25], [26], [34], [69], utilisent aussi un système de récupération comme un mécanisme de secours pour assurer la disponibilité et la continuité de service en cas de défaillance d'un des contrôleurs SDN.

Le tableau 2.1 donne un bref aperçu taxonomique sur les travaux SDVNs selon les caractéristiques les plus significatifs, autrement dit, le mode de contrôle (distribué ou centralisé), le nombre de contrôleurs utilisés (un ou multiple), la tolérance ou non aux pannes du contrôleur SDN, la prise en considération ou non des zones non couvertes par l'infrastructure fixe, le support ou non des applications sensibles aux délais et l'environnement d'applications.

Néanmoins, comme on peut clairement le remarquer à partir du tableau 2.1, toutes les solutions proposées sont basées sur l'appui de l'infrastructure fixe véhiculaire comme un endroit privilégié pour déployer le/les contrôleur(s) SDN. Autrement dit, de telles architectures ne peuvent fonctionner que dans les zones couvertes par l'infrastructure. Cette hypothèse semble non réaliste, surtout avec les premières étapes de déploiement des larges topologies des réseaux de véhicules, où la couverture totale du réseau par l'infrastructure fixe est loin d'être atteinte, soit puisqu'elle est insuffisante, coûteuse ou non-profitable.

En guise de conclusion et sur la base de la discussion taxonomique des architectures SDVNs, nous pouvons en déduire que la tendance générale devrait être l'utilisation de plusieurs contrôleurs SDN distribués installés à la périphérie du réseau avec un mécanisme de récupération de secours efficace. De plus, les zones sans infrastructure non couvertes des réseaux de véhicules doivent être prises en considération lors de la conception des futures architectures des réseaux de véhicules basées sur le paradigme de SDN.

2.5.7 Défis face à l'intégration de SDN dans les réseaux de véhicules

Malgré les avantages et les apports que promet le paradigme de SDN pour améliorer les performances des réseaux de véhicules, l'intégration de SDN dans les réseaux de véhicules soulève un certain nombre de nouveaux défis dans la conception et la mise en œuvre du réseau de véhicules. Cela est dû principalement aux caractéristiques inhérentes des VANETs, telles que la grande mobilité des véhicules et la taille du réseau. Parmi ces défis, nous citons [19] :

– Faible latence : les applications liées à la sécurité routière et les services en temps réel des VANETs sont très sensibles aux délais et ont des exigences strictes en termes de latence. En effet, les paquets doivent être acheminés dans les meilleurs délais. De ce fait, le contrôleur SDN doit diminuer le temps de configuration et d'installation des règles de flux pour répondre aux exigences des applications sensibles aux délais.

Travail	**Mode de contrôle**		**♯ Contrôleur**	**Tolérance aux pannes**	**Sensible aux délais**	**Zone non couverte**	**Environnement**
	Centralisé	Distribué					
[6]	√		Un	√	-	-	VANET
[12], [13], [14]	√		Un	-	-	-	VANET
[7]	√		Un	-	-	-	Vehicular Cloud
[8]	√		Un	-	-	-	VANET
[9]	√		Un	-	-	-	VANET
[10]	√		Un	-	-	-	5G-VANET
[15]	√		Un	-	-	-	5G-VANET
[16]	√		Un	-	-	-	5G-VANET
[11]	√		Un	-	-	-	VANET
[19]		√	Multiple	-	√	-	HetVNet + Cloud
[20]		√	Multiple	-	-	-	HetVNet + Cloud
[21]		√	Multiple	-	-	-	HetVNet + Cloud
[69]		√	Multiple	√	-	-	VANET + Cloud
[17]		√	Multiple	√	√	-	VANET + fog computing
[18]		√	Multiple	-	√	-	IoV
[22], [24]		√	Multiple	-	-	-	VANET
[23]		√	Multiple	-	-	-	VANET
[25]		√	Multiple	√	√	-	VANET
[26]		√	Multiple	√	√	-	VANET
[30]		√	Multiple	-	√	-	VANET
[31]		√	Multiple	-	√	-	VANET
[32]		√	Multiple	-	√	-	VANET
[34]		√	Multiple	√	√	-	VANET
[27], [33]		√	Multiple	-	√	-	5G-VANET + fog computing
[28]		√	Multiple	-	√	-	5G-VANET
[29]		√	Multiple	-	√	-	VANET + Edge computing

TABLE 2.1 – Aperçu taxonomique sur les architectures SDVNs.

- Courte durée de vie des liaisons véhiculaires : la topologie de réseau hautement dynamique provoquée par la grande mobilité des véhicules peut entraîner une courte durée des liaisons entre les véhicules. De plus, les règles au niveau des tables de flux des véhicules peuvent devenir rapidement obsolètes. Par conséquent, les véhicules doivent consulter fréquemment le contrôleur, ce qui augmente la surcharge des messages de contrôle.
- Goulot d'étranglement du contrôleur : dans certains scénarios denses des VANETs avec un grand nombre de véhicules, un contrôleur peut être rapidement submergé face aux grands nombres des demandes des véhicules. L'utilisation et la coopération entre plusieurs contrôleurs doivent donc être envisagées pour régler ce problème [71].
- Diversité des solutions véhiculaires existantes : actuellement différents services et protocoles sont conçus pour les réseaux de véhicules. Avec l'arrivée de SDN, une question se pose : doit-on ignorer toutes les solutions des VANETs traditionnels et concevoir de nouvelles solutions compatibles avec SDN ou faut-il profiter de la vue globale de SDN pour mieux utiliser d'une manière dynamique et plus informée les solutions existantes selon le contexte ? Le choix est évident, la seconde proposition est évidemment la plus adaptée.
- Coût de déploiement : pour permettre à SDN de fonctionner dans les environnements véhiculaires, il faut utiliser des équipements compatibles avec SDN et ses caractéristiques révolutionnaires. En outre, il faut développer de nouveaux protocoles et solutions pour mieux exploiter les avantages de SDN. Un sérieux problème de coût se pose dans ce cas de figure. En effet, qu'est-ce qu'il faut faire de l'infrastructure réseau actuel et est-ce qu'il faut tout changer ?
- Sécurité et fiabilité : l'architecture SDN est basée sur un mode de contrôle centralisé en utilisant souvent un seul contrôleur SDN déployé sur l'infrastructure fixe d réseau. Le risque d'une panne du contrôleur ou même d'une perte de liaison est omniprésent surtout avec la nature intermittente de la liaison sans fil et la grande mobilité des véhicules. La fiabilité du système dans ce cas peut être rapidement fragilisée. De plus, le risque de sécurité est très sérieux dans un tel système avec un mode de contrôle complètement centralisé en particulier lorsqu'un seul contrôleur est utilisé.

2.6 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base des réseaux de véhicules. Ensuite, nous avons décri un bref aperçu sur leurs évolutions, initialement vers les réseaux hétérogènes et plus récemment vers l'Internet de véhicules. Plus loin, nous avons cité les principaux défis et challenges propres aux réseaux de véhicules actuels. En fin, nous avons donné un bref aperçu sur le paradigme de SDN dans les réseaux de véhicules.

Étant donné les avantages de SDN présentés ci-dessous, nous pensons fermement que SDN est un choix prometteur pour combler les limitations des réseaux de véhicules. Dans le chapitre suivant, nous allons présenter une nouvelle architecture basée sur SDN pour les réseaux de véhicules hétérogènes.

CHAPITRE 3

ARCHITECTURE DÉFINIE PAR LOGICIEL POUR LES RÉSEAUX DE VÉHICULES HÉTÉROGÈNES.

3.1 Introduction

Depuis déjà de nombreuses années, les véhicules sont devenus une partie importante de l'expérience de voyage des gens. Dans ce contexte, les HetVNets ont suscité de nombreux intérêts dans le but d'améliorer la sécurité sur les routes, le confort et l'efficacité des déplacements.

De nos jours, les architectures des HetVNets sont rigides, difficiles à gérer et souffrent d'un manque de flexibilité dans la gestion et de d'évolutivité dans le déploiement des solutions à grande échelle [6]. Récemment, plusieurs travaux (voir [6–34]) ont manifesté un intérêt croissant à tirer profit des avantages du paradigme SDN pour faciliter la gestion des réseaux de véhicules et apporter de nouvelles solutions innovantes aux défis de leurs architectures. Ces travaux ont démontré que SDN peut apporter la flexibilité, l'évolutivité et la programmabilité aux architectures des réseaux de véhicules. Cependant, la majorité des architectures basées sur SDN pour les réseaux de véhicules ne sont pas totalement abouties et il reste beaucoup de défis à relever tels que la fiabilité dans les zones non-couvertes par l'infrastructure fixe, la gestion de l'hétérogénéité ou la résistance aux pannes des contrôleurs. Dans cette optique, nous proposons dans ce chapitre une nouvelle architecture semi-centralisée pour intégrer le paradigme de SDN dans les HetVNets.

Le suite de ce chapitre est organisé comme suit. Tout d'abord, nous présentons, dans la section 3.2, nos motivations et la problématique. Dans la section 3.3, nous détaillons l'architecture basée sur SDN pour les HetVNets. Ensuite, au niveau de la section 3.4, nous développons le mécanisme de récupération de secours. Puis, nous mettons, au sein de la section 3.5, l'accent sur quelques cas d'applications de l'architecture proposée. Enfin, nous présentons les résultats de la simulation dans la section 3.6 avant de conclure en section 3.7.

3.2 Problématique et motivations

De nos jours, les architectures des HetVNets souffrent des problèmes d'évolutivité car il est très difficile de déployer des solutions à une grande échelle. Ces architectures sont rigides, difficiles à gérer et souffrent d'un manque de flexibilité et d'adaptabilité dans le contrôle [6]. Par conséquent, il est difficile de choisir la solution adéquate, en fonction du contexte actuel, en raison de la diversité des environnements de déploiement et de la vaste variété des solutions. Ces contraintes limitent les fonctionnalités du système et conduisent souvent à une sous-exploitation des ressources du réseau. Ainsi, la nécessité de nouvelles architectures plus flexibles et évolutives devient une exigence absolue pour faire face aux nouveaux besoins des prochaines générations des réseaux de véhicules.

Au cours des dernières années, le paradigme de SDN a été proposé comme solution innovante pour gérer les réseaux à grande échelle tels que les HetVNets. En effet, plusieurs travaux (voir [6–34]) ont démontré qu'en séparant le plan de contrôle de celui des données et qu'en adoptant un mode de contrôle centralisé, SDN peut apporter la flexibilité, l'évolutivité et la programmabilité aux architectures des HetVNets. SDN permet également d'utiliser plus efficacement les ressources de réseau et d'introduire de nouveaux services.

À partir de la discussion sur les travaux SDVNs au niveau du chapitre précédent (voir la sous-section 2.5.6 du chapitre 2), nous avons conclu que la tendance générale lors de la conception de nouvelles architectures basées sur SDN pour les réseaux de véhicules, est d'utiliser plusieurs contrôleurs SDN dont une partie doit être déployée à la périphérie du réseau afin de mieux répondre aux exigences des services de sécurité. Il est également primordial de prévoir un mécanisme de récupération de secours en cas de la défaillance du contrôleur SDN. De plus, nous avons remarqué à partir de l'aperçu taxonomique dans le tableau 2.1 de la même sous-section du chapitre 2, que la quasi-majorité des solutions SDVNs sont basées sur des architectures dont le fonctionnement dépend totalement du support de l'infrastructure fixe, l'endroit où les contrôleurs SDN sont souvent hébergés. Ces efforts ignorent l'existence des zones véhiculaires non couvertes. Une hypothèse très discutable surtout avec les premières étapes de déploiement des réseaux de véhicules, où la couverture totale du réseau est loin d'être garantie comme c'est souvent le cas dans les tunnels ou certaines zones rurales où le déploiement de l'infrastructure fixe est difficile, coûteux ou non-rentable.

Motivée par cette discussion, nous proposons dans ce qui suit une architecture semi-centralisée basée sur SDN pour les HetVNets avec des contrôleurs hiérarchiques et un mécanisme de récupération de secours fiable. Notre architecture peut fonctionner non seulement dans les zones couvertes par l'infrastructure fixe, mais également dans les zones non couvertes. En plus, l'architecture proposée supporte des communications hétérogènes : véhicule à véhicule (V2V), véhicule à infrastructure (V2I), infrastructure à infrastructure (I2I) et infrastructure vers Cloud (I2Cloud).

3.3 Architecture définie par logiciel pour les réseaux de véhicules hétérogènes

Dans cette section, nous présentons une nouvelle architecture semi-centralisée qui tire profit des avantages de la technique de *clustering* (le partitionnement de réseau) pour faciliter une intégration efficace du paradigme de SDN dans les réseaux de véhicules hétérogènes, appelée *Cluster-based Software-Defined Heterogeneous Vehicular Networks (CSDHVN)*. En effet, le principe de la technique de *clustering* lorsque le chef de groupe (cluster head) centralise l'intelligence du groupe (cluster) et les membres du groupe (cluster members) détiennent une intelligence minimale, est légèrement similaire à celui utilisée par la logique de contrôle de SDN. Par ailleurs, le paradigme de SDN repose sur un mode de contrôle centralisé pour la gestion de réseau, ce qui semble impraticable dans les réseaux à grande échelle tels que le cas des HetVNets. Ainsi, nous proposons de combiner le SDN avec la technique de *clustering* afin de partitionner le réseau et attribuer à chaque partition un contrôleur SDN dédié. Chaque contrôleur SDN maintient une vue locale sur l'état du réseau de la zone qu'il contrôle. Une vue globale sur tout l'état du réseau peut être obtenue en échangeant les vues locales entre les contrôleurs SDN adjacents.

Notre architecture s'appuie sur une combinaison entre différents concepts innovants des réseaux pour améliorer les performances des HetVNets de nouvelles générations. Dans CSDHVN, nous proposons de : *(i)* exploiter la dense couverture et le haut débit des réseaux cellulaires (4G/5G), déjà existants et largement déployés, pour prolonger la faible couverture de l'infrastructure fixe véhiculaire ainsi que pour soutenir les communications de haut débit I2Cloud, *(ii)* utiliser le concept émergent de SDN avec des multi-contrôleurs SDN distribués et hiérarchiques pour introduire la flexibilité, la facilité de gestion et l'évolutivité au réseau, *(iii)* partitionner le réseau en plusieurs segments virtuels et affecter à chaque segment un contrôleur SDN dédié afin de bien maîtriser la surcharge du réseau, réduire les interférences, offrir plus de fiabilité et mieux gérer l'évolutivité, la densité et la mobilité. Aussi, le partitionnement du réseau le rendre plus stable, plus petit et moins dynamique pour un véhicule, *(iv)* exploiter les puissantes ressources de la technologie du *Cloud computing* pour augmenter les capacités de calcul et de stockage des contrôleurs SDN, et *(v)* bénéficier des faibles délais de réponse du concept de *fog computing* pour satisfaire les besoins des applications de sécurité et répondre plus rapidement aux services sensibles au délai. Cela, en déployant les contrôleurs SDN à la périphérie du réseau, physiquement proche des véhicules.

La principale contribution de l'architecture proposée, en plus de combiner entre différents paradigmes émergents de réseau (*i.e.*, SDN, *clustering*, *Cloud* et *fog computing*), est de combler le manque dans la littérature d'une architecture basée sur SDN complète qui fonctionne non seulement dans les zones couvertes par l'infrastructure fixe véhiculaires, mais aussi dans les zones non couvertes. En effet, CSDHVN est une architecture flexible qui fonctionne aussi bien que dans les zones couvertes que dans les zones non couvertes.

Le mode de contrôle, dans notre architecture CSDHVN, est semi-centralisé, logiquement centralisé mais physiquement distribué. CSDHVN peut bénéficier de l'évolutivité

et de la fiabilité de l'architecture distribuée tout en préservant la simplicité de la gestion centralisée. En effet, le mode de contrôle dans CSDHVN est centralisé dans les zones couvertes autour de trois types de multi-contrôleurs SDN hiérarchiques et multi-sauts distribué dans les zones non couvertes via des contrôleurs SDN déployés sur les véhicules chefs de groupes. Dans la version distribuée multi-sauts de CSDHVN, communément appelée *dSDiVN* pour *distributed Software Defined infrastructure-less Vehicular Network*, uniquement les communications V2V via IEEE 802.11p sont utilisées. Cette logique de contrôle dans les zones non couvertes peut être également utilisée dans les scénarios couverts comme un mécanisme de récupération de secours efficace si la connexion avec un des contrôleurs déployés sur l'infrastructure est perdue. Lors du passage d'un véhicule d'une zone couverte vers une zone non couverte et inversement, CSDHVN bascule entre le mode de contrôle centralisé et celui de multi-sauts distribué pour assurer une continuité transparente du service.

Dans ce qui suit, nous détaillons la description de l'architecture proposée.

3.3.1 Architecture physique du système

Comme il est illustré à la figure 3.1, l'idée de base de notre architecture CSDHVN peut être résumée dans les points suivants :

- Diviser la topologie de réseau selon la disponibilité ou non de la couverture de l'infrastructure fixe en deux grandes zones virtuelles : zones couvertes et non couvertes ;
- Organiser les véhicules selon des critères bien définis (la vitesse, la position, la direction, etc.) en deux groupes virtuels hiérarchiques : des groupes statiques de niveau 1 (N1), (appelé *zone définie par logiciel* ou *software-defined zone* (SD-zone)) et des groupes dynamiques de niveau 0 (N0), (appelé *domaine défini par logiciel* ou *software-defined domain* (SD-domain)) ;
- Elire un véhicule comme chef pour chaque groupe N0 ;
- Déployer sur chaque véhicule chef de groupe (communément appelé, *un véhicule contrôleur*) un contrôleur SDN dit *local* ;
- Déployer sur chaque station de base cellulaire (*e.g.*, eNodeB), l'équipement qui joue le rôle du chef des groupes N1, un contrôleur SDN dit *central* ;
- Déployer sur un serveur Cloudlet (un centre de données Cloud à petite échelle situé à la périphérie de l'Internet) un contrôleur SDN dit *global* ;
- Connecter les contrôleurs SDN adjacents (via l'interface X2 pour les contrôleurs SDN centraux et via IEEE 802.11p pour les contrôleurs SDN locaux) afin de créer une dorsale de contrôle pour le réseau et pouvoir appliquer des stratégies globales à une grande échelle ;
- Connecter le contrôleur SDN global avec les contrôleurs SDN centraux via l'interface filaire TLS (*Transport Layer Security*).

Dans notre architecture CSDHVN, les groupes statiques de niveau 1 sont formés lors de l'installation des stations de base cellulaires et ils sont gérés et mis à jour d'une manière centralisée en utilisant la version centralisée de l'algorithme de *clustering* LTE4V2X dans le travail [72]. Les groupes N1, ont une taille égale à la portée de la couverture radio d'une station de base cellulaire (*e.g.*, LTE/eNodeB) et regroupent tous les véhicules dans la

zone de couverture de la station de base. En outre, les groupes dynamiques de niveau 0, sont formés et mis à jour périodiquement en utilisant la version distribuée multi-sauts de l'algorithme de *clustering* LTE4V2X [73], que nous avons amélioré dans notre travail [48] afin de l'adapter à notre cas d'étude. En effet, dans les zones non couvertes, la route est divisée en segments virtuels de taille égale à 150 m chacun (la moitié de l'étendue de la zone de couverture radio expérimentale de l'interface IEEE 802.11p) pour garantir que les contrôleurs SDN adjacents sur les véhicules chefs de groupes soient toujours accessibles les uns aux autres, voir la figure 3.1. Chaque segment représente un groupe (cluster) virtuel qui regroupe tous les véhicules mobiles (appelés *véhicules mobiles définis par logiciel* ou *software-defined vehicles* (SD-vehicles)) qui roulent dans la même direction et se trouvent dans le même SD-domain. CSDHVN sélectionne comme chef de groupe, le véhicule ayant la plus longue durée de vie dans le SD-domain (*i.e.*, le temps de parcours le plus long du cluster). Le but principal étant d'augmenter la stabilité des groupes et de diminuer le nombre des opérations de réélections ainsi que celui des messages de contrôle résultats.

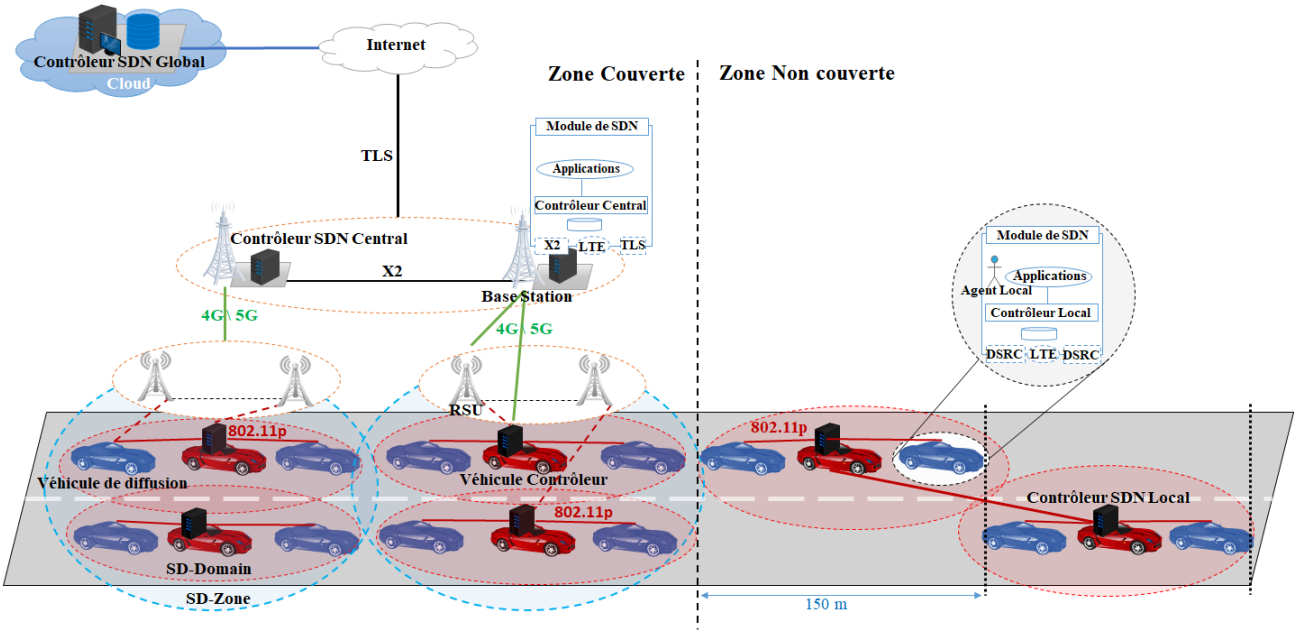


FIGURE 3.1 – Architecture physique de l'architecture semi-centralisée basée sur SDN pour les réseaux de véhicules hétérogènes.

Dans notre architecture, une station de base cellulaire compatible avec SDN est une station de base traditionnelle avec un module supplémentaire afin de pouvoir supporter les nouvelles fonctionnalités de SDN. Chaque station de base possède plusieurs interfaces de communication : *(i)* deux interfaces filaires : une interface X2 pour se connecter avec les stations de base adjacentes et permettre les communications de contrôle I2I entre les contrôleurs SDN centraux, et une interface Internet TLS qui est utilisée pour les communications de contrôle I2Cloud avec le contrôleur SDN global, et *(ii)* une interface cellulaire sans fil à large bande, utilisée pour les communications I2V entre les contrôleurs SDN centraux et les contrôleurs SDN locaux, voir la figure 3.1. Dans notre architecture, les contrôleurs SDN sont déployés sous forme de machines virtuelles installées sur un hyperviseur (une plateforme de virtualisation qui permet à plusieurs systèmes d'exploitation de fonctionner simultanément sur une même machine physique).

3.3.2 Architecture logique du système

CSDHVN est basée sur l'architecture logique à trois couches du paradigme de SDN, voir la figure 3.2 :

1) Couche du plan de données : elle est constituée de tous les équipements du réseau qui s'occupent uniquement de la collecte et de la transmission des informations, à savoir les véhicules mobiles membres des groupes de niveau 0 (appelés véhicules de diffusion), les RSUs et tous les équipements du réseau qui se trouvent entre la station de base cellulaires et le serveur Cloudlet. Dans notre architecture, la couche du plan de données est divisée selon la mobilité de ses composants en deux sous-couches :

a. Plan de données fixe : il est dit fixe, car il se compose des RSUs statiques et de tous les composants fixes du réseau qui assurent la transmission des données entre la station de base et le serveur Cloudlet.

b. Plan de données mobile : il est dit mobile, car il se compose des véhicules mobiles membres des groupes.

2) Couche du plan de contrôle : elle est constituée de tous les équipements du réseau qui hébergent les différents contrôleurs SDN. Ces derniers centralisent l'intelligence et le contrôle du réseau, autrement dit, les véhicules contrôleurs, les stations de base cellulaire (*e.g.*, eNodeB) et le serveur Cloudlet. La couche du plan de contrôle est divisée selon la mobilité de ses composants en deux sous-couches :

a. Plan de contrôle fixe : il est dit fixe, car il se compose de tous les contrôleurs SDN hébergés sur l'infrastructure fixe, autrement dit, le contrôleur SDN global déployé sur le serveur Cloudlet et les contrôleurs SDN centraux déployés sur les stations de base cellulaires.

b. Plan de contrôle mobile : il est dit mobile, car il se compose des contrôleurs SDN locaux déployés sur les véhicules contrôleurs.

Notre architecture CSDHVN s'appuie sur une hiérarchie de multi-contrôleurs SDN distribués afin d'alléger la tâche du contrôleur SDN global, en déléguant une partie des responsabilités générales à des contrôleurs moins puissants déployés à proximité des véhicules, et Aussi, pour réduire la surcharge du réseau et conserver la bande passante en n'envoyant aux contrôleurs de haut niveau uniquement des requêtes agrégées et compressées. En effet, CSDHVN définit trois types de contrôleurs SDN hiérarchiques :

i. Contrôleur SDN global : c'est le contrôleur SDN de niveau 2 déployé sur le serveur Cloudlet. Le contrôleur SDN global dispose d'une capacité de stockage et de calcul très puissante et détient une vue globale sur l'ensemble de la topologie du réseau. Il n'intervient que pour traiter les requêtes qui nécessitent une vue globale sur l'ensemble du réseau et/ou des ressources très importantes en termes de calcul et de stockage. Il traite aussi les requêtes qui ne peuvent pas être servies par les contrôleurs SDN centraux de niveau inférieur. Le contrôleur SDN global est considéré comme le maître de tous les contrôleurs SDN de réseau.

ii. Contrôleurs SDN centraux : ce sont des contrôleurs SDN distribués de niveau 1 déployés sur les stations de base cellulaires (*e.g.*, eNodeB). Ils sont dotés de puissantes capacités de stockage et de calcul. Les contrôleurs SDN centraux traitent les requêtes spécifiques qui nécessitent une vue centrale sur l'état du ré-

seau, de grandes ressources de calcul et de stockage, des opérations non sensibles aux délais ou qui ne peuvent pas être servies par les contrôleurs SDN locaux de niveau inférieur. Un contrôleur SDN central agit en termes de contrôle en tant qu'esclave du contrôleur SDN global, en tant que maître pour les contrôleurs SDN locaux et égal à égal par rapport aux autres contrôleurs SDN centraux voisins.

iii. Contrôleurs SDN locaux : ils représentent les contrôleurs SDN mobiles distribués de niveau 0 déployés sur les véhicules contrôleurs. Un contrôleur SDN local détient des capacités de stockage et de calcul modestes. Les contrôleurs SDN locaux s'occupent des requêtes d'ordre locales, les requêtes sensibles aux délais et les opérations qui ne nécessitent pas de grandes ressources de stockage et de calcul. Le contrôleur SDN local est considéré en termes de contrôle comme un maître des véhicules de diffusion de son groupe, un esclave de contrôleur SDN central et égal à égal par rapport à un autre contrôleur SDN local. Il est le seul contrôleur qui intervient dans les zones couvertes et non couvertes. Pour le meilleur de notre connaissance, ce type de contrôleur est nouveau et il n'a pas été encore abordé dans la littérature.

3) Couche du plan de services et d'applications : elle est constituée de toutes les applications du réseau et des services commerciaux installés sur les différents contrôleurs SDN tels que les services liés au routage ou à la sécurité.

4) Interfaces de communication : Actuellement, il n'y a pas des interfaces SDN standardisées qui peuvent être directement intégrées dans les HetVNets ni une version ad-hoc (non-IP) du protocole *OpenFlow* compatible avec les exigences des applications de sécurité routière des HetVNets. Pour cela, nous proposons d'utiliser une version personnalisée du protocole *OpenFlow* adaptée aux scénarios HetVNets comme un *API Southbound* pour communiquer entre le plan de contrôle et le plan de données, et utiliser une interface personnalisée comme une abstraction de haut niveau comme un *API Northbound* pour permettre la communication entre le plan de contrôle et celui des applications [20], voir la figure 3.2.

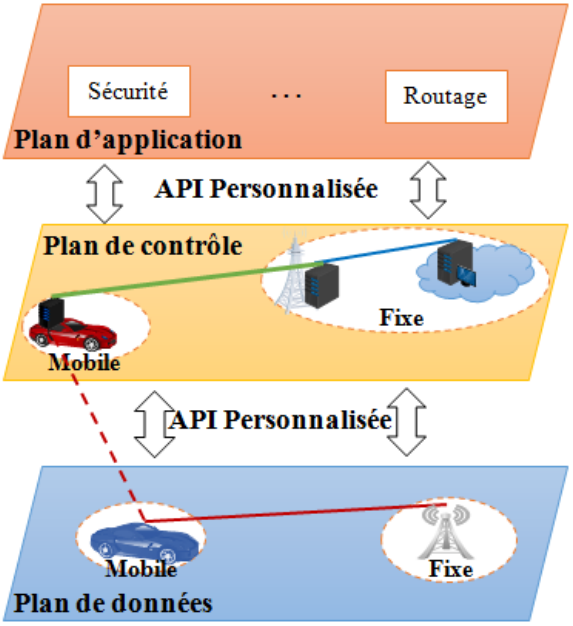


FIGURE 3.2 – Architecture SDN à trois couches de CSDHVN.

Dans ce qui suit nous détaillons l'architecture d'un véhicule compatible avec SDN.

3.3.3 Architecture d'un véhicule mobile défini par logiciel

Un véhicule mobile compatible avec le paradigme de SDN dans notre architecture (SD-vehicle), est un véhicule mobile traditionnel augmenté par un module supplémentaire, dit *module de SDN*. Il s'agit d'un ensemble de ressources matérielles et logicielles pour permettre au véhicule de prendre en charge et supporter les nouvelles fonctionnalités de SDN, voir la figure 3.3. Du point de vue matériel, il s'agit d'une unité de calcul et de stockage qui offre la plate-forme SDN pour installer les différents services et la table de flux. Pour limiter les modifications matérielles, il est préférable de réutiliser, dans la limite du possible, les ressources disponibles des OBUs. Du côté des logiciels, il comprend les composants SDN de base : un système d'exploitation pour SDN (SE SDN), des machines virtuelles sur l'hyperviseur, des services, etc. En particulier, un SD-vehicle dans notre architecture CSDHVN définit deux composants logiciels principaux, voir la figure 3.3 :

i. **Contrôleur SDN local :** il est initialement en mode 'en veille' et il est activé lorsque le véhicule est sélectionné comme chef de groupe. Il est dit mobile, car il est hébergé par un véhicule mobile et puisqu'il peut migrer d'un véhicule chef de groupe vers un autre. Le contrôleur SDN local centralise l'intelligence et contrôle tous les véhicules de son groupe.

ii. **Agent de surveillance et de collecte :** il assure la collecte et la transmission des données ainsi que la surveillance des paramètres du véhicule (*e.g.*, la position, la vitesse, la direction, etc.). Ces informations sont périodiquement communiquées au contrôleur SDN local. L'agent local de collecte et de surveillance reçoit et exécute les directives de contrôle à partir de son contrôleur SDN local via des entrées de règles de flux installées dans la table de flux de son véhicule.

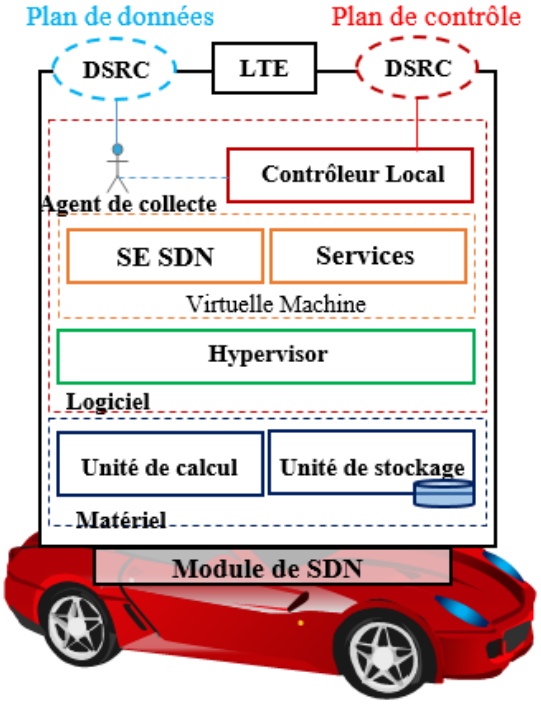


FIGURE 3.3 – Architecture interne d'un véhicule mobile défini par logiciel.

L'architecture d'un véhicule compatible avec SDN peut être implémentée au niveau de la couche *facilities* de la norme OSI CALM, qui suppose l'existence de plusieurs interfaces de communication sans fil au niveau d'un véhicule. En effet, dans notre architecture, chaque véhicule dispose de plusieurs interfaces de communication : *(i)* deux interfaces sans fil à large bande DSRC (IEEE 802.11p) afin d'éviter les interférences et séparer le trafic de contrôle de celui de données : une pour les communications de contrôle V2V entre les véhicules de diffusion et le contrôleur SDN local, et l'autre utilisée pour les communications de données I2V entre les véhicules de diffusion et le contrôleur SDN local, et *(ii)* une interface cellulaire sans fil à large bande (4G/5G) utilisée pour les communications de contrôle V2I entre les contrôleurs SDN locaux et les contrôleurs SDN centraux, voir la figure 3.3.

3.4 Mécanisme de récupération de secours

Dans une architecture avec un mode de contrôle entièrement centralisé telle que celle proposée par le paradigme de SDN, toute la fiabilité du système repose sur un seul contrôleur central. Par conséquent, il est nécessaire d'envisager un mécanisme de récupération de secours qui garantit la continuité du service en cas de défaillance de ce contrôleur central ou tout simplement en cas de perte de la connexion avec ce dernier [6]. Parmi les mécanismes de récupération de secours disponibles dans la littérature, les auteurs de [6], [17], [26], [34] et [69] ont proposé de basculer vers un protocole de routage traditionnel des VANETs si la connexion avec le contrôleur SDN est perdue. Cette solution risque d'augmenter la complexité de la conception du système [25]. Dans [20], les auteurs ont utilisé la prédiction de la trajectoire des véhicules pour préinstaller des entrées des règles de flux dans les tables de flux des véhicules. Ces dernières peuvent être utilisées plus tard si la connexion avec le contrôleur SDN est perdue. Cependant, cette initiative n'envisage aucune solution si la défaillance du contrôleur persiste après l'épuisement de la durée du validité des règles de flux. Dans [25], deux types de contrôleurs SDN hiérarchiques ont été utilisés et si le contrôleur SDN de haut niveau échoue, les contrôleurs SDN de niveau inférieur reprennent le relais pour assurer la continuité du service. En revanche aucune alternative n'est envisagée en cas de la défaillance d'un des contrôleurs SDN de bas niveau.

Notre architecture repose sur l'utilisation d'une hiérarchie de multi-contrôleurs SDN distribués. En tant que mécanisme de récupération de secours, nous proposons d'anticiper en permanence l'éventuelle défaillance de chaque contrôleur SDN à tous les niveaux et de préparer à l'avance le scénario de secours. Cela, en profitant d'une manière intelligente des avantages de la hiérarchie des contrôleurs SDN et des caractéristiques d'auto-organisation de la technique de *clustering*. En effet, dès qu'un véhicule est choisi comme chef de groupe, il commence par l'activation de son contrôleur SDN local. Une fois activé, le contrôleur SDN local commence à gérer une liste de candidature des contrôleurs SDN de secours afin d'anticiper sa potentielle défaillance et préparer le contrôleur SDN de remplacement. L'identifiant du meilleur candidat est périodiquement partagé avec tous les véhicules de diffusion de son groupe via l'installation d'une entrée de règles de flux. Pour cette raison, nous suggérons d'ajouter un nouveau champ dit de *secours* à la table de flux de chaque véhicule pour stocker l'identifiant. Une réplique (une copie) de la table de flux du contrôleur

SDN local déjà en service est périodiquement compressée et partagée avec le contrôleur SDN de secours afin de permettre une reprise rapide du service.

Dans le mécanisme de récupération de secours de CSDHVN, si un contrôleur SDN de niveau supérieur tombe en panne (niveau 2 (*resp.* niveau 1)), le système bascule au mode de contrôle distribué multi-sauts dans lequel les contrôleurs SDN du niveau strictement inférieur (niveau 1 (*resp.* Niveau 0)) reprennent le relais et collaborent entre eux pour assurer la continuité du service. Si un contrôleur de bas niveau (niveau 0) tombe en panne, le contrôleur SDN de secours pré-préparé reprend le relais pour assurer la continuité du service. Les véhicules de diffusion lorsqu'ils se rendent compte de la panne du contrôleur SDN peuvent désormais diriger leurs requêtes directement vers le contrôleur de secours dont l'identifiant est préalablement sauvegardé comme une mesure préventive dans leurs tables de flux. Le contrôleur SDN de secours, lorsqu'il sera activé, utilise la sauvegarde de la table de flux partagée auparavant pour commencer immédiatement à répondre aux demandes des véhicules.

Dans la section suivante, nous citons quelques cas d'applications possibles pour notre architecture CSDHVN.

3.5 Cas d'applications

L'intégration de SDN dans les HetVNets peut améliorer plusieurs applications des réseaux de véhicules, nous citons à titre d'exemple les applications suivantes :

- **SDN pour assurer une meilleure collecte et diffusion des données :** les données collectées au niveau de chaque véhicule ne sont pas utiles individuellement. Cependant, l'agrégation de toutes les données d'une zone géographique permet d'avoir une vue globale sur l'état du réseau. En effet, en séparant le plan de données du plan de contrôle, les contrôleurs SDN mobiles au niveau des véhicules contrôleurs peuvent centraliser la collecte des données, chacun dans son SD-domain. Contrairement aux réseaux de véhicules traditionnels, les contrôleurs SDN peuvent gérer les données collectées de manière plus informée pour extraire des informations utiles et améliorer les décisions du système. De plus, en collaborant les uns avec les autres, les contrôleurs SDN peuvent constituer une vue globale en temps réel sur tout l'état du réseau afin de choisir le chemin le plus optimal pour diffuser et acheminer les informations.
- **SDN pour assister les applications de sécurité routière :** d'un côté, notre architecture utilise des communications V2V via IEEE 802.11p, plus adaptées aux applications de sécurité [74]. De l'autre côté, la hiérarchie des multi-contrôleurs SDN en utilisant leurs vues globales sur l'état du réseau et les informations observées dans les conditions du trafic en temps réels peuvent collaborer entre eux pour : *i)* assurer une meilleure disponibilité des messages d'alerte, en définissant avec précision l'étendue de la zone de danger et la durée de vie de l'alerte, *ii)* choisir le chemin le plus rapide pour diffuser le message d'alerte, et *iii)* traiter le trafic d'urgence d'une manière plus prioritaire lors de la réservation des fréquences et des canaux.

- **SDN pour générer une cartographie globale et dynamique du réseau :** dans les réseaux de véhicules traditionnels, les véhicules collaborent entre eux et échangent différents types d'informations pour créer une cartographie locale et dynamique (*Local Dynamique Map*, LDM) sur l'état du réseau et les conditions aux alentours. Cette LDM comporte des informations statiques telles que les panneaux de la circulation et des informations dynamiques telles que l'état du trafic et les conditions météorologiques. L'intégration du SDN dans les réseaux de véhicules permet au contrôleur SDN de gérer d'une manière centralisée une cartographie dynamique globale (*Global Dynamique Map*, GDM) en collectant, combinant et filtrant de manière centralisée les LDMs distribués des différents véhicules. Cela permet au contrôleur SDN de construire une vue dynamique globale sur l'ensemble (ou une partie) de l'état du réseau, ce qui peut contribuer à prendre des décisions plus éclairées et aider à mieux affiner les réponses aux requêtes des véhicules.
- **SDN pour assister le délestage de calcul :** le délestage (*offloading*) de calcul consiste à transférer une tâche à une entité externe lorsqu'il n'est pas intéressant ou encore impossible de l'exécuter localement, en raison du manque de ressources nécessaires ou car l'exécution distante est plus bénéfique. Dans les réseaux de véhicules traditionnels, la décision de délestage repose sur une connaissance limitée sur l'état du réseau. En effet, cette décision repose, par exemple, sur des paramètres basés sur l'historique du réseau [15], qui ne peuvent pas refléter l'état actuel du réseau et aboutissent parfois à des décisions non optimales. En outre, dans un réseau de véhicules basé sur SDN, l'utilisation de la vue globale centralisée du contrôleur SDN peut aider à prendre de meilleures décisions de délestage de manière dynamique avec un meilleur raffinement, plus adapté aux besoins des utilisateurs et aux conditions de réseau en temps réel.
- **SDN pour une mise en cache distribuée efficace :** la vue globale centralisée et à jour des contrôleurs SDN sur l'état du réseau en temps réel peut être exploitée pour prédire les probabilités d'accès et par conséquent la popularité des contenus (audio, vidéo, etc.) en fonction de l'historique des requêtes des véhicules à chaque point du réseau. Ce qui peut aider à prendre des décisions plus informées, concernant quel contenu sera mis en cache, à quel emplacement et pour quelle période du temps.

3.6 Résultats de simulation

Dans cette section, nous présentons les résultats de simulation avec un scénario de simulation basé sur l'architecture du système CSDHVN décrite dans la sous-section 3.3.1. Nous avons utilisé le simulateur de réseau NS3 (Network Simulator 3) [75] et le simulateur de trafic SUMO (Simulation of Urban MObility) [76] pour implémenter notre architecture et réaliser les différentes expérimentations. L'objectif principal de cette série d'expérimentations est de démontrer la faisabilité, la fiabilité et l'efficacité de l'architecture proposée. Nous cherchons, également, à étudier l'impact de la distance de déploiement du contrôleur SDN sur le délai moyen d'installation des règles de flux et à évaluer la fiabilité de notre mécanisme de récupération de secours.

La simulation repose sur un simple scénario d'un réseau HetVNet, similaire à celui illustré à la figure 3.1. En effet, nous simulons un réseau HetVNet avec une topologie déployée sur une surface de 10000 x 10000 m qui contient une zone de la route non couverte par l'infrastructure fixe. Une station de base cellulaire (eNodeB) est placée au bord de la route à 1000 m des véhicules. La route est divisée en segments virtuels de taille égale à 150 m chacun. La densité des nœuds est de 200 véhicules. Chaque véhicule dispose de deux interfaces de communication sans fil : une interface DSRC (IEEE 802.11p) avec une portée de transmission allant jusqu'à 300 m et une interface cellulaire 4G. Les véhicules roulent à des vitesses comprises entre 15 et 25 m/s. Le temps de simulation est de 120 secondes et le taux de génération des paquets est de 5 paquets/s. Les principaux paramètres de simulation sont résumés dans le tableau 3.1.

Paramètre	**Valeur**
Taille de la zone de simulation	10000 x 10000 m
Distance de déploiement de l'eNodeB	1000 m
Taille d'un segment	150 m
Nombre des véhicules	200 véhicules
Vitesse de véhicule	15 – 25 m/s
Temps de simulation	120 s
Taux de génération des paquets	5 paquets/s

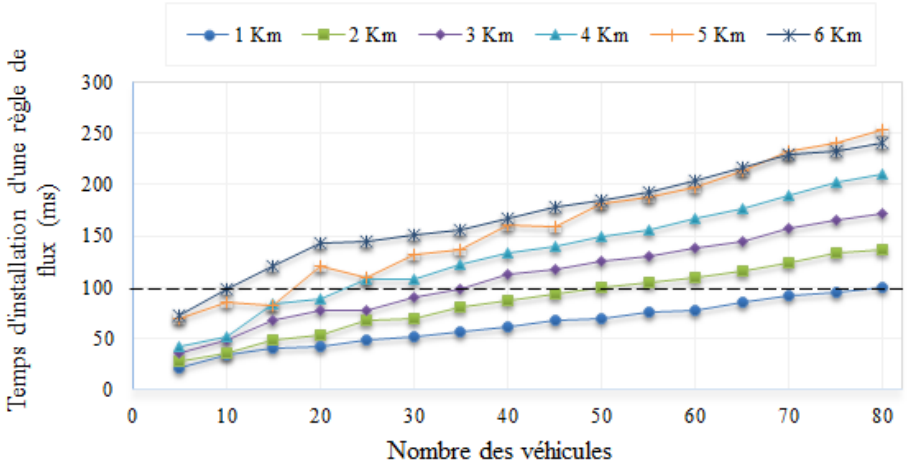
Table 3.1 – Paramètres de simulation.

Les mesures de performance que nous avons utilisées sont :
- Délai d'installation d'une règle de flux : il représente le temps écoulé depuis qu'un véhicule demande au contrôleur SDN une nouvelle règle de flux et le moment où la règle de flux est installée dans la table de flux du véhicule.
- Taux de délivrance des paquets : c'est le rapport entre le nombre des paquets reçus avec succès et le nombre total des paquets envoyés.

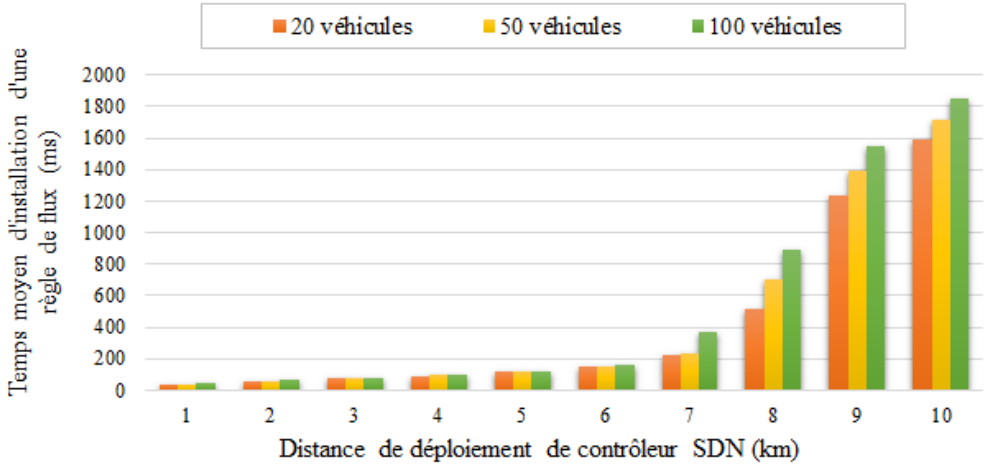
Dans ce qui suit nous détaillons la série des expérimentations réalisées.

3.6.1 Impact de la distance avec le contrôleur SDN sur le temps d'installation des règles de flux

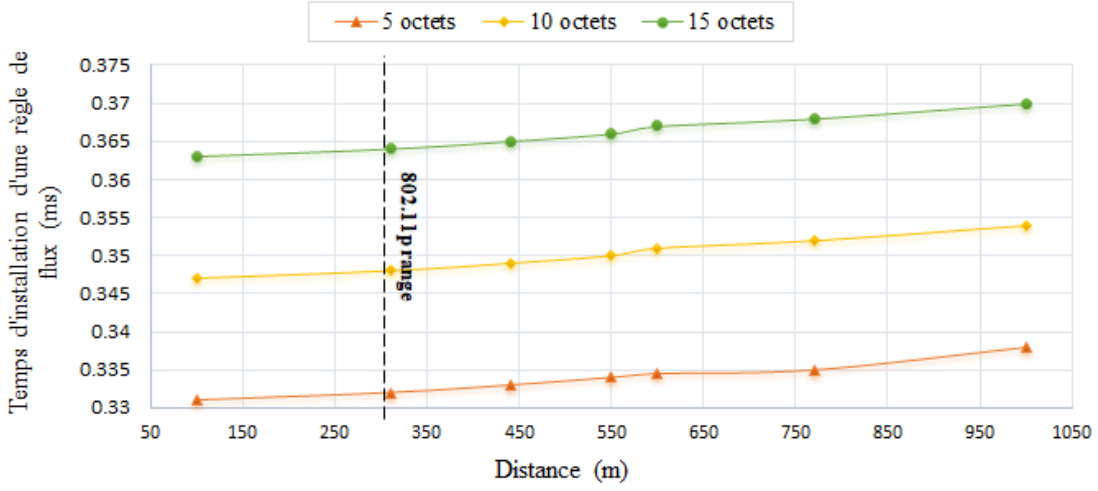
Dans cette expérimentation, nous étudions l'impact de la distance avec le contrôleur SDN, qui représente la distance physique entre le contrôleur SDN et les véhicules, sur le temps d'installation des règles de flux. Ainsi, nous simulons un simple scénario dans lequel un ensemble de véhicules envoient à la fois 100 paquets pour demander des nouvelles règles de flux à un contrôleur SDN et nous varions la distance entre le véhicule et le contrôleur SDN. Nous répétons cette expérimentation pour des tailles de paquets différentes et pour différents nombres de véhicules. Les résultats de simulation sont illustrés à la figure 3.4.



(a) Effet du nombre de véhicules sur le temps d'installation des règles de flux pour différents distances avec le contrôleur SDN.



(b) Effet de la distance avec le contrôleur SDN sur le temps d'installation des règles de flux pour differents nombre de véhicules.



(c) Effet de la distance avec le contrôleur SDN sur le temps d'installation des règles de flux pour différentes tailles de paquets.

FIGURE 3.4 – Impact de la distance avec le contrôleur SDN sur le temps d'installation des règles de flux.

Les résultats dans la figure 3.4 montrent clairement que le temps moyen d'installation des règles de flux augmente avec l'accroissement du nombre de véhicules, la taille des paquets ainsi qu'avec l'augmentation de la distance du déploiement de contrôleur SDN. Cela, peut-être justifié par le fait que lorsque le nombre de véhicules augmente les interférences augmentent et par conséquent le temps d'attente augmente. Aussi, lorsque la distance et la taille des paquets augmentent, il est clair que le temps nécessaire pour transmettre les paquets de bout en bout augmente.

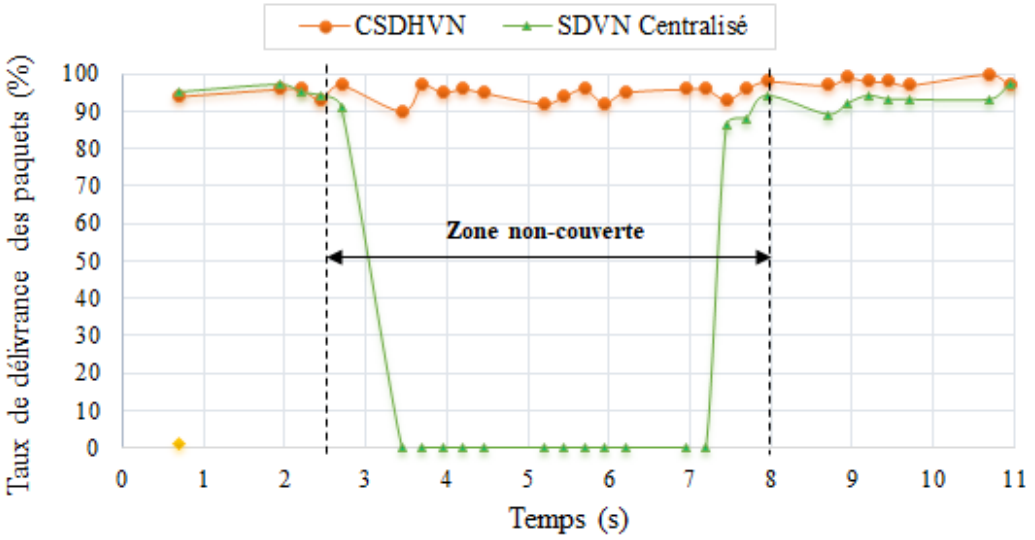
De plus, d'après les résultats dans les sous-figures 3.4(a) et 3.4(b), nous pouvons conclure que pour garantir un temps d'installation des règles de flux suffisant pour satisfaire les exigences d'une latence inférieure aux exigences de la plupart des applications de sécurité routière ($\leq$ 100 ms [6]) et répondre plus rapidement aux événements en temps réel dans des topologies de réseau aussi dynamiques et denses comme celles de HetVNet, le contrôleur SDN (ou une partie des contrôleurs SDN dans le cas de multi-contrôleurs distribués) doit être installé à la périphérie du réseau, le plus proche possible des véhicules, à environ 6 km pour les scénarios dispersés et pas plus de 2 à 3 km pour les scénarios denses. L'expérimentation dans la figure 3.4(c) est réalisée pour des petites distances de 10 km maximum. Cependant, les résultats obtenus peuvent être généralisés aux scénarios à grande échelle. Dans ce cas, si on considère que le système réagit de la même manière, les résultats peuvent être pires et le système peut ne pas garantir un délai de livraison des paquets assez suffisant pour satisfaire les exigences des services sensibles aux délais, plus particulièrement lorsque les distances atteignent des centaines de kilomètres comme il est souvent le cas dans les vastes topologies des HetVNets.

Notre architecture CSDHVN est basée sur des contrôleurs SDN hiérarchiques. Le dernier contrôleur SDN (niveau N0) dans la hiérarchie (le contrôleur SDN local) est déployé sur le véhicule contrôleur, à moins de 300 m des autres véhicules (la distance la plus éloignée avec un autre contrôleur SDN est de 150 m) et communiquent avec les autres véhicules du même groupe via des communications V2V en utilisant IEEE 802.11p (une technologie avec une faible latence [74]). Le second de niveau N1, le contrôleur SDN central, est déployé sur la station de base cellulaire (l'eNodeB) à environ 1 km de la route. Selon les résultats obtenus dans la sous-figure 3.4(c), cela suffit largement à garantir un temps d'installation des règles de flux adéquat pour satisfaire les exigences des applications de sécurité routière à faible latence et garantir une meilleure prise en charge des services sensibles au délai et les événements en temps réel.

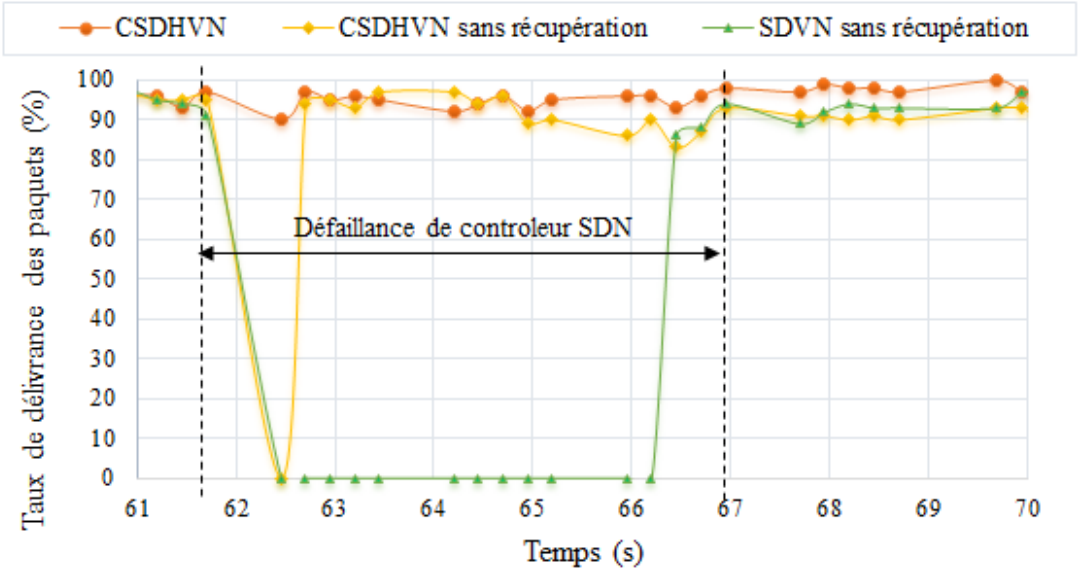
3.6.2 Impact de la défaillance du contrôleur SDN sur le taux de délivrance des paquets

Dans cette expérimentation, nous étudions comment notre architecture basée sur le paradigme de SDN avec le mécanisme de récupération de secours hiérarchique, réagit face à une défaillance du contrôleur SDN. Nous considérons deux cas : *(i)* le cas où le véhicule traverse une zone non couverte par l'infrastructure fixe, dans ce cas le contrôleur SDN est injoignable, et *(ii)* le cas où le contrôleur SDN tombe en panne pendant une certaine période du temps. Pour cela, nous simulons dans un premier scénario dans la sous-figure

3.5(a), une défaillance du contrôleur SDN qui dure 5 secondes à partir de la 61ième seconde et dans un second scénario dans la sous-figure 3.5(b), le passage d'un véhicule par une zone non couverte par l'infrastructure fixe véhiculaire. Par la suite, nous comparons les performances du mécanisme de récupération de secours de notre architecture (nous le notons, *CSDHVN*) avec la même architecture mais sans le mécanisme de récupération de secours (nous le notons, *CSDHVN sans récupération*). Nous le comparons également avec ceux des travaux SDVNs (voir [7–15]), basés sur un mode de contrôle centralisé avec des contrôleurs SDN déployés quelque part sur l'infrastructure fixe et qui n'utilisent aucun mécanisme de récupération du secours (nous le notons, *SDVN centralisé*) en fonction du taux de délivrance des paquets.



(a) Scénario avec une zone non-couverte.



(b) Scénario avec une défaillance de contrôleur SDN.

FIGURE 3.5 – Impact de la défaillance du contrôleur SDN sur le taux de délivrance des paquets.

À partir des résultats dans la figure 3.5, nous remarquons très clairement que juste après que la connexion avec le contrôleur SDN est perdue, soit à cause d'une panne de ce dernier ou parce que le véhicule vient d'entrer dans une zone non couverte, le taux de délivrance des paquets dans les travaux SDVNs centralisés sans récupération ou les SDVNs centralisés, commence immédiatement à chuter d'une manière considérable et il continue à baisser jusqu'à ce que le véhicule quitte la zone non couverte et que la connexion avec un des contrôleurs SDN soit rétablie (le contrôleur SDN reprend le service). Juste après, le système reprend un bon rapport de délivrance des paquets. Pareillement, dans le cas de CSDHVN sans récupération, le taux de délivrance chute mais brièvement, ce qui représente le temps nécessaire pour la réélection d'un nouveau chef de groupe et la synchronisation avec le contrôleur SDN nouvellement élu. Après cela, le système reprend un bon taux de délivrance des paquets. Beaucoup plus résistante, notre architecture CSDHVN subit un très léger effet et maintient un bon taux de délivrance des paquets même dans la zone non couverte ou sous une défaillance de contrôleur SDN.

Ce comportement peut être justifié par le fait que dans les solutions SDVNs centralisées sans mécanisme de récupération de secours (voir [7–15]), la fiabilité du système repose sur un mode de contrôle centralisé et dès que le contrôleur SDN tombe en panne, le système arrête l'installation des règles de flux. Par conséquent, le service sera interrompu juste après la fin de la durée de vie des entrées dans la table de flux. De manière analogue, dans CSDHVN sans récupération, le service sera interrompu pendant toute la phase de réélection du nouveau chef de groupe, après cela, il reprend un bon taux de délivrance. Plus efficace, notre architecture CSDHVN garantit un bon taux de livraison des paquets, que ce soit dans les scénarios avec une panne de contrôleur SDN ou dans les zones non couvertes sans infrastructure. Cela grâce aux contrôleurs SDN hiérarchiques multi-niveaux et en particulier grâce aux contrôleurs SDN locaux au niveau des véhicules contrôleurs qui reprend le relais pour assurer la continuité du service lorsque la connexion avec un des contrôleurs SDN sur l'infrastructure fixe est perdue. De plus, grâce à notre mécanisme de récupération de secours préventif (voir la section 3.4), basé sur le partitionnement du réseau avec une topologie à auto-organisation, l'anticipation des défaillances et la pré-préparation du contrôleur de secours, qui veille à assurer la disponibilité de service du contrôle. Ainsi, en cas de défaillance du contrôleur, les véhicules peuvent directement synchroniser et diriger leurs requêtes vers le contrôleur SDN de secours pré-sélectionné en utilisant l'identifiant pré-installé dans leurs tables de flux.

Cette expérimentation démontre la fiabilité et l'efficacité de notre architecture CSDHVN avec le mécanisme de récupération de secours par rapport aux autres travaux SDVNs basés sur l'appui de l'infrastructure fixe. Les résultats obtenus confirment aussi les résultats du travail de Ku et al. [6] : l'utilisation d'un mécanisme de récupération de secours est primordiale dans la conception des architectures SDVNs, en particulier avec un mode de contrôle complètement centralisé. Ces résultats confirment aussi que la prise en considération de l'existence des zones non couvertes est très importante vu l'impact négatif significatif qui peuvent représenter sur la fiabilité du système. Surtout, si on admet que la couverture complète de l'infrastructure fixe dans les HetVNets de nos jours n'est pas encore atteinte. Ce qui empire encore avec la nature intermittente des liaisons sans fil.

3.7 Conclusion

Nous avons présenté, dans ce chapitre une architecture semi-centralisée pour les réseaux de véhicules hétérogènes basée sur le paradigme SDN. Notre architecture est : *(i)* distribuée puisqu'elle utilise des multi-contrôleurs SDN hiérarchiques installés à la périphérie du réseau, *(ii)* flexible puisqu'elle fonctionne aussi bien dans les zones couvertes par l'infrastructure fixe que dans les zones non couvertes, et *(iii)* robuste grâce au mécanisme de récupération de secours hiérarchique. L'architecture proposée comble le manque d'architecture basée sur SDN pour les zones de la route non couvertes par l'infrastructure fixe véhiculaire.

Les résultats de simulation ont démontré la faisabilité et l'efficacité de l'architecture proposée comparé aux travaux SDVNs centralisés qui n'utilisent pas un mécanisme de récupération en cas de panne de contrôleur SDN. Ils ont aussi démontré la fiabilité de notre mécanisme de récupération de secours et l'importance de l'utilisation d'un tel mécanisme lors de la conception des architectures SDVNs afin de lutter contre les éventuelles défaillances du contrôleur SDN. De plus, les résultats ont montré l'important effet négatif que peut engendrer une grande distance du contrôleur SDN sur le temps d'installation des règles de flux.

La complexité et le coût de l'implémentation ainsi que la topologie clairsemée du réseau qui entraîne souvent le partitionnement du réseau constituent de sérieux défis face à l'intégration de SDN dans les environnements véhiculaires sans infrastructure. Comme perspective, nous prévoyons d'étendre notre architecture pour résoudre le problème de partitionnement du réseau en introduisant des mini-drones afin de connecter les contrôleurs SDN mobiles isolés.

Dans les deux chapitres suivants, nous explorons deux cas d'applications de notre architecture basée sur SDN. Le premier cas, dans le chapitre 4, concerne un scénario de secours pour un service de sécurité routière. Le deuxième, dans le chapitre 5, cas concerne la mise en cache du contenu dans les réseaux d'Internet de véhicules.

CHAPITRE 4

UN JEU SÉQUENTIEL POUR UN TRAITEMENT DES DONNÉES EFFICACE DANS LES RÉSEAUX DE VÉHICULES ASSISTÉS PAR DES DRONES.

4.1 Introduction

Dans les réseaux de véhicules, la couverture globale par une infrastructure de réseau fixe est difficile à assurer, que ce soit dans les zones sans infrastructure où la connectivité est absente ou lorsque le déploiement de l'infrastructure est difficile, coûteux ou peu rentable. Cette contrainte complique énormément la gestion du système et cause souvent le partitionnement de réseau.

Récemment, plusieurs travaux (voir [6–34]) ont utilisé SDN pour faciliter la gestion des VANETs. Par ailleurs, les drones (*Unmanned Aerial Vehicles*, UAVs) ont été employés comme une solution flexible pour aider les véhicules dans de nombreuses applications telles que les missions de secours et de recherche [35–38]. Les drones peuvent, en effet, être utilisés pour assister les véhicules dans l'exploration des tronçons de route dont l'accès est difficile à cause d'un accident par exemple, dans le but de collecter divers types de données tels que des photos et des vidéos aériennes. Ces données doivent être traitées le plus rapidement possible pour extraire des informations essentielles telles que le nombre et l'état des victimes dans un scénario d'accident. Le traitement de ce type d'information fait souvent appel à des techniques de reconnaissance de formes et de traitement de vidéos qui sont réputées être des tâches complexes à calcul intensif.

Dans ce chapitre, nous décrirons tout d'abord un scénario de sécurité routière relative à une mission de secours suite à un accident, dans lequel nous proposons d'équiper les véhicules de secours par des mini-drones pour les aider à explorer les tronçons de route touchés par un accident. Après, nous proposons une nouvelle architecture basée sur SDN pour faciliter la gestion des réseaux de véhicules sans infrastructure assistés par des drones. Ensuite, en se basant sur cette architecture, nous étudions une politique de traitement des données collectées par le drone comme un problème de prise de décision en matière de délestage (*offloading*)/partage de calcul des tâches entre le drone et les véhicules au

sol. Nous formulons ce problème de décision en tant qu'un jeu séquentiel à deux joueurs. L'objectif est d'atteindre le meilleur équilibre entre l'énergie et le délai de calcul.

Le reste de ce chapitre est organisé comme suit. Nous passons en revue les travaux connexes dans la section 4.2. Dans la section 4.3, nous décrirons le scénario de notre cas d'étude relatif à une mission de secours routière aidée par des drones, nous présentons l'architecture basée sur SDN pour les VANETs assistés par des drones, et nous discutons la motivation pour l'étude du problème de traitement des données. Nous abordons dans la section 4.4, la formulation du problème et nous décrirons le jeu séquentiel correspondant dans la section 4.5. La section 4.6 présente les résultats numériques. Finalement, la section 4.7 conclut le chapitre.

4.2 Travaux connexes

De nombreux chercheurs ont essayé, au cours des dernières années, de tirer profit des avantages du paradigme de SDN pour faciliter la gestion et améliorer les performances des architectures des réseaux de véhicules. Cependant, la quasi-totalité des architectures basées sur SDN pour les VANETs existantes dans la littérature (voir [6–34]) sont fondées sur le support de l'infrastructure fixe pour héberger leurs contrôleurs SDN. De nos jours, la couverture totale du réseau de véhicules par l'infrastructure fixe est loin d'être atteinte, même avec l'intégration de nouvelles technologies hétérogènes comme les réseaux cellulaires. En effet, des zones de réseau non couvertes existent toujours, par exemple dans certains endroits où la couverture n'est pas disponible ou certaines zones où l'infrastructure fixe est complètement absente car le déploiement est très difficile, onéreux ou peu rentable. De même, le risque d'une perte de connexion avec l'infrastructure fixe est omniprésent dans des topologies aussi dynamiques et vastes que celles des VANETs et avec la nature intermittente de l'interface sans fil. Cela représente un sérieux défi pour la fiabilité du système surtout avec la logique de contrôle centralisée de SDN. Par conséquent, les zones véhiculaires non couvertes par l'infrastructure doivent être prises en considération lors de la conception de nouvelles architectures basées sur SDN pour les VANETs.

Récemment, les drones ont été proposés comme une technologie révolutionnaire pour assister les prochaines générations des réseaux de véhicules dans les zones présentant une faible connectivité ou souffrant du problème de partitionnement du réseau. Dans [36], les auteurs ont proposé VDNet, un système qui exploite la flexibilité des drones pour améliorer la transmission des données dans les environnements véhiculaires clairsemés sans infrastructure. Oubatti et al. dans [35] et [77], ont proposé UVAR, un protocole de routage pour les VANETs assistés par des drones. Les auteurs de [37] ont utilisé les drones comme des relais pour connecter plusieurs segments de route isolés dans un réseau de véhicules. Dans [38], un système de multi-drones est proposé pour améliorer les performances des VANETs dans les environnements difficiles. Les drones ont été utilisés dans [78] et [79] en tant que nœuds aériens de *store-carry-forward* pour aider les véhicules au sol à réduire le délai de livraison des paquets ainsi que pour améliorer la connectivité en présence des véhicules non-coopératifs, respectivement. Sliem et al. dans [80] ont établi un protocole de routage pour les environnements véhiculaires clairsemés. Ils ont analysé le nombre opti-

mal de drones à déployer pour satisfaire la contrainte d'un meilleur délai de livraison des paquets entre le véhicule et le drone. Sharma et al. ont quant à eux proposé, dans [81], un système collaboratif pour les VANETs sans infrastructure assistés par des multi-drones comme une solution efficace qui permet le suivi des véhicules, l'analyse du comportement du conducteur et la détection des conducteurs en infraction, dans le but d'améliorer la sécurité sur les routes. Dans [82], une architecture basée sur SDN est suggérée pour prendre en charge des services hétérogènes dans un réseau de véhicule composé d'une infrastructure véhiculaire terrestre, de drones/de ballons aériens et de satellites spatiaux. En outre, les auteurs de [83] ont étudié le problème du placement de la station d'accueil des drones dans un système de transport intelligent dans le but de permettre au drone d'atteindre l'emplacement de l'incident dans un délai raisonnable ainsi que pour lutter contre le risque de l'épuisement de la batterie durant la mission d'exploration. De plus, les auteurs dans [84] ont étudié un problème de délestage de calcul dans un réseau de drones, où un drone peut délester le calcul des tâches intensives vers une station de base ou un serveur voisin afin d'équilibrer l'énergie et le délai. Dernièrement, d'autres travaux dans [85–88] ont proposé des architectures pour les réseaux de drones basées sur SDN afin d'améliorer les performances du plan de contrôle des drones, où ces derniers sont gérés à distance d'une manière centralisée par un contrôleur SDN terrestre. Tous ces travaux ont montré que l'utilisation des drones pour assister les véhicules peut améliorer le délai de livraison des paquets avec un coût raisonnable. Cependant, la quasi-totalité des travaux cités auparavant concentrent leurs efforts sur l'étude des problèmes relatifs à la transmission de données et aucun travail n'a exploré le problème de traitement des données.

Dans cette optique, nous investiguons dans ce qui suit comment tirer profit de la flexibilité, l'agilité et la facilité de déploiement des drones ainsi que de la simplicité et l'évolutivité de SDN pour successivement augmenter la couverture et faciliter la gestion des réseaux de véhicules sans infrastructure. Nous explorons également la manière d'assurer un traitement efficace des données collectées par le drone. À notre connaissance, il s'agit de la première initiative qui intègre le SDN dans les environnements véhiculaires sans infrastructure assistés par des drones.

4.3 Drones pour assister les véhicules dans l'exploration des zones inaccessibles

Dans cette section, nous commençons par décrire un scénario de sécurité routière relatif à une opération de secours routière aidée par des drones. Après, nous présentons une nouvelle architecture basée sur SDN pour les réseaux de véhicules assistés par des drones. Ensuite, nous discutons la motivation pour l'étude du problème de traitement des données collectées par des drones.

4.3.1 Missions de secours routière assistées par des drones

Récemment, les drones ont été proposés comme une solution flexible pour assister les véhicules dans des environnements sans infrastructure (voir [35–38], [81]). Principalement caractérisés par une grande fluidité et flexibilité, les drones peuvent assister de nombreuses

applications dans les réseaux de véhicules telles que les missions de recherche et de sauvetage. Généralement, lorsqu'un incident se produit sur la route, tel qu'un accident de la circulation ou n'importe quelle autre situation d'urgence, il en résulte souvent à un énorme embouteillage d'arrière qui peut parfois même causer le blocage complet de l'accès au tronçon de la route affecté. La contrainte qui peut retarder ou même empêcher les véhicules de secours (*e.g.*, les ambulances, les véhicules des équipes de sauvetage, etc.) d'accéder à la zone touchée dans les meilleurs délais. Motivé par cette discussion, nous proposons d'équiper les véhicules de secours par des mini-drones pour les assister dans l'exploration et l'investigation des tronçons affectés de la route dont l'accès est souvent difficile et parfois complètement impossible. En effet, face à une telle situation, les véhicules de secours peuvent envoyer une flotte de mini-drones pour explorer à distance la zone de la route touchée par l'accident, voir la figure 4.1. Étant capable d'assurer certaines des fonctionnalités de calcul aérien, le drone peut jouer le rôle d'une sonde aérienne afin de collecter différents types des données sur la zone affectée. Ces données ont besoin d'être traitées dans les meilleurs délais pour aider les équipes de secours à améliorer la qualité de l'intervention.

Dans ce qui suit, nous proposons de tirer profit des avantages du mode de contrôle centralisé de SDN (tel qu'il a été proposé dans les travaux [85–88]), afin de faciliter la gestion et le contrôle à distance des drones et mieux gérer les communications terrestres/aériennes du plan de contrôle entre les drones et le véhicule de secours au sol.

4.3.2 Architecture basée sur SDN pour les réseaux de véhicules assistés par des drones

Dans cette sous-section, nous présentons une nouvelle architecture distribuée basée sur SDN pour les réseaux de véhicules assistés par des drones. Cette architecture est principalement basée sur la version distribuée des zones non couvertes de l'architecture proposée dans le chapitre précédent. Ici, nous donnons uniquement les principales nouveautés et pour plus de détails référez-vous à la section 3.3 du chapitre 3. Comme il est illustré à la figure 4.1, notre architecture divise la route en plusieurs segments virtuels qui regroupent les véhicules qui roulent dans la même direction. Un contrôleur SDN local est déployé sur chaque véhicule chef de groupe (le véhicule contrôleur).

Dans note architecture, le véhicule de secours va jouer le rôle d'un contrôleur SDN principal terrestre pour gérer l'installation des requêtes et des paramètres des missions aériennes. Dans notre scénario d'étude, les drones sont considérés comme des dispositifs de collecte et de transmission qui assurent uniquement la surveillance des paramètres du drone (batterie, position, etc.) et la collecte et la transmission des informations. Un drone assez puissant (appelé *drone contrôleur*) va jouer le rôle d'un contrôleur SDN secondaire aérien pour une flotte de drones de transmission voisins, voir la figure 4.1. Le contrôleur SDN secondaire permet, de gérer plus efficacement et répondre plus rapidement aux requêtes de contrôle aériennes. Comme il permet d'optimiser l'énergie de transmission des drones, de réduire la surcharge des messages de contrôle et de minimiser les délais de réponse.

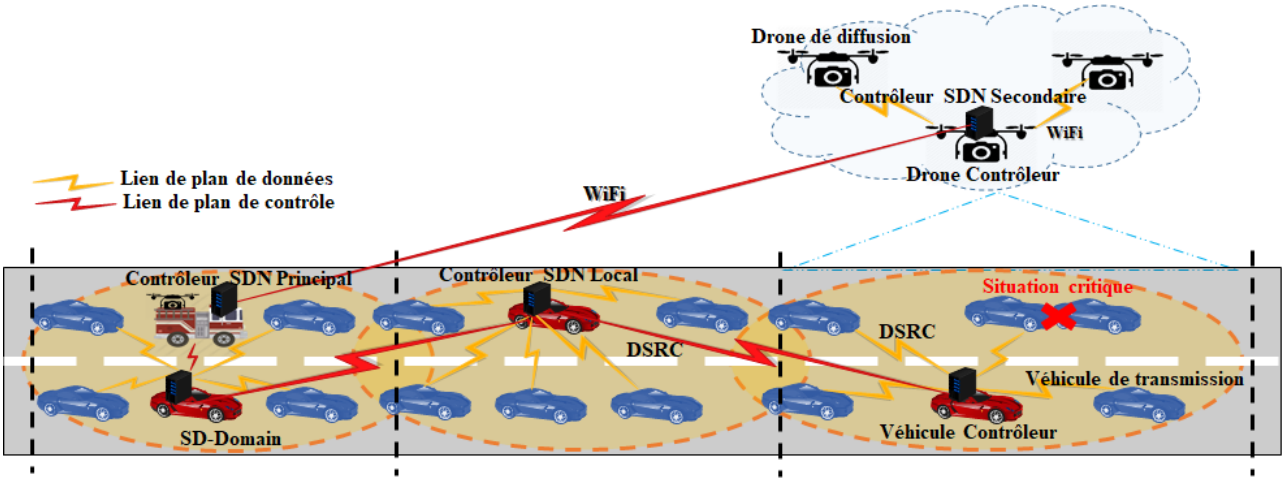


FIGURE 4.1 – Architecture basée sur SDN pour les réseaux de véhicules sans infrastructure assistés par des drones.

Comme le montre la figure 4.2, l'architecture logique du système proposé est basée sur l'architecture à trois couches de SDN :

1) **Couche du plan de données :** elle comporte tous les équipements du réseau qui assurent uniquement la collecte et la transmission des informations. Elle est composée d'une sous-couche du plan de données dite *terrestre* qui représente les véhicules de transmission et de collecte au sol et une sous-couche dite *aérienne* qui représente les drones de transmission et de collecte.
2) **Couche du plan de contrôle :** elle se compose de tous les équipements du réseau qui centralisent l'intelligence et contrôlent les autres équipements de transmission et de collecte. La couche du plan de contrôle est composée d'une sous-couche de plan de contrôle dite *terrestre* qui représente les véhicules contrôleurs au sol et une sous-couche dite *aérienne* qui représente les drones contrôleurs. Nous définissons trois types de contrôleurs SDN :
   a. **Contrôleur SDN local :** il est déployé sur les véhicules contrôleurs et communique avec les autres véhicules via des liaisons V2V en utilisant IEEE 802.11p. Le contrôleur SDN local gère les requêtes des véhicules de son groupe et détient des capacités de calcul moyennes.
   b. **Contrôleur SDN principal :** il est déployé sur le véhicule de secours. Il communique avec les autres véhicules via des liaisons V2V en utilisant IEEE 802.11p et avec les drones via WiFi. Le contrôleur SDN principal gère les requêtes de contrôle terrestres/aériennes entre les véhicules au sol et les drones. Il détient des capacités de calcul puissantes.
   c. **Contrôleur SDN secondaire :** il est déployé sur le drone contrôleur. Il communique avec les autres drones et les véhicules au sol en utilisant le WiFi. Le contrôleur SDN secondaire gère les requêtes aériennes de sa flotte de drones et détient de modestes capacités de calcul.
3) **Couche de services et d'applications :** elle regroupe tous les services et les applications du réseau.

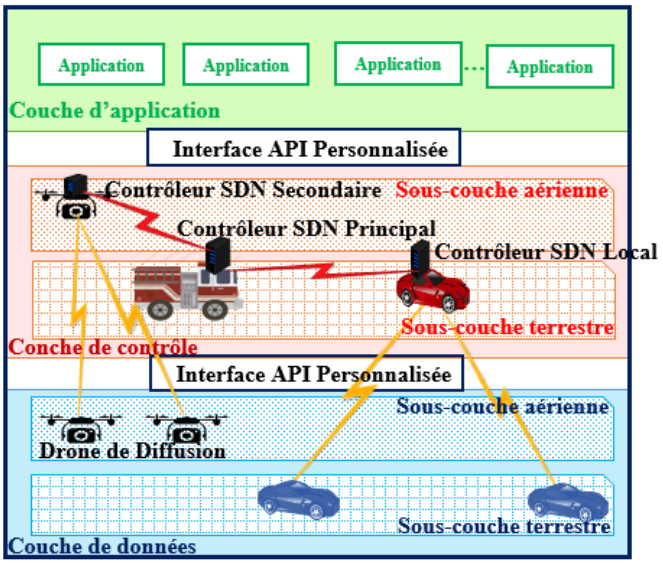


FIGURE 4.2 – Architecture SDN en trois couches du système.

4.3.3 Motivation pour l'optimisation du traitement des données

Dans les missions de secours sur les routes, les drones peuvent être utilisés pour aider les véhicules de secours à explorer et à investiguer les tronçons de la route dont l'accès est difficile. Les drones sont flexibles et peuvent offrir une couverture aérienne de la zone touchée par l'incident comme ils peuvent atteindre, survoler et collecter diverses informations et données sur cette zone à l'instar des vidéos et des photos aériennes. Vu la nature critique et urgente de la situation, ces données ont besoin d'être traitées le plus rapidement que possible. Un traitement rapide des données collectées par le drone permet, en effet, d'extraire des informations essentielles (par exemple : le nombre de véhicules endommagés et leurs immatriculations, le nombre de victimes et leurs identités, etc.) qui peuvent aider les équipes de secours à évaluer la gravité de la situation, estimer les dommages et anticiper d'autres renforts ainsi que pour préparer plus de logistique si nécessaire. Le traitement de ce type de donnés fait généralement appel à des techniques de reconnaissance des formes et de traitement des vidéos qui sont bien connues quand il s'agit de tâches à calcul intensif qui nécessitent des opérations de calcul complexes et consomment beaucoup de ressources en terme d'énergie et de CPU (*Central Processing Unit*). De plus, la taille des données collectées par le drone peut rapidement augmenter comme par exemple pour le cas des longues séquences des vidéos à haute qualité. Le calcul ce type de tâches intensives et la transmission de données volumineuses via un équipement à ressources limitées comme celui du drone peut engendrer une longue durée de calcul, un long délai de transmission et une consommation d'énergie élevée. Toutefois, dans les scénarios urgents comme celui des opérations de secours suite à un accident sur la route, le traitement des données collectées par le drone est très sensible au délai et revêt une importance vitale au point qu'il pourrait contribuer à sauver des vies humaines dans certaines situations. De plus, la durée de vie de la batterie du drone est déterminante pour le succès de l'opération de secours et la mission d'exploration.

Désormais, les drones peuvent collaborer avec les véhicules de secours au sol qui ont de puissante capacité de calcul et peu de contrainte énergétique pour exécuter et traiter efficacement les données collectées dans les plus brefs délais et avec un meilleur équilibre entre le temps de calcul et la consommation énergétique. A titre d'exemple, dans certains scénarios, il serait plus bénéfique pour le drone afin de conserver son énergie de traiter localement les données volumineuses avec une faible complexité de calcul et ne transférer au véhicule de secours au sol que les petites tailles des résultats au lieu d'envoyer l'intégralité de ces données volumineuses via son interface sans fil pour être traitées sur le véhicule de secours. Il est bien connu que l'interface radio est le composant qui consomme le plus d'énergie [95]. De plus, le véhicule de secours peut optimiser plus le délai de calcul des tâches complexes en partageant le calcul avec un véhicule contrôleur voisin.

Dans ce qui suit, nous explorons une nouvelle politique pour un traitement efficace des données sous forme d'un problème de décision en termes de délestage/partage de calcul.

4.4 Formulation du problème de traitement de données

Dans cette section, nous détaillons la politique du traitement des données dans le contexte d'une opération de secours dans les VANETs assistés par des drones. Nous commençons par présenter le modèle du système. Ensuite, nous détaillons le modèle de calcul. Puis, nous décrirons la fonction du coût de système.

4.4.1 Modèle du système

Le modèle du système décrit dans cette sous-section est basé sur l'architecture SDN présentée dans la section précédente. Pour la modélisation de notre politique de traitement des données, nous considérons un scénario composé d'un véhicule de secours (*Emergency Vehicle*, EV) équipé d'un mini-drone. L'EV est connecté à un ensemble B de m véhicules contrôleurs (*Controler Vehicle*, CV) voisins tel que $B = \{1, 2, \ldots, m\}$. Nous supposons que l'étendue de la zone de la route affectée est assez petite au point que le déploiement d'un seul drone sera suffisant pour explorer toute cette zone. Le drone est supposé être toujours accessible à un véhicule au sol via une connexion WiFi. Le véhicule de secours communique avec les autres véhicules via des communications V2V en utilisant DSRC (IEEE 802.11p), voir la figure 4.3. Ce scénario peut être facilement étendu pour prendre en charge des scénarios plus larges avec plusieurs drones.

Étant donné que nous considérons un réseau de véhicules sans infrastructure assisté par des drones, toutes les communications sont supposées être de type véhicule à véhicule. Nous supposons que le véhicule de secours est équipé de deux interfaces de communication : une interface WiFi (IEEE 802.11a) pour communiquer avec le drone et une interface DSRC (IEEE 802.11p) pour communiquer avec les véhicules contrôleurs voisins. Nous supposons aussi que toutes les communications sont fiables.

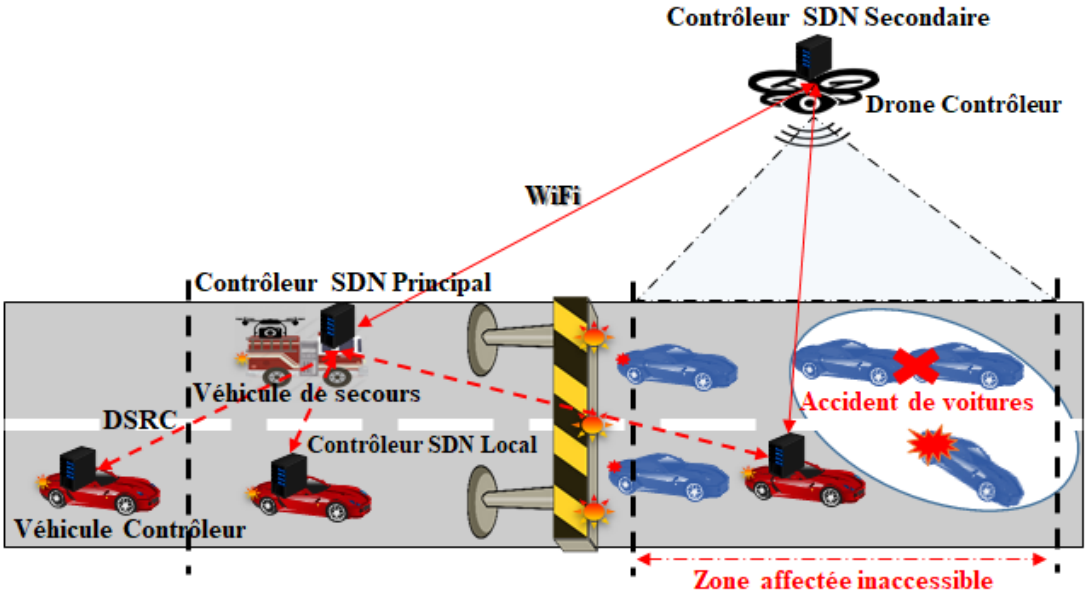


FIGURE 4.3 – Modèle du système de la politique de traitement des données pour le scénario de secours dans les VANETs assistés par les drones.

4.4.2 Description du modèle de calcul

Pour le modèle de calcul, nous supposons un système à instances (slots) temporelles pour décrire les différentes opérations de traitements de données. $T = \{t_1, t_2, \ldots, t_{|T|}\}$ dénote les instances temporelles considérées, alors qu'à chaque instant t_i, une seule tâche i est calculée. La durée de chaque instant est normalisée à l'unité de temps [10]. Puisque nous étudions une mission de secours suite à un accident sur la route avec un cas d'embouteillage de circulation, nous supposons un scénario quasi-statique, analogue à celui qui a été proposé dans les travaux [89–91], où le nombre de véhicules et la position des drones restent inchangés pendant un instant du temps lorsque la décision de calcul sera prise. Nous supposons également que chaque donnée collectée peut être divisée en plusieurs tâches de calcul atomiques indépendantes. Chaque tâche i peut être représentée par un triplet (C_i, Ds_i, Rs_i), $i \in N$, tel que C_i représente le nombre total des cycles CPU nécessaires pour calculer la tâche i, Ds_i indique la taille totale des données de la tâche (à savoir les paramètres d'entrée de calcul et le code de programme à exécuter) et Rs_i représente la taille des résultats de calcul de la tâche i [90].

Dans ce travail nous nous intéressons principalement à la manière de traiter le plus efficacement possible les données collectées par le drone. L'objectif principal étant de mieux équilibrer le temps de calcul des tâches des données à traiter et la consommation énergétique du drone. Le traitement de ces données et les tâches à calcul intensif correspondantes peuvent être *(i)* exécutés localement sur le drone par le contrôleur SDN secondaire, *(ii)* délestés en un seul saut via WiFi si le véhicule de secours se trouve dans la zone de couverture de l'interface WiFi du drone ou en multi-sauts initialement via WiFi et ensuite par DSRC pour être exécutés localement par le contrôleur SDN principal sur le véhicule de secours, ou *(iii)* partagés entre le contrôleur SDN principal et un des contrôleurs SDN locaux des véhicules contrôleurs voisins en utilisant DSRC, voir la figure 4.3.

Initialement, le drone recueille différents types de données (images, vidéos, etc.) sur la zone de la route touchée par l'incident, à laquelle le véhicule de secours ne peut pas accéder. Ensuite, le contrôleur SDN secondaire, après avoir exécuté la tâche i de traitement des données collectées par le drone, prend la décision de délestage qui est représentée par la variable binaire $d_i^o = \{0, 1\}$. Soit il choisit d'exécuter localement la tâche des données à calculer, dans ce cas $d_i^o = 0$, ou sinon il choisit de délester la tâche en utilisant le WiFi pour être calculer par le contrôleur SDN principal sur le véhicule de secours, dans ce cas $d_i^o = 1$. Si le contrôleur SDN secondaire décide de délester le calcul de la tâche, une fois celle-ci est reçue par le contrôleur SDN principal, il va prendre la décision de partager ou non le calcul de la tâche avec un de ses contrôleurs SDN locaux voisins. La décision de partage est représentée par la variable binaire $d_i^s = \{0, 1\}$. S'il choisit d'exécuter la tâche localement, alors $d_i^s = 0$, et s'il choisit de partager le calcul de la tâche avec un de ses contrôleurs SDN locaux voisins, alors $d_i^s = 1$, voir la figure 4.4.

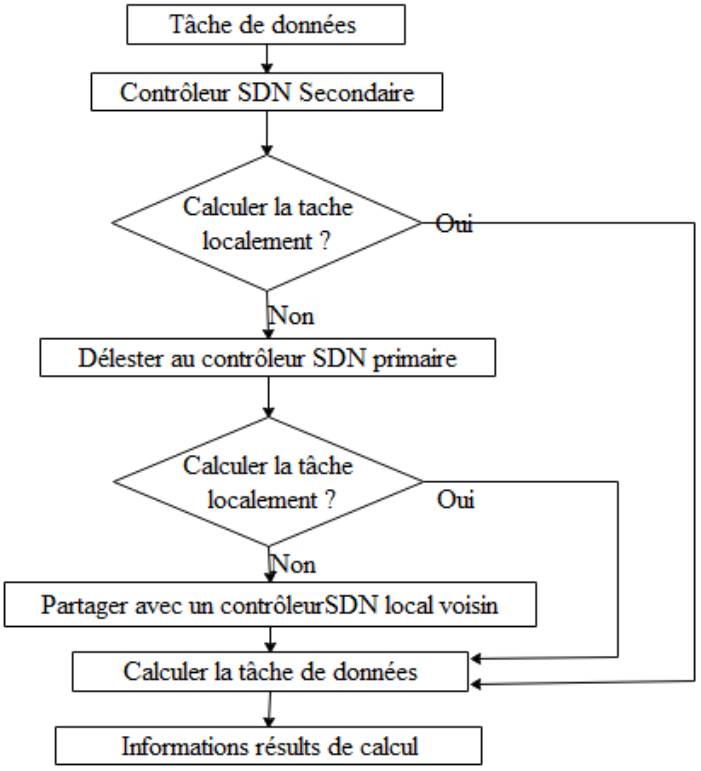


FIGURE 4.4 – Diagramme de flux de la politique de traitement des données.

Dans notre cas d'étude, le traitement des données collectées par le drone est une tâche sensible aux délais en raison de la nature critique et urgente de la situation (une mission de secours suite à un incident sur la route). De plus, la durée du vie de la batterie des drones est considérée comme un facteur déterminant pour le succès de la mission d'exploration. Ainsi, le délai de calcul et l'énergie consommée par le drone pour traiter les données, représentent les principaux paramètres de la prise de décision en matière de délestage/partage de calcul. Par ailleurs, nous considérons deux niveaux de prise de décision avec deux choix de décisions possibles pour chaque niveau : *(i)* le calcul par le contrôleur SDN secondaire avec un choix entre le calcul en local ou le délestage de calcul, et *(ii)* le calcul par le contrôleur SDN principal avec un choix entre le calcul en local ou le partage de calcul.

Dans ce qui suit, nous détaillons les formules de calcul des métriques de décision : le délai et l'énergie. Des formules similaires ont été proposées dans les travaux [89–91].

4.4.2.1 Sous-modèle de calcul du drone

Le contrôleur SDN secondaire lorsqu'il entame le traitement d'une donnée collectée par son drone, après estimation du délai ou de l'énergie nécessaire pour le calcul, choisit soit de calculer les tâches correspondantes localement, soit de les délester au véhicule de secours pour être calculer par le contrôleur SDN principal.

1) Calcul local par le contrôleur SDN secondaire : le délai total de calcul local de la tâche i par le contrôleur SDN secondaire sur le drone ($D_{l,i}^{Sc}$), est égal au temps de calcul local (T_i^{ls}) plus le temps d'envoi des résultats au contrôleur SDN principal (T_i^{Rs}), que ce soit en un saut en utilisant le WiFi si le véhicule de secours est dans la zone de couverture de l'interface WiFi du drone ou en mode multi-sauts en utilisant initialement le WiFi et après DSRC, dans le cas contraire. $D_{l,i}^{Sc}$ est calculé comme suit :

$$D_{l,i}^{Sc} = T_i^{ls} + T_i^{Rs} \tag{4.1}$$

Tel que

$$T_i^{ls} = \frac{C_i}{F_{CPU}^{UAV}} \tag{4.2}$$

Et

$$T_i^{Rs} = \frac{Rs_i}{R_{WiFi}^{UAV}} + \sum_{s=0}^{h} \frac{Rs_i}{R_{DSRC}} \tag{4.3}$$

F_{CPU}^{UAV} dénote la fréquence de calcul du contrôleur SDN secondaire exprimée en cycles de CPU par seconde, R_{WiFi}^{UAV} dénote le taux de transmission des données de l'interface WiFi et R_{DSRC} dénote le taux de transmission des données de l'interface DSRC. h est le nombre de sauts V2V nécessaires pour atteindre le véhicule de secours.

La consommation énergétique totale de calcul local ($E_{l,i}^{Sc}$) est égale à l'énergie de calcul local (E_i^{ls}) plus l'énergie nécessaire pour renvoyer les résultats (E_i^{Rs}). Nous supposons que les véhicules n'ont pas de contrainte énergétique. Par conséquent, nous ignorons l'énergie consommée par le véhicule pour le calcul local ainsi que pour l'envoi des données. (E_i^{Rs}) est donnée par :

$$E_{l,i}^{Sc} = E_i^{ls} + E_i^{Rs} \tag{4.4}$$

Tel que

$$E_i^{ls} = C_i \times e_{CPU}^{UAV} \tag{4.5}$$

Et

$$E_i^{Rs} = Rs_i \times e_{WiFi}^{UAV} \tag{4.6}$$

e_{CPU}^{UAV} dénote l'énergie consommée par le CPU du drone pour calculer localement la tâche i et e_{WiFi}^{UAV} dénote l'énergie consommée par l'interface WiFi du drone pour envoyer une unité de données au contrôleur SDN principal.

2) Calcul en délestage par le contrôleur SDN principal : le délai de calcul en délestage ($D_{o,i}^{Sc}$) est égal au temps nécessaire pour envoyer les données au véhicule de secours, soit en un seul saut via le WiFi, soit en mode multi-sauts en utilisant WiFi et DSRC. $D_{o,i}^{Sc}$ est donné par :

$$D_{o,i}^{Sc} = \frac{Ds_i}{R_{WiFi}^{UAV}} + \sum_{s=0}^{h} \frac{Ds_i}{R_{DSRC}} \tag{4.7}$$

L'énergie de calcul en délestage ($E_{o,i}^{Sc}$) est représentée par l'énergie consommée par le drone pour envoyer les données au véhicule de secours. $E_{o,i}^{Sc}$ est calculée par :

$$E_{o,i}^{Sc} = Ds_i \times e_{WiFi}^{UAV} \tag{4.8}$$

4.4.2.2 Description du sous-modèle de calcul du véhicule de secours

Lorsque le contrôleur SDN principal du véhicule de secours reçoit une tâche de données pour le calcul de la part du drone, il choisit soit de calculer toute la tâche localement ou de partager le calcul de la tâche avec un des contrôleurs SDN locaux sur un de ses véhicules contrôleurs voisins. Avant cela, le contrôleur SDN principal agit en deux étapes : il commence par sélectionner le meilleur véhicule contrôleur voisin pour partager avec lui le calcul (le contrôleur SDN local) et ensuite il va négocier avec ce dernier le pourcentage de la tâche qu'il accepte de partager le calcul. Sur la base de ce pourcentage, le contrôleur SDN principal peut prendre la décision du partage. Nous modélisons le délai potentiel (le temps d'attente) nécessaire pour effectuer ces deux opérations par la variable d_i.

i. Sélection du meilleur contrôleur SDN voisin pour partager le calcul : pour optimiser le temps de calcul des tâches, le contrôleur SDN principal peut partager le calcul d'une tâche avec un des contrôleurs SDN locaux sur un de ses véhicules contrôleurs voisins. Nous supposons que tous les véhicules voisins sont homogènes en termes de capacité de calcul et il est inapproprié d'utiliser un lien qui a déjà atteint sa capacité maximale. Ainsi, le contrôleur SDN principal sélectionne le véhicule de partage voisin qui offre la meilleure qualité de lien (*Quality of Link*, QoL), qui peut être exprimée par la probabilité d'une transmission réussite des paquets dans un intervalle du temps réduit [92]. Par conséquent, une liaison avec un débit de transmission élevé et un petit délai aura une qualité de lien plus élevée et le véhicule contrôleur associé aura une grande chance d'être sélectionné comme le meilleur véhicule pour partager le calcul [92]. Le contrôleur SDN principal attribue une probabilité de choix à chaque véhicule contrôleur voisin sur la base de sa QoL. La qualité de lien qui relie le véhicule de secours e au véhicule contrôleur voisin j à l'instant t, notée par $\phi_{e,j}$, est modélisée par la méthode de sélection des actions *softmax* en utilisant une distribution de *Boltzmann* tel qu'il est proposé dans [92], où un lien (e, j) est sélectionné avec une probabilité $\omega_{e,k}$ comme indiqué dans l'équation (4.9) [92] :

$$\phi_{e,j} = \frac{\sigma_2}{\sigma_1 + \omega_{e,j}} \tag{4.9}$$

Tel que

$$\omega_{e,j} = \frac{e^{\frac{q_{e,j}}{\xi}}}{\sum_{k=1}^{m} e^{\frac{q_{e,k}}{\xi}}} \tag{4.10}$$

Ici, $\omega_{e,j}$ représente la probabilité que le véhicule contrôleur voisin j puisse être choisi pour partager le calcul de la tâche parmi les k ($k = [1, m]$) véhicules contrôleurs voisins, autrement dit, la probabilité que le véhicule contrôleur j a la meilleure QoL. σ_1 et σ_2 dénotent les paramètres qui déterminent l'influence de $\omega_{e,k}$. $q_{e,j}$ dénote le débit réel du lien e, j. ξ représente le paramètre de température de Boltzman. Nous modélisons par ξ le délai de transmission réel du lien (e, j). Une valeur élevée de ξ donne une probabilité de choix égale pour tous les liens avec les autres véhicules et une valeur faible signifie une probabilité de choix élevée pour le lien ayant la meilleure QoL. Le meilleur lien correspond au lien avec le délai le plus court et le débit le plus élevé. La valeur du délai de chaque lien est ajustée via le paramètre ξ. Un lien avec un délai court est représenté par de faibles valeurs de ξ et un lien avec un délai élevé est représenté par des valeurs élevées de ξ. Nous supposons que le contrôleur SDN principal gère et met à jour périodiquement une table de voisinage contenant le délai et le débit en temps réels de chaque lien avec ses véhicules contrôleurs voisins.

ii. Négociation du pourcentage de partage : après la sélection du meilleur véhicule de partage ayant la meilleure qualité de lien, le contrôleur SDN principal négocie avec le contrôleur SDN local le pourcentage de la tâche de calcul qu'il peut partager le calcul. Le contrôleur SDN principal commence par envoyer au contrôleur SDN local le nombre de cycles CPU nécessaires pour le calcul de la tâche i. Après avoir évalué ses ressources, ce dernier va répondre en partageant le taux de calcul de la tâche qu'il accepte de partager. Nous représentons ce taux par la variable γ, tel que $\gamma = [0, 1]$: 0 si le contrôleur SDN local n'accepte pas le partage, 1 s'il accepte de partager le calcul de toute la tâche et entre 0 et 1 quand il accepte de partager le calcul d'uniquement une partie de la tâche.

Les différentes formules du temps de calcul des tâches en local par le contrôleur SDN principal et de calcul partagé avec un des contrôleurs SDN locaux voisins sont détaillées dans ce qui suit :

a) Calcul local par le contrôleur SDN principal : le délai total du calcul en local de la tâche i ($D_{l,i}^{Pr}$) par le contrôleur SDN principal est égal au temps d'attente plus le temps de calcul local (T_i^{lp}). $D_{l,i}^{Pr}$ est donné par :

$$D_{l,i}^{Pr} = d_i + T_i^{lp} \tag{4.11}$$

Et

$$T_i^{lp} = \frac{C_i}{F_{CPU}^{EV}} \tag{4.12}$$

Tel que F_{CPU}^{EV} dénote la fréquence de calcul local du contrôleur SDN principal en cycles de CPU par seconde.
Note : d_i est supposé être très petit comparé à T_i^{lp}.

b) Calcul partagé avec un contrôleur SDN local voisin : le temps de calcul partagé est égal au temps de calcul partiel des tâches en local par le contrôleur SDN principal (T_i^{ls}), plus le temps de transfert partiel des données vers le véhicule contrôleur voisin choisi (T_i^t), plus le temps de calcul partiel des tâches en local par le contrôleur SDN local (T_i^{lc}), plus le temps d'envoi des résultats (T_i^{Rs}) au contrôleur SDN principal.
Le temps de calcul partiel des tâches de données (T_i^{ls}) en local par le contrôleur SDN principal est exprimé par :

$$T_i^{ls} = (1-\gamma) \times \frac{C_i}{F_{CPU}^{EV}} \tag{4.13}$$

Le temps de transfert des tâches des données partiellement partagées (T_i^t) est exprimé par :

$$T_i^t = \gamma \times \frac{Ds_i}{R_{DSRC}} \tag{4.14}$$

Le temps de calcul en local des tâches des données partiellement partagées par le contrôleur SDN local j (T_i^{lc}) est exprimé par :

$$T_i^{lc} = \gamma \times \frac{C_i}{F_{CPU}^{CV_j}} \tag{4.15}$$

Le temps d'envoi des résultats de calcul au contrôleur SDN principal (T_i^{Rs}) est exprimé par :

$$T_i^{Rs} = \gamma \times \frac{Rs_i}{R_{DSRC}} \tag{4.16}$$

À partir de 4.13, 4.14, 4.15 et 4.16 nous obtenons :

$$T_{i,j}^{sh} = (1-\gamma) \times \frac{C_i}{F_{CPU}^{EV}} + \gamma \times \phi_{e,j} \times \left(\frac{Ds_i}{R_{DSRC}} + \frac{Rs_i}{R_{DSRC}} + \frac{C_i}{F_{CPU}^{CV_j}}\right) \tag{4.17}$$

Tel que, $F_{CPU}^{cv_j}$ indique la fréquence de calcul en local du contrôleur SDN local sur le véhicule voisin j exprimée en cycles de CPU par seconde.

Le temps total de calcul partagé ($D_{sh,i}^{Pr}$) est égal au temps d'attente plus le temps de calcul partagé. $D_{sh,i}^{Pr}$ est donné par :

$$D_{sh,i}^{Pr} = d_i + T_{i,j}^{sh} \tag{4.18}$$

4.4.3 Fonction du coût du système

Étant donné que nous étudions une mission de secours suite à un accident sur la route, les données collectées par le drone doivent être traitées en toute urgence afin d'extraire des informations pertinentes sur l'état des lieux (le nombre de victimes par exemple). Ainsi,

le temps de réponse pour obtenir les résultats de calcul est considéré comme une tâche de calcul sensible au délai. De plus, le drone joue un rôle important dans l'exploration de la zone touchée par l'incident et la durée de vie de sa batterie est un facteur déterminant pour le succès de la mission de secours. Donc, le délai de calcul et la consommation énergétique de drone pour le calcul constituent les principales métriques qui peuvent affecter la prise de décision en matière de délestage/partage de calcul des tâches. Par conséquent, la fonction de gain global du système, communément appelée *le coût du système* (notée P_i) est représentée par une combinaison entre l'énergie (l'énergie consommée par le drone pour calculer une tâche) et le délai (le temps de réponse total pour obtenir les résultats de calcul). P_i est exprimée comme suit :

$$P_i = \alpha \times \frac{D_i - minD}{maxD - minD} + \beta \times \frac{E_i - minE}{maxE - minE} \tag{4.19}$$

Tel que, α et β $(\alpha, \beta \in [0,1], \alpha + \beta = 1)$ représentent respectivement les facteurs de pondération pour le délai et l'énergie. $maxE$ (*resp.* $minE$) représente le maximum (*resp.* minimum) de l'énergie et le $maxD$ (*resp.* $minD$) maximum (*resp.* minimum) de délai nécessaire pour calculer la taille maximale (*resp.* minimale) des données d'une tâche.

Dans la sous-section suivante, nous utilisons un jeu séquentiel pour modéliser notre politique de traitement des données présentée précédemment ainsi que le problème de prise de décision en matière de délestage/partage de calcul des tâches correspondant.

4.5 Jeu séquentiel pour un traitement des données efficace dans les scénarios de secours des réseaux de véhicules assistés par des drones

Dans cette section, nous décrivons une politique basée sur un jeu séquentiel pour un traitement efficace des données dans un scénario de secours dans les VANETs assistés par des drones, qui consiste en un problème de prise de décision en matière de délestage/partage de calcul. L'objectif principal est d'atteindre le meilleur équilibre entre le délai et l'énergie de calcul. Nous investiguons l'existence de l'équilibre de Nash (EN) et nous proposons deux algorithmes distribués pour résoudre le problème.

La théorie des jeux est considérée comme un outil de modélisation mathématique puissant qui permet d'analyser les problèmes de prise de décisions stratégiques liés aux interactions entre plusieurs acteurs rationnels qui agissent pour réaliser leurs propres intérêts. Un jeu est défini par l'ensemble des joueurs (des entités ou des agents physiques ou logiques qui sont impliqués dans le jeu). Les joueurs doivent prendre des décisions et choisir des actions (stratégies) selon les règles de jeu pour obtenir un certain résultat souvent appelé utilité. L'utilité est une mesure qui reflète le degré de satisfaction d'un joueur en termes des gains et des pertes possibles pour chaque décision prise. Récemment, plusieurs travaux ont utilisé la théorie des jeux pour étudier des problèmes de délestage de calcul tel que dans [89–91]. La théorie des jeux a été également utilisée pour modéliser différents problèmes dans les VANETs basés sur SDN, tels que pour le partage des

ressources dans [92] et l'optimisation du plan de contrôle dans [10].

4.5.1 Jeu séquentiel pour le délestage/partage de calcul

Nous formulons le problème de prise de décision en matière de délestage/partage de calcul des tâches relatives au traitement des données collectées par le drone, sous forme d'un jeu séquentiel fini à deux joueurs, information parfaite et à somme non-nulle. Dans un jeu séquentiel à information parfaite, les joueurs agissent de manière séquentielle, lorsque chaque joueur peut observer les mouvements (stratégies ou actions) des autres joueurs qui agissent avant lui et effectuer des choix stratégiques en conséquence. Notre jeu séquentiel est joué entre le drone représenté par le contrôleur SDN secondaire et le véhicule de secours représenté par le contrôleur SDN principal. Comme les données sont initialement collectées par le drone, le contrôleur SDN secondaire agit en premier et le contrôleur SDN principal réagit après avoir observé la décision du contrôleur SDN secondaire et il ne joue que si le contrôleur SDN secondaire choisit de lui délester le calcul d'une tâche des données à traiter. Le nombre de tâches à calculer est fini et il est égal au nombre des tâches nécessaires pour traiter l'ensemble des données collectées par le drone. Notre jeu est fini et il se termine après un certain nombre de périodes de décision lorsque l'équilibre du système est obtenu (*i.e.*, tous les paramètres sont optimisés pour chaque donnée collectée) ou lorsque le niveau d'énergie de retour du drone est atteint (l'énergie nécessaire pour que le drone puisse retourner à son véhicule de secours).

Définition 1. le jeu séquentiel est défini par le triplet $G(N, A, \mu)$:

- $N = \{1, 2\}$, représente l'ensemble fini non-vide des joueurs : 1 représente le contrôleur SDN secondaire sur le drone et 2 représente le contrôleur SDN principal sur le véhicule de secours ;
- $A = \{a_1, a_2\}$, représente l'ensemble fini non-vide des actions ou des stratégies des joueurs. Tel que la stratégie de joueur 1 est $a_1 = \{s_j, \forall j \in (0\ \textit{calcul local}\ (\textit{Local computation}), 1\ \textit{délestage de calcul}\ (Offload\ computation))\}$ et la stratégie de joueur 2 est $a_2 = \{s_i, \forall i \in (0\ \ \textit{calcul local}\ (\textit{Local computation}), 1\ \textit{Partage de calcul}\ (\textit{Shared computation}))\}$.
- $\mu = (\mu_1, \mu_2)$, représente la fonction d'utilité ou la fonction des gains de chaque joueur. μ_1 c'est la fonction des gains du joueur 1 et μ_2 c'est la fonction des gains du joueur 2. μ_1 et μ_2 sont définies comme suit :

$$\begin{aligned}\mu_{1,i}(a_{1,i}) = \theta_i \times (\alpha \times \frac{D^{Sc}_{l,i} - minD}{maxD - minD} + \beta \times \frac{E^{Sc}_{l,i} - minE}{maxE - minE}) + (1 - \theta_i) \\ \times (\alpha \times \frac{D^{Sc}_{o,i} - minD}{maxD - minD} + \beta \times \frac{E^{Sc}_{o,i} - minE}{maxE - minE})\end{aligned} \tag{4.20}$$

Tel que

$$\theta_i = \begin{cases} 1, & \text{si } \ s_{1,i} = 0 \\ 0, & \text{si } \ s_{1,i} = 1 \end{cases} \tag{4.21}$$

Et

$$\mu_{2,i}(a_{2,i}) = \alpha \times (\vartheta_i \times \frac{D^{Pr}_{l,i} - minD}{maxD - minD} + (1 - \vartheta_i) \times \frac{D^{Pr}_{sh,i} - minD}{maxD - minD}) \tag{4.22}$$

Tel que

$$\vartheta_i = \begin{cases} 1, & \text{si } s_{2,i} = 0 \\ 0, & \text{si } s_{2,i} = 1 \end{cases} \tag{4.23}$$

Le coût du système pour calculer une tâche de donnée i à l'instant t est la somme des gains de chaque joueur et il est donné par :

$$P_i(t) = \mu_i(t) = \mu_{1,i}(a_{1,i}) + \mu_{2,i}(a_{2,i}) \tag{4.24}$$

Dans ce travail, nous cherchons à minimiser le coût total du système en optimisant conjointement le délai et l'énergie de calcul des tâches.

La figure 4.5 illustre la représentation stratégique de notre jeu séquentiel pour la politique de délestage/partage de calcul et sa matrice des gains correspondante.

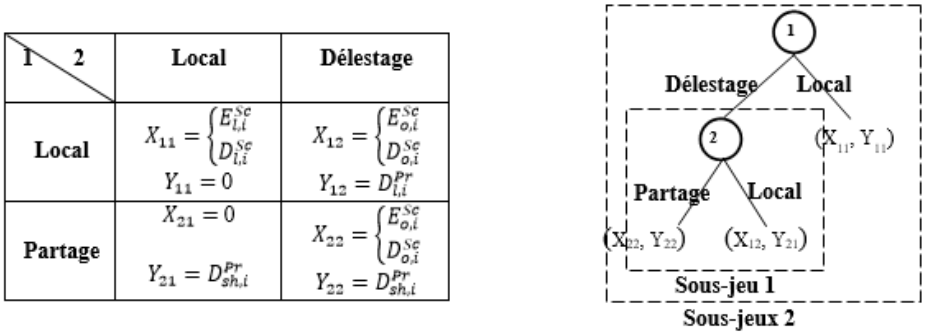


FIGURE 4.5 – Représentation stratégique de jeu séquentiel pour la politique de délestage/partage de calcul et la matrice des gains correspondante.

4.5.2 Equilibre de Nash

Dans cette sous-section, nous étudions l'existence de l'équilibre de Nash. Après, nous utilisons la technique de la récurrence à rebours (backward induction) pour résoudre ce jeu. Un équilibre de Nash est une situation mutuellement satisfaisante dans laquelle aucun joueur n'a intérêt à changer unilatéralement sa stratégie en tenant compte des stratégiques choisies par les autres joueurs.

Définition 2 : un équilibre de Nash à stratégie pure pour notre jeu séquentiel fini à deux joueurs pour le délestage/partage de calcul des tâches des données est un profil de décision $d^* = (d^*_{1,1}, \ldots, d^*_{1,i},\ d^*_{2,1}, \ldots, d^*_{2,i})$ tel que $\forall n \in N$ nous avons le suivant [90] :

$$P_{n,i}(d^*_{n,i}) \preceq P_{n,i}(d_{n,i}), \forall d_{n,i} \in A \tag{4.25}$$

À l'équilibre de Nash, le contrôleur SDN principal et le contrôleur SDN secondaire peuvent parvenir à une solution mutuellement satisfaisante pour laquelle aucun contrôleur n'aura l'incitation à s'écarter unilatéralement [90]. Cette solution satisfaisante est définie comme une stratégie optimale qui doit être sélectionnée par les deux joueurs pour

obtenir à la fois la plus petite énergie et le plus court délai pour le calcul de chaque tâche.

Théorème 1. (Kuhn) *"Les jeux séquentiels quand les joueurs ont un ensemble d'actions finies et agissent uniquement en un nombre fini de fois sont équivalents, et chaque jeu fini avec une forme extensive à information parfaite a un équilibre de Nash avec une pure stratégie"* [90].

Preuve : La preuve du théorème 1 est donnée dans [93].

Lemme 1 : $G(N, A, \mu)$ est un jeu séquentiel fini avec information parfaite.

Preuve : Voir la sous-section 4.5.1.

Le théorème 1 implique que pour notre jeu séquentiel à deux joueurs avec information parfaite (voir la sous-section 4.5.1), au moins un équilibre de Nash existe et il peut converger après un nombre fini de périodes de décision [90].

Théorème 2. (Zermolo) *"Chaque jeu fini avec information parfaite a un équilibre de Nash avec une stratégie pure qui peut être obtenu par la récurrence à rebours"* [94].

Preuve : la preuve du théorème 2 est donnée dans [94].

Comme le montre la figure 4.5, une étape de notre jeu séquentiel à information parfaite à un instant de décision t englobe deux sous-jeux : *(i)* le sous-jeu de contrôleur SDN principal pour le calcul local/partage de calcul et *(ii)* le sous-jeu de contrôleur SDN secondaire pour le calcul local/délestage de calcul. Nous résolvons notre jeu séquentiel par la récurrence à rebours, un principe de raisonnement par récurrence qui consiste à commencer à chercher l'équilibre à partir de la fin (le dernier sous-jeu) pour arriver à l'équilibre global au début (le jeu global). Premièrement, nous étudions la meilleure stratégie du contrôleur SDN principal (soit le calcul en local ou le partage de calcul). Après, nous évaluons la meilleure stratégie du contrôleur SDN secondaire (soit le calcul en local ou le délestage de calcul).

4.5.3 Meilleure décision du contrôleur SDN principal

Étant donné la décision du contrôleur SDN secondaire, le contrôleur SDN principal peut dériver la stratégie de partage de calcul optimale S_i en résolvant le problème :

$$\phi_{2,i}(a_i) = \arg\min_{D^{Pr}} \mu_{2,i}(a_{2,i}) = \vartheta_i \times \frac{D_{l,i}^{Pr} - minD}{maxD - minD} + (1 - \vartheta_i) \times \frac{D_{sh,i}^{Pr} - minD}{maxD - minD} \quad (4.26)$$

Tel que

$$\vartheta_i = \begin{cases} 1, & \text{si } \ s_{2,i} = 0 \\ 0, & \text{si } \ s_{2,i} = 1 \end{cases} \quad (4.27)$$

D^{Pr} représente le délai de calcul du contrôleur SDN principal.

Il est facile de vérifier que la fonction de gain du contrôleur SDN principal (4.26) est convexe par le calcul des dérivées partielles.

$$\frac{\partial^2 \mu_{2,i}(D_{l,i}^{Pr}, D_{sh,i}^{Pr})}{\partial^2 D_{l,i}^{Pr}} = 0, \frac{\partial^2 \mu_{2,i}(D_{l,i}^{Pr}, D_{sh,i}^{Pr})}{\partial^2 D_{sh,i}^{Pr}} = 0 \tag{4.28}$$

À partir des équations précédentes, nous concluons que la fonction de gain du contrôleur SDN principal est une fonction convexe (puisque la seconde dérivée est nulle). Donc, selon la théorie d'optimisation convexe, il existe plusieurs solutions optimales, et par conséquent il y'a au moins un équilibre de Nash pour le sous-jeu de contrôleur SDN principal. Cependant, puisque le nombre de joueurs est petit (deux joueurs), la fonction de gain est simple et facile à résoudre. Ainsi, la décision optimale correspondante à la meilleure stratégie de contrôleur SDN principal ($d_{2,i}^*$) est obtenue par la résolution du problème dans l'équation (4.26) comme suit :

$$d_{2,i}^* = \phi_{2,i}(a\_i) = \begin{cases} \arg\min\limits_{D_{l,i}^{Pr}} \mu_{2,i}, & \text{si} \quad s_{2,i} = 0 \\ \arg\min\limits_{D_{sh,i}^{Pr}} \mu_{2,i}, & \text{si} \quad s_{2,i} = 1 \end{cases} \tag{4.29}$$

- Algorithme pour la prise de décision de partage du calcul

Nous décrirons dans l'algorithme 1 la procédure distribuée pour calculer la meilleure décision du contrôleur SDN principal concernant le partage ou non de calcul des tâches. À chaque instant de décision t, le contrôleur SDN principal lorsqu'il reçoit une tâche à calculer agit en trois étapes (voir la sous-section 4.4.2.2). Premièrement, il sélectionne le meilleur véhicule contrôleur en fonction de la qualité du lien qu'il les relie pour partager le calcul avec son contrôleur SDN local. Dans une deuxième étape, il négocie le pourcentage de la tâche à partager le calcul avec le contrôleur SDN local du véhicule contrôleur sélectionné. Enfin, dans la troisième étape, il calcule et compare le délai du calcul local avec celui du partage et prend la meilleure décision en fonction des paramètres du système mesurés à l'instant t. Étant donné que le problème est formulé comme un jeu séquentiel à information parfaite, le contrôleur SDN principal doit informer le contrôleur SDN secondaire de sa décision. L'équilibre de Nash est atteint lorsque le délai et l'énergie de calcul seront optimisés pour chaque tâche i et qu'aucun joueur n'a la motivation à dévier unilatéralement de cet équilibre. Le contrôleur SDN principal envoie un message de fin de jeu au contrôleur SDN secondaire lorsque l'équilibre est atteint ou lorsque le niveau d'énergie de retour de drone est signalé. L'algorithme proposé est légèrement inspiré des algorithmes proposés dans les travaux [89, 90].

Algorithm 1 Algorithme de la prise de décision de partage de calcul

Entrée : Un ensemble des tâches à calculer délestées par le drone.
Sortie : Un profil optimal des décisions de partage de calcul des tâches.

1 : Initialisation
2 : Choisir le calcul local comme stratégie initiale du contrôleur principal : $d_{2,i}(0) \longleftarrow 0$
3 : Initialiser et mettre à jour la table de voisinage
4 : Fin Initialisation

5 : Début
6 : Répéter pour chaque instant de décision t :
7 : Si (*réception* d'une tâche i à calculer) **alors**
8 : *Choisir* le meilleur contrôleur SDN local pour partager le calcul
9 : *Négocier* le pourcentage de partage
10 : *Mesurer* les paramètres du système
11 : *Choisir* la nouvelle meilleure stratégie du contrôleur SDN principal $\phi_{2,i}(a_{-i})(t+1)$ en utilisant l'équation (4.29)
12 : *Calculer* la nouvelle valeur de P_{i+1} en utilisant l'équation (4.24)
13 : Si $((\phi_{2,i}(a_{-i})(t+1) < \phi_{2,i}(a_{-i})(t)))$ **alors**
14 : *Prendre* la décision $d_{2,i}(t+1) \leftarrow 1 - d_{2,i}(t)$
15 : FinSi
16 : Si (*réception* de message d'alerte de l'énergie de retour) **alors**
17 : *Arreter*
18 : Sinon *envoyer* la décision prise au contrôleur SDN secondaire
19 : FinSi
20 : Jusqu'à que l'équilibre soit atteint
21 : *Envoyer* le message de fin de jeu au contrôleur SDN secondaire
22 : Fin

4.5.4 Meilleure décision de calcul local/délestage du calcul du contrôleur SDN secondaire

Le contrôleur SDN secondaire peut dériver sa stratégie optimale en résolvant le problème suivant :

$$\phi_{1,i}(a_i) = \underset{D_{Sc},\ E_{Sc}}{\arg\min}\ \mu_{1,i}(a_{1,i}) = \theta_i \times (\alpha \times \frac{D_{l,i}^{Sc} - minD}{maxD - minD} + \beta \times \frac{E_{l,i}^{Sc} - minE}{maxE - minE}) + (1-\theta_i) \times (\alpha \times \frac{D_{o,i}^{Sc} - minD}{maxD - minD} + \beta \times \frac{E_{o,i}^{Sc} - minE}{maxE - minE}) \tag{4.30}$$

Tel que

$$\theta_i = \begin{cases} 1, & \text{si } \ s_{1,i} = 0 \\ 0, & \text{si } \ s_{1,i} = 1 \end{cases} \tag{4.31}$$

D^{Sc} et E^{Sc} représentent respectivement le délai et l'énergie nécessaire pour que le contrôleur SDN secondaire termine le calcul de la tâche i.

Il est facile de vérifier que la fonction de gain du contrôleur SDN secondaire (4.30) est une fonction convexe en calculant les dérivées partielles.

$$\begin{aligned} &\frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 D_{l,i}^{Sc}} = 0, \frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 D_{o,i}^{Sc}} = 0, \\ &\frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 E_{l,i}^{Sc}} = 0, \frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 E_{o,i}^{Sc}} = 0 \end{aligned} \tag{4.32}$$

À partir des équations précédentes, nous concluons que la fonction de gain du contrôleur SDN secondaire est une fonction convexe (puisque la seconde dérivée est nulle). Il existe donc plusieurs solutions optimales, et au moins un équilibre de Nash existe pour le sous-jeu du contrôleur SDN secondaire. Cependant, puisque le nombre de joueurs est petit (deux joueurs), la fonction de gain est simple et facile à résoudre. Ainsi, la décision optimale correspondante à la meilleure stratégie de contrôleur SDN secondaire ($d_{1,i}^*$) à l'instant t est obtenue par la résolution du problème dans l'équation (4.30) comme suit :

$$d_{1,i}^* = \phi_{1,i}(a\_i) = \begin{cases} \underset{D_{l,i}^{Sc}, E_{l,i}^{Sc}}{\arg\min}\ \mu_{1,i}, & \text{si} \quad s_{1,i} = 0 \\ \underset{D_{o,i}^{Sc}, E_{o,i}^{Sc}}{\arg\min}\ \mu_{1,i}, & \text{si} \quad s_{1,i} = 1 \end{cases} \tag{4.33}$$

- Algorithme pour la prise de décision de délestage de calcul

Nous décrirons dans l'algorithme 2 la procédure distribuée pour calculer la meilleure décision du contrôleur SDN secondaire concernant le délestage ou non de calcul. La logique de l'algorithme 2 est similaire à celle de l'algorithme 1. Le contrôleur SDN secondaire surveille continuellement l'énergie de son drone. S'il constate que le niveau d'énergie de retour est atteint, le contrôleur SDN secondaire envoie un message d'alerte pour signaler la faible énergie du drone au contrôleur SDN principal. L'algorithme se termine lorsque le message de fin est reçu ou lorsque l'équilibre est atteint. Puisque le contrôleur SDN secondaire agit en premier, l'algorithme 2 est exécuté avant l'algorithme 1.

4.6 Résultats numériques

Dans cette section, nous utilisons Matlab pour évaluer les performances de notre politique de traitement des données ainsi que celles des deux algorithmes distribués pour le délestage/partage de calcul. Le scénario simulé est basé sur le modèle du système décrit dans la sous-section 4.4.1. Pour les différentes expérimentations, on considère que la capacité de calcul du CPU du véhicule de secours (contrôleur SDN principal), F_{CPU}^{EV}, est dix fois plus puissante que celle du drone (contrôleur SDN secondaire), F_{CPU}^{UAV}, et la capacité de calcul du CPU du véhicule contrôleur (contrôleur SDN local), F_{CPU}^{CV}, est trois fois plus puissante que celle du drone (contrôleur SDN secondaire). Nous considérons aussi que l'énergie d'envoi d'une unité de données via l'interface sans fil WiFi du drone, e_{WiFi}^{UAV}, consomme mille fois plus d'énergie que le calcul local de la même unité de données sur le drone e_{CPU}^{UAV} [95]. Le taux de transmission de l'interface WiFi ($R^{WiFi} = 12Mbps$) est considéré comme deux fois plus rapide en terme de vitesse de transmission que celui de

Algorithm 2 Algorithme de la prise de décision de délestage de calcul

Entrée : Un ensemble de tâches à calculer correspondant au traitement des données collectées par le drone
Sortie : Un profil optimal des décisions de délestage

1 : Initialisation
2 : *Choisir* le calcul local comme stratégie initiale pour le contrôleur secondaire : $d_{1,i}(0) \longleftarrow 0$
3 : *Calculer* la valeur initiale de la fonction de coût P_i
4 : Fin Initialisation

5 : Début
6 : Répéter pour chaque tâche i à calculer et à chaque instant t :
7 : *Mesurer* les paramètres du système pour le calcul
8 : *Choisir* la nouvelle meilleure stratégie du contrôleur SDN secondaire $\phi_{1,i}(a_{-i})(t+1)$ en utilisant l'équation dans (4.33)
9 : *Calculer* la nouvelle valeur de P_{i+1} en utilisant l'équation dans (4.24)
10 : Si $(\phi_{1,i}(a_{-i})(t+1) \prec \phi_{1,i}(a_{-i})(t)$) **alors**
11 : *Prendre* la décision $d_{1,i}(t+1) \longleftarrow 1 - d_{1,i}(t)$
12 : Fin Si
13 : *Envoyer* les résultats ou les tâches à calculer au contrôleur SDN principal
14 : Si (le niveau d'énergie de retour est atteint) **alors**
15 : *Envoyer* un message d'alerte au contrôleur SDN principal
16 : Fin Si
17 : Jusqu'à la réception de message de fin à partir de contrôleur SDN principal
18 : Fin

l'interface DSRC ($R^{DSRC} = 6Mbps$) [96]. Étant donné que nous investiguons une mission de secours dans un scénario critique relatif à la sécurité routière où le calcul des tâches des données à traiter est considéré comme une opération sensible au délai, nous fixons un poids plus élevé ($\alpha = 0.7$) pour le délai de calcul dans la prise de décision par rapport à l'énergie de calcul ($\beta = 0.3$). Les principaux paramètres de simulation sont résumés dans le tableau 4.1.

Dans la suite de cette section, nous comparons les performances de nos deux algorithmes distribués pour le délestage/partage de calcul, appelé *Distributed Offloading Sharing Computation* (DOSC), basé sur le jeu séquentiel à deux joueurs pour le traitement des données avec trois autres scénarios : *(i)* le calcul local (*Local Computation*, LC), lorsque le contrôleur SDN secondaire du drone décide de calculer toutes les tâches localement, *(ii)* le délestage de calcul (*Offloading Computation*, OC), lorsque le contrôleur SDN secondaire sur le drone décide d'envoyer toutes les tâches au véhicule de secours et son contrôleur SDN principal décide de calculer toutes les tâches localement et, *(iii)* le partage de calcul (*Shared Computatio*n, SC), lorsque le contrôleur SDN secondaire du drone décide d'envoyer toutes les tâches au véhicule de secours et son contrôleur SDN principal décide de partager le calcul avec un des contrôleurs SDN locaux voisins.

Paramètre	Valeur
C_i	$[100, \ldots, 100000]$ $(\times 10^3)$
Ds_i	$[10, \ldots, 2000]$ $(\times 10^3)$
Rs_i	$[5, \ldots, 15]$
F_{CPU}^{UAV}	1 GHZ
F_{CPU}^{EV}	10 GHZ
F_{CPU}^{CV}	3 GHZ
R^{WiFi}	12 Mbps
R^{DSRC}	6 Mbps
e_{CPU}^{UAV}	1 u
e_{WiFi}^{UAV}	100 u
M	$[1, 4]$
h	$[0, 4]$
α	0.7
β	0.3
σ_1	0.5
σ_2	0.5
γ	0.5

TABLE 4.1 – Paramètres de simulation.

Dans l'expérimentation illustrée à la figure 4.6, nous comparons le coût moyen du système de différents scénarios de traitement de données.

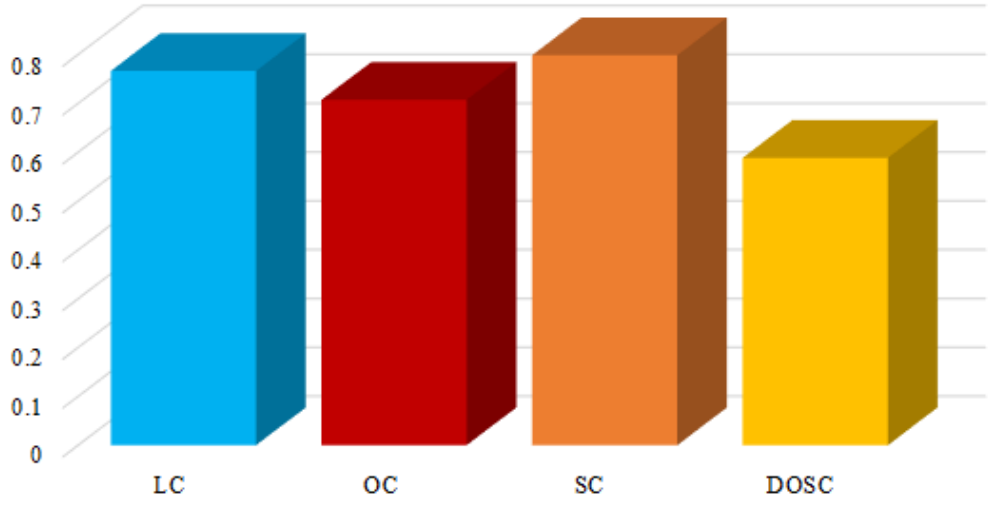


FIGURE 4.6 – Coût moyen du système pour différents scénarios de traitement des données.

Comme nous pouvons clairement le remarquer à partir des résultats dans la figure 4.6, notre algorithme DOSC est le plus efficace et surpasse les trois autres scénarios tout en réalisant un meilleur gain moyen de 28, 25 et 17% par rapport aux scénarios SC, LC et OC, respectivement. Cela est principalement dû au fait qu'à chaque instant de décision t, notre politique DOSC cherche à établir le meilleur équilibre entre les différentes métriques du système (le délai et l'énergie) en considérant l'état des ressources du réseau (*e.g.*, CPU, énergie du drone, QoL) en temps réel afin de prendre les décisions de calcul

les plus optimales. Ainsi, le véhicule de secours peut obtenir dans les plus brefs délais et avec une meilleure consommation énergétique des informations pertinentes sur l'état de la zone de l'accident. Cette vue en temps réel sur l'état des ressources du réseau est garantie grâce à la collaboration entre les contrôleurs SDN. Au contraire, les autres scénarios de prise de décision de calcul des tâches, prennent des décisions naïves sans tenir compte de l'état des ressources du réseau.

Dans la figure 4.7, nous étudions l'impact de la taille des données collectées par le drone sur le coût du système. Dans cette expérimentation, nous fixons la complexité des tâches à calculer en termes des cycles de CPU et nous faisons varier à chaque fois la taille des données.

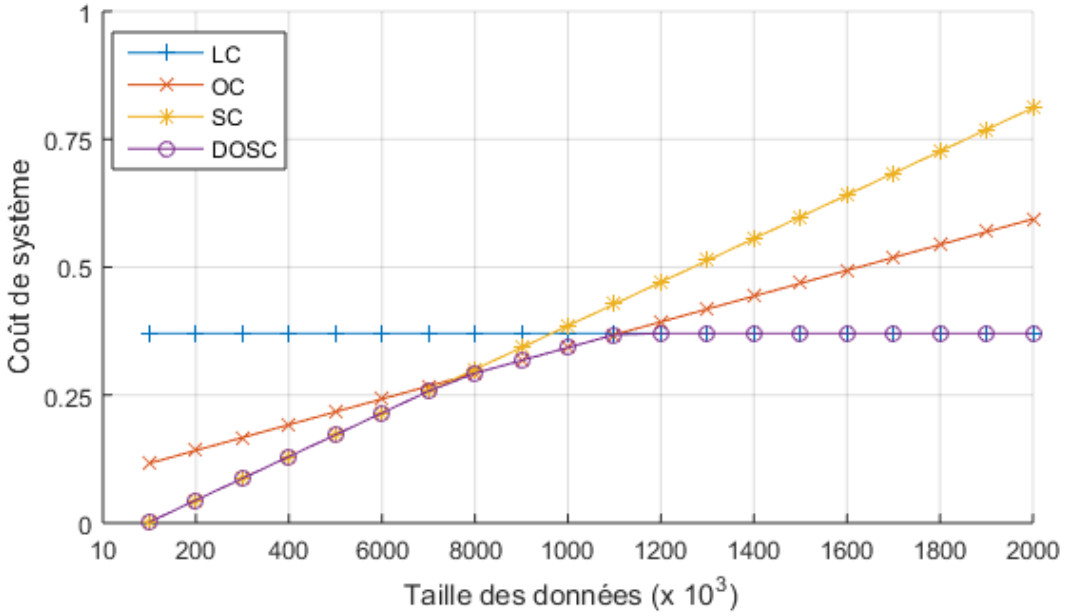


FIGURE 4.7 – Impact de la taille des données sur le coût du système.

À partir des résultats illustrés à la figure 4.7, nous pouvons constater que le coût du système pour les scénarios OC/SC augmente avec l'augmentation de la taille des données, alors que celle-ci a un impact très léger sur le scénario LC. Nous remarquons aussi que le scénario SC est meilleur qu'OC pour le calcul des tâches avec une taille de données relativement petite à l'inverse de cas des tailles plus importantes. Cela peut être justifié par le fait que le délestage/partage des données volumineuses via l'interface sans fil, entraînant une dégradation des performances du système causée par un long délai de transmission et une forte consommation de l'énergie de drone, contrairement au scénario de LC où uniquement des petites tailles des résultats sont transmises. De plus, dans le scénario SC, les données sont envoyées au véhicule de secours puis sont partagées avec un des véhicules contrôleurs. Ce qui consomme plus de délai et d'énergie de transmission. Par conséquent, nous concluons que LC est plus adapté au traitement des grandes tailles des données à l'inverse d'OC et SC qui semblent plus appropriés pour le traitement des données à moyennes et petites tailles, respectivement. Nous pouvons également remarquer l'efficacité notable de notre politique DOSC qui arrive à choisir toujours la meilleure stratégie qui donne le

coût optimal du système. Ce qui va permettre au système d'effectuer des calculs intensifs avec des délais de réponse acceptables et une consommation d'énergie raisonnable quelle que soit la taille des données de la tâche. Cela grâce au profil de décision dans l'EN obtenu grâce à nos deux algorithmes intelligents et au jeu séquentiel basé sur SDN.

Dans l'expérimentation à la figure 4.8, nous étudions l'impact de la complexité de calcul des tâches des données collectées par le drone en termes de cycles de CPU par rapport au coût du système. Dans cette expérimentation, nous fixons la taille des données des tâches à calculer et nous modifions à chaque fois le nombre des cycles du CPU.

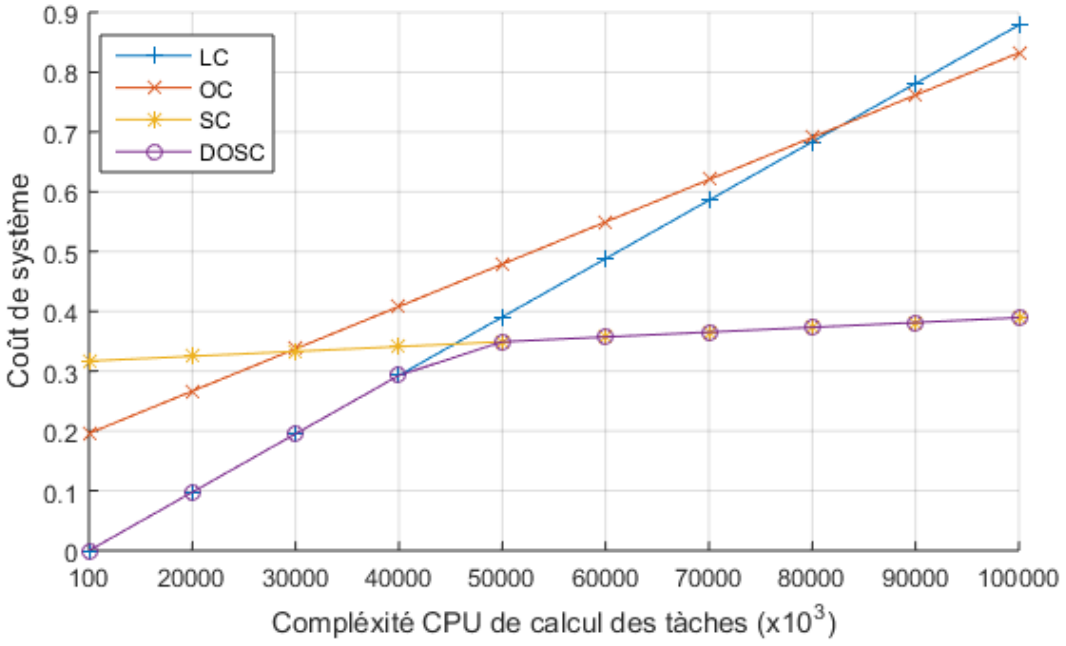


FIGURE 4.8 – Impact de la complexité CPU de calcul des tâches sur le coût de système.

Comme le montrent les résultats à la figure 4.8, le coût du système dans les scénarios LC/OC augmente avec l'augmentation du nombre des cycles de CPU, tandis que celui du scénario SC expérimente un léger impact. Ceci est principalement dû au fait que le calcul local par le contrôleur SDN secondaire ou le calcul en délestage par le contrôleur SDN principal des tâches complexes avec beaucoup de cycles de CPU nécessite plus d'énergie et/ou de temps de calcul selon la fréquence de calcul du CPU, ce qui justifie l'augmentation dans le coût du système. À l'inverse du scénario SC, où plusieurs contrôleurs SDN distribués collaborent entre eux pour optimiser les mesures du système en partageant le calcul des tâches. Par conséquent, nous pouvons conclure que le SC est plus adapté au calcul des tâches intensives tandis que le LC est plus adapté au calcul des tâches moins intensives. De plus, nous remarquons que l'application de notre politique DOSC basée sur SDN et l'utilisation des deux algorithmes d'optimisation distribués de notre jeu séquentiel adoptent toujours la stratégie de calcul la plus efficace correspondant au coût optimal du système qui garantit le meilleur équilibre possible entre un temps de calcul rapide et une consommation d'énergie réduite.

Dans la figure 4.9, nous étudions l'impact de la taille des données des tâches à calculer sur la consommation énergétique moyenne du drone. Nous modifions à chaque fois la taille des données et nous calculons la consommation énergétique moyenne correspondante pour différentes complexités des tâches en termes de cycles de CPU.

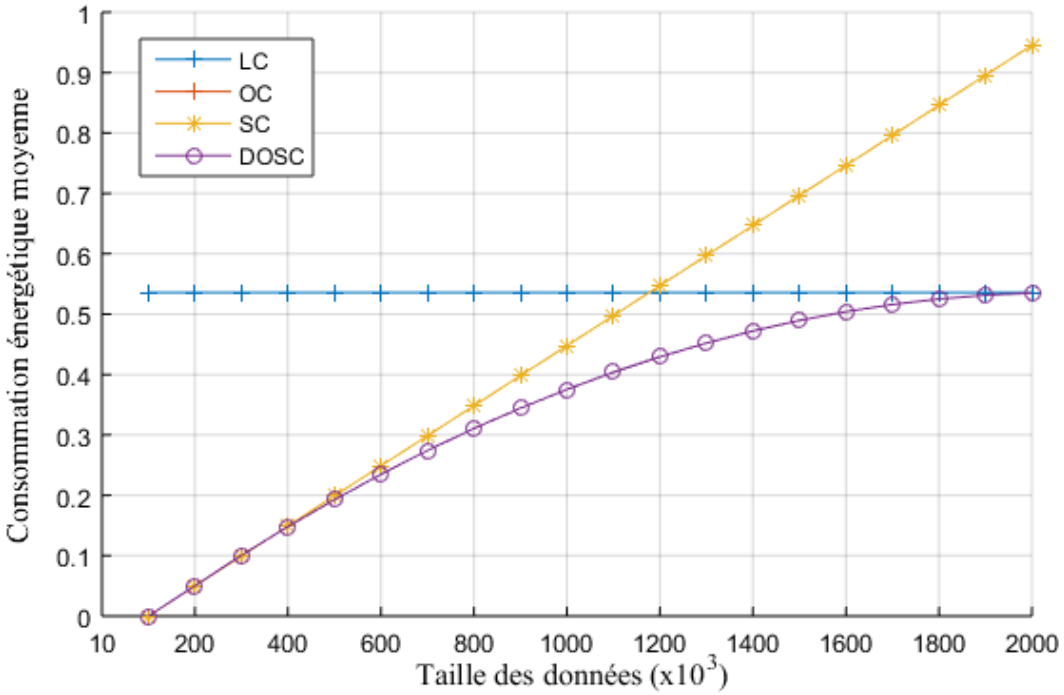


FIGURE 4.9 – Impact de la taille des données des tâches à calculer sur la consommation énergétique moyenne du drone.

Les résultats illustrés à la figure 4.9 montrent que la consommation énergétique dans les scénarios d'OC et SC augmente linéairement avec l'augmentation de la taille des données de la tâche à calculer à l'inverse de celle de LC, qui semble peu affectée par l'augmentation de la taille des données. De plus, étant donné que nous considérons que les véhicules n'ont pas de contrainte énergétique, la consommation d'énergie dans le cas d'OC et SC est identique et égale à l'énergie consommée par le drone pour la transmission des données. Cela peut être justifié par le fait que la transmission via l'interface sans fil d'une unité de données est largement plus gourmande en énergie que si cette même unité est calculée localement [95]. En effet, dans le scénario d'OC/SC, le calcul des tâches passe par la transmission de la totalité de la taille des données des tâches à travers l'interface sans fil. Cela se traduit par une forte consommation des ressources énergétiques limitées du drone, avant d'être calculée sur les véhicules de secours au sol, référez-vous à l'équation (4.8). Tandis que dans le scénario LC, les tâches seront calculées localement par le contrôleur SDN secondaire sur le drone et uniquement des petites tailles des résultats seront transmises via l'interface sans fil, ce qui implique une faible consommation énergétique, référez-vous à l'équation (4.4). En guise de conclusion, nous pouvons dire que l'OC est plus adapté au calcul des tâches avec une petite taille des données alors que LC est plus approprié pour les tâches avec une grande taille des données. De plus, il est clair que l'application de notre politique DOSC basée sur notre jeu séquentiel, l'établissement du profil de décision dans l'EN correspondant à chaque taille des données ainsi que la

collaboration entre les contrôleurs SDN, ont permis d'économiser d'une manière efficace l'énergie du drone en choisissant à chaque instant de décision la stratégie la plus optimale qui minimise la consommation énergétique.

Dans l'expérimentation à la figure 4.10, nous étudions l'impact de la taille des données des tâches à calculer sur le délai de calcul moyen. Dans cette expérimentation, nous modifions la taille des données et nous calculons le délai moyen pour différentes complexités des tâches en termes de cycles de CPU.

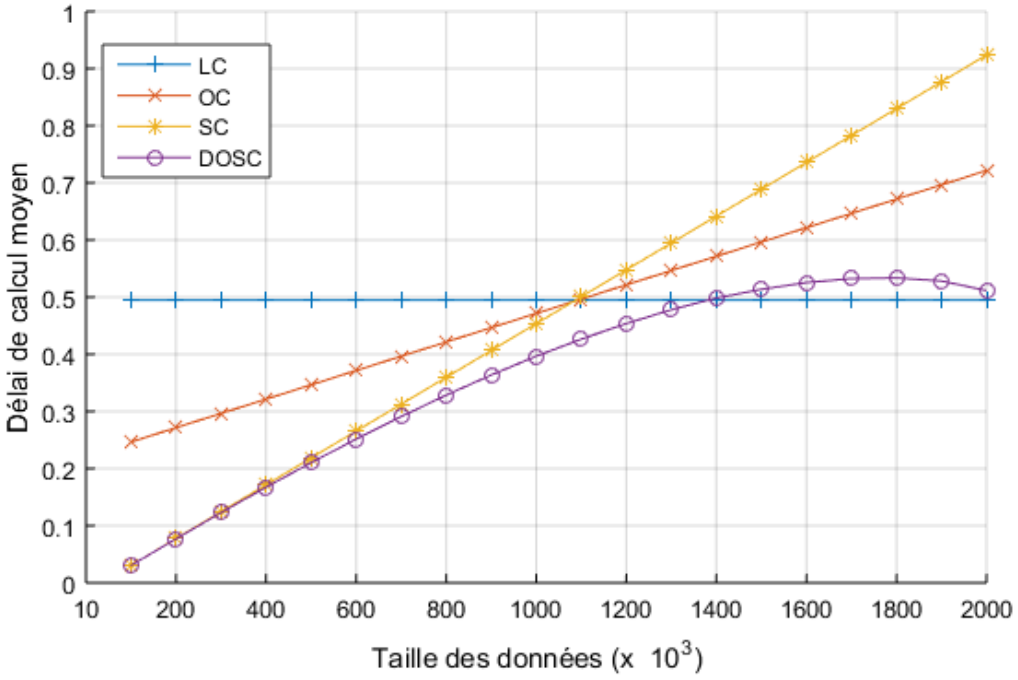
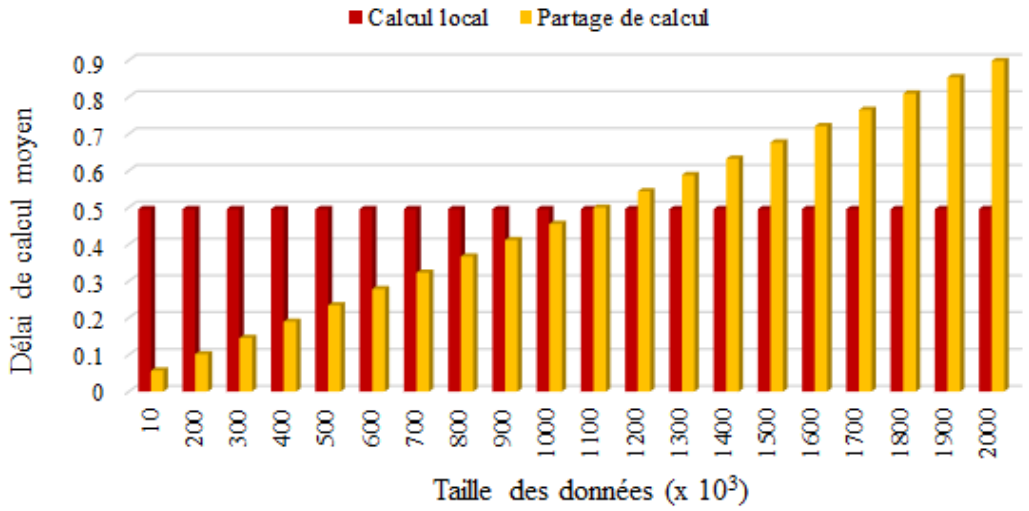


FIGURE 4.10 – Impact de la taille des données des tâches sur le délai de calcul.

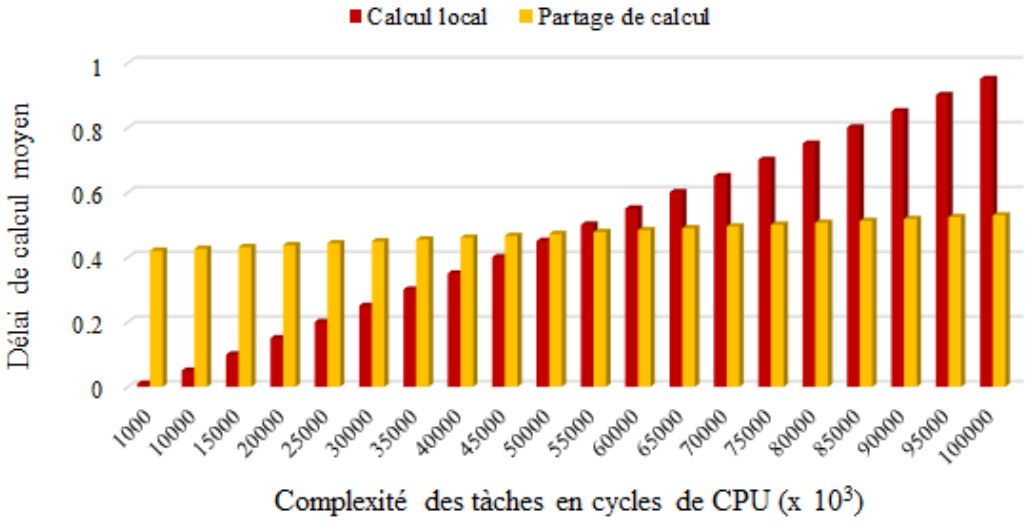
Les résultats illustrés à la figure 4.10 montrent que le délai de calcul pour les deux scénarios OC et SC augmente linéairement avec l'augmentation de la taille des données de la tâche à calculer alors que celui de LC expérimente un très léger effet. Par conséquent, il est clair que pour le calcul des tâches avec des petites tailles des données, le calcul OC/SC est plus approprié, tandis que pour les tâches avec de grandes tailles des données, LC est plus adapté. La justification de ces résultats est principalement liée à la logique de calcul d'OC et de SC qui consiste à transmettre via l'interface sans fil du drone la totalité des données volumineuses afin d'être traitées au niveau des véhicules au sol, référez-vous aux équations (4.7), (4.11) et (4.18), respectivement. Alors que dans le cas de LC, les tâches à calculer sont traitées durant un temps relativement réduit par le contrôleur SDN secondaire sur le drone et uniquement les petites tailles des résultats seront envoyées, voir l'équation (4.1). De plus, il est clair que notre politique DOSC permet de garantir le délai de calcul le plus optimal en sélectionnant la meilleure stratégie qui minimise le temps de calcul de la tâche.

Les résultats illustrés dans les figures 4.9 et 4.10, démontrent que notre algorithme DOSC permet d'obtenir le meilleur compromis entre la consommation énergétique et le délai de calcul.

Dans la figure 4.11, nous étudions l'impact de la décision de partage sur le délai de calcul moyen du contrôleur SDN principal en fonction de la taille des données de la tâche à calculer dans la sous-figure 4.11(a) et de la complexité de calcul en termes de cycles CPU dans la sous-figure 4.11(b).



(a) Taille des données *vs.* délai de calcul.



(b) Complexité CPU de calcul *vs.* délai de calcul.

FIGURE 4.11 – Impact de la décision de partage sur le délai de calcul du contrôleur SDN principal.

À partir des résultats illustrés dans les sous-figures 4.11(a) et 4.11(b), nous remarquons que le délai de calcul moyen augmente dans les scénarios de partage du calcul des tâches avec de grandes tailles des données et aussi lors du calcul local des tâches très complexes. Comme résultat, nous pouvons déduire que le calcul partagé est plus adapté aux tâches avec des calculs intensives et de petites tailles de données, tirant profit des avantages du calcul collaboratif distribué entre les contrôleurs SDN voisins. Au contraire, le calcul en local par le contrôleur SDN principal sur le véhicule de secours est plus approprié pour

les tâches avec des calculs moins intensives et de grandes tailles des données, souffrant de l'inconvénient de la transmission sans fil coûteuse des tâches avec des données volumineuses.

Dans l'expérimentation illustrée à la figure 4.12, nous évaluons le coût du système pour différents paramètres de pondération des mesures du système (délai et énergie) (α et β ($\beta = 1 - \alpha$)). Dans cette expérimentation, nous fixons la taille des données et la complexité des tâches et nous modifions à chaque fois les paramètres de pondération des mesures du système (α et β).

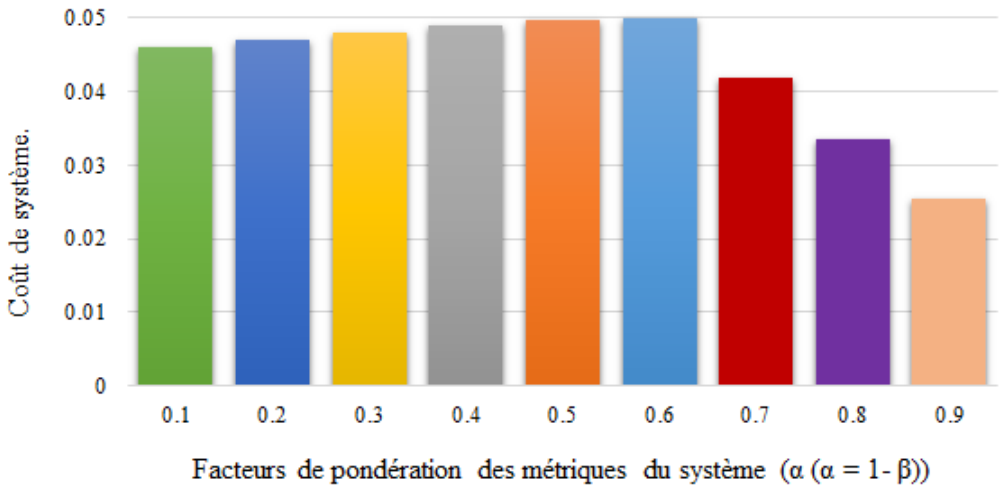


FIGURE 4.12 – Impact des facteurs de pondération des métriques du système sur la variation du coût de système.

À partir des résultats illustrés à la figure 4.12, nous remarquons que la variation des facteurs de pondération des métriques du système a un impact significatif sur la valeur du coût de système. Plus particulièrement, en définissant un facteur de pondération pour le délai (α) supérieur au facteur de pondération de l'énergie (β), notre politique DOSC peut améliorer jusqu'à 50% le coût du système par rapport à des valeurs égales des facteurs de pondération des métriques du système. Ces résultats confirment notre choix initial des valeurs des facteurs de pondération, où nous avons légèrement privilégié et attribué un poids plus important au délai de calcul par rapport à l'énergie du drone, compte tenu de la nature urgente, critique et sensible au délai du scénario étudié (une mission de secours suite à un accident sur la route). De même, notre modèle de jeu basé sur le paradigme de SDN, utilise ces paramètres de pondération comme une manière flexible pour équilibrer les mesures de performance du système en fonction du contexte en temps réel et afin de supporter d'une manière plus efficace les exigences de divers cas d'utilisation.

Dans l'expérimentation illustrée à la figure 4.13, nous simulons un simple scénario de transmission des données en mode multi-sauts à partir d'un drone à un véhicule au sol en passant par 4 sauts de véhicules. Nous comparons pour différentes tailles de données les performances de notre algorithme DOSC basé sur SDN en termes de délai de calcul moyen, (*i.e.*, le temps moyen jusqu'à ce que la destination obtienne les informations), avec deux

protocoles de transmission des données dans les VANETs assistés par des drones comme deux travaux de référence avec une sélection de chemin distribuée et qui n'utilise pas le paradigme de SDN : UVAR-S [77] et VDNet [36].

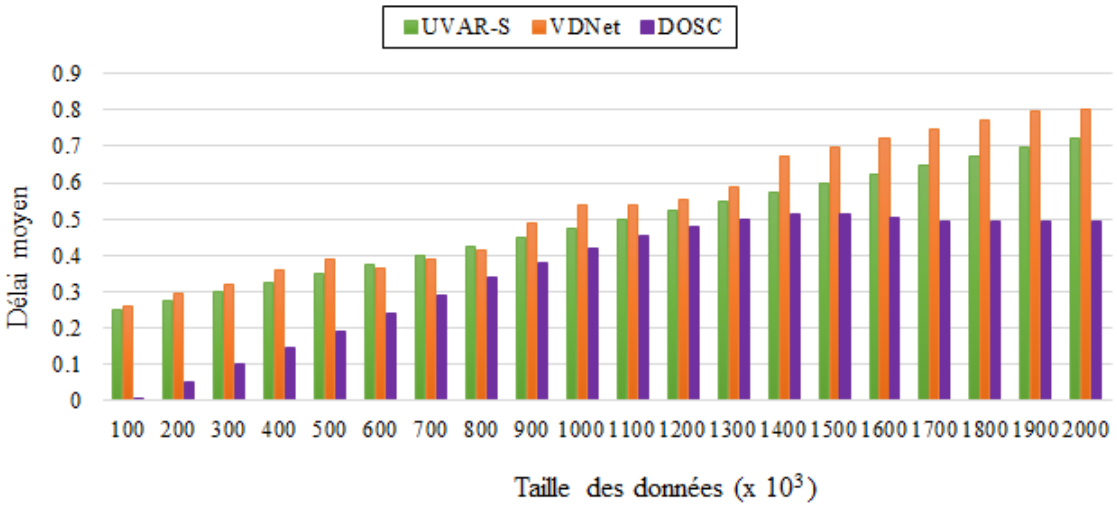


FIGURE 4.13 – Comparaison de notre politique de délestage/partage de calcul basée SDN avec des travaux sur la transmission des données dans les VANETs assistés par des drones.

Les résultats illustrés à la figure 4.13 montrent que notre algorithme DOSC basé sur le paradigme de SDN est le plus efficace et surpasse les travaux UVAR-S et VDNet tout en assurant de 35 à 95% de meilleur délai moyen de calcul. Cela, peut être justifié par le fait que notre algorithme DOSC, en utilisant le profil de décision obtenu dans l'EN, peut choisir la stratégie de calcul la plus efficace qui garantit le plus petit temps de réponse. Contrairement aux autres travaux d'UVAR-S et VDNet qui concentrent leurs efforts principalement sur la sélection et la disponibilité du chemin de transmission de bout en bout et qui ne se soucient pas de ce qui est mieux pour réduire le délai : le calcul en local des tâches avec de grands volumes des données par le drone et l'envoi seulement de petits tailles des résultats de calcul ou le calcul en délestage sur le véhicule de secours au sol après la transmission de la totalité des données quelque soient leurs tailles. En outre, les deux travaux relatifs et spécialement VDNet, présentent un délai supplémentaire en plus du délai de transmission qui est dû à un temps d'attente additionnel nécessaire pour exécuter deux phases périodiques pour la prédiction de l'emplacement de la destination et la sélection du chemin de bout en bout. Alors que, dans notre solution, toutes ces informations sont préalablement disponibles grâce à la vue globale et à jour de tout l'état de réseau détenu par les contrôleurs SDN. De plus, UVAR-S présente un petit avantage par rapport à VDNet en termes de délai, puisqu'il utilise la technologie DSRC caractérisée par une faible latence pour les communications entre les véhicules et même avec les drones. Enfin, il nous semble évident que le délai augmente avec l'augmentation de la taille des données, car la transmission de grande taille de données nécessite des délais plus importants.

4.7 Conclusion

Dans ce chapitre, nous nous sommes focalisés dans un premier temps sur l'étude d'un scénario de sécurité routière relatif à une mission de secours suite à un accident. Nous avons proposer d'incorporer des mini-drones afin d'assister les véhicules de secours à explorer et à enquêter sur les zones de la route dont l'accès est difficile à cause de la survenue d'un accident. Après, nous avons proposé une nouvelle architecture distribuée basée sur SDN pour les réseaux de véhicules assistés par des drones. Nous avons, ensuite, concentrés nos efforts dans un second temps, à la résolution du problème de garantie d'un traitement efficace pour les données collectées par le drone en termes d'optimisation de délai et d'énergie de calcul des tâches correspondantes. Nous avons formulé la politique de traitement de données comme un problème de prise de décision en termes de délestage/partage de calcul. Le principal objectif consiste à trouver le meilleur équilibre entre le délai de calcul et la consommation énergétique du drone. Nous avons modélisé le problème de décision de délestage/partage de calcul à l'aide d'un jeu séquentiel et nous avons étudié l'existence de l'équilibre de Nash. Nous avons également proposé deux algorithmes distribués pour résoudre le problème d'optimisation des résultats.

Les résultats numériques ont démontré que notre politique de traitement des données basée sur SDN assure des performances de calcul efficace en termes de délai et de l'énergie de calcul, tout en réalisant jusqu'à 28% du gain en termes de coût moyen du système et entre 35% et 95% de meilleur temps de réponse comparé aux autres scénarios de traitement de données naïfs et des travaux sur la diffusion des données dans les VANETs assistés par des drones [36, 77], respectivement.

Comme perspective future de ce travail, nous prévoyons d'étendre notre cas d'étude à un scénario plus général lorsque le véhicule de secours peut déployer une flotte de drones pour explorer des zones de la route plus vastes. Ces drones peuvent collaborer entre eux en partageant le calcul des tâches des données à traiter. Ainsi, nous généralisons notre jeu pour supporter N joueurs (plusieurs drones) avec de nouvelles stratégies comme le partage de calcul entre les drones. Il serait aussi plus intéressant de considérer l'énergie de déplacement du drone dans notre politique de calcul.

CHAPITRE 5

UN JEU DE *STACKELBERG* POUR L'INCITATION À LA MISE EN CACHE DANS LES RÉSEAUX D'INTERNET DE VÉHICULES DÉFINIS PAR LOGICIEL.

5.1 Introduction

L'Internet de véhicules (IoV) a émergé comme un concept prometteur pour améliorer les performances des nouvelles générations des réseaux de véhicules. Il devrait provoquer une avalanche du trafic de données mobiles principalement causée par une augmentation exponentielle du nombre de véhicules connectés et une demande explosive des applications gourmandes en bande passante telles que la vidéo à la demande, la vidéo en streaming, etc. Cependant et très souvent, un groupe de véhicules voisins est susceptible de manifester des intérêts semblables et demander des contenus similaires, conduisant à des transmissions redondantes des mêmes contenus parfois avec de grandes tailles de données [39]. Relever ces défis dans les IoVs, principalement soutenus par les communications cellulaires, devient une tâche difficile en raison de la grande surcharge du trafic des données sur les liaisons à capacité limitée des backhauls cellulaires [31, 97].

La technique de mise en cache (*caching*) a été proposée comme une technologie innovante pour supporter cette grande croissance de trafic de données. La mise en cache consiste à stocker les contenus populaires localement sur des nœuds physiquement proches de l'utilisateur final (station de base, point d'accès, véhicules, etc.) pour réduire les délais et alléger le backhaul cellulaire [40]. Cependant, une mise en cache réussie nécessite la coopération entre différents acteurs [40, 41]. En effet, les fournisseurs de contenu (*Content Provider*, CP), comme YouTube et Netflix, ont des contenus populaires et ont besoin de gestionnaires des caches, tels que les opérateurs de réseau mobile (*Mobile Network Operator*, MNO), afin de stocker leurs contenus populaires. Aussi, les MNOs ont besoin des CPs pour remplir et rentabiliser leurs caches ainsi que pour alléger leurs backhauls. Pour ce but, des mécanismes d'incitation devraient être développés pour encourager ces acteurs à collaborer afin d'assurer une mise en cache plus efficace [40].

Récemment, plusieurs travaux [39–45, 45], [97] ont considéré le problème de mise en cache avec une perspective commerciale en se concentrant sur la manière d'inciter les acteurs à participer à l'amélioration du processus de mise en cache. Cependant, la plupart de ces travaux considèrent le système de l'incitation à la mise en cache comme un marché monopolistique avec un MNO comme unique gestionnaire des caches. Ce qui représente une hypothèse peu réaliste dans les actuels marchés cellulaires et concurrentiels où plusieurs MNOs sont en concurrence permanente sur n'importe quelle source de revenus.

Motivés par cette discussion, nous proposons, dans ce chapitre une politique d'incitation à la mise en cache pour un système d'IoV basé sur SDN. Nous nous concentrons sur l'aspect économique de la mise en cache, en considérant un marché concurrentiel des propriétaires des caches, composé d'un seul CP et plusieurs MNOs. Le CP encourage les MNOs à stocker son contenu populaire dans les caches sur leurs petites stations de base (*Small Base Stations*, SBSs) avec la probabilité d'accès la plus élevée. Nous formulons l'interaction entre le CP et les MNOs en utilisant un jeu de type *Stackelberg* avec un sous jeu non-coopératif pour modéliser le conflit entre les MNOs. Nous explorons deux problèmes d'optimisation lorsque le CP vise à maximiser la satisfaction de ses utilisateurs en termes de délai d'accès/téléchargement, quand chaque MNO vise à maximiser son gain monétaire et enfin nous analysons l'équilibre de *Stackelberg*.

Le reste de ce chapitre est organisé comme suit. Nous donnons un aperçu sur les travaux connexes dans la section 5.2. Dans la section 5.3, nous décrivons le modèle du système et nous discutons la motivation de la politique d'incitation à la mise en cache compétitive. Nous formalisons dans la section 5.4 le jeu de type *Stackelberg* et nous analysons l'équilibre de *Stackelberg* dans le section5.5. La section 5.6 détaille les résultats numériques obtenus. Finalement, la conclusion clôture le chapitre dans la section 5.7.

5.2 Travaux connexes

Les problèmes de mise en cache sont sur-étudiés dans la littérature et la plupart des efforts se concentrent principalement sur les aspects techniques tels que le placement des caches. Récemment, quelques travaux [39–45, 45, 46] considèrent le problème de mise en cache d'un point de vue commercial en focalisant sur le problème de l'incitation des acteurs à améliorer le processus de mise en cache. De plus, la théorie des jeux est souvent utilisée comme un outil efficace pour étudier ce type de problème [46].

Dans [98], un jeu de type *Stackelberg* est proposé pour modéliser une politique d'incitation à la mise en cache dans laquelle un MNO offre un prix pour encourager les propriétaires des caches installés sur des SBSs à offrir l'espace de stockage et la bande passante nécessaire pour servir ses utilisateurs à proximité via des communications D2D (*Device to Device*). Chen et al. dans [99] ont proposé un mécanisme de l'incitation à la mise en cache en utilisant un jeu de type *Stackelberg*, où la SBS encourage et récompense les terminaux des utilisateurs pour qu'ils partagent davantage leur contenu en utilisant des communications D2D.

Les auteurs dans les travaux [40, 42, 43], ont proposé un mécanisme de l'incitation à la mise en cache proactive dans les réseaux cellulaires à petites cellules, afin de motiver les CPs à stocker leurs contenus populaires dans leurs caches sur les SBSs. L'interaction entre un MNO et plusieurs CPs est formulée en utilisant un jeu de type *Stackelberg* avec un sous-jeu non-coopératif pour modéliser le conflit entre les CPs sur l'espace de mise en cache disponible. Chaque CP vise à réduire le coût de la mise en cache et à maximiser la satisfaction de ses utilisateurs. Le MNO, en louant ses caches sur les SBSs aux CPs vise à maximiser son gain monétaire. Zhao et al. dans [44] ont utilisé un jeu de type *Stackelberg* pour modéliser l'interaction entre une SBS et plusieurs CPs dans un mécanisme d'incitation à la mise en cache commercial, en tenant compte de l'impact de la congestion sur les décisions des CPs.

Cependant, la quasi-totalité des travaux existants abordant les problèmes de l'incitation à la mise en cache commerciales considèrent un marché de mise en cache monopolistique avec un seul MNO qui détient et gère à lui seul tous les caches sur les SBSs. Cela semble loin de la réalité des marchés cellulaires actuels, où plusieurs MNOs sont en concurrence permanente sur n'importe quelle source de revenus. Récemment, Li et al. dans [39] ont formulé un jeu de type *Stackelberg* pour explorer une politique d'incitation à la mise en cache commerciale dans un réseau à petites cellules composé d'un distributeur de vidéo (*Video Retailer*, VR) et de plusieurs MNOs. Cependant, ce travail ne considère que la concurrence entre les MNOs sur le prix et ignore la quantité limitée de contenu populaire de VR. De plus, le paradigme de SDN a été utilisé dans [100] pour améliorer la mise en cache du contenu dans les réseaux de véhicules. Une approche de mise en cache basée sur SDN pour les réseaux centrés sur le contenu a été proposée dans [101], lorsque le contrôleur SDN a été utilisé pour fixer le seuil de popularité, déterminer les véhicules équipés avec des caches, ainsi que pour déclencher la mise en cache à la demande.

Motivé par la discussion ci-dessus, nous proposons dans ce chapitre une politique d'incitation à la mise en cache dans le contexte d'un réseau d'IoV basé sur SDN. Nous considérons un système de mise en cache concurrentiel composé d'un CP et de plusieurs MNOs qui offrent le service de mise en cache. Le CP encourage les MNOs à stocker son contenu populaire dans leurs caches sur des SBSs avec une popularité élevée. À notre connaissance, ce travail est la première tentative qui considère un problème de l'incitation à la mise en cache compétitive avec plusieurs gestionnaires des caches dans les scénarios d'IoV basés sur SDN.

5.3 Formulation du problème

Dans cette section, nous commençons par présenter le modèle du système de notre politique d'incitation à la mise en cache pour les scénarios IoVs basés sur SDN. L'architecture physique du modèle de système est une adaptation pour le contexte de l'IoV de notre architecture CSDHVN présentée précédemment dans la section 3.3 du chapitre 3, que nous appelons *Hierarchical Software-Defdined Inernet of Vehicle, HSD-IoV*. Ensuite, nous décrirons notre politique d'incitation à la mise en cache comme un cas d'application de cette architecture, et enfin, nous modélisons la distribution de la popularité du contenu.

5.3.1 Modèle du système

Comme illustré à la figure 5.1, nous considérons un système de l'incitation à la mise en cache pour les réseaux IoVs basés sur SDN composé d'un CP et de multiples MNOs. Chaque MNO est responsable d'un ensemble de SBSs géographiquement distribuées, qui desservent un ensemble de véhicules. Les véhicules sont hétérogènes et ont plusieurs abonnements avec différents MNOs. Chaque véhicule à un instant t est desservi par une SBS d'un MNO particulier. Notons $\mathcal{M} = \{1, 2, \ldots, M\}$ l'ensemble des M MNOs et $\mathcal{V} = \{1, 2, \ldots, \vartheta\}$ l'ensemble des véhicules hétérogènes.

Sans perte de généralité, nous supposons que tous les MNOs ont un ensemble de B SBSs, $\mathcal{B} = \{B_1, B_2, \ldots, B_M\}$, tel que $B_M = \{1, \ldots, B\}$ désigne le sous-ensemble des SBSs de MNO m, $\cap_{m\in\mathcal{M}} B_M = \emptyset$ *et* $\cup_{m\in\mathcal{M}} B_M = \mathcal{B}$. Chaque SBS i de MNO j est équipée d'un cache avec une capacité de stockage de taille limitée, notée $sc_{i,j}$ ($sc_{i,j} > 0$), pouvant servir de mémoire pour stocker le contenu populaire du CP. Tous les caches sur les SBSs sont supposés avoir la même capacité de stockage limitée [105]. Le CP gère localement une bibliothèque $\mathcal{L}$ composée de l'ensemble fini de ces $F (F < \infty)$ fichiers populaires (notés c_f, $0 < f \leq F$), tel que $\mathcal{L} = \{c_1, c_2, \ldots, c_F\}$. Tous les fichiers c_f ($f \in F$) sont assumés avoir la même unité de taille [105].

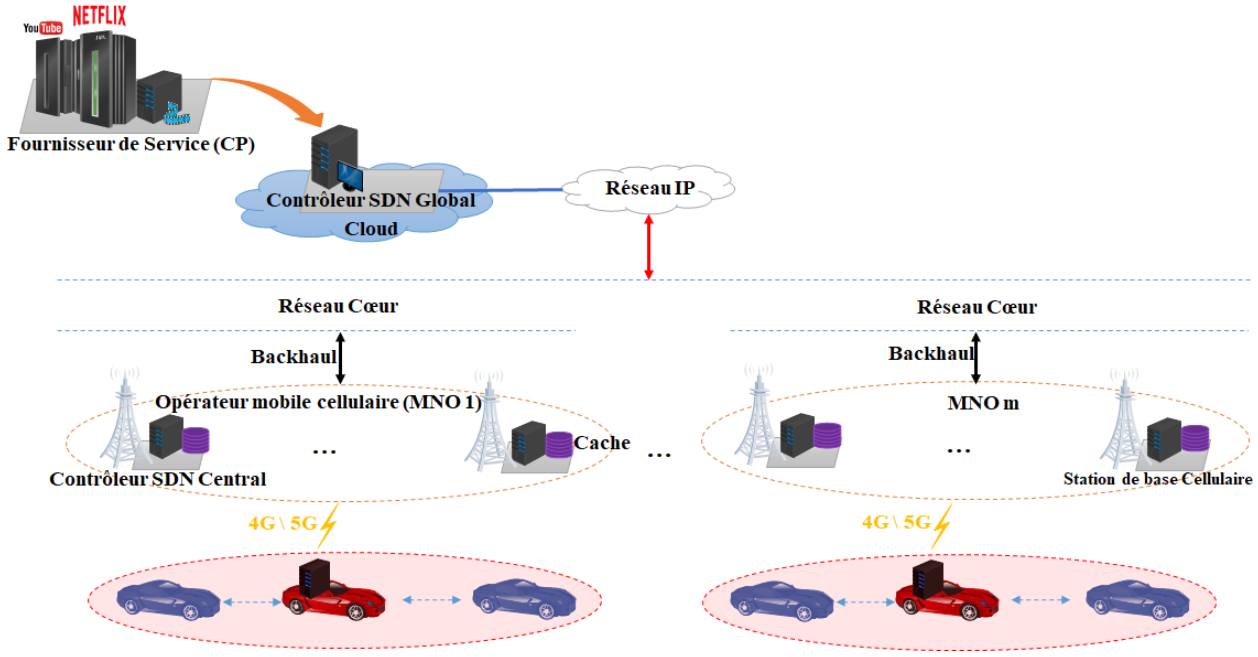


FIGURE 5.1 – Modèle du système de la politique d'incitation à la mise en cache dans les réseaux IoVs basés sur SDN.

Dans notre système de l'incitation à la mise en cache, chaque cache est géré par un contrôleur SDN central. Le contrôleur SDN global gère les interactions avec le CP. Les contrôleurs SDN centraux sont connectés au contrôleur SDN global via la backhaul à capacité limitée du réseau cœur de MNO en utilisant une fibre optique ou un lien cellulaire. Les véhicules communiquent avec le contrôleur SDN central via une interface cellulaire 4G/5G [40]. Nous supposons que toutes les communications sont fiables.

5.3.2 Politique d'incitation à la mise en cache basée sur SDN

L'idée de base de notre politique de mise en cache consiste à placer le contenu populaire du CP à la périphérie du réseau, *i.e.*, sur les SBSs équipés par les mémoires caches des MNOs, le plus proche possible des véhicules [40]. En effet, le CP a besoin des MNOs pour leurs SBSs équipés par les mémoires caches afin de stocker son contenu populaire et améliorer la satisfaction de ces utilisateurs. Les MNOs ont également besoin du CP pour son contenu populaire afin de remplir et rentabiliser leurs caches sur les SBSs. Par conséquent, ces deux acteurs ont intérêt à collaborer entre eux pour un déploiement réussi et efficace de la mise en cache.

Motivé par cette discussion, nous proposons une politique d'incitation à la mise en cache lorsque le CP encourage et incite les MNOs pour stocker son contenu populaire localement dans les caches installés sur les SBSs avec une probabilité d'accès élevée. En mettant en cache son contenu populaire, le CP vise à améliorer la satisfaction de ses utilisateurs à travers l'amélioration de leurs qualité d'expérience (*Quality of Experience*, QoE), en profitant d'un débit de transmission plus élevé, d'un délai d'accès/de téléchargement plus rapide, etc., tout en essayant de minimiser le coût de la mise en cache. Effectivement, le CP récompense les MNOs en fonction du nombre de contenus qu'ils acceptent de stocker en cache en payant un prix monétaire pour chaque fichier stocké. Les MNOs, en acceptant de louer leurs espaces de stockage des caches sur les SBSs, visent à gagner plus de profit monétaire et à décharger leur backhaul en servant la plupart des demandes des utilisateurs localement sur les SBSs du réseau d'accès.

Dans notre système de l'incitation à la mise en cache basé sur l'architecture HSD-IoV, le contrôleur SDN global va représenter le CP en défendant ses intérêts, alors que chaque contrôleur SDN central sur une SBS équipée avec un cache va représenter le MNO de la SBS qu'elle héberge en défendant ses intérêts. De même, l'utilisation de SDN permet une prise de décision de mise en cache plus informée grâce à la vue globale centralisée en temps réel sur l'état de réseau du plan de contrôle. En effet, les contrôleurs SDN sont supposés connaître à tout moment les préférences de chaque véhicule et la charge du trafic (*i.e.*, la probabilité ou la popularité d'accès) de chaque contenu du CP pour toutes les zones du réseau, en gérant l'historique des requêtes des véhicules. Il faut bien noter que la taille des contenus populaires du CP annoncés pour la mise en cache est limitée, et elle est inférieure à la capacité de stockage totale de tous les caches sur les SBSs des MNOs. Par conséquent, les contrôleurs SDN centraux en défendant les intérêts des MNOs entrent en compétition sur cette ressource limitée afin de remplir au maximum leurs mémoires caches.

De plus, la popularité d'un contenu peut varier d'une SBS à une autre en fonction des préférences des véhicules. Pour cela, le CP via le contrôleur SDN global préfère stocker ses fichiers les plus populaires sur le cache de la SBS avec la probabilité d'accès la plus élevée. Ce dernier doit respecter au minimum un seuil de popularité de mise en cache τ. Cela signifie que seul un contenu ayant été demandé un nombre de fois égal ou supérieur au seuil de popularité τ fixé par le CP sera stocké localement.

En outre, étant donné la variété de la popularité des fichiers et la capacité de stockage limitée des caches sur les SBSs, il se trouve que c'est dans l'intérêt du CP de distribuer ses contenus populaires sur diverses SBSs de différents MNOs avec plusieurs copies afin de satisfaire le maximum de requêtes des véhicules. Ainsi, nous supposons qu'un fichier i peut être mis en cache par le MNO m, fi fois en fonction de sa popularité [40].

5.3.3 Popularité du contenu

La distribution de la popularité du contenu de CP représente la distribution des probabilités de sélection des demandes des véhicules parmi le contenu populaire du CP dans la bibliothèque $\mathcal{L}$. Cette popularité est représentée par un vecteur $p^m = [p_1^m, p_2^m, \ldots, p_F^m]$. Les véhicules desservis par un contrôleur SDN central sur une SBS de MNO m équipée d'un cache demandent indépendamment le $f - ième$ item de contenu c_f, $f = 1, \ldots, F$, avec la probabilité p_f^m, tel que $\sum_{f \in F} p_f^m = 1$. La popularité du contenu est généralement modélisée en utilisant une distribution de la loi *Zipf* tel que dans les travaux [39,102]. La popularité de $f - ième$ contenu par rapport aux abonnés de MNO m, p_f^m, est donnée par :

$$p_f^m = \frac{c_f^{-\delta}}{\sum_{j=1}^{F} c_j^{-\delta}}, \quad \forall \ \ 0 < f \leq F \tag{5.1}$$

Tel que : $f \leq F$ est le nombre des fichiers dans la bibliothèque $\mathcal{L}$ et δ représente l'exposant de la loi *Zipf*, caractérisant la popularité de contenu. δ peut varier d'un MNO à un autre. Les fichiers les plus populaires avec des requêtes plus importantes correspondent à une valeur de δ plus élevée. Autrement dit, la distribution de la loi *Zipf* trie le contenu de CP par ordre de popularité décroissant, *i.e.*, le contenu avec un f (l'ordre dans la bibliothèque) plus petit correspond à une plus grande popularité. La popularité globale du contenu de tous les MNOs est donnée par le vecteur $\mathcal{P} = [p^1, p^2, \ldots, p^m]$, qui est géré par le contrôleur SDN global basé sur l'historique des requêtes des véhicules.

Dans la littérature, la théorie des jeux et particulièrement le jeu de type *Stackelberg* est souvent utilisé comme un outil puissant pour modéliser l'interaction entre les différents acteurs dans l'étude des problèmes d'incitation à la mise en cache [40, 44, 46]. Dans le même contexte, nous présentons dans la section suivante un jeu de type *Stackelberg* pour modéliser notre politique d'incitation à la mise en cache.

5.4 Jeu de *Stackelberg* pour l'incitation à la mise en cache dans les réseaux d'IoVs définis par logiciel

Sur la base du modèle de l'incitation à la mise en cache basé sur SDN présenté ci-dessus, nous formulons l'interaction entre le CP et les MNOs sous forme d'un jeu de type *Stackelberg*. Un jeu de *Stackelberg* est un jeu séquentiel qui modélise l'interaction compétitive entre deux types des acteurs rationnels, à savoir le leader et les suiveurs. Le leader a l'avantage d'agir en premier alors que les suiveurs réagissent après.

Le jeu de type *Stackelberg* proposé est joué entre le CP représenté par le contrôleur SDN global et les propriétaires des caches installés sur les SBSs, autrement dit, les M

MNOs, représentés par les contrôleurs SDN centraux comme suit. Le contrôleur SDN global agit en premier comme le leader en prédisant les stratégies des suiveurs avant d'annoncer sa stratégie initiale. La stratégie du leader consiste à annoncer la quantité de contenus populaires qu'il souhaite la mettre en cache (Q^R) et le seuil de popularité (τ), une popularité minimale sous elle un contenu ne peut pas être mis en cache. Le leader vise à maximiser la satisfaction de ses utilisateurs en maximisant la quantité de son contenu le plus populaire stocké en cache tout en essayant de minimiser le coût monétaire de la mise en cache. Ensuite, compte tenu de la stratégie du leader, les contrôleurs SDN centraux répondent comme des suiveurs en réagissant avec leurs meilleures stratégies. Tous les suiveurs ont la même stratégie qui consiste à déterminer la quantité optimale du contenu populaire du CP (Q^C) qu'ils acceptent de mettre en cache selon le seuil de popularité (τ) et le prix optimal de mise en cache (Π). Ce dernier est fixé par le MNO sur la base de la popularité de contenu de CP annoncé pour la mise en cache et l'espace de stockage disponible. Chaque suiveur vise à maximiser son revenu monétaire en maximisant la quantité de contenu stockée. Enfin, le leader optimise son utilité en fonction des meilleures réponses des suiveurs par l'ajustement du seuil optimal de popularité de la mise en cache et par la suite il peut déduire le nombre optimal des fichiers les plus populaires qu'il va réellement mettre en cache. Il faut bien noter que tous les acteurs sont rationnels avec des stratégies conflictuelles et visent à optimiser leurs propres stratégies. Cependant, en raison du nombre limité de contenus populaires annoncés pour la mise en cache par le CP, les contrôleurs SDN centraux (les MNOs) sont considérés comme des acteurs compétitifs en concurrence sur cette quantité limitée. Par conséquent, un sous-jeu non-coopératif est formulé pour représenter l'intéraction entre les contrôleurs SDN centraux.

Dans ce qui suit, nous fournissons les fonctions d'utilité du leader et des suiveurs, respectivement. La fonction d'utilité est exprimée comme une fonction générale, telle que *Utilité = Gain - Coût*.

5.4.1 Modèle du leader

Comme il est décrit ci-dessus, le contrôleur SDN global agit en premier en annonçant sa stratégie qui consiste à la quantité du contenu populaire qu'il souhaite mettre en cache (Q^R) et le seuil de popularité (τ).

5.4.1.1 Gain

Le gain du leader peut être exprimé à travers sa fonction de satisfaction qui reflète la satisfaction de ses utilisateurs. En effet, il est approprié de modéliser le gain de sorte que le CP se sent plus satisfait lorsque la quantité de son contenu stockée et la probabilité d'accès à ce contenu augmentent et ce gain devient insignifiant lorsque la quantité stockée en cache augmente de manière que les fichiers stockés deviennent redondants [41]. La fonction de satisfaction ou de récompense (gain) du leader (CP), notée $\mathcal{R}_{cp}$, peut être facilement définie comme la fonction de satisfaction moyenne de tous les suiveurs. Nous modélisons cette fonction de satisfaction comme dans [41] à travers une fonction logarithmique monotone croissante en fonction de la quantité de contenu populaire stockée en cache par le MNO i selon le seuil de popularité τ (noté $Q^C_{MNO,i}$), ainsi qu'en fonction de

la popularité du contenu annoncé à la mise en cache Q^R (noté $\sigma_{MNO,i}$). Cette fonction de satisfaction augmente rapidement jusqu'à un certain seuil lorsque la quantité et la popularité des fichiers cachés augmentent, puis augmente lentement lorsque les fichiers stockés deviennent redondants. $\mathcal{R}_{cp}$ est donné par :

$$R_{cp}\left(Q^R, \quad \tau\right) = \sum_{i=1}^{M} \frac{log\left(1 + \left((\sigma_{MNO,i} - \tau) \times Q^C_{MNO,i}\right)\right)}{M} \tag{5.2}$$

Tel que

$$\sigma_{MNO,i} = \sqrt{\frac{\sum_{j \in Q^R} \left(p_{f_{j,i}} - \frac{\sum_{j=1}^{Q^R} p_{f_{i,j}}}{Q^R}\right)^2}{Q^R}} \tag{5.3}$$

$\sigma_{MNO,i}$ et $p_{f_{j,i}}$ dénote l'écart type de la popularité de contenus annoncés par le CP pour la mise en cache (Q^R) et la charge de trafic générée (*i.e.*, la probabilité d'accès) du fichier j par les utilisateurs du CP affectés à l'opérateur MNO i, respectivement.

5.4.1.2 Coût

Le principal coût du leader est la charge monétaire globale de la mise en cache qui doit payer pour les suiveurs en retour de chaque fichier stocké. Ce coût de mise en cache, noté $\mathcal{C}_{cp}$, est modélisé comme une fonction croissante de la quantité de contenu stockée et le prix de la mise en cache correspondant de tous les M MNOs. $\mathcal{C}_{cp}$ est donné par :

$$\mathcal{C}_{cp}\left(Q^C_{MNO,i}, \Pi_{MNO,i}\right) = \sum_{i=1}^{M} \Pi_{MNO,i} \times Q^C_{MNO,i} \tag{5.4}$$

Tel que $\Pi_{MNO,i}$ représente le prix de mise en cache que le leader doit payer pour le suiveur i pour chaque fichier stocké.

5.4.1.3 Fonction d'utilité

La fonction d'utilité de leader, notée μ_{cp}, peut être facilement modélisée comme une fonction croissante de la récompense de la satisfaction et une fonction décroissante du coût monétaire de la mise en cache. μ_{cp} est donnée par :

$$\mu_{cp}\left(Q^R, \tau\right) = \mathcal{R}_{cp}\left(Q^R, \tau\right) - \mathcal{C}_{cp}\left(Q^C_{MNO,i}, \Pi_{MNO,i}\right) \tag{5.5}$$

Tel que le revenu $\mathcal{R}_{cp}$ et le coût $\mathcal{C}_{cp}$ sont définis dans les deux équations (5.2) et (5.4), respectivement.

5.4.2 Modèle des suiveurs

Compte tenu de la stratégie annoncée par le leader, tous les suiveurs entrent en concurrence sur la quantité limitée du contenu de CP annoncée pour la mise en cache et répondent simultanément avec leurs stratégies optimales. Ces dernières sont représentées par un profil de mise en cache défini par le duplet $\left(Q^C, \Pi\right)$), tel que $Q^C \triangleq$

$\left(Q^C_{MNO,1}, Q^C_{MNO,2}, \ldots, Q^C_{MNO,M}\right)$ et $\Pi \triangleq (\Pi_{MNO,1}, \Pi_{MNO,2}, \ldots, \Pi_{MNO,M})$ indique la quantité stockée en cache et le prix de mise en cache de chaque suiveur, respectivement.

5.4.2.1 Gain

Le gain du suiveur i, noté $\mathcal{R}_{MNO,i}$, est défini comme une fonction monotone croissante du gain monétaire payé par le leader pour la quantité de son contenu populaire stockée en cache. $\mathcal{R}_{MNO,i}$ est donné par :

$$\mathcal{R}_{MNO,i}\left(Q^C_{MNO,i}, \Pi_{MNO,i}\right) = \Pi_{MNO,i} \times Q^C_{MNO,i} \tag{5.6}$$

Le prix de mise en cache est défini par chaque suiveur comme une fonction sigmoïde monotone croissante dont la valeur est dans l'intervalle [0, 1], en fonction de la charge du trafic de contenu annoncé pour la mise en cache et la quantité de stockage disponible. La fonction sigmoïde est généralement utilisée pour modéliser des phénomènes où il existe un seuil et différents niveaux [103, 104]. Le suiveur (MNO) i charge un prix plus élevé pour stocker un contenu très populaire de CP avec une probabilité d'accès élevée (la charge de trafic, $\sigma_{MNO,i}$) afin de compenser l'énergie supplémentaire nécessaire pour gérer l'augmentation de la charge du trafic et les interférences résultats [41]. De même, le prix de la mise en cache peut être classé en fonction de la capacité de stockage disponible en mémoire cache de MNO i ($\varsigma^A_{MNO,i}$) en plusieurs niveaux avec un prix différent pour chaque niveau. Cela signifie que plus la capacité du stockage disponible est limitée, plus le prix de stockage sera élevé. Le prix de mise en cache est donné par :

$$\Pi_{MNO,i} = \frac{1}{1 + e^{-\frac{1}{\varsigma^A_{MNO,i}} \times \sigma_{MNO,i}}} \tag{5.7}$$

Tel que $\sigma_{MNO,i}$ est l'écart type de la probabilité d'accès au contenu stocké dans le cache de MNO i et il est défini à l'équation (5.3).

5.4.2.2 Coût

La dépense principale du suiveur i, noté $\mathcal{C}_{MNO,i}$, est le coût de stockage en cache. Il est défini comme dans [40] par une fonction barrière de la capacité de mise en cache de tous les caches sur les SBSs de MNO i et du nombre de fois que chaque fichier est stocké en cache.

$$\mathcal{C}_{MNO,i}\left(Q^C_{MNO,i}\right) = \begin{cases} \dfrac{1}{\varsigma^A_{MNO,i} - Q^C_{MNO,i}} & Si \;\; 0 < Q^C_{MNO,i} < \varsigma^A_{MNO,i} \\ \infty & Autrement \end{cases} \tag{5.8}$$

Tel que $\varsigma^A_{MNO,i}$ représente la taille de stockage disponible de tous les caches sur les SBSs de MNO i et il est donné par :

$$\varsigma^A_{MNO,i} = \sum_{j=1}^{B} \left(sc_{i,j-\beta}\right) \tag{5.9}$$

Et

$$Q^{C}_{MNO,i} = \sum_{j=1}^{B} \sum_{k \in Q^{C}_{MNO,i} < F} fl_{i,j,k} \tag{5.10}$$

β représente la quantité d'espace de stockage occupée d'un cache de MNO i, et $fl_{i,j,k}$ représente le nombre des copies de fichier k qui est stocké par le MNO i sur le cache de la SBS j.

5.4.2.3 Fonction d'utilité

La fonction d'utilité du suiveur i, notée $\mu_{MNO,i}$, peut être facilement définie comme une fonction croissante de la récompense de mise en cache et une fonction décroissante de la capacité de stockage des caches. $\mu_{MNO,i}$ est donné par :

$$\mu_{MNO,i} = \mathcal{R}_{MNO,i}\left(Q^{C}_{MNO,i}, \Pi_{MNO,i}\right) - \mathcal{C}_{MNO,i}\left(Q^{C}_{MNO,i}\right) \tag{5.11}$$

Tel que $\mathcal{R}_{MNO,i}$ et $\mathcal{C}_{MNO,i}$ sont définis dans les deux équations (5.6) et (5.8), respectivement.

Il faut bien noter que dans notre processus de mise en cache, les joueurs doivent s'assurer que leur utilité est non-négative en garantissant que le gain couvre au moins le coût [41].

5.4.3 Sous-jeu non-coopératif des suiveurs

Un sous-jeu non-coopératif est utilisé pour modéliser la concurrence entre les suiveurs sur la quantité limitée du contenu populaire de CP.

Définition 1. le sous-jeu non-coopératif entre les suiveurs est défini par le triplet $G = (\mathcal{M}, \mathcal{S}, \mu)$:

- $\mathcal{M} = \{1, \ldots, M\}$, représente l'ensemble fini non-vide des suiveurs, *i.e.*, les sous-ensembles des contrôleurs SDN centraux représentant les M MNOs ;
- $\mathcal{S} = \{s_1, \ldots, s_M\}$, représente l'ensemble fini non-vide des stratégies des M joueurs. s_i est la stratégie de joueur i, définie par le duplet $\left(Q^{C}_{MNO,i}, \Pi_{MNO,i}\right)$, qui représente la quantité de contenu populaire stockée en cache et le prix correspondant ; et
- $\mu = \{\mu_{MNO,1}, \ldots, \mu_{MNO,M}\}$, représente les fonctions d'utilité des M joueurs. La fonction d'utilité du joueur i est définie dans l'équation (5.11).

Pour un jeu non-coopératif à stratégie pure, l'équilibre de Nash est la solution standard pour l'état d'équilibre. L'EN représente un état dans lequel aucun MNO ne peut améliorer son bénéfice en stockant une quantité différente des fichiers populaires du CP compte tenu de la quantité stockée par les autres MNOs [40].

Un équilibre de Nash à stratégie pure de sous-jeu non-coopératif des suiveurs est un profil de décision de la quantité stockée et le prix de mise en cache :
$d = \left(Q^{C^*} \triangleq \left\{Q^{C^*}_{MNO,1}, \ldots, Q^{C^*}_{MNO,m}\right\}, \Pi^* \triangleq \left\{\Pi^*_{MNO,1}, \ldots, \Pi^*_{MNO,m}\right\}\right), \forall m \in M$ nous avons ce qui suit :

$$\begin{aligned}\mu_{MNO,m}\left(\left(Q^{C*}_{MNO,m},\Pi^{*}_{MNO,m}\right),\left(Q^{C*}_{MNO,-m},\Pi^{*}_{MNO,-m}\right)\right)\geq\\ \mu_{MNO,m}\left(\left(Q^{C}_{MNO,m},\Pi_{MNO,m}\right),\left(Q^{C*}_{MNO,-m},\Pi^{*}_{MNO,-m}\right)\right),\\ \forall\left(Q^{C}_{MNO,m},\Pi_{MNO,m}\right)\in Q^{C}\times\Pi\end{aligned}\tag{5.12}$$

Dans la section suivante, nous dérivons les stratégies optimales du leader et des suiveurs.

5.5 Stratégies optimales de la mise en cache

Le jeu du type *Stackelberg* proposé dans ce chapitre comprend deux sous-problèmes d'optimisation. Le leader vise à maximiser la satisfaction de ses utilisateurs en maximisant la quantité de ses contenus les plus populaires qui peuvent être mis en cache en fonction du seuil de popularité le plus haut, tout en essayant de minimiser autant que possible le coût monétaire de la mise en cache. Les suiveurs visent à gagner plus de revenus monétaires en maximisant la quantité de contenu stockée, tout en réduisant le coût de la mise en cache en termes de l'espace de stockage. Le but principal de ce jeu est de définir l'équilibre de *Stackelberg* (ES). L'ES représente un état mutuellement satisfaisant dont ni le leader (CP) ni les suiveurs (MNOs) ont intérêt à dévier unilatéralement [43, 99]. L'ES est exactement le point que le leader souhaite atteindre [40]. L'ES de notre sous-jeu non-coopératif est défini comme suit [99] :

Définition 2. Soit $\left(Q^{C*},\Pi^{*}\right)$, *tel que* $Q^{C*}\triangleq\left[Q^{C*}{}_{MNO,1},Q^{C*}{}_{MNO,2},\ldots,Q^{C*}{}_{MNO,m}\right]$ *et* $\Pi^{*}\triangleq\left[\Pi^{*}_{MNO,1},\Pi^{*}_{MNO,2},\ldots,\Pi^{*}_{MNO,m}\right]$ une solution pour le sous-jeu des suiveurs dans l'étape II, et soit $\left(Q^{R*},\tau^{*}\right)$ la solution de problème d'optimisation du leader dans l'étape I. Le profil de la stratégie $\left(\left(Q^{R*},\tau^{*}\right),\left(Q^{C*},\Pi^{*}\right)\right)$, est un équilibre de *Stackelberg* si et seulement si :

$$\begin{aligned}\mu_{cp}\left(\left(Q^{R*},\tau^{*}\right),\left(Q^{C*},\Pi^{*}\right)\right)\geq\mu_{cp}\left(\left(Q^{R},\tau\right),\left(Q^{C*},\Pi^{*}\right)\right),\quad et\\ \mu_{MNO,m}\left(\left(Q^{C*}{}_{MNO,m},\Pi^{*}_{MNO,m}\right),\left(Q^{C*}{}_{MNO,-m},\Pi^{*}_{MNO,-m}\right),\left(Q^{R*},\tau^{*}\right)\right)\geq\\ \mu_{MNO,m}\left(\left(Q^{C}_{MNO,m},\Pi_{MNO,m}\right),\left(Q^{C*}{}_{MNO,-m},\Pi^{*}_{MNO,-m}\right),\left(Q^{R*},\tau^{*}\right)\right),\forall m\in M\end{aligned}\tag{5.13}$$

Nous utilisons la technique de la récurrence à rebours (backward induction) pour résoudre notre jeu du type *Stackelberg* avec information complète, le leader et les suiveurs sont en connaissance de tous les paramètres du système, grâce à la connaissance globale de SDN. Pour cela, nous commençons à chercher la meilleure stratégie du suiveur avant d'explorer comment optimiser la stratégie du leader.

5.5.1 Stratégie optimale du suiveur

Étant donnée la stratégie annoncée par le leader (la quantité de son contenu populaire et le seuil de popularité), les suiveurs vont répondre avec leurs stratégies optimales (le prix de la mise en cache et la quantité de contenu populaire qu'ils peuvent stocker). En effet, chaque suiveur commence premièrement par définir indépendamment le prix optimal

qui va charger pour être payé par le CP en contrepartie du stockage de chaque fichier en utilisant :

$$\Pi^{*}_{MNO,i}\left(\sigma_{MNO,i}\right) = \frac{1}{1+e^{-\frac{1}{\varsigma^{A}_{MNO,i}}\times\sigma_{MNO,i}}} \tag{5.14}$$

Une fois le prix optimal de la mise en cache ($\Pi^{*}_{MNO,i}$) est défini, tous les suiveurs vont entrer en compétition entre eux pour la quantité limitée de contenu populaire de CP annoncée pour la mise en cache. La quantité optimale stockée par chaque suiveur i peut être dérivée en se basant sur le prix défini précédemment en résolvant le problème suivant :

$$\begin{aligned} Q^{C^{*}}_{MNO,i} = \underset{Q^{C}_{MNO,i}}{\arg\max} \;\; & \mu_{MNO,i}\left(Q^{C}_{MNO,i},\;\; \Pi_{MNO,i}\right) \\ s.t. \;\; & 0 < Q^{C}_{MNO,i} \leq \varsigma^{A}_{MNO,i} \\ & Q^{C}_{MNO,i} \leq Q^{R} \\ & \Pi_{MNO,i} > 0 \end{aligned} \tag{5.15}$$

Tel que $\mu_{MNO,i}$ est défini dans l'équation (5.11).

Pour résoudre le problème d'optimisation des suiveurs dans l'équation (5.15), nous étudions dans une première étape la meilleure réponse de la stratégie du suiveur i dans un instant t avant d'explorer la stratégie optimale dans tous les instants conjointement (l'EN) dans la deuxième étape [92]. Pour ce but, nous vérifions tout simplement si ce problème est une optimisation convexe en procédant par le calcul des dérivées partielles du premier et de second ordre de $\mu_{MNO,i}$ pour le suiveur i par rapport à $Q^{C}_{MNO,i}$ en fixant $\Pi_{MNO,i}$.

$$\frac{d\mu_{MNO,i}\left(Q^{C}_{MNO,i},\;\; \Pi_{MNO,i}\right)}{dQ^{C}_{MNO,i}} = \Pi_{MNO,i} - \frac{1}{\left(\varsigma^{A}_{MNO,i} - Q^{C}_{MNO,i}\right)^{2}} \tag{5.16}$$

$$\frac{d^{2}\mu_{MNO,i}\left(Q^{C}_{MNO,i},\;\; \Pi_{MNO,i}\right)}{d^{2}Q^{C}_{MNO,i}} = -\frac{2}{\left(\varsigma^{A}_{MNO,i} - Q^{C}_{MNO,i}\right)^{3}} \tag{5.17}$$

Il est évident que $\dfrac{d^{2}\mu_{MNO,i}\left(Q^{C}_{MNO,i},\;\; \Pi_{MNO,i}\right)}{d^{2}Q^{C}_{MNO,i}} < 0$ puisque $\varsigma^{A}_{MNO,i} - Q^{C}_{MNO,i} > 0$. Donc, la fonction $\mu_{MNO,i}\left(Q^{C}_{MNO,i},\;\; \Pi_{MNO,i}\right)$ est concave par rapport à la quantité des fichiers stockés et donc elle accepte une solution optimale. Ainsi, au moins une quantité maximale globale existe $Q^{C^{*}}_{MNO,i}$ qui peut être obtenue par la condition de l'optimisation sous contraintes de *Karush-Kuhn-Tucker*. Par conséquent, l'équilibre de Nash de sous-jeu non-coopératif existe [92].

La meilleure décision du suiveur i peut être facilement obtenue en résolvant :

$$\frac{d\mu_{MNO,i}\left(Q^{C}_{MNO,i},\;\; \Pi_{MNO,i}\right)}{dQ^{C}_{MNO,i}} = \Pi_{MNO,i} - \frac{1}{\left(\varsigma^{A}_{MNO,i} - Q^{C}_{MNO,i}\right)^{2}} = 0 \tag{5.18}$$

$$\frac{1}{\Pi_{MNO,i}} - \left(\varsigma^{A}_{MNO,i} - Q^{C}_{MNO,i}\right)^2 = 0, \quad \frac{1}{\sqrt{\Pi_{MNO,i}}} = \left|\varsigma^{A}_{MNO,i} - Q^{C}_{MNO,i}\right| \tag{5.19}$$

$$Q^{C^*}_{MNO,i} = \varsigma^{A}_{MNO,i} - \frac{1}{\sqrt{\Pi_{MNO,i}}} \tag{5.20}$$

Il faut bien noter qu'uniquement une quantité entière positive du contenu annoncé à la mise en cache est considérée dans notre modèle. Autrement dit, un fichier est stocké seulement comme un segment tout entier ou sous forme de plusieurs segments de tailles égales. Pour cette raison, la fonction round $\lfloor\ \rceil$ est appliquée à la quantité de contenu stockée avant la réponse du suiveur [40]. De plus, chaque MNO peut stocker uniquement une partie de la quantité annoncée pour la mise en cache $\left(0 \leq Q^{C}_{MNO,i} < \varsigma^{A}_{MNO,i}\right)$. Donc, le résultat $Q^{C^*}_{MNO,i} = \varsigma^{A}_{MNO,i} + \frac{1}{\sqrt{\Pi_{MNO,i}}}$ est ignoré.

La stratégie optimale des suiveurs dans l'EN peut être exprimée via le système d'équations linéaires suivant :

$$\begin{cases} 1 \times Q^{C}_{MNO,1} + 0 \times Q^{C}_{MNO,2} + \cdots + 0 \times Q^{C}_{MNO,M} = \varsigma^{A}_{MNO,1} - \frac{1}{\sqrt{\Pi_{MNO,1}}} \\ 0 \times Q^{C}_{MNO,1} + 1 \times Q^{C}_{MNO,2} + \cdots + 0 \times Q^{C}_{MNO,M} = \varsigma^{A}_{MNO,2} - \frac{1}{\sqrt{\Pi_{MNO,2}}} \\ \vdots \\ 0 \times Q^{C}_{MNO,1} + 0 \times Q^{C}_{MNO,2} + \cdots + 1 \times Q^{C}_{MNO,M} = \varsigma^{A}_{MNO,M} - \frac{1}{\sqrt{\Pi_{MNO,M}}} \end{cases} \tag{5.21}$$

Nous transformons les m équations du système linéaire à une représentation sous forme matricielle dénotée $A.B = C$, tel que la matrice A et les vecteurs B et C sont définis respectivement dans ce que suit :

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} Q^{C}_{MNO,1} \\ Q^{C}_{MNO,2} \\ \vdots \\ Q^{C}_{MNO,M} \end{pmatrix} = \begin{pmatrix} \varsigma^{A}_{MNO,1} - \frac{1}{\sqrt{\Pi_{MNO,1}}} \\ \varsigma^{A}_{MNO,2} - \frac{1}{\sqrt{\Pi_{MNO,2}}} \\ \vdots \\ \varsigma^{A}_{MNO,M} - \frac{1}{\sqrt{\Pi_{MNO,M}}} \end{pmatrix} \tag{5.22}$$

La quantité stockée dans l'EN de chaque suiveur i peut être obtenue en appliquant la règle de *Cramer* comme il était proposé dans le travail [40] :

$$Q^{C}_{MNO,i}{}^{NE} = \frac{det\,(A_m)}{det\,(A)} \tag{5.23}$$

Tel que *det(A)* est le déterminent de la matrice A et $det(A_m)$ est le déterminent de la matrice A_m qui est formée en replaçant la $m^{i\grave{e}me}$ colonne de A par la colonne de vecteur C.

$$Q_{MNO,i}^{C}{}^{NE} = \frac{\varsigma_{MNO,i}^{A} - \frac{1}{\sqrt{\Pi_{MNO,i}}}}{1} = \varsigma_{MNO,i}^{A} - \frac{1}{\sqrt{\Pi_{MNO,i}}} \tag{5.24}$$

5.5.2 Stratégie optimale du leader

Compte tenu des meilleures stratégies des suiveurs, la stratégie optimale du leader (le seuil de popularité optimal et la quantité de contenu populaire demandée pour la mise en cache) peut être dérivée en résolvant le problème :

$$\begin{gathered} \tau^{*} = \arg\max_{Q^{R},\tau} \ \mu_{cp}\left(Q^{R}, \ \tau\right) \\ s.t. \ \ 0 < Q^{R} \leq F \\ \tau > 0 \end{gathered} \tag{5.25}$$

Tel que μ_{cp} est défini dans l'équation (5.5).

Pareil comme dans la sous-section précédente, nous utilisons la méthode de l'optimisation convexe pour résoudre le problème d'optimisation du leader dans l'équation (5.25). Nous calculons les dérivées partielles du premier et de second ordre de la fonction d'utilité du leader en fixant Q^{R} avec le respect de τ. Ici, nous définissons le taux de satisfaction du leader noté $\mathcal{S}_i$, en tant qu'une fonction croissante de la popularité et de la quantité des fichiers stockés en cache, tel que $\mathcal{S}_i$ est donnée par :

$$\mathcal{S}_i = (\sigma_{MNO,i} - \tau) \times Q_{MNO,i}^{C}{}^{NE} \tag{5.26}$$

En remplaçant la valeur de $\mathcal{S}_i$ dans l'équation (5.5), nous obtenons :

$$\mu_{cp}\left(\mathcal{S}_i, \tau\right) = \sum_{i=1}^{M} \left(\frac{log(1+\mathcal{S}_i)}{M} - \left(\Pi_{MNO,i} \times \frac{\mathcal{S}_i}{(\sigma_{MNO,i} - \tau)} \right) \right) \tag{5.27}$$

Il est clair que $\mu_{cp}(Q^{R}, \tau)$ est une fonction concave par rapport au taux de satisfaction $\mathcal{S}_i$ puisque la seconde dérivée est négative.

$$\frac{d\mu_{cp}\left(\mathcal{S}_i, \tau\right)}{d\mathcal{S}_i} = \sum_{i=1}^{M} \left(\frac{1}{M \times (\mathcal{S}_i + 1)} - \frac{\Pi_{MNO,i}}{(\sigma_{MNO,i} - \tau)} \right) \tag{5.28}$$

$$\frac{d^{2}\mu_{cp}\left(\mathcal{S}_i, \tau\right)}{d^{2}\mathcal{S}_i} = \sum_{i=1}^{M} -\frac{1}{M \times (\mathcal{S}_i + 1)^{2}} < 0 \tag{5.29}$$

Donc, le taux de satisfaction optimal du leader $\mathcal{S}_i^{*}$ peut être obtenu en résolvant :

$$\frac{d\mu_{cp}\left(\mathcal{S}_i, \tau\right)}{d\mathcal{S}_i} = \sum_{i=1}^{M} \left(\frac{1}{M \times (\mathcal{S}_i + 1)} - \frac{\Pi_{MNO,i}}{(\sigma_{MNO,i} - \tau)} \right) = 0 \tag{5.30}$$

$$\mathcal{S}_i^{*} = \frac{(\sigma_{MNO,i} - \tau)}{\Pi_{MNO,i} \times M} - 1 \tag{5.31}$$

En plus, le seuil de popularité optimal (τ^{*}) de leader peut être dérivé en résolvant :

$$\mathcal{S}_i^* = \frac{\sigma_{MNO,i} - \tau}{\Pi_{MNO,i} \times M} - 1 = (\sigma_{MNO,i} - \tau) \times {Q_{MNO,i}^C}^{NE} \tag{5.32}$$

$$\tau_i^* = \frac{M \times \Pi_{MNO,i} - \sigma_{MNO,i} + M \times \sigma_{MNO,i} \times \Pi_{MNO,i} \times {Q_{MNO,i}^C}^{NE}}{M \times \Pi_{MNO,i} \times {Q_{MNO,i}^C}^{NE} - 1} \tag{5.33}$$

Et donc

$$\tau^* = \sum_{i=1}^{M} \left(\frac{(M \times \Pi_{MNO,i}) \times \left(1 + \left(\sigma_{MNO,i} \times {Q_{MNO,i}^C}^{NE}\right)\right) - \sigma_{MNO,i}}{M \times \Pi_{MNO,i} \times {Q_{MNO,i}^C}^{NE} - 1} \right) \tag{5.34}$$

Une fois le seuil de popularité optimal est obtenu τ^*, le leader peut choisir tout simplement la quantité ou le nombre effective de son contenu populaire qu'il souhaite mettre en cache (Q^{R^*}) avec une popularité supérieure à τ^* à partir de la bibliothèque $\mathcal{L}$. Q^{R^*} est une variable discrète exprimée via un entier positif qui peut être facilement obtenue par :

$$Q^{R^*} = \{c_i \in \mathcal{L}, i = 1, \ldots, \tau^* / i \in F \;\; et \;\; 0 < \tau^* \leq F\} = \{c_1, c_2, \ldots, c_{\tau^*-1}, c_{\tau^*}\} \tag{5.35}$$

5.6 Résultats numériques

Dans cette section, nous utilisons Matlab pour évaluer les performances de notre jeu de type *Stackelberg* pour l'incitation à la mise en cache dans les réseaux d'IoVs basés sur SDN. Sans perte de généralité, nous simulons dans la série des expérimentations suivantes, un CP représenté par le contrôleur SDN global et six MNOs représentés par un ensemble de contrôleurs SDN centraux. Le CP souhaite mettre en cache ses fichiers populaires d'une taille entre 100 et 150 giga-octets pour des contenus multimédias de haute qualité. Chaque MNO a un ensemble de 20 SBSs de même taille avec une capacité de stockage totale égale à 1,5 Go. Nous considérons une bibliothèque de contenu avec un ensemble de 1000 fichiers populaires. La popularité des fichiers suit une distribution *Zipf* avec le paramètre $\delta = 0.1$. β représente la quantité de cache occupée. Nous considérons des véhicules hétérogènes assignés à différents MNOs. Les principaux paramètres de simulation sont résumés dans le tableau 5.1.

Paramètres	**Valeur**
$\#CP$	1
$\#MNO$	6
$\#SBSs$	20
sc	1.5 Gbps
F	1000
Q^R	100 - 150 Gbps
δ	0.1

TABLE 5.1 – Paramètres de simulation.

Dans l'expérimentation illustrée à la figure 5.2 et à la figure 5.3, nous étudions l'impact du nombre des suiveurs (MNOs) sur l'utilité de la mise en cache des joueurs (CP et MNOs) et leurs décisions optimales (la quantité stockée en cache par l'opérateur MNO ($Q^C_{MNO,i}$) et le seuil de popularité de la mise en cache (τ)) dans l'ES, respectivement.

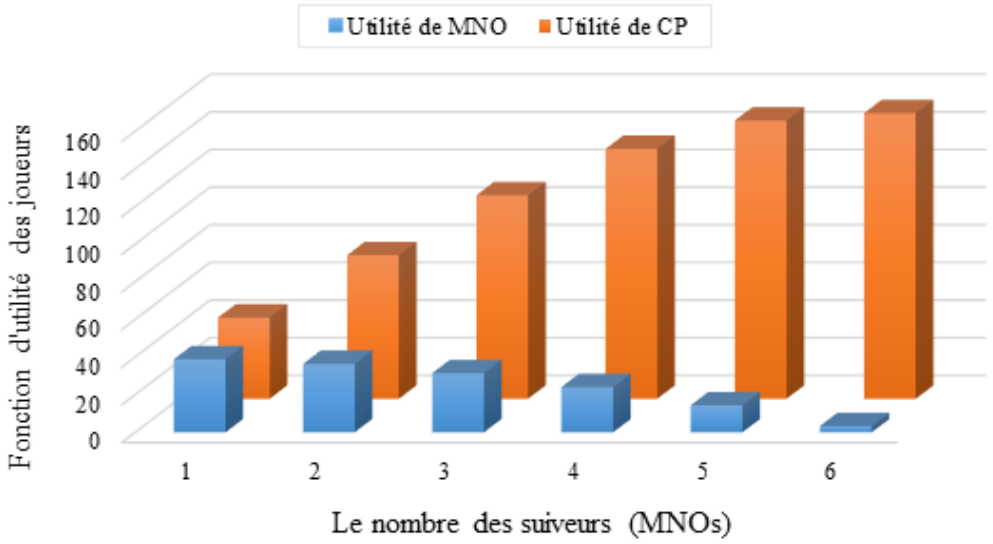


FIGURE 5.2 – Impact du nombre des suiveurs (MNOs) sur l'utilité des joueurs.

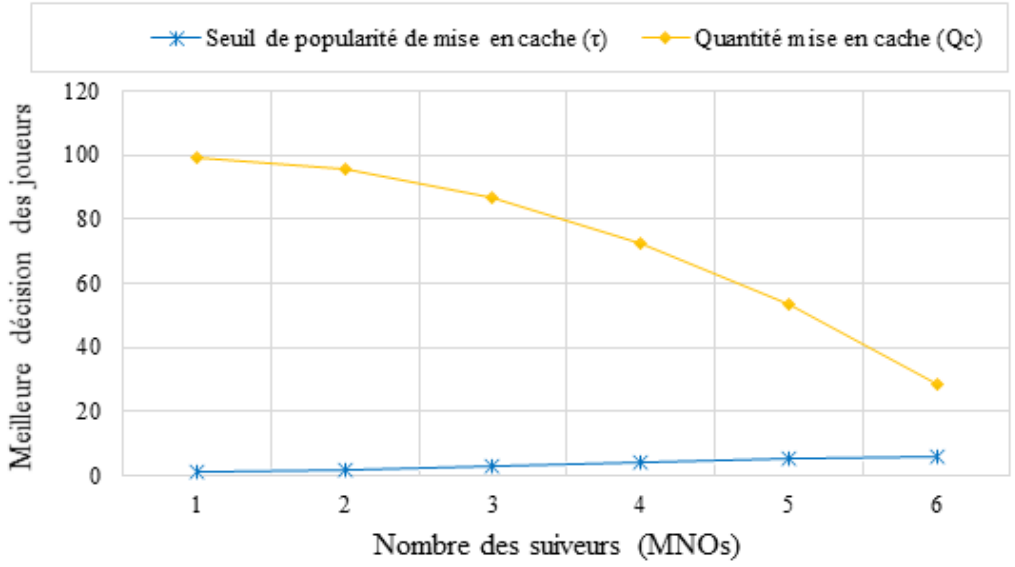


FIGURE 5.3 – Impact du nombre des suiveurs (MNOs) sur la décision optimale des joueurs.

Les résultats dans les figures 5.2 et 5.3 montrent qu'avec l'augmentation du nombre des suiveurs (MNOs) : *(i)* le seuil de popularité du leader (CP) augmente, *(ii)* la quantité optimale de contenu stockée en cache d'un suiveur (MNO) diminue, et *(iii)* l'utilité du leader augmente alors que celle des suiveurs diminue. Cela peut être justifié par le fait que lorsque le nombre des MNOs augmente, l'offre en termes de l'espace de stockage disponible pour la mise en cache sera plus grande et éventuellement les probabilités d'accès (*i.e.*, la popularité) aux contenus populaires du CP peuvent être plus élevées. Ainsi, le

leader peut annoncer une plus grande quantité de ses contenus tops populaires à mettre en cache avec un seuil de popularité (τ) plus élevé. En stockant les contenus les plus populaires à la périphérie du réseau, le leader peut améliorer la QoE et la satisfaction de ses utilisateurs en diminuant le délai d'accès/téléchargement par exemple. Par conséquent, le gain du CP va augmenter et son coût de mise en cache va diminuer en bénéficiant de la diminution du prix de la mise en cache, résultat de la grande disponibilité de l'espace de stockage en cache offert par les différents MNOs, référez-vous aux équations (5.2), (5.4) et (5.7), respectivement. Tous les arguments cités auparavant vont contribuer à l'amélioration de l'utilité du leader, voir l'équation (5.5). En autre, l'augmentation du nombre des MNOs va créer plus de concurrence entre les suiveurs sur la quantité limitée de contenu populaire du CP annoncé pour la mise en cache. Par conséquent, les contrôleurs SDN centraux des différents MNOs seront obligés en quelque sorte de partager la quantité de contenu du CP et donc une plus petite quantité sera potentiellement disponible pour être stockée par chaque MNO i. Comme résultat, les suiveurs vont générer de plus petits revenus à partir de l'opération de la mise en cache, référez-vous à l'équation (5.6). En plus, le gain de suiveur peut être aussi affecté par la baisse de la rémunération en termes de prix de mise en cache, principalement affecté par la concurrence entre les MNOs et la grande disponibilité de l'espace de stockage en cache, référez-vous à l'équation (5.7). Tous ces facteurs vont contribuer à la détérioration de l'utilité du suiveur, voir l'équation (5.11).

Dans la figure 5.4, nous considérons un seul suiveur (MNO), et nous étudions l'impact de la variation de la quantité de contenu populaire de CP stockée en cache ($Q^C_{MNO,i}$) sur l'utilité du suiveur (MNO) pour différents prix de mise en cache ($\Pi_{MNO,i}$).

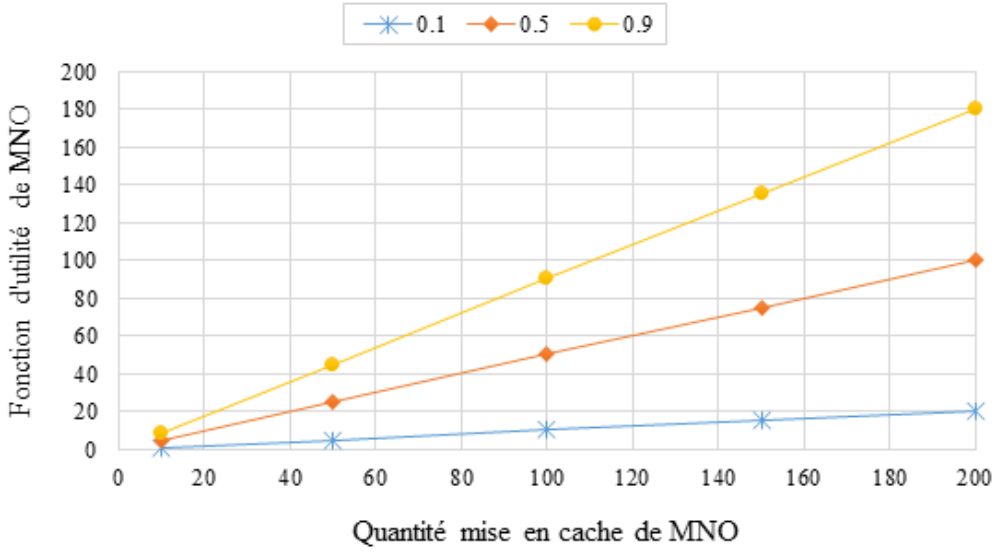


FIGURE 5.4 – Impact de la quantité mise en cache ($Q^C_{MNO,i}$) sur la fonction d'utilité de suiveur (MNO) pour différents prix de mise en cache ($\Pi_{MNO,i}$).

À partir des résultats illustrés à la figure 5.4, nous pouvons clairement constater que l'utilité du suiveur (MNO) croît avec l'augmentation de la quantité de contenu populaire stockée en cache et une utilité supérieure correspondent à un prix de mise en cache plus élevé. Cela parce que, lorsque la quantité du contenu stockée en cache augmente, le suiveur générera plus de revenu monétaire qui contribue à l'amélioration de sa récompense,

référez-vous à l'équation (5.6). En plus, lorsque la quantité stockée en cache augmente, le suiveur va utiliser davantage de mémoire pour stocker le contenu populaire et par conséquent moins d'espace restera pour stocker de nouveaux contenus. Cela, va augmenter le coût de mise en cache en termes de capacité de stockage du suiveur, référez-vous à l'équation (5.8). Dans notre politique de mise en cache, le suiveur pourra compenser ce manque à gagner (le coût de mise en cache élevé) en facturant un prix de mise en cache relativement plus élevé pour les nouvelles demandes de mise en cache, référez-vous à l'équation (5.7). Tous les facteurs cités auparavant vont contribuer à l'amélioration de l'utilité du suiveur, voir l'équation (5.11). Aussi, il est logique que lorsque le prix de la mise en cache augmente, le gain monétaire du suiveur augmente pour la même quantité stockée et son utilité sera ainsi améliorée, référez-vous aux équations (5.6) et (5.11), respectivement.

Dans l'expérimentation illustrée à la figure 5.5, nous étudions l'impact du seuil de popularité de la mise en cache (τ) sur l'utilité du leader (CP) pour différentes quantités des contenus populaires du CP annoncées pour la mise en cache (Q^R).

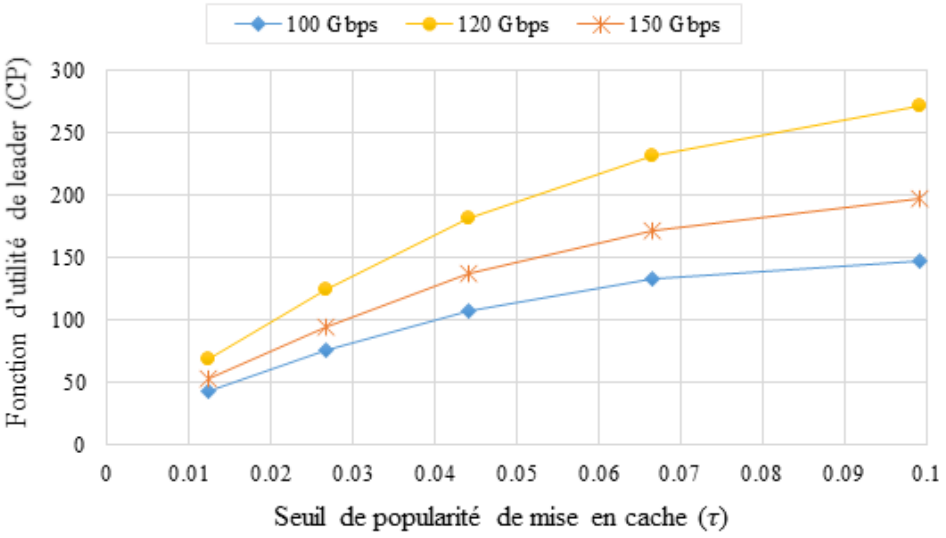


FIGURE 5.5 – Impact de seuil de popularité (τ) sur la fonction d'utilité du leader (CP).

Les résultats à la figure 5.5 montrent que, au fur et à mesure que le seuil de popularité de la mise en cache augmente, l'utilité du leader (CP) augmente et elle admet toujours une valeur globale optimale pour différentes quantités de contenus annoncés pour la mise en cache. Ce résultat confirme nos résultats analytiques dans la sous-section 5.5.2 et prouve l'existence de l'ES. Aussi, l'utilité de leader est nettement améliorée lorsque la quantité du contenu populaire du CP annoncée pour la mise en cache augmente. Cela peut être justifié par le fait que le stockage en cache d'une plus grande quantité de contenus de CP avec un seuil de popularité élevé, permet d'améliorer davantage la satisfaction du leader. De plus, lorsque le CP annonce une plus grande quantité de son contenu populaire pour la mise en cache, les MNOs pourront stocker plus de contenu populaire sur leurs caches à la périphérie du réseau (l'offre de l'espace de stockage en cache des MNOs est plus grande que la demande de contenu populaire du CP pour la mise en cache), ce qui diminuera les délais d'accès/téléchargement et donc améliora la satisfaction et la récompense de leader, référez-vous à l'équation (5.2). En outre, la concurrence entre les MNOs sur cette grande

quantité de contenu populaire du CP annoncée pour la mise en cache peuvent entraîner une diminution du prix de mise en cache et donc une baisse du coût de mise en cache pour le leader, référez-vous à l'équation (5.4). Tous ces éléments vont contribuer à l'amélioration de l'utilité du leader, voir l'équation (5.5). En plus, nous remarquons que les résultats à la figure 5.5 montrent que l'utilité du leader augmente de manière significative pour les petites valeurs du seuil de la popularité de la mise en cache τ après elle augmente plus lentement lorsque τ atteint des hauts niveaux de popularité qui sont facturés par un prix de mise en cache plus élevé. Cela va causer une augmentation du coût de mise en cache chargé à être payé par le leader. La raison qui explique le ralentissement de la croissance de l'utilité du leader.

Dans l'expérimentation à la figure 5.6 et à la figure 5.7, nous étudions l'impact de la variation du paramètre de la loi *Zipf* (δ) qui caractérise le niveau de popularité du contenu sur l'utilité du leader (CP) et celle de suiveur (MNO), respectivement.

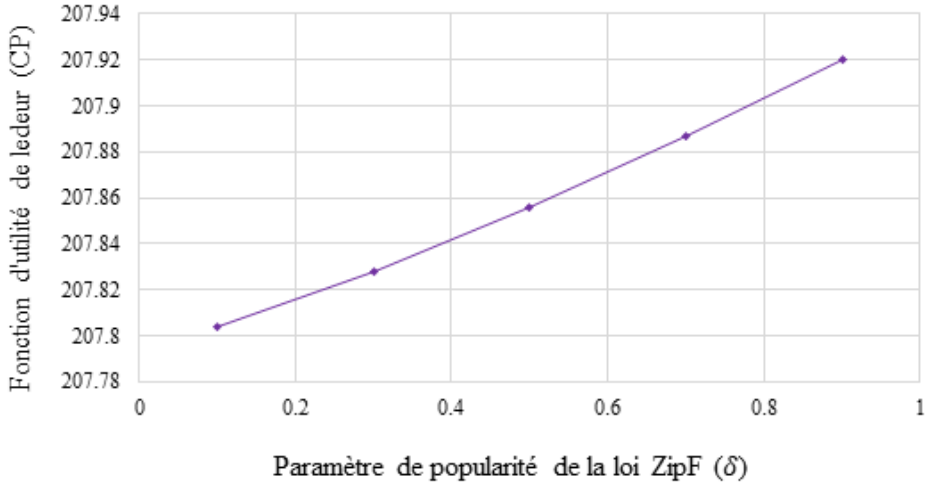


FIGURE 5.6 – Impact de paramètre de popularité (δ) sur l'utilité de leader (CP).

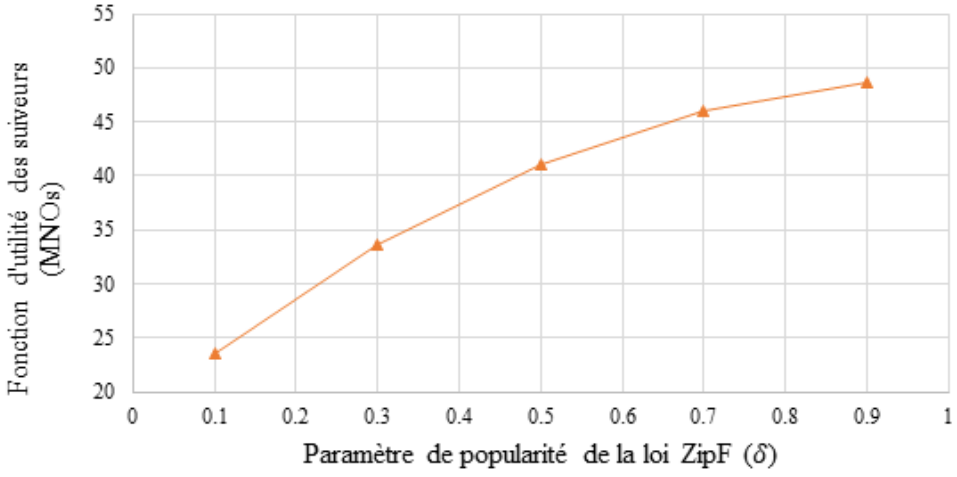


FIGURE 5.7 – Impact de paramètre de popularité (δ) sur l'utilité des suiveurs (MNOs).

À partir des résultats illustrés à la figure 5.6 et à la figure 5.7, nous pouvons clairement remarquer que l'utilité du leader (CP) et celle du suiveur (MNO) augmentent avec l'augmentation du paramètre de la loi de *Zipf* (δ). Cela est principalement dû au fait qu'une valeur plus élevée de δ signifie qu'une plus grande partie de la quantité de contenu du CP annoncée pour la mise en cache correspond aux fichiers les plus populaires. Autrement dit, des fichiers les tops populaires vont être mis en cache, ce qui va augmenter la satisfaction de leader et contribuer ainsi à l'amélioration de son utilité, référez-vous à l'équation (5.5). De plus, lorsque le paramètre δ augmente, la popularité de contenu du CP qui peut être mis en cache par un suiveur augmente. Ainsi, le MNO va donc facturer un prix plus élevé pour stocker dans son cache des contenus aussi populaires afin de compenser l'énergie supplémentaire nécessaire pour gérer l'augmentation de la charge du trafic et les interférences. La raison qui va augmenter le revenu monétaire du suiveur et contribue à l'amélioration de son utilité, référez-vous aux équations (5.6) et (5.11).

Dans l'expérimentation illustrée à la figure 5.8 et à la figure 5.9, nous étudions l'impact de la variation du prix de mise en cache (Π) sur l'utilité des joueurs (CP et MNOs). Ensuite, nous comparons l'utilité des joueurs dans notre politique d'incitation à la mise en cache basée sur SDN avec des propriétaires des caches compétitives (plusieurs MNOs), nous le notons *mise en cache compétitive*, avec un travail de référence de Shen et al. [40] avec un modèle de l'incitation à la mise en cache monopolistique avec un MNO comme le seul gestionnaire des caches, nous le notons *mise en cache monopolistique*, en fonction du prix de mise en cache. Les résultats de la comparaison peuvent être facilement généralisés à tous les travaux similaires avec une politique de mise en cache monopolistique avec un seul propriétaire des caches.

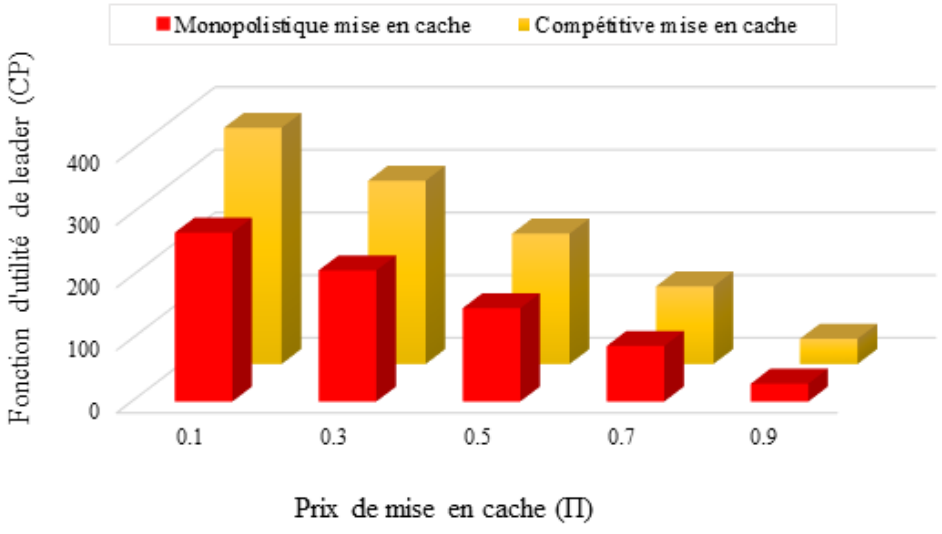


FIGURE 5.8 – Impact du prix de mise en cache (Π) sur la fonction d'utilité de leader (CP).

Les résultats illustrés à la figures 5.8 et à la figure 5.9 montrent que, lorsque le prix de mise en cache augmente, l'utilité du leader (CP) augmente et celle de suiveur (MNO) diminue à la fois dans notre travail de mise en cache compétitive et le travail de mise en cache monopolistique. La justification est due au fait que lorsque le prix de mise en

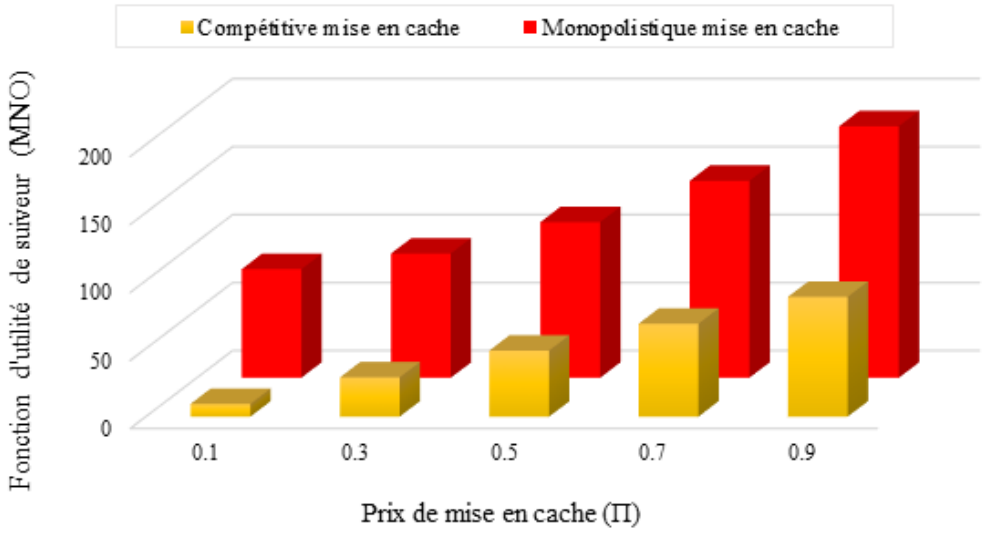


FIGURE 5.9 – Impact du prix de mise en cache (Π) sur la fonction d'utilité de suiveur (MNO).

cache augmente, le leader doit payer un coût de mise en cache plus élevé ce qui cause la dégradation de son utilité. Au contraire, le suiveur encaissera un gain monétaire plus élevé et expérimentera donc une meilleure utilité. Aussi, l'utilité de suiveur dans notre système de mise en cache compétitif est plus petite que celle du travail de mise en cache monopolistique. Cela parce que dans le scénario de la mise en cache compétitive, les suiveurs (MNOs) devraient d'une manière ou d'une autre partager la quantité disponible du contenu populaire annoncée à la mise en cache par le CP et donc partager le gain potentiel de la mise en cache. Cela entraînera la détérioration de l'utilité de suiveur. Dans le travail de la mise en cache monopolistique, un seul MNO monopolise et stocke à lui seul toute la quantité du contenu annoncée par le CP en récoltant tous les revenus de mise en cache et donc peut bénéficier d'une meilleure utilité. De plus, l'utilité du leader dans notre travail est plus élevée que celle dans le travail de mise en cache monopolistique. Cela est dû au fait que, dans notre scénario de mise en cache compétitive, l'espace de mise en cache disponible est plus grand (plusieurs opérateurs offrent le service de stockage), ce qui peut amener à un prix de mise en cache plus petit à cause de la grande offre de l'espace de stockage, référez-vous à l'équation (5.7). Cela va encourager le leader (CP) à annoncer une plus grande quantité de son contenu populaire pour la mise en cache afin d'améliorer la satisfaction de ses utilisateurs, ce qui va contribuer à l'amélioration de son utilité. En outre, nous pouvons clairement voir que notre stratégie de mise en cache compétitive avec plusieurs MNOs peut : *(i)* améliorer l'utilité du leader de 30% et *(ii)* causer jusqu'à 48% de détérioration de l'utilité du suiveur, par rapport au travail de mise en cache monopolistique. Cela est justifié principalement par l'augmentation de l'offre de service de mise en cache, ce qui va augmenter l'espace de stockage disponible et ainsi peut entraîner une amélioration du gain de leader, profitant de la diminution des prix. Aussi, la concurrence entre les suiveurs se traduit par une baisse de leurs revenus à cause de la compétition sur la quantité limitée de contenu du CP, ce qui résulte dans de plus petites quantités de contenu possible pour la mise en cache pour chaque suiveur.

En guise de conclusion finale à partir des résultats précédents, nous pouvons en déduire que la prise en compte de plusieurs opérateurs de mise en cache (les MNOs dans notre cas d'étude) dans l'étude des systèmes de l'incitation à la mise en cache dans les scénarios IoVs est important, en raison de l'impact significatif qu'elle représente sur les utilités des acteurs de mise en cache. Effectivement, d'un point de vue économique, créer la concurrence entre les acteurs de la mise cache peut considérablement améliorer la satisfaction des utilisateurs avec une meilleure QoE, tout en réduisant le coût de mise en cache. Une conséquence directe du brisement de l'aspect monopole sur le marché de mise en cache.

5.7 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle politique pour l'incitation à la mise en cache dans le contexte d'un réseau d'IoV basée sur le paradigme de SDN. Le modèle d'incitation à la mise en cache proposé considère un marché concurrentiel de mise en cache avec un CP qui souhaite mettre en cache son contenu populaire et plusieurs MNOs, chacun gère un ensemble de caches sur ses SBSs. Les MNOs sont en concurrence permanente pour le stockage de la quantité limitée du contenu populaire de CP. Le principal objectif est de motiver les MNOs et le CP à collaborer pour améliorer le processus de mise en cache. Nous avons discuté la politique de mise en cache d'un point de vue économique et nous avons modélisé l'interaction entre le CP et les MNOs sous forme d'un jeu de type *Stackelberg* avec un sous-jeu non-coopératif. De plus, nous avons utilisé l'optimisation convexe pour étudier et prouver l'équilibre de *Stackelberg* ainsi que pour résoudre le problème d'optimisation et dériver les paramètres optimaux de l'opération de la mise en cache.

Nous avons également évalué notre jeu pour l'incitation à la mise en cache sous des simulations intensives et nous avons comparé ses performances à un travail de référence d'incitation à la mise en cache monopolistique. Les résultats numériques ont montré que la prise en compte d'un marché de mise en cache compétitif avec plusieurs gestionnaires des caches dans l'étude des scénarios de l'incitation à la mise en cache, peut améliorer significativement la QoE des utilisateurs et contribuer à la diminution du coût de l'opération de mise en cache pour le fournisseur de service.

Comme perspectives futures à ce travail, nous envisageons généraliser notre jeu de type *Stackelberg* pour prendre en charge plusieurs CPs et plusieurs MNOs en utilisant un jeu multi-leaders multi-suiveurs. En outre, il semble intéressant d'exploiter les contrôleurs SDN locaux sur les véhicules chefs de groupes en tant que des caches mobiles pour décharger les SBSs, améliorer le délai de bout en bout et bénéficier des petites latences des communications V2V.

CHAPITRE 6

CONCLUSION ET PERSPECTIVES

6.1 Conclusion

Les réseaux ad hoc de véhicules (VANETs) ont émergé comme une nouvelle technologie pour améliorer la sécurité sur les routes, les conditions de trafic et le confort de voyage. De nos jours, les architectures VANETs souffrent de problèmes d'évolutivité vu qu'il est très difficile de déployer des services à grande échelle. Ces architectures sont rigides, difficiles à gérer et souffrent d'un manque de flexibilité et d'adaptabilité au contrôle. Récemment, plusieurs travaux ont montré que l'intégration de SDN dans les réseaux de véhicules peut apporter de la flexibilité, de la programmabilité et permet de mieux supporter l'évolutivité.

Dans ce contexte, nous nous sommes intéressés, dans cette thèse, à l'étude du problème de l'intégration efficace du paradigme de SDN pour améliorer les performances des prochaines générations des réseaux de véhicules. Nous avons donc commencé par proposer une nouvelle architecture semi-centralisée qui tire profit des avantages de différents concepts de réseaux à l'instar des réseaux cellulaires, le concept de *Cloud/fog computing* et la technique de *clustering* pour faciliter une intégration efficace du paradigme de SDN dans les réseaux de véhicules hétérogènes, appelée *Cluster-based Software-Defined Heterogeneous Vehicular Networks (CSDHVN)*. Le mode de contrôle dans notre architecture est semi-centralisé avec une hiérarchie de trois types de contrôleurs SDN déployés à la périphérie du réseau pour mieux supporter les exigences des services de sécurité sensibles aux délais. Notre architecture est agile et fonctionne aussi bien dans les zones couvertes par l'infrastructure fixe véhiculaires, que dans les zones non couvertes grâce à des contrôleurs SDN mobiles déployés sur certains véhicules mobiles. Comme mécanisme de récupération de secours en cas de panne d'un des contrôleurs SDN, nous avons procédé par l'anticipation et la pré-préparation du scénario de secours tout en exploitant intelligemment la hiérarchie des contrôleurs SDN et l'auto-organisation des véhicules grâce à la technique de *clustering*. Les résultats numériques ont démontré la faisabilité et l'efficacité de l'architecture proposée. Ils ont prouvé aussi la fiabilité de notre mécanisme de récupération de secours et montré l'important effet négatif que peut représenter la distance lointaine du déploiement de contrôleur SDN sur les performances du système.

Nous nous sommes focalisés, ensuite, sur la version distribuée multi-sauts de CSDHVN pour étudier un cas d'utilisation de sécurité routière relatif à une mission de secours suite à un accident sur la route dans les environnements véhiculaires sans infrastructure. Nous avons équipé les véhicules de secours par des mini-drones afin de les assister dans l'exploration des tronçons de route inaccessibles et nous avons défini une nouvelle architecture basée sur le paradigme de SDN pour faciliter la gestion à distance des drones. Puis, nous avons concentré nos efforts sur l'étude de la manière d'assurer un traitement efficace des données collectées par les drones. Nous avons formulé le problème à l'aide d'un jeu séquentiel à deux joueurs comme un problème de prise de décision en matière de délestage (*offloading*) de calcul des tâches entre le drone et le véhicule de secours et/ou le partage de calcul des tâches entre le véhicule de secours et ses voisins. Nous avons proposé deux algorithmes distribués pour résoudre le problème d'optimisation qui consiste à trouver le meilleur équilibre entre le délai et l'énergie de calcul. Nous avons utilisé SDN pour faciliter la gestion à distance des drones ainsi que pour prendre de meilleures décisions mieux informées en termes de délestage/partage de calcul. Les résultats numériques montrent que la politique de traitement des données basée sur SDN assure des performances de calcul efficace avec le meilleur équilibre entre le délai et l'énergie de calcul.

Enfin, nous avons projeté l'architecture CSDHVN sur un deuxième cas d'utilisation concernant l'incitation à la mise en cache dans les réseaux d'Internet de véhicules. Nous avons considéré un scénario de mise en cache concurrentiel composé d'un seul CP et plusieurs MNOs. Le CP encourage les MNOs à stocker son contenu populaire dans leurs caches sur les SBSs avec la plus grande popularité. Les MNOs sont en concurrence permanente pour le stockage de la quantité limitée des fichiers populaires de CP. Nous avons utilisé un jeu de type *Stackelberg* pour modéliser l'interaction entre le CP et les MNOs avec un sous-jeu non-coopératif pour formuler le conflit entre les MNOs. Nous avons utilisé la hiérarchie des contrôleurs SDN pour gérer l'interaction entre le CP et les MNOs ainsi que pour estimer la popularité du contenu et améliorer les décisions de mise en cache. Nous avons ensuite utilisé l'optimisation convexe pour étudier deux problèmes d'optimisation lorsque le CP vise à maximiser la satisfaction de ses utilisateurs en termes de délai d'accès/téléchargement, quand chaque MNO vise à maximiser son gain monétaire. Les résultats numériques ont montré que la prise en compte de plusieurs propriétaires des caches dans l'étude des scénarios de la mise en cache peut améliorer de manière significative la qualité d'expériences des utilisateurs et contribuer à la diminution du coût de la mise en cache.

En guise de conclusion, sur la base de la discussion de la taxonomie des architectures des réseaux de véhicules basées sur SDN dans le chapitre 2, nous pensons que la tendance dans la conception des futures architectures basées sur SDN pour les réseaux de véhicules, devrait être l'utilisation de plusieurs contrôleurs SDN distribués installés à la périphérie du réseau avec un mécanisme de récupération de secours efficace. De plus, les zones sans infrastructure non couvertes des réseaux de véhicules doivent être prises en considération lors de la conception des futures architectures des réseaux de véhicules basées sur le paradigme de SDN. Le coût et la complexité de déploiement restent les grands défis qui s'opposent à une intégration rapide de SDN dans les réseaux de véhicules. De plus, comme

nous assistons de nos jours à les premières étapes de déploiement des réseaux de véhicules et vu les avantages futuristes qu'offrent le paradigme de SDN, nous croyons profondément que SDN constitue le futur des architectures des réseaux en général et celui des réseaux de véhicules en particulier surtout avec l'arrivée très attendue de la 5G dans les VANETs et le rapide développement du véhicule autonome. En conséquence, il nous semble qu'il est grand temps de se mettre à développer des solutions véhiculaires basées sur SDN avant la prolifération des réseaux de véhicules afin de limiter les coûts de déploiement. Par ailleurs, les progrès techniques et scientifiques ainsi que les résultats très motivants des travaux de l'intégration de SDN dans les VANETs, le paradigme SDN a besoin de plus du temps pour se mûrir et être standardisé surtout de côté des menaces de sécurité.

6.2 Perspectives

Les travaux présentés dans cette thèse peuvent être étendus pour accomplir d'autres objectifs. Comme perspectives futures à court terme de ce travail, nous envisageons de :

- Généraliser notre modèle de délestage/partage de calcul pour supporter le partage de calcul avec plusieurs drones voisins et le partage de calcul avec différents contrôleurs SDN sur les véhicules contrôleurs à proximité. Il serait également intéressant d'ajouter une stratégie pour le délestage et le partage de calcul avec un serveur dans le Cloud une fois que la couverture de l'infrastructure est de nouveau disponible. De plus, le système peut être encore amélioré avec un ajustement dynamique des poids de pondération des métriques de la fonction d'utilité (délai et énergie) selon le type des applications : sensibles ou non au délai et/ou à la bande passante. Comme conséquence, le jeu séquentiel pourrait être enrichi pour supporter un cas d'étude plus général faisant participer N joueurs avec de nouvelles stratégies.
- Généraliser notre politique de l'incitation à la mise en cache pour prendre en compte plusieurs CPs qui souhaitent mettre en cache leurs contenus populaires et plusieurs MNOs qui offrent le service de mise en cache sur leurs stations de base. Ainsi, utiliser un jeu de type *Stackelberg* multi-leaders multi-suiveurs pour modéliser l'interaction entre les différents joueurs. En outre, il semble intéressant d'exploiter les contrôleurs SDN locaux sur les véhicules chefs de groupe en tant que caches mobiles pour décharger les stations de base, améliorer le délai de bout en bout et bénéficier de plus petits temps de communication entre les véhicules. Un mécanisme d'incitation peut être aussi utilisé par la station de base afin d'encourager les contrôleurs SDN pour stocker et partager leurs contenus populaires avec d'autres utilisateurs à proximité.

Concernant les perspectives à moyen et long termes de ce travail, nous prévoyons de :

- Exploiter le paradigme de SDN pour améliorer les décisions d'une nouvelle politique de mise en cache. Dans cette politique, les contrôleurs SDN centraux sur les stations de base peuvent former des coalitions et entrer en compétition les unes contre les autres. Chaque coalition défend un intérêt commun qui converge vers le profit de son opérateur mobile. Un jeu coopératif sera utilisé pour modéliser l'interaction entre les joueurs.
- Sachant que l'une des plus grandes faiblesses qui s'oppose à une rapide adoption du paradigme SDN est le problème de la sécurité lié à son mode de fonctionnement centralisé, nous envisageons de le prendre en compte afin d'améliorer la fiabilité de

notre architecture semi-centralisée CSDHVN et de protéger le système contre les menaces de sécurité telles que l'attaque de déni de service.

BIBLIOGRAPHIE

[1] A. Zekri and W. Jia. Heterogeneous vehicular communications : A comprehensive study. *Ad Hoc Networks*, 75–76( ) :52–79, 2018.

[2] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou. A survey on architecture challenges, and solutions. *IEEE Communications Surveys and Tutorials*, 17(4) :2377–2396, 2015.

[3] J. Chen, Z. Haibo, W. Xu, Q. Yu, L. Gui, and X. Shen. Service-oriented dynamic connection management for software-defined internet of vehicles. *IEEE Trans Intell Transp Syst*, 18(10) :2826–2837, 2017.

[4] F. Yang, S. Wang, J. Li, Z. Liu, and Q. Sun. An overview of internet of vehicles. *China Communications*, 11(10) :1–15, 2014.

[5] R. Amin, M. Reisslein, and N. Shah. Hybrid sdn networks : A survey of existing approaches. *IEEE Communications Surveys and Tutorials*, 20(4) :3259–3306, 2018.

[6] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira. Towards software-defined vanet : Architecture and services. In *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, Piran, 2014.

[7] Y. C. Chang, , J. L. Chen, Y. W. Ma, and P. S. Chiu. Vehicular cloud serving systems with software-defined networking. In *IOV 2015 (Springer LNCS)*, 2015.

[8] B. Baron, P. Spathis, H. Rivano, M. D. de Amorim, Y. Viniotis, and J. Clarke. Software-defined vehicular backhaul. In *2014 IFIP Wireless Days (WD)*, Rio de Janeiro, Brazil, 2014.

[9] Y. Cao, J. Guo, and Y. Wu. Sdn enabled content distribution in vehicular networks. In *Fourth edition of the International Conference on the Innovative Computing Technology (INTECH 2014)*, Luton, 2014.

[10] H. Li, M. Dong, and K. Ota. Control plane optimization in software-defined vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 65(10) :7895–7904, 2016.

[11] W. Huang, L. Ding, D. Meng, J. Hwang, Y. Xu, and W. Zhang. Qoe-based resource allocation for heterogeneous multi-radio communication in software-defined vehicle networks. *IEEE Access*, 6 :3387–3399, 2018.

[12] K. L. K. Sudheer, M. Ma, and P. H. J. Chong. Link dynamics based packet routing framework for software defined vehicular networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Singapore, 2017.

[13] K. N. Kalokhe, Y. Park, and S. Y. Chang. Resilient sdn-based communication in vehicular network. In *International Conference on Wireless Algorithms, Systems, and Applications*, June 2018.

[14] Y. Liu, C. Chen, and S. Chakraborty. A software defined network architecture for geobroadcast in vanets. In *2015 IEEE International Conference on Communications (ICC)*, London, 2015.

[15] X. Duan, Y. Liu, and X. Wang. Sdn enabled 5g-vanet : Adaptive vehicle clustering and beamformed transmission for aggregated traffic. *IEEE Communications Magazine*, 55(7) :120–127, 2017.

[16] C. Lai, Y. Chang, H. Chao, M. S. Hossain, and A. Ghoneim. A buffer-aware qos streaming approach for sdn-enabled 5g vehicular networks. *IEEE Communications Magazine*, 55(8) :68–73, 2017.

[17] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane. Software defined networking-based vehicular adhoc network with fog computing. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, 2015.

[18] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani. Software-defined networking for rsu clouds in support of the internet of vehicles. *IEEE Internet of Things Journal*, 2(2) :133–144, 2015.

[19] K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei. Soft-defined heterogeneous vehicular network : architecture and challenges. *IEEE Network*, 30(4) :72–80, 2016.

[20] Z. He, J. Cao, and X. Liu. Sdvn : enabling rapid network innovation for heterogeneous vehicular communication. *IEEE Network*, 30(4) :10–15, 2016.

[21] D. Zhang Z. He and J. Liang. Cost-efficient heterogeneous data transmission in software defined vehicular networks. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, New York, 2015.

[22] M. Zhu, Z. P. Cai, M. Xu, and J. N. Cao. Software-defined vehicular networks : Opportunities and challenges. *Energy Science and Applied Technology, CRC Press*, pages 247–251, 2015.

[23] R. Soua, E. Kalogeiton, G. Manzo, J. M. Duarte, M. R. Palattella, A. D. Maio, T. Braun, T. Engel, L. A. Villas, and G. A. Rizzo. Sdn coordination for ccn and fc content dissemination in vanets. *Springer Ad Hoc Networks*, page 221—233, 2017.

[24] K. Liu, J. K. Y. Ng, V. C. S. Lee, S. H. Son, and I. Stojmenovic. Cooperative data scheduling in hybrid vehicular ad hoc networks : Vanet as a software defined network. *IEEE/ACM Transactions on Networking*, 24(3) :1759–1773, 2016.

[25] A. Kazmi, M. A. Khan, and M. U. Akram. Devanet : Decentralized software-defined vanet architecture. In *IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, Berlin, 2016.

[26] K. S. K. Liyanage, M. Ma, and P. H. J. Chong. Controller placement optimization in hierarchical distributed software defined vehicular networks. *Computer Networks*, 135 :226–239, 2018.

[27] A. A. Khan, M. Abolhasan, and W. Ni. 5g next generation vanets using sdn and fog computing framework. In *2018 15th IEEE Annual Consumer Communications and Networking Conference (CCNC)*, Las Vegas, 2018.

[28] Y. F. Huang, H. C. Chen, B. C. He, T. H. Tan, S. C. Huang, and F. Chang. Performance of transmission latency in software defined vehicle networks. In *International Conference on Broadband and Wireless Computing, Communication and Applications*, 2018.

[29] J. Al-Badarneh, Y. Jararweh, M. Al-Ayyoub, R. Fontes, M. Al-Smadi, and C. Rothenberg. Cooperative mobile edge computing system for vanet-based software-defined content delivery. *Computers and Electrical Engineering*, 71 :388–397, 2018.

[30] H. Ghafoor and I. Koo. Cr-sdvn : A cognitive routing protocol for software-defined vehicular networks. *IEEE Sensors Journal*, 18(4) :1761–1772, 2018.

[31] R. D. R. Fontes, C. Campolo, C. E. Rothenberg, and A. Molinaro. From theory to experimental evaluation : Resource management in software-defined vehicular networks. *IEEE Access*, 5 :3069–3076, 2017.

[32] K. L. K. Sudheera, M. Ma, G. G. M. N. Ali, and P. H. J. Chong. Delay efficient software defined networking based architecture for vehicular networks. In *2016 IEEE International Conference on Communication Systems (ICCS)*, Shenzhen, 2016.

[33] X. Ge, Z. Li, and S. Li. 5g software defined vehicular networks. *IEEE Communications Magazine*, 55(7) :87–93, 2017.

[34] S. Correia, A. Boukerche, and R. I. Meneguette. An architecture for hierarchical software-defined vehicular networks. *IEEE Communications Magazine*, 55(7) :80–86, 2017.

[35] O. S. Oubbati, A. Lakas, N. Lagraa, and M. B. Yagoubi. Uvar : An intersection uav-assisted vanet routing protocol. In *2016 IEEE Wireless Communications and Networking Conference*, Doha, 2016.

[36] X. Wang, L. Fu, Y. Zhang, X. Gan, and X. Wang. Vdnet : An infrastructure-less uav-assisted sparse vanet system with vehicle location prediction. *Wireless Communications and Mobile Computing*, 16 :2991–3003, 2016.

[37] P. Shilin, R. Kirichek, A. Paramonov, and A. Koucheryavy. Connectivity of vanet segments using uavs. In *the internet of things, smart spaces, and next generation networks and systems*, 2016.

[38] Y. Zhou, N. Cheng, N. Lu, and X. S. Shen. Multi-uav-aided networks : Aerial-ground cooperative vehicular networking architecture. *IEEE Vehicular Technology Magazine*, 10(4) :36–44, 2015.

[39] J. Li, J. Sun, Y. Qian, F. Shu, M. Xiao, and W. Xiang. A commercial video-caching system for small-cell cellular networks using game theory. *IEEE Access*, 4 :7519–7531, 2016.

[40] F. Shen, K. Hamidouche, E. Bastug, and M. Debbah. A stackelberg game for incentive proactive caching mechanisms in wireless networks. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Washington, 2016.

[41] K. Hamidouche, W. Saa, and M. Debbah. Breaking the economic barrier of caching in cellular networks : incentives and contracts. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Washington, 2016.

[42] FD. Pellegrini, A. Massaro, L. Goratti, and R. El-Azouzi. A pricing scheme for content caching in 5g mobile edge clouds. In *Proceedings of the international Conference on Wireless Networks and Mobile Communications (WINCOM)*, Fez, 2016.

[43] J. Li, H. Chen, Y. Chen, Z. Lin, B. Vucetic, and L. Hanzo. Pricing and resource allocation via game theory for a small-cell video caching system. *IEEE J Sel Areas Communs*, 34(8) :2115–2129, 2016.

[44] K. Zhao, S. Zhang, N. Zhang, Y. Zhou, Y. Zhang, and X. Shen. Incentive mechanism for cached-enabled small cell sharing : A stackelberg game approach. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Singapore, 2017.

[45] T. Liu, J. Li, F. Shu, and Z. Han. Resource trading for a small-cell caching system : A contract-theory based approach. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, 2017.

[46] F. Shen, K. Hamidouche, E. Bastug, and M. Debbah. The business side of wireless edge caching : a game theoretical view. *Submitted to IEEE Commun Mag*, 2017.

[47] A. Alioua, S. M. Senouci, S. Moussaoui, H. Sedjelmaci, and A. Boualouache. Software-defined heterogeneous vehicular networks : Taxonomy and architecture. In *Proceedings of the The Global Information Infrastructure and Networking Symposium (IEEE GIIS)*, Saint Pierre, Reunion Island, France, 25-27 October 2017.

[48] A. Alioua, S. M. Senouci, S. Moussaoui, E. Alemneh, M. A. A. Derradji, and F. Benaziza. A distributed multi-hop clustering algorithm for infrastructure-less vehicular ad-hoc networks. In *Proceedings of the International Conference on ICT for Development for Africa (ICT4DA)*, Bahir Dar, Ethiopia, 25-27 September 2017.

[49] A. Alioua, S. M. Senouci, and S. Moussaoui. dsdivn : a distributed software-defined networking architecture for infrastructure-less vehicular networks. In *Proceedings of the International Conference on Innvations for Community Services (I4CS)*, Darmstadt, Germany, 26-28 June 2017.

[50] A. Alioua, S. M. Senouci, H. Sedjelmaci, M. A. Messous, and S. Moussaoui. Efficient data processing in software-defined uav-assisted vehicular networks : A sequential game approach. *International Journal of Wireless Personal Communications*, 101(1) :2255–2286, 2018.

[51] A. Alioua, S. M. Senouci, H. Sedjelmaci, and S. Moussaoui. Incentive edge caching in software defined internet of vehicules : A stackelberg approch. *International Journal of Communication Systems*, pages 1–20, 2018.

[52] R. Naja et al. Wireless vehicular networks for car collision avoidance. In *R. Naja (ed.) Springer Science*, New York, July 2013.

[53] V. Hoa LA and A. Cavalli. Security attacks and solutions in vehicular ad hoc networks : a survey. *International Journal on AdHoc Networking Systems (IJANS)*, 4(2), 2014.

[54] H. Moustafa and G. Bourdon. Vehicular networks deployment view : Applications, deployment architectures and security means. *Ubiquitous Computing and Communication Journal, special issue on Ubiquitous Roads*, 2008.

[55] F. Filali J. Harri and C. Bonnet. Mobility models for vehicular ad hoc networks : A survey and taxonomy. *IEEE communications surveys and tutorials*, 11(4) :19–41, 2009.

[56] J. Santa, A.F. Gómez-Skarmeta, and M. Sánchez-Artigas. Architecture and evaluation of a unified v2v and v2i communication system based on cellular networks. *Computer Communications*, 31(12) :2850– 2861, 2008.

[57] T. Kosch adn C.J. Adler, S. Eichler, and C. Schroth. The scalability problem of vehicular ad hoc networks and how to solve it. *IEEE Wireless Communications*, 13 :22–28, 2006.

[58] A. Mahajan, N. Potnis, K. Gopalan, and An-I A. Wang. Urban mobility models for vanets. In *IEEE Workshop On Next Generation Wireless Networks*, 2006.

[59] M. Fiore, J. Harri, F. Filali, and C. Bonnet. Vehicular mobility simulation for vanets. In *Proceedings of the 40th Annual Simulation Symposium*, Norfolk, 2007.

[60] Y. Khaled, M. Tsukada, J. Santa, J. Choi, and T. Ernst. A usage oriented analysis of vehicular networks : from technologies to applications. *Journal of communications*, 4(5) :22–28, 2009.

[61] A. Boualouache, S. M. Senouci, and S. Moussaoui. A survey on pseudonym changing strategies for vehicular ad-hoc networks. *IEEE Communications Surveys and Tutorials*, Early view, 2018.

[62] J. Cheng, J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu. Routing in internet of vehicles : A review. *IEEE Transactions on Intelligent Transportation Systems*, 16(5) :2339–2352, 2015.

[63] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie. A survey on software-defined networking. *IEEE Communications Surveys and Tutorials*, 17(1) :27–51, 2015.

[64] Openstack. https ://www.openstack.org/.

[65] Opendaylight. http ://www.opendaylight.org/.

[66] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud. Software-defined networking : Challenges and research opportunities for future internet. *Computer Networks*, 75(A) :453–471, 2014.

[67] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Open flow : enabling innovation in campus networks. *SIGCOMM Comput. Commun*, 38(2) :69–74, 2008.

[68] Open networking fondation, https ://www.opennetworking.org/.

[69] M. Azizian, S. Cherkaoui, and A. S. Hafid. Vehicle software updates distribution with sdn and cloud computing. *IEEE Communications Magazine*, 55(8) :74–79, 2017.

[70] I. Yaqoob, I. Ahmad, E. Ahmed, A. Gani, M. Imran, and N. Guizani. Overcoming the key challenges to establishing vehicular communication : Is sdn the answer ? *IEEE Communications Magazine*, 55(7) :128–134, 2017.

[71] M. ZHU, Z.P. CAI, and M. XU. Software-defined vehicular networks : Opportunities and challenges. In *2nd International Conference on Energy Science and Applied Technology (ESAT)*, Wuhan, 2016.

[72] G. Remy, M. Cherif, SM. Senouci, F. Jan, and Y. Gourhant. Lte4v2x : Lte for a centralized vanet organization. In *IEEE GLOBECOM'2011*, Houston, 2011.

[73] G. Remy, M. Cherif, SM. Senouci, F. Jan, and Y. Gourhant. Lte4v2x - collection, dissemination and multi-hop forwarding. In *IEEE ICC'2012*, Ottawa, 2012.

[74] M. S. Rayeni, A. Hafid, and P. K. Sahu. Quality of service aware multicasting in heterogeneous vehicular networks. *Vehicular Communications*, 13 :38–55, 2018.

[75] Network simulator 3 (ns-3), http ://www.nsnam.org.

[76] Simulation of urban mobility (sumo), http ://sumo-sim.org.

[77] O. Oubbati, L. Lakas, F. Zhou, M. Gunes, L. Lagraa, and M. B. Yagoubi. Intelligent uav-assisted routing protocol for urban vanets. *Computer Communications*, 107 :93–111, 2017.

[78] W. Fawaz, R. Atallah, C. Assi, and M. Khabbaz. Unmanned aerial vehicles as store-carry-forward nodes for vehicular networks. *IEEE Access*, 5 :23710–23718, 2017.

[79] W. Fawaz. Effect of non-cooperative vehicles on path connectivity in vehicular networks : A theoretical analysis and uav-based remedy. *Vehicular Communications*, 11 :12–19, 2017.

[80] H. Seliem M. H. Ahmed, R. Shahidi, and M. S. Shehata. Delay analysis for drone-based vehicular ad-hoc networks. In *IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*, Montreal, 2017.

[81] V. Sharma, H. Chen, and R. Kumar. Driver behaviour detection and vehicle rating using multi-uav coordinated vehicular networks. *Journal of Computer and System Sciences*, 86 :3–32, 2017.

[82] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang, and X. S. Shen. Software defined space-air-ground integrated vehicular networks : Challenges and solutions. *IEEE Communications Magazine*, 55(7) :101–109, 2017.

[83] H. Ghazzai, H. Menouar, and A. Kadri. On the placement of uav docking stations for future intelligent transportation systems. In *IEEE 85th vehicular technology conference (VTC Spring)*, Sydney, 2017.

[84] M. A. Messous, A. Arfaoui, A. Alioua, and S. M. Senouci. A sequential game approach for computation-offloading in an uav network. In *IEEE GLOBECOM*, Singapore, 2017.

[85] L. Gupta, R. Jain, and G. Vaszkun. Survey of important issues in uav communication networks. *IEEE Communications Surveys and Tutorials*, 18(2) :1123–1152, 2016.

[86] M. Sara, I. Jawhar, and M. Nader. A softwarization architecture for uavs and wsns as part of the cloud environment. In *2016 IEEE international conference on cloud engineering workshop (IC2EW)*, Berlin, 2016.

[87] A. Ramaprasath, A. Srinivasan, C. H. Lung, and M. St-Hilaire. Intelligent wireless ad hoc routing protocol and controller for uav networks. In *Y. Zhou and T. Kunz (Eds.), Ad Hoc Networks. Social informatics and telecommunications engineering : Lecture notes of the institute for computer sciences*, Berlin, 2017.

[88] Z. Yuan, X. Huang, L. Sun, and J. Jin. Software defined mobile sensor network for micro uav swarm. In *2016 IEEE international conference on control and robotics engineering (ICCRE)*, Singapore, 2016.

[89] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5) :2795–2808, 2016.

[90] M. Deng, H. Deng, and X. Lyu. Adaptive sequential offloading game for multi-cell mobile edge computing. In *2016 23rd international conference on telecommunications (ICT)*, Thessaloniki, 2016.

[91] X. Chen. Decentralized computation offloading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 26(54) :974–983, 2014.

[92] R. Yu, J. Ding, X. Huang, M. T. Zhou, S. Gjessing, and Y. Zhang. Decentralized computation offloading game for mobile cloud computing. ieee transactions on parallel and distributed systems. *IEEE Transactions on Vehicular Technology*, 65(10) :7844–7856, 2016.

[93] H. W. Kuhn. Extensive games and the problem of information. *Annals of Mathematical Studies*, 2(28) :193–216, 1953.

[94] U. Schwalbe and P. Walker. Zermelo and the early history of game theory. *Games and Economic Behavior*, 34 :123–137, 2001.

[95] A. Stetsko, L. Folkman, and V. Matay. Neighbor-based intrusion detection for wireless sensor network. In *6th international conference on wireless and mobile communications*, Valencia, 2010.

[96] Y. J. Li, H. Deng, and X. Lyu. An overview of the dsrc/wave technology. In *international conference of on heterogeneous networking for quality, reliability, security and robustness*, 2012.

[97] H. Zhang, B. Liu, M. Li, H. Ji, X. Li, and VCM Leung. Pricing, caching selection, and content delivery in wireless networks : a hierarchical approach. In *Proceedings of the IEEE Global Communications Conference - GLOBECOM*, Singapore, 2017.

[98] K. Poularakis, G. Iosifidis, and L. Tassiula. A framework for mobile data offloading to leased cache-endowed small cell networks. In *Proceedings of the IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, Philadelphia, 2014.

[99] Z. Chen, Y. Liu, B. Zhou, and M. Tao. Caching incentive design in wireless d2d networks : A stackelberg game approach. In *Proceedings of the IEEE International Conference on Communications (ICC)*, Kuala Lumpur, 2016.

[100] K. S. Sahoo, B. Sahoo, D. K. Lobiyal, D. P. Mohapatra, A. Nagar, and M. N. Sahoo. Sdn architecture on fog devices for realtime traffic management : A case study. In *Proceedings of the International Conference on Signal, Networks, Computing, and Systems*, New Delhi, 2017.

[101] J.M. Duarte, E. Kalogeiton, and R. Soua et al. A multi-pronged approach to adaptive and con aware content dissemination in vanets. *Mobile Netw Appl*, page 1–13, 2017.

[102] S. W. Jeon, S. N. Hong, M. Ji, G. Caire, M.R. Palattella, A. D. Maio, and A.F. Molisch. Wireless multi hop device-to-device caching networks. *IEEE Trans Inf Theory*, 63(3) :1662–1676, 2017.

[103] C. Xu, T. Li, M. Shen, and J. Li. Self-organized dynamic caching space sharing in virtualized wireless networks. In *Proceedings of the IEEE Globecom Workshops (GC Wkshps)*, 2017.

[104] Y. Xu, J. Wang, Q. Wu, Z. Du, L. Shen, and A. Anpalagan. A game-theoretic perspective on self-organizing optimization for cognitive small cells. *IEEE Commun Mag*, 53(7) :100–108, 2015.

[105] Y. Xu, J. Wang, Q. Wu, Z. Du, L. Shen, and A. Anpalagan. A low-complexity approach to distributed cooperative caching with geographic constraints. In *Proceedings of the ACM Meas Anal Comput Syst.*, 2017.