Projet Ingénierie des Réseaux
Rapport

Nesrine HARBAOUI, Yann MARSEILLAN, Selma OUJID, Anthony PEYRONNET

Département Sciences du Numérique - 2SN Réseaux
2022-2023

Table des matières

1 Introduction

Le développement de technologies visant à assurer la sécurité des utilisateurs vulnérables de la route (VRU) constitue une préoccupation majeure dans le domaine de la sécurité routière. L'une des approches prometteuses pour atteindre cet objectif consiste à utiliser des applications de protection des VRU qui exploitent les informations de localisation GPS pour évaluer la proximité dangereuse entre les VRU et les véhicules. Ces applications sont capables de générer des alertes qui peuvent être transmises à différents acteurs tels que les agents intelligents embarqués dans les véhicules, les VRU eux-mêmes ou les infrastructures de bordure de la route.

Cependant, il convient de noter que toutes les routes ne sont pas équipées d'infrastructures dédiées pour la communication entre les différents acteurs impliqués. Par conséquent, il est nécessaire de prendre des décisions sur la manière dont la communication est établie entre les VRU et les véhicules, en tenant compte de divers facteurs tels que la qualité du signal, la proximité, la charge du réseau, etc.

Dans ce contexte, une approche basée sur les Multi-Armed Bandit (MAB) se révèle particulièrement pertinente pour la gestion de la communication entre les VRU et les véhicules. Les algorithmes MAB permettent de prendre des décisions intelligentes en explorant et en exploitant différentes options disponibles, en fonction de critères préalablement définis. Ces critères peuvent inclure la qualité du signal GPS, la proximité des VRU, la charge du réseau, et d'autres facteurs pertinents.

Durant ce projet, nous avons d'abord inspecté l'état de l'art sur le sujet. Nous avons également réfléchi sur l'aspect QoS du problème avant de nous lancer nous même dans des simulations et l'implantation d'algorithmes MAB. Ce rapport retrace les étapes de notre projet.

2 Etat de l'art

La première étape du projet est de faire un état de l'art autour de notre sujet. Plusieurs chercheur.euse.s de plusieurs université se sont penché.e.s sur le développent de technologies V2X (Vehicule to Everything).

Selon les recherches, une méthode qui permet de bien répondre à la problématique est le Reinforcement Learning (RL). Le RL est utilisé dans le contexte de la sélection du mode de communication V2V et de l'adaptation de la puissance dans les réseaux 5G car il permet de résoudre le problème d'optimisation. Dans ce scénario, des ressources spectrales limitées sont occupées de manière concurrentielle par plusieurs liens V2V, et différents modes de transmission peuvent être formés selon la manière dont les liens V2V occupent les ressources spectrales. Cela peut être modélisé comme un problème de RL multi-agent lorsque le lien V2V est considéré comme un agent. Ainsi, le RL permet aux liens V2V de prendre des décisions intelligentes dans le réseau de communication V2V, ce qui peut améliorer la satisfaction de chaque indicateur cible dans la sélection de motifs en temps réel.

Un aspect important de la recherche est l'utilisation des réseaux 5G dans les communications. La 5G est utilisée car elle offre de nombreux avantages par rapport aux générations précédentes de réseaux cellulaire. Ces avantages sont notamment une plus grand bande passante, une latence plus faible, une plus grande fiabilité et une connectivité plus large. Ceux-ci permettent le développement de nouvelles applications et services qui n'étaient pas possibles auparavant, tel que les véhicules autonomes, et permet de prendre en charge un grand nombre d'appareil et de capteurs dans l'écosystème de l'Internet des Objets (IoT).

3 QoS et métrique

Notre projet porte sur la mise en place d'un algorithme de reinforcement learning en l'occurrence le multi-armed bandit, afin d'établir une sélection entre une communication directe ou une communication via une infrastructure. L'objectif est de développer une application de protection des utilisateurs de la route vulnérables qui permet l'envoie d'alertes entre les véhicules et les utilisateurs vulnérables (directe ou via infrastructure).

Pour assurer le bon fonctionnement de cette application et la mise en place de l'algorithme de reinforcement learning, il est primordial de définir différents critères de qualité de service. Notre projet portant sur un système critique à temps réel, plusieurs métriques sur les caractéristiques réseau sont à prendre en compte : le délai de transmission, les pertes de paquet, la gigue. Il est aussi possible de définir des métriques au niveau de l'application et du fonctionnement du système souhaité qui oriente le choix du multi-armed bandit : la notion de risque qui est évaluée avec l'aide de la proximité entre usagers, l'état du réseau... En effet, selon la proximité des usagers les alertes à transmettre peuvent être critiques, il est donc nécessaire d'assurer un délai de transmission assez faible et d'éviter les pertes de paquet.

Pour caractériser notre ordonnancement nous avons eu le débat autour du working vs non working : En QoS (Quality of Service), la notion de work-conserving vs non-work-conserving se rapporte à la manière dont les ressources réseau sont utilisées pour fournir différents niveaux de qualité de service. Un ordonnancement work-conserving est un ordonnancement qui utilise toutes les ressources disponibles à chaque instant pour traiter les demandes entrantes. Autrement dit, lorsque des demandes arrivent, elles sont immédiatement traitées et les ressources sont utilisées pour traiter ces demandes, même si cela signifie interrompre temporairement les demandes en cours de traitement. Dans un système work-conserving, toutes les demandes sont traitées dès que possible, et il n'y a aucun temps d'arrêt ou d'inactivité des ressources.

D'un autre côté, un système non-work-conserving est un système qui n'utilise pas toutes les ressources disponibles à chaque instant. Dans ce cas, le système peut réserver des ressources pour des demandes futures, ou ne pas utiliser certaines ressources même si des demandes sont en attente. Cela peut être utile pour des applications qui nécessitent des garanties de délai strictes ou qui ont des exigences de bande passante minimales.

Dans notre cas nous nous sommes orientés naturellement vers du work conserving, nous sommes dans une application temps réel, nous avons donc besoin d'une rapidité de traitement, ainsi le work conserving nous a semblé le choix le plus opportun notamment sur notre contrainte de délai puisque un ordonnancement non work conserving entraîne un plus grand délai de bout en bout. Par ailleurs ne pas traiter des demandes en attentes même si des ressources sont disponibles ne fait pas sens dans une telle application, les tailles des données transmises étant aussi très restreintes faire de la réservation n'est pas forcément utile et nécessaire.

De ce fait, dans le cadre d'un ordonnancement work conserving nous avons opté pour l'utilisation de l'algorithme d'ordonnancement préemptif EDF (Earliest Deadline First). C'est un algorithme à priorité dynamique qui permet d'attribuer une priorité à chaque requête en fonction de l'échéance de cette dernière, les tâches dont l'échéance est proche recevant la priorité la plus élevée. Cet algorithme nous semble donc idéal pour notre système temps réel. En revanche, la mise en place de cet algorithme est assez complexe, il est donc nécessaire d'en évaluer sa complexité ce qui correspond alors à une métrique de mesure de performance.

Sur ce projet nous avons aussi eu le débat sur notre choix de se placer en Intserv ou DiffServ, qui sont deux approches différentes pour assurer de la QoS dans un réseau IP (ce qui est notre cas ici). Intserv est un modèle de QoS basé sur la réservation de ressources. Il s'agit d'une approche basée sur la qualité de service pour le trafic à faible volume, où les applications qui ont besoin

d'une qualité de service garantie doivent signaler leurs besoins de bande passante et de délai à travers le réseau. Les routeurs dans le réseau maintiennent ensuite des ressources dédiées pour chaque flux de trafic, en garantissant des débits minimums et des délais maximums.

D'autre part, DiffServ est un modèle de QoS basé sur la classification de trafic. Il est conçu pour les réseaux à haut débit, où les routeurs doivent être en mesure de traiter un grand volume de trafic sans sacrifier la qualité de service. Dans DiffServ, les paquets sont marqués avec des étiquettes de priorité pour identifier la classe de service souhaitée. Les routeurs dans le réseau peuvent ensuite appliquer des politiques de traitement de trafic pour chaque classe de service en fonction des besoins de l'application.

A la description des deux approches nous en avons conclu que l'approche Diffserv convenait plus à notre application. Cela s'explique premièrement par le fait que IntServ effectue de la réservation de ressources, ce qui peut impliquer à terme un impossibilité de transmission de trafic si toutes les ressources sont déjà réservées, c'est un problème majeur pour notre application si un trafic urgent temps réel doit être transmis il peut être bloqué le cas échéant, même si c'est peut être faire du zèle puisque très probablement que le rapport en nos ressources disponibles/données à transmettre (BW, Mémoire..) sera très largement suffisant au vu des taille des messages transmis (message d'alerte court). Néanmoins un autre argument nous fait définitivement pencher du côté de Diffserv, celui de la réalité des systèmes embarqués qui doivent avoir une empreinte mémoire faible pour différentes raisons :

— Limitations matérielles : Les SE ont souvent des ressources matérielles limitées, tels que la mémoire et la puissance de traitement, qui doivent être utilisées de manière efficace pour garantir une performance optimale. Un SE avec une empreinte mémoire importante pourrait entraîner une utilisation excessive de la mémoire ou une consommation excessive de CPU, ce qui peut ralentir le dispositif ou réduire sa durée de vie.

— Communication de données : Les appareils embarqués dans notre cas sont souvent conçus pour communiquer avec des serveurs distants, qui peuvent être situés sur des réseaux à faible bande passante ou avec des connexions intermittentes. Les SE doivent donc être optimisés pour minimiser la quantité de données qui sont échangées, afin de réduire les coûts de transmission des données et la consommation d'énergie des appareils.

— Sécurité : Les dispositifs embarqués peuvent être vulnérables aux attaques de sécurité car ils sont souvent déployés dans des environnements non sécurisés et peuvent être facilement compromis. Un SE avec une empreinte mémoire importante peut augmenter la surface d'attaque des appareils.

L'approche Intserv nous aurait permis de faire de la gestion de congestion avec RSVP notamment, mais nous avons décidé de ne pas vraiment faire de QoS de type congestion au vu de la taille et de la sporadicité du trafic. Diff Serv étant aussi plus scalable et permettant de rester cohérent avec notre type de QoS basé sur la priorité nous nous sommes tournés vers cette approche, elle permet aussi indirectement de gérer les quelques soucis de congestion que nous pourrions avoir en faisant passer les messages prioritaires en premier, sachant de des messages non prioritaires même perdu ne sont pas critiques pour notre application (exemple : moto dans arrivant à 100m d'un poid lourd). Diffserv étant d'une complexité supérieure on peut prendre en compte la métrique de mesure de complexité pour évaluer la performance de cette approches DiffServ.

Afin d'optimiser le délai de transmission, l'utilisation du protocole de transport UDP permet d'éviter la mise en place de connexion entre les usagers avant de pouvoir communiquer ce qui ajoute évidemment du délai. Par ailleurs, l'absence de contrôle de congestion dans le protocole assure que la circulation ne soit pas bridée. Cependant, cela implique d'évaluer les pertes de paquet qui pourrait provenir de congestion afin d'établir si un contrôle de congestion est tout de même nécessaire. Il semble que les risques de congestion restent assez faibles en considérant que les alertes transmises entre usagers sont assez simples et en prenant en compte le fait qu'il y a deux moyens de communiquer (direct ou via infrastructure). Donc L'état du réseau est bien une

métrique à prendre en compte afin d'orienter la sélection du moyen de communication entre les usagers.

Pour conclure, afin d'assurer le bon fonctionnement de notre système temps réel, il est nécessaire de prendre en compte plusieurs types de métriques relatif aux différents critères de QoS souhaités. Nous avons défini des métriques sur le réseau (délai de transmission, pertes de paquet, gigue), au niveau de l'application (notion de risque, état du réseau) et enfin afin de mesurer les performances des différentes solutions (complexité d'un algorithme, efficacité, précision...).

4 Simulation

4.1 NS-3

Network Simulator 3 (NS-3) est un environnement de simulation open source permet de modéliser et de simuler des réseaux complexes afin d'évaluer leurs performances et d'étudier le comportement des protocoles de communication. Nous utilisons donc cet environnement pour nos simulation car c'est un programme très répandu dans le domaine de la recherche et du développement en réseaux de télécommunications.

4.2 Simulations Python

Nous avons mis en place des codes python afin de simuler les communications entre les différentes entités. Le script principal simule soit une communication entre utilisateurs (piéton, motard ou voiture), qu'on appelle une communication direct, soit une communication entre les utilisateurs via une infrastructure.

Voici comment nous avons implémenté notre code avec différentes classes :

La classe Message représente un message envoyé entre utilisateurs. Elle contient des attributs tels que l'identifiant de l'expéditeur, l'identifiant du destinataire, la priorité et la taille du message.

La classe User représente un utilisateur (véhicule, moto, piéton) dans le réseau. Chaque utilisateur a un identifiant, un protocole de communication, une portée de communication, une capacité de traitement et un type d'utilisateur. Les utilisateurs peuvent envoyer des messages à d'autres utilisateurs et recevoir des messages. Il y a également une file d'attente pour stocker les messages en attente de transmission.

La classe Protocol représente un protocole de communication. Chaque protocole a un nom, une charge de réseau, un taux de perte de paquets, un taux de réussite de transmission et un temps de transmission.

La classe Metrics est utilisée pour collecter et mettre à jour les métriques de communication telles que le délai de transmission, le taux de perte de paquets et la charge du réseau.

La classe Infrastructure représente une infrastructure dans le réseau V2I. Elle est similaire à la classe User mais a des fonctionnalités spécifiques aux infrastructures.

La partie principale du code initialise les protocoles, les utilisateurs, les infrastructures et les métriques, puis simule la communication V2V ou V2I en envoyant des messages entre utilisateurs et en traitant les files d'attente. Les utilisateurs se déplacent à chaque étape de simulation.

A la fin de la simulation, nous avons trois métriques qui ressortent : le délai moyen, la charge moyenne du réseau et le taux de perte. Notre but par la suite est de récupérer ces trois métriques dans un fichier CSV. Ensuite, nous allons lire le fichier dans le code du Multi-Armed Bandit : celui-ci utilise deux méthodes différentes, UCB et Epsilon-greedy.

Le code MAB va quant à lui lire le ficher CSV. Nous allons ensuite attribuer les métriques soit au cas de communication directe, soit au cas de communication entité/infrastructure. Par la suite, une fois le code lancé, selon les métriques que nous avons trouvé dans la simulation, on peut choisir quel mode de communication est le plus adapté.

La simulation à l'aide de scripts python implique de faire des simplifications sur le modèle que l'on souhaite simuler. Dans ce contexte, il est compliqué de mettre en place l'ensemble des mécanismes de QoS vu précédemment. En revanche, nous avons tout de même inclus une gestion de file d'attente, des niveaux de priorités selon le type d'utilisateurs, un taux de succès de transmission pour le protocole de communication entre utilisateurs différent du protocole de communication avec une infrastructure. Cela permet de mettre en valeur la fiabilité des infrastructures. Par ailleurs, nous avons aussi pris en compte la charge actuelle du réseau et la distance entre les utilisateurs dans la réussite de l'envoie de messages.

5 Implantation de MAB

L'algorithme du Multi-Armed Bandit, ou bandit à plusieurs bras en français, est une approche utilisée pour résoudre le problème de l'exploration-exploitation. Il tire son nom des bandits à un bras, où un joueur doit choisir entre plusieurs machines à sous (bras) et décider laquelle jouer pour maximiser ses gains.

Dans le contexte de l'algorithme du multi-armed bandit, chaque "bras" représente une action parmi plusieurs choix possibles. Chaque bras a une certaine récompense associée. L'objectif est de maximiser les récompenses totales en prenant des décisions séquentielles quant à la sélection des bras à tirer.

Au début, toutes les actions sont considérées comme inexplorées. Les récompenses associées à chaque bras peuvent être inconnues ou initialement estimées. L'algorithme commence par explorer différentes actions pour collecter des informations sur les récompenses potentielles de chaque bras. Il choisit aléatoirement un bras à tirer (ou utilise une stratégie plus sophistiquée, comme l'algorithme epsilon-greedy que nous allons détailler par la suite), observe la récompense associée à cette action et met à jour ses estimations de récompense pour ce bras. À mesure que l'algorithme explore, il utilise les informations accumulées pour prendre des décisions d'exploitation. Il choisit généralement l'action avec la plus grande estimation de récompense pour maximiser les récompenses à court terme. L'algorithme continue à alterner entre les phases d'exploration et d'exploitation, en explorant périodiquement des actions pour mettre à jour les estimations de récompense et en exploitant les actions les plus prometteuses pour maximiser les gains à long terme.

Dans notre projet, l'algorithme Multi-Armed Bandit se décline en deux méthode. Nous avons utilisé l'apporche epsilon-greedy ainsi que l'approche UCB.

— L'algorithme Epsilon-Greedy suit une politique d'exploration-exploitation, qui tente d'équilibrer entre l'exploration des bras pour découvrir de nouvelles actions potentiellement bénéfiques (exploration) et l'exploitation des bras qui ont donné de bonnes récompenses par le passé (exploitation). Lorsque l'algorithme Epsilon-Greedy sélectionne une action à exécuter, il utilise un paramètre epsilon qui détermine la probabilité d'exploration. Si un nombre aléatoire entre 0 et 1 est inférieur à epsilon, l'algorithme sélectionne un bras au hasard parmi tous les bras possibles (exploration). Sinon, l'algorithme sélectionne le bras qui a donné la meilleure récompense jusqu'à présent (exploitation). Cette approche permet d'explorer de nouvelles actions tout en exploitant les bras prometteurs.

— L'algorithme UCB (Upper Confidence Bound) utilise une stratégie d'exploration basée sur une borne supérieure de confiance. Il attribue une valeur à chaque bras basée sur une combinaison de la récompense moyenne du bras et d'un terme d'exploration qui dépend de l'incertitude associée à la récompense. L'idée principale est de favoriser les bras qui semblent prometteurs tout en explorant suffisamment les autres bras pour estimer leurs récompenses potentielles. L'algorithme UCB calcule la valeur UCB pour chaque bras en utilisant la formule UCB : valeur du bras + coefficient d'exploration * racine carrée du logarithme naturel du nombre total d'étapes divisé par le décompte du bras. La valeur UCB estime la récompense potentielle d'un bras en tenant compte de la récompense moyenne actuelle et de l'incertitude

associée. L'algorithme UCB sélectionne le bras avec la plus grande valeur UCB pour être exécuté. Au fil du temps, les estimations des récompenses moyennes et des incertitudes sont mises à jour en fonction des récompenses reçues.

Ces deux algorithmes offrent une approche équilibrée entre l'exploration et l'exploitation pour résoudre les problèmes de bandit multi-bras. L'algorithme Epsilon-Greedy explore plus agressivement en sélectionnant des bras aléatoires, tandis que l'algorithme UCB utilise une stratégie plus sophistiquée en tenant compte de l'incertitude associée aux récompenses. Nous avons fait le choix d'utiliser ces deux approches pour répondre aux exigences spécifiques du problème à résoudre.

6 Scénario d'étude

Nous avons fait plusieurs simulations que nous avons intégré aux algorithmes Multi-Armed Bandit Epilon-greedy et UCB.

Les configuration de simulation sont les suivantes :

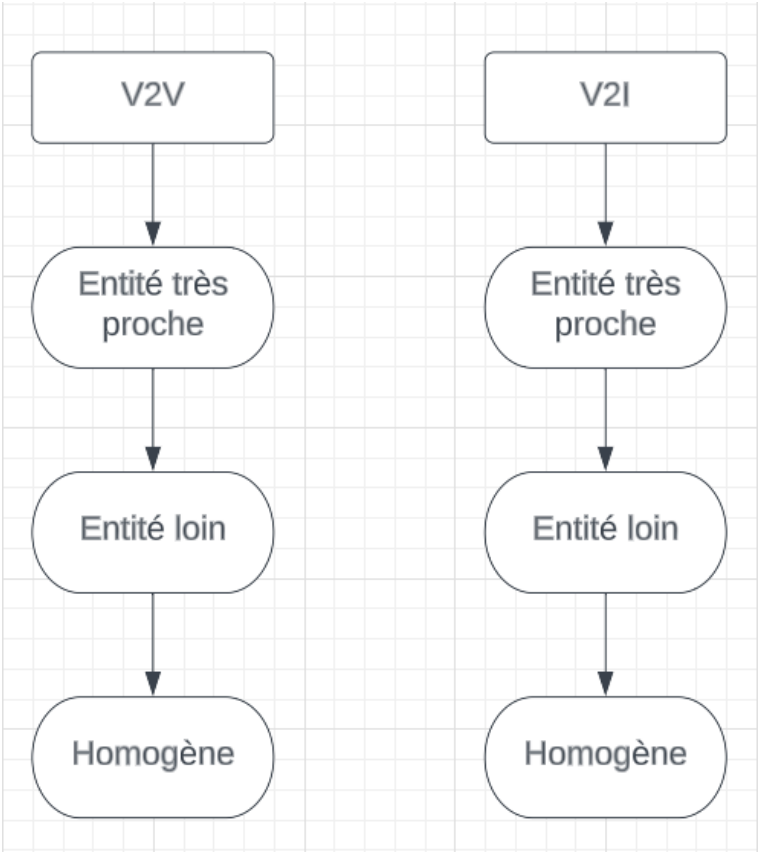

FIGURE 1 – Simulation

Il s'agit d'un protocole à priorité pour les deux scénarios V2V et V2I. La distribution de la population à priorité se fait en 3 niveaux :

Automobile (Poids = 3)

Motard (Poids = 2)

Piéton (Poids = 1)

Les métriques à prendre en compte et à utiliser pour les choix sont :

— La charge

— Le délai

— Le taux de perte

Après avoir fait des codes pour une simulation, voici les résultats que nous pouvons en tirer :

Résultats :

	Métriques V2V	Métriques V2I	Choix UCB	Choix epsilon-g.
Scénario-1 (proche)	Délai : 0.050323 Taux de pertes : 0.3 Charge : 0.258087	Délai : 0.050689 Taux de pertes : 0.25 Charge : 0.372391	V2I	V2V
Scénario-2 (homogène)	Délai : 0.085701 Taux de pertes : 0.3 Charge : 0.488981	Délai : 0.105674 Taux de pertes : 0.2 Charge : 0.523045	V2V	V2V
Scénario-3 (éloigné)	Délai : 0.119313 Taux de pertes : 0.5 Charge : 0.541394	Délai : 0.165917 Taux de pertes : 0.2 Charge : 0.577715	V2I	V2V

FIGURE 2 – Résultats de notre simulation

Analyse des résultats

Les métriques suivent une bonne logique : les délais augmentent bien avec les distances entre les utilisateurs, les taux de pertes sont assez constants mais on observe une augmentation pour V2V dans le cas éloigné (trop grande distance pour la communication directe d'où une perte plus importante de paquets) et la charge augmente avec la distance car les paquets restent plus longs dans le réseau . On observe d'ailleurs une charge un peu plus importante avec les infrastructures, ce qui est logique avec des entités et des communications supplémentaires par rapport à V2V.

Remarque : les taux des pertes sont intentionnellement assez élevé pour la simulation car il y a peu de messages qui transitent donc une perte a plus d'impact sur la valeur finale.

Au niveau du choix des algorithmes de Multi-Armed Bandit, on remarque que selon la méthode utilisée, le choix final n'est pas forcément identique. La politique d'exploration n'étant pas la même, c'est tout à fait logique. Mise à part le scénario homogène, les algorithmes de Multi-Armed Bandit ne font pas le même choix. En effet, les différences entre les métriques se compensant d'une métrique à une autre, cela laisse une plus grande liberté de décision d'où les différences dans le choix .

Axe d'amélioration :
Le choix dépend grandement du poids que l'on attribue à chaque métrique. Dans le cadre de notre simulation nous avons posé les poids suivant :

$$delay\_weight = 1,\ loss\_rate\_weight = 1 \text{ et } load\_weight = 0.2$$

Dans notre cadre d'étude, il aurait pu être intéressant d'accentuer le poids du taux de perte de paquets, celui-ci ayant intentionnellement une valeur élevée. En effet, un taux de perte à 0.5 devrait empêcher le MAB epsilon greedy de choisir la communication V2V.

7 Conclusion

En conclusion, ce rapport a présenté notre projet d'ingénierie des réseaux axé sur le développement d'une application de protection des utilisateurs vulnérables de la route. Nous avons exploré l'utilisation des algorithmes Multi-Armed Bandit (MAB) pour gérer la communication entre les utilisateurs vulnérables et les véhicules, en prenant en compte des critères tels que la qualité du signal GPS, la proximité des utilisateurs et la charge du réseau.

Nous avons effectué un état de l'art pour comprendre les avancées dans le domaine de la sécurité routière et les technologies V2X (Vehicle-to-Everything). Nous avons constaté que le Reinforcement Learning (RL) est largement utilisé dans la sélection du mode de communication V2V et dans l'adaptation de la puissance dans les réseaux 5G.

Nous avons également abordé la question de la qualité de service (QoS) et identifié des métriques telles que le délai de transmission, les pertes de paquet et la gigue comme des critères importants pour évaluer la performance de notre application. Nous avons opté pour un ordonnancement work-conserving avec l'algorithme préemptif EDF (Earliest Deadline First) pour assurer une réponse rapide aux demandes critiques en temps réel.

En ce qui concerne l'approche de QoS, nous avons choisi l'approche DiffServ (Differentiated Services) plutôt que l'approche IntServ (Integrated Services) en raison de sa scalabilité et de sa compatibilité avec les contraintes des systèmes embarqués. Nous avons utilisé le protocole de transport UDP (User Datagram Protocol) pour minimiser le délai de transmission, tout en évaluant les pertes de paquet pour déterminer si un contrôle de congestion était nécessaire.

Enfin, nous avons exploré des approches et des techniques appropriées pour développer une application de protection des utilisateurs vulnérables de la route, en mettant l'accent sur l'utilisation des algorithmes MAB et l'optimisation de la communication entre les utilisateurs et les véhicules.

8 Références

[1] https://link.springer.com/article/10.1007/s11276-022-03191-7 - 5G-enabled V2X communications for vulnerable road users safety applications : a review

[2] https://5ginfire.eu/vru-safe/ Vulnerable Road Users Safety using a hybrid Cloud RAN and Edge Computing model

[3] https://ieeexplore.ieee.org/document/9046279 A Reinforcement Learning Method for Joint Mode Selection and Power Adaptation in the V2V Communication Network in 5G

[4] https://aleksandarhaber.com/multi-armed-bandit-with-python-codes-and-implementation/ Multi Armed Bandit with Python Codes and Implementation