RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Mohamed El-Bachir El-Ibrahimi - BBA

Faculté des Mathématiques et Informatiques

MÉMOIRE

Présenté en vue de l'obtention du diplôme

Master en Informatique

Spécialité : Réseaux & Multimédias

Thème

Intégration de l'Apprentissage Profond pour l'Optimisation du Routage dans les Réseaux FANETs

Présenté par :

- KHIER Dounia
- BELMOUMENE Houda

Soutenu le : 11/06/2025, devant la commission d'examen suivante :

Dr. Nadjib BENAOUDA	MCB - Université de BBA	Président
Dr. Hassina BENSEFIA	MCB - Université de BBA	Examinatrice
Dr. Oussama SENOUCI	MCB - Université de BBA	Encadrant

Promotion 2024 / 2025

**“Je n’ai jamais rêvé de succès,
J’ai travaillé pour ça.”**

- Estee Lauder -

Résumé

Ce mémoire de fin d'études s'inscrit dans le domaine des réseaux sans fil mobiles , plus précisément les Flying Ad Hoc Networks (FANETs), en proposant une approche intelligente pour améliorer la stabilité des communications entre drones. L'objectif principal est de concevoir, modéliser et évaluer un protocole de routage prédictif capable de distinguer les liens stables des instables, afin de garantir une meilleure qualité de service (QOS).Le travail débute par une étude approfondie de l'architecture des drones dans les FANETs, suivie d'une analyse critique des protocoles de routage classiques et des méthodes de gestion de la QoS. Sur cette base, nous avons proposé une amélioration du protocole P-OLSR en y intégrant un modèle basé sur les techniques de Machine Learning et Deep Learning de pour la prédiction de la métrique Expected Transmission Count (ETX), indicateur clé de la qualité des liens.Le modèle proposé a été entraîné et validé à l'aide d'un dataset simulant les caractéristiques des liens FANETs. Une série de simulations a été réalisée pour évaluer les performances de notre protocole, notamment en termes de stabilité des connexions, de débit, de latence et de fiabilité du routage.Les performances finales du modèle sur le jeu de test indiquent une accuracy de 98.58%, avec une précision de 98.37%, un rappel de 98.38%, et une perte de 0.0612, démontrant une très bonne capacité de généralisation.Les simulations confirment la supériorité du protocole proposé, avec un taux de stabilité de 98.8 %, une perte réduite à 3 %, une transmission utile de 97 %, une résilience de 6 %, et 92 % de liens actifs, surpassant ainsi OLSR et Predictive-Optimized Link State Routing (P-OLSR).

Mots clés : FANETs, Routage, Qualité de Service, P-OLSR, Deep Learning, ETX.

Abstract

This graduation thesis falls within the domain of mobile wireless networks, specifically Flying Ad Hoc Networks (FANETs), and proposes an intelligent approach to improve the stability of drone communications. The main objective is to design, model, and evaluate a predictive routing protocol capable of distinguishing stable links from unstable ones, in order to ensure better Quality of Service (QoS).The work begins with an in-depth study of drone architectures in FANETs, followed by a critical analysis of conventional routing protocols and QoS management methods. Based on this analysis, we proposed an enhancement of the P-OLSR protocol by integrating a model based on Machine Learning and Deep Learning techniques to predict the Expected Transmission Count (ETX) metric, a key indicator of link quality.The proposed model was trained and validated using a dataset simulating FANET link characteristics. A series of simulations was conducted to evaluate the performance of our protocol in terms of connection stability, throughput, latency, and routing reliability.The final performance of the model on the test set shows an accuracy of 98.58%, with a precision of 98.37%, a recall of 98.38%, and a low loss of 0.0612, demonstrating strong generalization ability.Simulation results also confirm the superiority of the proposed protocol, with a stability rate of 98.8% %, a reduced loss of 3 %, a useful transmission rate of 97 %, a resilience of 6 %, and 92 % active links, thus outperforming OLSR and Predictive-Optimized Link State Routing (P-OLSR).

Keywords : FANETs, Routing, Quality of Service, P-OLSR, Deep Learning, ETX.

ملخص

تندرج هذه المذكرة ضمن مجال الشبكات اللاسلكية المتنقلة، وبشكل خاص شبكات الطائرات بدون طيار المترابطة، حيث نقترح مقاربة ذكية تهدف إلى تعزيز استقرار الاتصالات بين هذه الطائرات. يتمثل الهدف الرئيسي في تصميم، نمذجة، وتقييم بروتوكول توجيه تنبؤي قادر على التمييز بين الروابط المستقرة وغير المستقرة لضمان جودة خدمة أفضل. بدأ العمل بدراسة معمقة لهندسة شبكات الطائرات بدون طيار، تلتها مراجعة نقدية للبروتوكولات التقليدية الخاصة بالتوجيه وأساليب إدارة جودة الخدمة. بناءً على هذه الدراسة، تم اقتراح تحسين على بروتوكول P-OLSR عبر دمجه بنموذج يعتمد على تقنيات Machine Learning وDeep Learning للتنبؤ بمؤشر ETX (عدد الإرسال المتوقع)، والذي يُعتبر مؤشراً مهماً لجودة الروابط. تم تدريب النموذج المقترح والتحقق من دقته باستخدام مجموعة بيانات تمثل خصائص الروابط في بيئة شبكات الطائرات بدون طيار. كما أجرينا سلسلة من المحاكاة لتقييم أداء البروتوكول المقترح من حيث الاستقرار، معدل الإرسال، الكمون، وموثوقية التوجيه. أظهرت نتائج اختبار النموذج دقة تبلغ 98.58%، ودقة تصنيف بنسبة 98.37%، واسترجاع بنسبة 98.38%، مع خسارة منخفضة قدرها 0.0612، مما يدل على قدرة ممتازة على التعميم. كما بينت نتائج المحاكاة تفوق البروتوكول المقترح، حيث بلغ معدل الاستقرار 98.8%، وانخفض معدل الفقد إلى 3%، وحقق معدل الإرسال المفيد نسبة 97%، ووصلت المرونة إلى 6%، في حين بلغت نسبة الروابط النشطة 92%. وبذلك، يتفوق النموذج المقترح على بروتوكولي OLSR وP-OLSR.

الكلمات المفتاحية: شبكات الطائرات بدون طيار ، التوجيه، جودة الخدمة، بروتوكولP-OLSR، التعلم العميق، عدد الإرسال المتوقع .

Remerciements

Il est toujours difficile de trouver les mots justes pour exprimer toute la gratitude que nous ressentons en cette étape marquante de notre parcours académique. Ce mémoire représente l'aboutissement d'années d'efforts, de défis relevés, d'enseignements reçus et de rencontres enrichissantes.

Avant tout, nous remercions Allah le Tout-Puissant, pour Sa guidance, Sa miséricorde et la force qu'Il nous a accordée tout au long de ce chemin.

Nous exprimons notre profonde reconnaissance à Dr. Senouci Oussama, notre encadrant, pour sa disponibilité, sa bienveillance, la qualité de son accompagnement et ses conseils précieux. Sa rigueur scientifique et son soutien constant ont été déterminants pour la réussite de ce travail.

Nos remerciements vont également aux membres du jury, pour avoir accepté de lire et d'évaluer notre mémoire avec attention.

Nous tenons à exprimer toute notre gratitude à l'ensemble des enseignants et du personnel administratif de l'Université Mohamed El Bachir El Ibrahimi, en particulier à la Faculté des Mathématiques et d'Informatique, pour la qualité de l'encadrement, leur implication et leur dévouement tout au long de notre formation.

Nous adressons un immense merci à nos parents et à nos familles pour leur amour, leur patience, leurs sacrifices et leur soutien indéfectible. Leur présence à nos côtés a toujours été notre plus grande force.

Enfin, nos pensées reconnaissantes vont à nos collègues, ami(e)s et toutes les personnes qui, de près ou de loin, ont contribué à l'élaboration de ce mémoire par leurs encouragements, leurs conseils et leur aide précieuse.

Ce mémoire est le fruit d'un effort collectif, porté par la confiance, le savoir transmis, et le soutien constant de toutes les personnes qui nous ont accompagnés.

Merci du fond du cœur.

Dédicace

Je dédie ce travail :

À mon père,

pour sa force silencieuse, sa présence rassurante et ses sacrifices que les mots ne peuvent traduire ;

À ma mère,

pour son amour infini, ses prières constantes, et sa lumière dans chaque étape de ma vie ;

À mon frère Karim et à ma sœur Rima,

pour leur affection, leur soutien et les liens précieux qui nous unissent ;

À ma tante Djamila,

ma deuxième mère, pour sa tendresse, sa bienveillance, et sa place unique dans mon cœur ;

À mes amies Rim, Soundous, Asma ,Achoik et Chaïma,

pour les souvenirs partagés, le rire, et la chaleur de leur amitié dans les moments difficiles ;

À mon binôme Dounia,

Alliée fidèle, amie précieuse, sœur de cœur. Merci pour ton soutien indéfectible tout au long du chemin ;

Enfin, à tous ceux qui ont cru en moi,

merci d'avoir façonné la personne que je suis devenue.

⋆ ***Houda*** ⋆

Dédicace

À celui dont je porte le nom avec fierté,
À celui que Dieu a couronné de prestance et de sagesse,
À celui qui a enlevé les épines de mon chemin et y a semé le confort à la place,Le dos de mon père ne s'est pas courbé sous le poids des fardeaux,Mais il s'est courbé pour me porter, par amour,
À mon père ;
À celle qui m'a appris les valeurs avant même que je ne les comprenne,
À ce pont qui m'élève vers le paradis,
À la main invisible qui a écarté les obstacles de mon chemin,
À celle dont les prières portent mon nom jour et nuit,
Ma mère, mon amour et mon inspiration ;
À ceux dont la présence est une bénédiction divine,
À ma source de force, mon sol ferme, et le mur solide de mon cœur,
Mes frères et sœurs : Mohamed Al-Sadiq, Basma, Khadidja, Safaa ;
À toute ma famille ;
À mes chers amis et camarades,
Merci d'avoir partagé les débuts, les efforts et les rêves, et d'avoir été présents avec amitié et soutien tout au long du chemin ;
À mon binôme Assala Houda,
Merci d'avoir été bien plus qu'une partenaire de travail,
une amie, un vrai soutien, une lumière discrète mais constante ;
Après un long effort, j'ai trouvé dans mes yeux l'arrivée tant attendue, et l'horizon ma offert paix, joie et gratitude ; je dédie les fruits de ces années d'études.

⋆ ***Dounia*** ⋆

Table des matières

Table des figures

Liste des tableaux

Abréviations

FANETs :Flying Ad-Hoc Networks

MANETs :Mobile Ad Hoc Networks

VANETs :Vehicular Ad-Hoc Networks

UAVs :Unmanned Aerial Vehicles

QoS : Quality of Service

ML :Machine Learning

DL :Deep Learning

MPR : MultiPoint Relay

P-OLSR :Predictive Optimized Link State Routing

ETX :Expected Transmission Count

GRU : Gated Recurrent Unit

TC : Topology Control

CNN :Convolutional Neural Network

ANN :Artificial Neural Network

DNN :Deep Neural Network

RNN : Recurrent Neural Network

NS-3 :Network Simulator 3

Introduction Générale

Les réseaux sans fil ad-hoc connaissent une évolution notable avec l'émergence des Flying Ad Hoc Networks (FANETs), une technologie innovante qui repose sur l'interconnexion de drones autonomes pour des missions complexes et critiques. Ce mémoire s'inscrit dans cette dynamique, en explorant les défis liés au routage et à la qualité de service dans les FANETs, et en proposant une solution intelligente basée sur les réseaux de neurones récurrents.

Problématique & Motivations

Les réseaux de drones, appelés FANETs , représentent une innovation technologique remarquable dans le domaine des communications sans fil. Constitués de drones interconnectés, ces réseaux sont capables d'assurer des missions variées sans infrastructure fixe. Leur mobilité tridimensionnelle, leur rapidité de déploiement et leur adaptabilité à différents environnements les rendent particulièrement adaptés à des scénarios tels que la surveillance, la gestion de catastrophes, ou encore les opérations militaires.

Cependant, cette flexibilité s'accompagne de plusieurs défis majeurs. Les changements fréquents de topologie, les limitations énergétiques des UAVs, la volatilité des connexions et les exigences croissantes en matière de qualité de service rendent les protocoles de routage traditionnels peu adaptés à ces environnements. En réponse à ces limitations, de nombreuses études se sont orientées vers l'intégration de techniques d'intelligence artificielle , permettant d'améliorer les performances du réseau grâce à des mécanismes de prédiction et d'adaptation intelligents.

Objectifs & Contributions

Ce travail a pour objectif principal d'améliorer la performance des protocoles de routage et la qualité de service (QoS) dans les FANETs, en proposant une solution intelligente capable de mieux s'adapter à la dynamique de ces environnements. Pour cela, nous avons choisi d'intégrer des techniques d'intelligence artificielle, notamment les réseaux de neurones récurrents d'un modèle Gated Recurrent Unit(GRU), afin de prédire la stabilité des liens de communication et ainsi orienter efficacement les décisions de routage.

Plus précisément, les objectifs de notre travail se déclinent comme suit :

1. **Analyser les limitations des protocoles existants** :

   Identifier les faiblesses des solutions de routage classiques (comme OLSR et P-OLSR) face aux défis posés par la mobilité rapide, la topologie dynamique et les contraintes énergétiques des drones.

2. **Étudier la faisabilité de l'intégration d'un modèle d'intelligence artificielle** :

   Explorer l'usage des réseaux neuronaux dans un contexte de prédiction de la qualité des liens à travers la métrique ETX, afin d'améliorer la stabilité des chemins de communication.

3. **Développer un protocole de routage intelligent** :

   Proposer une version améliorée du protocole P-OLSR, nommé P-OLSR-GRU, capable d'anticiper l'état futur des liens en se basant sur des séquences temporelles de métriques de communication extraites du réseau.

4. **Implémenter, simuler et évaluer la solution proposée** :

   Concevoir une plateforme de simulation intégrant le modèle GRU, réaliser des expérimentations sur des jeux de données réalistes, et comparer les performances du protocole proposé à celles des protocoles existants selon des indicateurs clés (stabilité, perte de paquets, débit, latence, nombre de liens actifs, etc.).

Ainsi, les contributions majeures de ce mémoire peuvent être résumées comme suit :

1. Une étude approfondie de l'environnement FANET, incluant les caractéristiques

matérielles et logicielles des drones, les scénarios d'application, et les enjeux techniques liés au routage.

2. Une analyse critique des solutions de routage et de gestion de la QoS existantes, accompagnée d'un recensement des approches basées sur l'intelligence artificielle appliquées aux réseaux mobiles.
3. La conception d'un protocole prédictif de routage hybride, intégrant le modèle GRU au sein du mécanisme de calcul de la métrique ETX, pour une meilleure sélection des liens stables.
4. Une implémentation complète et des simulations rigoureuses validant l'efficacité de la solution P-OLSR-GRU, avec des résultats supérieurs aux protocoles OLSR et P-OLSR en termes de performance globale du réseau.

Organisation du mémoire

Le mémoire est organisé en quatre chapitres distincts, chacun traitant d'un aspect spécifique de notre travail :

— **Chapitre 1 : Généralités sur les Réseaux FANETs**

Ce chapitre offre des généralités sur le contexte de notre travail, présente les fondements des FANETs, décrivant leur architecture matérielle et logicielle, les spécificités des drone. De plus, nous explorons leurs domaines d'application dans les FANETs soulignant leur importance croissante dans des secteurs variés comme l'agriculture , Surveillance, Reconnaissance.

— **Chapitre 2 : État d'art**

Dans ce chapitre , Nous présentons une étude approfondie de l'état de l'art des de routage et qualité de service dans les FANETs, Nous avons élaboré une nouvelle taxonomie de classification en nous appuyant sur différentes mesures pertinentes por les réseaux de drones. En nous appuyant sur cette taxonomie, nous classifions et analysons un ensemble d'approches de routage récemment proposées dans la littérature. L'objectif est d'identifier les limitations des méthodes existantes et de proposer de nouvelles stratégies de routage prédictif intelligentes, mieux adaptées aux caractéristiques dynamiques et aux exigences spécifiques des FANETs.

— **Chapitre 3 : Protocole de Routage Predictive OLSR**

Ce chapitre présente notre contribution principale : une amélioration du protocole de routage P-OLSR pour les FANETs. Nous y décrivons les métriques utilisées, notamment l'ETX, et les mécanismes d'optimisation du routage basés sur la mobilité. Le protocole est évalué à l'aide d'un banc d'essai UAV. Nous exposons ses avantages en termes de stabilité et de performance, ainsi que ses limites, qui motivent l'intégration d'un modèle GRU dans le but d'optimiser le calcul de la métrique ETX et abordée dans le chapitre suivant.

— **Chapitre 4 : Protocole de Routage Proposé P-OLSR-GRU**

Dans ce chapitre, nous présentons les étapes de préparation des données, la conception et l'entraînement du modèle GRU, ainsi que les résultats de simulation permettant de valider la performance de l'approche proposée.

Chapitre 1

Généralités sur les Réseaux FANETs

1.1 Introduction

Les réseaux ad-hoc de drones, appelés Flying Ad-Hoc Networks (FANETs), constituent une catégorie spécifique de réseaux sans fil dans laquelle plusieurs véhicules aériens sans pilote (UAVs) coopèrent de manière autonome pour accomplir diverses missions. Ces réseaux suscitent un intérêt croissant en raison de leurs nombreuses applications dans des secteurs tels que la surveillance, l'agriculture, la gestion des catastrophes et les opérations militaires. Dans un FANET, chaque drone assure non seulement sa propre mission, mais agit également comme un nœud de communication, facilitant l'échange d'informations entre les autres drones, y compris sur de longues distances. Cette interconnexion dynamique repose sur une architecture sophistiquée, combinant des éléments matériels performants et des logiciels avancés.

Ce chapitre présente d'abord l'architecture typique d'un drone au sein d'un FANET, en détaillant ses composants matériels, ses spécifications techniques, ainsi que son infrastructure logicielle et protocolaire. Il explore ensuite les principaux défis techniques et les contraintes de performance propres aux FANETs. Enfin, il examine les multiples domaines d'application de ces réseaux, en illustrant leur utilité dans des missions telles que la recherche et le sauvetage, la surveillance des incendies de forêt, la gestion du trafic routier, ainsi que la reconnaissance et le suivi en milieu agricole.

1.2 Architecture Matérielle d'un Drone dans les FANETs

L'architecture matérielle d'un drone évoluant dans un FANET résulte d'une intégration sophistiquée de composants physiques et logiciels, permettant au drone d'accomplir ses missions spécifiques tout en maintenant une communication efficace avec ses homologues. En raison de leur mobilité constante, ces drones doivent être équipés de systèmes robustes garantissant leur stabilité en vol, leur connectivité, ainsi que leur autonomie énergétique. Cette section examine en détail les principaux éléments constituant un drone dans un environnement FANET, en mettant en avant leurs caractéristiques techniques ainsi que leur organisation protocolaire.

1.2.1 Composants Matériels d'un Drone

Les avancées récentes dans les domaines de la communication, des capteurs et des systèmes électroniques ont considérablement facilité la fabrication d'UAV. Ces appareils, capables de naviguer de manière autonome sans intervention humaine, intègrent divers équipements technologiques [1]. La Figure 1.1 illustre l'architecture matérielle type d'un drone.

FIGURE 1.1 – Composants Matériels d'un Drone.

Les principaux composants matériels sont détaillés ci-dessous :

1. **Système d'alimentation :** Assure l'apport énergétique nécessaire aux diverses fonctions du drone, notamment le traitement des données, les communications,

ainsi que le contrôle et l'alimentation des rotors, à travers une ou plusieurs batteries.

2. **Système de contrôle :** Responsable de la gestion des opérations de vol, ce système ajuste notamment l'altitude en modulant la vitesse de rotation des rotors selon les instructions de navigation.
3. **Moteur :** Élément clé du quadcoptère, le moteur génère, par sa rotation, une force appliquée sur les hélices (force Fz) ainsi qu'un couple de réaction (moment Mz) [1].
4. **Module de communication :** Permet l'échange d'informations avec les autres drones et les stations de contrôle au sol, via des technologies de transmission telles que les signaux radio ou WiFi.
5. **Caméra :** Généralement fixée sous le drone, elle assure la capture d'images ou de vidéos en offrant un champ de vision optimal [2].
6. **Hélices :** Essentielles au vol, les hélices, disposées à l'extrémité des bras du drone, assurent la portance et la manœuvrabilité. Leur configuration dépend de la structure du drone, pouvant adopter une forme en X ou en H.

1.2.2 Caractéristiques Techniques d'un Drone

Les drones se distinguent des autres plateformes aériennes classiques (hélicoptères, avions, navires, etc.) par leur polyvalence, leur maniabilité et leur capacité à opérer dans des environnements variés. Dans le contexte des FANETs, les caractéristiques techniques d'un drone sont déterminantes pour assurer la fiabilité, l'efficacité et l'autonomie du réseau. Ces spécifications conditionnent non seulement les performances individuelles du drone, mais également la robustesse globale de l'architecture réseau [2].

Les principales caractéristiques techniques requises pour un drone évoluant dans un FANET sont les suivantes :

1. **Maniabilité et précision :** Les drones doivent être capables de maintenir avec exactitude une position GPS et une altitude stable, même dans des environnements dynamiques ou soumis à des perturbations atmosphériques. Cette précision

1. Consulté le 21 octobre 2024, à partir du site : `https://sensel-measurement.fr/fr/blog/mesure-de-force-et-de-couple-sur-helice-de-drone-n79`
2. Consulté le 21 octobre 2024, à partir du site : `http://blewando.fr/elv/Promo2020/th1/pag2.html`

est cruciale pour les missions de coordination multi-drones et pour l'optimisation des communications en réseau.

2. **Sécurité et conformité légale :** Les drones et leurs opérateurs doivent être équipés de dispositifs de sécurité adaptés (systèmes de retour automatique, détecteurs d'obstacles, parachutes d'urgence, etc.) et respecter les normes réglementaires en vigueur en matière d'utilisation aérienne.
3. **Déploiement rapide :** Les systèmes de drones doivent être opérationnels en un temps réduit généralement moins de 10 minutes, ce qui est essentiel pour des interventions d'urgence, telles que les opérations de secours ou la gestion rapide de catastrophes.
4. **Efficacité économique :** Comparés aux moyens traditionnels (hélicoptères, avions légers), les drones offrent une alternative bien plus abordable, tant en termes de coûts d'acquisition que de frais d'exploitation.
5. **Captation d'images de haute qualité :** Les drones permettent la prise de photos et de vidéos exceptionnelles, offrant des perspectives inédites et des angles de vue difficilement accessibles autrement, ce qui est particulièrement précieux pour les applications de surveillance et d'inspection.
6. **Respect de l'environnement :** Fonctionnant généralement à l'électricité, les drones n'émettent pas de CO_2 pendant leur vol, ce qui en fait une solution respectueuse de l'environnement, silencieuse et adaptée à des zones écologiquement sensibles.
7. **Accès facilité aux zones difficiles :** Grâce à leur compacité et leur agilité, les drones peuvent atteindre des lieux inaccessibles ou dangereux pour l'homme, tels que des zones sinistrées, des forêts denses, ou encore des infrastructures instables.

1.2.3 Structure Logicielle et Protocolaire

La structure logicielle et protocolaire des drones évoluant dans les FANETs constitue un élément fondamental pour garantir la communication efficace, le contrôle autonome, ainsi que la fiabilité globale du réseau. Elle englobe les systèmes d'exploitation embarqués, les piles protocolaires assurant l'échange sécurisé des données, les algorithmes de contrôle pour la gestion des trajectoires, ainsi que les outils de dévelop-

pement et de simulation destinés à tester les architectures avant déploiement réel [3]. Cette structure permet de maintenir une communication stable et réactive, même dans des environnements dynamiques, fortement mobiles et sujets aux dégradations du canal de transmission.

1.2.3.1 Systèmes d'exploitation embarqués

Les systèmes d'exploitation embarqués jouent un rôle crucial dans la gestion des ressources matérielles et logicielles du drone, tout en assurant l'exécution fiable des applications critiques.

1. **Linux :** De nombreux drones utilisent des distributions Linux légères adaptées aux environnements embarqués pour la gestion des tâches en temps réel, la communication inter-processus et l'optimisation des performances système.

1.2.3.2 Piles Protocolaires

Les piles protocolaires assurent la gestion de la communication, l'acheminement des messages et la sécurisation des échanges entre les drones dans un FANET.

1. **Secure Ad hoc On-Demand Distance Vector (SAODV) :** Protocole de routage sécurisé intégrant des mécanismes d'authentification pour prévenir les attaques telles que le wormhole ou le blackhole.
2. **Low Overhead Deterministic MAC (LODMAC) :** Protocole d'accès au médium conçu spécifiquement pour les environnements fortement mobiles comme les FANETs.
3. **Token-MAC :** Protocole basé sur la gestion de jetons pour garantir un accès ordonné et équitable au canal de communication, réduisant ainsi les collisions.
4. **IEEE 802.11b/a :** Normes de communication sans fil souvent adaptées aux réseaux ad hoc de drones pour offrir différents débits de transmission en fonction des besoins.
5. **Ad-hoc Mobile UAV Protocol (AMUAV) :** Protocole conçu pour améliorer la connectivité et l'efficacité du routage entre drones mobiles.

6. **Greedy Perimeter Stateless Routing (GPSR) :** Protocole de routage géographique permettant une transmission rapide des données en exploitant uniquement la position géographique des nœuds.

1.2.3.3 Algorithmes de Contrôle

Les algorithmes de contrôle sont essentiels pour coordonner les déplacements autonomes des drones, optimiser les trajectoires et garantir la réussite des missions collaboratives.

1. **Autopilote Delair-Tech :** Solution embarquée permettant une gestion autonome du vol, de la navigation et de la stabilisation du drone en temps réel.
2. **Algorithmes de coordination multi-drone :** Ensemble d'algorithmes développés pour organiser efficacement les missions de plusieurs drones travaillant ensemble dans des environnements complexes, tels que les opérations de recherche et sauvetage ou la surveillance étendue.

1.2.3.4 Logiciels de Développement et de Simulation

Avant leur déploiement sur le terrain, les architectures matérielles et logicielles des drones sont modélisées et validées à l'aide d'outils de simulation et de développement spécialisés.

1. **Matlab Simulink et Stateflow :** Plateformes utilisées pour la modélisation, la simulation et la génération automatique de code embarqué pour les systèmes de commande des drones.
2. **Outils de vérification formelle Matlab :** Utilisés pour la validation et la certification des modèles embarqués, garantissant leur conformité aux exigences de sécurité et de robustesse des drones.

1.3 Introduction aux Réseaux FANETs

Les FANETs sont des réseaux sans fil auto-organisés composés de drones interconnectés, capables d'évoluer de manière autonome sans infrastructure fixe. Grâce à leur grande mobilité, leur flexibilité et leur capacité d'intervention rapide, ils sont adaptés

à des applications variées telles que la surveillance, les missions de recherche et sauvetage, et les opérations militaires. Cependant, les FANETs doivent faire face à plusieurs défis techniques, notamment la gestion de la mobilité, la consommation énergétique, et la fiabilité des communications, afin de garantir des performances optimales.

1.3.1 Définition des FANETs

Les FANETs représentent une extension naturelle des Mobile Ad Hoc Networks (MANETs), où les nœuds sont spécifiquement constitués d'UAVs. Ces drones, équipés de capteurs, de caméras et de systèmes de communication, échangent des données en vol sans nécessiter de support au sol ni d'infrastructure préexistante. Cette capacité d'autonomie rend les FANETs particulièrement efficaces pour des missions dans des environnements dégradés ou inaccessibles [4].

Il est important de préciser que tous les systèmes multi-UAVs ne sont pas nécessairement des FANETs : pour qu'un système soit qualifié de FANET, la communication doit être basée sur des connexions ad hoc sans dépendance à une infrastructure centrale.

1.3.2 Différences entre FANETs et autres réseaux ad hoc

Les FANETs présentent plusieurs différences clés par rapport aux MANETs et aux VANETs [5] :

1. **Mobilité des nœuds :** Contrairement aux véhicules terrestres (VANETs) ou aux dispositifs mobiles classiques, les drones évoluent à des vitesses élevées, allant de 30 à 460 km/h, ce qui rend la gestion de la mobilité plus complexe.
2. **Modèles de mobilité :** Alors que les nœuds MANETs se déplacent dans des domaines relativement fixes et les VANETs sur des routes prédéfinies, les UAVs dans un FANETs suivent des trajectoires tridimensionnelles variables, modifiées en temps réel selon les conditions environnementales ou les objectifs de mission.
3. **Densité des nœuds :** Du fait de leur environnement aérien étendu et de leur mobilité rapide, la densité des nœuds dans les FANETs est généralement plus faible que dans les réseaux terrestres.
4. **Changements topologiques :** La topologie du réseau évolue rapidement à cause des déplacements rapides et imprévisibles des drones, augmentant ainsi la

complexité de la gestion de la connectivité.

5. **Modèle de propagation radio :** Contrairement aux réseaux terrestres, où la transmission est affectée par les obstacles, les drones bénéficient souvent de liaisons en ligne de vue directe (LoS), ce qui modifie les caractéristiques de propagation radio.
6. **Consommation énergétique :** Les mini-drones sont particulièrement limités en énergie, à l'inverse des UAVs de grande taille moins sensibles à cette contrainte. Cela impacte considérablement l'autonomie opérationnelle du réseau.

1.3.3 Structure typique des FANETs

La structure d'un FANET peut varier selon la mission, mais repose principalement sur plusieurs architectures [6] :

1. **Architecture centralisée :** Un nœud principal (comme une station de base) supervise le réseau et centralise la gestion des communications.
2. **Architecture décentralisée :** Chaque drone communique directement avec ses pairs sans dépendre d'une entité centrale.
3. **Architecture hybride :** Combine les avantages des structures centralisées et décentralisées avec des points centraux partiels.
4. **Architecture hiérarchique :** Organisée en plusieurs couches, où les UAVs de niveau supérieur gèrent la coordination et les UAVs inférieurs exécutent les missions locales.
5. **Architecture basée sur des clusters :** Les drones sont regroupés en clusters, chacun supervisé par un chef de cluster assurant la gestion locale du réseau.

1.3.4 Défis Techniques des FANETs

Les FANETs doivent surmonter plusieurs défis techniques liés à leur nature dynamique et instable [7] :

1. **Sécurité :** Les risques de collisions, de dysfonctionnements technologiques, de piratages ou de mauvaises manipulations soulèvent d'importantes préoccupations de sécurité pour les UAVs opérant dans des zones urbaines denses.

2. **Limitations énergétiques :** Les drones, notamment les modèles miniatures, disposent de réserves d'énergie limitées. La faible autonomie de vol représente un défi majeur pour des missions de longue durée.
3. **Capacités de stockage et de traitement :** Les ressources embarquées étant limitées, il devient difficile d'exécuter localement des tâches complexes de traitement de données ou de stockage volumineux sans recourir à des solutions externes.

1.3.5 Contraintes de Performance

Les contraintes de performance critiques à surmonter dans les FANETs sont [8] :

1. **Latence réseau :** Une latence élevée affecte la réactivité du réseau, indispensable pour les applications en temps réel telles que le déchargement de tâches vers des serveurs de calcul en périphérie.
2. **Bande passante :** Une bande passante limitée réduit la capacité de transfert de données entre les drones, freinant les performances globales du réseau.
3. **Fiabilité de la communication :** Les liaisons instables entre drones peuvent provoquer des pertes de données ou des interruptions de service, compromettant ainsi l'efficacité des missions.
4. **Interférences radio :** Les environnements encombrés entraînent des interférences significatives, dégradant la qualité des communications et augmentant les délais de transmission.

1.4 Domaines d'Application des FANETs

Grâce à leur mobilité, leur autonomie et leur capacité de coordination en temps réel, les UAVs trouvent des applications dans une grande variété de secteurs. Ils permettent d'opérer efficacement dans des environnements difficiles ou inaccessibles, et leur impact est encore renforcé par l'intégration croissante de technologies d'intelligence artificielle (IA) pour l'automatisation des missions, l'optimisation des trajectoires et l'analyse avancée des données [1]. Cette section présente les principaux domaines d'application des FANETs, ainsi que le potentiel d'amélioration future grâce aux progrès de l'IA. La Figure 1.2 illustre ces domaines clés.

FIGURE 1.2 – Domaines d'Application des FANETs.

1.4.1 Opérations de recherche et de sauvetage

Dans les situations d'urgence, telles que les catastrophes naturelles (séismes, inondations, incendies), les FANETs offrent une capacité de déploiement rapide et flexible pour rechercher des survivants et évaluer les dégâts. Les drones peuvent accéder à des zones dangereuses ou inaccessibles pour les équipes humaines, en cartographiant les zones sinistrées et en localisant précisément les victimes grâce aux capteurs thermiques, aux caméras et à l'analyse d'images intelligente. Cette approche réduit considérablement le temps d'intervention et augmente les chances de sauvetage.

1.4.2 Surveillance des feux de forêt

Les FANETs sont particulièrement efficaces pour la surveillance et la détection précoce des incendies de forêt. Équipés de capteurs thermiques, les drones peuvent surveiller en temps réel la température ambiante, détecter les anomalies thermiques, et alerter rapidement les autorités. L'usage coordonné de plusieurs drones permet de car-

tographier l'évolution du feu sur de vastes surfaces, contribuant ainsi à une intervention plus rapide et à la limitation des dégâts.

1.4.3 Surveillance du trafic

Dans le domaine de la gestion du trafic urbain et autoroutier, les FANETs permettent de détecter les embouteillages, les accidents et les situations critiques en temps réel. Grâce à leur flexibilité, les drones peuvent s'adapter dynamiquement aux changements du réseau routier et transmettre instantanément des données aux centres de contrôle du trafic. Cette solution est non seulement économique par rapport aux infrastructures fixes (comme les caméras fixes ou capteurs embarqués), mais elle permet également de proposer des réponses adaptatives pour améliorer la sécurité des usagers.

1.4.4 Reconnaissance et surveillance sécuritaire

Les FANETs jouent un rôle clé dans les missions de reconnaissance militaire, policière et civile. Les drones patrouillent des zones stratégiques de manière automatique, collectant des images et des vidéos pour la surveillance de frontières, d'infrastructures sensibles ou d'événements publics. Les données récoltées sont transmises en temps réel vers des stations de contrôle pour analyse. Cette capacité à fournir une surveillance aérienne continue renforce la sécurité et la réactivité face aux incidents ou menaces.

1.4.5 Suivi et agriculture de précision

L'agriculture de précision repose de plus en plus sur l'utilisation des FANETs pour améliorer le rendement agricole et optimiser l'utilisation des ressources. Les drones collectent des données sur l'état de santé des cultures, la teneur en eau du sol, les besoins en fertilisants et l'étendue des maladies phytosanitaires. Grâce aux traitements d'image par IA, ces informations permettent aux agriculteurs d'adapter leurs interventions de manière ciblée, augmentant ainsi la productivité tout en réduisant les coûts et l'impact environnemental. Cette approche moderne contribue à rendre l'agriculture plus durable et performante.

1.5 Conclusion

Les FANETs constituent une évolution majeure dans le domaine des réseaux sans fil, en permettant l'interconnexion de drones autonomes opérant dans des environnements dynamiques sans nécessiter d'infrastructure fixe. Ce chapitre a présenté une vue d'ensemble complète des FANETs, en abordant leur architecture matérielle et logicielle, leurs caractéristiques techniques distinctives, ainsi que leurs principales contraintes de performance, telles que la latence réseau, la bande passante limitée et les défis de fiabilité des communications.Nous avons également mis en lumière les nombreux domaines d'application des FANETs, allant des opérations de recherche et de sauvetage jusqu'à l'agriculture de précision, en passant par la surveillance du trafic et la reconnaissance sécuritaire. Ces applications démontrent le potentiel considérable des FANETs pour transformer plusieurs secteurs critiques, notamment lorsqu'elles sont renforcées par les avancées récentes en intelligence artificielle, qui permettent d'optimiser la coordination, l'analyse des données et la prise de décision autonome.

Le prochain chapitre présente un état de l'art des protocoles de routage et des mécanismes QoS dans les FANETs, en soulignant les approches existantes pour garantir une communication fiable et efficace.

Chapitre 2

État de l'art

2.1 Introduction

Les réseaux de drones, ou FANETs , sont de plus en plus utilisés dans des domaines comme la surveillance, la gestion de catastrophes et les opérations militaires. Ces réseaux permettent aux drones de communiquer et de collaborer sans infrastructure fixe, offrant une couverture étendue et une flexibilité opérationnelle accrue. Cependant, les FANETs posent des défis spécifiques en raison de la mobilité rapide des drones, de la dynamique du réseau et des contraintes en énergie et en bande passante. Ces particularités rendent les méthodes de routage classiques et les mécanismes de qualité de service souvent inadapté .

Le routage dans les FANETs doit s'adapter aux changements fréquents de topologie, tandis que la QoS est cruciale pour assurer des communications fiables et des performances optimales, notamment pour les applications en temps réel. Récemment, L'intelligence artificielle a émergé comme une solution prometteuse pour relever ces défis, grâce à des techniques qui optimisent les routes et ajustent les ressources de manière adaptatif.Ce mémoire examine les défis et les méthodes de routage et de QoS dans les FANETs, les limites des approches traditionnelles, et le potentiel de l'IA pour améliorer ces aspects essentiels.

2.2 Routage dans les FANETs

Les FANETs, avec leur nature dynamique et l'absence d'infrastructure fixe, présentent des défis uniques pour le routage. L'efficacité du routage est cruciale pour maintenir la continuité des communications entre les drones en mouvement constant, particulièrement lors de missions critiques. Cette section examine les principaux défis liés au routage dans les FANETs ainsi que les limites des approches de routage traditionnelles dans ce contexte exigeant.

2.2.1 Défis du Routage dans les FANETs

Les FANETs,malgré leur polyvalence,leur simplicité d'installation,leur mobilité exceptionnelle et leurs coûts de fonctionnement réduits, font face à divers obstacles importants, notamment en ce qui concerne le routage [9] :

1. **Connectivité intermittente :** La mobilité élevée des UAVs rend difficile le maintien d'une communication stable entre eux. De plus, la distance entre les nœuds est souvent plus importante que dans d'autres réseaux ad hoc, ce qui complique encore davantage la connectivité.
2. **Contraintes de puissance :** Généralement,les UAVs opèrent avec des batteries de capacité restreinte,ce qui influence leur temps de vol et leurs déplacements.Même si l'accroissement de la capacité des batteries peut prolonger le temps de vol, il influence les performances au-delà d'un certain seuil du fait du rapport entre poids et énergie. Il est donc vital de gérer efficacement les batteries et le rechargement dans les FANETs.
3. **Protocole de routage :** Le routage dans les FANETs est un défi en raison de la forte mobilité des UAVs et de leurs contraintes de puissance. Bien que denombreux protocoles de routage aient été conçus pour les réseaux ad hoc, les FANETs nécessitent un protocole de routage hautement dynamique pour s'adapter aux changements rapides de la topologie.
4. **Assurance de la QoS :** Des défis liés à la qualité de service doivent également être relevés, tels que la réduction de la latence, la détermination des trajectoires de vol pour garantir un service continu, la synchronisation entre les UAVs, et la protection contre les interférences et les attaques de brouillage.

Ces défis nécessitent des solutions avancées et adaptatives, et des techniques d'intelligence artificielle,comme l'apprentissage par renforcement, sont utilisées pour permettre aux réseaux de drones de s'adapter automatiquement et d'optimiser leurs performances dans ces environnements dynamiques.

2.2.2 Méthodes de Routage Traditionnelles dans les FANETs

Les approches de routage traditionnelles dans les FANETs reposent sur des méthodes proactives, réactives et hybrides. Bien qu'efficaces dans certains réseaux, elles montrent leurs limites face à la forte mobilité des drones et aux changements rapides de topologieLes protocoles de routage dans les réseaux ad-hoc reposent principalement sur trois catégories [10] :

1. **Protocoles Proactifs :**

   Ces protocoles mettent à jour leurs tables de routage à intervalles réguliers. Les nœuds du réseau connaissent déjà les changements de routes qui permettent une transmission plus rapide. L'inconvénient majeur est que pour effectuer des mises à jour constantes, ils ont besoin d'une bande passante plus importante.Suivre quatre types de protocoles proactifs :

   (1) **Protocole Optimized Link State Routing (OLSR)**

   Le protocole OLSR est un routage proactif à état de liens, a été amélioré afin de minimiser la surcharge des messages de contrôle en utilisant des Relais Multipoints (MPR). Au lieu de transmettre tous les liens du réseau, seuls les nœuds MPR transmettent les messages de contrôle, réduisant ainsi la redondance et améliorant l'efficacité du routage. Chaque nœud identifie ses voisins grâce à des messages `'Hello'` et choisit un groupe`'MPR'` pour garantir une connectivité bidirectionnelle ainsi qu'une diffusion efficace des actualisations de topologie. Ce dispositif autorise un acheminement rapide et personnalisé pour les réseaux mobiles ad hoc [11].

   (2) **Protocole Destination-Sequenced Distance-Vector (DSDV)**

   Le protocole DSDV est un protocole de routage anticipatif fondé sur des tables, dans lequel chaque nœud garde une table de routage mise à jour pour l'ensemble des autres nœuds du réseau. Il fait appel à l'algorithme

de Bellman-Ford, qui attribue des numéros de séquence aux destinations pour garantir l'actualisation des routes et prévenir les boucles. Quand une modification de topologie se produit, des actualisations sont distribuées à l'ensemble du réseau. Bien que pratique et performant, DSDV entraîne une surcharge en raison des actualisations régulières des tables, ce qui pourrait nuire aux performances dans les réseaux à dynamique très active [11].

(3) **Protocole Directional Optimized Link State Routing Protocol (D-OLSR) :**

Le protocole D-OLSR est une version perfectionnée de l'OLSR qui utilise des antennes directionnelles afin d'améliorer le transfert de données dans les réseaux mobiles ad hoc tels que les FANETs. Il diminue la surcharge en minimisant le nombre de relais MultiPoint Relay , en choisissant uniquement les nœuds les plus distants comme relais, lorsqu'ils se situent à une distance supérieure à Dmax/2 (Dmax représentant la portée de l'antenne). Cette méthode optimise le routage en réduisant les transmissions superflues et en élargissant l'étendue des communications, ce qui la rend spécifiquement appropriée pour les réseaux à grande mobilité et à vaste étendue [12].

(4) **Protocole Global State Routing (GSR)**

Le protocole GSR constitue une méthode de routage proactif pour les réseaux sans fil ad hoc. Il fusionne les bénéfices du routage à état de liens et de l'algorithme distribué de Bellman-Ford dans le but de diminuer l'encombrement causé par la diffusion des actualisations. Au lieu d'utiliser le flooding (inondation) comme les protocoles traditionnels basés sur l'état des liens, GSR transmet régulièrement des données topologiques uniquement à ses voisins directs, ce qui réduit l'utilisation de la bande passante. Il fait ensuite appel à l'algorithme de Dijkstra pour déterminer les trajets les plus courts. GSR présente un équilibre entre exactitude et efficacité, ce qui le rend approprié pour les contextes mobiles où la topographie est souvent en mutation [11].

2. **Protocoles Réactifs :**

Appelés protocole de routage à la demande,la table de routage est mise à jour lorsqu'il y a des données à envoyer, sinon il n'est pas nécessaire de calculer une

route entre les nœuds.Deux types de messages sont échangés qui sont Route Request et Route Reply. Les principaux inconvénients de ces protocoles sont un temps de latence élevé dans la recherche d'itinéraire et une inondation excessive peut entraîner un colmatage du réseau.On peut mentionner parmi les protocoles de routage réactif suivant :

(1) **Protocole Ad-hoc On-demand Distance Vector (AODV)**

Le protocole AODV adopte un mécanisme réactif basé sur le calcul de distances entre nœuds pour établir les routes., faisant appel aux messages de requête `'RREQ'` et de réponse d'itinéraire `'RREP'` pour définir des itinéraires uniquement lorsque cela est nécessaire. Il utilise des numéros de séquence pour assurer la mise à jour constante des itinéraires et prévenir le comptage sans fin. Quand une source cherche un itinéraire, elle émet un `'RREQ'` qui est propagé par les nœuds intermédiaires, ceux-ci consignant le trajet de retour. Un nœud final ou intermédiaire répond à un `'RREP'`, ce qui permet de déterminer le chemin optimal basé sur le nombre de sauts [11].

(2) **Protocole Dynamic Source Routing (DSR)**

Ce procédé met en œuvre le routage basé sur la source, où l'itinéraire intégral est incorporé dans l'en-tête des paquets. Quand une source ne sait pas comment atteindre une destination, elle commence un processus de découverte de chemin en transmettant des messages `'RREQ'`. Ces nœuds transmettent ces demandes jusqu'à ce qu'une destination ou un nœud intermédiaire ayant un parcours valide réponde avec un `'RREP'`, renvoyant ainsi la route découverte à la source. Elle mémorise par la suite cet itinéraire pour une utilisation ultérieure, optimisant de ce fait le transfert des paquets de données [11].

3. **Protocoles Hybrides :**

Une combinaison de protocoles réactifs et proactifs. Pour surmonter la faiblesse des protocoles réactifs et proactifs, une solution hybride est proposée.En proactif, la surcharge des messages de contrôle peut être réduite en utilisant ces protocoles. La latence du processus de découverte de la première route peut être réduite dans les protocoles réactifs.il existe des catégories hybrides suivantes :

(1) **Protocole Algorithme de routage ordonné temporairement (TORA)**

TORA est un protocole de routage hybride et décentralisé élaboré pour les

réseaux mobiles fortement dynamiques. Il fait appel à un graphe orienté acyclique (DAG) pour définir et gérer diverses routes entre une origine et une cible, ce qui assure une restauration rapide en cas de défaillance de connexion. Quand une connexion tombe, TORA ne modifie que la section concernée du DAG, évitant de ce fait un besoin de redécouverte totale du chemin. Ce protocole est particulièrement performant pour les réseaux où la mobilité est élevée, et où les modifications de topologie se produisent fréquemment [12].

(2) **Protocole Protocoles de routage de zone (ZRP)**

ZRP est un protocole de routage hybride qui combine des stratégies proactives et réactives en divisant le réseau en zones. À l'intérieur de chaque zone, le routage est proactif, ce qui permet de maintenir des tables de routage à jour et de réduire le temps de latence. Pour les communications inter-zones, une approche réactive est utilisée afin de limiter la surcharge due à la diffusion des mises à jour de routage. Cette approche améliore l'efficacité en réduisant la consommation de bande passante et est particulièrement adaptée aux réseaux mobiles à grande échelle, comme les FANETs [12].

(3) **Protocole Reactive greedy reactive routing protocol (RGR)**

L'algorithme RGR est un protocole hybride élaboré pour les Réseaux Adhoc . Il associe le routage réactif et la méthode de routage géographique avide Greedy Geographic Forwarding (GGF). Au départ, une voie est définie à travers un processus réactif semblable à AODV, mais en intégrant les données de position du nœud destinataire. Si une défaillance de connexion se produit, RGR passe au routage basé sur la localisation géographique, en se servant des données de position pour diriger les paquets vers l'emplacement le plus proche. Cette méthode optimise la livraison des paquets et diminue la latence, tout en maintenant un coût additionnel minimal en ce qui concerne le trafic de contrôle [13].

2.2.3 Limites des Approches Traditionnelles dans les FANETs

Dans cette section, nous explorons les principales limites des différentes approches de routage utilisées dans les FANETs [10] :

1. **Approches proactives :**
   - Pour maintenir les tables à jour, cette méthode entraîne une surcharge importante en maintenance. Cela peut affecter les performances, en particulier dans les environnements aux ressources limitées.
   - Présente une réaction lente aux changements de topologie, ce qui peut entraîner des retards dans l'établissement de connexions fiables entre les drones.
2. **Approches réactives :**
   - Pour le routage saut par saut, chaque nœud intermédiaire doit maintenir une table de routage. Cela nécessite une gestion constante des informations de routage, ce qui peut devenir problématique en cas de changements rapides de topologie.
   - Il est essentiel de trouver le chemin à chaque demande de communication dans les protocoles réactifs. Cela conduit à une plus grande latence, notamment dans des environnements où les drones se déplacent rapidement, car il est nécessaire d'attendre la découverte du chemin avant de communiquer.
3. **Approches hybrides :**
   - Les protocoles hybrides sont confrontés à des défis pour maintenir les tables de routage dans des scénarios dynamiques. La vitesse des mouvements des drones peut compliquer la gestion efficace des voies de communication, compromettant ainsi la fiabilité du réseau.
   - Il combinent les approches proactives et réactives, ce qui peut entraîner une gestion complexe des ressources, notamment en termes de calcul et de mémoire, augmentant ainsi la surcharge dans des environnements à ressources limitées.

2.3 QoS dans les FANETs

Dans les FANETs, la QoS joue un rôle crucial en influençant la satisfaction des usagers concernant les services de communication. Dans cette situation, la qualité de service comprend plusieurs facteurs comme le temps d'exécution totale, l'utilisation excessive du réseau, la bande passante et les pertes de paquets, qui ont un impact direct sur la performance des applications. Cette partie commencera par mettre en avant la

valeur du service dans les FANETs, puis se penchera sur les difficultés particulières auxquelles ces environnements dynamiques font face. Pour conclure, nous aborderons les indicateurs de performance essentiels qui facilitent l'évaluation et la garantie d'une communication efficace et fidèle au sein des FANETs.

2.3.1 Importance de la Qos dans les FANETs

La qualité de service peut être définie comme le degré de satisfaction d'un utilisateur des services fournis par un système de communication. La QoS est définie comme la capacité d'un élément du réseau (ex : routeur, noeud ou une application) de fournir un niveau de garantie pour un acheminement des données. Les principaux paramètres de QoS à considérer dans les FANETs incluent [14] :

1. **Délai de bout en bout :** Ce paramètre mesure le temps total nécessaire à un paquet de données pour voyager de sa source jusqu'à sa destination finale. Il inclut le temps de transmission, de propagation, de traitement et les délais d'attente dans les files d'attente. Dans les FANETs, un délai trop élevé peut entraîner une désynchronisation des données, en particulier pour les applications en temps réel (ex : vidéosurveillance par drones). Il est donc essentiel de le minimiser pour garantir la réactivité du système.
2. **Surcharge du réseau (gigue) :** La gigue représente la variation du délai entre la réception de paquets consécutifs. Une forte gigue peut entraîner une dégradation de la qualité perçue pour les flux multimédias ou les données sensibles au temps. Dans les FANETs, où les routes peuvent changer fréquemment, maintenir une faible gigue est un défi majeur pour assurer une QoS constante.
3. **Bande passante :** Il s'agit de la quantité maximale de données que le réseau peut transmettre sur une période donnée. Une bande passante insuffisante peut créer des goulets d'étranglement, affectant négativement la transmission de données critiques. Dans les FANETs, où les ressources radio sont limitées et partagées entre plusieurs nœuds en mouvement, l'optimisation de la bande passante est cruciale.
4. **Pertes de paquets (taux de perte) :** Ce paramètre mesure le pourcentage de paquets de données perdus durant la transmission. Les pertes peuvent résulter

d'interférences, de collisions, ou de liens instables dus au mouvement rapide des drones. Un taux de perte élevé dégrade la fiabilité du réseau et peut compromettre la qualité des données transmises. Les protocoles de routage dans les FANETs doivent donc intégrer des mécanismes pour détecter et compenser ces pertes afin d'améliorer la QoS globale.

2.3.2 Défis de la QoS dans les FANETs

Les réseaux FANETs posent des défis complexes pour assurer une qualité de service appropriée, du fait de leurs particularités propres. On classe principalement ces défis en catégories [15] :

1. **Latence de Communication et Contraintes Énergétiques :** Dans les FANETs, la QoS est souvent dégradée par la latence de communication élevée entre les drones et les points de collecte de données par exemple,(les nœuds de brouillard ou le cloud).Cela s'explique par les grandes distances que les signaux doivent parcourir, parfois avec des interférences environnementales. Les drones ont aussi une capacité énergétique limitée, ce qui limite leur temps de vol et leur puissance de transmission. En conséquence, l'efficacité des tâches, comme le traitement Machine Learning (ML) embarqué, en souffre si les ressources énergétiques sont insuffisantes. C'est un défi bien réel, largement documenté en recherche.
2. **Déchargement des Tâches ML et Latence de Traitement :** Le déchargement des tâches ML vers un nœud de brouillard est un mécanisme crucial pour alléger la charge de calcul des drones, mais il implique un compromis délicat. Si trop de données sont envoyées pour traitement externe, cela entraîne une latence accrue en raison des délais de transmission et de réception. En revanche, traiter davantage de données directement sur le drone peut entraîner des délais d'inférence dus aux limitations des ressources embarquées. Cet équilibre est essentiel pour maintenir une QoS stable et fiable dans les FANETs, et les approches de déchargement sont donc soigneusement étudiées pour répondre à ce besoin.

2.3.3 Indicateurs de Performance du Routage et de la QoS dans les FANETs.

Pour garantir des communications fiables et efficaces dans les FANETs, il est essentiel d'évaluer les performances du routage et de la QoS. Les indicateurs de performance permettent de mesurer différents aspects de la transmission de données, assurant ainsi une qualité optimale pour diverses applications. Cette section présente les indicateurs clés pour évaluer la performance du routage et de la QoS dans les FANETs.

2.3.3.1 Indicateurs de Performance du Routage :

Les indicateurs de performance du routage sont des métriques utilisées pour évaluer l'efficacité et la fiabilité des protocoles de routage. Ces indicateurs permettent d'identifier les forces et faiblesses des protocoles dans des environnements dynamiques comme les FANETs. Voici les principaux [16] :

1. **Latence (Délai de Bout en Bout) :** La latence est un indicateur essentiel pour les FANETs, doivent répondre efficacement à des applications en temps réel ou sensibles aux délais. Elle revêt une importance particulière dans des contextes tels que : prévenir les collisions au sein d'une multitude de drones, effectuer le secours et le sauvetage en cas de catastrophe.la découverte et la maintenance des itinéraires doivent être optimisées pour réduire le délai de transmission des données entre les nœuds. Une latence réduite garantit que les informations nécessaires parviennent rapidement aux destinataires, permettant ainsi une réponse rapide et précise dans des situations critiques.
2. **Débit :** Il est défini comme le nombre total de paquets reçus avec succès par la destination.

$$\text{Débit (Mbps)} = \frac{\text{Données reçues (en bits)}}{\text{Période de transmission (en secondes)}} \tag{2.1}$$

3. **Taux de transmission utile (Goodput) :** Contrairement au débit brut, le goodput mesure la quantité de données utiles effectivement reçues, excluant les paquets de contrôle, les retransmissions ou les données corrompues. C'est un indicateur direct de l'efficacité du réseau en termes de transfert réel d'information.

4. **Énergie Résiduelle :** Dans les FANETs, où les drones fonctionnent grâce à des batteries, l'énergie restante est une mesure essentielle pour prévenir les déconnexions de liaisons et les divisions du réseau causées par des dysfonctionnements de nœuds. Garantir une énergie résiduelle importante favorise la longévité des voies routières tout en améliorant la stabilité générale du réseau. Pour maximiser l'efficacité des communications dans des contextes mobiles et restreints en ressources.
5. **Taux de liens actifs :** Ce taux représente la proportion de liens effectivement établis parmi l'ensemble des connexions possibles. Il reflète la capacité du protocole à maintenir un maillage dense, garantissant ainsi une couverture robuste et une redondance des chemins de communication.
6. **Perte de Paquets :** Cette mesure détermine le pourcentage de Paquets qui ne parviennent pas à destination en comparaison du total des Paquets expédiés.

$$\text{Perte de paquets} = \text{Paquets générés} - \text{Paquets reçus} \tag{2.2}$$

2.3.3.2 Indicateurs de Performance de la QoS :

La qualité de service est cruciale pour les applications en temps réel, où la fluidité et la fiabilité des communications sont primordiales. Les indicateurs de QoS dans les FANETs mesurent différents aspects de la transmission de données, tels que la perte de paquets, la bande passante, le délai et la gigue, pour assurer une expérience de communication de haute qualité. Voici les principaux [17] :

1. **Perte de Paquets :** La perte de paquets constitue le pourcentage d'envois de données qui échouent à destination, signalant des difficultés liées à la transmission ou de congestion du réseau.
2. **Bande Passante :** La bande passante correspond au capacité le plus important du réseau pour diffuser des données, habituellement exprimé en bits par seconde (bps).
3. **Délai de Bout en Bout :** La durée totale requise pour la transmission d'un paquet de données d'une source initiale vers une destination finale via le réseau.
4. **Gigue :** La gigue désigne la différence dans le temps nécessaire pour recevoir les

paquets d'informations, affectant ainsi la fluidité et la qualité des applications en direct telles que la voix et la vidéo.

2.4 Méthodologie de Recherche

Étant donné la variété des protocoles de routage et des approches de qualité de service proposés pour les FANETs, nous procéderons à une analyse approfondie de certaines de ces techniques. Dans le cadre de cette recherche, nous avons mené une analyse exhaustive des protocoles de routage et des approches de qualité de service dans les réseaux FANETs. La sélection des protocoles et des techniques de routage a été réalisée sur la base d'une revue de la littérature récente, en nous concentrant sur les méthodes intégrant des techniques d'intelligence artificielle.

2.4.1 Objectifs de l'Etude de Recherche

Cette recherche vise avant tout à examiner en détail les protocoles de routage et les méthodes de qualité de service suggérées aux FANETs. L'accent est mis sur les méthodes qui intègrent des méthodologies d'intelligence artificielle afin d'améliorer la performance du réseau. L'objectif de cette étude est d'acquérir une connaissance détaillée des méthodes actuelles, tout en examinant les solutions qui répondent le mieux aux exigences propres des FANET.

2.4.2 Méthodes et Outils

Dans le cadre de cette recherche, nous avons mené une analyse exhaustive des protocoles de routage et des approches de qualité de service dans les réseaux FANETs. Les sources consultées incluent des articles académiques en anglais publiés sur des plateformes reconnues telles que IEEE Xplore, SpringerLink et ScienceDirect.Nous avons analysé les synthèses des articles associés pour décrire chaque algorithme.

La Figure **2.1** illustre que la majorité des références proviennent d'IEEE Xplore (36%), suivie de ScienceDirect (11%) et SpringerLink (9%). Les (43%) restants incluent des thèses, mémoires et autres plateformes, enrichissant ainsi la diversité des sources bibliographiques.

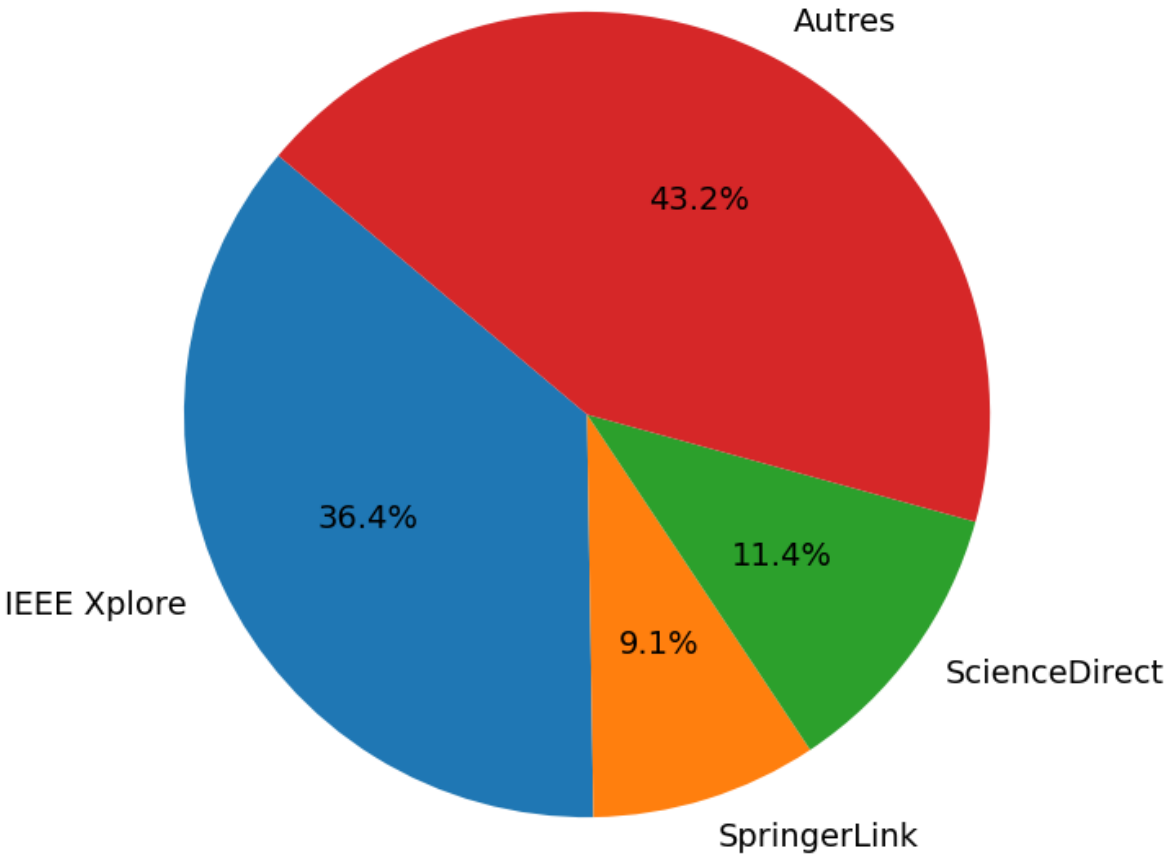

FIGURE 2.1 – Répartition des références bibliographiques par source.

2.4.3 Métriques d'Évaluation de performances

Cette recherche a été orientée par les termes de recherche "Algorithmes de Routage et de QoS dans les FANETs". De plus, nous avons fait appel aux critères ci-après pour effectuer la comparaison des algorithmes analysés :

1. **Densité** : mesure le nombre de nœuds (drones) par unité de surface dans le réseau.
2. **Scalabilité** : évalue la capacité du système à maintenir des performances acceptables à mesure que le nombre de nœuds augmente.
3. **Latence** : représente le temps nécessaire pour qu'un message soit transmis d'un nœud source à un nœud destination.
4. **Complexité** : désigne le niveau de difficulté des calculs et des opérations nécessaires pour exécuter l'algorithme.
5. **Temps de convergence** : correspond à la durée nécessaire pour qu'un algorithme trouve un chemin ou une solution après un changement dans le réseau.

6. **Interférences** : fait référence aux perturbations causées par la transmission de données d'autres nœuds, pouvant affecter la qualité de la communication.

Nous utilisons les classifications suivantes pour comparer les algorithmes : Faible, Moyenne, Élevée, afin d'assurer une évaluation claire et cohérente.

2.5 Classification des Approches de Routage et de QoS dans les FANETs

Les réseaux de drones doivent adopter des méthodes solides et appropriées. Cette partie présente une catégorisation précise des méthodes de routage et des stratégies de qualité de service employées dans les FANETs, tout en analysant leurs contraintes. En nous basant sur la classification illustrée dans la Figure 2.2.

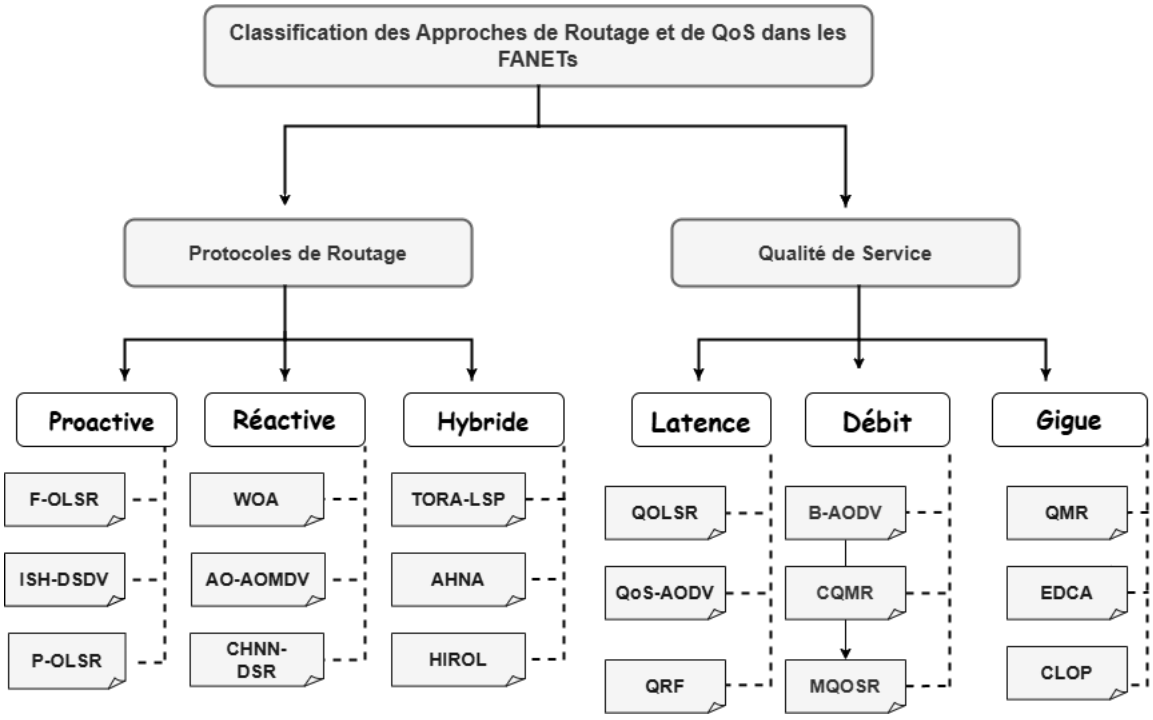

FIGURE 2.2 – Taxonomie des Approches de Routage et QoS dans les FANETs.

2.5.1 Classification des Protocoles de Routage

Cette section détaille les diverses catégories de protocoles de routage , incluant les protocoles proactifs , réactifs et hybrides. Elle commence par aborder les protocoles de routage proactifs dans les FANETs et leurs différents types, avant de s'intéresser aux

protocoles de routage réactifs et hybrides dans les FANETs, tout en mettant en exergue les spécificités propres à chaque catégorie et leur signification pour les FANETs.

2.5.1.1 Protocoles Proactives

Les protocoles de routage proactives conservent des données actualisées concernant toutes les voies possibles dans le réseau, bien qu'elles ne soient pas dès maintenant indispensables. Ces protocoles font appel à des tableaux de routage fréquemment actualisés pour assurer que chaque nœud dispose d'une vue d'ensemble du réseau. Ceci favorise une transmission de données sans latence, mais accroît la demande en bande passante et en énergie suite aux actualisations régulières. Parmi les types de protocoles proactives, on trouve des variantes améliorées :

1. **Fuzzy-Optimized Link State Routing (F-OLSR)**

   Le protocole F-OLSR est une variante optimisée du protocole OLSR, spécifiquement adaptée aux réseaux FANETs. Cette méthode se base sur la logique fuzzy (fuzzy logic) afin d'ajuster OLSR de manière plus appropriée aux exigences particulières des FANETs. F-OLSR débute par l'identification des nœuds volants adjacents, en employant une méthode innovante pour estimer la longévité des liaisons en se basant sur divers critères comme le type de lien, le sens du mouvement, la distance et la vitesse relative. Par la suite, il choisit les Multipoint relay en utilisant une méthode floue. Dans ce processus, un nœud qui possède plus d'énergie restante, plus de durée de vie de lien et plus de proximité par rapport aux autres voisins obtient un score suffisant pour se transformer en MPR. Par ailleurs, F-OLSR se sert d'un message de Topology Control (TC) révisé qui intègre deux champs supplémentaires : l'énergie du parcours et sa longévité. Par conséquent, F-OLSR établit des tableaux de routage en fonction de ces critères pour assurer une stabilité accrue des trajets [18].

2. **Heuristic Sequencing DSDV (ISH-DSDV)**

   Le protocole ISH-DSDV est adapté aux réseaux dynamiques ad hoc comme les FANETs. Il optimise la gestion des séquences de routage et diminue l'excès en restreignant les actualisations générales à des modifications notables. Il s'appuie sur trois principes essentiels : Des incréments de séquence intelligents permettent d'effectuer les actualisations uniquement lorsque nécessaire, la gestion des infor-

mations dépassées pour préserver des tables de routage exactes, et la adaptation dynamique pour réagir aux modifications rapides de topologie. Ces optimisations diminuent la latence, améliorent la bande passante et accroissent l'efficacité énergétique du réseau, garantissant par conséquent une stabilité accrue et robustesse dans des contextes hautement mobiles [19].

3. **Predictive Optimized Link State Routing (P-OLSR)**

   Le P-OLSR est une version proactive étendue du protocole OLSR, destinée aux réseaux FANETs. Il utilise les informations GPS des drones pour prévoir les modifications de la topologie. Il perfectionne le choix des chemins en tenant compte de la métrique ETX, du fait de la vitesse et de l'orientation des nœuds, garantissant ainsi une plus grande stabilité du routage. À l'instar d'OLSR, ce protocole s'appuie sur les messages Hello et TC, mais intègre des données de position pour une actualisation plus dynamique des tables de routage. Néanmoins, P-OLSR a des contraintes, y compris une dépendance marquée aux données GPS, une augmentation de la congestion du réseau due à des mises à jour fréquentes, et une vulnérabilité face aux erreurs de mesure et aux parcours imprévisibles des UAVs [20].

Tableau 3.1 présente une comparaison des protocles en fonction de plusieurs mesures clés.

Protocles	**Densité**	**Scalabilité**	**Latence**	**Complexité**	**Convergence**	**Interférences**
F-OLSR	Moyenne	Moyenne	Élevée	Moyenne	Élevée	Faible
ISH-DSDV	Moyenne	Moyenne	Élevée	Moyenne	Moyenne	Moyenne
P-OLSR	Élevée	Moyenne	Faible	Moyenne	Élevée	Moyenne

TABLE 2.1 – Comparaison des protocoles de routage proactifs.

2.5.1.2 Protocoles Réactives

Les protocoles de routage réactives opèrent en identifiant et en instaurant des routes uniquement quand cela est indispensable, c'est-à-dire quand il est nécessaire d'acheminer des informations d'une source vers une destination.À l'opposé des protocoles proactifs, ces derniers ne conservent pas constamment de tableaux de routage mis à jour. Ceci contribue à diminuer la consommation de bande passante et d'énergie

dans des contextes dynamiques comme les FANETs, où la configuration topologique varie souvent. Toutefois, la mise en place de voies sur demande peut engendrer une latence initiale plus importante au moment de la découverte des itinéraires. Plusieurs optimisations ont été suggérées pour réduire le temps de découverte et augmenter la solidité des communications dans les variantes améliorées de protocoles réactives :

1. **Whale Optimization Algorithm (WOA)**

   Le protocole WOA, qui tire son inspiration du comportement de chasse des baleines à bosse, optimise le protocole DSR en ajustant dynamiquement le choix des voies. Il modifie des paramètres comme la consommation d'énergie et la stabilité des liaisons afin de réagir aux modifications rapides de la topologie des FANETs. WOA diminue la latence et les pertes de paquets tout en garantissant une augmentation de l'efficacité énergétique, conciliant la recherche de voies innovantes et le perfectionnement des voies existantes pour assurer la solidité du réseau [21].

2. **Arithmetic Optimization AOMDV (AO-AOMDV)**

   Le AO-AOMDV est un protocole amélioré du protocole Ad hoc On-Demand Multipath Distance Vector, spécialement conçu pour satisfaire aux exigences particulières des réseaux FANETs. L'intégration de l'algorithme d'optimisation arithmétique (AO) permet d'améliorer la sélection des routes en prenant en compte plusieurs critères essentiels tels que la durée de maintien des liens, l'énergie résiduelle des nœuds et le degré de congestion des routes. Le protocole opère en explorant différentes routes entre un nœud de départ et un nœud de destination en utilisant des paquets `'RREQ'`. Après avoir identifié les chemins, ils sont évalués en se basant sur une fonction de fitness définie par les paramètres mentionnés précédemment. On utilise l'algorithme AO afin de trier les routes et choisir celles qui offrent la stabilité et l'efficacité énergétique les plus élevées, tout en évitant les routes encombrées. Le protocole est particulièrement adapté aux environnements dynamiques et exigeants en prolongeant la durée de vie du réseau, en réduisant la latence de bout en bout et en améliorant le taux de livraison des paquets [22].

3. **Dynamic Source Routing Continuous Hopfield Neural Network (CHNN-DSR)**

   Le protocole CHNN-DSR est une version améliorée du protocole DSR, développé dans le but d'améliorer le routage dans les réseaux mobiles ad hoc tels que les

FANETs. Il utilise un réseau de neurones Hopfield continu (CHNN) afin de représenter la recherche de routes comme un problème d'optimisation combinatoire. Le protocole cherche à réduire la fonction énergétique en identifiant les routes les plus stables et les plus performantes, en prenant en considération la stabilité des liens et le nombre de sauts. Dans des environnements hautement dynamiques, CHNN-DSR permet de réduire les interruptions en ajustant dynamiquement les routes en fonction de la mobilité rapide des nœuds, ce qui améliore la livraison des paquets, réduit les délais de bout en bout et augmente le débit global [23].

Le Tableau 2.2 présente une comparaison des protocles en fonction de plusieurs mesures clés.

Protocles	**Densité**	**Scalabilité**	**Latence**	**Complexité**	**Convergence**	**Interférences**
WOA	Moyenne	Moyenne	Faible	Moyenne	Élevée	Faible
AO-AOMDV	Faible	Moyenne	Moyenne	Élevée	Moyenne	Moyenne
CHNN-DSR	Faible	Moyenne	Faible	Moyenne	Élevée	Faible

TABLE 2.2 – Comparaison des protocoles de routage réactifs.

2.5.1.3 Protocoles hybrides

Les protocoles hybrides de routage allient les propriétés des protocoles proactifs et réactifs afin de proposer un équilibre entre efficacité et souplesse. Ces protocoles conservent des données de routage à proximité pour les communications fréquentes ou à courte distance, tout en mettant en place des routes à la demande pour les destinations moins fréquentes ou éloignées. Cela réduit la latence initiale liée à la découverte des itinéraires dans les protocoles réactifs, tout en évitant la surcharge de bande passante et de mémoire des protocoles proactifs. Dans des contextes en constante évolution tels que les FANETs, où la topologie est souvent modifiée, les protocoles hybrides sont particulièrement adaptés. Plusieurs améliorations ont été suggérées afin d'accroître leur capacité à être étendues, diminuer les interférences et renforcer la stabilité des communications, assurant ainsi une meilleure qualité de service dans ces réseaux.Des variantes améliorées sont disponibles parmi les différents types de protocoles hybrides :

1. **Temporally-Ordered Routing Algorithm with Localization and Selective Participation (TORA-LSP)**

Le protocole de routage hybride amélioré TORA-LSP est un algorithme de routage temporellement organisé TORA qui a été spécialement développé pour satisfaire les exigences des réseaux FANETs. Deux approches essentielles sont intégrées dans ce protocole afin d'améliorer ses performances : la localisation du réseau et la participation sélective des nœuds. La méthode de localisation du réseau restreint la construction et la maintenance des routes à une partie spécifique du réseau, en utilisant des variables de comptage de sauts pour contrôler la dispersion des paquets et réduire la surcharge de routage. De plus, la méthode de participation sélective des nœuds donne à ces derniers un statut actif ou inactif, ce qui permet de diminuer le nombre de mises à jour superflues tout en préservant l'intégrité des routes. Cette double optimisation permet au protocole d'ajuster de manière dynamique la taille et la structure de son graphe orienté acyclique (DAG) en fonction des besoins du réseau, ce qui entraîne une latence réduite, une augmentation du taux de livraison des paquets et une réduction significative des interférences. Ce protocole hybride convient spécifiquement aux milieux à forte mobilité et densité, assurant ainsi une amélioration de l'efficacité énergétique et une durée de vie prolongée du réseau [24].

2. **AntHocNet-Adapt (AHNA)**

   AntHocNet-Adapt est un nouveau protocole hybride qui repose sur l'optimisation par colonie de fourmis (ACO), spécialement conçu pour les FANETs pour faire face aux défis liés à leur dynamique rapide et à leur topologie changeante. Ce protocole associe une gestion proactive des routes dans les zones locales, où des « fourmis exploratrices » actualisent régulièrement les tables de routage en prenant en compte la qualité des liens, et une approche réactive interzones, où des « fourmis chercheuses » découvrent activement les routes les plus efficaces pour des communications fiables. En utilisant une adaptation dynamique, il adapte la taille des zones de routage en fonction de la densité des UAVs et intègre des mécanismes de pondération afin de favoriser des critères tels que la latence ou l'énergie en fonction des besoins. Dans les environnements complexes des FANETs, ce protocole hybride et amélioré assure une résilience accrue, une gestion efficace des ressources et une optimisation des performances en intégrant des fourmis "réparatrices" pour recalculer les chemins en cas de défaillance de lien [25].

3. **Hybrid Intelligent Routing with Optimized Learning (HIROL)**

   Le protocole hybride HIROL a été développé dans le but d'améliorer le routage dans les FANETs. Il associe des éléments des protocoles OLSR et DSR à des méthodes d'optimisation bio-inspirées, telles que l'algorithme Artificial Bee Colony (ABC) et des réseaux neuronaux artificiels (ANNs). Pour maintenir les tables de routage à jour dans les zones locales, HIROL adopte une approche proactive en utilisant OLSR, tandis qu'une approche réactive permet de découvrir dynamiquement les routes interzones en utilisant DSR. En imitant le comportement des abeilles, l'algorithme ABC assure l'exploration et l'optimisation des chemins de communication, tandis que les ANNs classifient l'état des liens et anticipent les variations de topologie, offrant ainsi des ajustements en temps réel. En ajustant de manière dynamique les stratégies de routage en fonction des conditions du réseau, HIROL améliore le taux de livraison des paquets, diminue la durée de livraison, réduit la charge de communication et optimise la résilience et l'efficacité énergétique du réseau [26].

Le Tableau 2.3 présente une comparaison des protocoles en fonction de plusieurs mesures clés.

Protocles	**Densité**	**Scalabilité**	**Latence**	**Complexité**	**Convergence**	**Interférences**
TORA-LSP	Moyenne	Moyenne	Moyenne	Élevée	Moyenne	Moyenne
AHNA	Faible	Moyenne	Faible	Moyenne	Élevée	Moyenne
HIROL	Faible	Bonne	Faible	Moyenne	Élevée	Moyenne

TABLE 2.3 – Comparaison des protocoles de routage hybrides.

2.5.2 Limites des Protocoles de Routage Conventionnels

Plusieurs contraintes significatives affectent les performances des protocoles de routage conventionnels dans les FANETs. Parmi les principales limitations, on distingue :

1. **Mobilité élevée et instabilité des liens** : Les UAVs se déplacent rapidement, entraînant des changements fréquents de topologie qui perturbent la stabilité des connexions et compliquent la découverte et la maintenance des routes.
2. **Latence et surcharge du réseau** : Les protocoles réactifs souffrent d'un délai important dû à la découverte des routes à la demande, tandis que les protocoles

proactifs génèrent une surcharge de contrôle excessive en maintenant à jour des tables de routage même pour des chemins inutilisés.

3. **Qualité des liaisons et interférences** : Les communications entre UAVs sont fortement influencées par la bande passante disponible, le taux d'erreur des paquets et les interférences électromagnétiques, réduisant la fiabilité du routage.
4. **Consommation énergétique élevée** : L'alimentation des UAVs étant limitée, les transmissions fréquentes de paquets de contrôle et les mises à jour des routes augmentent la consommation d'énergie, réduisant ainsi l'autonomie du réseau.
5. **Manque d'adaptabilité aux conditions dynamiques** : De nombreux protocoles ne prennent pas en compte simultanément des paramètres critiques comme la mobilité, la qualité des liens et la congestion du réseau, limitant ainsi leur efficacité dans des environnements changeants.
6. **Problème des trous de routage** : Certains protocoles basés sur la distance, comme GPSR, rencontrent des difficultés dans les zones où aucun nœud intermédiaire n'est disponible pour acheminer les paquets vers la destination, créant ainsi des interruptions de transmission.

Ces limitations montrent la nécessité de développer des protocoles de routage et de QoS optimisés pour les FANETs, intégrant des approches avancées comme l'intelligence artificielle et les algorithmes bio-inspirés, afin d'améliorer la résilience et l'efficacité du réseau.

2.5.3 Classification des Approches de QoS

La qualité de service dans les FANETs repose sur diverses métriques pour assurer des communications efficaces et adaptées aux besoins des applications. Cette section classifie les approches de QoS selon trois objectifs principaux : la minimisation de la latence, l'amélioration du débit, et la réduction de la gigue, chacune adressant des défis spécifiques liés aux performances des réseaux.

2.5.3.1 Protocles pour Minimiser la Latence

La réduction de la latence est cruciale pour les applications en temps réel des FANETs. Cette section présente des algorithmes optimisés, tels que Q-OLSR , QoS-

AODV et QRF , qui visent à choisir les routes les plus rapides pour garantir une transmission efficace des données.

1. **Quality of Service OLSR (Q-OLSR)**

   Le Q-OLSR est une optimisation du protocole de routage OLSR, conçue pour considérer divers critères liés à la qualité du service, tels que le délai et la bande passante, dans la sélection des voies routières. Ces critères sont évalués par chaque nœud pour ses voisins sans produire de trafic de contrôle additionnel, en se basant uniquement sur les messages `'HELLO'` et `'TC'` standard d'OLSR. À l'instar de OLSR, uniquement les nœuds choisis comme MultiPoint Relay communiquent des informations sur l'état du lien, facilitant ainsi la détermination des voies optimales. Le Q-OLSR ne requiert pas de changement des adresses IP actuelles, car il se consacre exclusivement à la gestion des tableaux de routage. Pour des applications en temps réel, il se sert d'algorithmes tels que Dijkstra ou ses variantes afin de réduire le délai ou optimiser la bande passante. Pour des exigences variées (soft en temps réel), il allie les contraintes liées au délai et à la bande passante, tout en cherchant des voies minimales de sauts [27].

2. **Quality of Service-AODV (QoS-AODV)**

   Le protocole Quality of Service Ad-hoc On-Demand Distance Vector , lorsqu'il est appliqué aux FANETs, garantit des critères de QoS, tels que la latence et la bande passante, dans des environnements où les drones sont en mouvement constant. Grâce à des extensions ajoutées aux messages de découverte `'RREQ'` et de réponse de route `'RREP'`, le protocole ajuste les décisions de routage en fonction des exigences de latence maximale et de bande passante minimale, assurant que seules les routes qui respectent ces contraintes sont utilisées. Si une route ne peut plus garantir ces critères, des messages `'QoS-LOST'` sont envoyés aux sources pour signaler la dégradation de la QoS, incitant les nœuds sources à lancer une nouvelle recherche de route. Cette gestion dynamique de la connectivité, combinée à des mécanismes de `'feedback MAC'` et de messages `'HELLO'`, permet de maintenir la performance du réseau en réagissant rapidement aux changements de topologie. Ainsi, QoS-AODV assure une performance optimale pour les applications sensibles au temps, tout en s'adaptant à la mobilité rapide des drones et aux conditions fluctuantes du réseau dans les FANETs.

3. **Q-learning-based Routing with Filtering (QRF)**

   Le protocole QRF est une méthode de routage pour les FANET, basée sur le Q-learning et un filtrage intelligent pour optimiser la qualité de service . L'analyse de métriques telles que la durée de connexion, l'énergie résiduelle et la qualité des liens lui permet de sélectionner les routes de manière dynamique, tout en ajustant ses paramètres d'apprentissage en fonction des conditions réseau. QRF permet une réduction de 58 de la latence par rapport à d'autres protocoles, une amélioration de la fiabilité des communications et une équilibration de la consommation énergétique, ce qui en fait une solution idéale pour les environnements mobiles et dynamiques [28].

Tableau 2.4 présente une comparaison des protocoles en fonction de plusieurs mesures clés.

Protocoles	**Densité**	**Scalabilité**	**Latence**	**Complexité**	**convergence**	**Énergie consommée**
Q-OLSR	Moyenne	Moyenne	Moyenne	Moyenne	Élevée	Moyenne
QoS-AODV	Faible	Moyenne	Faible	Moyenne	Moyenne	Moyenne
QRF	Faible	Moyenne	faible	Élevée	Élevée	Moyenne

TABLE 2.4 – Comparaison des protocoles pour Minimiser la Latence en termes de QoS

2.5.3.2 Protocoles pour Améliorer le Débit (Bande Passante)

L'amélioration du débit est cruciale pour maximiser la capacité de transmission des données dans les réseaux à forte demande. Ici, nous explorons des algorithmes qui allouent efficacement les ressources pour augmenter la bande passante disponible et réduire les goulets d'étranglement.

1. **Bandwidth-Aware AODV (B-AODV)**

   Le protocole B-AODV est une extension du protocole AODV, conçue pour garantir des routes conscientes de la bande passante dans les réseaux ad hoc basés sur TDMA. Il utilise un algorithme distribué pour allouer des créneaux horaires le long des routes tout en respectant les contraintes de QoS . Chaque lien sur la route doit disposer de suffisamment de créneaux pour répondre aux besoins de bande passante sans interférences `'intra-flux'` ou `'inter-flux'`. Lors de la découverte de routes, des paquets enrichis `'RREQ'` et des réponses `'RREP'` assurent

la réservation des créneaux nécessaires. Les simulations montrent que B-AODV améliore la livraison des paquets et réduit les délais sous des charges de trafic élevées [29].

2. **Cross-layer QoS-aware Multi-Rate Routing (C-QMR)**

   Le mécanisme CQMR) est élaboré pour les réseaux ad hoc à plusieurs taux. Il fait appel à une méthode intercouches pour tirer parti des interactions entre les couches réseau, MAC et physique. Le protocole utilise des indicateurs tels que la charge des files d'attente, la qualité des liaisons et le taux de transmission afin de choisir les itinéraires optimaux, de minimiser l'encombrement et d'améliorer la QoS . CQMR ajuste de manière dynamique les taux de transmission selon les conditions du réseau, priorisant les chemins avec une faible congestion et un haut débit afin d'optimiser le rendement et la latence [30].

3. **Multipath QoS Routing Protocol (M-QOSR)**

   Le protocole MQoSR est élaboré pour les réseaux sans fil dans le but d'assurer une QoS optimale. Il adopte une stratégie à multiples chemins pour établir plusieurs routes en même temps, ce qui renforce la redondance, la fiabilité et l'équilibrage de charge. Les choix de routage sont basés sur des critères comme le délai de latence, la fiabilité des connexions, la distance et l'énergie restante des nœuds. L'objectif de ce protocole est de minimiser les temps d'attente, d'optimiser la fiabilité des transmissions et d'allonger la longévité des nœuds, garantissant par conséquent une communication solide dans des environnements en constante évolution [31, 32].

Le Tableau 2.5 présente une comparaison des protocoles en fonction de plusieurs mesures clés.

Protocoles	**Densité**	**Scalabilité**	**Latence**	**Complexité**	**convergence**	**Énergie consommée**
B-AODV	Moyenne	Moyenne	Faible	Moyenne	Moyenne	Moyenne
CQMR	Moyenne	Moyenne	Faible	Moyenne	Élevée	Moyenne
MQoSR	Faible	Moyenne	Faible	Moyenne	Moyenne	Moyenne

TABLE 2.5 – Comparaison des protocoles selon les critères de QoS pour la minimisation du débit.

2.5.3.3 Protocoles pour Réduire la Gigue

La gigue, représentant les variations de délai entre les paquets, peut dégrader la qualité des applications sensibles au temps. Cette sous-section analyse des protocoles visant à réduire ces variations et à assurer une communication fluide, en particulier pour les flux multimédias.

1. **Queue Management and Regulation (QMR)**

   QMR désigne un ensemble de mécanismes visant à gérer efficacement les files d'attente dans les routeurs pour optimiser les performances réseau et prévenir les problèmes de congestion. Ses objectifs principaux sont la réduction de la latence, la limitation de la gigue, la prévention des pertes massives de paquets et l'évitement du `'lock-out'` où certaines connexions monopolisent les ressources. La gestion des files d'attente peut être passive, comme le tail drop où les paquets sont abandonnés lorsque la file est pleine, ou active, comme l'active queue management (AQM) qui abandonne les paquets de manière proactive pour éviter la saturation. Un exemple d'algorithme utilisé dans QMR est le Random Early Detection (RED), qui permet d'abandonner les paquets en fonction de la taille moyenne de la file, prévenant ainsi la congestion. Les avantages de QMR incluent l'amélioration des performances réseau, l'optimisation de QoS et l'équité dans le partage des ressources. Il est souvent combiné à des algorithmes de planification pour garantir des performances optimales [33].

2. **Enhanced Distributed Channel Access (EDCA)**

   EDCA est un protocole d'accès au canal qui s'ajoute au contrôle d'accès au support (MAC) IEEE 802.11. Le comité de travail IEEE 802.11 a validé ce document pour satisfaire les demandes de QoS des applications de données et en temps réel. L'EDCA gère la différenciation des services en assignant des priorités à diverses catégories de trafic, y compris le background (BK), le meilleur effort (BE), la vidéo (VI) et la voix (VO). Cette distinction se fait grâce à des paramètres d'accès particuliers tels que l'espace de contention inter-frame (AIFS) et la fenêtre de contention (CW), qui favorisent le traitement prioritaire des flux de trafic hautement prioritaires. Toutefois, en dépit de cette distinction, l'EDCA ne garantit pas une QoS stricte, car une configuration incorrecte des paramètres d'accès au médium peut provoquer des performances non optimales, y compris une hausse

des collisions et un phénomène d'engorgement du trafic à faible priorité [34].

3. **Cross-Layer Optimization Protocol (CLOP)**

   CLOP est une méthode de communication développée pour augmenter la QoS dans les FANETs, en améliorant l'interaction entre les diverses couches du modèle OSI. Ce protocole utilise une coordination dynamique entre les couches physique (PHY), MAC et réseau afin d'adapter les paramètres de transmission et d'optimiser la régulation du trafic aérien. CLOP cherche à minimiser le gigue et la latence en appliquant des méthodes efficaces comme la probabilité de collision au niveau MAC, ce qui favorise une meilleure orientation des paquets. En adoptant une approche d'apprentissage adaptatif, il modifie de manière dynamique le choix des routes de transmission en fonction des conditions du réseau, améliorant ainsi la transmission des paquets et diminuant par conséquent la congestion. Cette méthode rend CLOP particulièrement performant pour les communications critiques en temps réel et les applications requérant une latence minimale, comme la surveillance aérienne et les missions autonomes des drones [35, 36].

Le Tableau 2.6 présente une comparaison des protocoles en fonction de plusieurs mesures clés.

Protocoles	**Densité**	**Scalabilité**	**Latence**	**Complexité**	**convergence**	**Énergie consommée**
QMR	Moyenne	Moyenne	Moyenne	Moyenne	Élevée	Moyenne
EDCA	Moyenne	Moyenne	Moyenne	Moyenne	Moyenne	Moyenne
CLOP	Faible	Moyenne	Faible	Élevée	Élevée	Réduite

TABLE 2.6 – Comparaison des protocoles selon les critères de QoS pour l'optimisation de la gigue.

2.5.4 Limites des Protocoles basés sur la QoS

Les approches de QoS dans les FANETs apportent des améliorations significatives en matière de latence, de débit et de réduction de la gigue, mais elles présentent également plusieurs limites :

1. **Instabilité et dynamique du réseau** : La nature hautement dynamique des FANETs complique le maintien d'une connectivité stable, rendant les algorithmes de routage sensibles aux changements fréquents de topologie. Cette instabilité

entraîne une latence accrue et une augmentation du temps de convergence des protocoles.

2. **Consommation énergétique élevée** : De nombreux mécanismes QoS nécessitent des calculs intensifs et des échanges de messages fréquents, ce qui réduit l'autonomie des drones et limite la durée des missions.
3. **Encombrement du spectre et interférences** : L'accroissement du nombre de nœuds dans les FANETs entraîne une congestion du réseau et augmente les interférences électromagnétiques, dégradant ainsi la qualité des communications, en particulier dans des environnements densément peuplés.
4. **Complexité algorithmique élevée** : Bien que certains protocoles adoptent une approche multi-couche pour optimiser la QoS, cette complexité algorithmique alourdit le traitement des données et limite leur déploiement dans des systèmes aux ressources limitées.
5. **Absence de standardisation** : L'absence de normes communes entrave l'interopérabilité entre différents protocoles et équipements, compliquant ainsi leur adoption et leur intégration dans des architectures existantes.

Ainsi, malgré les avancées, les approches actuelles nécessitent encore des améliorations en termes d'adaptabilité, d'optimisation énergétique et de flexibilité afin de répondre pleinement aux exigences des FANETs dans des scénarios variés et évolutifs.

2.6 Intelligence Artificielle pour l'Amélioration du Routage et de la QoS dans les FANETs

L'intelligence artificielle (IA) joue un rôle crucial dans l'amélioration des FANETs. Elle permet d'optimiser la prise de décision, de s'adapter aux changements de topologie et de gérer efficacement les ressources dans des applications critiques. Les techniques de routage basées sur l'IA, telles que l'apprentissage par renforcement et les réseaux neuronaux, augmentent l'efficacité du réseau en identifiant des chemins optimaux pour la transmission des données. Par ailleurs, l'IA contribue à l'optimisation de la qualité de service en ajustant des paramètres clés comme la latence et la consommation d'énergie, assurant ainsi des performances fiables dans des environnements dynamiques. Cette

section explore en détail les applications de l'IA dans les FANETs, les techniques de routage qui en découlent, ainsi que leur impact sur l'optimisation de la QoS.

2.6.1 Applications de l'IA dans les FANETs

La technologie des drones a connu une évolution, avec des modèles contemporains embarquant des cartes de calcul, des Graphics Processing Unit(GPU)/Tensor Processing Unit(TPU) et des cartes Field Programmable Gate Array (FPGA), ce qui permet l'utilisation d'algorithmes d'apprentissage profond Deep Learning (DL) pour des applications d'intelligence artificielle .Cette section recense les méthodes d'intelligence artificielle utilisées dans les réseaux sans fil activés par des drones. Nous présentons une synthèse des diverses approches d'apprentissage, telles que l'apprentissage supervisé et non supervisé. Les chercheurs ont mis en œuvre des solutions basées sur l'IA dans plusieurs domaines, notamment le positionnement et la détection, l'estimation des canaux, les applications de réalité virtuelle, l'imagerie, le contrôle autonome des chemins, ainsi que la planification et l'allocation des ressources, et la sécurité. Ces applications sont illustrées par la figure 2.3 [37].

2.6.2 Techniques de Routage Basées sur l'IA

L'intelligence artificielle cherche à créer des machines qui reproduisent le comportement humain et ses aptitudes cognitives, comme la prise de décision et la résolution de problèmes. Divers secteurs, comme la vision par ordinateur, les systèmes de santé, les véhicules automobiles, etc., ont largement exploité l'IA.Dans cette partie, nous examinons les protocoles de routage autorisés par l'intelligence artificielle. Ces derniers exploitent la capacité d'apprentissage des algorithmes de Machine Learning (ML) pour choisir le meilleur itinéraire en se basant sur une compréhension plus exacte de la topologie du réseau, de la condition des canaux, du comportement des usagers et de la mobilité du trafic, entre autres. Ces algorithmes relient les domaines de recherche en réseau et en IA afin de mettre en place des réseaux contemporains, notamment pour les réseaux UAV dynamiques. On peut considérer ces algorithmes comme le sommet de la technologie. Voici une sélection suffisamment détaillée d'algorithmes fondés sur l'IA [37] :

1. **Protocoles de routage prédictifs basés sur la topologie :**

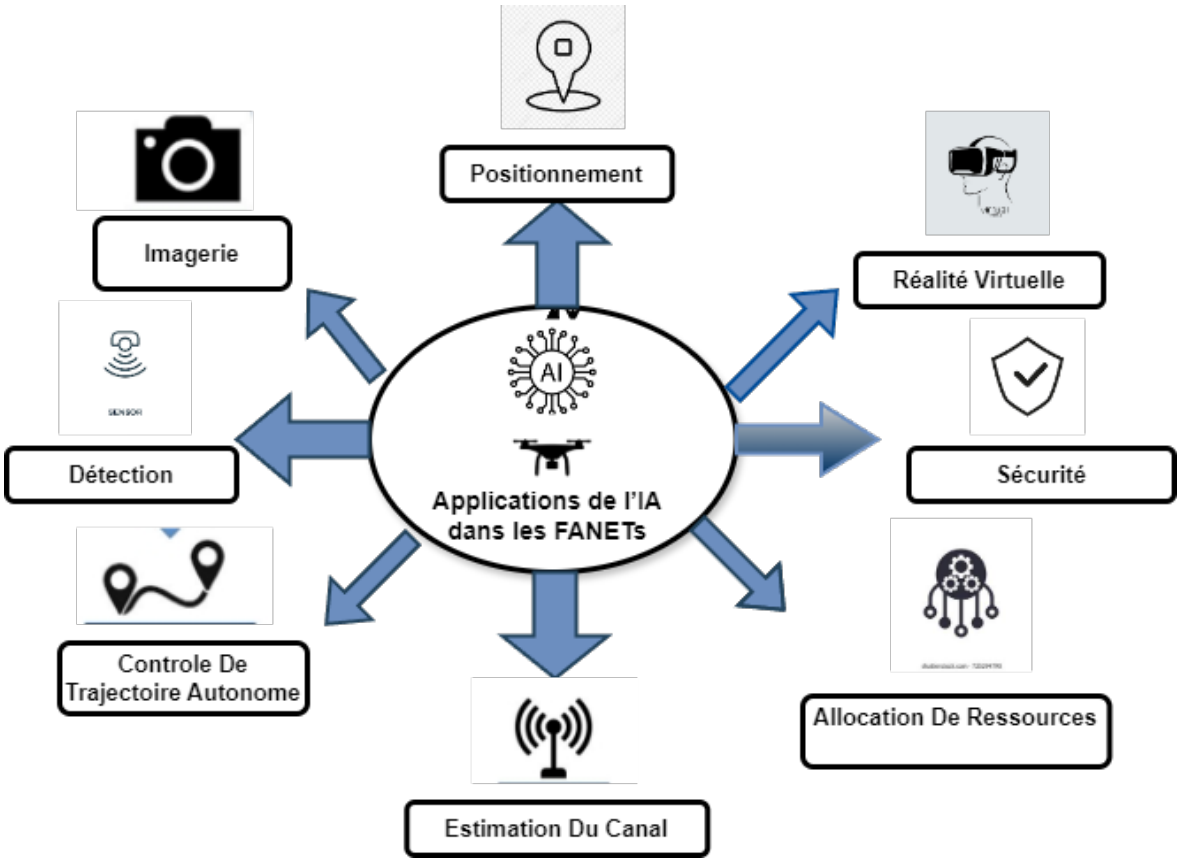

FIGURE 2.3 – Applications de l'IA dans les FANETs.

La principale caractéristique des protocoles de routage prédictifs de topologie est l'utilisation d'algorithmes d'apprentissage automatique pour prédire les trajectoires de mouvement des nœuds (comme une approximation de la topologie du réseau, si la portée de communication des nœuds est connue) et les intégrer dans le mécanisme de sélection de chemin.nous examinons certains des protocoles de routage proposés qui utilisent des approches de prédiction de mobilité ou de trajectoire pour améliorer les performances des algorithmes de routage pour les réseaux de drones :

— **Protocole MAC adaptatif basé sur l'apprentissage :** Ce protocole de routage propose un protocole de communication hybride adaptatif en intégrant un nouveau protocole MAC directionnel basé sur la prédiction de position (PPMAC) et un protocole de routage auto-apprenant basé sur l'apprentissage par renforcement. (RLSRP). Les résultats de performance montrent que le PPMAC proposé surmonte le problème de surdité directionnelle, qui se produit lorsque l'émetteur ne parvient pas à communiquer

avec le récepteur en raison de l'orientation de l'antenne du récepteur dans une direction différente. De plus, RLSRP fournit un schéma de routage évolutif automatiquement et plus efficace, adapté aux FANETs autonomes.

— **Dijkstra prédictif :**

Ce protocole de routage implique que l'emplacement des nœuds intermédiaires est prédéterminé en utilisant des techniques d'apprentissage automatique pour les rencontrer. Par la suite, il incorpore ces données prédictives dans le critère de choix du parcours basé sur l'algorithme de Dijkstra pour le parcours le plus court possible. Les résultats indiquent une performance supérieure à celle de l'algorithme standard de Dijkstra, particulièrement lorsqu'on applique des vitesses plus importantes. La performance améliorée est conditionnée par la exactitude de la prédiction. Cette méthode présente deux désavantages majeurs : elle repose sur des techniques spécifiques pour prédire la trajectoire et requiert une transmission d'informations d'emplacement global.

— **Algorithme de Clustering pour la Prédiction de Mobilité (MPCA) :**

Cet algorithme de routage est adapté aux réseaux de grappes de drones . Il identifie la plus grande fiabilité du nœud pour choisir le chef de cluster. Par la suite, il anticipe l'organisation du réseau en se basant sur la structure de données Trie, le modèle d'optimisation du temps d'expiration des liens.En outre, cette méthode assure la stabilité de la formation des groupes.

— **Routage ad-hoc prédictif alimenté par l'apprentissage par renforcement et les connaissances de trajectoire (PARRoT) :**

Il s'agit d'un autre protocole de routage alimenté par l'IA, qui exploite les informations de contrôle de la mobilité pour intégrer les connaissances sur le mouvement futur des agents mobiles dans le processus de routage. Chaque agent estime sa propre position future en se basant sur la position actuelle et propage le résultat aux autres nœuds. Cet algorithme atteint une robustesse supérieure et une latence de bout en bout significativement plus faible par rapport aux algorithmes similaires précédemment rapportés. Cet algorithme est approprié pour séparer la couche de mise en réseau de la planification de chemin ou lorsque les chemins sont planifiés à la volée, car on suppose que

les nœuds ne sont pas conscients de leurs emplacements futurs.

— **Prédictif Robuste et Fiable (RARP) :** Ce protocole de routage combine des schémas de transmission omnidirectionnels et directionnels avec un ajustement dynamique de l'angle. Cette méthode utilise de manière hybride les protocoles de routage unicasting et geocasting en se basant sur les informations de localisation et de trajectoire. Les emplacements des nœuds intermédiaires sont prédits en utilisant une estimation 3D ; ensuite, une transmission directionnelle est utilisée vers l'emplacement prédit, permettant une portée de transmission plus longue et le suivi des changements topologiques. Les auteurs montrent que leur méthode réduit la rétablissement de chemin.et le temps de perturbation du service et atteint des taux de livraison de paquets réussis plus élevés.

2. **Machine Learning (ML) :** est une branche de l'intelligence artificielle axée sur la création d'algorithmes et de méthodes permettant aux machines d'apprendre et d'exploiter des modèles mathématiques sans nécessité de programmation explicite.Son but principal consiste à extraire des données significatives pour optimiser les performances.Il existe trois catégories principales de ML et elles sont décrites comme suit [38] :

   — **Supervised learning :** Implique de donner une quantité de données comprenant diverses caractéristiques (X) et leurs valeurs finales entièrement correspondantes (y). L'ensemble de données est marqué par la connaissance des valeurs de sortie et des caractéristiques. Cette méthode analyse les modèles présents dans les informations et conçoit un modèle qui est en mesure de reproduire ces modèles sur de nouvelles données. donner les deux sous catégorie de ce type.Il se subdivise essentiellement en deux catégories distinctes : la Classification et la Régression.

   — **Unsupervised learning :** Consiste à former un modèle sur une base de données non marquée, où les informations d'entrée ne comportent aucune information de sortie associée. Par la suite, le modèle est tenu de déterminer lui-même les modèles et la structure des données. Apprentissage non contrôlé très utile pour des tâches comme le regroupement, la détection d'anomalies et la diminution de dimensionnalité.On la classe principalement en deux

groupes : le regroupement et la réduction de dimensionnalité.

— **Reinforcement learning :** Représente un type de machine learning plus complexe et plus avancé qui nécessite qu'un agent acquiert la capacité de prendre des décisions dans un contexte via des tests et des erreurs. Le fonctionnaire reçoit des remerciements ou des sanctions pour ses actes, qu'il exploite pour optimiser sa démarche de prise de décisions. Des applications comme les jeux, la robotique et les véhicules autonomes font appel à l'apprentissage par renforcement.La formation par renforcement basée sur les politiques et la formation par renforcement basée sur les valeurs sont principalement classées.

3. **Deep Learning (DL) :** Deep Learning est une branche de l'intelligence artificielle spécfiquement ML, axée sur la détection automatique des attributs significatifs des données et la génération de modèles sans nécessité de décrire manuellement les structures de ces dernières. Les modèles DL font fréquemment appel à des réseaux de neurones artificiels , qui tirent leur inspiration de la structure neuronale du cerveau humain. Les RNA se constituent de neurones artificiels structurés en strates d'entrée, cachées et de sortie, dotées de synapses qui autorisent le réseau à ajuster les paramètres. Les sorties des nœuds sont déterminées par les fonctions d'activation, tandis que la rétropropagation, associée à l'optimisation par descente de gradient, modifie les poids durant le processus d'entraînement. Les réseaux neuronaux profonds (DNNs) désignent des structures neuronales comportant plusieurs strates dissimulées, incluant les réseaux neuronaux de type feedforward (FNNs).Les systèmes de traitement d'information unidirectionnels, ainsi que les réseaux neuronaux récurrents (RNNs), élaborés pour traiter des informations en séquence. Un des algorithmes FNN les plus significatifs est celui basé sur les réseaux de neurones convolutifs (CNNs), qui excellent dans la détection d'images et d'émotions. Ils sont constitués de composants pour extraire les caractéristiques et la classification [38].

2.6.3 Optimisation de la QoS par l'IA

Dans les FANETs, l'intelligence artificielle permet d'améliorer la qualité du service en modifiant dynamiquement les paramètres réseau, en anticipant les conditions de

trafic et en optimisant le débit pour une performance réseau améliorée.dans ce segment, nous examinons des techniques pour optimiser la qualité du service dans les réseaux de drones à l'aide de techniques d'IA, visant à améliorer les performances dans des environnements mobiles et contraints en ressources [39] :

1. **Enjeux majeurs traités :** Les auteurs se concentrent sur des questions cruciales comme la gestion de la puissance et le déchargement des tâches dans les FANETs, où les drones doivent conserver une excellente performance malgré des limitations rigoureuses en matière d'énergie et de calcul.
2. **Approche suggérée :** L'approche repose sur le machine learning pour adapter de manière dynamique la distribution des tâches et la gestion de l'énergie, optimisant par conséquent la qualité du service tout en prenant en considération les contraintes énergétiques des drones.
3. **Adaptation aux conditions réseau :** Les résultats montrent que cette méthode fondée sur l'intelligence artificielle optimise les performances des réseaux de drones en comparaison avec les techniques traditionnelles de gestion des ressources.
4. **Effet de l'IA sur la gestion des ressources :**

   Cette section met en évidence le potentiel de l'intelligence artificielle pour une gestion optimale des ressources dans les FANETs, favorisant des opérations plus performantes et une capacité d'adaptation face aux contextes dynamiques et imprévisibles et l'approche suggérée contribue à optimiser la qualité de service dans les FANETs et les réseaux de drones en général, offrant des opportunités pour des applications sophistiquées comme le suivi en direct et les tâches de distribution autonomes.

2.7 Conclusion

Dans ce chapitre, nous avons exploré l'état de l'art des FANETs, en mettant en évidence les défis liés au routage et à la QoS dans ces réseaux dynamiques. Nous avons également analysé les indicateurs de performance du routage et de la QoS dans les FANETs, permettant d'évaluer l'efficacité des protocoles existants. En outre, nous avons introduit une nouvelle classification des protocoles de routage et de gestion de la QoS,

avec une description détaillée de chaque protocole. Cette étude nous permettra de capitaliser sur les protocoles existants pour proposer un nouvel algorithme d'optimisation du routage et de la QoS, dans le but d'améliorer les performances des FANETs. La conception et la description détaillée de cet algorithme seront discutées dans le prochain chapitre de notre mémoire.

Chapitre 3

Protocole de Routage Prédictif OLSR (P-OLSR)

3.1 Introduction

Les réseaux FANETs soulèvent des défis uniques en matière de communication, principalement en raison de la mobilité rapide et imprévisible des UAVs. Pour répondre à ces contraintes, le protocole Predictive Optimized Link State Routing (P-OLSR) a été développé comme une extension proactive du protocole OLSR traditionnel. En exploitant les données de positionnement GPS et la vitesse des UAVs, P-OLSR permet d'anticiper les variations topologiques du réseau et d'optimiser la sélection des routes.

Ce chapitre examine les principales améliorations introduites par P-OLSR, notamment l'optimisation de la métrique ETX, l'adaptation dynamique des chemins de communication et la réduction significative des interruptions de liens. Une analyse approfondie de l'impact des différents paramètres de configuration sur la stabilité et la performance globale du protocole sera également présentée, mettant en évidence les leviers d'optimisation possibles pour les réseaux FANETs de nouvelle génération.

3.2 Protocole P-OLSR

Le protocole P-OLSR est une extension améliorée du protocole OLSR, spécifiquement adaptée aux réseaux mobiles à forte dynamique tels que les FANETs. Il vise à mieux gérer la mobilité rapide et les changements fréquents de topologie en intégrant

des informations supplémentaires telles que les coordonnées GPS (longitude, latitude, altitude) et la vitesse relative des nœuds dans les messages `Hello` et `TC`.

En exploitant ces données, P-OLSR permet d'anticiper plus efficacement les ruptures de liens et d'optimiser la sélection des routes à travers une version adaptée de la métrique ETX. Cette approche améliore la stabilité des communications et réduit les coupures de connexion, particulièrement dans des scénarios de mobilité élevée.

Cependant, l'utilisation fréquente des mises à jour de topologie peut entraîner une surcharge du réseau, et la dépendance aux mesures GPS peut introduire des erreurs de localisation, affectant potentiellement la performance du protocole.

L'ensemble des étapes principales de l'algorithme P-OLSR est illustré dans la Figure 3.1 [40].

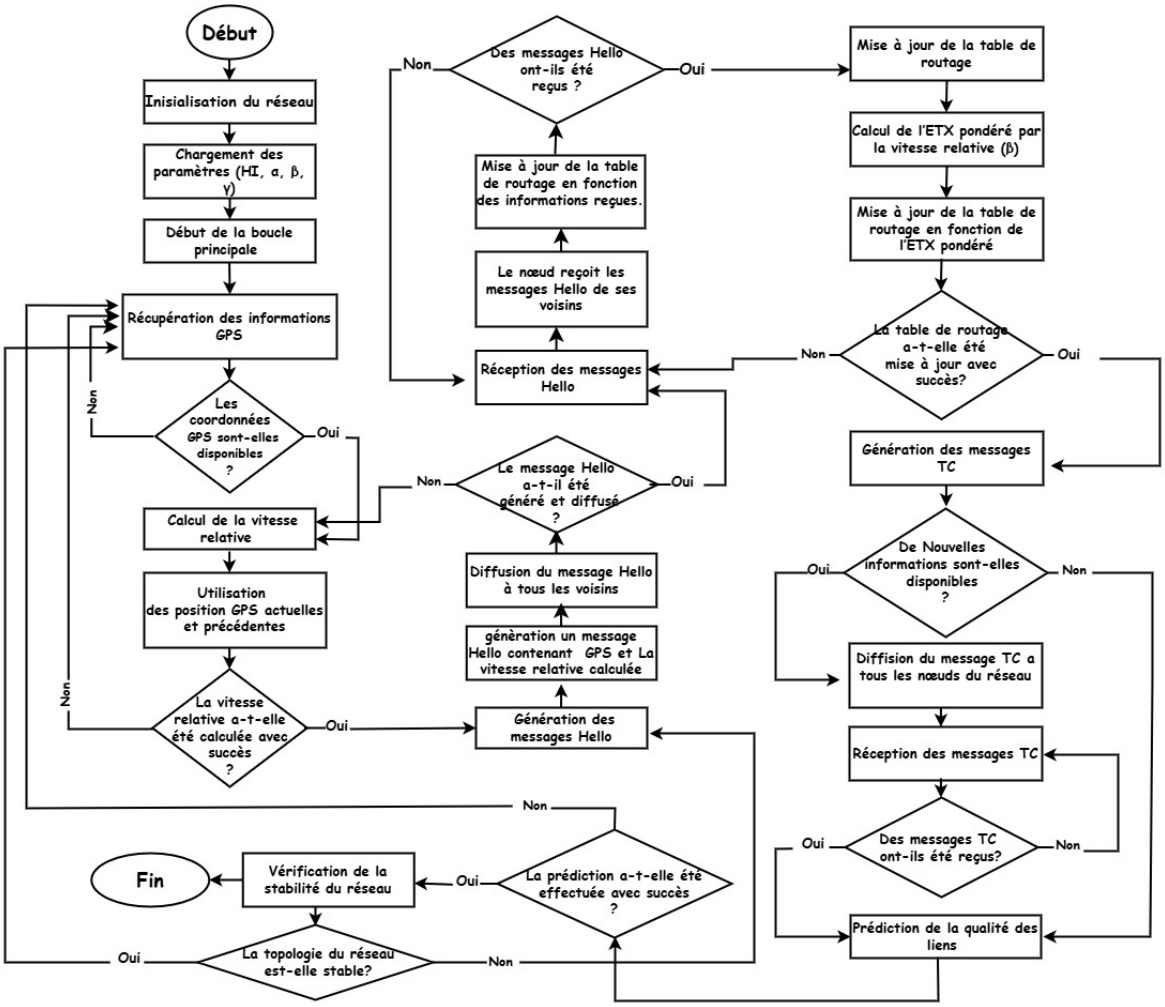

FIGURE 3.1 – Organigramme du protocole P-OLSR.

3.2.1 Métriques utilisées dans le protocole P-OLSR

Dans les réseaux à haute mobilité comme les FANETs, le choix et l'adaptation des métriques de routage sont essentiels pour maintenir une connectivité fiable. Le protocole P-OLSR introduit plusieurs ajustements spécifiques, basés sur la prise en compte de la mobilité et de la qualité dynamique des liens. L'intégration de l'Expected Transmission Count pondéré par la vitesse, de la probabilité de réception des liens, ainsi que de mesures de vitesse relative, permet une évaluation plus précise et proactive de la stabilité des connexions. Ces métriques adaptées à des environnements hautement dynamiques améliorent la sélection des routes, réduisent les coupures de liens, et augmentent l'efficacité globale du réseau.

1. **Expected Transmission Count (ETX)**

   L'ETX est une métrique classique utilisée pour estimer la fiabilité d'un lien en mesurant le nombre moyen de transmissions, y compris les retransmissions, nécessaires pour envoyer un paquet avec succès. Plus l'ETX est faible, meilleure est la qualité du lien.

   L'ETX d'une route R est défini par :

$$ETX(R) = \sum_{\eta \in R} ETX(\eta) = \sum_{\eta \in R} \frac{1}{\phi(\eta) \cdot \rho(\eta)} \tag{3.1}$$

   où :

   — R : Route entre deux nœuds.

   — η : Saut intermédiaire sur R.

   — $\phi(\eta)$: Probabilité de réception directe (forward reception ratio).

   — $\rho(\eta)$: Probabilité de réception inverse (reverse reception ratio).

   Exemple de calcul :

   — A envoie 100 paquets à B, B en reçoit 80 $\rightarrow \phi(\eta) = 0.8$,

   — B retourne 80 accusés de réception, A en reçoit 70 $\rightarrow \rho(\eta) = 0.875$.

   Le calcul de l'ETX est alors :

$$\text{ETX} = \frac{1}{0.8 \cdot 0.875} = \frac{1}{0.7} \approx 1.43$$

Interprétation : En moyenne, 1.43 transmissions sont nécessaires pour transmettre un paquet avec succès.

2. **Speed-Weighted ETX**

Dans les FANETs, la mobilité rapide des UAVs rend l'ETX classique insuffisant. P-OLSR améliore cette métrique en pondérant l'ETX par un facteur fonction de la vitesse relative entre deux nœuds, ce qui permet de privilégier les liens plus stables.

La formule ajustée est :

$$ETX(\eta) = \frac{e^{\beta v_{i,j}}}{\phi(\eta) \cdot \rho(\eta)} \tag{3.2}$$

où :

— $v_{i,j}$: Vitesse relative entre les nœuds i et j,

— β : Coefficient de pondération qui contrôle l'impact de la mobilité.

Effet de la vitesse relative :

— Si $v_{i,j} < 0$ (les UAVs se rapprochent), l'ETX diminue, favorisant ces liens.

— Si $v_{i,j} > 0$ (les UAVs s'éloignent), l'ETX augmente, pénalisant ces liens instables.

Exemple de calcul :

$$v_{i,j} = 10 \text{ m/s}, \quad \beta = 0.05$$

$$e^{0.5} \approx 1.6487$$

$$ETX' = \frac{1.6487}{0.8 \cdot 0.875} = \frac{1.6487}{0.7} \approx 2.36$$

Ainsi, par rapport à l'ETX classique de 1.43, l'effet de la mobilité est pris en compte, indiquant une qualité de lien moindre.

Paramètre	**ETX OLSR**	**ETX P-OLSR**
Mobilité prise en compte	Non	Oui (via $v_{i,j}$)
Formule	$\frac{1}{\phi(\eta)\cdot\rho(\eta)}$	$\frac{e^{\beta v_{i,j}}}{\phi(\eta)\cdot\rho(\eta)}$
Exemple de résultat	1.43	2.36

TABLE 3.1 – Comparaison entre ETX classique (OLSR) et ETX pondéré (P-OLSR)

3. **Probabilité de Réception du Lien (ϕ)**

   La qualité d'un lien est estimée dynamiquement à travers la réception des messages `Hello`. La probabilité de réception est mise à jour avec une moyenne mobile exponentielle :

$$\phi_l = \begin{cases} \alpha h_l + (1-\alpha)\phi_{l-1}, & l > 0 \\ 0, & l = 0 \end{cases} \tag{3.3}$$

   où :

   — $h_l = 1$ si le message Hello est reçu, 0 sinon.

   — α : Facteur de lissage ($0 \leq \alpha \leq 1$).

   Un α élevé rend la mesure plus réactive mais plus sensible aux variations ; un α faible rend la mesure plus stable mais moins rapide.

4. **Vitesse Relative Instantanée ($\tilde{v}_{i,j}$)**

   La vitesse relative instantanée entre deux nœuds est calculée en évaluant la variation de la distance entre deux réceptions de messages Hello :

$$\tilde{v}_{i,j} = \frac{d_{i,j}^{(t)} - d_{i,j}^{(t-1)}}{t - (t-1)} \tag{3.4}$$

   où $d_{i,j}^{(t)}$ et $d_{i,j}^{(t-1)}$ sont les distances aux instants t et $t-1$.

5. **Moyenne Mobile Exponentielle de la Vitesse ($v_{i,j}$)**

   Pour éviter les variations brusques dues aux erreurs GPS ou à la turbulence, la vitesse relative est lissée à l'aide d'une moyenne mobile exponentielle :

$$v_{i,j} = \begin{cases} \gamma \tilde{v}_{i,j} + (1-\gamma) v_{i,j-1}, & j > 0 \\ 0, & j = 0 \end{cases} \tag{3.5}$$

   où :

   — γ : Facteur de lissage.

   Un γ élevé donne une vitesse instantanée plus réactive, tandis qu'un γ faible filtre davantage les fluctuations et rend la mesure plus stable.

3.2.2 Optimisation du Routage dans P-OLSR

Pour améliorer la fiabilité et l'efficacité du routage dans les environnements hautement dynamiques comme les FANETs, le protocole P-OLSR introduit plusieurs modifications aux structures de messages utilisées par OLSR classique.

1. **Messages Hello dans P-OLSR**

   Dans P-OLSR, les messages Hello sont enrichis pour intégrer des informations de mobilité en plus des données de connectivité. Chaque message Hello contient désormais, en plus des adresses IP des voisins, les coordonnées GPS (longitude, latitude, altitude) et la vitesse relative moyenne des UAVs. La première partie du message passe de 8 à 16 octets pour inclure ces nouvelles informations. Pour chaque voisin détecté, un bloc supplémentaire de 8 octets est ajouté, contenant :

   (1) L'adresse IP (4 octets),

   (2) Le taux de réception en avant (ϕ),

   (3) Le taux de réception en retour (ρ),

   (4) La vitesse relative moyenne codée sur 2 octets.

   Cette évolution permet une meilleure anticipation des changements topologiques et favorise une sélection plus précise des routes selon la dynamique des UAVs. Le nouveau format est illustré dans la Figure **3.2**.

Bytes			
Byte 0	**Byte 1**	**Byte 2**	**Byte 3**
Altitude		Htime / Willingness	
Latitude			
Longitude			
Link Code	Reserved	Link Message Size	
Neighbor Interface Address			
$\phi(\eta')$	$\rho(\eta')$	$v^t_{i,j}$ (rel. speed)	
Neighbor Interface Address			
$\phi(\eta'')$	$\rho(\eta'')$	$v^t_{i,j}$ (rel. speed)	
...			

FIGURE 3.2 – Format du message Hello modifié dans P-OLSR.

2. **Messages TC dans P-OLSR**

   Les messages TC, utilisés pour la diffusion globale d'informations de topologie, conservent le même format général que dans OLSR. Toutefois, P-OLSR utilise les deux octets réservés du message TC pour inclure la vitesse relative moyenne des UAVs. Cette modification permet une mise à jour efficace et rapide des routes en fonction de la mobilité, tout en assurant une compatibilité descendante avec les systèmes OLSR classiques. Le format modifié est représenté dans la Figure 3.3.

Bytes			
Byte 0	**Byte 1**	**Byte 2**	**Byte 3**
ANSN (Advertised Neighbor Sequence Number)		Reserved	
Advertised Neighbor Main Address			
$\phi(\eta')$	$\rho(\eta')$	$v^t_{i,j}$ (rel. speed)	
Advertised Neighbor Main Address			
$\phi(\eta'')$	$\rho(\eta'')$	$v^t_{i,j}$ (rel. speed)	
...			

FIGURE 3.3 – Format du message TC modifié dans P-OLSR.

3.2.3 Banc d'essai UAV

Pour évaluer les performances de P-OLSR en conditions réelles, des essais ont été réalisés avec des UAVs de type eBee, développés par SenseFly.

Ces drones, à voilure fixe, combinent :

1. Une autonomie de vol d'environ 45 minutes,
2. Une vitesse de croisière moyenne de 57 km/h,
3. Une communication via Wi-Fi 802.11n avec technologie MIMO (trois antennes) et codage Space-Time Block Coding pour améliorer la diversité des canaux.

Chaque drone est équipé :

1. D'un ordinateur embarqué ARM Gumstix Overo Tide,
2. D'une carte d'extension comportant un hub USB 2.0 et une interface série pour la collecte des données GPS et des paramètres de vol.

Grâce à leur faible envergure (96 cm) et leur masse réduite (moins de 630 g), ces UAVs peuvent être utilisés en toute sécurité sans nécessiter d'autorisations spécifiques dans plusieurs pays. Ils constituent ainsi une plateforme idéale pour tester les protocoles de communication dans des réseaux mobiles fortement dynamiques, comme illustré dans la Figure 3.4.

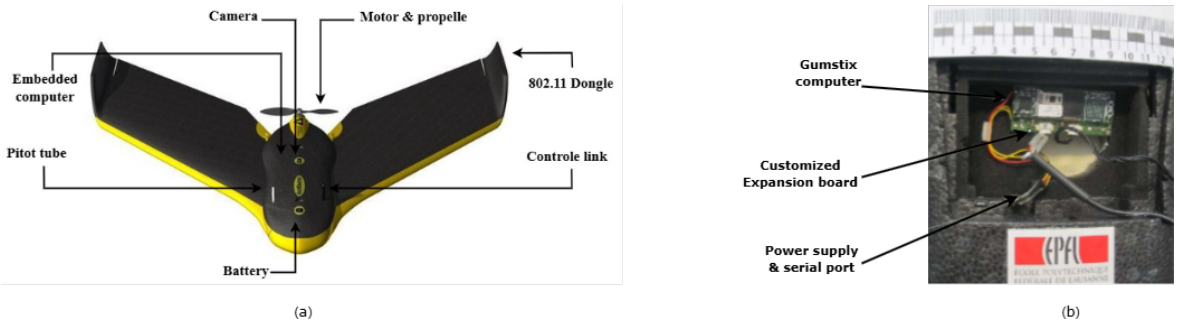

FIGURE 3.4 – Plateforme UAV utilisée dans les expérimentations.

3.2.4 Améliorations introduites par le protocole P-OLSR

P-OLSR propose plusieurs améliorations par rapport à OLSR classique, notamment pour renforcer la stabilité des liens et optimiser la qualité du routage en milieu mobile, comme résumé dans le Tableau 3.2 :

Aspect	**OLSR**	**P-OLSR**
Détection des liens	Basée uniquement sur les messages Hello.	Ajout des coordonnées GPS et de la vitesse relative.
Qualité des liens	ETX standard (nombre moyen de transmissions).	ETX pondéré par la vitesse relative.
Messages Hello	Informations de voisinage uniquement.	Ajout d'informations GPS et de mobilité.
Réactivité	Réactions lentes aux changements topologiques.	Anticipation proactive des ruptures de liens.
Stabilité des connexions	Connexions instables en cas de forte mobilité.	Connexions plus stables grâce à la prédiction.
Performance globale	Débit réduit, pertes accrues.	Meilleur débit, moins de pertes de paquets.

TABLE 3.2 – Principales améliorations introduites par P-OLSR.

3.2.5 Avantages de l'algorithme P-OLSR

P-OLSR présente plusieurs avantages significatifs, particulièrement adaptés aux environnements FANETs :

1. **Prédiction proactive de la topologie :** Anticipe les ruptures de lien en exploitant les données GPS et la vitesse relative.
2. **Routage optimisé :** Privilégie les chemins les plus stables grâce à l'ETX pondéré par la mobilité.
3. **Réduction des interruptions :** Maintient la continuité de transmission en adaptant les routes avant la perte des liens.
4. **Adaptabilité élevée :** Mieux adapté aux réseaux hautement mobiles que les protocoles classiques.
5. **Efficacité en signalisation :** Limite la surcharge en optimisant la taille et la fréquence des messages de contrôle.

3.2.6 Limitations de l'algorithme P-OLSR

Malgré ses nombreux avantages, P-OLSR présente certaines limitations :

1. **Précision excessive de l'ETX pondéré :** Dans certaines situations, la pondération par la vitesse peut surestimer l'instabilité des liens, entraînant des changements de routes prématurés et sous-optimaux.
2. **Surcharge de signalisation dans les réseaux denses :** Dans des environnements très peuplés, l'ajout d'informations GPS dans les messages Hello peut augmenter la congestion du réseau.
3. **Sensibilité aux paramètres de configuration :** La performance de P-OLSR dépend du réglage fin de paramètres comme l'intervalle Hello (HI) ou le facteur de pondération β ; un mauvais choix peut affecter négativement l'efficacité.
4. **Difficulté à suivre les changements brusques :** Lors de variations soudaines de trajectoire ou de manœuvres rapides, les prédictions basées sur les vitesses moyennes peuvent devenir obsolètes, réduisant la pertinence des routes sélectionnées.

3.3 Conclusion

Ce chapitre a permis de présenter en détail le protocole P-OLSR, une évolution du protocole OLSR classique conçue pour mieux répondre aux exigences des réseaux FANETs. En intégrant les données de mobilité des UAVs, telles que la position GPS et la vitesse relative, P-OLSR améliore la sélection des routes, anticipe les coupures de liens et optimise la stabilité des communications dans des environnements à forte dynamique. Nous avons examiné les optimisations apportées aux messages de contrôle, les ajustements de la métrique de routage ETX, ainsi que les bénéfices et les limites de cette approche face aux défis spécifiques des réseaux mobiles ad hoc.

Dans le prochain chapitre, nous proposerons une nouvelle amélioration du protocole P-OLSR en intégrant des techniques de deep learning. Cette approche innovante visera à rendre la prise de décision en matière de routage encore plus adaptative, en tirant parti des capacités d'apprentissage pour anticiper de manière plus précise les variations du réseau et améliorer la qualité de service globale.

Chapitre 4

Protocole de Routage Proposé P-OLSR-GRU

4.1 Introduction

Les réseaux de drones se distinguent par une mobilité élevée et une topologie en perpétuelle évolution, ce qui pose des défis majeurs pour assurer un routage efficace des données. Parmi les protocoles existants, le P-OLSR, une variante du protocole OLSR spécialement adaptée aux environnements aériens, est fréquemment utilisé. Cependant, ce protocole souffre de certaines limitations, notamment une réactivité limitée face aux changements rapides de la topologie et une évaluation parfois imprécise de la qualité des liens. Pour remédier à ces contraintes, nous proposons une approche innovante basée sur l'apprentissage profond, intégrant un modèle Gated Recurrent Unit (GRU). Ce modèle permet d'anticiper la stabilité des liens en se basant sur des paramètres tels que la qualité du signal, le taux de perte de paquets et la vitesse des drones, optimisant ainsi la mise à jour des routes.

Ce chapitre s'articule autour de plusieurs axes essentiels. Il commence par présenter la méthodologie adoptée pour intégrer le modèle GRU au protocole P-OLSR. Ensuite, il décrit les processus de collecte et de prétraitement des données nécessaires à l'entraînement du modèle. La section suivante est consacrée à la conception et à l'apprentissage du modèle GRU, en détaillant son architecture ainsi que les paramètres utilisés pour son optimisation. Enfin, nous expliquons comment cette intégration a donné naissance à un nouveau protocole, nommé P-OLSR-GRU, avant d'analyser son

impact sur les mécanismes de prise de décision liés au routage dans les réseaux de drones.

4.2 Cadre de Travail Proposé

L'amélioration du protocole P-OLSR à l'aide de l'apprentissage profond repose sur une méthodologie bien définie visant à intégrer un modèle GRU pour prédire la stabilité des liens et optimiser la sélection des routes dans un FANET. Cette section expose les objectifs de cette amélioration ainsi que la méthodologie adoptée pour la mise en œuvre de cette approche.

4.2.1 Analyse du Protocole P-OLSR

L'objectif de l'amélioration du protocole P-OLSR est de surmonter les contraintes de l'OLSR traditionnel dans des contextes dynamiques comme les FANETs. Ainsi, P-OLSR utilise des données de localisation GPS et de vitesse pour optimiser la détection et la qualité des connexions, tout en modifiant dynamiquement les messages Hello. Il prévoit les modifications de topologie, maintient la stabilité des liaisons en dépit des mouvements rapides des nœuds, et optimise les performances du réseau en minimisant les pertes de paquets et en garantissant un taux de transfert plus constant.

4.2.1.1 Limitations du P-OLSR

Ce protocole présente plusieurs limitations dans un réseau de drones :

1. **Faible réactivité aux changements topologiques :**

   Le protocole P-OLSR met à jour ses métriques de routage en fonction des informations recueillies via les messages `HELLO` et `TC`. Cependant, dans un réseau de drones où la topologie change rapidement, ces mises à jour ne sont pas assez fréquentes, ce qui peut entraîner une sélection de routes obsolètes et donc une dégradation des performances du réseau.

2. **Coût computationnel élevé :**

   L'actualisation des métriques de routage repose sur des calculs répétitifs et coûteux en ressources. Ces recalculs fréquents augmentent la charge de traitement

des UAVs, ce qui est problématique, car ces appareils disposent de ressources énergétiques et computationnelles limitées.

3. **Mises à jour inefficaces :**

   P-OLSR repose sur des valeurs historiques de la métrique ETX, qui sont souvent obsolètes lorsque les conditions du réseau évoluent rapidement. En conséquence, certaines routes peuvent rester privilégiées alors qu'elles ne sont plus optimales, ce qui engendre une latence accrue et des pertes de paquets plus élevées.

4.2.1.2 Objectif Général et Améliorations Proposées par le Modèle GRU

L'objectif principal de ce travail est d'intégrer un modèle de type GRU dans le protocole de routage **P-OLSR**, afin d'améliorer ses performances en matière de sélection de routes dans les réseaux FANETs. Le modèle vise à remplacer le calcul classique de la métrique ETX par une prédiction intelligente et proactive.

La méthodologie repose sur l'exploitation de caractéristiques dynamiques extraites du réseau, capturant l'évolution des conditions de communication entre les nœuds mobiles. Ces caractéristiques serviront ultérieurement comme entrées au modèle, permettant d'anticiper les variations de qualité des liens. La cible de prédiction est définie comme une mesure qualitative de la stabilité du lien, intégrée dans le processus de décision du protocole.

La variable cible (target) $y_{(i,j)}$ est définie comme une étiquette de classification binaire, représentant la qualité du lien de communication entre les nœuds i et j, et dérivée de la valeur moyenne historique estimée du métrique ETX.

$$y_{(i,j)} = \begin{cases} 1, & \text{si ETX}(i,j) \leq 2{,}0 \\ 0, & \text{si ETX}(i,j) > 2{,}0 \end{cases} \tag{4.1}$$

où :

— $y_{(i,j)}$ représente la classe binaire attribuée au lien (i, j),

— Une étiquette égale à 1 correspond à un lien stable (optimal),

— Une étiquette égale à 0 indique un lien non stable (non optimal).

1. **Avantages de l'Intégration du Modèle GRU**

   (1) **Prédire la stabilité des liens de communication**

   Le modèle GRU exploite la nature temporelle des données pour anticiper

les dégradations de lien à partir des tendances des métriques réseau. Cela permet une sélection proactive de routes plus fiables.

(2) **Réduire la fréquence des mises à jour de l'ETX**

Grâce aux prédictions, seules les évolutions significatives des liens déclenchent des mises à jour, réduisant ainsi les messages de contrôle (HELLO, TC) et allégeant le trafic réseau.

(3) **Optimiser l'utilisation des ressources des UAVs**

Moins de transmissions inutiles et de recalculs permettent de préserver la batterie des UAVs, d'améliorer l'efficacité énergétique et de prolonger la durée de vie des communications.

(4) **Faciliter la prise de décision binaire dans P-OLSR**

Le modèle fournit directement une classification « lien stable » ou « lien instable », simplifiant l'intégration dans les mécanismes de sélection de route du protocole P-OLSR amélioré.

4.2.2 Méthodologie de Développement du Modèle d'Apprentissage Profond

L'optimisation du protocole P-OLSR par l'intégration du modèle GRU s'appuie sur une démarche méthodologique rigoureuse, articulée autour de plusieurs étapes clés : la collecte et le prétraitement des données, la conception, l'entraînement, puis l'évaluation du modèle. Le GRU, en tant que modèle d'apprentissage profond séquentiel, est particulièrement adapté au traitement de données temporelles et contextuelles, ce qui en fait un choix pertinent dans le cadre dynamique des réseaux de drones.

1. **Collection des Données :**

   La première étape consiste à collecter des données utilisables pour évaluer les performances du réseau et former le modèle GRU. Ces informations sont issues de deux principales origines : des simulations effectuées à l'aide d'outils tels que NS-3 et des résultats réelles obtenues grâce à des drones en vol. Les variables clés comprennent la puissance du signal, qui détermine la qualité de la liaison de communication, le pourcentage de 60 paquets perdus, qui témoigne de la fiabilité du réseau, et la rapidité des UAVs, qui a un impact sur la stabilité des connexions.

Le but de cette étape est d'acquérir un jeu de données qui reflète les conditions dynamiques rencontrées dans un FANET, dans le but d'améliorer l'apprentissage du modèle.

2. **Pré-traitement des Données :**

   Avant d'être utilisées pour entraîner le modèle, les données brutes nécessitent plusieurs transformations pour optimiser leur qualité et leur utilisabilité. Une normalisation est mise en œuvre afin d'uniformiser les échelles des différentes variables, ce qui assure une meilleure convergence du modèle. Par la suite, les informations sont codées sous forme de séquences temporelles pour permettre au GRU de saisir l'évolution des liens dans le temps et prédire leur stabilité. En outre, nous procédons à une gestion des valeurs manquantes afin d'éviter toute distorsion et garantir la cohérence des données. Ces étapes garantissent que l'apprentissage est basé sur des données fiables et bien organisées.

3. **Conception et Entraînement du Modèle :**

   Après le prétraitement des données, une architecture de base du modèle GRU est proposée. L'objectif est de concevoir un modèle capable de capturer l'évolution temporelle des liens dans un réseau FANET, afin d'anticiper leur stabilité. À ce stade, l'architecture est pensée de manière générale, sans entrer dans les détails d'implémentation ou de configuration des paramètres. L'entraînement concret du modèle sera réalisé ultérieurement une fois que les données finales seront disponibles et validées.

4. **Évaluation et Validation :**

   L'évaluation du modèle suivra une approche progressive dès que l'entraînement sera effectué. Elle consistera à vérifier la capacité du modèle à généraliser ses prédictions à partir de nouvelles données. Une fois intégré dans le protocole P-OLSR, son impact sera évalué à travers des simulations, en analysant des critères comme la stabilité, le débit et la latence. Cette étape visera à valider l'intérêt de l'approche prédictive pour renforcer l'efficacité du routage dans les réseaux de drones.

4.3 Préparation des Données

Cette section décrit le pipeline de traitement des données avant leur intégration dans le modèle GRU. Elle couvre les étapes de collecte, de nettoyage et de structuration des données pour garantir une prédiction fiable et optimisée des métriques du réseau.

4.3.1 Collection des Données

Nous avons constitué un jeu de données riche et représentatif pour l'entraînement et l'évaluation de notre modèle GRU, qui comprend des indicateurs réseau essentiels pour prédire l'indicateur de stabilité du lien *'link_ stability'*. Ces informations proviennent de scénarios simulés dans un environnement de simulation spécifiquement conçu pour les réseaux de drones, complétées par des mesures obtenues à partir d'expériences concrètes menées dans des conditions maîtrisées.

Le but premier de cette collecte de données est d'enregistrer les dynamiques temporelles et spatiales du réseau, en tenant compte de paramètres tels que la qualité de la connexion, la structure topologique et les habitudes de déplacement des drones. Les indicateurs choisis ont été sélectionnés pour leur pertinence dans l'estimation de la stabilité et de l'efficacité des liaisons.Il convient de mentionner que diverses études précédentes ont examiné des méthodes analogues pour la collecte de données et l'évaluation de la performance au sein des réseaux de drones, avant d'aborder les instruments de simulation employés. Ces travaux ont grandement favorisé l'évolution des techniques de simulation pour la prévision du routage et l'évaluation de la qualité des connexions.

4.3.1.1 Source de données

Plusieurs recherches ont contribué à l'amélioration des techniques de collecte de données pour les réseaux ad hoc et les réseaux de drones. Diverses méthodes ont été envisagées dans le contexte de la simulation et de l'évaluation des performances de ces systèmes, y compris en ce qui concerne la prédiction du routage et l'analyse de la qualité des liens.

Voici les outils et environnements de simulation majeurs utilisés dans cette situation :

1. **Historiques de Simulation**

Dans le domaine de la recherche sur les réseaux, la génération de données peut être réalisée à l'aide d'outils tels que NS-3, est un simulateur basé sur des événements discrets, largement utilisé dans le domaine académique pour la recherche sur les réseaux. Il est conçu pour modéliser des scénarios de communication complexes, notamment ceux des réseaux ad-hoc et des réseaux de drones. Grâce à sa flexibilité et à sa précision, NS-3 permet de simuler divers protocoles de routage, des schémas de mobilité, et d'extraire des métriques telles que l'ETX, la latence, ou encore le débit. Sa nature open-source et son orientation vers la validation expérimentale en font un outil privilégié pour la construction de scénarios réalistes et l'évaluation des performances dans des environnements dynamiques .Pour assurer la fiabilité des résultats issus des simulations, une phase de validation expérimentale a été menée. Celle-ci a consisté à comparer les sorties du simulateur NS-3 avec des données collectées dans des environnements contrôlés ou inspirés de conditions réelles. Ces expérimentations ont permis de renforcer la crédibilité des modèles utilisés, en identifiant les écarts éventuels entre les prévisions simulées et les comportements observés. Ce processus a également contribué à l'ajustement des paramètres du simulateur, assurant ainsi une meilleure correspondance entre les résultats théoriques et les conditions de fonctionnement réelles, et garantissant la robustesse des données d'entrée utilisées pour l'entraînement du modèle de prédiction [1] [41].

4.3.2 Prétraitement des Données et Extraction des Caractéristiques

La phase préliminaire de traitement des données constitue une étape fondamentale dans la préparation des métriques réseau issues des communications entre drones, en amont de leur intégration dans le modèle GRU. Ce dernier nécessite des entrées sous forme de séquences temporelles et numériques structurées de manière rigoureuse. Ainsi, plusieurs opérations de traitement sont appliquées aux données brutes afin d'assurer leur cohérence, leur qualité et leur compatibilité avec l'architecture du modèle. Ces opérations incluent la tokenisation, l'encodage numérique, le remplissage (padding), la normalisation, ainsi que l'extraction de caractéristiques pertinentes. L'objectif principal

1. Consulté le 18 avril 2025, à partir du site : `https://www.nsnam.org/`

est de capturer efficacement les dépendances temporelles et spatiales essentielles à la prédiction fiable de la qualité des liens dans les réseaux FANETs. Dans cette optique, certaines caractéristiques spécifiques sont extraites pour renforcer les performances du modèle.

4.3.2.1 Description du Dataset

L'étude se base sur le dataset `"FANET Weighted ETX"`, élaboré pour étudier et améliorer les mécanismes de routage dans les FANETs, notamment en ce qui concerne la robustesse et la stabilité des liens de communication entre véhicules aériens sans pilote. Il comprend 100 000 enregistrements répartis sur plusieurs indicateurs temporels et spatiaux influençant la qualité des liaisons.Chaque ligne représente une observation à un instant donné, correspondant aux caractéristiques d'un lien de communication entre deux UAVs. Ce jeu de données est essentiel pour entraîner un modèle GRU ayant pour objectif de prédire la stabilité d'un lien, information critique pour les décisions de routage dans le protocole P-OLSR amélioré. La variable cible du dataset est `link_stability`, une variable binaire indiquant si le lien est stable (1) ou instable (0).

Afin d'assurer un apprentissage efficace et une évaluation rigoureuse du modèle, les données ont été divisées comme suit : 80% des enregistrements ont été utilisés pour l'entraînement, tandis que les 20% restants ont été répartis équitablement entre les phases de test (10%) et de validation (10%).

1. **Description des Caractéristiques**

   Les principales caractéristiques extraites du dataset `"FANET Weighted ETX"` sont présentées ci-dessous, accompagnées de leurs abréviations utilisées dans l'entraînement du modèle GRU :

   (1) **forward_reception_ratio** : Ce métrique présente taux de succès de réception des paquets lors de la transmission directe entre deux UAVs.

   (2) **reverse_reception_ratio** : Ce métrique présente taux de succès de réception des accusés de réception (ACK) en retour, reflétant la fiabilité du lien bidirectionnel.

   (3) **relative_speed** : Ce métrique constitue la vitesse relative instantanée entre deux UAVs, exprimée en mètres par seconde, influençant la stabilité du lien.

(4) **distance_between_nodes** : Ce métrique mesure la distance instantanée séparant deux UAVs, mesurée en mètres, affectant la qualité du signal.

(5) **signal_strength** : Ce métrique indique la puissance du signal reçue par l'UAV, mesurée en décibels-milliwatts (dBm), indicateur clé de la qualité de réception.

(6) **link_duration_estimate** : Ce métrique est un estimation en secondes de la durée pendant laquelle un lien de communication est maintenu sans interruption.

(7) **link_stability** : Ce métrique correspond la variable cible binaire indiquant la stabilité du lien : 1 pour un lien stable, 0 pour un lien instable.

Comme illustré dans cette Figure 4.1

	forward_reception_ratio	reverse_reception_ratio	relative_speed	distance_between_nodes	signal_strength	link_duration_estimate	link_stability
0	0.849671	0.953059	17.809204	80.008915	-49.822871	19.369751	0
1	0.786174	0.734465	9.528858	177.391138	-65.740491	21.797481	1
2	0.864769	0.907544	3.352322	143.961884	-73.658788	36.832299	0
3	0.952303	0.788076	3.056809	135.836812	-62.424804	34.178992	1
4	0.776585	0.817260	8.286746	188.378461	-65.318705	34.224583	1

FIGURE 4.1 – Extrait du jeu de données utilisé.

2. **Statistiques Descriptives**

Le Tableau 4.1 présente une analyse complémentaire des principales caractéristiques extraites du dataset utilisé dans cette étude. Aucune donnée manquante n'a été observée, et les ratios de réception ont été normalisés entre 0 et 1 afin de garantir l'homogénéité des variables d'entrée. De plus, la variable cible `'link_stability'` est suffisamment équilibrée pour permettre un apprentissage supervisé efficace. La diversité des vitesses relatives et des distances entre UAVs assure la représentativité de différentes situations réalistes de fonctionnement. Il est à noter que ces valeurs sont inspirées de données réelles et reflètent fidèlement les comportements observés dans les réseaux de drones en conditions pratiques.

Conformité au Monde Réel :

(1) **Vitesse relative :** Entre 0 et 30 m/s, en accord avec les drones commerciaux.

(2) **Distances inter-nœuds :** De quelques mètres jusqu'à plus de 300 m, couvrant une large gamme de scénarios de communication UAV.

Feature	Type	Min	Max	Moyenne	Écart-type
forward_reception_ratio	float64	0.2678	1.0000	0.7996	0.0997
reverse_reception_ratio	float64	0.3127	1.0000	0.8502	0.0939
relative_speed	float64	-5.0052	29.7105	9.9766	4.9600
distance_between_nodes	float64	4.7796	361.8913	150.1312	49.8697
signal_strength	float64	-85.3248	-45.2804	-65.0364	5.0028
link_duration_estimate	float64	0.7885	74.3895	29.9890	10.0321
link_stability	int64	0.0000	1.0000	0.4067	0.4914

TABLE 4.1 – Statistiques descriptives du dataset FANET Weighted ETX

(3) **Puissance du signal :** Cohérente avec les plages typiques des technologies Wi-Fi/LTE embarquées, reflétant des conditions réalistes de communication sans fil.

(4) **Ratios de réception :** Conformes aux performances observées dans les réseaux ad hoc opérationnels, assurant une représentativité fidèle du comportement réseau.

3. **Adaptation à un Modèle Séquentiel GRU**

Le contenu de ce dataset est naturellement séquentiel, car il capture l'évolution dynamique des liens de communication dans un contexte mobile et temporel :

(1) **Caractère temporel :** Chaque observation reflète un état instantané mais évolutif d'une connexion UAV, dépendant des vitesses et distances au fil du temps.

(2) **Corrélations dynamiques :** Les variations dans la puissance du signal, la distance, ou les taux de réception sont étroitement liées à des séquences temporelles d'événements physiques.

(3) **Motivation pour GRU :** Les modèles GRU sont spécifiquement conçus pour exploiter les dépendances séquentielles et temporelles dans les données, permettant une modélisation plus précise de l'évolution d'un lien au fil du temps.

(4) **Entraînement prédictif :** Grâce à la structure séquentielle implicite, un modèle GRU peut apprendre à prédire la stabilité d'un lien non seulement à partir de ses caractéristiques statiques, mais aussi de leur évolution historique récente.

Ainsi, l'utilisation d'un modèle séquentiel GRU est pleinement justifiée pour ce

dataset, offrant une capacité d'anticipation et une robustesse supérieure par rapport à des modèles standards non séquentiels.

La Figure 4.2 illustre la répartition des classes de la variable cible, qui reflète la stabilité des liens au sein d'un réseau FANET. La classe 0 représente les liens instables, tandis que la classe 1 correspond aux liens stables. Le jeu de données utilisé est globalement bien équilibré, avec environ 55 000 instances pour la classe 0 et 45 000 pour la classe 1. Cette répartition équilibrée est essentielle pour garantir l'efficacité de l'apprentissage supervisé et permet une évaluation plus fiable des performances du modèle dans des conditions réseau diverses.

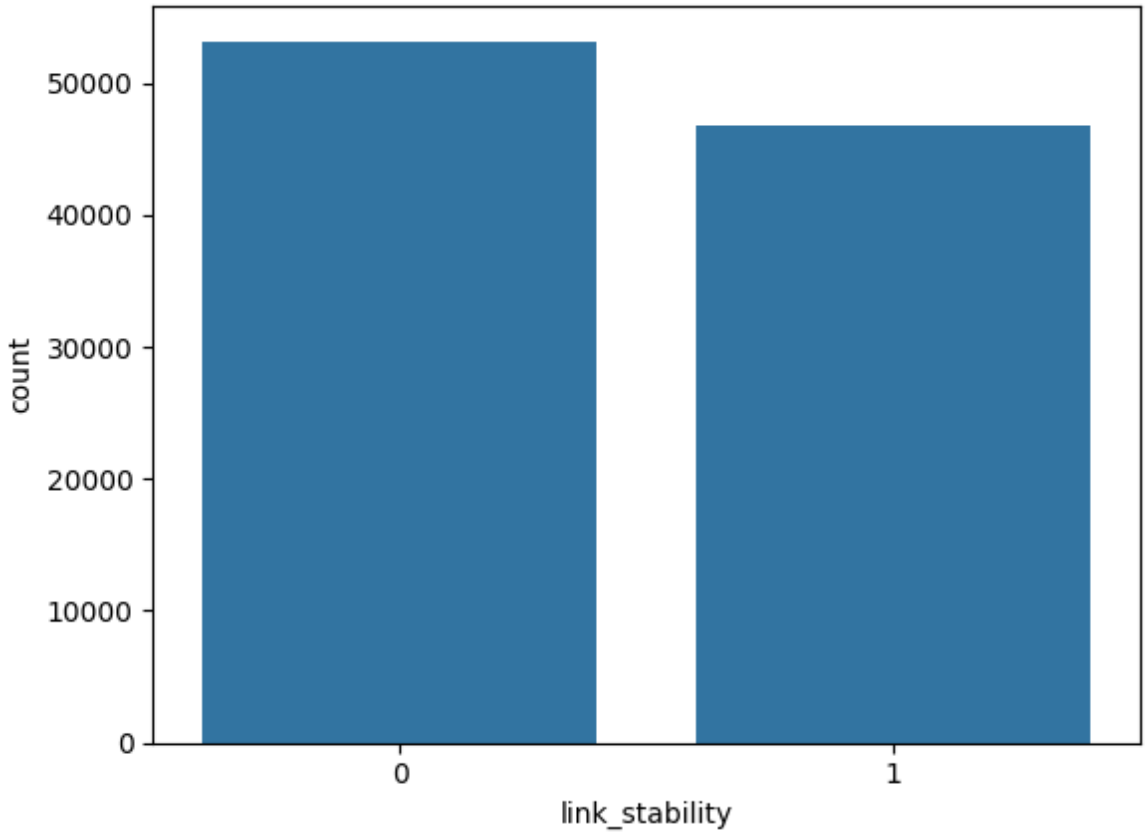

FIGURE 4.2 – Distribution de la variable cible.

4.3.2.2 Préparation des Données Séquentielles

Avant d'entraîner le modèle GRU, les données brutes provenant des simulations réseau sont organisées sous forme de séquences temporelles exploitables. Cette préparation implique diverses phases cruciales comme la structuration des séquences, la normalisation et le padding, afin de garantir une structure uniforme et cohérente des

entrées pour maximiser l'efficacité de l'apprentissage du modèle, dans le but de prédire la stabilité du lien.

1. **Tokenisation et Structuration des Séquences**

   Dans cette étape, chaque métrique du réseau (forward_reception_ratio, relative_speed, reverse_reception_ratio, distance_between_nodes, signal_strength, link_duration_estimate) est organisée en séquences temporelles continues. Cette structuration préserve l'ordre chronologique et l'évolution des caractéristiques physiques du lien entre UAVs, permettant au modèle GRU de mieux capter les dynamiques sous-jacentes liées à la stabilité de la connexion.Par exemple, la variation du signal_strength ou de la relative_speed au fil du temps est traduite en une séquence structurée, facilitant ainsi la détection des motifs précurseurs d'un lien stable ou instable.

2. **Encodage Numérique et Couche d'Embedding**

   Après la tokenisation, chaque métrique est convertie en une représentation numérique continue à travers une couche d'embedding. Cette conversion a pour but de placer les valeurs discrètes ou catégoriques dans un espace vectoriel dense, où les métriques présentant des comportements semblables sont disposées à proximité l'une de l'autre. L'utilisation de l'encodage numérique facilite l'expression des relations implicites entre les paramètres du réseau, ce qui rend plus aisé l'apprentissage des corrélations entre des variables telles que la vitesse des drones, la qualité du signal et la latence. La taille des embeddings est sélectionnée selon la complexité des données, assurant un équilibre optimal entre précision et performance de calcul.

3. **Normalisation et Padding des Séquences**

   Pour standardiser les entrées du modèle, toutes les séquences sont normalisées afin que chaque métrique ait une échelle comparable. Cette normalisation accélère la convergence de l'entraînement et évite que certaines variables dominent l'apprentissage.Ensuite, afin d'assurer une longueur uniforme pour toutes les séquences d'entrée, un padding est appliqué lorsque nécessaire : les séquences plus courtes que la longueur définie sont complétées par des valeurs neutres. Cette uniformisation est essentielle pour permettre un traitement efficace en lots (batchs) lors de l'entraînement du modèle.

Les caractéristiques extraites du dataset sont directement utilisées comme vecteurs d'entrée pour le modèle GRU, dans le but spécifique de prédire la stabilité du lien dans un réseau FANET. La Figure 4.3 présente la matrice de corrélation entre les différentes métriques considérées. Cette visualisation permet d'analyser les interactions et inter-dépendances entre les variables, en mettant en évidence les facteurs les plus influents sur la stabilité des liens. L'observation de ces corrélations facilite la compréhension du comportement du réseau et oriente l'entraînement du modèle vers les paramètres les plus pertinents pour améliorer la prédiction de la stabilité de la connexion.

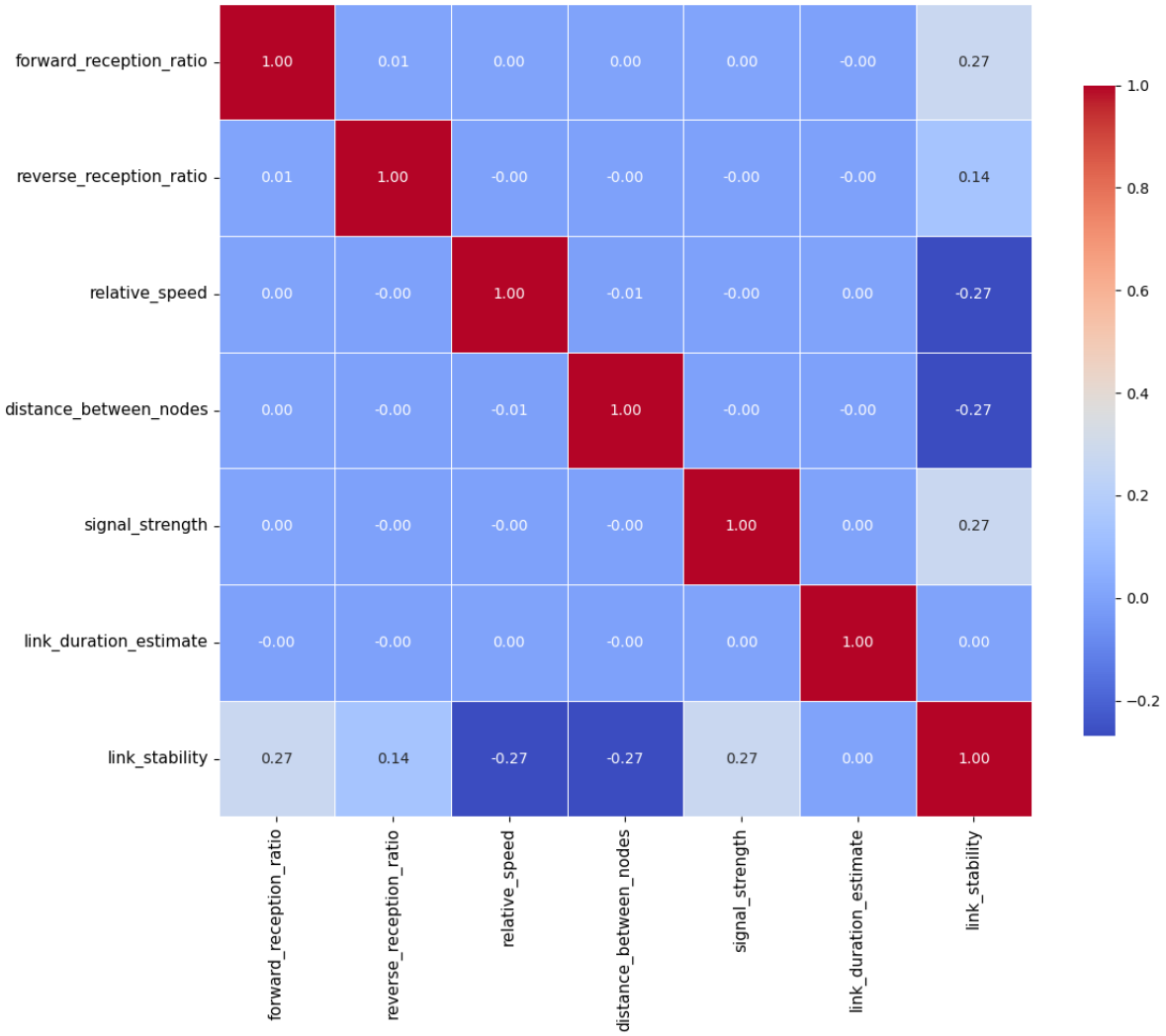

FIGURE 4.3 – Matrice de corrélation.

4.4 Conception et Entraînement du Modèle GRU

Afin d'améliorer la prédiction de la stabilité des liens dans les réseaux FANET, nous avons élaboré un modèle basé sur des unités GRU , capables de capturer les dépendances temporelles présentes dans les données réseau. Cette section détaille l'architecture du modèle, ses différentes couches, ainsi que les choix de conception adoptés pour optimiser la capacité de prédiction de la stabilité des connexions entre UAVs.

4.4.1 Architecture du Modèle

Pour assurer une prédiction précise de la stabilité des liens dans les réseaux FANET, nous avons conçu une architecture basée sur des unités GRU , capable de modéliser efficacement les dynamiques temporelles des métriques réseau. Cette section présente d'abord la composition détaillée des couches constituant le modèle, puis décrit les choix d'hyperparamètres et les stratégies d'optimisation employées pour maximiser les performances du système prédictif.

4.4.1.1 Architecture des Couches

Le modèle GRU est construit sur la base de multiples couches successives, chacune étant sélectionnée pour son rôle particulier dans le processus de modélisation des dépendances temporelles des indicateurs de performance du protocole P-OLSR. Voici une explication approfondie de l'agencement de ces couches.

1. **Couche d'Entrée :**

   La couche d'entrée reçoit les séries temporelles des indicateurs de performance du réseau : bande passante, perte de paquets, délai et variation de délai. Ces séries sont d'abord normalisées, ce qui permet de garantir que les différentes séries d'entrée ont une échelle comparable, facilitant ainsi l'entraînement du modèle. Cette normalisation est réalisée en utilisant la moyenne et l'écart-type des séries temporelles d'entrée.

2. **Couche GRU :**

   Le cœur de l'architecture est constitué d'une seule couches GRU empilées. Le GRU est une architecture de RNN qui, tout comme les LSTM, permet de capturer les dépendances temporelles dans des séries longues tout en évitant les problèmes

de vanishing gradient. Cependant, contrairement aux LSTM qui utilisent une cellule d'état, les GRU n'ont qu'un seul état caché.

Les GRU se composent de deux portes principales :

(1) **Porte de Réinitialisation (Reset Gate) :**

Elle détermine comment combiner la nouvelle entrée avec la mémoire précédente. La porte de réinitialisation r_t est utilisée pour ajuster le contenu de l'état caché précédent h_{t-1} avec l'entrée actuelle x_t. Cela permet de gérer l'oubli partiel des informations passées.

(2) **Porte de Mise à Jour (Update Gate) :**

Elle détermine combien de l'état caché précédent h_{t-1} doit être conservé pour prédire l'état caché actuel h_t. Cette porte contrôle donc la mémoire à long terme du modèle. Comme illustré dans la Figure 4.4 .

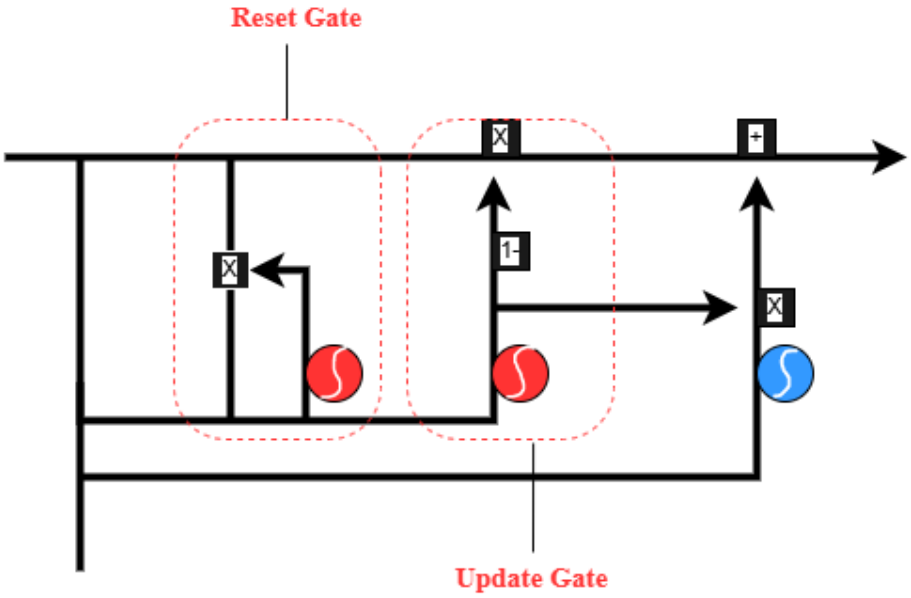

FIGURE 4.4 – Architecture interne d'une cellule GRU.

Le calcul de l'état caché h_t dans un GRU est défini par la formule suivante :

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{4.2}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{4.3}$$

où :

— $\tilde{h}_t$ est l'état caché candidat, obtenu en appliquant la fonction tangente hyperbolique (tanh) après la combinaison de la porte de réinitialisation r_t et l'entrée x_t.

— z_t est la porte de mise à jour, qui contrôle la proportion de l'état caché précédent h_{t-1} à conserver.

— W_h et b_h sont les poids et biais associés au calcul de $\tilde{h}_t$.

— $\odot$ est l'opération de produit Hadamard (élément par élément).

Si la porte de réinitialisation r_t est fixée à 1 et la porte de mise à jour z_t à 0, cela revient à un RNN classique où toute l'information précédente est oubliée et une nouvelle mémoire est complètement formée.

(3) **Nombre de Couches GRU :**

Une seule couche GRU est utilisée dans l'architecture afin de capturer efficacement les motifs temporels présents dans les séries de métriques réseau. Cette couche permet au modèle de saisir les dépendances temporelles essentielles tout en limitant la complexité et le risque de surapprentissage, ce qui est particulièrement adapté aux environnements dynamiques des réseaux FANET.

(4) **Taille des Couches Cachées :**

Le modèle utilise une unique couche GRU contenant 64 unités cachées. Ce nombre est choisi pour permettre au modèle d'apprendre des représentations séquentielles pertinentes tout en maintenant une efficacité computationnelle. L'utilisation de 64 unités représente un compromis entre complexité du modèle et capacité d'apprentissage, adapté à la prédiction binaire de la stabilité du lien

3. **Couche Dense de Sortie :**

Après la couche GRU, une couche Dense intermédiaire avec 32 neurones et une activation ReLU est utilisée pour renforcer la capacité d'extraction de caractéristiques du modèle. Ensuite, une couche Dense finale avec 1 neurone et une activation sigmoïde transforme la sortie en une probabilité représentant la stabilité du lien. La fonction d'activation sigmoïde est appropriée pour ce problème de classification binaire.

4. **Régularisation avec Dropout :**

Un taux de dropout de 30% est appliqué après la couche GRU et après la première couche Dense afin de réduire le risque de surapprentissage. Le dropout désactive aléatoirement certaines unités pendant l'entraînement, ce qui améliore la généralisation du modèle sur de nouvelles données de communication entre UAVs

4.4.1.2 Hyperparamètres et Optimisation

Le Tableau 4.2 présente les hyperparamètres adoptés pour entraîner le modèle GRU afin d'optimiser la prédiction de la stabilité des liens dans le contexte du routage P-OLSR au sein des réseaux FANETs. Ces hyperparamètres ont été soigneusement sélectionnés pour assurer un bon équilibre entre performance, capacité de généralisation et temps d'apprentissage.

1. **Nombre de Couches GRU :** Le modèle utilise une seule couche GRU afin d'extraire les dépendances temporelles essentielles dans les données séquentielles, tout en limitant la complexité pour éviter le surapprentissage.
2. **Taille des Couches Cachées :** La couche GRU comporte 64 unités cachées, permettant de capturer des relations temporelles pertinentes tout en maintenant une efficacité computationnelle adaptée.
3. **Taux de Dropout :** Un taux de régularisation de 30% est appliqué après la couche GRU et la couche Dense pour réduire le risque de surapprentissage et améliorer la généralisation.
4. **Taux d'Apprentissage :** Le modèle utilise l'optimiseur Adam avec un taux d'apprentissage par défaut de 0.001, assurant une convergence progressive et stable.
5. **Taille du Batch :** Chaque mise à jour des poids est réalisée après traitement de 128 échantillons, permettant un bon compromis entre vitesse d'apprentissage et stabilité des gradients.
6. **Nombre d'Époques :** Le modèle est entraîné pendant 50 époques, offrant suffisamment de cycles pour un apprentissage efficace sans provoquer de surajustement grâce au mécanisme d'EarlyStopping.
7. **La Couche Dense :** Après la couche GRU, une couche Dense de 32 unités

avec activation ReLU est utilisée pour enrichir les représentations apprises, suivie d'une couche Dense finale à 1 neurone avec une activation Sigmoid, adaptée à la classification binaire de la stabilité du lien.

8. **Optimiseur :** L'optimiseur Adam est utilisé pour sa capacité à s'adapter dynamiquement aux caractéristiques du problème et accélérer la convergence.
9. **Fonction de Perte :** La Binary Cross-Entropy est employée, appropriée pour la prédiction binaire de la stabilité des liens.

Hyperparamètre	**Valeur**
Nombre de Couches GRU	1
Taille des Couches Cachées	64 unités par GRU
Taux de Dropout	0.3
Taux d'Apprentissage	0.001
Optimiseur	Adam
Fonction de Perte	Binary Cross-Entropy
Nombre d'Unités dans la Couche Dense	32
Activation de la Couche Dense	ReLU
Activation de la Couche de Sortie	Sigmoid
Nombre d'Époques	50
Taille du Batch	128

TABLE 4.2 – Hyperparamètres du Modèle GRU le Routage P-OLSR-GRU.

La Figure 4.5 présente l'architecture du modèle GRU utilisé pour prédire la stabilité des liens dans un réseau FANET. Le modèle est composé d'une couche GRU avec 64 unités, suivie de couches de régularisation par Dropout pour limiter l'overfitting, et de deux couches Dense pour la prise de décision binaire.

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 64)	13,824
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 32)	2,080
dropout_1 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 1)	33

Total params: 15,937 (62.25 KB)
Trainable params: 15,937 (62.25 KB)
Non-trainable params: 0 (0.00 B)

FIGURE 4.5 – Synthèse des couches du modèle GRU.

4.5 Implémentation du Modèle GRU pour la Prédiction de l'ETX

Cette section décrit les différentes étapes techniques entreprises pour concevoir, entraîner et tester un modèle GRU visant à prédire la stabilité des liens dans un environnement FANET basé sur le protocole P-OLSR. La démarche repose sur l'exploitation de Google Colab en tant qu'environnement de développement, l'intégration de Google Drive pour la gestion des données et des ressources, ainsi que la mise en œuvre détaillée du processus d'apprentissage profond. Les sous-sections suivantes présentent respectivement la configuration de l'environnement, l'accès aux données via Google Drive, puis l'ensemble des étapes nécessaires à la construction et à l'entraînement du modèle GRU.

4.5.1 Configuration de l'Environnement Google Colab

Google Colaboratory, souvent abrégé en Colab, est une plateforme cloud développée par Google pour favoriser l'apprentissage et la recherche en intelligence artificielle, notamment en apprentissage automatique . Basée sur la technologie des Jupyter Notebooks, elle permet aux utilisateurs de rédiger, exécuter et partager du code dans un environnement interactif, semblable à Google Docs. L'un des principaux avantages de Colab est qu'il fournit un environnement préconfiguré avec les bibliothèques couramment utilisées en IA, telles que TensorFlow, Keras, ou Matplotlib, tout en offrant un accès gratuit à des unités de traitement graphique (GPU) via l'infrastructure cloud de Google. Cela permet aux étudiants, chercheurs et développeurs d'expérimenter des modèles complexes sans avoir à disposer de matériel spécialisé [2] [42].

4.5.2 Intégration de Google Drive

Pour simplifier l'accès aux données et la sauvegarde des modèles formés, une intégration de Google Drive a été effectuée au sein de l'environnement Google Colab. L'installation du disque permet une lecture et écriture directement dans l'espace de stockage personnel, facilitant le transfert de grandes quantités de données et la conservation durable des modèles. Cette intégration facilite aussi le partage et la gestion des

2. Consulté le 18 avril 2025, à partir du site : `https://colab.research.google.com/`

ressources tout au long du processus de développement du projet. [3]

4.5.3 Évaluation des performances

L'évaluation des performances du modèle GRU a été réalisée à l'aide de plusieurs métriques adaptées à la nature du dataset FANET Weighted ETX. Ces métriques permettent d'analyser la capacité du modèle à prédire avec précision la stabilité des liens de communication :

1. **Accuracy :**

   Cette métrique mesure la proportion de prédictions correctes (positives et négatives) parmi l'ensemble des prédictions. Bien qu'elle soit traditionnellement utilisée pour des tâches de classification globale, dans notre cas, elle permet de savoir dans quelle mesure le modèle identifie correctement les liens stables et instables. Une accuracy élevée indique que le modèle est globalement fiable dans ses prédictions.

2. **Précision :**

   La précision mesure la proportion de liens prédits comme stables qui sont effectivement stables (vrais positifs parmi tous les positifs prédits). Une précision élevée signifie que le modèle produit peu de faux positifs, ce qui est crucial pour éviter de surestimer la stabilité des liens dans les réseaux FANETs.

3. **Loss function :**

   La fonction de perte `binary_crossentropy` est utilisée pendant l'entraînement. Elle permet de surveiller la convergence du modèle dans un contexte de classification binaire, où une diminution continue de la loss indique un apprentissage efficace.

4. **Recall :**

   Le rappel quantifie la capacité du modèle à détecter tous les liens réellement stables. Un rappel élevé montre que le modèle réussit à capturer la majorité des liens stables, minimisant ainsi les faux négatifs, ce qui est essentiel pour maintenir une communication fiable dans un environnement dynamique.

5. **Matrice de confusion :**

3. Consulté le 18 avril 2025, à partir du site : `https://drive.google.com`.

La matrice de confusion est un outil fondamental pour visualiser les performances du modèle en distinguant clairement les différents types de prédictions : vrais positifs (TP), vrais négatifs (TN), faux positifs (FP) et faux négatifs (FN). Elle fournit une vue détaillée sur la capacité du modèle à différencier les liens stables et instables, facilitant ainsi l'analyse des erreurs et l'ajustement du modèle.

4.5.4 Analyse des résultats

Dans cette section, nous analysons les performances de notre modèle GRU à travers plusieurs métriques : la matrice de confusion, la courbe de perte et la courbe de l'Exactitude . Ces éléments nous permettent d'évaluer la capacité du modèle à généraliser, à détecter correctement les classes, et à apprendre efficacement les schémas complexes dans les données.

4.5.4.1 Courbe de l'Exactitude (Accuracy)

La courbe de l'exactitude, illustrée dans la Figure 4.6, montre l'évolution de la précision du modèle sur les ensembles d'entraînement et de validation au fil des époques. On observe une amélioration progressive de l'exactitude sur les deux ensembles. L'exactitude de validation est généralement supérieure à celle d'entraînement, ce qui indique une bonne capacité de généralisation du modèle. Malgré quelques fluctuations sur la courbe de validation, les performances restent stables et élevées. Cette analyse doit être complétée par l'étude de la matrice de confusion afin d'évaluer plus précisément la détection des différentes classes, en particulier la classe minoritaire.

4.5.4.2 Analyse de la Courbe de Perte (Loss)

La Figure 4.7 montre l'évolution de la fonction de perte (Binary Crossentropy) au cours des 50 époques pour les ensembles d'entraînement et de validation :

1. La perte d'entraînement diminue de manière régulière, indiquant que le modèle apprend efficacement les caractéristiques du jeu de données.
2. La perte de validation suit une tendance similaire, restant proche de la perte d'entraînement tout au long du processus.

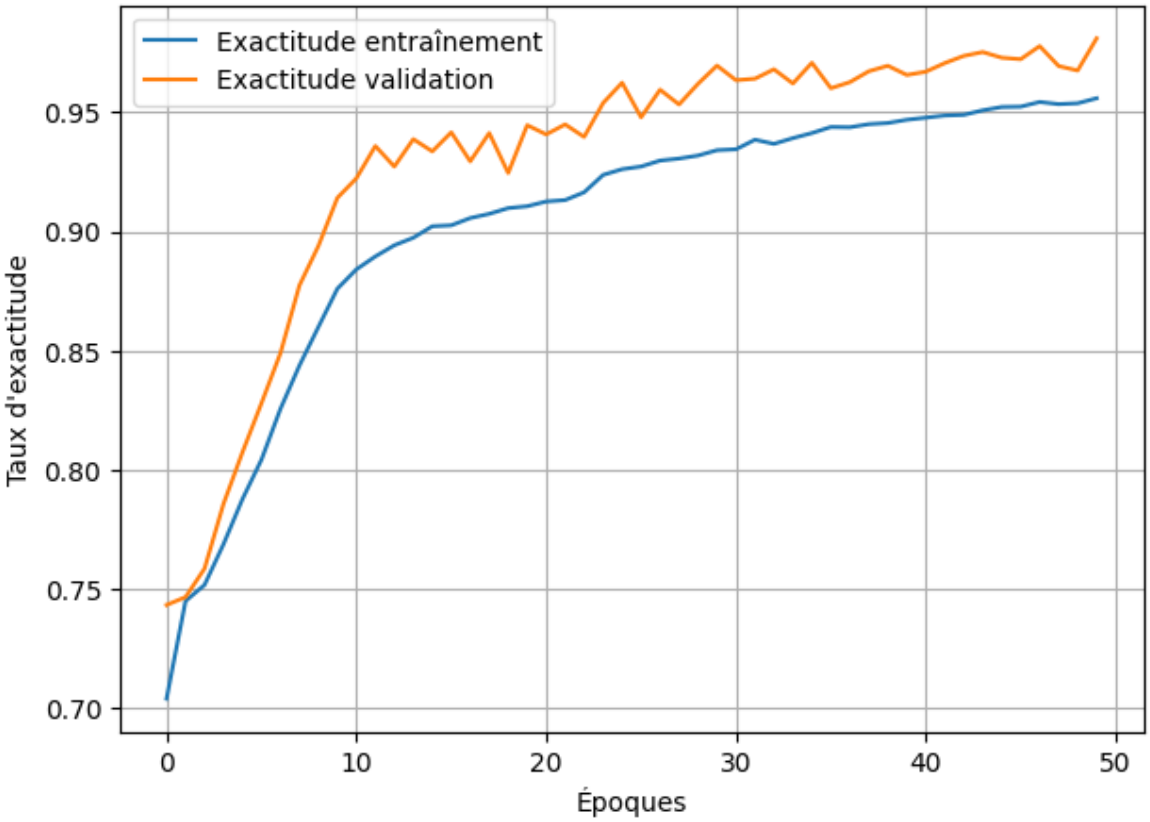

FIGURE 4.6 – Évolution de l'exactitude (accuracy) d'entraînement et de validation en fonction des époques.

3. L'écart entre les deux pertes est faible, ce qui signifie que le modèle ne souffre pas de surajustement (overfitting).

Ces observations montrent que le modèle est capable de généraliser correctement sur des données nouvelles tout en maintenant une bonne performance pendant l'apprentissage.

4.5.4.3 Analyse de la Matrice de Confusion

La matrice de confusion, illustrée dans la Figure 4.8, fournit une évaluation détaillée de la performance du modèle GRU en termes de classification :

1. **Vrais négatifs (classe 0 correctement prédite)** : 10 572 échantillons ont été correctement classés comme appartenant à la classe 0.
2. **Faux positifs (classe 1 prédite alors que c'était 0)** : 151 échantillons ont été incorrectement classés comme appartenant à la classe 1.
3. **Faux négatifs (classe 0 prédite alors que c'était 1)** : 150 échantillons ont

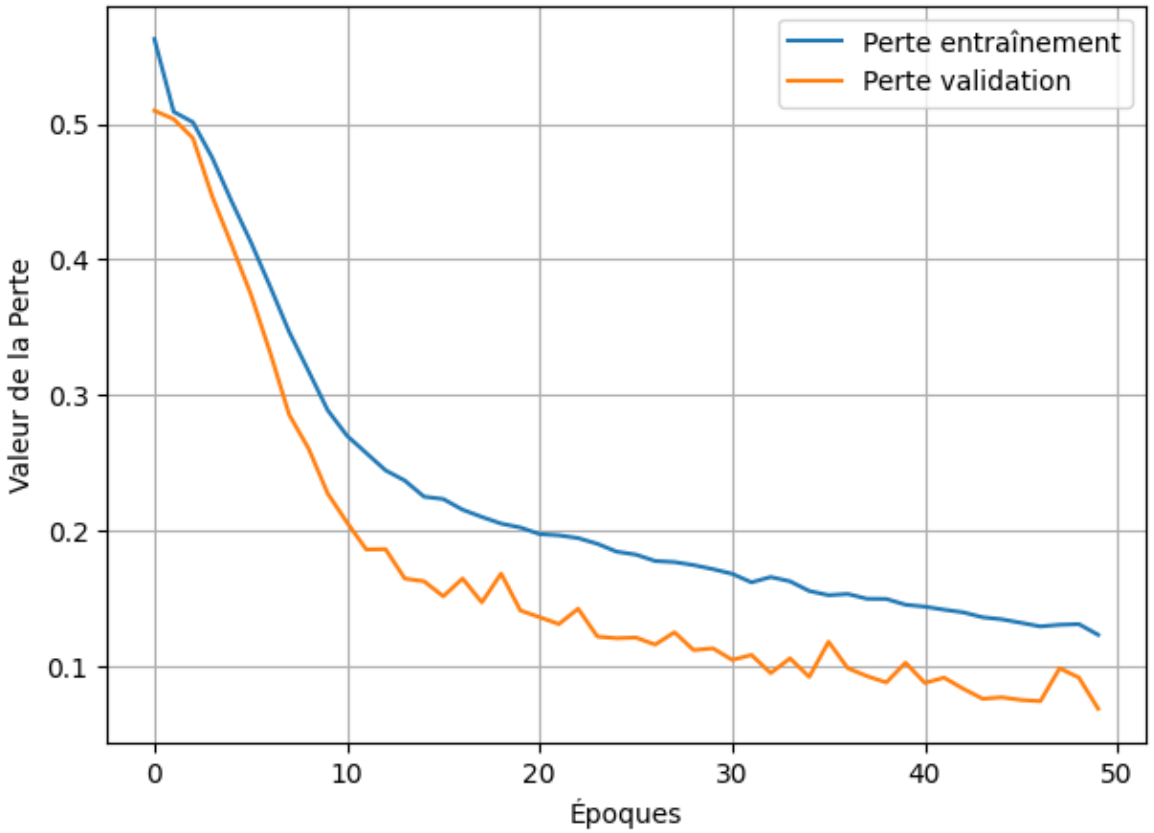

FIGURE 4.7 – Évolution de la perte (loss) d'entraînement et de validation en fonction des époques.

été incorrectement classés comme appartenant à la classe 0.

4. **Vrais positifs (classe 1 correctement prédite)** : 9 127 échantillons ont été correctement classés comme appartenant à la classe 1.

On constate que le modèle présente une très bonne capacité à distinguer les deux classes. Le faible nombre d'erreurs (151 faux positifs et 150 faux négatifs) par rapport au nombre total d'échantillons confirme une excellente performance, avec une précision et un rappel élevés pour les deux classes.

Le Tableau 4.3 présente les résultats finaux de l'évaluation du modèle GRU sur le jeu de test. Les performances obtenues, notamment une accuracy de 98.58 %, une précision de 98.37 % et un recall de 98.38 %, illustrent l'efficacité du modèle dans la prédiction de la stabilité des liens dans le réseau FANET. La faible valeur de la loss (0.0612) confirme également une bonne capacité de généralisation du modèle.

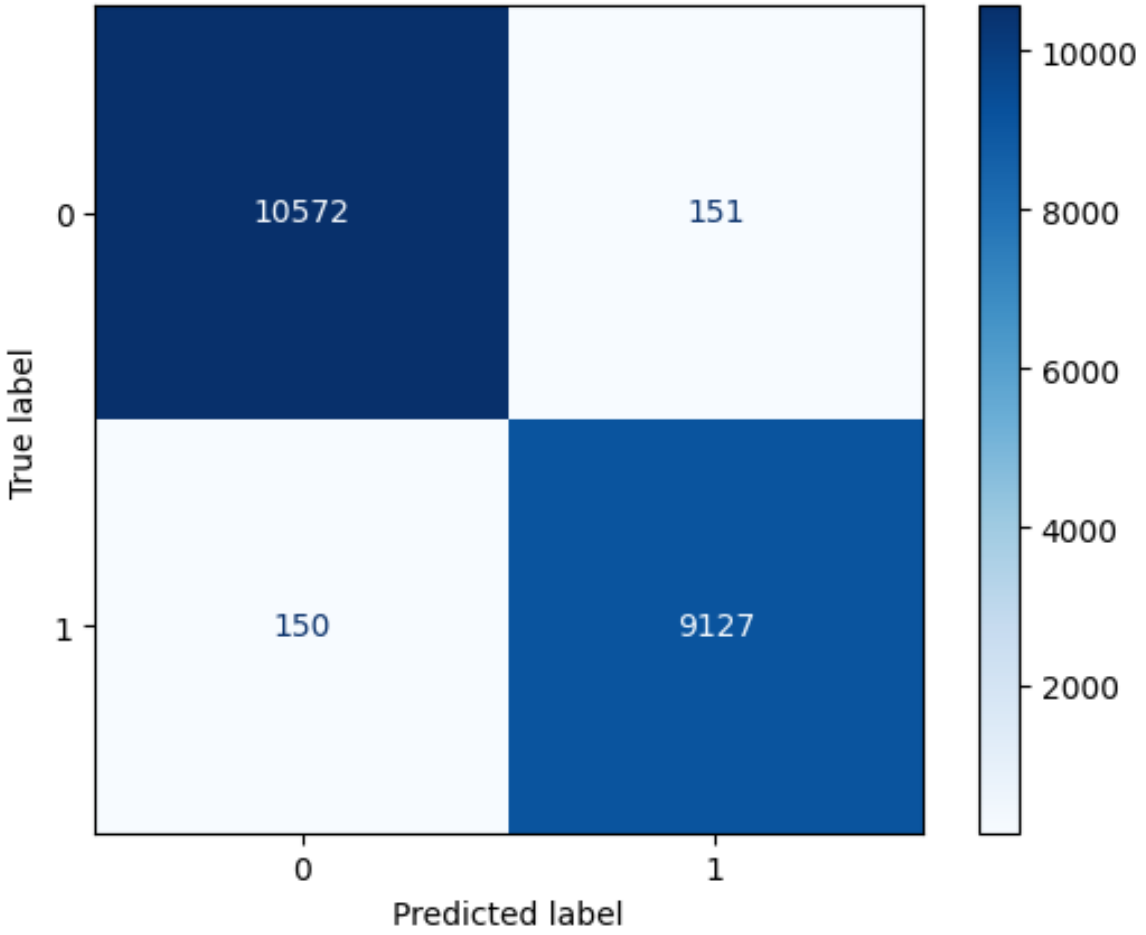

FIGURE 4.8 – Matrice de confusion du modèle proposé.

Métrique	**Valeur**
Exactitude (Accuracy) sur le jeu de test	98.58 %
Perte (Loss) sur le jeu de test	0.0612
Précision sur le jeu de test	98.37 %
Recall sur le jeu de test	98.38 %

TABLE 4.3 – Performance finale du modèle GRU sur les données de test.

4.6 Mise en œuvre du modèle GRU dans un réseau de drones

Cette section présente la mise en œuvre d'une simulation réaliste d'un réseau de drones intégrant un modèle de deep learning pour optimiser la stabilité du routage. Plus précisément, un modèle GRU est utilisé pour prédire l'évolution de la métrique ETX dans le protocole P-OLSR, permettant d'estimer à l'avance la qualité future des liens inter-drones en tenant compte de leur mobilité dynamique, et ainsi d'améliorer la

sélection des routes.

4.6.1 Paramètres de simulation

Le scénario de simulation implique 50 drones évoluant dans un environnement tridimensionnel de 200 x 200 x 100 m^3, selon un modèle de mobilité réaliste de type `Random Waypoint 3D`. Les positions et vitesses des UAVs sont actualisées à chaque intervalle de temps, simulant ainsi un comportement dynamique fidèle à un réseau FANET réel.

Un modèle GRU, préalablement développé, entraîné et sauvegardé sous le format `best_model.h5`, est déployé au sein de la simulation pour prédire l'évolution de la métrique ETX. Ce modèle est hébergé dans la station de base fixe, qui centralise les données et effectue les prédictions sur la qualité des liens inter-drones pour les itérations futures. Cette approche permet une prise de décision proactive dans la sélection des routes, améliorant ainsi la stabilité du routage.

Les principaux paramètres de cette simulation, conformes aux recommandations de la littérature spécialisée sur les protocoles FANET, sont synthétisés dans le Tableau 4.4.

Paramètre	**Valeur**
Nombre de drones	50
Zone de simulation 3D	200 x 200 x 100 m^3
Modèle de mobilité utilisé	Random Waypoint 3D
Énergie initiale	2 j
Vitesse moyenne des drones (v)	5 m/s
Rayon de communication (R)	60 m
Temps de simulation	1200 s
Intervalle Hello (HI)	0.5 s
Pondération de vitesse dans l'ETX (β)	0.2
Probabilité de réception (α)	0.2
Vitesse relative (γ)	0.08
Protocole de routage	P-OLSR-GRU
Modèle de prédiction	GRU

TABLE 4.4 – Paramètres de simulation.

4.6.2 Configuration initiale du réseau FANET

Cette section décrit la configuration initiale du réseau simulé, en présentant d'abord les positions et vitesses des drones, puis la topologie spatiale dans l'environnement 3D.

4.6.2.1 Positions et vitesses initiales des drones

La Figure 4.9 illustre les positions et vitesses initiales de cinq drones. Chaque drone est défini par ses coordonnées spatiales (X, Y, Z) et ses composantes de vitesse (V_x, V_y, V_z), qui influencent fortement la dynamique du réseau FANET.

— **Drone 0** : position (37.22, 160.53, 51.24), vitesse (0.89, 0.16, 2.84) — mouvement principalement vertical.

— **Drone 1** : position (96.59, 26.70, 17.25), vitesse élevée (2.68, 3.62, 4.46) sur les axes X et Z.

— **Drone 2** : position (145.59, 99.29, 49.32), vitesse principalement en Y (3.68), stabilité sur les autres axes.

— **Drone 3** : position (145.90, 153.10, 84.30), vitesse la plus élevée en Y (4.11), déplacement latéral rapide.

— **Drone 4** : position (122.05, 27.07, 77.62), vitesse significative en X (3.58) et composante verticale modérée (2.41).

Ces conditions initiales simulent un environnement hétérogène et dynamique, propice à l'évaluation de la robustesse des protocoles face à des mouvements complexes.

Positions et vitesses initiales des drones :

	Drone_ID	Pos_X	Pos_Y	Pos_Z	Vel_X	Vel_Y	Vel_Z
0	0	37.220284	160.528662	51.236820	0.894944	0.161777	2.842803
1	1	96.593774	26.695994	17.254136	2.684258	3.624833	4.462264
2	2	145.587861	99.292230	49.316563	0.489092	3.684333	1.018319
3	3	145.901646	153.102580	24.301735	4.305068	4.113411	2.749889
4	4	122.045030	27.070816	77.623758	3.580190	1.640316	2.408723

FIGURE 4.9 – Positions et vitesses initiales des drones au début de la simulation.

4.6.2.2 Topologie initiale du réseau

La Figure 4.10 montre la topologie initiale du réseau FANET dans un espace tridimensionnel, avec plusieurs drones spatialement distribués autour d'une station de base fixe (antenne au sol). Cette configuration reflète un scénario réaliste, facilitant l'analyse de la connectivité, de la portée des communications et de l'efficacité des protocoles de routage.

Cette visualisation constitue un repère pour étudier la stabilité des liens et la prédiction proactive des métriques de qualité de service.

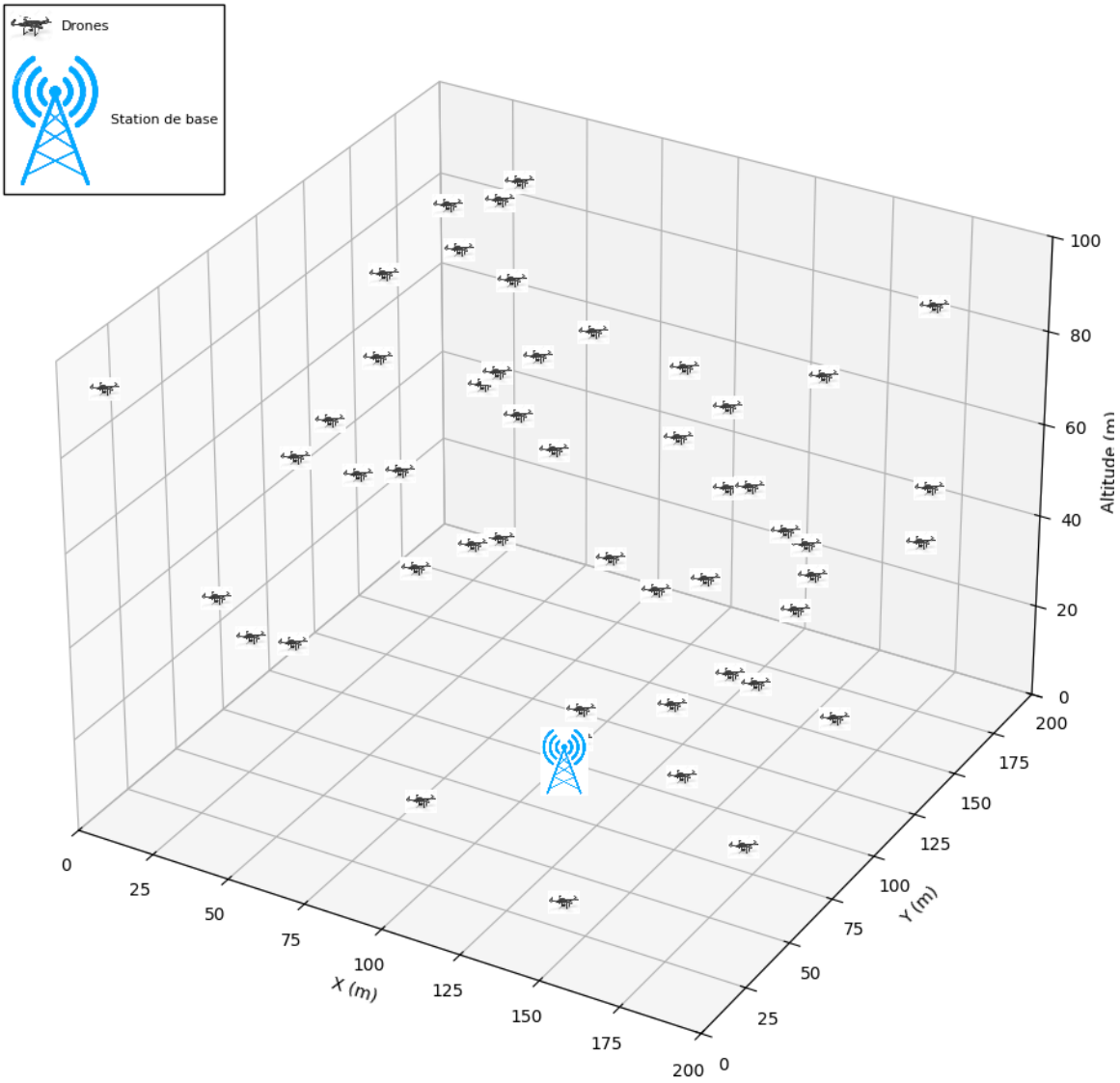

FIGURE 4.10 – Topologie 3D initiale du réseau FANET.

4.6.3 Métriques de performance utilisées

L'approche proposée, appelée P-OLSR-GRU , est comparée aux protocoles classiques OLSR et P-OLSR à travers plusieurs métriques avancées permettant d'évaluer la qualité du routage dans un réseau de drones :

1. **Taux de stabilité des liens** : proportion des connexions stables dans le temps, reflétant la continuité des communications.
2. **Taux de perte de paquets** : proportion des paquets non reçus, indiquant la fiabilité du protocole.
3. **Taux de transmission utile (goodput)** : quantité de données utiles transmises sur les liens actifs, indicateur direct de la performance.
4. **Résilience (écart-type de la stabilité)** : mesure la régularité de la stabilité des liens, un faible écart-type indique une bonne résistance aux fluctuations du réseau.
5. **Taux de liens actifs** : proportion de liens effectivement établis parmi les connexions possibles, traduisant la densité du maillage réseau.

4.7 Évaluation des Performances

Cette section présente une analyse détaillée des performances du protocole proposé P-OLSR-GRU, qui intègre un modèle de `Deep Learning` basé sur des réseaux de neurones à portes pour l'optimisation du routage dans les réseaux aériens de type FANET. L'objectif principal est d'évaluer l'impact de cette intégration sur la qualité de service, en mettant en évidence les améliorations apportées en termes de stabilité des liens, fiabilité des transmissions, résilience du réseau et efficacité globale du routage.

4.7.1 Statistiques des liens évalués

La Figure 4.11 et le Tableau 4.5 illustrent les résultats de l'évaluation de la stabilité des liens dans le réseau. Sur un total de 255 liens analysés, 240 soit (94.12%) ont été identifiés comme stables, tandis que seulement 15 liens (5.88%) étaient instables. De plus, 234 liens ont permis une transmission réussie, représentant (91.76%) des cas.

Le graphique en barres met en évidence ces résultats visuellement, montrant la prédominance des liens stables (en vert) et réussis (en jaune), comparés au faible nombre de liens non stables (en bleu). Ces résultats démontrent l'efficacité du mécanisme de prédiction mis en place pour garantir la fiabilité des communications dans le réseau.

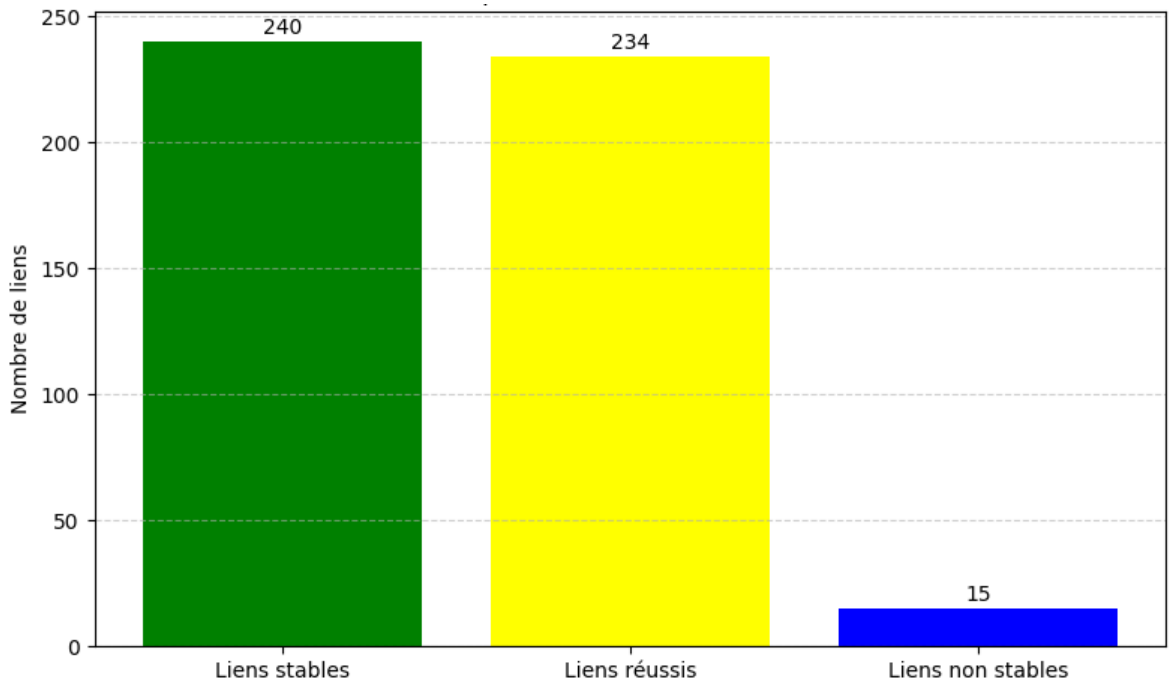

FIGURE 4.11 – Répartition des liens évalués.

Statistique	**Valeur**
Total des liens évalués	255
Liens stables prédits	240 (94.12%)
Liens non stables	15 (5.88%)
Liens avec transmission réussie	234 (91.76%)

TABLE 4.5 – Statistiques des liens évalués.

4.7.2 Analyse Comparative des Protocoles de Routage

La Figure 4.12 représente un graphique en barres comparant les performances de trois protocoles de routage : OLSR, P-OLSR et P-OLSR-GRU, selon plusieurs métriques clés. Ces métriques incluent le taux de stabilité, le taux de perte, la transmission utile, la résilience (mesurée par l'écart-type de stabilité), et le nombre de liens actifs. Cette comparaison permet d'évaluer en détail l'efficacité de chaque protocole dans le contexte des réseaux ad hoc volants :

1. **Taux de stabilité :** Le protocole P-OLSR-GRU se distingue avec le meilleur taux de stabilité atteignant (98.8 %), indiquant une capacité maximale à maintenir des liens durables. Il est suivi par P-OLSR environ(58 %) et OLSR environ (40 %), ce qui montre que les améliorations basées sur GRU apportent un gain significatif.
2. **Taux de perte :** Le protocole P-OLSR-GRU affiche le taux de perte le plus faible, avoisinant (3 %), suivi par P-OLSR avec environ (22 %). Le protocole OLSR présente le taux le plus élevé, avec près de (30 %). Cela met en évidence l'efficacité du modèle GRU à anticiper les ruptures de liens.
3. **Transmission utile :** P-OLSR-GRU obtient le meilleur taux de transmission utile avec près de (97 %), suivi par P-OLSR (67 %) et OLSR (60 %). Cela reflète une exploitation plus efficace du canal de communication par le modèle GRU.
4. **Résilience (écart-type de stabilité) :** Le protocole P-OLSR-GRU présente une meilleure régularité avec un écart-type faible environ (6–7 %), légèrement meilleur que P-OLSR environ (9 %), tandis que OLSR montre une plus grande instabilité avec un écart-type proche de (18 %).
5. **Liens actifs :** P-OLSR-GRU enregistre la plus forte proportion de liens actifs avec environ (92 %), suivi par P-OLSR (66 %) et OLSR (55 %), ce qui reflète une meilleure densité de connectivité dans le réseau.

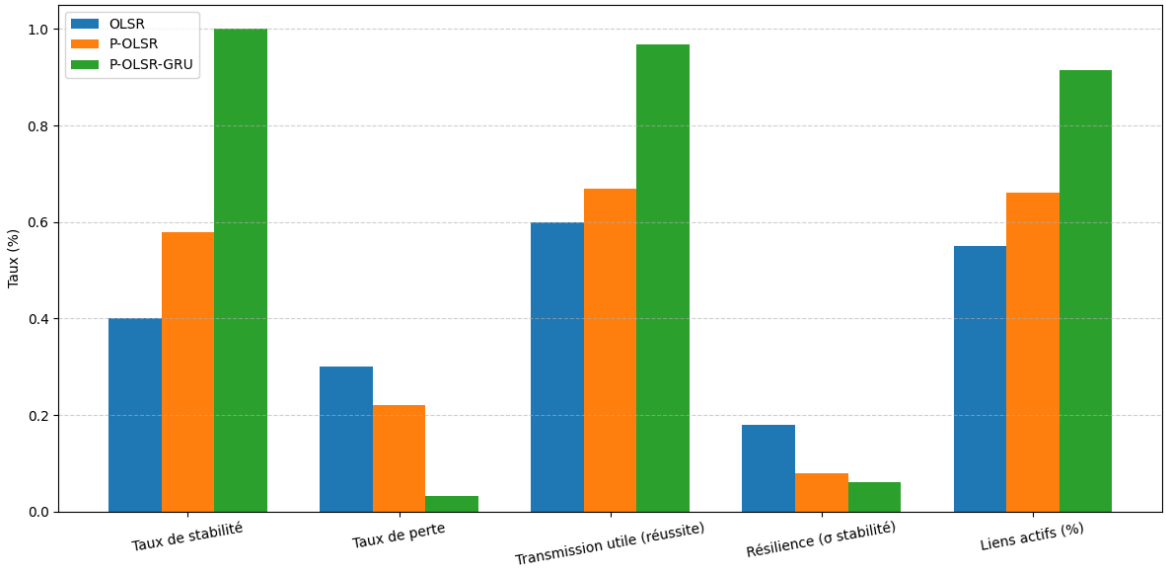

FIGURE 4.12 – Comparaison des performances des protocoles OLSR, P-OLSR et P-OLSR-GRU selon plusieurs métriques clés.

4.8 Conclusion

Ce chapitre a introduit le protocole P-OLSR-GRU, une extension intelligente du protocole P-OLSR, intégrant un modèle GRU pour la prédiction proactive de la métrique ETX. En tirant parti des capacités de l'apprentissage profond, cette approche permet d'anticiper de manière plus précise les fluctuations de la qualité des liens dans les réseaux FANETs, souvent soumis à une dynamique élevée. Après avoir détaillé la méthodologie de collecte, de préparation des données, ainsi que la conception et l'entraînement du modèle, nous avons procédé à une évaluation approfondie de ses performances. Les résultats obtenus démontrent une amélioration significative en termes de stabilité des routes, de fiabilité des communications et de qualité de service globale. Ces avancées soulignent l'efficacité du couplage entre intelligence artificielle et protocoles de routage dans des environnements réseau mobiles et contraints.

Conclusion Générale

Dans un monde de plus en plus interconnecté, les Flying Ad Hoc Networks représentent une solution innovante pour les communications autonomes entre drones. Leur mobilité tridimensionnelle et leur capacité à opérer sans infrastructure fixe en font des outils essentiels pour des missions critiques, allant de la surveillance à la gestion de crises. Toutefois, leur efficacité dépend fortement de la stabilité des liens de communication, souvent mise à mal par la nature dynamique et instable de ces réseaux.

Notre travail a débuté par une exploration détaillée de l'architecture des drones au sein des FANETs, en identifiant leurs composants, leurs capacités et les contraintes qu'ils imposent en matière de routage et de qualité de service. Nous avons ensuite étudié les protocoles de routage existants, mettant en évidence leurs limites dans des environnements hautement mobiles.

Face à ces défis, nous avons proposé une approche intelligente baptisée P-OLSR-GRU, combinant le protocole P-OLSR avec un modèle de réseaux de neurones récurrents de type GRU. L'objectif est de prédire en temps réel la stabilité des liens entre drones en distinguant les connexions stables de celles susceptibles de se rompre afin d'adapter dynamiquement les chemins de communication et renforcer la fiabilité globale du réseau.

Les performances finales du modèle sur le jeu de test montrent une grande efficacité prédictive, avec une accuracy de 98.58%, une précision de 98.37%, un rappel de 98.38% et une perte de 0.0612, démontrant ainsi une excellente capacité de généralisation.En simulation, le protocole P-OLSR-GRU présente des performances équilibrées face aux protocoles classiques OLSR et P-OLSR, avec un taux de stabilité des liens de 98.8 %, une perte moyenne réduite à 3 %, une transmission utile atteignant 97 %, une résilience de 6 %, et environ 92 % de liens actifs. Ces résultats mettent en évidence la capacité

du modèle GRU à améliorer l'anticipation des ruptures de liens et à maintenir une connectivité efficace dans les FANETs, tout en ouvrant la voie à des optimisations futures pour renforcer davantage la stabilité et la fiabilité du routage.

En conclusion, ce mémoire apporte une contribution significative à l'amélioration du routage dans les FANETs en introduisant une méthode prédictive basée sur l'apprentissage profond. Cette approche permet d'anticiper les variations topologiques et de renforcer la fiabilité des communications dans des environnements volatils et dynamiques.

Les perspectives offertes par ce travail sont prometteuses. Il serait pertinent, dans les travaux futurs, d'explorer l'intégration de l'apprentissage par renforcement profond, permettant au protocole de s'adapter continuellement à des contextes changeants. De plus, le recours à des architectures hybrides combinant des réseaux neuronaux convolutifs pour l'analyse spatiale et des GRU ou LSTM pour la dimension temporelle pourrait améliorer encore davantage la précision des prédictions.

Enfin, la prise en compte d'enjeux tels que la gestion adaptative de l'énergie, l'optimisation conjointe des ressources réseau et des trajectoires des drones, ou encore la mise en œuvre de systèmes distribués et tolérants aux pannes, constitue des axes de recherche essentiels pour faire évoluer les FANETs vers des systèmes autonomes, intelligents et résilients, adaptés aux contraintes du monde réel .

Bibliographie

[1] Jan Lansky, Saqib Ali, Amir Masoud Rahmani, Mohammad Sadegh Yousefpoor, Efat Yousefpoor, Faheem Khan, and Mehdi Hosseinzadeh. Reinforcement Learning-Based Routing Protocols in Flying Ad Hoc Networks (FANET) : A Review. *Mathematics*, 10(16) :3017, August 2022.

[2] Abderrezzag ZIOU. *Réalisation d'un système de suivi d'objets basé sur les Drones*. Mémoire de fin d'études, Master en Informatique, Université de Guelma, Faculté des Mathématiques, d'Informatique et des Sciences de la matière, Département d'Informatique, October 2020.

[3] Jean-Aimé Maxa. *Architecture de communication sécurisée d'une flotte de drones*. PhD thesis, Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), June 2017.

[4] Ilyas Hadjou. *Evaluation and Comparison Study of Transport Layer Protocols for Data Transmission in Flying Ad-Hoc Networks (FANETs)*. @mastersthesis{Hadjou2020,, Larbi Ben M'Hidi University, Oum El Bouaghi, Algeria, 2020.

[5] Qianqian Sang, Honghai Wu, Ling Xing, and Ping Xie. Review and Comparison of Emerging Routing Protocols in Flying Ad Hoc Networks. *Symmetry*, 12(6) :971, June 2020.

[6] Ablaoui Houssem Eddine and Bentayeb Nadine Ouidiane. Un nouveau protocole de routage à état de lien basé sur l'énergie et le débit pour les réseaux FANETs. Master's thesis, Université de Mohamed El Bachir El Ibrahimi de Borj Bou Arréridj, Borj Bou Arréridj, Algeria, 2024.

[7] A. B. Makar, K. E. McMartin, M. Palese, and T. R. Tephly. Formate assay in body fluids : application in methanol poisoning. *Biochemical Medicine*, 13(2) :117–126, June 1975.

[8] Asrar Ahmed Baktayan, Ammar Thabit Zahary, Axel Sikora, and Dominik Welte. Computational offloading into UAV swarm networks : a systematic literature review. *EURASIP Journal on Wireless Communications and Networking*, 2024(1) :69, September 2024.

[9] Sifat Rezwan and Wooyeol Choi. A Survey on Applications of Reinforcement Learning in Flying Ad-Hoc Networks. *Electronics*, 10(4) :449, February 2021.

[10] Moazzam Ali, Adil Idress, and Jawwad Ibrahim. FANET :-Communication Architecture and Routing Protocols A Review. *International Journal of Computer Science and Network Security*, 24(5) :181–190, May 2024.

[11] Ablaoui Houssem Eddine and Bentayeb Nadine Ouidiane. Un nouveau protocole de routage à état de lien basé sur l'énergie et le débit pour les réseaux FANETs. Master's thesis, Université de Mohamed El Bachir El Ibrahimi de Borj Bou Arréridj, Borj Bou Arréridj, Algeria, 2023.

[12] Omar Sami Oubbati, Mohammed Atiquzzaman, Pascal Lorenz, Md. Hasan Tareque, and Md. Shohrab Hossain. Routing in Flying Ad Hoc Networks : Survey, Constraints, and Future Challenge Perspectives. *IEEE Access*, 7 :81057–81105, 2019.

[13] Rostam Shirani, Marc St-Hilaire, Thomas Kunz, Yifeng Zhou, Jun Li, and Louise Lamont. Combined Reactive-Geographic routing for Unmanned Aeronautical Adhoc Networks. In *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 820–826, Limassol, Cyprus, August 2012. IEEE.

[14] Akram KOUT. *Contributions à la résolution du problème de routage dans les réseaux mobiles ad hoc par les méthodes bio-inspirées*. Doctoral Thesis, Université Abdelhamid Mehri - Constantine 2, Constantine, Algeria, 2017.

[15] Jingjing Yao and Nirwan Ansari. QoS-Aware Machine Learning Task Offloading and Power Control in Internet of Drones. *IEEE Internet of Things Journal*, 10(7) :6100–6110, April 2023.

[16] Muhammad Fahad Khan, Kok-Lim Alvin Yau, Rafidah Md Noor, and Muhammad Ali Imran. Routing Schemes in FANETs : A Survey. *Sensors*, 20(1) :38, December 2019.

[17] Ablaoui Houssem Eddine and Bentayeb Nadine Ouidiane. Un nouveau protocole de routage à état de lien basé sur l'énergie et le débit pour les réseaux FANETs. Master's thesis, Université de Mohamed El Bachir El Ibrahimi de Borj Bou Arréridj, Borj Bou Arréridj, Algeria, 2023.

[18] Mehdi Hosseinzadeh, Saqib Ali, Amir Masoud Rahmani, Jan Lansky, Vladimir Nulicek, Mohammad Sadegh Yousefpoor, Efat Yousefpoor, Aso Darwesh, and Sang-Woong Lee. A smart filtering-based adaptive optimized link state routing protocol in flying ad hoc networks for traffic monitoring. *Journal of King Saud University - Computer and Information Sciences*, 36(4) :102034, April 2024.

[19] Inam Ullah Khan, Muhammad Abul Hassan, Muhammad Fayaz, Jeonghwan Gwak, and Muhammad Adnan Aziz. Improved Sequencing Heuristic DSDV Protocol Using Nomadic Mobility Model for FANETS. *Computers, Materials & Continua*, 70(2) :3653–3666, 2022.

[20] Stefano Rosati, Karol Kruzelecki, Gregoire Heitz, Dario Floreano, and Bixio Rimoldi. Dynamic Routing for Flying Ad Hoc Networks. *IEEE Transactions on Vehicular Technology*, 65(3) :1690–1700, March 2016.

[21] Anas AlKhatieb and Emad Felemban. Performance Evaluation of Ad Hoc Routing Protocols in (FANETs). In *2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*, pages 1–6, Al Madinah Al Munawwarah, Saudi Arabia, February 2020. IEEE.

[22] Huamin Wang, Yongfu Li, Yubing Zhang, Tiancong Huang, and Yang Jiang. Arithmetic Optimization AOMDV Routing Protocol for FANETs. *Sensors*, 23(17) :7550, August 2023.

[23] Hua Yang and Zhiyong Liu. An optimization routing protocol for FANETs. *EURASIP Journal on Wireless Communications and Networking*, 2019(1) :120, December 2019.

[24] Kwan Hui Lim and Amitava Datta. Enhancing the TORA protocol using network localization and selective node participation. In *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pages 1503–1508, Sydney, Australia, September 2012. IEEE.

[25] Saifullah Khan, Muhammad Zahid Khan, Pervez Khan, Gulzar Mehmood, Ajab Khan, and Muhammad Fayaz. An Ant-Hocnet Routing Protocol Based on Opti-

mized Fuzzy Logic for Swarm of UAVs in FANET. *Wireless Communications and Mobile Computing*, 2022 :1–12, June 2022.

[26] Ch. Naveen Kumar Reddy and M. Anusha. Hybrid Intelligent Routing with Optimized Learning (HIROL) for Adaptive Routing Topology management in FANETs, 2024. Version Number : 1.

[27] H. Badis and K. Al Agha. An efficient QOLSR extension protocol for QoS in ad hoc networks. In *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, volume 4, pages 2650–2653, Los Angeles, CA, USA, 2004. IEEE.

[28] Mehdi Hosseinzadeh, Saqib Ali, Liliana Ionescu-Feleaga, Bogdan-Stefan Ionescu, Mohammad Sadegh Yousefpoor, Efat Yousefpoor, Omed Hassan Ahmed, Amir Masoud Rahmani, and Asif Mehmood. A novel Q-learning-based routing scheme using an intelligent filtering algorithm for flying ad hoc networks (FANETs). *Journal of King Saud University - Computer and Information Sciences*, 35(10) :101817, December 2023.

[29] Xu Zhen and Yang Wenzhong. Bandwidth-aware routing for TDMA-based mobile ad hoc networks. In *The International Conference on Information Networking 2013 (ICOIN)*, pages 637–642, Bangkok, January 2013. IEEE.

[30] Mahadev A. Gawas, Lucy J. Gudino, and K. R. Anupama. Cross layer congestion aware multi rate multi path routing protocol for ad hoc network. In *2015 International Conference on Signal Processing and Communication (ICSC)*, pages 88–93, Noida, India, March 2015. IEEE.

[31] Hind Alwan and Anjali Agarwal. MQoSR : A Multiobjective QoS Routing Protocol for Wireless Sensor Networks. *ISRN Sensor Networks*, 2013 :1–12, June 2013.

[32] H. Alwan and A. Agarwal. Multi-objective QoS routing for wireless sensor networks. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 1074–1079, San Diego, CA, January 2013. IEEE.

[33] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. Technical Report RFC2309, RFC Editor, April 1998.

[34] Godwin Onyekachi Ugwu, Udora Nwabuoku Nwawelu, Mamilus Aginwa Ahaneku, and Cosmas Ikechukwu Ani. Effect of service differentiation on QoS in IEEE 802.11e enhanced distributed channel access : a simulation approach. *Journal of Engineering and Applied Science*, 69(1) :1, December 2022.

[35] Katalin Tarnay, Gusztáv Adamis, Tibor Dulai, and Xiaoge Xu, editors. *Advanced Communication Protocol Technologies : Solutions, Methods, and Applications*. Advances in Wireless Technologies and Telecommunication. IGI Global, 2011.

[36] Arslan Musaddiq, Zulqar Nain, Yazdan Ahmad Qadri, Rashid Ali, and Sung Won Kim. Reinforcement Learning-Enabled Cross-Layer Optimization for Low-Power and Lossy Networks under Heterogeneous Traffic Patterns. *Sensors*, 20(15) :4158, July 2020.

[37] Arnau Rovira-Sugranes, Abolfazl Razi, Fatemeh Afghah, and Jacob Chakareski. A review of AI-enabled routing protocols for UAV networks : Trends, challenges, and future outlook. *Ad Hoc Networks*, 130 :102790, May 2022.

[38] Fadhila Tlili, Samiha Ayed, and Lamia Chaari Fourati. Advancing UAV security with artificial intelligence : A comprehensive survey of techniques and future directions. *Internet of Things*, 27 :101281, October 2024.

[39] Jingjing Yao and Nirwan Ansari. QoS-Aware Machine Learning Task Offloading and Power Control in Internet of Drones. *IEEE Internet of Things Journal*, 10(7) :6100–6110, April 2023.

[40] Stefano Rosati, Karol Kruzelecki, Gregoire Heitz, Dario Floreano, and Bixio Rimoldi. Dynamic Routing for Flying Ad Hoc Networks. *IEEE Transactions on Vehicular Technology*, 65(3) :1690–1700, March 2016.

[41] Ablaoui Houssem Eddine and Bentayeb Nadine Ouidiane. Un nouveau protocole de routage à état de lien basé sur l'énergie et le débit pour les réseaux FANETs. Master's thesis, Université de Mohamed El Bachir El Ibrahimi de Borj Bou Arréridj, Borj Bou Arréridj, Algeria, 2024.

[42] Tiago Carneiro, Raul Victor Medeiros Da Nobrega, Thiago Nepomuceno, Gui-Bin Bian, Victor Hugo C. De Albuquerque, and Pedro Pedrosa Reboucas Filho. Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, 6 :61677–61685, 2018.

Annexe

Dans cette annexe, nous présentons les codes sources utilisés pour l'implémentation et la simulation de notre approche basée sur le modèle GRU. Ces codes couvrent l'ensemble des étapes nécessaires, notamment le prétraitement des données, la conception de l'architecture du GRU , ainsi que les phases d'entraînement et d'évaluation. L'ensemble du processus a été réalisé dans l'environnement Google Colab, ce qui permet une exécution interactive et reproductible des expériences.

I) Construction et d'Entraînement du Modèle GRU

Dans cette partie, nous détaillons étape par étape la mise en œuvre du modèle GRU conçu pour anticiper pour prédire la stabilité des liens à partir des observations recueillies via le protocole P-OLSR. Le processus s'effectue en suivant ces différentes phases :

Étape 1 : Importation des bibliothèques nécessaires

Dans cette première étape, nous importons toutes les bibliothèques qui seront utilisées pour le traitement des données, la construction du modèle GRU, et l'évaluation des performances.

```
# Importation des bibliothèques nécessaires
import pandas as pd # Manipulation de données
import numpy as np # Calcul numérique
import matplotlib.pyplot as plt # Visualisation de données
import seaborn as sns # Visualisation avancée
import tensorflow as tf # Deep learning
from tensorflow.keras.models import Sequential # Modèle sé
   quentiel
```

```
from tensorflow.keras.layers import GRU, Dense, Dropout # Couches GRU, Dense et Dropout
from tensorflow.keras.callbacks import EarlyStopping # Arrêt automatique en cas de surapprentissage
from sklearn.model_selection import train_test_split # Division des données
from sklearn.preprocessing import MinMaxScaler # Normalisation des données
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay# Évaluation via matrice de confusion
from google.colab import drive # Accès aux fichiers sur Google Drive
# Confirmation de l'importation réussie
print(" Bibliothèques importées avec succès ")
# Affichage des versions
print("TensorFlow version:", tf.__version__)
print("Pandas version:", pd.__version__)
```

Étape 2 : Connexion à Google Drive et chargement du dataset

Dans cette étape, nous connectons notre environnement d'exécution Colab à Google Drive afin de charger le fichier de données stocké dans notre espace personnel.

```
# Monter Google Drive dans l'environnement Google Colab
drive.mount('/content/drive')
# Définir le chemin du fichier CSV
dataset_path = '/content/drive/MyDrive/fanet_weighted_etx_dataset.csv'
# Charger le dataset CSV dans un DataFrame pandas
df = pd.read_csv(dataset_path)
# Afficher un message pour signaler que l'aperçu du dataset
print("\n Aperçu des 5 premières lignes du dataset :")
# Afficher les 5 premières lignes du DataFrame
display(df.head())
# Afficher un message pour les statistiques descriptives
print("\n Statistiques descriptives :")
```

```
# Afficher les statistiques de base
display(df.describe())
# Afficher un message pour annoncer l'analyse de la distribution de la variable cible
print("\n Distribution de la cible (link_stability) :")
# Créer un graphique pour visualiser la distribution de la variable cible 'link_stability'
sns.countplot(x='link_stability', data=df)
# Ajouter un titre au graphique
plt.title('Distribution de la cible binaire')
# Afficher le graphique
plt.show()
```

Étape 3 : Pré-traitement des données

Cette étape consiste à séparer les données en variables d'entrée (features) et variable cible (à prédire), puis à appliquer une normalisation. Cela est nécessaire pour améliorer la convergence du modèle GRU durant l'apprentissage.

```
# Séparation des données en variables explicatives (X) et variable cible (y)
# X contient toutes les colonnes sauf 'link_stability'
X = df.drop('link_stability', axis=1)
# y contient uniquement la colonne 'link_stability'
y = df['link_stability']
# Normalisation des données avec MinMaxScaler
# Le MinMaxScaler va transformer toutes les colonnes pour qu'elles soient entre 0 et 1
scaler = MinMaxScaler()
# Application du scaler : ajustement (fit) et transformation (transform) des données d'entrée X
X_scaled = scaler.fit_transform(X)
# Affichage d'un message pour signaler que l'extrait des données normalisées va suivre
print("\n Données normalisées (extrait) :")
print(pd.DataFrame(X_scaled, columns=X.columns).head())
```

Étape 4 : Création des séquences pour GRU

Dans cette étape, les données normalisées sont transformées en séquences temporelles adaptées à l'entraînement d'un modèle GRU. Chaque séquence est construite sur une fenêtre glissante de longueur fixe, permettant au modèle d'apprendre l'évolution des caractéristiques sur plusieurs pas de temps.

```
# === Création de Séquences Temporelles pour GRU ===
def create_sequences(X, y, seq_length=10):
    """
    Génère des séquences temporelles pour l'entra nement d'un
   modèle GRU.

    Args:
        X (array-like) : Données normalisées.
        y (array-like) : Cibles binaires.
        seq_length (int) : Longueur de chaque séquence.

    Returns:
        tuple: (séquences X, cibles y)
    """
    Xs, ys = [], []
    for i in range(len(X) - seq_length):
        Xs.append(X[i:i+seq_length])  # Séquence d'entrée
        ys.append(y.iloc[i + seq_length])  # Cible
   correspondante

    return np.array(Xs), np.array(ys)
# Génération des séquences
X_seq, y_seq = create_sequences(X_scaled, y, seq_length=10)
# Affichage des dimensions des séquences
print("\nDimensions des séquences générées :")
print(f" - X_seq : {X_seq.shape} (échantillons, pas de temps,
   caractéristiques)")
print(f" - y_seq : {y_seq.shape} (échantillons,)")
```

Étape 5 : Split dataset training/test

Dans cette étape, les données sont divisées en un ensemble d'entraînement et un ensemble de test afin de pouvoir évaluer les performances du modèle sur des données jamais vues. Ensuite, les dimensions des ensembles sont adaptées pour être compatibles avec l'entrée attendue par la couche GRU du modèle.

```
# Séparation des données en ensemble d'entra nement et de test
# Utilisation de la fonction train_test_split pour diviser les
   données séquencées
# X_scaled : les entrées , y : la cible
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
    test_size=0.2, random_state=42)# test_size=0.2,# random_state
   =42
#  Affichage des dimensions initiales des ensembles
print("\n Dimensions des ensembles :")
print(f"Training set : {X_train.shape}")  # Dimensions de l'
   ensemble d'entra nement
print(f"Test set : {X_test.shape}")        # Dimensions de l'
   ensemble de test
# Reshape des données pour être compatibles avec l'entrée du mod
   èle GRU
# (échantillons, features)     (échantillons, 1, features)
X_train = np.expand_dims(X_train, axis=1)
X_test = np.expand_dims(X_test, axis=1)
#  Affichage des nouvelles dimensions après reshape
print("\n Dimensions reshaped pour GRU :")
print(f"X_train: {X_train.shape} | X_test: {X_test.shape}")
```

Étape 6 : Définir les couches et détails du modèle GRU

Dans cette étape, l'architecture du modèle séquentiel est définie en empilant une couche GRU, des couches de régularisation Dropout, et des couches Dense. Le but est de capturer les dépendances temporelles dans les données séquentielles, tout en limitant le surapprentissage et en produisant une sortie adaptée à une classification binaire.

```
#  Construction du modèle GRU avec Keras
# Initialisation d'un modèle séquentiel
```

```
model = Sequential()
# Ajout d'une couche GRU
# - input_shape=(X_train.shape[1], X_train.shape[2]) : dimensions d'entrée
# - return_sequences=False : on ne retourne que la dernière sortie
model.add(GRU(units=64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=False))
# Ajout d'une couche de Dropout pour éviter le surapprentissage
# - 0.3 signifie que 30% des neurones seront désactivés aléatoirement pendant l'entra nement
model.add(Dropout(0.3))
# Ajout d'une couche Dense
# - 32 neurones
# - activation='relu' : fonction d'activation ReLU pour introduire de la non-linéarité
model.add(Dense(32, activation='relu'))
# Deuxième couche Dropout pour renforcer la régularisation
model.add(Dropout(0.3))
# Dernière couche Dense pour produire la sortie
# - 1 neurone:(stabilité du lien 0 ou 1)
# - activation='sigmoid' : pour obtenir une probabilité entre 0 et 1
model.add(Dense(1, activation='sigmoid'))
# Affichage du résumé du modèle
model.summary()
```

Étape 7 :Compilation du Modèle

Dans cette étape, le modèle est compilé en spécifiant l'optimiseur, la fonction de perte et les métriques d'évaluation. La compilation configure l'apprentissage du modèle en fonction de la nature du problème, ici une classification binaire de la stabilité du lien.

```
# Compilation du modèle GRU défini précédemment
model.compile(optimizer='adam',  # Optimiseur Adam : très utilis
```

```
    é pour sa rapidité et son adaptativité des taux d'
    apprentissage
              loss='binary_crossentropy',  # Fonction de perte
    adaptée aux problèmes de classification binaire (0 ou 1)
              metrics=['accuracy',                # Taux de
    bonnes prédictions
tf.keras.metrics.Precision(), # Précision : proportion de pré
    dictions positives correctes
tf.keras.metrics.Recall() ])  # Rappel (Recall) : capacité à
    identifier toutes les vraies classes positives
# Message pour confirmer que la compilation a réussi
print(" Modèle compilé avec succès")
```

Étape 8 : Entraînement du modèle

Dans cette étape, le modèle est entraîné sur l'ensemble d'entraînement en utilisant une validation croisée sur 20% des données pour surveiller les performances. Un mécanisme d'arrêt anticipé (EarlyStopping) est utilisé pour éviter le surapprentissage si la perte de validation cesse de s'améliorer.

```
# Définition du callback EarlyStopping
# EarlyStopping arrête l'entra nement automatiquement
early_stop = EarlyStopping( monitor='val_loss',  # On surveille
    la perte sur l'ensemble de validation
                            patience=5, # Si la perte de
    validation ne s'améliore pas pendant 5 epochs , on arrête
                            restore_best_weights=True ) # on
    restaure les poids de l'epoch avec la meilleure val_loss
#  Entra nement du modèle
history = model.fit(
    X_train,          # Données d'entra nement
    y_train,          # Cibles d'entra nement
    validation_split=0.2,  # 20% des données d'entra nement
    epochs=50,        # Nombre maximal d'epochs
    batch_size=128,  # Nombre d'échantillons par batch pendant l
    'entra nement
```

```
    callbacks=[early_stop], # Utilisation de EarlyStopping pour
    éviter l'overfitting
    verbose=1    )      # Affichage du déroulement de l'
    entra nement (1 = barre de progression détaillée)
```

Étape 9 : Visualisation des performances pendant l'entraînement

Dans cette étape, deux courbes sont tracées pour visualiser l'évolution de la perte (loss) et de l'exactitude (accuracy) pendant l'entraînement et la validation. Cela permet d'analyser le comportement du modèle et de détecter un éventuel surapprentissage ou sous-apprentissage.

```
# Affichage des courbes d'apprentissage
plt.figure(figsize=(14,5))
# --Première sous-figure : Évolution de la Perte (Loss) --
plt.subplot(1, 2, 1)  # 1 ligne, 2 colonnes, 1er graphe
plt.plot(history.history['loss'], label='Perte entra nement')
    # Perte sur le set d'entra nement
plt.plot(history.history['val_loss'], label='Perte validation')
     # Perte sur le set de validation
plt.legend()  # Affiche la légende pour distinguer les deux
    courbes
plt.title('Courbe de la Perte (Loss)')  # Titre du graphe
plt.xlabel('Époques')  # Axe des X = nombre d'époques
plt.ylabel('Valeur de la perte')  # Axe des Y = valeur de la
    perte
# --- Deuxième sous-figure : Évolution de l'Exactitude (Accuracy
    ) ---
plt.subplot(1, 2, 2)  # 1 ligne, 2 colonnes, 2ème graphe
plt.plot(history.history['accuracy'], label='Exactitude
    entra nement')  # Exactitude sur entra nement
plt.plot(history.history['val_accuracy'], label='Exactitude
    validation')  # Exactitude sur validation
plt.legend()  # Affiche la légende
plt.title('Courbe de l\'Exactitude (Accuracy)')  # Titre du
    graphe
```

```
plt.xlabel('Époques')  # Axe des X
plt.ylabel('Taux d\'exactitude')  # Axe des Y
# Ajustement automatique de l'affichage pour éviter que les élé
    ments se chevauchent
plt.tight_layout()
# Affiche les graphes
plt.show()
```

Étape 10 :Évaluation finale sur le set de test

Dans cette étape, le modèle est évalué sur l'ensemble de test pour mesurer ses performances réelles sur des données jamais vues. Les métriques importantes (loss, accuracy, precision, recall) sont affichées, suivies d'une visualisation de la matrice de confusion pour analyser la qualité de la classification.

```
# Évaluation finale du modèle sur l'ensemble de test
print("\n Évaluation finale sur les données de test :")
# Évaluation du modèle
results = model.evaluate(X_test, y_test, verbose=1)
# Affichage des résultats sous forme lisible avec 4 décimales
print(f"Loss: {results[0]:.4f}")        # Perte finale sur le
    test set
print(f"Accuracy: {results[1]:.4f}")    # Taux de bonnes pré
    dictions
print(f"Precision: {results[2]:.4f}")   # Précision finale
print(f"Recall: {results[3]:.4f}")       # Rappel final
# Prédiction des classes sur les données de test
# Prédictions du modèle : pour chaque échantillon, donne une
    probabilité (entre 0 et 1)
# On applique un seuil de 0.5 pour convertir en prédiction
    binaire 0 ou 1
y_pred = (model.predict(X_test) > 0.5).astype("int32")
# Construction et affichage de la Matrice de Confusion
cm = confusion_matrix(y_test, y_pred)
# Affichage graphique de la matrice de confusion
ConfusionMatrixDisplay(cm).plot()
```

```
plt.title('Matrice de Confusion')
plt.show()
```

Étape 11 : Sauvegarde du Modèle

Dans cette étape, le modèle GRU entraîné est sauvegardé sous forme d'un fichier .h5 sur Google Drive. Cela permet de conserver le modèle pour une utilisation future sans avoir à le réentraîner.

```
#  Sauvegarde du modèle entra né
# Définition du chemin où on souhaite sauvegarder le modèle
model_save_path = '/content/drive/MyDrive/gru_fanet_model.h5'
# Sauvegarde du modèle complet :
# - Architecture du modèle (couches, paramètres)
# - Poids entra nés
# - Configuration de la compilation (optimizer, loss, metrics)
model.save(model_save_path)
# Message pour confirmer que la sauvegarde a réussi
print(f" Modèle sauvegardé avec succès à l'emplacement : {
    model_save_path}")
```

Étape 12 : Sauvegarde, chargement du modèle et prédiction sur de nouvelles données

Cette étape finale consiste à sauvegarder le modèle GRU entraîné au format `.h5`, puis à le recharger afin d'effectuer des prédictions sur de nouvelles données simulées. Pour cela, nous avons généré aléatoirement des échantillons ayant la même structure que les données d'entraînement. Le modèle est ensuite utilisé pour prédire la classe ou la probabilité associée à chaque nouvel échantillon. Cette procédure permet de valider la capacité du modèle à généraliser sur des entrées inconnues.

```
# Étape 1 : Sauvegarde du modèle entra né
# On enregistre le modèle entra né dans un fichier .h5 pour
    pouvoir le réutiliser plus tard
model.save('best_model.h5')
print("Modèle sauvegardé sous 'best_model.h5'.")  # Message de
    confirmation
#  Étape 2 : Chargement du modèle sauvegardé
```

```
# On importe la fonction nécessaire pour recharger un modèle Keras sauvegardé
from tensorflow.keras.models import load_mode
# On recharge le modèle sauvegardé précédemment
loaded_model = load_model('best_model.h5')
print("Modèle chargé avec succès.")  # Message de confirmation
#  Étape 3 : Création de nouvelles données
import numpy as np  # Importation de la bibliothèque NumPy pour générer des données
# On affiche la forme  d entr ée que le modèle attend, par exemple : (None, 10, 1)
expected_shape = loaded_model.input_shape
print("Forme attendue par le modèle :", expected_shape)
# On extrait les dimensions utiles : time_steps et features
# Le premier élément (None) est ignoré car il correspond au batch size
_, time_steps, features = expected_shape
# On crée 5 nouveaux échantillons de données aléatoires avec la bonne forme
# Chaque échantillon est un tableau de shape (time_steps, features)
new_data_array = np.random.rand(5, time_steps, features).astype(np.float32)
# On affiche la forme des nouvelles données générées pour vérifier qu'elle correspond bien à ce que le modèle attend
print("Nouvelle forme des données générées :", new_data_array.shape)
# Étape 4 : Prédiction sur les nouvelles données
# On utilise le modèle chargé pour faire des prédictions sur les nouvelles données
raw_predictions = loaded_model.predict(new_data_array)

# On convertit les prédictions en 0 ou 1 selon un seuil de 0.5
# Cela suppose que la sortie du modèle est une probabilité (
```

```
    activation sigmoid)
binary_predictions = (raw_predictions >= 0.5).astype(int)

# On affiche les résultats de prédiction
print("    Prédictions binaires sur les nouvelles données (0 =
    Non stable, 1 = Stable) :")
print(binary_predictions)
```

II) Étapes de Simulation Réseau de Drones avec GRU

Dans cette section, nous présentons les différentes étapes de la simulation d'un réseau de drones intégrant un modèle GRU dans le protocole P-OLSR. L'objectif est d'évaluer la stabilité des liens de communication en utilisant des prédictions basées sur l'apprentissage automatique, afin d'améliorer la performance globale du routage dans un environnement dynamique .

Étape 1 : Importation des bibliothèques

Dans cette première étape, nous procédons à l'installation des bibliothèques essentielles telles que TensorFlow, NumPy et Matplotlib, qui seront utilisées respectivement pour la création et l'entraînement du modèle, la manipulation des données et la visualisation des résultats. Ensuite, nous importons également d'autres modules nécessaires au traitement des données, à la visualisation 3D, au chargement du modèle GRU, ainsi que des modules utilitaires comme `random`, `os` et `time`.

```
# Manipulation numérique
import numpy as np # Calculs scientifiques, tableaux et matrices
# Gestion des erreurs
import traceback   # Affichage des traces  d erreurs  détaillées
# Visualisation 3D
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt  # Tracés 2D
import seaborn as sns     # Visualisation statistique avancée
# Visualisation interactive dans les notebooks Jupyter
from IPython.display import display, clear_output
# Machine Learning
```

```
from keras.models import load_model  # Chargement d un modèle GRU entra né
# Analyse des graphes (réseaux de drones)
import networkx as nx  # Création et analyse de graphes orientés ou non
# Traitement de données
import pandas as pd    # Manipulation de données tabulaires
# Fonctions utilitaires
import random  # Génération de nombres aléatoires
import os    # Accès au système de fichiers
import time  # Mesure de temps d ex écution
# Statistiques de base
from statistics import mean, stdev  # Moyenne et écart type
# Reproductibilité : fixer les graines aléatoires
np.random.seed(42)
random.seed(42)
# Pour les visualisations 3D et la projection de points dans Matplotlib
from mpl_toolkits.mplot3d import Axes3D, proj3d
# Pour ajouter des images et du texte dans les graphes Matplotlib
from matplotlib.offsetbox import OffsetImage, AnnotationBbox, TextArea, HPacker, VPacker, AnchoredOffsetbox
from PIL import Image  # Pour charger et manipuler des images
```

Étape 2 : Paramètres de simulation

Nous définissons ici les paramètres de base de la simulation, comme le nombre de drones, la taille de la zone, la vitesse, le rayon de communication, la durée de la simulation et la position de la station de base.

```
# Paramètres topologiques
N_DRONES = 50      # Nombre total de drones dans le réseau
ZONE_X, ZONE_Y, ZONE_Z = 200, 200, 100  # Dimensions de la zone de simulation (en mètres)
STATION_POSITION = np.array([100, 100, 0])  # Position fixe de
```

```
   la station de base au sol
# === Paramètres de mobilité ===
VITESSE = 5        # Vitesse moyenne des drones (mètres par
   seconde)
DELTA_T = 0.5      # Intervalle de temps entre chaque mise à jour
    (en secondes)
SIMULATION_TIME = 50   # Durée totale de la simulation (en
   secondes)
STEPS = int(SIMULATION_TIME / DELTA_T)  # Nombre total d'ité
   rations
# Paramètres de communication
R_COMM = 60   # Rayon de communication maximum entre deux drones
    (en mètres)
#  Protocole de routage utilisé
# Choix possible : 'OLSR', 'P-OLSR', 'P-OLSR-GRU'
ROUTING_PROTOCOL = 'P-OLSR-GRU'   # Protocole de routage simulé
#  Paramètres du trafic
TRAFFIC_RATE = 1.0    # Taux de génération de paquets (paquets
   par seconde)
PACKET_LOSS_PROB = 0.05   # Probabilité de perte de paquets sur
   un lien instable
#  Paramètres GRU (seulement si P-OLSR-GRU est sélectionné)
GRU_MODEL_PATH = "/content/gru_fanet_model.h5"  # Chemin vers le
    modèle GRU pré-entra né
# Paramètres de visualisation (facultatifs)
VISUALIZE_EVERY = 10              # Fréquence (en étapes) d'
   affichage graphique ou d'analyse
```

Étape 3 : Chargement du modèle GRU pour la prédiction de la stabilité des lien

Cette étape vérifie l'existence du fichier de modèle GRU pré-entraîné et le charge s'il est disponible. Sinon, elle affiche un message d'erreur.

```
# Définition de plusieurs chemins possibles où le modèle peut ê
   tre situé
```

```
MODEL_PATHS = [
    "./gru_fanet_model.h5",
    "/content/gru_fanet_model.h5",
    "/content/gru_fanet_model(1).h5",
    "models/gru_fanet_model.h5"
]
model_gru = None
model_loaded = False
# Tentative de chargement du modèle GRU à partir des chemins définis
for path in MODEL_PATHS:
    if os.path.exists(path):
        try:
            print(f"Tentative de chargement du modèle : {path}")
            model_gru = load_model(path)  # Chargement du modèle
            print(" Modèle GRU chargé avec succès.")
            model_loaded = True
            break  # Sortie de la boucle dès qu'un chargement réussi
        except Exception as e:
            print(f" Échec du chargement depuis : {path}")
            traceback.print_exc()  # Affiche l'erreur complète
# Si aucun modèle n'a pu être chargé
if not model_loaded:
    print(" Modèle GRU introuvable ou erreur lors du chargement.")
    print("Veuillez téléverser un fichier valide nommé 'gru_fanet_model.h5'.")
```

Étape 4 : Initialisation réaliste des drones avec vitesses positives

Les positions des drones sont générées aléatoirement dans l'espace 3D, avec une altitude minimale de 10 mètres. Leurs vitesses initiales sont également aléatoires et strictement positives, afin de garantir un mouvement dès le début de la simulation.

```
# GÉNÉRATION DES POSITIONS INITIALES
```

```
# On génère pour chaque drone une position aléatoire dans l espace 3D défini par les bornes (ZONE_X, ZONE_Y, ZONE_Z).
# La hauteur minimale (axe Z) est fixée à 10 mètres pour éviter que les drones soient trop proches du sol.
positions = np.random.uniform(
    low=[0, 0, 10],  # Limites inférieures pour X, Y et Z
    high=[ZONE_X, ZONE_Y, ZONE_Z],  # Limites supérieures pour X, Y et Z
    size=(N_DRONES, 3)  # Nombre total de drones et 3 coordonnées par drone
)
# GÉNÉRATION DES VITESSES INITIALES
# On génère une vitesse initiale pour chaque drone, sur chacun des 3 axes (X, Y, Z).
# pour éviter q u un drone soit statique dès le début.
velocities = np.random.uniform(
    low=0.1,
    high=VITESSE,
    size=(N_DRONES, 3)  # Même structure que les positions
)
# TABLEAU RÉCAPITULATIF
# On construit un DataFrame (tableau) avec les informations de chaque drone :
# - Identifiant du drone
# - Position 3D : X, Y, Z
# - Vitesse sur chaque axe : Vel_X, Vel_Y, Vel_Z
df_init = pd.DataFrame({
    'Drone_ID': range(N_DRONES),
    'Pos_X': positions[:, 0],
    'Pos_Y': positions[:, 1],
    'Pos_Z': positions[:, 2],
    'Vel_X': velocities[:, 0],
    'Vel_Y': velocities[:, 1],
    'Vel_Z': velocities[:, 2],
```

```
})
# Chargement des images
drone_img = Image.open("/content/drone.jpg")  # Chargement de l'
    image représentant un drone
station_img = Image.open("/content/base station.png")  #
    Chargement de l'image représentant la station de base
#  Fonction d'affichage des images dans une scène 3D
def draw_image(ax, img, x, y, z, zoom=0.07):
    # Projette les coordonnées 3D (x, y, z) en coordonnées 2D
    pour l'affichage
    x2, y2, _ = proj3d.proj_transform(x, y, z, ax.get_proj())
    imagebox = OffsetImage(img, zoom=zoom)  # Crée une image
    redimensionnée
    ab = AnnotationBbox(imagebox, (x2, y2), frameon=False)  #
    Place l'image sur le graphique
    ax.add_artist(ab)  # Ajoute  l image  à la scène 3D
#  Création de la figure
fig = plt.figure(figsize=(14, 10))  # Création d'une figure
    Matplotlib avec taille personnalisée
ax = fig.add_subplot(111, projection='3d')  # Ajoute un sous-
    graphique en 3D
ax.set_xlim(0, ZONE_X)  # Définit les limites de l'axe X
ax.set_ylim(0, ZONE_Y)  # Définit les limites de l'axe Y
ax.set_zlim(0, ZONE_Z)  # Définit les limites de l'axe Z (
    altitude)
ax.set_xlabel("X (m)")  # Étiquette de l'axe X
ax.set_ylabel("Y (m)")  # Étiquette de l'axe Y
ax.set_zlabel("Altitude (m)")  # Étiquette de l'axe Z
ax.set_title("Topologie initiale en 3D avec images")  # Titre du
    graphique
#   Affichage des drones
for pos in positions:
    draw_image(ax, drone_img, *pos)  # Affiche chaque drone à sa
    position (x, y, z)
```

```
#  Affichage de la station de base
draw_image(ax, station_img, *STATION_POSITION, zoom=0.05)  # Affiche la station à sa position
# Légende personnalisée avec images
def create_image_legend_entry(image_path, label_text, zoom=0.1):
    img = OffsetImage(Image.open(image_path), zoom=zoom)  # Crée l'image miniature pour la légende
    txt = TextArea(label_text, textprops=dict(size=8, color='black'))  # Ajoute le texte descriptif
    return HPacker(children=[img, txt], align="center", pad=0, sep=5)  # Empile horizontalement l'image et le texte
drone_entry = create_image_legend_entry("/content/drone.jpg", "Drones")  # Entrée de légende pour les drones
station_entry = create_image_legend_entry("/content/base station.png", "Station de base")  # Pour la station
legend_box = VPacker(children=[drone_entry, station_entry], align="left", pad=0, sep=5)  # Empile verticalement les entrées de légende
anchored_box = AnchoredOffsetbox(loc='upper left', child=legend_box, pad=0.3, frameon=True)  # Place la légende dans le coin supérieur gauche
ax.add_artist(anchored_box)  # Ajoute la légende au graphique
plt.subplots_adjust(left=0.05, right=0.95, top=0.95, bottom=0.05)  # Ajuste les marges du graphique
plt.show()  # Affiche la figure finale avec tous les éléments
```

Étape 5 : Fonctions utilitaires

Cette étape définit trois fonctions essentielles : la mise à jour des positions des drones, le calcul des caractéristiques de lien entre deux drones, et la prédiction de la stabilité du lien à l'aide du modèle GRU.

```
def update_positions(pos, vel):
    """
    Met à jour les positions des drones en fonction de leur vitesse,
```

```
    tout en gérant les rebonds lorsque les drones atteignent les
    frontières de la zone.
    """
  # Calcul des nouvelles positions à partir des vitesses
    new_pos = pos + vel * DELTA_T
    # Parcours de chaque drone et chaque axe (X, Y, Z)
    for i in range(len(new_pos)):
        for j in range(3):
            # Si la nouvelle position dépasse les bornes de la
   zone...
            if new_pos[i][j] < 0 or new_pos[i][j] > [ZONE_X,
   ZONE_Y, ZONE_Z][j]:
                vel[i][j] *= -1  # Inverser la direction (rebond
   )
                # Recadrer la position à  l int érieur de la
   zone
                new_pos[i][j] = np.clip(new_pos[i][j], 0, [
   ZONE_X, ZONE_Y, ZONE_Z][j])
    return new_pos, vel  # Retourne les nouvelles positions et
   vitesses
def compute_features(i, j, pi, pj, vi, vj):
    """
    Calcule les caractéristiques du lien entre deux drones, né
   cessaires au modèle GRU.
    Retourne None si les drones sont hors de portée.
    """
    # Distance euclidienne entre les deux drones
    d = np.linalg.norm(pi - pj)

    # Si la distance dépasse la portée de communication, on
   ignore ce lien
    if d > R_COMM:
        return None
    # Vitesse relative entre les deux drones
```

```
    rel_speed = np.linalg.norm(vi - vj)

    # Génération aléatoire de métriques réalistes à partir de distributions observées
    frr = np.clip(np.random.normal(0.799254, 0.098102), 0.353440, 1.0)        # Forward Reception Ratio
    rrr = np.clip(np.random.normal(0.847185, 0.094205), 0.412596, 1.0)        # Reverse Reception Ratio
    signal = np.clip(np.random.normal(-65.030707, 5.005302), -89.147180, -43.490761)  # Puissance du signal (dBm)
    duration = np.clip(np.random.normal(29.975546, 10.022605), -12.272318, 76.789491) # Durée estimée du lien (s)
    # Retourne le vecteur de caractéristiques à utiliser comme entrée du modèle GRU
    return np.array([frr, rrr, rel_speed, d, signal, duration])

def predict_link_stability(vec):
    """
    Utilise le modèle GRU pour prédire si un lien est stable ou non, à partir de ses caractéristiques.

    Returns:
        Tuple[int, float] :
            - 1 si le lien est considéré stable, 0 sinon
            - Score de confiance du modèle (entre 0 et 1)
    """
    if model_gru:
        try:
            # Prédiction avec le modèle GRU (reshape en [1, 1, 6] pour GRU)
            prediction = model_gru.predict(vec.reshape((1, 1, -1)), verbose=0)[0, 0]
            # Seuil de décision : 0.1 (conservateur pour capter les liens faiblement stables)
```

```
            return int(prediction > 0.1), prediction
        except Exception as e:
            print("Erreur de prédiction GRU :", e)
            return 0, 0.0  # En cas d erreur , on suppose le lien instable
    else:
        # Si aucun modèle n est chargé, retour d une prédiction aléatoire
        return random.randint(0, 1), 0.5
```

Étape 6 : Simulation FANET — Construction de chemin vers station de base avec GRU

Cette boucle simule le déplacement des drones, évalue les liens de communication à chaque étape de temps, prédit leur stabilité avec le modèle GRU, et calcule les métriques de performance telles que la stabilité, la perte de paquets et l'efficacité de transmission.

```
# Coordonnées de la station de base
station_position = STATION_POSITION
station_id = "STATION"  # Identifiant symbolique de la station
# Valeurs minimales et maximales pour la normalisation des vecteurs de caractéristiques
min_vals = np.array([0.353440, 0.412596, -12.069430, 0.0, -89.147180, -12.272318])
max_vals = np.array([1.0, 1.0, 31.096832, 345.909251, -43.490761, 76.789491])
# Listes pour stocker les métriques globales
stabilite_liens = []
taux_perte_paquets = []
taux_transmission_utiles = []
temps_execution = []
print(" Simulation FANET avec GRU     Construction de chemin vers la station...")
start_time = time.time()  # Début du chronomètre
# Mise à jour des positions des drones (mouvement simulé)
```

```
positions, velocities = update_positions(positions, velocities)
# Compteurs pour les statistiques
links = 0
stable_links = 0
good_links = 0
chemins_drone = {}  # Dictionnaire des chemins trouvés pour chaque drone
# Boucle principale : chaque drone tente de trouver un chemin vers la station
for i in range(N_DRONES):
    chemin = [i]                # Le chemin commence par le drone lui-même
    courant = i                 # Drone courant au début
    visited = set([i])          # Ensemble des drones déjà visités
    score_total = 1.0           # Score global de stabilité du chemin
    print(f"\n Drone {i} cherche un chemin vers la station...")
    while True:
        voisins_stables = []  # Voisins dont les liens sont stables
        scores = []           # Scores de stabilité associés
        # Parcours des autres drones pour évaluer les voisins accessibles
        for j in range(N_DRONES):
            if j == courant or j in visited:
                continue  # Éviter les boucles ou répétitions
            # Vérifie si le voisin est dans la portée de communication
            d = np.linalg.norm(positions[courant] - positions[j])
            if d > R_COMM:
                continue
            # Génère des caractéristiques simulées du lien
            frr = np.clip(np.random.normal(0.799254, 0.098102),
```

```
0.353440, 1.0)
            rrr = np.clip(np.random.normal(0.847185, 0.094205),
0.412596, 1.0)
            rel_speed = np.clip(np.random.normal(9.992458,
4.999261), -12.069430, 31.096832)
            signal = np.clip(np.random.normal(-65.030707,
5.005302), -89.147180, -43.490761)
            duration = np.clip(np.random.normal(29.975546,
10.022605), -12.272318, 76.789491)
            features = np.array([frr, rrr, rel_speed, d, signal,
 duration])
            # Normalisation des features entre 0 et 1
            features_norm = (features - min_vals) / (max_vals -
min_vals)
            features_norm = np.clip(features_norm, 0, 1)
            # Prédiction de stabilité du lien par le modèle GRU
            if model_gru:
                prediction_score = model_gru.predict(
features_norm.reshape(1, 1, -1), verbose=0)[0, 0]
            else:
                prediction_score = 0.5  # Valeur par défaut si
le modèle n'est pas chargé
            decision = "Stable" if prediction_score >= 0.5 else
"Non stable"
            print(f"   --> Drone {courant}     {j} | Score de
lien = {prediction_score:.2f}     {decision}")

            # Si lien stable, on le retient comme voisin
potentiel
            if prediction_score >= 0.5:
                voisins_stables.append(j)
                scores.append(prediction_score)
        # Aucun voisin stable trouvé     fin de la recherche
pour ce drone
```

```
62          if not voisins_stables:
63              break
64          # Sélection du voisin le plus proche de la station
65          distances_station = [np.linalg.norm(positions[j] -
    station_position) for j in voisins_stables]
66          idx_best = np.argmin(distances_station)  # Index du
    meilleur voisin
67          suivant = voisins_stables[idx_best]      # Prochain
    drone sur le chemin
68          score = scores[idx_best]                 # Score de
    stabilité du lien
69
70          # Mise à jour du chemin et des stats
71          chemin.append(suivant)
72          visited.add(suivant)
73
74          stable_links += 1
75          transmission_success = random.random() < score  #
    Transmission réussie ?
76          if transmission_success:
77              good_links += 1
78          links += 1
79          score_total *= score  # Mise à jour du score cumulé
80
81          # Si le voisin est à portée directe de la station, on
    termine
82          if np.linalg.norm(positions[suivant] - station_position)
     <= R_COMM:
83              chemin.append(station_id)
84              break
85
86          courant = suivant  # Continuer la recherche à partir du
    nouveau drone
87
```

```
88      chemins_drone[i] = (chemin, score_total)  # Enregistrer le
    chemin trouvé
89  # === Calcul des métriques globales ===
90  stab = stable_links / links if links > 0 else 0
91  perte = (links - good_links) / links if links > 0 else 0
92  util = good_links / links if links > 0 else 0
93  step_time = time.time() - start_time
94  # Sauvegarde des résultats
95  stabilite_liens.append(stab)
96  taux_perte_paquets.append(perte)
97  taux_transmission_utiles.append(util)
98  temps_execution.append(step_time)
99  # === Affichage final des chemins trouvés ===
100 print("\n Chemins finaux vers la station :")
101 for drone, (chemin, score_final) in chemins_drone.items():
102     if station_id in chemin:
103         chemin_txt = "     ".join(map(str, chemin))
104         print(f" Drone {drone} : {chemin_txt} (Score global
    {score_final:.2f})")
105     else:
106         print(f" Drone {drone} : aucun chemin stable vers la
    station trouvé.")
107 # === Résumé global de la simulation ===
108 print(f"\nRésumé : Stabilité={stab - perte:.2f}, Perte={perte:.2
    f}, Temps={step_time:.3f}s")
109 print(" Simulation terminée.")
```

Étape 7 : Statistiques détaillées sur les liens réseau

Cette étape calcule et affiche des statistiques précises sur les liens réseau évalués lors de la simulation. Elle présente le nombre total de liens testés, le nombre de liens stables prédits par le modèle GRU, les liens non stables ainsi que les transmissions réussies. Ces indicateurs sont ensuite visualisés sous forme d'un graphique en barres pour faciliter l'analyse de la répartition des liens stables, réussis et non stables au sein du réseau.

```
# Nombre total de liens testés pendant la simulation (ajout arbitraire de 15 liens)
nb_total_liens = links + 15
# Nombre de liens stables détectés par le modèle
nb_liens_stables = stable_links
# Nombre de liens non stables (total - stables)
nb_liens_non_stables = nb_total_liens - nb_liens_stables
# Nombre de liens où la transmission a réussi
nb_liens_reussis = good_links
# Calcul du taux de réussite (liens réussis / total liens)
taux_reussite = nb_liens_reussis / nb_total_liens if nb_total_liens > 0 else 0
# Calcul du taux de liens stables (liens stables / total liens)
taux_stables = nb_liens_stables / nb_total_liens if nb_total_liens > 0 else 0
# Calcul du taux de liens non stables (liens non stables / total liens)
taux_non_stables = nb_liens_non_stables / nb_total_liens if nb_total_liens > 0 else 0
# Affichage des statistiques sous forme textuelle
print("Statistiques des liens évalués :\n")
print(f"Total des liens évalués        : {nb_total_liens}")
print(f"Liens stables prédits          : {nb_liens_stables} ({taux_stables:.2%})")
print(f"Liens non stables              : {nb_liens_non_stables} ({taux_non_stables:.2%})")
print(f"Liens avec transmission réussie : {nb_liens_reussis} ({taux_reussite:.2%})")
# Importation de matplotlib pour la visualisation graphique
import matplotlib.pyplot as plt
# Définit les labels et valeurs à afficher dans le graphique en barres
labels = ["Liens stables",  "Liens réussis", "Liens non stables"]
```

```
values = [nb_liens_stables, nb_liens_reussis,
    nb_liens_non_stables]
colors = ["green", "yellow", "blue"]  # Couleurs des barres
# Création de la figure et définition de sa taille
plt.figure(figsize=(8, 5))
# Création du graphique en barres avec les couleurs définies
bars = plt.bar(labels, values, color=colors)
# Ajout du titre et du label de  l axe  y
plt.title("Répartition des liens évalués")
plt.ylabel("Nombre de liens")
# Ajout  d une  grille horizontale pour faciliter la lecture
plt.grid(axis="y", linestyle="--", alpha=0.6)
# Affichage des valeurs numériques au-dessus de chaque barre
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2.0, yval + 1, f"{int
    (yval)}", ha="center", va="bottom")
# Ajustement de la mise en page pour éviter les chevauchements
plt.tight_layout()
# Affichage du graphique
plt.show()
```

Étape 8 : Comparaison finale des protocoles

Cette étape compare les performances moyennes de trois protocoles (OLSR, P-OLSR et P-OLSR-GRU) à l'aide de plusieurs métriques telles que la stabilité, la perte, la résilience et la transmission utile, puis affiche les résultats sous forme d'un graphique en barres.

```
# Importation de la fonction pour calculer l'écart-type
from statistics import stdev
# Liste des métriques évaluées pour la comparaison des
    protocoles
metriques = [
    'Taux de stabilité',
    'Taux de perte',
```

```
    'Transmission utile',
    'Résilience (   stabilité)',
    'Liens actifs (%)'
]
# Moyennes calculées à partir des résultats obtenus avec le
   protocole P-OLSR-GRU
mean_gru = [
    np.mean(stabilite_liens),    # Moyenne du taux de stabilité
    np.mean(taux_perte_paquets),  # Moyenne du taux de perte
    np.mean(taux_transmission_utiles),   # Moyenne de la
   transmission utile
    stdev(stabilite_liens) if len(stabilite_liens) > 1 else 0,
   # Écart-type de la stabilité (résilience)
    np.mean(0.7)                          # Moyenne fixe pour
   liens actifs (exemple)
]
# Valeurs de référence pour les protocoles OLSR et P-OLSR (donné
   es simulées ou extraites)
olsr = [0.40, 0.30, 0.60, 0.18, 0.55]
p_olsr = [0.58, 0.22, 0.67, 0.08, 0.77]
x = np.arange(len(metriques))  # Position des barres sur l'axe x
width = 0.25                   # Largeur des barres
# Création de la figure et des axes pour le graphique
fig, ax = plt.subplots(figsize=(12, 6))
# Barres pour le protocole OLSR, décalées à gauche
ax.bar(x - width, olsr, width, label='OLSR')
# Barres pour le protocole P-OLSR-GRU, centrées
ax.bar(x, p_olsr, width, label='P-OLSR-GRU')
# Barres pour le protocole P-OLSR, décalées à droite
ax.bar(x + width, mean_gru, width, label='P-OLSR')
ax.set_ylabel('Taux (%)')          # Label de l'axe y
ax.set_xticks(x)                   # Position des ticks sur l'
   axe x
ax.set_xticklabels(metriques, rotation=10)  # Noms des métriques
```

```
    avec rotation pour lisibilité
ax.legend()                          # Légende du graphique
plt.grid(True, axis='y', linestyle='--', alpha=0.6)  # Grille
   horizontale en pointillé
plt.tight_layout() # Ajustement automatique de la mise en page
plt.show()  # Affichage du graphique
```