Sécurisation des communications dans un réseau ad hoc au sein d'un essaim de drones

Christophe Guerber

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *18 janvier 2022* **par :**

Christophe GUERBER

Sécurisation des communications dans un réseau ad hoc au sein d'un essaim de drones

---

JURY

Mme Thi Mai Trang Nguyen	Rapporteure
M Rida Khatoun	Rapporteur
M Serge Chaumette	Examinateur
M Jean-François Lalande	Examinateur
M Vincent Nicomette	Président
M Mickaël Royer	Directeur de thèse

École doctorale et spécialité :
SYSTEMES : Systèmes embarqués
Unité de Recherche :
ENAC-LAB - Laboratoire de recherche ENAC
Thèse dirigée par :
Mickaël Royer
Rapporteurs :
Mme Thi Mai Trang Nguyen et *M Rida Khatoun*

Sécurisation des communications dans un réseau ad hoc au sein d'un essaim de drones

Christophe GUERBER

23 mars 2022

Résumé

Les drones sont de plus en plus présents, dans nos vies pour le loisir comme dans l'industrie. Les prévisions sur le marché des drones civils envisagent une croissance importante sur les prochaines années, marché qui pourrait atteindre 10 à 20 milliards d'euros au niveau mondial.

Si les missions confiées aux drones ont tout d'abord considéré des drones isolés, de nouveaux types de missions nécessitent la collaboration de plusieurs d'entre eux au sein d'une flotte.

Une flotte de drones nécessite la mise en œuvre et la disponibilité d'un réseau sans fil pour toutes les tâches ayant trait d'une part à la mission et d'autre part à toute coordination ou synchronisation. Les réseaux sans fil sont par nature ouverts sur l'extérieur et il se pose donc la question de leur sécurisation. Plusieurs travaux de recherche ont abordé cette question sous différents angles : la couche physique ; les protocoles de routage ; les systèmes multi agents. Mais aucun n'aborde la question de la sécurisation de l'accès à ce réseau et peu ont étudié la question des réponses à apporter en cas d'attaque.

Dans cette thèse nous proposons une architecture orientée vers la sécurité permettant une meilleure maîtrise des communications dans le réseau, et s'affranchissant entièrement de toute infrastructure fixe au sol. Cette architecture allie les réseaux définis par logiciels (SDN), qui est une technologie qui a émergé récemment, avec AODV, un protocole de routage adapté aux réseaux ad hoc de type FANET. Cette signalisation du plan de contrôle est réalisée dans la bande et ne nécessite qu'une seule carte réseau sans fil, ce qui en facilite l'intégration. Nous démontrons que cette architecture permet de protéger le réseau contre la plupart des attaques depuis l'extérieur. Elle nous permet également d'obtenir une bonne connaissance de l'activité dans le réseau, qui est une condition pour améliorer la sécurité.

De cette connaissance, nous proposons d'une part une technique de détection d'injection de trafic depuis l'extérieur et une méthode pour s'en défendre. D'autre part, nous proposons un ensemble de caractéristiques mesurables de l'activité du réseau propres à être utilisées avec un algorithme d'apprentissage automatique.

Nous démontrons la pertinence de ces mesures en entraînant un modèle de classification par apprentissage supervisé de type Random Forest sur un ensemble de

captures réseaux présentant des attaques sur le réseau : déni de service (DoS), balayage de ports, découverte de mot de passe (brute force) et déni de service distribué (DDoS). Les performances en terme de détection d'attaques basées sur ces caractéristiques sont prometteuses, non seulement en terme de précision mais également en terme de vitesse de détection, offrant ainsi la possibilité d'une réaction en temps réel. Cette réaction peut être mise en œuvre grâce à l'architecture proposée dans cette thèse. Des tests sur des scénarios représentatifs d'un trafic réseau pour une flotte de drones montrent que le modèle est capable de généraliser avec de bonnes performances sur notre cas d'étude.

Mots-clés

FANET, SDN, AODV, Architecture de securité, Apprentissage automatique, Classificateur Random Forest

Abstract

Drones become more and more frequent in our everyday life as a leasure and also in the industry. Analysts forecast a steady growth of the civilian drones market which could reach 10 to 20 billion euros worldwide in the coming years.

Nowadays, missions mostly operate single Unmaned Aerial Vehicle (UAV). But researchers are now considering swarms of drones to be more efficient to solve specific problems. Drones now have to collaborate and coordinate.

Swarms of drones require the availability of a wireless network for all the tasks required by the mission but also for additional coordination and synchronisation needs. Wireless networks are open to the outside by nature and securing such network is a challenging task. Several solutions proposed to tackle this issue from different aspects of data communication and securing either the physical layer or the routing protocols, or at the application level in multi-agent systems. None, however, considered securing the access to the network and few proposed efficient counter-measures.

In this thesis, we propose a security oriented network architecture that allows controlling the communication in the network, with no fixed ground based infrastructure and with a single wireless interface card. It brings emerging Software Defined Network (SDN) technology together with AODV, a routing protocol suitable for flying ad hoc network (FANET). We demonstrate that the architecture allows to protect the network against most sorts of attacks from the outside. In addition, it brings a good knowledge of the activity within the network, which is a prerequisite to further improve security.

From this knowledge, we propose a detection algorithm for traffic injection attacks from an outside node and the corresponding counter-measure. Then, we propose a set of measurable features on the activity within the network suitable for a machine learning algorithms to detect abnormal behaviors.

We demonstrate the relevance of these features by training a Random Forest classification machine learning algorithm on a dataset consisting of network captures including several network attacks: denial of service (DoS), port scan, password cracking using bruteforce and distributed denial of service (DDoS). The performances of the detection based on these features are promissing, not only in terms of precision

but also in terms of speed, paving the way for applying counter measures in real time. The latter may be conveniently put in place using the proposed architecture. Tests using representative scenarios of network traffic for a swarm of drones show a good generalization of the machine learning model and good performances.

Keywords

FANET, SDN, AODV, Security architecture, Machine Learning, Random Forest Classifier

Remerciements

Je tiens en premier lieu à remercier mes directeurs de thèse Nicolas et Mickaël pour m'avoir accordé leur confiance et pour leurs précieux conseils et aide durant ma thèse.

Je voudrais également remercier Alain Pirovano pour m'avoir encouragé à faire cette thèse, ainsi que la direction de l'ENAC et en particulier Mathy, Patrick et Eric pour m'avoir permis de réaliser cette thèse dans d'excellentes conditions.

Je remercie Emmanuel Lavinal pour le temps qu'il m'a consacré durant ma thèse, son avis éclairé sur les pistes à explorer et sur son soutien.

Mes remerciements vont ensuite à mon comité de suivi Pascal Berthou du LAAS-CNRS et Guthemberg de l'ENAC, aux enseignants-chercheurs de l'équipe ReSCo que je n'ai pas encore cité : Emmanuel et Jean-Christophe qui ont su m'encourager, me conseiller et me relire, ainsi bien sûr qu'aux (ex)enseignants de l'équipe : Luc, David et Camila pour leurs encouragements et leur enthousiasme.

Enfin, j'adresse mes remerciements les plus chaleureux à mes parents, mon frère et mon entourage qui m'ont supporté tout au long de ces quatre années, et tout particulièrement à Sophie, Laura et Juliette pour les longues relectures avisé-e-s, leur soutien indéfectible, leur compréhension et leur patience !

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

Nous présentons dans ce chapitre le contexte de cette thèse, à savoir l'utilisation des drones pour des missions de plus en plus complexes pouvant nécessiter dans certains cas l'utilisation d'une flotte de drone. Ensuite nous présentons la problématique abordée durant cette thèse : la nécessité de sécuriser les communications avec et au sein de l'essaim pour protéger les drones et leur mission, et permettre ainsi une intégration avec les autres usagers de l'espace aérien.

Sommaire

1.1 Les drones

Bien que l'on puisse faire remonter les débuts de l'histoire des drones peu après la seconde guerre mondiale, ce n'est que récemment que ce secteur s'est réellement développé. Les prévisions sur le marché des drones civils considèrent que sa croissance sera de plus de 10% durant les prochaines années et qu'il pourrait atteindre entre 10 à 20 milliards d'euros d'ici 2027. Ceci a été rendu possible par la miniaturisation et donc la réduction des coûts d'acquisition et d'opération de tels aéronefs sans pilote, leur simplicité de mise en œuvre et de pilotage a permis une utilisation pour le loisir.

Il existe plusieurs types de drones que l'on peut classifier selon plusieurs critères. Le premier est le type de voilure de l'aéronef comme illustré sur la figure 1.1 :

— les drones à voilure fixe possèdent des ailes et un moteur. Ce sont donc des avions, ils ont une grande autonomie de par leurs qualités de vol et permettent de couvrir de longues distances et d'atteindre de hautes altitudes. Ils ne décollent pas ni n'atterrisent verticalement et ont donc besoin de place pour ces deux phases de vol ;

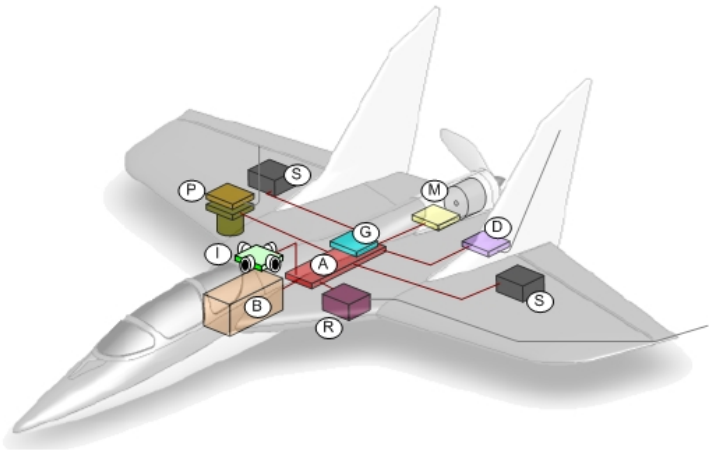
— les drones à voilure tournantes et en particulier les multi-rotors, plus proche de l'hélicoptère, ont la capacité de décoller et atterrir verticalement ou Vertical Take Off and Landing (VTOL), de faire du vol stationnaire ou a très faible vitesse. Toutefois, leur capacité à se maintenir en l'air dépend directement de la charge de leur batterie et leurs vols sont significativement plus courts. Ils compensent cet inconvénienrt par leur grande agilité.

(a) Drones à voilure fixe SUMO ©Joachim Reuder (b) Quadcopter ©Adonyi Gábor (CC0)

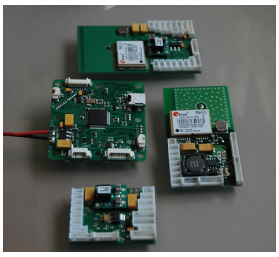
FIGURE 1.1 – Types de drones selon leur voilure

La figure 1.2 présente un exemple des différents constituant d'un drone à voilure fixe du projet Paparazzi [1]. En plus des moteurs (M) et des surfaces mobiles (S) qui lui permettent de voler et de contrôler sa trajectoire, le drone contient un ensemble de capteurs lui donnant toutes les informations nécessaires sur le vol (I) : de simples

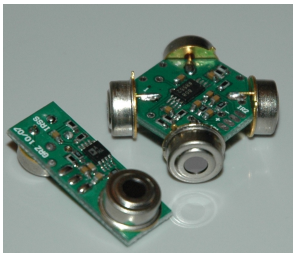
1. https ://wiki.paparazziuav.org/



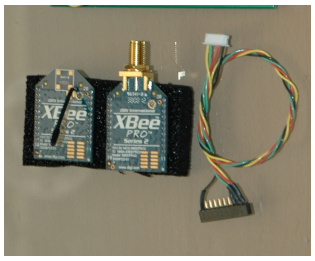
(a) Écorché d'un drone à voilure fixe Paparazzi (©Poine)



(b) Auto-pilotes (A) ©David Conger



(c) Capteurs infrarouge (I) ©David Conger



(d) Modems Xbee (D) ©David Conger

FIGURE 1.2 – Éléments d'un drone

capteurs infrarouge ou des capteurs plus évolués pour l'angle d'attaque, les vitesses, les accélérations, etc. La navigation quant à elle nécessite un ensemble de données supplémentaires comme la position géographique obtenue généralement par géolocalisation par un système de satellites (G), et éventuellement le cap. Ces données sont utilisées par le pilote automatique (A) pour le vol et la navigation. Enfin, la charge utile du drone (P) dépend de la mission de ce dernier. Il peut s'agir de caméras ou de tout autre capteur, comme d'éléments permettant d'interagir avec l'environnement du drone.

A tout cela, il faut bien sûr ajouter la batterie (B) qui est un facteur limitant pour la mission car elle détermine l'autonomie de vol du drone. Les moyens de communication numériques (D) (émetteur/récepteur et antennes) permettent au drone de communiquer avec l'extérieur. Enfin, pour des raisons réglementaires, un pilote de sécurité peut reprendre le contrôle du drone grâce à un récepteur de commandes à distance (R).

La navigation du drone peut être prédéterminée, mais on peut imaginer qu'elle soit déterminée en fonction de l'analyse des capteurs. C'est le cas par exemple lorsqu'un drone possède un mode suivi de sujet où il est capable de suivre une cible qui est généralement l'utilisateur lui même souhaitant par exemple réaliser une vidéo lors d'exploits sportifs. Le drone vole alors de manière autonome.

Il existe également une classification liée à la réglementation. Celle-ci [2] distingue trois catégories d'opération d'un drone en extérieur :

— la catégorie "ouverte" limite le poids du drone à 25 kg et l'altitude à 120m, impose que le drone soit opéré toujours à la vue du pilote et à distance de toute personne, et interdit de survoler des rassemblements de personnes. L'exploitation d'un drone dans cette catégorie n'est soumise à aucune autorisation d'exploitation préalable, ni à une déclaration d'exploitation de l'exploitant du système de drone ou Unmanned Aerial System (UAS), avant l'exploitation ;

— la catégorie "spécifique" regroupant les cas de figure qui ne correspondent pas à la classe ouverte mais où l'exploitant démontre par une étude que les risques sont acceptables (notamment l'absence de survol ou de transport de personnes). L'exploitation d'un drone dans cette catégorie nécessitent une autorisation d'exploitation délivrée par l'autorité compétente ;

— la catégorie "certifiée" regroupant les cas de survol ou de transport de personnes ainsi que l'emport de marchandises dangereuses. L'exploitation d'un drone dans cette catégorie nécessite la certification de l'UAS, la certification de l'exploitant et l'octroi d'une licence au pilote à distance.

Ainsi, selon la catégorie la réglementation européenne [3] impose des contraintes sur le matériel en terme de poids et de performances (altitude, vitesse, niveau de puissance acoustique), mais également sur l'identification de l'aéronef notamment par des feux de position. Enfin, des exigences portent également sur :

— l'émission d'informations concernant l'identité de l'aéronef, ainsi que la position du drone et de l'opérateur,

— l'emport d'une fonction de géovigilance alertant l'opérateur sur d'éventuelles risques d'intrusion dans une zone réglementée (zones interdites ou dangereuses).

La réglementation définit l'exploitation d'un drone comme autonome dès lors que le pilote à distance ne peut pas intervenir sur le vol. Ces cas de figure sont toutefois exclus de la catégorie ouverte, à l'exception du mode poursuite qui est autorisé jusqu'à une distance de 50m. Les autres cas de vol autonome rentrent dans le cadre des catégories spécifique ou certifiée selon la mission.

1.2 Les essaims de drones

Dans certaines missions, la mise en œuvre d'un seul drone n'est pas efficace voir même impossible. Par exemple pour la recherche ou la surveillance sur de larges

2. Règlement d'exécution (UE) 2019/947 de la commission
3. Règlement délégué (UE) 2019/945 de la commission

zones, un seul drone nécessiterait de longue périodes de survol pour couvrir la zone alors même que son autonomie est limitée. Il semble alors plus pertinent d'utiliser un ensemble de drones, volant en flotte ou en essaim [2] [3].

Il ne s'agit toutefois pas ici de dupliquer un drone pour les faire voler en formation et constituer ainsi une caméra virtuelle à large champ de vision. Car comme dans la nature et dans les champs de recherche bio-inspirés, on peut envisager créer une intelligence de groupe à partir de cet essaim. Cette notion a d'ailleurs été popularisée dans une série télévisée mettant en scène des essaims d'insectes.

L'intelligence de groupe passe souvent par un moyen de communication, comme dans le cas des abeilles ou des fourmis. Dans le cas de drones, nous considérerons ici la communication numérique sans fil. En comparaison d'un drone isolé qui communique principalement, si ce n'est exclusivement avec sa station de contrôle au sol, l'essaim de drones doit permettre en plus un ensemble de schémas de communication plus variés. La réalisation de la mission et son succès reposent sur la capacité à maintenir cette intelligence de groupe qui dépend directement de la disponibilité du moyen de communication.

À cet impératif de succès de la mission, s'ajoute la considération d'intégration de l'essaim de drones dans l'espace aérien. En effet, comme l'indique le résumé sur la réglementation sur les drones, des missions en autonomie devront faire l'objet de dossiers d'évaluation des risques. La sécurité de l'essaim et des drones qui le constituent est de première importance pour garantir une intégration et une acceptation de tels systèmes dans l'espace aérien dans le futur.

La notion de gestion du risque se décline généralement en deux aspects : la sécurité et la sûreté.

1.3 Sécurité et sûreté

L'utilisation de ces deux termes pose la question de leur définition exacte et des différences qui ont conduit à les désigner différemment. A cette question étymologique s'ajoute également la question de leur usage et de leur traduction, en particulier dans un domaine pour lequel l'environnement international est une raison d'être : l'aviation civile.

Ces deux domaines traitent tout d'abord de gestion du risque c'est à dire de l'acceptabilité du risque lié à l'utilisation d'un système et de ses données. Si cette notion est une perception subjective, sa gestion implique un traitement plus systématique et quantitatif ainsi que des méthodes mathématiques et un système de preuves.

Indépendamment de la terminologie spécifique utilisée dans chacun de ces deux domaines, ils tendent à identifier les évènements que l'on souhaite gérer, leur vraisemblance ou la probabilité qu'ils adviennent, et leurs conséquences. On ajoute enfin

à ces éléments, la cause de ces événements. On cherche alors à déterminer s'il est nécessaire et possible d'intervenir sur chacun de ces quatre piliers pour que le couple (conséquence, vraisemblance) soit rendu acceptable, ou encore, en vulgarisant, qu'un accident grave arrive suffisamment rarement. Dans l'affirmative, on concevra les moyens pour y parvenir.

Les acceptions anglaises de ces deux mots sont claires et rarement confondues. La *safety* traite de l'utilisation normale d'un système, qu'il faut comprendre comme un ensemble de systèmes technologiques, d'opérateurs humains et de procédures, et inclut la formation des opérateurs humains à l'ensemble. La *security* traite des intentions malveillantes dans l'utilisation dudit système. Nous traiterons dans cette thèse uniquement de la seconde, bien que les solutions proposées puissent fournir des outils pour la première.

En français, l'aviation civile traduit ces deux termes de la manière suivante : *security* est traduit par sûreté [4] et *safety*, par sécurité [5] . A l'inverse, on parle dans le monde informatique de cybersécurité et de sûreté de fonctionnement, en opposition avec les usages dans l'aviation civile.

Nous avons choisi d'utiliser dans la suite du document la terminologie en usage en informatique par soucis de clarté pour un lecteur issu de l'informatique et des réseaux, le terme cybersécurité étant entré dans le langage courant.

1.4 Les communications de drones

Garantir la disponibilité des communications pour les drones de l'essaim nécessite de prendre en compte les différentes dimensions de la communication numérique, c'est à dire considérer les premières couches du modèle OSI, dans un environnement sans fil. En matière de sécurité, la première barrière contre les attaques est la protection contre l'accès physique au système, ce que les communications sans fil ne permettent pas.

Des solutions existent sur la couche physique avec en particulier l'évasion de fréquence ou dans une certaine mesure, l'étalement de spectre, et nous n'aborderons pas ces problématiques ici. Comme présenté dans l'état de l'art, de nombreux travaux ont été menés sur la couche réseau. Toutefois, dans la grande majorité, ces travaux ont consisté à combler les faiblesses de protocoles de routage existants.

Enfin, les drones sont des plateformes aux ressources relativement limitées. En dehors des drones militaires qui ont souvent une grande autonomie et une puissance de calcul adéquate, les drones civils ont des exigences de coût qui les rendent moins

4. Sûreté : protection de l'aviation civile internationale contre les actes d'intervention illicite, annexe 17 à la convention relative à l'aviation civile internationale

5. Gestion de la sécurité, annexe 19 à la convention relative à l'aviation civile internationale

endurants et limitent leur puissance de calcul.

Avec ces contraintes, comment peut-on sécuriser les communications au sein d'un essaim de drones de manière à le protéger contre des tentatives d'attaques pouvant compromettre la mission ? C'est la problématique à laquelle mes travaux de recherche se sont attachés.

1.5 Contributions

Dans cette thèse nous proposons une architecture basée sur l'approche Software Defined Network (SDN) offrant une complète maîtrise des communications dans le réseau, tout en nous affranchissant entièrement d'une infrastructure fixe au sol ainsi que de toute problématique de placement de nœuds. Nous proposons également un ensemble de caractéristiques mesurables de l'activité du réseau et les mettons en œuvre avec un algorithme d'apprentissage automatique dont les performances en terme de détection d'attaques sont prometteuses. Nous démontrons ainsi la faisabilité d'une sonde de détection d'intrusion permettant de détecter un large spectre d'attaques internes et ce en temps réel, la réponse à ces attaque pouvant être mise en œuvre simplement sur la base de l'architecture proposée.

1.6 Plan du mémoire

Le chapitre suivant présente un état de l'art dans les réseaux de drones et leur mise en sécurité. Il présente également les travaux sur la supervision distribuée, les systèmes multi-agents et les algorithmes d'apprentissage automatisé en lien avec la sécurité. Nous étudions dans le chapitre 3 différentes architectures de réseau en vue de mettre en place les éléments nécessaires à la sécurisation du réseau de l'essaim. Nous mettons en œuvre une évaluation des performances de l'architecture en comparaison d'une architecture classique pour un réseau ad hoc de drones. Le chapitre 4 présente l'étude de l'architecture d'un point de vue cybersécurité puis propose différentes techniques pour répondre aux attaques les plus critiques. Enfin, le chapitre 5 présente la conclusion de nos travaux de recherche. Il synthétise les contributions majeures et présente les pistes à explorer pour la suite.

Chapitre 2

État de l'art

Après avoir introduit la problématique traitée dans cette thèse, ce chapitre fait un état de l'art sur les sujets abordés dans nos travaux. Dans une première partie, nous considérons la sécurité du point de vue de l'utilisation de drones, en traitant d'abord de missions pour un drone seul. Ensuite, les spécificités propres à une flotte seront introduites en considérant celle-ci comme un système multi-agents. Nous nous focaliserons sur les aspects liés aux moyens de communication et à la nature des protocoles utilisés.

La deuxième partie se concentrera plus précisément sur la sécurité dans les réseaux et en particulier sur ceux utilisés pour des drones. La protection d'un tel réseau requiert une supervision de l'activité dans le réseau et, en écho à la nature multi-agent d'une flotte de drones, nous aborderons les travaux de recherche ayant trait à la supervision distribuée. Au cours de cette thèse, nous avons décidé d'orienter notre solution vers la mise en œuvre de techniques novatrices dans les réseaux et avons opté pour les réseaux définis par logiciels qui apportent des éléments de supervision ainsi que des moyens pour contrôler les communications dans le réseau. Cette deuxième partie se termine donc par un état de l'art sur cette technologie ainsi que ses relations avec la sécurité dans les réseaux.

Notre seconde contribution dans cette thèse porte sur la détection sur un réseau défini par logiciels, d'attaques sur le réseau de la flotte de drones. Les avancées récentes en apprentissage automatique offrent des outils pertinents et efficaces pour ce type de tâches. La dernière partie de ce chapitre traite donc de ces techniques et de leur utilisation dans le domaine de la sécurité des réseaux.

Sommaire

2.1 Drones et essaims

L'utilisation de drones est de plus en plus répandue et continue à se développer. Les missions assurées par les drones de nos jours couvrent de nombreux domaines, comme par exemple la prise de vue aérienne, l'agriculture, l'inspection d'installations industrielles et demain peut-être, le transport de personnes [4]. La sécurité de ces systèmes a logiquement été l'objet de travaux de recherche qui sont synthétisés dans la première partie de cette section.

Les limites dues à l'utilisation d'un drone unique (notamment une couverture restreinte par une autonomie limitée) ont conduit les chercheurs à envisager la mise en œuvre de flottes de drones plus ou moins coopératifs [5] : manipulation d'objets, recherche de victimes ou de cibles, surveillance de larges zones, etc. La taille de l'essaim peut varier grandement, d'une dizaine à plusieurs centaines de drones, bien que de telles tailles d'essaim n'aient sans doute été étudiées que par simulation [6]. Ces essaims de drones peuvent être vus comme des systèmes multi-agents, c'est pourquoi la suite de cette section se concentre sur ce type de systèmes, ainsi que sur les solutions permettant de les sécuriser.

2.1.1 Sécurité des drones

La sécurité pour les drones est une condition à leur plus grande intégration dans l'espace aérien et ainsi à une adoption plus large. De nombreuses études ont donc abordé ces sujets par le passé.

Dans [7], les auteurs proposent une analyse de risque conforme aux standards européens [8] pour les systèmes de drones. Ils étudient les cibles et les vecteurs des attaques contre ces derniers en considérant la confidentialité, l'intégrité et la disponibilité des données et des sous-systèmes. Ils suivent en cela le modèle de menaces en cybersécurité défini par [9] et présentent des diagrammes en bloc pour modéliser le drone. Les moyens de communication ne sont toutefois pas identifiés séparément : les auteurs considèrent que ces deniers sont englobés dans tous les modules d'un drone et que tout signal de contrôle ou de donnée, toute commande entrante ou sortante passent nécessairement par eux.

Les communications dans un système à drones ont une topologie en point-à-point ou éventuellement en étoile lorsque plusieurs drones sont en lien avec une seule station sol. On classe généralement les flux de données échangées selon les catégories suivantes :

— Command and Control (C2) depuis la station de contrôle du drone vers ce dernier ;

— La télémétrie depuis le drone vers la station ;

Critiques	Non critiques
Écoute passive	Brouillage
Usurpation	Distorsion de signaux
Modification de messages (commande et contrôle)	Attaques inter-couches et multi protocoles
Déni de service	Ingénierie sociale
Intégrité des signaux	Modification du trafic de données
Virus, vers, logiciels malveillants, chevaux de Troie et enregistreurs	Exploitation de code malin et de sous-routines

TABLE 2.1 – Classification des menaces sur un système de drone

— Données de mission a priori dans les deux directions.

Les auteurs évaluent alors la probabilité et l'impact de chaque menace pesant sur les éléments identifiés. Le risque est défini et calculé comme le produit de la probabilité et de l'impact. Les menaces identifiées et leur criticité sont données dans le Table 2.1. Celles qui ont été classées comme non critiques ont été considérées soit peu vraisemblables, soit conduisant à des conséquences avec un faible impact, soit les deux.

La contribution présentée dans [10] complète cette première étude en listant toutes les attaques qui ont été implémentées, soit sur des systèmes de drones réels soit en simulation. Bien que les attaques listées visaient des drones militaires, les mêmes attaques atteindraient vraisemblablement leur but sur de petits drones commerciaux. Une part importante de ces attaques visaient le signal Global Positionning System (GPS) par brouillage ou par usurpation. Mais approximativement la moitié concerne les flux de communication, comme cible ou bien comme vecteur de l'attaque. L'étude liste également les vulnérabilités du drone Bebop pour lequel une exploitation a été mise en œuvre avec succès :

— attaque par dépassement de mémoire tampon conduisant à l'arrêt du logiciel après l'envoi d'une requête de prise de contrôle du drone dont l'entrée dans le fichier JSON comptait environ mille caractères ;

— attaque en déni de service ayant conduit à l'arrêt du logiciel après l'envoi en parallèle d'une multitude de ces mêmes requêtes JSON (environ un millier) ;

— empoisonnement du tampon ARP conduisant à l'atterrissage du drone après que le contrôleur du drone ait été déconnecté lorsque les données ont été émises vers l'attaquant ;

— contrôle et modification de la trajectoire du drone en exploitant des données partagées pour tromper le système anticollision.

Les auteurs proposent une taxonomie complétée s'appuyant sur une précédente étude sur les véhicules autonomes [11]. Ils allouent enfin les attaques présentées

précédemment à la taxonomie proposée et montrent ainsi que la plupart d'entre elles ciblent le signal GPS. Il pointent toutefois le manque de recherches sur les attaques sur les moyens de communication.

Bien que la cible de ces attaques soient les applications embarquées, le vecteur utilisé exploite souvent les faiblesses du réseau. De plus, si aucune barrière ne vient limiter ce qu'un attaquant peut faire sur le réseau, chaque vulnérabilité se retrouve exposée et facilement accessible.

On peut également lister [12] et [13] où les auteurs présentent plusieurs mises en œuvre d'attaques sur des drones commerciaux, dont certaines ciblent précisément le réseau :

— des-authentification sur le wifi ;

— injection de commandes pour prendre le contrôle du drone en inondant ce dernier avec des commandes émises par l'attaquant ;

— accès non autorisé, où l'attaquant se connecte au serveur telnet du drone, ce dernier étant exécuté en tant que super-utilisateur et ne demandant aucun mot de passe, permettant ainsi à l'attaquant de mener un large éventail d'attaques (téléversement de fichiers vérolés sur le drone, arrêt de processus, démarrage de logiciels malveillants ou de portes dérobées) ;

— empoisonnement de la mémoire tampon ARP ;

— déni de service en exploitant une vulnérabilité dans le protocole applicatif similaire au vol de session TCP [14].

Chacune de ces attaques est, soit liée au réseau (empoisonnement du cache ARP par exemple), soit la conséquence d'une conception défaillante. Certaines de ces faiblesses pourraient être évitées si le système de drone avait un meilleur contrôle sur son propre réseau, par exemple en authentifiant les communications et en leur associant une forme d'autorisation.

2.1.2 L'essaim de drones : un système multi-agents

En comparaison avec un système mettant en œuvre un drone isolé et sa station sol, l'essaim de drones constitue un système multi-agents et met en jeu d'autres formes de communication. Bien que cette thèse n'aborde pas ce champ de recherche et que les solutions étudiées ne dépendent pas des données transportées, il parait pertinent de considérer qu'une partie au moins des communications entre drones sera vraisemblablement liée à la mise en œuvre d'une telle architecture.

Les communications dans un système multi-agents suivent des schémas différents des communications point-à-point évoquées précédemment : des communications entre drones sont nécessaires pour résoudre les différents problèmes inhérents aux

systèmes multi agents et à leur organisation [15]. On peut citer l'allocation de tâches ou le consensus qui donnent lieu à des protocoles spécifiques.

L'exemple de l'allocation de tâche a conduit à la définition d'un protocole de contrat [16] ou d'enchères. Dans ce protocole, chaque agent ayant une tâche à confier, émet une demande à tous les agents susceptibles de remplir cette tâche, attendant d'eux qu'ils fassent une proposition chiffrée (l'unité de prix dépend du contexte) afin de pouvoir prendre une décision et allouer la tâche au mieux disant.

Un protocole de consensus revient à trouver une valeur commune entre les différents agents et nécessite également des communications entre drones. La vitesse de convergence de ce protocole dépend de la topologie du réseau.

La diffusion vers les autres nœuds prend généralement l'une des formes de communication suivantes [17] : la diffusion (*broadcast* ou *multicast*) lorsque cette forme de transmission est disponible ; le tableau noir (*blackboard*) où un nœud est responsable de la retransmission des données vers ceux qui l'ont demandé et sinon, la communication point-à-point.

2.1.3 La sécurité dans un système multi-agents

Les systèmes multi-agents mettent en œuvre des architectures qui les exposent à des risques spécifiques. Il s'agit pour la plupart de risques rencontrés fréquemment au niveau applicatif. La recherche liée à la sécurité et aux systèmes multi agents a pris principalement deux directions : l'utilisation de systèmes multi-agents pour apporter de la sécurité d'une part, et la recherche de solutions pour sécuriser les échanges au sein du système d'autre part. La première approche n'étant pas dans le périmètre de cette thèse, nous nous focaliserons sur la seconde.

Les auteurs dans [18] recensent les principales menaces et exigences de sécurité selon les caractéristiques suivantes d'un système multi-agents :

— la conscience de l'environnement (*situatedness*) où l'origine de l'information doit être vérifiable ;
— l'autonomie n'est assurée qu'à condition de maîtriser l'accès aux agents ;
— la sociabilité qui repose avant tout sur la sécurité des moyens de communication ;
— la mobilité des agents qui peuvent passer d'un hôte à un autre, exposant ainsi le système à l'injection d'agents malveillants ou la corruption d'un agent par un hôte malintentionné ;
— la coopération qui peut amener à la divulgation d'informations sensibles (par exemple un état interne).

Il existe des solutions au niveau applicatif comme au niveau des systèmes d'exploitation permettant de renforcer le niveau de sécurité. Il a été d'ailleurs proposé

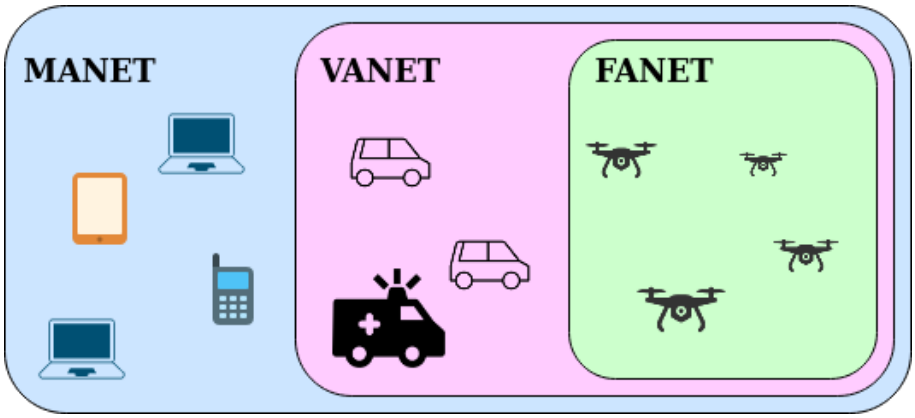


FIGURE 2.1 – MANET vs. VANET vs. FANET

d'intégrer ces bonnes pratiques dans les systèmes multi agent pour atteindre un niveau de sécurité acceptable [19]. Mais la protection du réseau et des moyens de communication au sein d'un tel système reste une question centrale en particulier sur un réseau sans fil. En particulier, la première menace identifiée dans [20] est l'injection de code qui parait tout à fait applicable dans le cas des agents mobiles.

2.2 Réseaux et sécurité

2.2.1 Réseaux de drones

Dans le cadre de cette thèse, les communications au sein de l'essaim sont assurées par un réseau sans fil. Mais il existe différents types de réseaux avec différentes topologies : réseaux cellulaires, réseaux satellites, réseaux ad hoc. Nous n'excluons a priori aucune de ces solutions. Toutefois, les contraintes physiques liées à leur topologie peuvent limiter le spectre des missions confiées à l'essaim : limite de propagation de la cellule si celle-ci est assurée par une infrastructure au sol, limite de propagation des signaux satellites, utilisation de multi-lien soit pour augmenter la disponibilité, soit pour augmenter le débit disponible.

On trouve également plusieurs classifications de réseaux ad hoc mobiles [21] selon les caractéristiques de ces mobiles en terme de vitesse de déplacement, de trajectoire, de densité et conséquemment, de vitesse de changement de topologie au sein du réseau. Comme illustré dans la figure 2.1, on trouve tout d'abord les réseaux Mobile Ad Hoc Network (MANET) dans lesquels les mobiles sont supposés être des équipements personnels au sens large (téléphones, ordinateur portable). La vitesse de déplacement y est basse, il n'existe a priori pas de schéma particulier pour leur trajectoire et la densité est moyenne à élevée. Les Vehicular Ad Hoc Network (VANET) appliquent ces mêmes principes pour des véhicules sur une infrastructure routière. Les trajectoires sont donc bien plus contraintes, les vitesses plus élevées

et la densité variable. Enfin on trouve les Flying Ad Hoc Network (FANET) qui regroupent l'ensemble des réseaux ad hoc où les mobiles peuvent voler. Bien qu'il existe un réseau de routes aériennes, qui pourraient contraindre les déplacements à l'image des VANET, les FANET considèrent généralement que les mobiles suivent des trajectoires liées à leur mission ou bien aux caractéristiques de vol, en particulier pour les aéronef à voilure fixe qui requièrent une vitesse minimale et n'autorisent pas les virages au delà de certaines limites. Cette dernière catégorie regroupe des situation très variées et les caractéristiques retenues les concernant dépendent souvent du cas considéré par les auteurs : drones civils ou militaires, drones à voilure fixe ou tournante, mission d'exploration de transport de personnes et de marchandises, de guerre même... D'ailleurs d'autres appellations sont apparues pour désigner ces réseaux en particulier dans le cas de réseaux d'une flotte de drones. Ils sont alors parfois désignés sous l'appellation Unmanned Aerial Ad Hoc Network (UAANET) [22].

L'essaim représente certainement encore un cas particulier. En effet, l'essaim représente une entité à part entière à laquelle on peut prêter une intention voir un comportement qui sont liés à la mission, au delà des comportements individuels de ses constituants. Si un individu peut à l'occasion s'extraire de l'essaim pour un temps et mener une mission en autonomie, la raison d'être de l'essaim vient de la collaboration des individus dans un but commun. Cette collaboration implique une communication déjà évoquée précédemment concernant les systèmes multi-agents. Mais cette notion d'essaim a également des implications sur les trajectoires des individus : elles ne sont plus indépendantes mais centrées sur l'essaim et sa mission. Il en résulte sans doute des caractéristiques spécifiques, moins en terme de vitesse ou de trajectoire que de densité et de vitesse de changement de topologie.

Cette spécificité est d'ailleurs indirectement confirmée au travers de certaines études sur les technologies sans fil adaptées à des équipes ou flottes de drones [4] où les auteurs confirment que le wifi répond aux exigences de débit et de délais pour de nombreuses applications ne requérant qu'un nombre limité de sauts.

Il existe différents protocoles mettant en œuvre un routage dynamique au sein d'un réseau ad hoc mobile [22]. Leurs principes et politiques de routages sont aussi divers que les mobiles qui constituent un tel réseau et il est difficile d'en désigner un qui soit meilleur quelles que soient les hypothèses : proactifs comme par exemple Destination-Sequenced Distance Vector (DSDV) [23] et Optimized Link State Routing (OLSR) [24], réactifs à l'exemple de Dynamic Source Routing (DSR) [25] et Ad hoc On-demand Distance Vector (AODV) [26], géographiques comme Greedy Perimeter Stateless Routing (GPSR) [27], hybride ou hiérarchique. Les auteurs dans [28] comparent différentes approches sur une flotte de drones et concluent à une meilleure adaptation du protocole AODV à ces conditions d'évaluation.

Routage proactif

Les protocoles de routage proactifs sont très proches des protocoles de routage rencontrés dans les réseaux filaires. Leur principe est la propagation des informations de routage vers tous les nœuds du réseau, de manière indépendante des besoins réels applicatifs, d'où la notion de proactivité. Dans cette approche, chaque nœuds maintient donc en mémoire l'ensemble des routes disponibles et ces protocoles sont parfois désignés sous l'appellation de protocoles de routage basés sur des tables (*table-driven*) . On y retrouve les mêmes méthodes de dissémination et de calcul de route que pour le filaire (par exemple OSPF ou BGP) : vecteur distance ou état de lien.

Dans la méthode par vecteur distance, chaque nœud fournit à ses voisins sa propre table de relayage en indiquant la distance associée à chaque destination. Ainsi chaque nœud complète sa table avec les informations de ses voisins et est capable de choisir le plus court chemin. Cette méthode est basée sur l'algorithme de Bellman-Ford distribué. Cette méthode est efficace dans le sens où il y a peu d'information redondante qui circule dans le réseau, mais les modifications topologiques engendrent des vagues de mise à jour et le temps de convergence de tels protocoles sont relativement longs, selon la profondeur du réseau. On trouve dans cette catégorie : DSDV.

Dans la méthode par état de lien, chaque nœud fournit l'état de ses liens avec son environnement direct à l'ensemble des nœuds du réseau. Chaque nœud reconstitue donc la topologie du réseau est peut alors appliquer un algorithme de recherche de l'ensemble des chemins les plus court : l'algorithme de Dijkstra. On en déduit aisément que ces protocoles sont plus consommateurs en ressources ce qui peut limiter la mise à l'échelle, mais sont généralement plus réactifs et on des temps de convergence inférieurs. On trouve dans cette catégorie : OLSR.

Ce dernier a été optimisé pour une utilisation sur un réseau ad hoc en hiérarchisant les nœuds du réseau. Le protocole inclut ainsi la désignation de Multi Point Relays (MPRs) qui ont pour tâche de relayer les messages en diffusion (*broadcast*). De plus, ces nœuds ont la responsabilité de générer les états de lien avec la possibilité de se limiter aux liens avec d'autres MPR. L'arbre est ainsi élagué et la quantité d'information en est réduite.

Routage réactif

Contrairement aux précédents, les protocoles de routage réactifs ne cherchent à établir une route que lorsqu'elle est nécessaire. C'est donc au moment où une application initie un dialogue avec un autre nœud que la route doit être déterminée. Ce type de protocole est ainsi constitué de procédures de découverte et de maintien

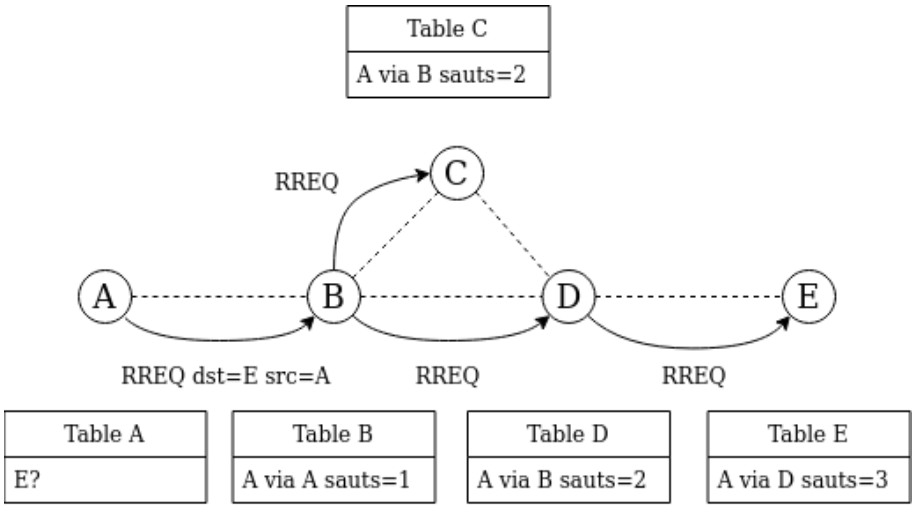


FIGURE 2.2 – Principe d'établissement d'une route de A vers E en AODV : diffusion du RREQ (simplifié)

de routes.

L'établissement d'une route consiste en la recherche du destinataire par la diffusion d'un paquet protocolaire Route Request (RREQ). Ce paquet est retransmis de proche en proche jusqu'à arriver à la destination, qui répondra alors avec un paquet Route Reply (RREP). Enfin, le maintien des routes consiste à s'assurer que le prochain saut est capable de recevoir la transmission : acquittement par la sous couche MAC, réception de transmissions, etc. La mise à jour des routes suite à la perte de connectivité avec un voisin se fait au travers de l'émission de paquet Route Error (RERR).

Afin de garantir un routage fonctionnel quelle que soit la destination, il est primordial que chaque nœud ait l'opportunité de recevoir le paquet RREQ. Ce paquet étant émis en diffusion, chaque protocole prévoit toutefois un moyen de contrôler la profondeur de l'exploration (champ Time to Live (TTL) du protocole IP ou champ spécifique du protocole). L'exploration peut ainsi être réalisée par cercles s'élargissant au fur et à mesure de la recherche.

DSR est un exemple de protocole réactif. Il utilise le routage à la source. Autrement dit, l'algorithme de recherche de route retourne la route complète à l'émetteur et cette dernière est fournie dans chaque paquet à l'émission. Il n'est donc pas nécessaire de maintenir une table pour ce protocole, mais chaque paquet contient les données de route restante vers la destination augmentant la quantité de données transmises.

AODV est un autre exemple de protocole réactif. En plus des paquets précédents, il définit un paquet *Hello* qui permet d'une part de surveiller la visibilité radio avec ses voisins, mais également de créer des routes vers les voisins directs sans recourir à la procédure de détection par RREQ. Le routage à la source n'étant pas utilisé ici,

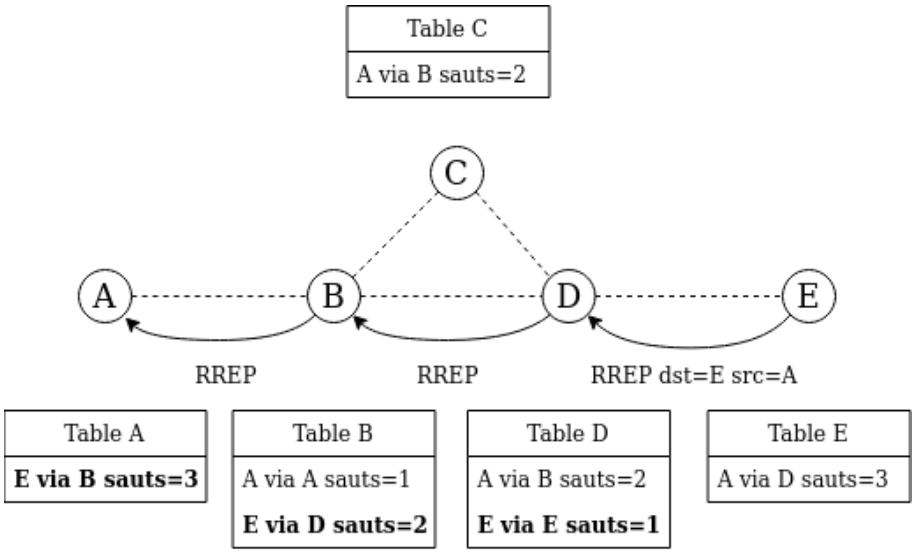


FIGURE 2.3 – Principe d'établissement d'une route de A vers E en AODV : retour du RREP et mémorisation de la route

chaque nœud doit maintenir des tables le long des routes établies. C'est ce qui est fait lorsqu'un RREQ est diffusé au sein du réseau comme illustré dans la figure 2.2 : chaque nœud garde une entrée pour la route retour. Ces dernières sont utilisées pour relayer le paquet RREP vers la source. Au passage du RREP, chaque nœud crée également une route pour pouvoir relayer les paquets dans le sens voulu comme illustré dans la figure 2.3.

Enfin, il existe des protocoles dits hybrides qui combinent approche proactive et réactive.

Routage géographique

Dans ce type de routage, les nœuds sont capables de déterminer leur position géographique, par exemple au travers d'un système de navigation terrestre par satellite ou Global Navigation Satellite System (GNSS), et de la transmettre aux autres ou à un service centralisé de localisation. Ainsi, chacun connaît ou peut demander la position de tous les nœuds du réseau. Il est alors possible de déterminer le prochain nœud à qui transmettre un paquet : c'est le nœud qui me permet de me rapprocher le plus de la destination tout en restant dans ma couverture radio. Il est donc nécessaire de connaître la position des nœuds en visibilité radio. GPSR est un exemple de protocole géographique.

Attaques

La sécurité dans ces réseaux a été étudiée dans plusieurs travaux de recherche avec différentes approches. Les auteurs dans [29] classifient les attaques spécifiquement au niveau du réseau entre passives et actives. Les passives consistent princi-

palement en la captation des transmissions des mobiles et leur analyse et n'altèrent en rien le fonctionnement du réseau. Les attaques actives identifiées consistent en :

— privation du sommeil (*sleep deprivation*) par sollicitations licites mais répétitives et prolongées conduisant à l'épuisement des batteries, par exemple en émettant continuellement des RREQ en AODV ;

— dégradation des performances du réseau, souvent par perte intentionnelle malveillante de paquets (malicious packet dropping) qu'elle soit totale ou partielle : un nœud intermédiaire dans le réseau détruit ou retarde des paquets au lieu de les retransmettre au plus vite au prochain saut le long des routes établies.

Elles peuvent être précédées par des attaques cherchant à corrompre les mécanismes impliqués dans le routage et nécessitent alors l'émission de requêtes ou de réponses par l'attaquant :

— trou noir ou gris (*black hole* ou *gray hole*) où le routage est corrompu par l'attaquant qui émet des réponses aux requêtes de découverte (RREQ en AODV) pour faire partie d'un nombre important de routes, et ainsi pouvoir soit perdre une partie des paquets volontairement (variante *gray hole*) voir même l'ensemble des paquets (variante *black hole*) ;

— *rushing* où l'attaquant cherche à prendre de vitesse les autres nœuds afin de s'imposer comme nœud intermédiaire ;

— par imposture ou attaque sybil où l'attaquant se fait passer pour un autre nœud.

On peut également lister l'attaque par trou de ver (*wormhole*) [30]. Elle consiste à créer un tunnel entre deux points du réseau, créant ainsi un chemin plus court qui aura donc la priorité lors de l'établissement des routes. Cette attaque peut être menée par simple rejeu des paquets de contrôle entre les deux extrémités du tunnel. L'attaque est donc une altération du routage qui peut mener soit à une dégradation des performances du réseau, soit à la divulgation d'informations.

Diverses approches ont été proposées pour répondre à ces attaques : le renforcement des protocoles présentant des faiblesses [22] avec de nombreuses propositions comme Secure AODV (SAODV) [31], Secure Efficient Ad hoc Routing (SEAR) [32] par exemple, ou la détection de ces attaques [29].

SAODV permet l'authentification des messages de contrôle par l'utilisation de signatures et de chaînes de hachage. Ainsi, les champs statiques comme les adresses IP sont signés et ainsi vérifiés lors de la réception du paquet. Les champs mutables sont protégés par une chaîne de hachage dont la fonction n'est a priori connue que par les nœuds légitimes du réseau. L'utilisation de ce protocole interdit la participation de nœuds illégitimes au routage dans le réseau. Il est toutefois sensible aux attaques de type trou de ver.

Les auteurs dans [33] proposent le protocole Secure UAV Ad hoc routing Protocol (SUAP) qui reprend les principes et les bonnes propriétés du protocole SAODV en ajoutant des mécanismes de vérification supplémentaires pour contrer les attaques *wormhole*. Ce mécanisme vérifie la corrélation entre le nombre de sauts le long de la route et la distance relative entre deux nœuds. Il nécessite une synchronisation au sein du réseau qui est rendue possible par l'utilisation de systèmes de géolocalisation comme par exemple Galiléo. Les performances de ce protocole permettent l'échange des données temps réel avec une qualité de service acceptable.

2.2.2 Supervision distribuée

La détection d'intrusion ou d'anomalies repose en premier lieu sur la capacité à observer le système considéré. Il est donc nécessaire de pouvoir le superviser en échangeant un ensemble de mesures ou leur description statistique, avec l'élément responsable de la détection. La qualité de ces mesures ainsi que le délai pour leur prise en compte déterminent la précision et la vitesse de détection.

Dans le cas d'un essaim de drones, il ne s'agit pas d'un seul système mais d'un ensemble d'agents ou de sous-systèmes qui doivent être supervisés. Nous nous intéressons donc à la possibilité de mettre en œuvre une supervision distribuée.

La supervision est une tâche bien connue dans les réseaux filaires par nature stables. Les auteurs de [34] présentent les défis rencontrés pour superviser un réseau mobile de type MANET (certains sont également applicables à tout type de réseau) :

— la limitation de la consommation dans un contexte de ressources limitées (processeur, énergie, débits) ;

— les performances de la supervision en précision de la mesure comme en délai ;

— la sécurité de la supervision afin de garantir l'intégrité et la confidentialité des données ;

— la capacité à survivre à une attaque ou une défaillance, en particulier lorsque des nœuds ont des rôles particuliers, soit pour relayer les données, ou pour les stocker.

Ils décrivent ensuite les différentes approches et solutions et les classent en fonction de leurs principes de fonctionnement selon les critères suivants :

— l'organisation du réseau avec trois conformations : le modèle centralisé par groupement, le modèle distribué par groupement, et le modèle distribué à plat ;

— l'intention de la supervision : la détection de panne, la configuration, la sécurité, les performances ou la facturation ;

— le type de communication et de protocoles pour échanger les données de supervision : les protocoles dédiés à la gestion, l'analyse de trafic, les protocoles de routage et les requêtes ;

— les moyens utilisés pour mettre en œuvre la politique de supervision : par agents, par langage ou par sondes actives.

L'approche la plus répandue est l'utilisation d'un protocole dédié pour la supervision. Les solutions client-serveur comme par exemple Simple Network Management Protocol (SNMP) [35] ou un dérivé sont considérées par les auteurs peu utilisables dans un réseau ad hoc. En effet, le superviseur doit émettre des requêtes de lecture à chaque nœud de manière répétée (technique dite de *polling*). Ces interrogations répétées sont gourmandes en ressources, non seulement à cause de la requête, mais également parce que la cible est interrogée même lorsqu'il ne se passe rien de particulier. La précision de la détection dépend directement de la fréquence de l'interrogation et un juste équilibre entre précision et consommation de ressources est à trouver.

Une alternative à la méthode de polling est de définir une politique de supervision en définissant des règles donnant les conditions d'émission spontanée d'une information vers le superviseur. Ces règles peuvent être définies de manière statique (comme par exemple dans le protocole SNMP pour les *trap*) ou bien définies dynamiquement avec un formalisme adapté, c'est à dire un langage permettant de décrire la politique de surveillance sous la forme d'un couple *(condition,action)* [36]. Cette approche a été implémentée dans une version distribuée par groupements [37].

L'utilisation d'agents mobiles pour la distribution de la tâche de supervision a été proposée dans [38] dans une approche distribuée par groupements. Des nœuds de gestion nomades produisent des sondes actives (des agents mobiles) par groupe. Toutefois les nœuds en bordure situés loin des nœuds de gestion ne sont pas suffisamment couverts.

Une dernière approche est d'exploiter toutes les données déjà disponibles et échangées pour d'autre besoins, comme par exemple via les protocoles de routage [39] [40] [41]. En effet, le routage est une tâche qui nécessite déjà une forme de supervision et les protocoles qui les supportent mettent donc déjà en œuvre les moyens nécessaires pour y parvenir. Ils fournissent ainsi des informations sur la topologie du réseau ou éventuellement l'utilisation des liens. Cette dernière approche ne nécessite donc aucun protocole supplémentaire pour collecter les données, les données complémentaires à superviser sont acheminées à l'intérieur du protocole de routage.

Ainsi, on compte plus d'une dizaine de propositions avec une approche distribuée qui ont trait à la détection de pannes, la performance et la sécurité. Notons en particulier Identifying Monitoring Nodes with Selection of Authorized Nodes (IMNSAN) [42] et Neighborhood Monitoring based Collaborative Alert Mechanism (NMCAM)

[43] qui traitent spécifiquement de sécurité.

IMNSAN [42] limité l'accès au MANET uniquement à des nœuds autorisés. Ensuite, les nœuds ayant le plus de batterie assurent une surveillance en capturant des paquets. Cette recherche d'intrus par signature est fortement consommatrice et une nouvelle sélection peut être demandée à tout instant. Si une intrusion est détectée, une alerte est propagée à ses voisins.

NMCAM [43] est une architecture distribuée pour la détection de comportements anormaux. Il étend le protocole de routage Dynamic Source Routing [25] en lui ajoutant un superviseur, un système de réputation et un gestionnaire de chemins. La supervision est utilisée pour bâtir un système de réputation qui permet ensuite de calculer des routes évitant tout nœud se comportant mal. Pour cela les nœuds surveillent leurs voisins et vérifient qu'ils relaient effectivement les paquets comme attendu.

Après avoir constaté qu'il n'était pas possible de comparer directement les performances des solutions étudiées, les auteurs de [34] discutent des différentes approches de supervision du point de vue de l'utilisation des ressources et concluent :

— les approches par agrégation de données diminuent le nombre de messages de monitoring mais nécessitent pour être efficaces, de gérer les agrégats pour maintenir une bonne répartition et une bonne couverture en positionnant les relais de manière adéquate, ce qui est une tâche complexe et coûteuse ;

— la définition de politiques de supervision augmente l'autonomie des nœuds et améliore les délais en comparaison d'une solution requête-réponse ;

— les approches réutilisant les données issues de protocoles de routage sont les plus adaptées et il est possible de les combiner avec les deux précédentes techniques pour en augmenter encore l'efficacité.

Ils remarquent également qu'aucune des approches étudiées n'est réellement distribuée. Il existe toujours une hiérarchie entre un ou plusieurs superviseurs et ces solutions cherchent à collecter des données depuis les nœuds supervisés vers le sommet de la structure. Il n'existe aucun retour d'information ni de réelle coopération entre nœuds.

Dans [44], l'auteur propose, en plus de deux solutions hiérarchiques relativement centralisées, une solution complètement distribuée pour répondre à ce manque de solutions. Cette dernière repose sur un protocole de consensus et l'utilisation d'un registre distribué de façon similaire à une blockchain, mais dédié à la supervision.

Certaines solutions répondent donc pour partie à la problématique de supervision mais elles sont rarement entièrement distribuées. Nous avons donc envisagé différentes approches pour répondre au problème de sécurisation des réseaux, en étudiant particulièrement les solutions novatrices. Les réseaux définis par logiciels

répondaient ainsi à plusieurs critères : ils offrent une grande souplesse d'utilisation et de contrôle des flux dans le réseaux, et produisent également des données de supervision en son élément central, propres à dévoiler des comportement anormaux dans le réseau. Nous avons donc considéré la possibilité de porter cette technologie dans notre cas d'étude.

2.2.3 Les réseaux définis par logiciel

Les réseaux définis par logiciel ou SDN sont devenus un sujet omniprésent dans la recherche sur les réseaux depuis plusieurs années. Tout d'abord orienté vers les réseaux filaires ce champ de recherche a peu à peu intégré les réseaux sans fil et la 5G.

Principes

SDN repose sur une définition d'un ensemble d'actions élémentaires qui couvre la totalité des comportements à mettre en œuvre dans les équipements réseau dédiés : routeur, commutateur, parefeu, etc. Il définit ainsi trois plans [45] :

— le plan de données où circulent les données applicatives des utilisateurs du réseau ;

— le plan de contrôle où circulent les définitions des actions à réaliser par les composants du réseau (qu'on nomme des commutateurs) au travers de connexions qu'ils ont avec un ou plusieurs contrôleurs ;

— le plan des applications SDN où l'on trouve les applications.

Le plan de contrôle permet également de faire remonter des paquets et des statistiques vers le contrôleur afin de lui permettre de dialoguer au travers des commutateurs vers d'autres éléments ou vers lui-même, et de lui donner une connaissance complète de l'état du réseau.

Le réseau physique qui supporte le plan de contrôle doit fonctionner par lui même et est souvent un réseau Ethernet. Les commutateurs SDN peuvent être physiques ou logiciels [46].

Le contrôleur interagit avec les commutateurs SDN et assure donc la mise en œuvre d'un protocole sur son interface inférieure qualifiée de descendante (*Southbound interface*). Le protocole OpenFlow [47] est un protocole reconnu de l'interface inférieure et est mis en œuvre dans nombre de contrôleurs et de commutateurs. L'interface supérieure ascendante (*Northbound interface*) permet quant à elle à des applications dédiées (on parle de plan des applications SDN) de déterminer le comportement du réseau. Cette architecture est représentée sur la figure 2.4.

Le contrôleur ([48] [49] [50] [51]) peut offrir une interface générique aux applications, par exemple une interface de programmation (API) REST en HTTP comme

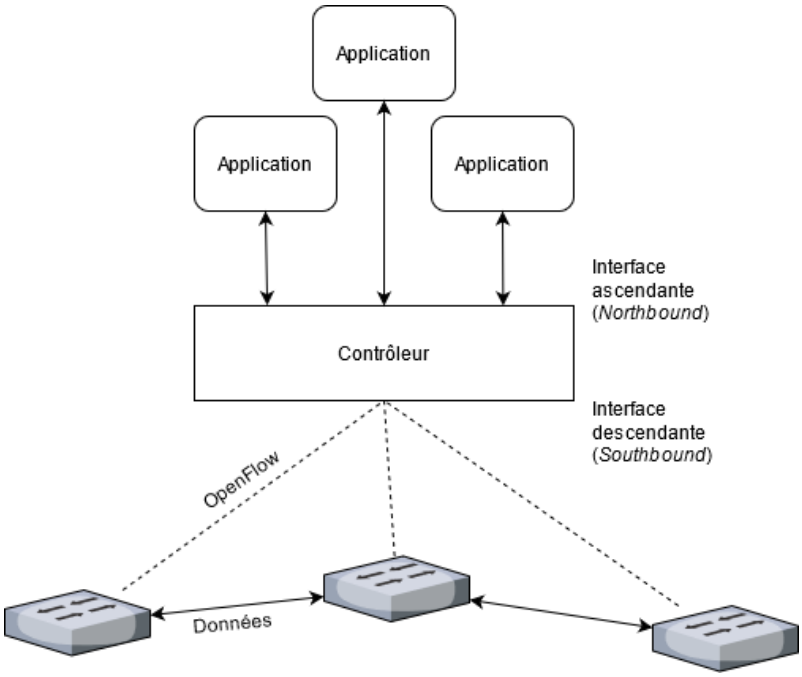


FIGURE 2.4 – L'architecture SDN

RESTCONF [52], ou une interface de programmation qui lui est propre, sous la forme d'extensions (*plugin*). Des langages de définition ad hoc ont également été proposés. Dans [53], les auteurs proposent une taxonomie des différents langages, depuis les formalismes de bas niveau comme *OpenFlow* jusqu'aux langages de programmation spécifiques au domaine comme *FlowLog* [54], *Frenetic* [55], *Pyretic* [56] ou *NetKAT* [57].

Les contrôleurs les plus aboutis ont des fonctionnalités proposant une architecture distribuée pour le contrôle du réseau, permettant ainsi d'améliorer leur disponibilité ou de répartir la charge de traitement [58].

Le plan des applications SDN

Les applications SDN déterminent le comportement ou les fonctionnalités mises en œuvre pour le réseau ainsi constitué. Ce sont elles qui répondent aux sollicitations des commutateurs, décident de leur fonctionnement et traitent également les paquets du réseau qui leurs sont remontés : la commutation des trames, le relayage de paquets, le filtrage sont tous le résultat des applications SDN présentes dans le contrôleur ou clientes de l'interface supérieure. Lorsque cela est nécessaire, les applications SDN permettent l'interopérabilité avec des protocoles mis en œuvre par des éléments traditionnels (non SDN) du réseau (par exemple BGP, OSPF).

La découverte de la topologie d'un réseau est un exemple d'application SDN élémentaire [59] et est illustrée dans la figure 2.5. Il s'agit habituellement d'une implémentation du protocole OpenFlow Discovery Protocol (OFDP) qui utilise le

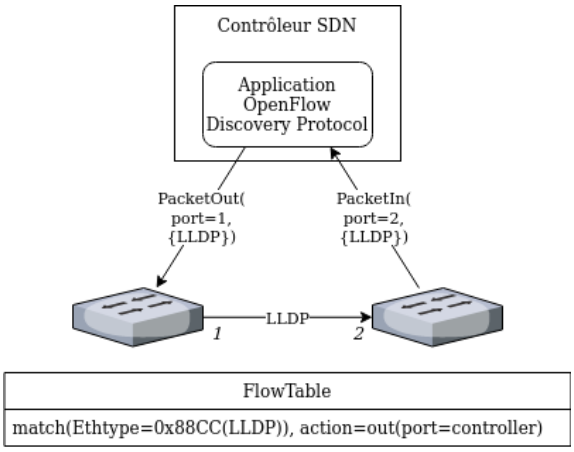


FIGURE 2.5 – L'application SDN de découverte de la topologie

format du protocole de découverte de lien Link Layer Discovery Protocol (LLDP) [60]. Les commutateurs SDN sont tout d'abord configurés par le contrôleur pour qu'ils lui remontent les trames LLDP dès réception : si la trame contient le champ EtherType égal à 0x88CC, la trame est envoyée au contrôleur. Il suffit ensuite de faire émettre une telle trame sur un port d'un des commutateurs (*PacketOut*). La trame LLDP circulera sur le câble branché sur le port en question et sera reçu par l'équipement à l'autre bout. Ce dernier remonte alors la trame au contrôleur (*PacketIn*) qui, par corrélation avec la trame émise initialement, sait maintenant que les deux commutateurs sont reliés et connait même les ports concernés. Cette découverte est donc totalement commandée par le contrôleur et nécessite deux émissions sur le plan de contrôle (la commande d'émission et l'information de réception) et une émission sur le plan de données (la trame LLDP elle même) comme indiqué sur la figure 2.5. Il est à noter que la version standard de ce protocole contient des failles de sécurité et que des versions sécurisées ont été proposées [61].

Si les mises en œuvre initiales de SDN ont eu recours à des commutateurs extrêmement simples, plusieurs travaux ont ouvert la voie à leur ajouter des fonctionnalités plus intelligentes. L'ajout de tables de groupes par exemple a permis de mettre en œuvre des mécanismes de basculement en cas de panne ou de gestion de la qualité de service sur les dernières versions du protocole OpenFlow. Le projet BEBA [62] propose la prise en compte d'états : ces états permettent de spécifier des comportements dynamiques qui s'adaptent à des flux variables dans le temps ou au trafic. Ils ont montré que, bien qu'un commutateur intelligent soit moins générique, SDN pouvait s'adapter à un tel scénario tout en améliorant la programmabilité du réseau.

Intégration des réseaux sans fil

L'intégration de réseaux sans fil dans SDN a été proposée dans plusieurs travaux de recherche et en particulier sur des réseaux ad hoc de véhicules (VANET) [63]. On note toutefois que les mobiles dans ces réseaux sont contraints par une infrastructure routière. Les propositions se basent logiquement sur l'utilisation d'infrastructures au sol localisées le long des routes. Les auteurs dans [64] proposent une architecture hybride utilisant SDN, mais également dépendante de l'infrastructure sol pour couvrir les zones où circulent les véhicules. Dans le cas d'une flotte de drones, aucune infrastructure de routes n'est supposée exister et se baser sur une infrastructure sol limiterait les missions envisageables.

Une mise en œuvre de SDN dans un réseau MANET a été proposée dans différents travaux. En particulier, [65] utilise deux interfaces sans fil. Dans [66] les auteurs y proposent un protocole de routage multisauts spécifique et comparent la version SDN à d'autres solutions de routage : AODV, OLSR et GPSR. En particulier, la solution SDN est légèrement moins consommatrice d'énergie qu'avec OLSR, mais bien plus que les deux autres protocoles. Toutefois, les délais de bout en bout ainsi obtenus sont inférieurs à tous les autres protocoles de routage MANET.

Les auteurs de [67] proposent la mise en œuvre de SDN sur un réseau ad hoc, dans un environnement sans infrastructure comme par exemple un réseau tactique d'une troupe armée en territoire ennemi. Ils mettent en œuvre l'architecture dans une preuve de concept autour d'une solution dite SDN flexible. Les nœuds du réseau opèrent sur l'architecture SDN tant que la connectivité avec le contrôleur est disponible. Un protocole spécifique permet de recréer une forme de routage à l'image des protocole de routage dynamique de type MANET. Il permet la détection des voisins, la dissémination de messages de contrôle par un mécanisme d'abonnement complexe et un mécanisme de relayage de paquets. L'hybridation entre ces deux routages permet aux nœuds d'opérer soit en mode SDN, soit en mode MANET, pour tirer partie du meilleur des deux.

2.2.4 Sécurité et SDN

Comme les auteurs de [1] le résument dans leur article, le champ de recherche sur les réseaux définis par logiciel et la sécurité est dual : d'un côté, on utilise le SDN pour améliorer la sécurité des réseaux, et de l'autre, on sécurise l'architecture SDN elle-même. Bien que la sécurité par le SDN est l'approche principal de ce mémoire, l'importance de la sécurité pour SDN ne doit pas être négligée.

Problèmes et menaces
Accès non autorisé
Divulgation non autorisée d'informations
Modification non autorisée d'informations du réseau
Destruction d'informations du réseau
Interruption de service
Erreur de configuration
Mauvaise configuration des mécanismes d'authentification, de confiance et de vérification

TABLE 2.2 – Problèmes et menaces d'attaques contre une architecture SDN [1]

Sécurisation de l'architecture SDN

Concernant cette dernière, les auteurs identifient sept problèmes et menaces d'attaques qui sont donnés en Table 2.2. Cette liste s'applique aux différentes origines et causes qui sont nécessairement parmi les composants du SDN : l'application, le contrôle (qui inclut le contrôleur et l'interface supérieure), le canal de contrôle et enfin l'infrastructure.

L'authentification, le contrôle d'accès et la dissimulation des communications de contrôle sont la pierre angulaire de la sécurité d'une architecture SDN. Ces mesures doivent être appliquées à chaque niveau : le contrôleur envers le commutateur, de même que l'application envers le contrôleur, et vice-versa. Ainsi seules les applications autorisées peuvent s'interfacer avec le contrôleur, qui accepte uniquement les commutateurs de l'infrastructure. Sur la base d'une configuration correcte de ces mécanismes, et en dehors de toute exploitation d'une vulnérabilité du jour zéro, la majorité des objectifs des attaques sont protégés : seules l'interruption de service et les erreurs de configuration restent possibles.

L'interruption de service peut prendre trois formes selon les auteurs : l'inondation par paquets, l'inondation de la table des flux et l'injection d'un paquet mal formé pour arrêter soit le commutateur soit le contrôleur. Il convient donc pour l'application, le contrôleur comme pour les commutateurs, de se prémunir contre ces types d'attaques.

Les erreurs de configuration ne constituent pas un objectif pour un attaquant. Toutefois, cette catégorie inclut la génération par une application d'entrées de flux qui seraient en conflit avec au moins une autre entrée. Il est possible alors pour l'attaquant de chercher à exploiter cette faille. L'utilisation d'un langage de haut niveau en lieu et place d'une définition *OpenFlow*, peut permettre la vérification de la cohérence des entrées de flux dans les tables, en particulier dans un scénario avec contrôleurs multiples [53].

La sécurité du réseau par SDN

Dans [68] les auteurs suggèrent que l'amélioration de la sécurité par SDN est rendue possible grâce aux quatre fonctions suivantes apportées par cette technologie :

- un contrôle dynamique des flux offrant les briques de base pour une fonction dynamique de contrôle d'accès et autorisant la séparation des flux malveillants des flux bénins ;
- une visibilité sur l'ensemble du réseau avec un contrôle de flux centralisé facilitant la surveillance du réseau et offrant une vue holistique du réseau, facilitant ainsi la détection de et la protection contre les flux malveillants ;
- la programmabilité du réseau, éliminant les équipements intermédiaires ad hoc de sécurité qui sont difficiles à déplacer, en les remplaçant par des applications SDN ou des tunnels à travers le réseau, ce qui fait de SDN un complément important aux fonctions de réseau virtualisées ;
- un plan de données simplifié permettant une extensibilité qui accroît son applicabilité à la sécurité.

On distingue plusieurs approches dans la sécurité des réseaux par SDN. L'une d'entre elle s'appuie sur la virtualisation des fonctions réseau. Cette approche se base sur la flexibilité du plan de données pour contrôler l'écoulement des flux dans le réseau et les amener vers les serveurs, mettant en œuvre les fonctions virtualisées en question [69]. Dans le domaine de la sécurité on trouve ici les pare-feu, les sondes de détection d'intrusion réseau ou les systèmes de protection contre les intrusions. Toutefois, toute la puissance et la flexibilité de cette approche conduit généralement à un allongement des flux dans le réseau. Ceci nous parait donc peu adapté à des systèmes ayant des contraintes sur la consommation d'énergie comme c'est le cas pour les drones.

D'autres approches sont basées sur l'inspection de paquets par le contrôleur [70], à la recherche de signatures dans les paquets à analyser afin d'identifier des attaques déjà rencontrées. Cette technique repose comme précédemment sur la transmission de données supplémentaires et engendre donc une consommation supplémentaire pour relayer les paquets à analyser vers le contrôleur ou la sonde.

Les protocoles définis pour SDN, OpenFlow par exemple, fournissent pourtant déjà des données sur l'activité dans le réseau. Les auteurs dans [71] par exemple proposent une méthode pour déduire l'utilisation d'un lien dans un réseau basé sur les flux comme SDN, ne nécessitant aucun trafic de supervision additionnel. La méthode repose sur la capture et l'analyse de messages de contrôle. Sa précision dépend de la quantité de messages de contrôle qui dépend de la granularité des flux et de certains paramètres du protocole. Un bon équilibre entre estimations passives et supervision active doit être trouvé dans le cas de certaines politiques de flux

comme par exemple le recours à la définition proactive de flux, la présence de flux de longue durée, ou l'utilisation de longues temporisations.

Les auteurs dans [72] analysent la transformation vers toujours plus de logiciels dans différentes catégories des réseaux de drones au travers de l'utilisation du SDN et du Network Function Virtualization (NFV). Ils montrent l'intérêt récent de la communauté scientifique pour ce sujet, chercheurs comme industriels. Les propositions abordent différents verrous technologiques depuis le routage jusqu'au positionnement, incluant les réseaux de capteurs sans fil et les VANET assistés par drones, les communications cellulaires et par satellites et la supervision. Mais peu des travaux recensés dans cet article abordent la sécurité.

Les auteurs de [73] et [74] ciblent les attaques par inondation TCP SYN. Cette attaque vise à consommer l'ensemble des ressources de la cible jusqu'à ce qu'elle ne puisse plus répondre à de nouvelles demandes de connexion pour des clients légitimes. La position centrale du contrôleur leur permet ainsi d'identifier des comportements malveillants sur la base de compteurs et de seuils.

Les auteurs dans [75] utilisent une analyse par graphes de dispersion du trafic et cherchent à détecter des schémas anormaux dans les flux comme par exemple dans le cas d'un déni de service distribué. L'approche proposée est basée sur des mesures statiques et dynamiques sur des graphes et un algorithme permettant la comparaison de graphes. Ils utilisent alors des jeux de données comme par exemple CAIDA [1] pour générer des modèles de graphes d'activités malveillantes à comparer avec le trafic réel. Ils identifient avec succès plusieurs attaques en déni de service distribué trouvées dans un jeu de données de l'université POSTECH [2].

Des travaux dédiés à la sécurité pour les réseaux de drones utilisant SDN sont recensés dans [76]. Comme le disent les auteurs, la plupart de ces recherches se focalisent sur la réduction des risques liés aux attaques en déni de service distribué, bien que plusieurs d'entre elles abordent les problèmes de brouillage et d'usurpation.

2.3 Apprentissage automatique

Comme évoqué précédemment, les réseaux définis par logiciels génèrent, de par leur fonctionnement, des données sur les communications transitant dans le réseau, et concentrent cette connaissance dans son élément central. Par ailleurs, la protection du réseau, en plus des problématiques d'authentification et d'autorisation, passe par la détection de comportements anormaux et potentiellement dangereux afin soit d'informer l'opérateur à des fins d'investigation, soit de répondre à l'attaque pour protéger le réseau. Ce même traitement est effectué dans les sondes de détection ou

1. http ://www.caida.org/
2. http ://postech.ac.kr/

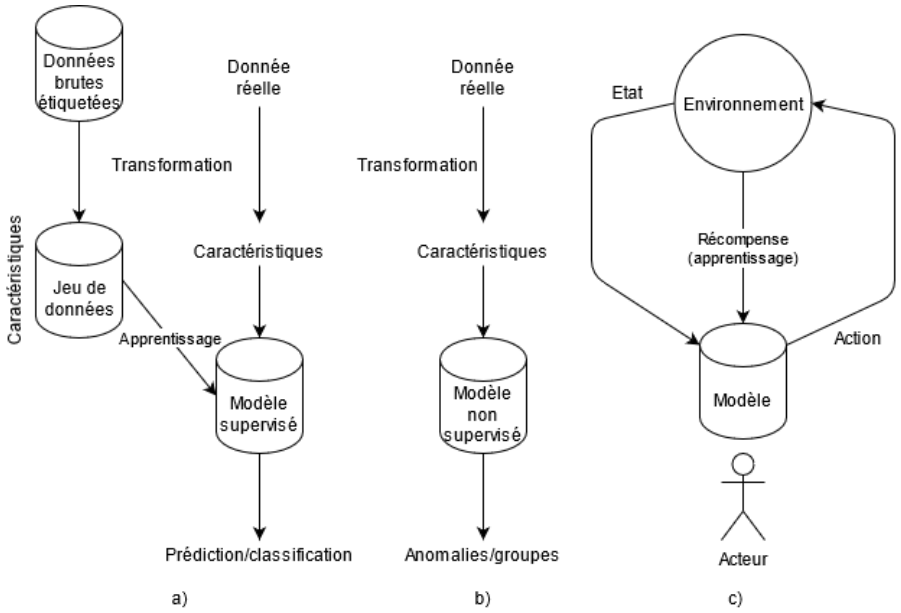


FIGURE 2.6 – Principales familles en apprentissage automatique : a) apprentissage supervisé ; b) apprentissage non supervisé ; c) apprentissage par renforcement.

de protection d'intrusion. Notre seconde contribution traite de cette détection en utilisant des techniques d'apprentissage automatique.

2.3.1 Techniques d'apprentissage automatique

L'apprentissage automatique est un ensemble de méthodes et d'algorithmes qui permettent l'extraction automatique de fonctions à partir d'un ensemble de données et de méthodes d'apprentissage. Il existe différents modèles et différentes techniques d'apprentissage avec un fondement mathématique : réseaux de neurones, arbre de décision, machines à vecteurs supports, apprentissage par renforcement, etc. L'interprétation de la fonction de sortie permet alors de mettre en œuvre différents types d'utilisation comme par exemple la prédiction ou la classification.

On distingue deux familles d'apprentissage selon si les données en entrée ont ou non une étiquette permettant de connaître la valeur prévue pour la fonction de sortie : l'apprentissage supervisé prend en entrée des données ayant été étiquetées alors que l'apprentissage non-supervisé utilise des données sans étiquette. À ces deux grandes familles s'ajoute l'apprentissage semi supervisé qui se situe entre les deux précédentes approches. Enfin, l'apprentissage par renforcement ne requiert pas de jeu de données en entrée mais une fonction de retour de valeur. L'algorithme explore donc l'environnement et apprend à partir des résultats, positifs ou négatifs de ses actions. Ces trois familles sont représentées dans la figure 2.6.

Dans le cas d'une apprentissage supervisé, les valeurs de sortie de la fonction dépendent directement des valeurs de l'étiquette. Ainsi, l'algorithme apprendra sur

la base d'une connaissance a priori des valeurs de sortie.

Le jeu de données contient généralement un ensemble suffisamment important d'échantillons pour permettre à l'algorithme d'apprendre. Un échantillon contient un ensemble de données ou caractéristiques (*features*) qui dépendent directement des données à analyser. Il peut s'agir d'images comme de mesures, ou encore de valeurs énumérées. Le jeu de données brutes doit souvent être préparé afin de l'adapter à l'algorithme qui peut, soit ne pas accepter de valeurs énumérées, soit nécessiter une normalisation des caractéristiques, ce qui évite que celles qui ont des valeurs extrêmes soient prépondérantes sur celles variant très peu. De même, dans le cas d'une classification, la répartition de chaque classe doit de préférence être équilibrée.

On évalue les modèles sur la base d'un jeu de données de test. Dans certains cas, l'algorithme renvoie une mesure de la performance du modèle sur les données d'entraînement (score *out-of-bag*). Enfin, la validation du modèle évaluant sa capacité à généraliser le modèle à des comportements jamais rencontrés auparavant doit se faire sur des données qui n'ont aucun lien avec les données d'apprentissage. On parle sinon de fuites entre jeux de données (*leakage*) qui introduisent un biais dans la validation et aboutissent à des scores supérieurs à la réalité.

Les algorithmes d'apprentissage automatique souffrent généralement de deux limitations qu'il faut prendre en compte lors de l'optimisation du modèle : le sous-ajustement et le surajustement. Dans le premier cas, le modèle n'apprend pas les caractéristiques des données ou les simplifie grossièrement : il généralise trop et rend des résultats très médiocres même sur les données d'apprentissage. Dans le second, le modèle a trop bien appris les données d'apprentissage et n'est plus capable de généraliser à des valeurs nouvelles, il a donc de très bons résultats sur les données d'apprentissage mais médiocres sur les données de test.

Le sous-ajustement provient soit d'un modèle trop simple, soit parce que le modèle n'a pas été entraîné avec suffisamment de données. L'utilisation d'un modèle plus complexe peut résoudre ce premier point mais introduit le risque du surajustement si le processus d'apprentissage est poussé trop loin.

2.3.2 Utilisation dans le domaine de la sécurité

Les techniques d'apprentissage automatique ont été proposées pour répondre à pratiquement toutes les problématiques posées dans un réseau comme il est montré dans [77] et la sécurité ne fait bien entendu pas exception.

Les auteurs classent ainsi les modèles en trois catégories : la détection de signatures connues d'attaques, la détection d'anomalies et les approches hybrides utilisant les deux techniques. Ils recensent également l'utilisation de l'apprentissage profond et de l'apprentissage par renforcement.

La plupart de ces propositions sont basées sur le jeu de données utilisé pour la compétition *Knowledge Discovery in Databases* de l'année 1999 (communément dénommé KDD99 [78]), ou bien de sa version améliorée (dénommé NSL-KDD [79]) corrigeant certains biais identifiés sur le jeu de données original. Plusieurs modélisations proposées atteignent même des performances remarquables. Toutefois, on peut remarquer que le jeu de données collecte un ensemble d'informations qui parfois sont en dehors du domaine du réseau et sont donc inaccessibles à une sonde de détection d'intrusion sur le réseau, à un contrôleur SDN ou encore à un commutateur : nombre d'accès en tant que super-utilisateur, nombre d'opération de création de fichiers ou encore nombre de *shell*. On note également une information de durée, qui ne peut être connue qu'à la fin du flux et retardant d'autant toute tentative de détection d'intrusion.

Quelques rares propositions d'utilisation de l'apprentissage automatique dans le domaine du réseau se basent effectivement sur des caractéristiques purement réseau des communications. C'est le cas par exemple de [80] où les auteurs proposent d'utiliser des caractéristiques SDN pour chaque flux : le nombre de paquets, le nombre total d'octets, le débit en octets, le débit en paquets, la longueur du premier paquet et la longueur moyenne des paquets. Le modèle est entraîné sur des données relatives à des *malware*. Le délai nécessaire pour l'identification de l'attaque n'est pas mesuré dans l'étude. Toutefois, comme certaines caractéristiques ne peuvent être connues qu'à la fin du flux (dont la durée, le nombre de paquets ou la longueur moyenne), l'algorithme de classification ne pourra probablement pas détecter l'attaque avec une bonne précision dès le début du flux.

D'autres approches utilisent les informations issues de l'entête des paquets comme dans [81] et sont assez proches des sondes de détection d'intrusion : il est proposé d'écouter et d'analyser les paquets de manière continue. Aucune information concernant le temps de détection et d'information de l'administrateur n'est donnée dans l'étude.

L'apprentissage automatique n'est pas limité à la sécurité au niveau de la couche réseau. Ces techniques ont été appliquées dans différents domaines traitant des systèmes de drones comme montré dans [82]. Les auteurs classent les solutions basées sur l'apprentissage automatique selon les quatre aspects des communications : la couche physique, le positionnement, la gestion des ressources et la sécurité et la sûreté. À côté de la détection d'usurpation de signal GPS, la plupart des solutions de sécurité traitent des aspects physiques de la transmission plus que les aspects réseau : l'écoute passive, le brouillage et l'usurpation. Les algorithmes d'apprentissage automatique recouvrent tous les types (supervisé ou non, semi-supervisé, renforcement), et incluent une large étendue d'algorithmes : réseaux de neurones dont apprentissage profond, *random forest*, machines à vecteurs supports ou *Support Vector Machine*

(SVM) et apprentissage par renforcement.

Un exemple d'utilisation de l'apprentissage automatique pour la sécurité est donné dans [83]. Les auteurs se concentrent sur l'atténuation du risque lié à quatre cyber-attaques regroupées en deux catégories : attaques sur l'intégrité avec l'usurpation GPS et la dissémination de fausses informations, et attaques en déni de service correspondant au brouillage et aux attaques *gray* et *black hole* [84]. Dans ce but, chaque drone calcule et émet un rapport basé sur un ensemble de règles de détection et le transmet vers la station sol. Un algorithme de machines à vecteur support classe les nœud selon leur comportement rapporté. Toutefois, la solution ne propose pas de contre-mesures et implique un surcoût, même minime, pour l'émission des rapports.

Enfin, les auteurs dans [76], après avoir indiqué que la majorité des solutions listées dans l'article visaient la détection d'attaques en déni de service distribué, concluent ainsi : l'intersection du SDN avec l'apprentissage automatique semble offrir le plus de promesses.

C'est justement une solution basée sur cette intersection qui est proposée dans le cadre de cette thèse : la mise en œuvre de SDN avec un plan de contrôle multi-saut, pour sa flexibilité, sa capacité à superviser et contrôler les flux dans le réseau, et un modèle de classification temps réel pour identifier les attaques et ainsi les contrer via SDN.

2.4 Conclusion

La sécurisation des communications depuis et vers un drone est un domaine de recherche important en vue de l'exploitation de ces systèmes en extérieur. Il prend une dimension encore supérieure lorsqu'il s'agit de sécuriser les communications au sein d'un essaim de drones.

Cette question a été traitée aux différents niveaux du modèle OSI, en particulier au niveau routage et d'un point de vue application dans le cadre des systèmes multi-agents. Toutefois, dans le cas plus général de l'utilisation d'un essaim, la protection du système incluant les drones et le réseau de communication dans son ensemble s'est essentiellement limitée aux routes et aux échanges au niveau physique.

Une telle protection requiert une vision sur les interactions et activités ayant lieu dans le système, alors que le réseau est par essence d'une part ouvert sur l'extérieur et d'autre part mobile. Une approche totalement distribuée pour acquérir cette connaissance est limitée par l'efficacité des moyens de surveillance du réseau de manière continue et précise. À l'inverse, une approche SDN qui apporterait cette connaissance holistique n'a pas été conçue pour des réseaux mobiles même si des propositions de mise en œuvre existent.

Au-delà de la supervision d'un réseau mobile, la détection des attaques pose également question dans le contexte d'un réseau ouvert et mobile. Les approches conventionnelles ne prennent pas ces contraintes en compte, ni même l'autonomie en énergie des nœuds du réseau, ni un quelconque coût lié à la captation pour analyse des données dans le réseau.

Au-delà de l'objectif de performance inhérent à une solution de protection (taux de détection, vitesse de traitement et de détection, efficacité de la protection), la mise à l'échelle d'une solution à une multitude de nœuds sur un réseau sans fil ainsi que le coût lié aux communications supplémentaires en terme de débit et d'énergie, et les limites en ressouces de calcul et en autonomie des nœuds du réseau, remettent en question l'application de solutions existantes.

Dans cette thèse, nous explorons la mise en oeuvre d'une architecture SDN sur un réseau de type ad hoc et son utilisation dans le cadre de la sécurisation et la protection du système en exploitant SDN et des techniques d'apprentissage automatique en réponse à ces verrous scientifiques.

Chapitre 3

Architecture

Dans ce chapitre, nous étudions les architectures permettant de mettre en œuvre les services et fonctions pour sécuriser un réseau de drones : la collecte d'informations sur l'activité dans le réseau ainsi que le contrôle des flux de données. Nous identifions quatre architectures pour ce faire, la première basée sur des protocoles et des outils utilisés communément dans les réseaux ad hoc et la sécurité et les trois autres sur l'utilisation du SDN.

Nous décrivons ensuite la plate forme mise en œuvre pour réaliser nos expériences.

Enfin, nous étudions et comparons l'architecture de référence avec celle qui répond le mieux aux contraintes d'une flotte de drones. Après avoir décrit le scénario, nous analysons les performances de chacune.

Sommaire

3.1 Solutions proposées

Nous considérons ici un essaim de drones coopérants, et plus particulièrement son réseau. Ce dernier supporte les transmissions entre applications hébergées par les éléments du réseau. Les applications habituellement rencontrées concernent, d'une part les communications de contrôle et de commande (C2) entre les drones et la station sol, ainsi que la télémétrie, et d'autre part les applications liées à la mission. Ces dernières peuvent mettre en œuvre différents types de communication dans des échanges entre drones ou bien avec la station sol : point-à-point ou point-à-multipoint comme cela peut être le cas pour les protocoles de réseau contractuels [16] [85].

Ces échanges nécessitent tout d'abord de disposer d'un service de routage au sein du réseau de l'essaim. La protection du réseau implique ainsi la protection de ce service de routage. Les premières barrières mises en œuvre dans un réseau filaire sont généralement les suivantes :

— la protection contre l'accès physique aux réseaux et aux équipements qui les constituent ;

— une liste des transmissions autorisées et les règles associées pour les mettre en place ;

— des accès et des échanges d'informations de routage sécurisés et authentifiés ;

— la non-divulgation d'informations sensibles ;

— l'isolation et la segmentation du réseau en zones afin de limiter la contagion et faciliter la définition et l'application des règles précédemment citées.

Dans le cas d'un réseau sans fil plusieurs de ces barrières sont mises à mal. Premièrement, la protection contre l'accès physique est illusoire. De même, la non-divulgation d'informations en dehors du réseau est également compliquée par le fait que toute station dans la couverture radio d'un nœud du réseau recevra chacune de ses transmissions. Enfin, la segmentation du réseau dépend du type de missions, mais peut être impossible si les drones sont indifférenciés et sont mobiles au sein du réseau.

Au-delà du service élémentaire de routage, nous chercherons aussi à mettre en place des outils afin de renforcer les défenses du réseau contre les attaques. Nous considérerons en premier lieu les attaques depuis un nœud externe à l'essaim, mais l'hypothèse d'une attaque depuis un nœud du réseau n'est pas à exclure. Ceci pourrait arriver suite à une cyberattaque par exemple.

3.1.1 Éléments d'architecture

D'une manière générale les systèmes permettant de renforcer les défenses d'un réseau contre les attaques sont de deux types : les systèmes de détection et les

systèmes de protection. Ces derniers se distinguent des premiers par leur capacité à agir contre l'attaque. Ils comprennent donc les quatre éléments suivants :

— l'alimentation du système en données d'observation du réseau, soit directement lorsque le système est situé sur le passage des flux de données, soit grâce à des moyens de surveillance spécifiques ;

— des algorithmes d'identification permettant la détection d'une action déloyale au travers des données collectées ;

— un processus de décision qui, en fonction de l'action déloyale détectée, doit identifier la contre-mesure adéquate ;

— enfin, un ensemble de moyens d'actions pour mettre en œuvre la ou les contre-mesures identifiées.

Il s'agira donc de définir une architecture réseau prenant en compte l'ensemble de ces contraintes et offrant des performances comparables aux protocoles de routage ad hoc actuels. Les métriques de performances prises en compte dans le cadre de cette thèse seront la disponibilité et les délais.

Dans le cadre de cette thèse nous avons étudié plusieurs architectures permettant de répondre aux quatre éléments décris précédemment. Tout d'abord, nous avons considéré l'utilisation d'un protocole de routage ad hoc adapté à une flotte de drones [28] auquel nous ajoutons des outils classiques du domaine de la sécurité réseau. Ce scénario nous servira par ailleurs de scénario de référence.

En alternative, nous avons étudié la possibilité d'utiliser un réseau défini par logiciel ou SDN comme outil pour renforcer la sécurité au sein du réseau comme cela a été proposé dans d'autres travaux sur les réseaux filaires [77]. Différentes architectures seront présentées en fonction du nombre et de la portée du réseaux sans fil, ainsi que de l'organisation du plan de contrôle.

Routage AODV avec filtrage *iptables* AODV est un des protocoles de routage développé pour les réseaux ad hoc. Ses performances sur un réseau de drones ont été démontrées [28] et il semble donc naturel de le proposer en première approche.

Ce protocole de routage réactif établit tout d'abord les voisinages entre chaque routeur AODV par l'émission régulière de paquets Hello. Chaque nœud surveille donc la réception de ces paquets pour chaque voisin : leur absence signifie une perte de voisinage.

Pour tout paquet à émettre vers une destination D pour laquelle il n'existe pas localement de route récente, le routeur effectue une recherche concentrique de route auprès des nœuds voisins en leur transmettant une requête de route (RREQ). Ces requêtes sont retransmises si aucune route n'est présente. La source de la requête contrôle la profondeur de la recherche au travers du champs de durée de vie (TTL).

Dès qu'un routeur possède une route vers la destination D demandée, soit parce qu'il y est directement relié soit parce qu'il a lui même une route récente vers celle-ci, il répond positivement à la requête en transmettant un paquet RREP au nœud d'origine. Ceci est rendu possible par l'enregistrement de la route retour lors des retransmissions successives. Lorsqu'un routeur détecte une perte de voisinage impliquée dans une de ses routes, un paquet d'erreur est émis pour informer les nœuds en amont de la perte de connectivité. Ces routes devront alors être réétablies. Les routes sont maintenues durant des périodes courtes en cas d'inactivité ; le chronomètre étant fixé à 3 secondes par défaut pour chaque route établie. Il est redémarré pour la route concernée à chaque émission de paquet.

La protection du service de routage nécessite l'utilisation d'une version sécurisée d'AODV, pour laquelle une infrastructure de clés publiques ou *Public Key Infrastructure (PKI)* doit être définie et mise en place. Celle-ci est hors du champ de cette thèse.

À ce protocole de routage, il est nécessaire d'ajouter la possibilité de surveiller le réseau et de pouvoir mettre en œuvre la politique capable de renforcer la sécurité dans le réseau. Dans un réseau ad hoc, chaque nœud est également un routeur dans le réseau, point de passage obligé des données. Il semble alors naturel de vouloir adjoindre au routeur la capacité de filtrer le trafic en fonction des communications considérées comme légitimes au sein de notre réseau.

Ce filtrage, similaire à ce que pourrait faire un pare-feu, peut être mis en œuvre par exemple, à l'aide du logiciel *iptables* sur un système de type Linux. Le déploiement de ces règles nécessiterait des protocoles spécifiques ou pourrait être réalisé au travers de commandes à distance, comme par exemple un *shell* sécurisé (SSH).

Enfin, AODV ne fournit pas d'information de surveillance et il n'engendre que très peu d'échange d'information avec l'extérieur. Il faudrait donc, soit modifier AODV, soit disposer d'un protocole permettant l'interrogation des nœuds du réseau. On pense ici en premier lieu à SNMP, mais tout autre protocole de surveillance pourrait convenir. On peut toutefois noter que ces techniques de surveillance nécessitent généralement des interrogations régulières et engendrent beaucoup de trafic. L'équilibre entre la quantité de données et la précision de la surveillance est également une difficulté de cette méthode. Alternativement, on peut mettre en place une politique de surveillance où les nœuds reportent leur état sur la base d'un ensemble de règles prédéfinies ou dynamiques. Cette solution peut réduire le nombre de requêtes nécessaires. Dans tous les cas, il convient de protéger ces transmissions contre une attaque qui chercherait à rendre la surveillance inefficace.

Ainsi les informations de surveillance, qu'elles soient collectées au travers d'un protocole dédié (par exemple SNMP) ou bien issues d'autres applications comme la télémétrie, seraient alors traitées par une application de contrôle. Celle-ci aurait pour

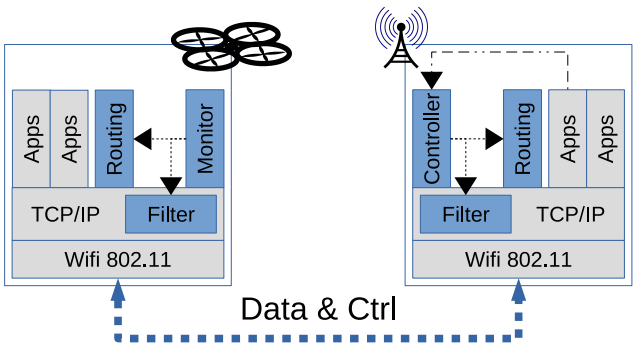


FIGURE 3.1 – Architecture AODV

tâche de détecter les anomalies dans les échanges et de déployer des contre-mesures au travers d'un protocole à définir comme évoqué ci-dessus. Ces contre-mesures pourraient influer sur le routage ou le filtrage des flux.

La figure 3.1 illustre cette architecture ainsi que les ajouts nécessaires pour disposer de moyens de surveillance du réseau et d'application des politiques de sécurité.

3.1.2 Application du SDN

Les réseaux définis par logiciel apportent une grande souplesse dans la gestion des réseaux : ils ne nécessitent qu'un seul type de plateforme matérielle avec un langage unique de configuration et permettent une intégration avec les anciens équipements et protocoles distribués. Par ailleurs, le contrôle et donc la gestion du réseau est centralisée ce qui facilite d'autant l'adaptation du réseau aux flux de données et aux changements de topologie.

SDN a également été utilisé afin d'améliorer la sécurité dans les réseaux filaires. Il parait donc intéressant de l'appliquer dans le contexte de réseaux de drones.

Dans un réseau défini par logiciel, on distingue le plan de données qui concerne le transport des données applicatives, du plan de contrôle qui concerne le contrôle et la définition du réseau. SDN nécessite normalement un réseau pour chacun des deux plans. Il est important de noter que le réseau du plan de contrôle ne peut être contrôlé par le contrôleur SDN. Il doit fonctionner de manière nominale par lui-même.

Plusieurs stratégies ont été proposées : en particulier l'approche hors-bande (*out-of-band*) où les deux réseaux sont distincts, et l'approche dans la bande (*inband*) où les deux plans utilisent un seul et même réseau.

SDN avec un réseau sans fil de grande portée

Dans cette architecture, nous considérons un réseau support hors-bande pour le plan de contrôle. Ce réseau sans fil à large couverture permet au contrôleur de

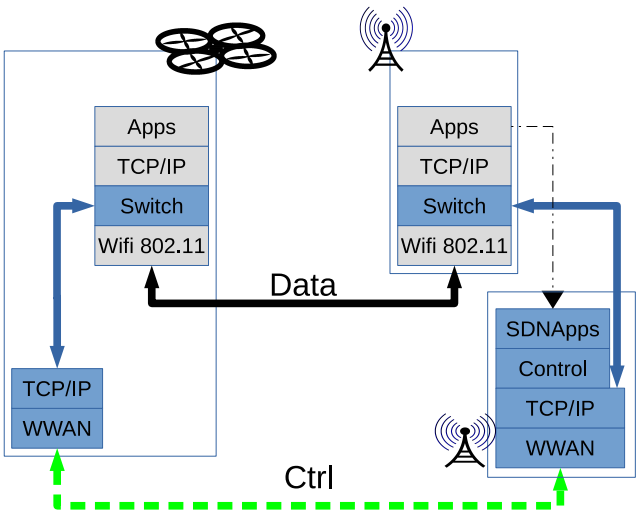


FIGURE 3.2 – Architecture de réseau SDN avec plan de contrôle hors bande sur réseau sans fil à large couverture

dialoguer directement avec les nœuds du réseau.

Quant au placement des commutateurs SDN entre les nœuds, nous pouvons remarquer que la mise en place de contre-mesures n'est possible qu'en présence de ces derniers. Sauf contraintes particulières sur les ressources (mémoire, puissance de calcul), chaque nœud emportera un commutateur.

La figure 3.2 illustre cette architecture avec l'utilisation de deux réseaux sans fil. On y trouve notamment les applications au niveau du contrôleur. SDN fournit des primitives protocolaires pour interroger les commutateurs et ainsi collecter des informations de surveillance. Comme précédemment, des applications non SDN pourraient également apporter des données utiles pour la détection de comportements anormaux ou leur localisation (par exemple la télémétrie). Toutefois, les protocoles SDN comme *OpenFlow* ne proposent aucun mécanisme pour la découverte de la topologie ou la détection de changements topologiques.

Réseau SDN hiérarchique

L'architecture précédente est fortement dépendante de la disponibilité du réseau supportant le plan de contrôle. Les conditions de propagation influent directement sur la capacité du contrôleur à dialoguer avec les nœuds du réseau et un tel réseau sans fil peut ne pas être disponible sur toute la zone de la mission. Afin de gagner en flexibilité, nous avons donc envisagé l'utilisation de relais entre le contrôleur et les nœuds d'extrémité.

Comme précédemment, il existe deux réseaux dans cette architecture : un premier entre le contrôleur et les drones relais et un second entre les drones relais et l'essaim. La solution ne repose plus sur un réseau à large couverture et permet d'atteindre des zones où cette couverture ne serait pas disponible. Ceci requiert toutefois une

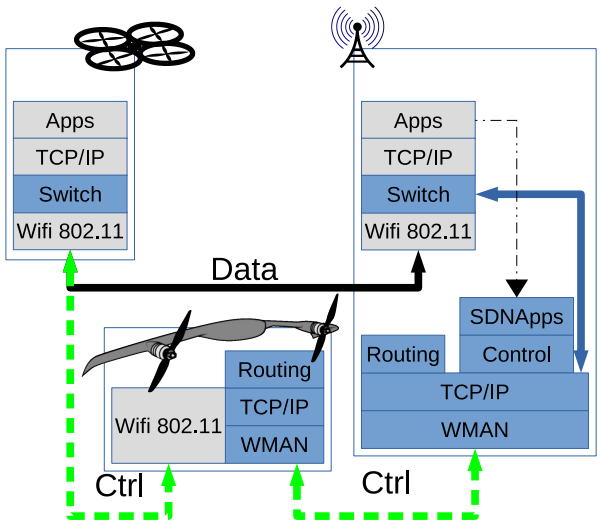


FIGURE 3.3 – Architecture de réseau SDN hiérarchique

hiérarchie entre les drones et introduit une complexité supplémentaire, notamment sur le placement et la trajectoire de ces relais. Enfin, le protocole de découverte de la topologie et de ses modifications reste à définir.

La figure 3.3 illustre cette architecture avec un drone relais qui étend la couverture du réseau de contrôle au plus près de l'essaim.

Routage sur le plan de contrôle

La configuration des deux architectures précédentes avec un réseau hors-bande pour le plan de contrôle limite a priori les scénarios possibles. Nous proposons donc l'utilisation d'un plan de contrôle dans la bande, c'est à dire que ce dernier partage le même réseau sans fil que le plan de données. Comme évoqué précédemment, ceci n'est possible que si le réseau de contrôle est autonome d'un point de vue routage. Ceci nécessite donc l'ajout d'un protocole de routage pour assurer le relayage des paquets entre le contrôleur et les commutateurs.

Parmi les protocoles de routage au sein d'un réseau ad hoc, pour les raisons décrites précédemment, nous avons choisi AODV. Il s'agit ici d'établir et de maintenir la route entre un commutateur et le contrôleur. Il n'est donc pas nécessaire de pouvoir établir n'importe quelle route au sein du réseau.

La figure 3.4 reprend l'ensemble de ces éléments d'architecture.

Connaissance de la topologie La découverte de la topologie ainsi que de ses changements dans un réseau SDN est généralement réalisée en utilisant OFDP qui est basé sur des protocoles actifs tel que LLDP comme illustré en figure 3.5. Une version optimisée est proposée dans [86]. Ceci nécessite l'émission régulière de requêtes depuis le contrôleur vers les commutateurs. Ces paquets reçus sont renvoyés vers

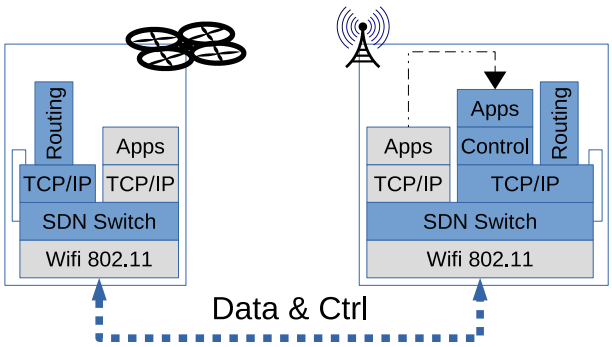


FIGURE 3.4 – Architecture de réseau SDN avec plan de contrôle dans la bande, routé par AODV

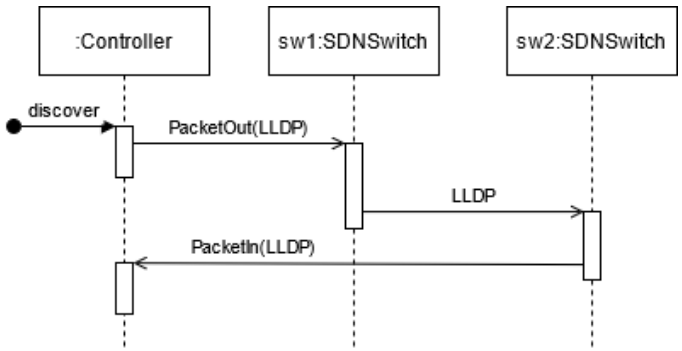


FIGURE 3.5 – Découverte de topologie par exploration des liens en utilisant LLDP

le contrôleur qui détecte alors chaque liaison. Ce protocole donc peu efficace dans le cas d'un réseau à diffusion comme un réseau sans fil multi-sauts : le contrôleur devrait émettre une requête d'émission pour chaque drone de manière régulière et recevrait autant d'information de réception qu'il y aurait de voisins.

En effet, si l'on note N le nombre de drones et donc de commutateurs, M le nombre moyen de voisins par nœud et H le nombre moyen de sauts entre le contrôleur et les nœuds du réseau sur le plan de contrôle, on obtient pour une exploration complète de la topologie : $N * H$ transmission de *PacketOut*, N trames LLDP, $N*M*H$ transmissions de *PacketIn*. Dit autrement, la trame LLDP est retransmise $H(M+1)$ fois pour découvrir le voisinage d'un seul nœud.

Exploiter AODV pour la découverte de la topologie Comme pour tout protocole de routage dans un réseau ad hoc, AODV collecte des informations de voisinage. Nous proposons de réutiliser la connaissance du voisinage de chaque nœud déjà construite par AODV sur le plan de contrôle pour alimenter le processus de découverte du plan de données. Ainsi, lorsqu'un nœud établit la connexion avec le contrôleur SDN, il émet l'état actuel de son voisinage. Par la suite, toute modification (ajout ou suppression d'un voisin) sera remontée de manière à ce que l'application de SDN qui gère la mobilité au niveau du contrôleur puisse maintenir la topologie du

réseau à jour. On peut ainsi lister les primitives protocolaires suivantes, qui ont été implémentées au travers des extensions expérimentales proposées par le protocole OpenFlow :

— AODV_STATUSREQUEST est émis par le contrôleur afin d'obtenir l'état courant du voisinage,

— AODV_STATUSRESPONSE est la réponse au précédent message,

— AODV_ADD_NEIGHBOR est émis dès qu'un nouveau voisin est détecté (lors de la réception de paquets HELLO),

— AODV_DELETE_NEIGHBOR est émis lorsque la perte d'un voisin est détectée.

Sélection des architectures L'utilisation d'un second réseau sans fil, en particulier lorsque les distances de transmission sont importantes, induit des contraintes physiques notamment pour l'installation de deux antennes. Par ailleurs, le découplage nécessaire entre ces antennes peut augmenter ces contraintes physiques. Enfin, une augmentation de la distance entre émetteur et récepteur implique généralement une augmentation de la puissance à l'émission et par conséquent des besoins en énergie plus importants. C'est pourquoi nous avons écarté toute architecture nécessitant un second réseau sans fil.

L'architecture hiérarchique reporte les contraintes citées précédemment sur quelques drones spécifiques. Elle pose le problème de placement de ces drones ainsi que la nécessité pour l'opérateur des drones de gérer une flotte hétérogène.

Nous avons donc choisi de concentrer notre étude sur la dernière de ces architecture, en comparaison avec la première qui sera pour nous celle de référence.

3.2 Émulation

Afin de vérifier l'architecture proposée et mesurer ses performances, nous avons opté pour l'émulation plutôt que la simulation. Si la simulation apporte beaucoup de souplesse dans l'étude et l'analyse, ce choix permet d'envisager sa mise en œuvre même partielle, par exemple sur une plateforme de démonstration.

Le choix des différents composants nécessaires à la mise en œuvre des deux architectures a été fait sur la base de leur représentativité et de leur facilité de modification.

Ainsi, nous avons choisi l'implémentation AODV de l'Université d'Upsala car c'est une implémentation reconnue [87] qui présente de bonnes performances. Elle est basée sur un module noyau Linux et une partie en espace utilisateur. Un portage sur un noyau récent prenant en compte les *net-namespace* a toutefois été nécessaire.

Concernant le commutateur SDN OpenFlow, nous avons préféré choisir une version en espace utilisateur plus facile à modifier pour nos propres besoins. Le commutateur logiciel *OFSoftSwitch13*, également dénommé *Basic Open Flow User Space Switch*, a été utilisé dans d'autres travaux comme BEBA [62]. Il offre une implémentation pratiquement complète d'*OpenFlow* 1.3 et dispose d'une option *inband*. Des améliorations présentées dans le projet BEBA ont été appliquées afin d'améliorer les performances de ce commutateur. Mais il est à noter qu'elles restent significativement inférieures à un commutateur entièrement implémenté dans le noyau comme *OpenVSwitch*.

Enfin, le choix du contrôleur SDN s'est fait sur des critères de simplicité, autant pour l'implémentation d'applications SDN, que pour sa modification pour compléter le protocole *OpenFlow*, que pour la complexité du contrôleur. Ryu est un contrôleur écrit en Python, ce qui lui donne une grande souplesse de modification, au détriment sans doute de sa performance pure en quantité de requêtes traitées par seconde [88]. Notons toutefois qu'un réseau wifi ne demande pas les mêmes performances qu'un équivalent pour un réseau filaire.

L'application SDN qui gère la mobilité est limitée aux fonctions principales, sans chercher à optimiser systématiquement les routes. En particulier, la modification proactive des routes en cas de changements topologiques n'est pas mise en œuvre : ces changements requièrent donc l'émission d'un *PacketIn* par le nœud concerné. Il n'y a pas de routes alternatives précalculées.

3.2.1 Docker

Nous avons développé une première plateforme de test et d'évaluation sur la base de conteneurs Docker. L'objectif de cette première plateforme était de simplifier le développement des éléments nécessaires aux architectures étudiées. En effet, la mise en oeuvre de plusieurs drones ne nécessite pas de matériel supplémentaire, et leur synchronisation en est simplifiée.

Chaque nœud du réseau est ainsi totalement indépendant des autres d'un point de vue routage. L'un des nœud représente la station sol. Des outils de mesure classiques sont présents : *ping*, *iperf* et *wget*. La figure 3.6 montre les différents composants ainsi que les conteneurs dockers.

Le réseau wifi est préfiguré par un réseau interne Docker qui permet de relier différents conteneurs sans interaction avec le reste de la machine. La visibilité entre deux nœuds est contrôlée avec des filtres sur les adresses MAC.

Un script Python permet de contrôler l'ensemble et d'interagir avec les différents nœuds, soit au travers des commande docker, soit au travers d'une connexion Secured Shell (SSH). Il permet notamment : de démarrer un nœud ; de contrôler sa visibilité

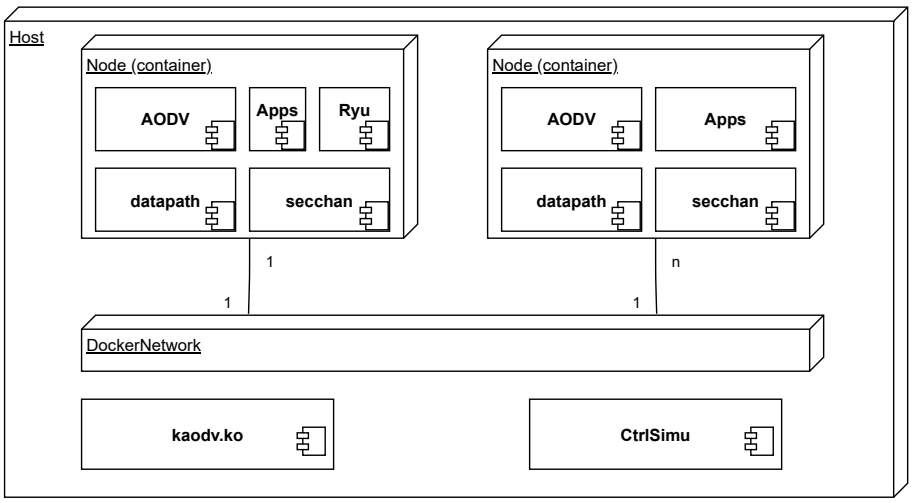


FIGURE 3.6 – Plateforme de test avec des containers Docker

avec les autres nœuds ; de lancer le rejeu d'une capture réseau en modifiant à la volée les adresses source et destination ; et d'effectuer différents tests.

3.2.2 Mini plateforme

La plateforme a ensuite été adaptée pour que chaque nœud soit déployé sur une machine réelle. Le réseau Docker a été remplacé par un réseau filaire commuté. La figure 3.7 présente le déploiement des différents éléments sur l'architecture de la plateforme.

Le contrôleur accède aux nœuds via une connexion SSH sur un réseau spécifique hors-bande. La visibilité radio entre les nœud est réalisée exactement comme avec la plateforme précédente.

Cette plateforme nécessite également une synchronisation horaire entre les nœuds afin de pouvoir comparer les traces et captures. Cette synchronisation est réalisée au travers de Network Time Protocol (NTP).

3.3 Mesures de performance

3.3.1 Objectif de l'évaluation

L'architecture envisagée fournit les outils nécessaires au contrôle des flux dans le réseau. Cet apport pour la sécurité du réseau entraîne toutefois une plus grande complexité et il est nécessaire d'en évaluer les performances pour en vérifier sa pertinence. Nous nous intéressons aux mesures suivantes :

— Le dialogue entre le contrôleur et les commutateurs génère des données supplémentaires. En particulier, l'installation d'une route engendre des transmissions

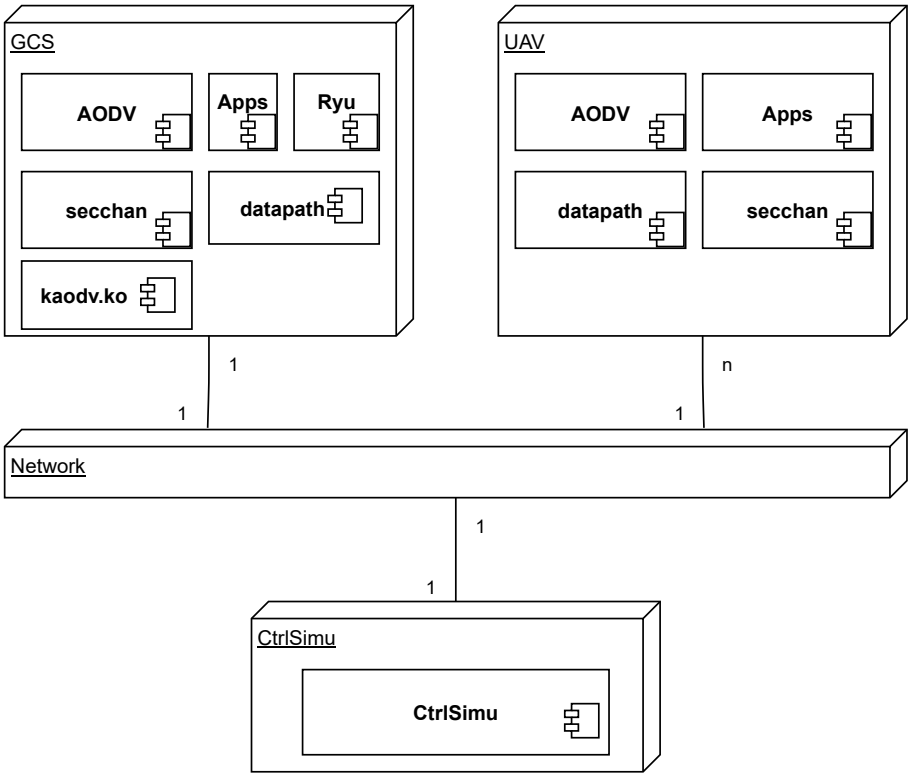


FIGURE 3.7 – Plateforme de test avec machines réelles, le wifi étant remplacé par un réseau filaire

depuis le contrôleur vers chaque nœud concerné. Nous en évaluons la proportion par rapport au trafic de mission.

— L'application des contrôles et des entrées de flux nécessite un traitement plus long de chaque paquet en comparaison du relayage IP habituel. Nous mesurons donc le temps nécessaire au relayage des paquets dans le réseau.

— Enfin, l'efficacité de la gestion de la mobilité est fortement liée au temps de convergence. Ce délai mesure le temps nécessaire au réseau pour revenir à un fonctionnement normal suite à une modification dans le réseau ; comme par exemple un changement de topologie. On considère généralement qu'il est constitué des délais suivants : la détection, la transmission, le calcul et la mise à jour. La mise en œuvre du SDN ayant potentiellement un impact sur chacun de ces constituants, nous mesurons ce délai.

3.3.2 Implémentation

Outre les choix d'implémentation déjà exposés précédemment, nous décrivons ici les détails d'implémentation principaux : tout d'abord la contrainte de rafraîchissement régulier des routes vers le contrôleur sur le plan de contrôle, ensuite les contraintes apportées par l'implémentation du SDN.

D'après le paragraphe 6.2 de [26], lorsqu'un paquet doit être émis par un routeur

AODV, il vérifie si une entrée correspondante est présente dans sa table de routage. Si ce n'est pas le cas, il crée une entrée dans cette table qui sera éventuellement mise à jour. La durée de vie d'une entrée dans la table est déterminée soit par le champ *lifetime* du paquet RREP, ou bien par la constante ACTIVE_ROUTE_TIMEOUT dont la valeur par défaut est de trois secondes. Notons que cette constante est utilisée dans le calcul d'autres paramètres du protocole. Sa modification n'est donc pas neutre.

La valeur à utiliser pour la durée de vie d'une entrée dans la table dépend de l'activité sur le lien *OpenFlow*. Dans notre scénario, les flux sont extrêmement constants (initialisés au début) et la mobilité au sein de l'essaim est relativement lente. Il s'ensuit une activité très modérée entre le contrôleur et les commutateurs, souvent au-delà des trois secondes. Pourtant, comme évoqué précédemment, la détection d'anomalies au sein du réseau nécessitera la collecte de données de surveillance sur le plan de contrôle, générant ainsi un trafic suffisant pour éviter l'expiration de la route. Dans notre scénario de test, nous avons simulé ces interactions de surveillance afin de maintenir l'activité sur le plan de contrôle. La méthode utilisée repose sur l'activation de messages de vie sur la connexion TCP (*KeepAlive*). L'émission de paquets de découverte (*Hello*) du protocole *OpenFlow*, un temps envisagé, n'a pas été retenue car elle n'était pas neutre et nécessitait des modifications protocolaires.

Les modifications de topologie sont transmises par chaque nœud dès que AODV les identifie. Pour mémoire, cette exploration de la topologie n'est pas réalisée par le contrôleur SDN comme cela est fait habituellement en utilisant par exemple un protocole comme LLDP. Toutefois, ces changements de topologie nécessitent une prise en compte la plus rapide possible. Par exemple, lors de la perte d'un voisin et s'il n'y a pas de modification des entrées dans la table de flux, le nœud continuera l'émission de trames à destination du voisin hors couverture radio. C'est pourquoi l'entité AODV dialogue avec le commutateur pour retirer les entrées de flux correspondantes. Dans notre implémentation, le *daemon* AODV se comporte comme un contrôleur local pour le commutateur qui informe ensuite de la suppression du flux au contrôleur principal avec une raison OFPRR_DELETE qui n'est pas spécifique ([47] paragraphe A.4.2).

Dans *OpenFlow* 1.3, le commutateur supprime les entrées de la table en comparant les critères de sélection du flux avec ceux fournis dans le champ *match* de la commande OFPFC_DELETE. Il faudrait que l'entité AODV connaisse ou interroge les flux pour déterminer ceux qui sont relayés vers le voisin hors de portée, puis qu'elle émette une commande de suppression pour chacun. Afin de réduire la complexité de cette opération, nous avons eu recours au champ *cookie* mis à disposition dans le protocole.

Ce champ est fourni par le contrôleur lors de la création d'une entrée de flux et

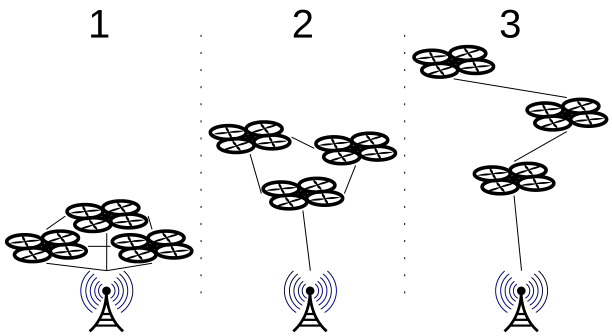


FIGURE 3.8 – Topologie de l'essaim et son évolution au cours du scénario d'évaluation des performances

mémorisé par le commutateur. La commande OFPFC_DELETE peut fournir une valeur pour ce champ afin de sélectionner les flux à effacer.

Nous y avons enregistré l'adresse de destination du flux. De cette manière, il est possible de supprimer en une seule commande l'ensemble des flux à destination du voisin en question.

3.3.3 Scénario d'évaluation

La vérification de la plateforme a été réalisée à partir de scénarios fixes : établissement du routage AODV sur le plan de contrôle ; initialisation et maintien de l'activité de la connexion entre le commutateur et le contrôleur ; création de flux sur le plan de données lors du lancement d'applications clientes.

En ce qui concerne l'évaluation des performances, il était nécessaire de développer un scénario dynamique réaliste. Nous avons envisagé une flotte de trois drones mobiles durant une mission. Nous présenterons d'abord la mobilité des nœuds les uns par rapport aux autres, puis nous décrirons l'aspect flux de données.

Mobilité

Nous avons considéré une mission d'exploration et d'observation d'une zone distante, utilisant la caméra embarquée. Nous avons séparé ce scénario en trois étapes que nous pouvons jouer en boucle, comme représenté sur la figure 3.8 : la flotte part depuis la station sol, transite vers la zone d'observation, puis se répartit géographiquement lors de la mission.

Lorsque les drones sont au sol ou proches de l'antenne de la station sol (phase 1), nous obtenons un réseau pleinement maillé. Durant le transit vers la zone (phase 2), la connectivité de tous les nœuds avec la station ne peut être maintenue mais l'essaim reste compact. Sur zone (phase 3), nous considérons que des ruptures de liaison sont possibles entre les drones et que les drones sont répartis sur la zone à

observer, s'éloignant autant que possible. Lorsque la mission est terminée, la flotte suit les même étapes en sens inverse.

Dans la figure précédente, les traits pleins représentent la connectivité, c'est à dire le fait que deux nœuds soient suffisamment proches pour pouvoir échanger des données. Cela ne représente ni le trafic applicatif ni les routes déterminées pour l'écouler. Comme dans tout réseau IP ad hoc, deux nœuds en visibilité radio s'échangent les données en direct. Si cette condition n'est pas vérifiée, alors le paquet est émis vers un troisième qui le relaiera au plus près de la destination. Les routes doivent donc être dynamiquement déterminées.

Nous avons exclu du scénario la possible scission du réseau en deux. Les liaisons sont rompues l'une après l'autre, sans isoler de nœud. Par ailleurs, les flux de données simulés sont uniquement en étoile, depuis un drone vers la station sol et vice-versa.

Dans la suite du document nous désignerons les drones par leur numéro d'ordre dans l'étape 3 du scénario. Ainsi, UAV1 est en visibilité de la station sol tout au long du scénario, alors que UAV3 va voir sa route avec la station sol changer au gré des pertes de liaison induites par la mobilité.

La situation initiale du scénario est que chaque drone a la visibilité directe avec la station sol et échange des données applicatives avec elle. Les événements qui sont simulés sur ce scénario sont les suivants :

— UAV3 et UAV2 perdent la liaison directe qu'ils avaient avec la station sol, et doivent utiliser UAV1 comme nœud intermédiaire pour relayer leurs paquets vers la station sol, nécessitant ainsi un second saut,

— UAV3 perd la liaison avec UAV1 et utilise UAV2 comme nœud intermédiaire, la route ayant maintenant trois sauts jusqu'à la station sol,

— UAV3 retrouve la liaison directe avec UAV1,

— UAV3 est à nouveau en visibilité de la station sol.

Dans les deux derniers changements topologiques, notre application SDN de mobilité étant simplifiée, nous n'avons pas mis en place d'optimisation proactive des routes, c'est pourquoi en l'absence de perte de liaison, le chemin SDN aurait été conservé en l'état. Afin de générer deux événement supplémentaires et forcer le contrôleur à basculer sur la liaison optimale, la liaison directe avec le prochain saut a été systématiquement rompue.

Ce scénario ne visait pas la mesure des délais d'établissement des routes. Le nombre d'événements de changement topologiques reste relativement limité. Par ailleurs, le délai d'établissement dépend du type de changement topologique et sa mesure nécessiterait donc une grande diversité d'évènements.

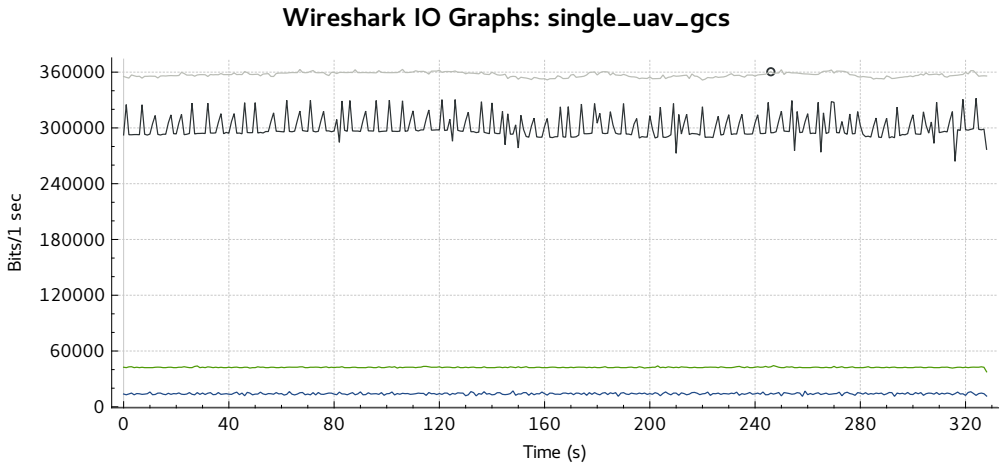


FIGURE 3.9 – Caractérisation du trafic échangé pour chaque nœud durant le scénario d'évaluation des performances

Trafic réseau

Afin de rendre le trafic réseau le plus représentatif possible, nous utilisons un enregistrement réel effectué sur un drone seul, et le faisons rejouer par chacun des participants au scénario décrit ci-dessus. Ces données sont donc par nature représentatives d'un trafic de drone dans une mission de surveillance ou d'observation avec caméras. On peut imaginer d'autres types de missions pour un essaim de drones. Le scénario ne peut être représentatif de chacun d'eux. Toutefois, toute mission où des flux vont des capteurs jusqu'à la station sol seront proches de la situation décrite ici, aux débits près.

Nous avons capturé le trafic entre la station sol et un drone réel qui diffusait le flux d'images de sa caméra en IEEE 802.11. Pour ce faire, nous avons utilisé le logiciel open-source Paparazzi pour le pilote automatique et la station sol, générant ainsi un trafic de commande et contrôle ainsi que de la télémétrie. La figure 3.9 montre les débits induits par ces flux dans la capture, ainsi que le débit total et montre ainsi les flux suivants :

— un flux vidéo UDP/RTP à 10 images par seconde conforme à [89], représentant environ 300kbps ;

— un flux de télémétrie et autres informations du drone vers la station sol, pour environ 14kbps ;

— un flux de commande et contrôle de la station sol vers le drone qui atteint environ 42kbps.

La plupart des paquets transmis sont de petite taille et utilisent UDP comme protocole transport. Le flux vidéo engendre quant à lui des paquets pouvant aller jusqu'à 1500 octets. Au total, le débit généré est d'environ 356kbps en incluant le surdébit protocolaire. La station sol transmet ainsi environ 95 petits paquets par

seconde (moins de 27 octets de données applicatives) alors que le drone en génère 45.

En émission depuis le drone, on compte environ 43% de paquets dont la taille est inférieure à 110 octets et environ 38% dont la taille est de 1462. La médiane est de 1162 octets : aucun paquet n'a de taille intermédiaire entre environ 110 et 1110 octets. En émission depuis la station sol, les paquets font 50, 56 ou 68 octets mais la plupart (99.7%) ont la taille de l'intermédiaire.

Durant le scénario d'évaluation des performances, nous faisons rejouer ce même trafic par chaque drone de la topologie. La station sol reçoit donc les flux vidéo des trois drones ainsi que la télémétrie et émet autant de flux pour la commande et le contrôle.

Résultats des évaluations

Nous avons comparé les performances des deux architectures réseau suivantes : un réseau purement AODV (notre référence de comparaison), et l'architecture SDN avec plan de contrôle dans la bande et routé par AODV.

En utilisant la plateforme décrite dans les sections précédentes et le scénario ci-dessus, nous avons mesuré les performances de délai de bout en bout. Les mesures ont été réalisées hors-ligne à partir des fichiers de capture obtenus sur différents points d'écoute.

Ici, nous avons comparé le délai de traversée du réseau et la disponibilité liée à la gestion de la mobilité. Le débit maximum n'a pas pu être mesuré, et nous ne cherchons pas ici à étudier la résilience de l'architecture par rapport à des attaques.

La figure 3.10 montre la fonction de distribution cumulative du délai entre la station sol et le drone UAV3 au cours du scénario dynamique, pour le sens montant et le sens descendant, dans chacune des deux architectures.

Les délais ont été obtenus par comparaison des fichiers de capture de la station sol et du drone le plus éloigné au milieu du scénario. Le délai calculé ne prend en compte que l'aller. Pour cela, la correspondance entre un paquet émis et le même paquet reçu est faite sur la base des adresses IP, de l'identifiant de paquet dans l'entête IP, ainsi que du protocole transport et de la somme de contrôle transport. Malgré la synchronisation horaire entre les machines, il subsiste un écart de l'ordre de quelques microsecondes entre les deux stations. Il en résulte l'apparition de valeurs négatives qui sont liées au manque de précision de la synchronisation horaire des différents systèmes.

Par ailleurs, les délais sont obtenus par différence entre les datations des paquets fournis par les fichiers de capture. La datation d'un paquet est faite juste avant l'émission ou juste après la réception par la carte réseau. La différence de date ne prend pas en compte les délais induits par l'application et le noyau (protocoles des

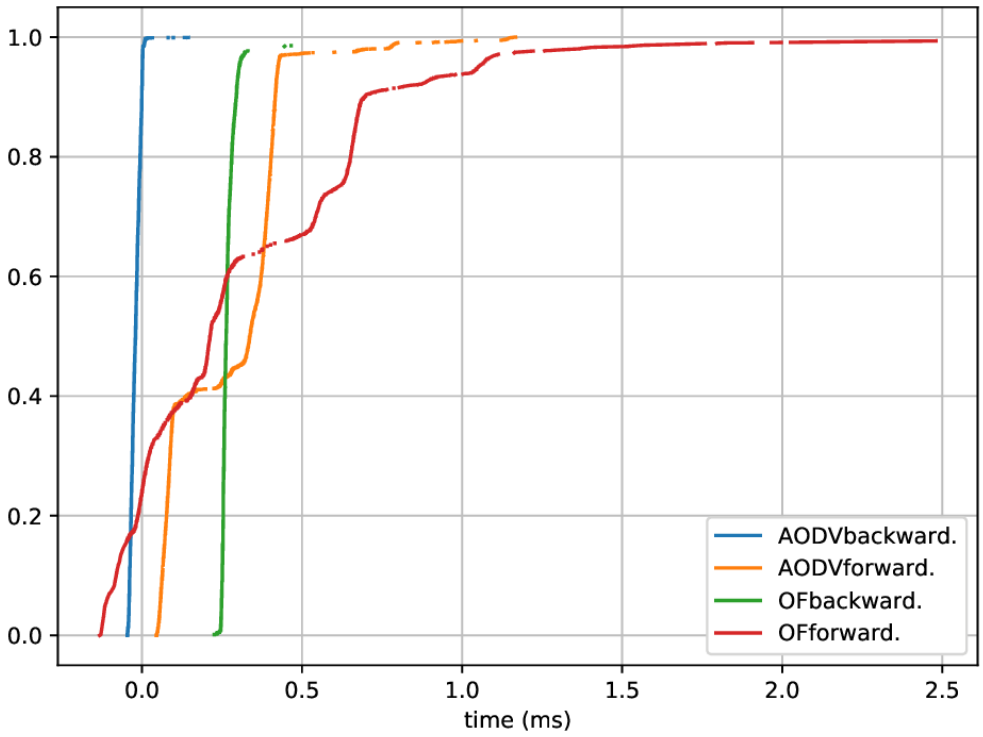


FIGURE 3.10 – Fonction de distribution cumulative des délais obtenus sur l'ensemble du scénario

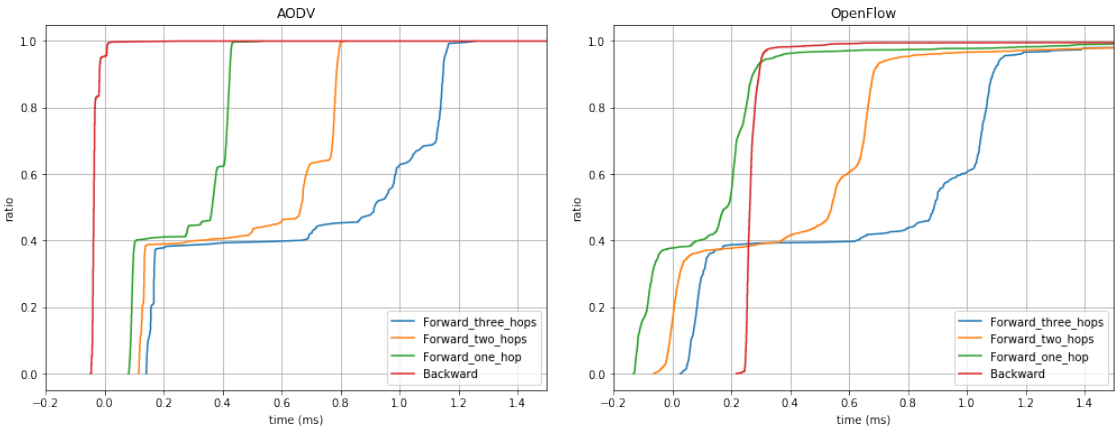
couches transport et réseau). Ils sont donc inférieurs à ceux que l'on aurait obtenus avec la commande ping.

Les délais mesurés lorsqu'il n'y a aucun saut permettent toutefois une évaluation de l'écart de synchronisation horaire. Nous avons vu que le trafic généré contenait des paquets de très petite taille autant dans le sens montant que dans le sens descendant. On peut considérer que l'écart horaire est représenté par l'écart entre les délais minimums montant et descendant. En effet, le délai de A vers B δT_{AB} peut s'exprimer sous la forme $\delta T_{AB} = T_E + \delta H$ où T_E est le temps à l'émission, δH est l'écart d'horloge. Le délai de B vers A est alors : $\delta T_{BA} = T_E - \delta H$ et on en déduit : $\delta H = (\delta T_{AB} - \delta T_{BA})/2$.

Pour le scénario AODV, nous obtenons une évaluation d'écart d'horloge d'environ 75μs et pour le scénario SDN, de 175μs.

On remarque tout d'abord deux marches. La première représente environ 40%. Elle correspond bien aux caractéristiques du trafic généré où on trouve la même proportion de paquets de très petite taille. Le second palier correspond lui à la taille maximale. D'ailleurs, en calculant la corrélation entre longueur et délai minimal par longueur, on constate un assez fort coefficient de corrélation (0.67).

Ces marches sont encore plus évidentes si l'on différencie le délai constaté en fonction du nombre de sauts. Ces trois courbes permettent également de mesurer le temps de relayage d'un nœud, tout en annulant l'effet de l'écart de synchronisation



(a) AODV (b) OpenFlow

FIGURE 3.11 – Fonction de distribution cumulative des délais en fonction du nombre de sauts entre la source et la destination

Sauts	**Taille**	**AODV**	**SDN**
2	<110	0.034	0.069
	1k	0.361	0.414
3	<110	0.060	0.159
	1k	0.72	0.817

TABLE 3.1 – Délais (ms) dûs aux relais en AODV et en SDN

horaire.

La figure 3.11 ainsi que la table 3.1 reprennent l'ensemble de ces mesures. On évalue ainsi que le nœud SDN (*OpenFlow*) met un peu plus de 35μs de plus pour traiter et retransmettre un paquet de taille minimale qu'un nœud AODV. En faisant le calcul équivalent sur la seconde marche, on confirme que la proportion n'est pas conservée puisque la majorité du délai est due au temps de sérialisation indépendant de SDN ou AODV. Mais il faut 50μs de plus pour *OpenFlow* en comparaison de AODV. Ce dernier semble donc moins sensible à la longueur du paquet. Il est à noter que le choix d'un commutateur SDN en espace utilisateur a un impact négatif sur ses performances.

Toutefois, les délais restent comparables et relativement négligeables en comparaison du délai que la couche liaison pourrait induire dans le cas d'un réseau sans fil en semi-duplex [90]. Ces mesures sont à replacer dans le contexte de cette émulation où la puissance de calcul des ordinateurs utilisés est supérieure à celle que l'on pourrait rencontrer sur un drone.

L'implémentation choisie s'appuie sur les routines du noyau Linux de la pile IP. Les délais sont de l'ordre de la milliseconde sur notre plateforme.

Les pertes sont principalement dues aux changements de topologie du réseau

		AODV		SDN	
direction	**#**	**durée**	**paquets**	**durée**	**paquets**
GCS to UAV	1	1.012s	97	1.431s	136
(95pps)	2	1.149s	110	1.964s	188
	3	0s	0	1.169s	112
	4	0s	0	1.460s	141
UAV to GCS	1	1.008s	52	1.396s	73
(48pps)	2	0.449s	25	1.943s	96
	3	1.159s	52	1.104s	60
	4	0s	0	0s	0

TABLE 3.2 – Pertes de liaison au cours du scénario avec mobilité

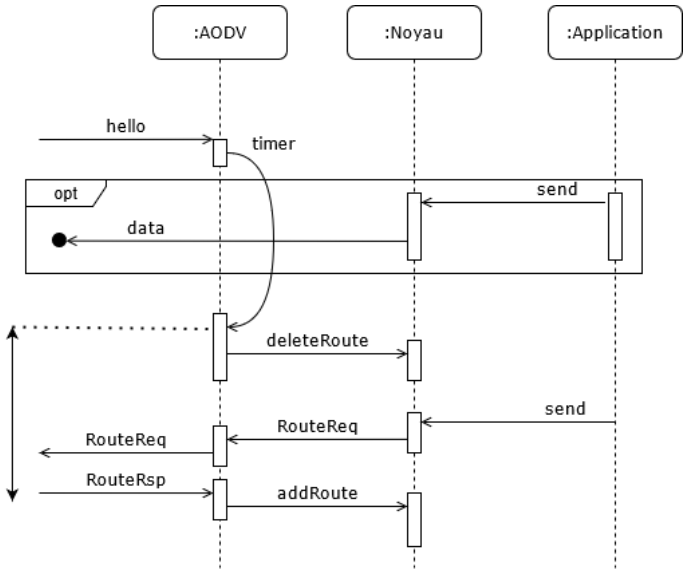


FIGURE 3.12 – Diagramme de séquence du rétablissement du routage en AODV pur, montrant d'éventuelles pertes de paquets

et au temps nécessaire à leur détection par AODV. Ce temps peut être estimé en première approche entre une et deux secondes [87] en considérant que le délai entre deux émissions de *Hello* est d'une seconde. Entre un changement topologique et sa détection, les paquets concernés sont perdus et n'apparaissent donc pas dans la figure 3.11. Dès qu'AODV détecte la perte il émet un nouveau RREQ, et les nouveaux paquets sont mis en file dans l'attente du rétablissement d'une route.

La table 3.2 montre la durée des pertes de paquets. On constate en effet que ces délais sont compris entre 1 et 2 secondes. La durée des périodes de pertes de paquets pour SDN est généralement plus longue de plusieurs centaines de millisecondes et un nombre de paquets perdus plus important.

Comme illustré dans la figure 3.12, le temps de rétablissement du routage en

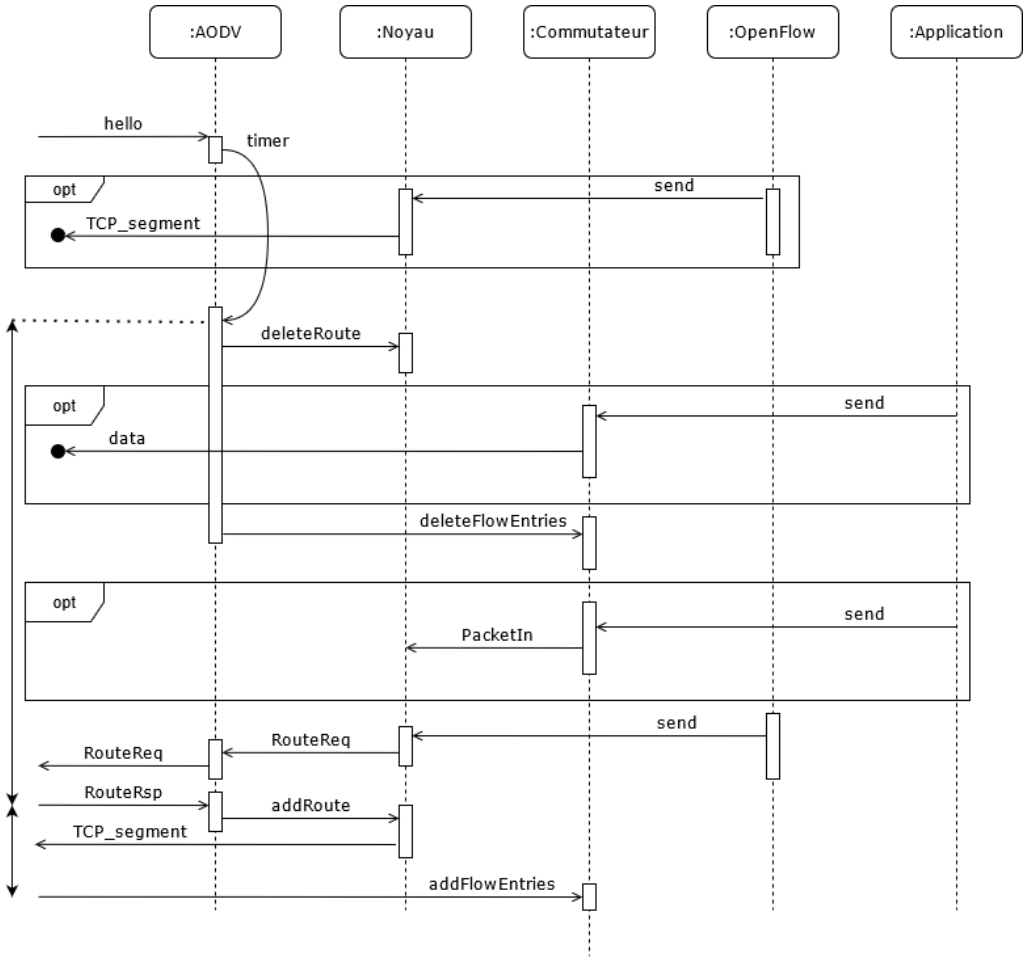


FIGURE 3.13 – Diagramme de séquence du rétablissement du routage pour l'architecture SDN, montrant d'éventuelles pertes de paquets sur le plan de contrôle et sur le plan de données

AODV pur dépend principalement du temps de détection de la perte de voisinage. Une fois celle-ci identifiée, le module noyau détruit les routes passant par le voisin disparu et met en file tout nouveau paquet empruntant ces dernières. Une requête de route est alors émise et les transmissions se rétablissement dès qu'une réponse est reçue. Cette mise en file induit donc un délais supplémentaire mais aucune perte. Notons que toutes les transmissions (applicatives ou de contrôle) passent par UDP dans cette architecture.

La figure 3.13 illustre le cas de l'architecture SDN. Le délai supplémentaire dans ce cas a principalement deux causes. Tout d'abord, le rétablissement de la route vers le contrôleur ne suffit pas à rétablir le trafic applicatif : il faut encore transmettre les données vers le contrôleur, que ce dernier prenne en compte le changement de topologie et y réponde en transmettant de nouvelles entrées de flux. Mais avant cela, la communication OpenFlow utilisant TCP comme protocole transport, il peut apparaître des délais de retransmission de paquets dûs au temps nécessaire pour détecter les pertes, puis d'en demander la retransmission.

	Total			Spécifique		
Protocole	**PDU**	**Octets**	**Débit** (bps)	**PDU**	**Octets**	**Débit** (bps)
Ethernet	487	6818	571	-	-	-
IPv4	487	9740	816	-	-	-
UDP	98	784	66	-	-	-
AODV	98	2708	227	98	2708	227
TCP	389	36104	3024	281	20940	1754
OpenFlow	108	13124	1099	108	13124	1099

TABLE 3.3 – Détail des quantités de données échangées sur le plan de contrôle en fonction du protocole

Le nombre plus important de paquets perdus en SDN s'explique en partie par ces délais supplémentaires. En effet, les paquets perdus doivent être envoyés au contrôleur pour que ce dernier les relaie vers la destination (*PacketIn*). Dans notre implémentation, ces paquets sont simplement ignorés et donc perdus. Ensuite, il peut y avoir un léger décalage entre la détection de la perte de voisinage par AODV et la destruction des entrées correspondantes dans la table de flux, et des pertes additionnelles peuvent apparaître dans cet intervalle. Ce délai de traitement reste toutefois très limité.

Dans certains cas, AODV obtient une route alternative avant même que la coupure de la liaison ne survienne. Les données sont routées alors directement via cette nouvelle route et le changement de topologie n'induit pas de pertes. Au contraire, en SDN, c'est au contrôleur de recalculer les routes en cas de changement de topologie. En absence de modification proactive du routage par le contrôleur SDN génère des pertes.

Ce délai minimal d'une seconde pourrait être réduit par l'utilisation du retour d'information de la couche liaison de données [87]. Ceci ne peut toutefois pas être testé sur le réseau filaire de la plateforme.

Notons également que lors de la perte de liaison entre un nœud et le contrôleur, le protocole transport (TCP) peut induire des délais supplémentaires en cas de perte de paquets. Toute nouvelle transmission ne sera traitée par le contrôleur que lorsque toutes les données précédemment perdues auront été retransmises et correctement reçues. Cela peut engendrer des délais supplémentaires et pourrait être évité en utilisant une connexion auxiliaire sur un protocole transport en mode datagramme.

On constate également que la solution SDN génère plus de trafic que l'architecture AODV. Le tableau 3.3 reprend les mesures de quantité de données et de débit sur l'architecture SDN. Dans notre scénario où le trafic est en étoile, le pro-

tocole de routage AODV n'a pas besoin d'établir plus de routes qu'en SDN. Par contre, SDN génère du trafic pour la mise en place du routage (paquets *FlowMod*) ce qui représente un trafic OpenFlow d'environ 1100bps en moyenne sur ce scénario, soit environ 1660bps au niveau Ethernet. De plus, nous avons utilisé les messages de vie TCP pour maintenir artificiellement les routes en place pour un peu plus de 1750bps, acquittements TCP compris, soit environ 2680bps au niveau Ethernet. En comparaison du trafic applicatif estimé à 356kbps, cela représente moins de 1 pourcent.

Bien qu'il paraisse probable que l'architecture SDN soit plus consommatrice en données, le scénario choisi ne permet pas de conclure sur ce point.

3.4 Synthèse

Nous avons proposé plusieurs architectures en vue d'assurer les communications au sein d'un essaim de drones, tout en apportant des fonctionnalités permettant leur sécurisation. Nous avons sélectionné deux de ces architectures et les avons étudié. La première est basée seulement sur AODV et est assez proche des protocoles habituellement utilisés pour des réseaux de drones. Elle nécessite de définir comment surveiller le réseau et mettre en place des mesures de protection. La seconde utilise l'approche SDN et apporte donc une grande souplesse et des outils utiles dans le domaine de la sécurité des réseaux.

Cette étude visait à estimer les performances de ces deux architectures dans un scénario type : délais de transmission, durée d'indisponibilité en cas de changement de topologie due à la mobilité, quantité de données perdues lors de ces événements, débits supplémentaires induits par les architectures.

Cette étude montre un ratio entre les performances de l'architecture et les apports pour la sécurisation du réseau, très en faveur de l'architecture SDN, sans avoir à définir de nouveaux protocoles pour les besoins en sécurité réseau.

La grande souplesse de SDN peut être mise à profit pour implémenter d'autres stratégies de routage (redondance, multicast, géorouting par exemple) à moindre effort. D'ailleurs, l'application SDN n'est pas limitée aux données collectées par OpenFlow et d'autres sources de données peuvent être agrégées pour améliorer le routage dans le réseau.

Surtout, cette grande souplesse et l'agrégation de l'ensemble des données du réseau représentent une opportunité également dans le domaine de la sécurité et de la protection contre les attaques. La figure 3.14 reprend les éléments de l'architecture et la transpose dans le contexte de la sécurisation du réseau d'un essaim de drones. Une fois un routage sécurisé mis en place sur le plan de contrôle (1), une connexion protégée est établie entre la station sol et chaque drone sur la base d'une

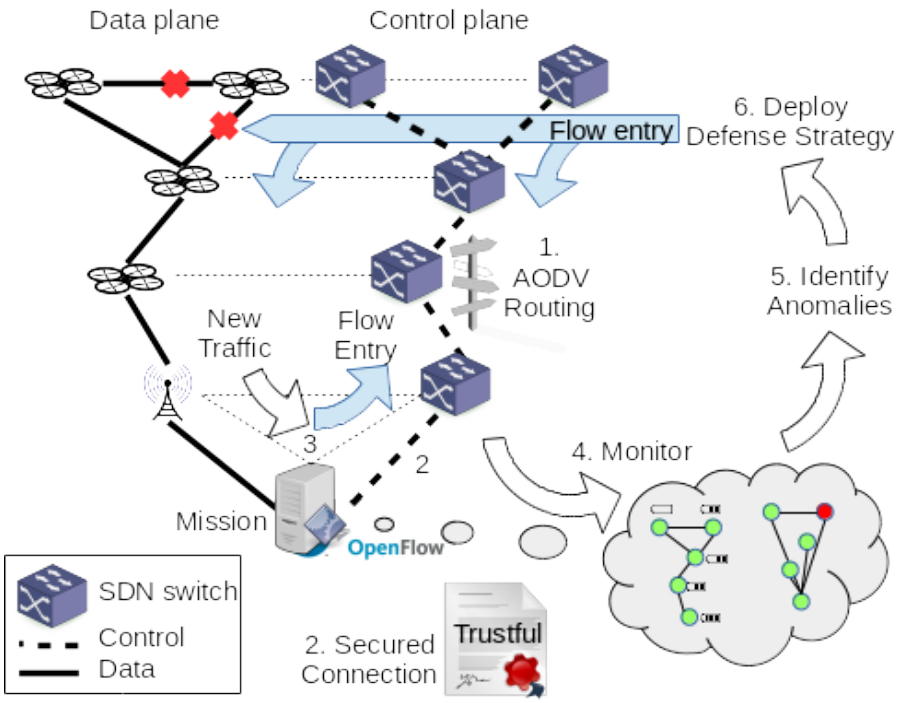


FIGURE 3.14 – Vue d'ensemble de l'architecture et cas d'utilisation pour l'application de la sécurité sur le réseau d'un essaim de drones

authentification sécurisée (2). Le contrôleur gère l'ensemble du réseau sur la base des demandes d'établissement de flux entre les nœuds du réseau (3). Il collecte également les informations nécessaires, via SDN ou bien via une application liée à la mission (télémétrie par exemple) pour former une vue globale de l'essaim (4). Ces données sont utilisées dans la détection et l'identification d'attaques ou de comportements anormaux (5). Enfin le déploiement de mesures de protection afin d'assurer la pérennité de la mission utilise les mécanisme SDN (6).

A partir de cette architecture, il est possible de mettre en œuvre une surveillance de l'activité et des comportements dans le réseau à partir de l'ensemble des informations collectées.

Une fois détectée, SDN apporte également les outils pour mettre en œuvre des actions de protection contre les différentes attaques.

Différentes études et optimisations pourraient améliorer les performances de notre implémentation :

— l'utilisation de techniques plus rapides que la détection de la perte de voisinage basée sur des paquets *Hello* espacés de 1 seconde (retour d'information de la couche liaison),

— l'étude d'autres protocoles qu'AODV compte tenu de la nature très spécifique du routage (en étoile),

— la mise en œuvre de commutateurs en espace noyau et d'un contrôleur SDN plus performants, pour réduire les délais d'acheminement des paquets,

	AODV sécurisé avec iptables	SDN
Délais	Minimaux	30μs par nœud traversé
Convergence	Minimale	+0.5 à 1.5s
Surcharge	Limitée (hors supervision)	+1%
Routage	Distribué	Centralisé
Complexité	Réduite	Complexe
Protection	Statique et générique[1]	Dynamique et spécifique[2]
Supervision	À définir	Incluse dans SDN[3]
Réponse	A posteriori	A priori
Mise en œuvre	À définir (p. ex. SSH)	Incluse dans SDN

[1] Définition a priori des ports ouverts, par mission, indépendamment du besoin de chaque drone.
[2] Les ports sont ouverts à la demande et de manière automatique, pour chacun des drones indépendamment.
[3] SDN n'est pas un protocole de supervision, mais permet d'obtenir un ensemble de données statistiques ainsi que des événements pertinents pour la sécurité, comme il sera démontré par la suite.

TABLE 3.4 – Comparatif des deux architectures par rapport au scénario de test

— l'utilisation de connexions auxiliaires au travers d'un protocole transport sécurisé sans connexion comme Datagram Transport Layer Security (DTLS) pour réduire le délais d'acheminement suite à une perte de voisinage,

— le développement d'une application SDN traitant de la mobilité de manière proactive afin de se rapprocher des performances AODV en terme d'indisponibilité suite à un changement de topologie du réseau.

Le tableau 3.4 propose un comparatif synthétique des architectures AODV avec iptables et SDN. Cette comparaison est basée sur le scénario de test et les mesures précédemment présentées. La seconde partie du tableau propose un comparatif étendu aux caractéristiques principales de ces deux architectures dans la perspective de la mise en œuvre d'une détection d'attaques et des contre-mesures associées.

Chapitre 4

Application à la sécurité

Dans ce chapitre, nous nous attachons à envisager l'architecture proposée précédemment dans un contexte cybersécurité. Pour cela, nous commencerons par décrire le modèle de sécurité envisagé et les hypothèses de l'étude. Nous dégageons ensuite deux scénarios d'attaque : de l'extérieur ou de l'intérieur du réseau de la flotte.

La seconde partie de ce chapitre reprend l'architecture telle que décrite précédemment et en étudie les avantages pour contrer les attaques du premier scénario. Les principales menaces restantes sont ainsi identifiés. Ensuite, nous proposons une solution pour détecter et contrer les injections de trafic en utilisant les données disponibles au travers du protocole OpenFlow. Une démonstration de la méthode clos cette partie.

Enfin, nous étudions la possibilité de contrer les attaques dites internes en nous basant uniquement sur les données disponibles au niveau du contrôleur SDN. Nous proposons différentes métriques sur l'activité et le comportement des nœuds dans le réseau. Ensuite, en nous basant sur un ensemble de données issues d'un réseau, nous testons les différentes métriques afin d'en trouver les plus pertinentes dans le cadre d'un modèle de classification par apprentissage automatique. Nous concluons cette partie par la validation du modèle sur des données représentatives d'une mission de flotte de drones.

Sommaire

Avant tout il est nécessaire de préciser le cadre dans lequel nous envisageons cette analyse, qui correspond à un environnement réaliste de l'utilisation d'un essaim de drones.

4.1 Caractérisation des menaces

Nous détaillerons tout d'abord le modèle de sécurité de la présente analyse et identifierons ainsi clairement les hypothèses prises.

4.1.1 Hypothèses de l'étude

Nous considérerons un essaim de drones d'une taille limitée à une dizaine de nœuds ; chacun exécutant un nombre limité d'applications : contrôle du drone, télémétrie, plus deux à trois applications liées à la mission. Cette hypothèse semble raisonnable compte tenu de la spécialisation nécessaire des drones aux charges qui leurs sont allouées, ainsi qu'aux débits disponibles sur un réseau sans fil ad hoc. Les missions qui seront envisagées ici ont une durée limitée à quelques heures. Enfin nous n'envisagerons pas la possibilité qu'un attaquant puisse avoir un accès physique au drone.

Cette solution serait sans doute applicable dans un cadre plus large avec plus de nœuds ou d'applications, mais l'étude de sa mise à l'échelle est laissée pour une étude ultérieure.

Compte tenu de l'utilisation de réseaux sans fil, nous considérerons que l'attaquant à un libre accès en écoute et en émission au canal de communication. L'attaquant est donc à même de recevoir les transmissions des drones, dans les limites imposées par les phénomènes physiques de propagation. Il peut également transmettre des signaux qui seront reçus par les drones. Nous considérerons toutefois que l'attaquant ne dispose pas de capacités de calcul illimitées.

Comme présenté précédemment, l'architecture proposée repose sur des technologies de réseau logiciel. La confidentialité de son plan de contrôle est assuré par un protocole de type Transport Layer Security (TLS) ou DTLS. Nous considérerons qu'une infrastructure de clés publiques (PKI) est en place et permet une authentification sûre des différentes parties. De plus, les clés seront créées spécifiquement pour la durée de la mission. Enfin, les algorithmes de cryptage et la longueur des clés utilisés seront suffisants pour que l'attaquant, compte tenu de ses capacités de calcul, ne soit pas en mesure de casser ces codes sur la durée de la mission. Cette limite dépend de la puissance de calcul raisonnablement atteignable et n'est donc pas fixe dans le temps. Ainsi, bien qu'ayant accès au canal, il lui sera impossible de lire, d'altérer ou de produire des données sur le plan de contrôle de notre réseau.

Nous considérerons dans cette analyse uniquement les aspects liés au réseau. Les systèmes d'exploitation et les applications présentes sur les nœuds du réseau seront supposées libres de toute faille connue et inconnue, et que des mécanismes d'authentification seront en place et correctement configurés de manière à ce qu'aucun accès non autorisé au système ne soit possible. Nous n'envisagerons pas les méthodes d'escalade de privilège.

De la même façon, les composants du réseau logiciel ne peuvent être modifiés ou altérés, et sont libres de failles connues ou inconnues. Si d'autres applications SDN sont hébergées par le contrôleur, les flux générés sont cohérents et n'engendrent ni conflits ni recouvrements qu'un attaquant pourrait exploiter. Ceci pourrait nécessiter des composants logiciels supplémentaires qui ne sont pas décrits ici.

Le libre accès au plan de données rend la plupart des applications potentiellement vulnérables en l'absence de mesures supplémentaires. Les travaux présentés dans ce mémoire apportent des solutions au niveau du réseau pour renforcer la sécurité au sein de l'essaim de drones, mais ne sauraient prémunir les applications embraquées contre tous les types d'attaques. Elles doivent avant tout avoir été conçues dans cette optique.

L'utilisation de protocoles transport sécurisés implique que seuls les nœuds légitimes du réseau ont accès au plan de contrôle. Il en résulte que nous devons traiter deux scénarios distincts selon si le nœud attaquant est interne ou externe au réseau.

4.1.2 Attaques depuis un noeud extérieur

Le cas des attaques depuis un nœud extérieur au réseau est sans doute le plus naturel à envisager. Un attaquant, équipé d'un récepteur capte et peut ainsi analyser les transmissions, ce qui représente une atteinte à la confidentialité des données. On parle alors d'écoute passive. Celle-ci permet par ailleurs l'identification des nœuds du réseau et des applications utilisées.

S'il est également équipé d'un émetteur, il peut alors mener d'autres types d'attaques en transmettant à son tour.

Parmi les attaques que l'on envisage ici, on peut lister : des tentatives pour obtenir l'empreinte numérique d'un drone dans le but d'exploiter des failles connues ; des attaques en force brute pour deviner un mot de passe ; ou simplement un déni de service par inondation.

L'attaquant peut également chercher à influencer le relayage des paquets dans le réseau pour devenir un point de passage. On parle ici d'attaques de «l'homme du milieu». Dans un réseau Ethernet/IP, ceci peut être réalisé par exemple par une attaque de corruption du cache ARP en influant sur l'association entre l'adresse IP et l'adresse Ethernet. Il est également possible d'intervenir au niveau IP avec

l'émission d'une redirection ICMP. D'autres attaques sont possibles en visant les protocoles de routage utilisés.

Comme l'écoute passive est possible sans avoir à influer sur le relayage, le but de l'attaquant dans ces cas sera soit l'altération de données en relayant un paquet modifié, soit un déni de service en ne retransmettant pas les paquets reçus vers leur véritable destinataire. Dans les deux cas, il est essentiel que la cible soit amenée à transmettre les données à destination de l'adresse MAC de l'attaquant. Cette remarque implique que l'adresse Ethernet de l'attaquant est nécessairement spécifique, ce qui est particulier à ce scénario d'attaque depuis l'extérieur du réseau.

Il n'est toutefois pas exclu que l'attaquant se fasse passer pour un nœud existant dans le réseau. On parle alors d'usurpation d'adresses. En particulier, émettre des commandes frauduleuses vers le drone peut se faire avec ou sans usurpation d'identité : l'injection de trafic peut permettre de prendre le contrôle de la trajectoire d'un drone.

4.1.3 Attaques depuis un nœud interne

Les attaques depuis l'intérieur semblent moins évidentes au premier abord, mais font partie des menaces potentielles identifiées, et incluent l'utilisation de virus ou de vers. Elles se décomposent donc en deux phases. Une première où il s'agit de compromettre ou infecter un nœud du réseau, même partiellement. Une seconde dont le but sera de mener des attaques depuis celui-ci à destination d'autres nœuds ou dans le but de porter atteinte au réseau en lui même. C'est cette seconde étape que nous envisageons par la suite, au moment où l'attaquant cherche à tirer profit de la situation.

Il est bien sûr possible que l'attaquant cherche à porter atteinte au drone infecté et il pourrait être souhaitable de pouvoir traiter ce cas. Toutefois, s'il s'agit de rendre le drone inopérant par une bombe logique, cette dernière ne nécessitera vraisemblablement aucune activité sur le réseau et nous sommes de facto hors du cadre de cette thèse. D'autre part, tout accès réseau vers l'extérieur a été traité dans la partie précédente et sera étudié comme tel. Il reste donc l'hypothèse d'une prise de commande à distance, qui nécessiterait un accès vers l'extérieur (Internet) via la station sol. Ce point est évoqué plus loin avec l'utilisation d'un pare-feu.

Nous envisageons ici les mêmes attaques que dans le cas d'un nœud externe. Mais il est question ici de nœuds réseau ayant une adresse connue et légitime dans le réseau. Bien que l'on puisse exécuter ces mêmes attaques, certaines requièrent un niveau de privilège élevé, soit pour placer la carte dans un mode espion, soit pour émettre des trames façonnées idoines. En particulier, l'écoute passive et l'usurpation d'identité semblent exclues a priori par les hypothèses du modèle de sécurité qui

Attaque	Externe	Interne
Écoute passive	Oui	Non[1]
Usurpation d'identité	Oui	Non[1,2]
Homme du milieu	Oui	Non[1]
Empreinte numérique	Oui	Oui
Déni de service	Oui	Oui
Force brute	Oui	Oui

[1] Nécessité de privilèges élevés
[2] Moindre intérêt

TABLE 4.1 – Catégories d'attaques

écarte l'accès aux droits super-utilisateur. Il en est de même avec des attaques de l'homme du milieu.

Dans un tel scénario, l'attaquant aura sans doute recours à des techniques d'intrusion usuelles. Il lui faudra identifier son environnement (c'est à dire des machines qui pourraient être attaquées), puis rechercher une faille par l'identification du type de système d'exploitation et la liste des applications hébergées, et enfin tenter de corrompre la victime. Cette corruption peut être réalisée par des moyens spécifiques à l'application (exécution de code par dépassement de pile) ; des méthodes génériques (injection SQL par exemple [20]) ; ou simplement par l'obtention d'un mot de passe.

Le déni de service est une autre possibilité pour un nœud interne de porter atteinte à la mission de l'essaim. On pense principalement à des techniques comme l'inondation SYN en TCP. Les attaques en déni de service distribué ne pourraient pas profiter d'un nombre important de relais et ne sont pas parmi les scénarios privilégiés. Toutefois, compte tenu des ressources limitées d'un drone et de la possibilité d'exploiter une application avec un fort bras de levier, il ne peut être exclu totalement.

Le tableau 4.1 reprend l'ensemble de ces éléments.

4.2 L'architecture face aux menaces

Ayant détaillé et caractérisé les attaques sur un essaim de drones, nous allons faire une comparaison de la vulnérabilité d'une architecture purement AODV à une architecture SDN/AODV telle que présentée dans le chapitre précédent face aux menaces identifiées. Nous allons donc décrire leur mise en œuvre et étudier en quoi l'architecture basée sur du SDN apporte des réponses face aux attaques.

4.2.1 Cas des attaques externes

Dans un réseau purement AODV, chaque nœud prend ses décisions sur le traitement des paquets de manière totalement autonome. Chaque nœud est donc une cible potentielle pour l'attaquant.

L'authentification des transmissions n'est pas assurée par défaut par le protocole réseau. L'identification des transmissions venant d'un nœud externe nécessiterait la mise en place de techniques additionnelles et la diffusion à chacun d'informations sur l'identité des participants du réseau. A défaut, l'usurpation d'adresse est donc possible et rien ne permet de s'en défendre.

Il existe des attaques visant spécifiquement le protocole AODV mais des solutions de sécurisation de ce protocole existent [31] [32] [91].

Au contraire, dans l'architecture SDN, c'est au contrôleur que revient la responsabilité d'accepter chaque demande de flux reçue au travers d'une communication authentifiée et sécurisée. Si cette centralisation représente un risque en terme de disponibilité, elle apporte en contrepartie un avantage certain dans la connaissance de facto exhaustive des différents participants. De plus, le fait d'avoir le commutateur SDN au niveau de chaque nœud permet également la vérification de la source du paquet initial d'un flux et donc, la possibilité de le rejeter s'il n'émane pas d'un des composants en lisière du réseau. Autrement dit, même en cas d'usurpation d'adresse, il est possible de vérifier si l'adresse source d'un flux correspond bien à la source elle même. Dans la négative, le paquet sera rejeté.

Comme dans le cas précédent, l'attaque du protocole de routage AODV est possible et sa sécurisation réduit en grande partie ce risque. Mais on peut noter que dans cette architecture SDN, AODV ne concerne que le trafic du plan de contrôle qui est protégé contre l'écoute passive ou active par l'utilisation d'une couche transport sécurisée. Il n'est donc plus possible d'influencer l'écoulement dans le réseau, même en cas de corruption du routage AODV.

Afin d'étudier les attaques externes et de comparer les deux architectures, nous avons soumis un noeud aux divers types d'attaques identifiés précédemment, une fois en AODV et une fois avec l'architecture proposée :

— *ARP cache poisoning* pour l'attaque de l'homme du milieu ;

— *port scanning* pour la recherche de failles et l'empreinte numérique ;

— *ICMP redirect attack* une autre attaque de l'homme du milieu ;

— *smurf attack* est une attaque en déni de service distribué ;

— *SYN flood attack* est une attaque en déni de service classique.

Les résultats de ces attaques sont résumés dans la table 4.2.

Attaque	AODV	SDN/AODV
Écoute passive	Vulnérable	Vulnérable
ARP cache poisoning	Vulnérable	Non vulnérable[1]
Port scan (avec et sans usurpation)	Vulnérable	Non vulnérable
Empreinte numérique	Vulnérable	Non vulnérable
Inondation SYN	Vulnérable	Non vulnérable[2]
Usurpation MAC ou IP	Vulnérable	Vulnérable[3]

[1] Ceci ne prémunit pas contre les attaques de l'homme du milieu au niveau applicatif.
[2] Les paquets SYN sont ignorés par la victime et le but de l'attaque n'est pas atteint, mais les performances réseau sont, bien entendu, gravement dégradées.
[3] Le commutateur SDN n'est pas capable de détecter une injection de paquets avec usurpation d'identité (adresses MAC et IP) sur un flux pré-existant. Ce point est abordé dans la section ci-après.

TABLE 4.2 – Vulnérabilité des architectures AODV et SDN/AODV face aux attaques visant le réseau

Attaques de l'homme du milieu

Nous avons ici simulé une attaque de type ARP cache poisoning. Il existe d'autres attaques, mais celle-ci est extrêmement simple à mettre en œuvre. Il suffit pour cela d'émettre une trame ARP qui associe à une adresse IP existante l'adresse Ethernet de l'attaquant. La machine victime transmet alors les paquets à destination de cette adresse IP dans une trame avec l'adresse Ethernet de l'attaquant qui peut alors choisir soit de la retransmettre à une autre machine, soit de l'altérer, soit de la détruire.

Dans le cas d'AODV, aucun mécanisme ne permet de s'en défendre et une telle attaque fonctionne.

Dans le cas de l'architecture SDN/AODV, la séparation des plans de contrôle et de données nous impose de différencier les deux cas :

— Dans le plan de données, chaque transmission est traitée par le commutateur SDN et ARP n'est pas utilisé. Par ailleurs, une réponse ARP émanant d'un nœud extérieur serait de toute façon filtrée comme dans les autres cas.

— Dans le plan de contrôle, chaque machine est dans une configuration très proche d'un AODV classique. Toutefois, puisque les seuls interlocuteurs sur ce plan sont les voisins AODV, l'utilisation d'une version sécurisée de ce protocole comme SUAP [91], nous permet d'authentifier l'adresse MAC lors des annonces AODV et ainsi alimenter le cache ARP pour nous passer d'ARP et filtrer ces trames.

Il est à noter qu'une attaque basée sur un autre protocole qu'AODV se verrait redirigée vers le plan de données, et donc filtrée par le commutateur SDN. Ainsi seules l'écoute passive du plan de données et l'injection de trafic dans un flux existant ont pu être exécutées avec succès.

Attaque par injection

Le risque principal que l'architecture SDN ne couvre pas, hormis l'écoute passive du plan de données, est donc l'injection de trafic. Il s'agit ici pour l'attaquant, d'identifier un flux existant et d'émettre des données avec usurpation des adresses MAC et IP. Par exemple pour inonder le drone de commandes illégitimes, comme cela a été démontré dans d'autres travaux [12].

Il est à noter que l'injection de trafic avec usurpation n'est pas nécessairement transparente, en particulier lorsque la couche transport est orientée connexion comme par exemple avec TCP. Émetteur et récepteur suivent les numéros de séquence de chaque message et les comparent avec la valeur de leur compteur local. L'injection de paquets mène à une désynchronisation entre émetteur et récepteur et par conséquent, à l'émission d'acquittements supplémentaires [92] que l'on pourrait rechercher et identifier. La technique de détection présentée ici couvre en partie cette situation mais de manière différente. Elle ne permettrait pas de retrouver un fonctionnement normal : il faudrait pour cela que l'application termine la connexion en cours et en rétablisse une nouvelle.

L'injection de paquets sur une connexion transport peut aussi avoir pour but de porter atteinte à la connexion elle-même ou à la performance des communications. L'émission d'un simple paquet de fermeture, négociée (FIN) ou non (RST), suffit à fermer une connexion ouverte en une seule transmission [14]. Les mesures spécifiques à mettre en place pour contrer ce type d'attaques ne seront pas étudiées ici.

4.2.2 Cas des attaques internes

Ayant caractérisé l'intérêt de l'architecture SDN/AODV face aux attaques externes dans la section précédente, nous nous intéressons maintenant aux cas d'attaques depuis un nœud du réseau.

La distinction entre attaque interne et attaque externe n'est significative que pour l'architecture cible SDN. En l'absence de SDN rien ne permet de différencier ces deux cas et un système basé sur du routage AODV par exemple sera tout aussi vulnérable.

Notons tout d'abord que, contrairement aux attaques externes, le trafic engendré par une attaque interne ne présente aucune caractéristique spécifique permettant une détection directe. Le nœud respecte les règles imposées par l'architecture, en

émettant une demande au contrôleur pour chaque nouveau flux : le flux émane d'un nœud à la lisière de notre réseau et possède une adresse connue.

Comme évoqué précédemment, les scénarios considérés ici mettent en œuvre un nœud infecté préalablement à la mission, par exemple au travers d'une liaison avec la station sol, ou corrompu a posteriori en exploitant une faille logicielle. Il est alors possible d'ouvrir des sessions à distance, récupérer des données du drone ou au contraire télécharger et exécuter du code malin comme des virus ou des vers à destination des autres drones.

Bien que nous n'ayons pas spécifiquement considéré ce scénario ici, il existe des projets de drones mettant en œuvre une interconnexion avec des calculateurs au sol. Ce cas pourrait permettre à un attaquant de porter atteinte à la mission en passant par ce canal de communication, c'est à dire via une station reliée à Internet.

Les techniques habituellement utilisées dans les réseaux filaires pour parer de telles éventualités se basent sur la mise en place de pare-feu et de sondes de détection ou de protection d'intrusion. Il existe différents types de sondes, selon leur localisation :

— les pare-feu permettent la mise en œuvre d'une politique de sécurité définie par la liste des communications autorisées. Ils impliquent donc une notion de point de passage avec un réseau interne de confiance et un réseau externe duquel nous nous méfions.

— les sondes de détection d'intrusion réseau ou *Network Intrusion Detection System (NIDS)* sont placées à des endroits stratégiques dans le réseau afin de surveiller l'activité et considèrent uniquement des mesures concernant le réseau et son trafic. Elles peuvent se baser sur la détection par signatures en inspectant les paquets par exemple *Deep Packet Inspection (DPI)*.

— les sondes de détection de machine ou *Host Intrusion Detection System (HIDS)* sont placées dans les machines du réseau et surveillent leur activité ainsi que leurs communications. Outre les mesures concernant le réseau, les sondes HIDS ont accès au système qui les héberge et peuvent donc également surveiller ses ressources mémoire (RAM, mémoire de masse), l'utilisation du processeur, les processus, etc.

— les sondes de protection, en plus d'être capables de détecter des intrusions, peuvent mettre en œuvre des mesures de protection du réseau ou de la machine.

Dans le cadre des réseaux SDN, ces mêmes techniques sont utilisées. SDN permet d'ailleurs de réaliser une fonction de pare-feu au niveau d'un commutateur SDN, sans matériel spécifique.

Les derniers développements en SDN mettent en avant sa grande souplesse dans la gestion des flux avec la notion de fonctions réseau virtuelles ou NFV. Cette ar-

chitecture permet d'amener les trafics vers un serveur logiciel sans la nécessité de déployer un serveur physiquement.

Dans le cas d'un réseau de drones, la plupart de ces techniques sont inadéquates. La mobilité rend toute notion de réseau interne et externe inapplicable ; si ce n'est dans le cas de l'interconnexion avec Internet, où la station sol pourrait jouer ce rôle puisque par définition elle n'est pas mobile. De même, le placement d'une sonde dans un réseau mobile n'a pas de sens. Seules les fonctions réseau virtuelles gardent toute leur raison dans un tel réseau, mais elle engendrent nécessairement un va-et-vient supplémentaire, du drone émetteur vers le serveur, puis vers la destination. Ceci implique non seulement un délai supplémentaire et des risques de pertes, mais également une consommation plus importante pour chaque nœud ayant à relayer un paquet. Enfin, le déploiement d'une sonde de type HIDS pour une analyse in situ de l'activité du drone, bien que possible techniquement, se heurte à deux écueils : la consommation d'énergie liée à son activité et la place mémoire nécessaire pour un tel logiciel. Il nous a donc semblé difficile de concilier la mise en œuvre de ces sondes avec les exigences de consommation d'énergie liée à la longévité des batteries du drone.

Nous nous sommes donc tournés vers une autre approche du problème : la connaissance que le contrôleur acquiert sur l'activité des drones peut être une source d'informations pertinentes pour un expert connaissant les communications usuelles des drones durant la mission. En effet, le contrôleur est informé de tout nouveau flux créé par chaque drone. Or l'activité d'un drone lors d'attaques est suffisamment caractéristique pour chercher à identifier de tels comportements anormaux. Nous proposons donc d'entraîner un modèle d'apprentissage automatique, basé sur l'observation des événements collectés par le contrôleur SDN comme nous le détaillerons par la suite.

4.2.3 Protection face à l'injection

Détection Nous nous intéressons ici à la détection et la protection contre des injections continues de paquets sur un flux existant.

La mise en œuvre de SDN avec OpenFlow exige que les commutateurs maintiennent des compteurs de paquets et d'octets pour différentes mesures : tables de flux, entrée de ces tables, ports, etc. Ces compteurs peuvent nous permettre de déterminer s'il y a eu injection de trafic et la localisation de cette attaque. Nous donnons ici l'expression mathématique de l'invariant correspondant.

Soit l'ensemble V des drones dans l'essaim et E l'ensemble des arcs orientés représentant la visibilité radio actuelle entre les drones (le voisinage), telle que donnée

par la réception d'un paquet Hello AODV, nous avons :

$$E = \{(x, y) \in V^2\}$$

$G(V, E)$ est donc le graphe orienté des voisinage au sein de l'essaim.

Dans le cas d'une injection de trafic avec usurpation des adresses, l'attaquant utilise un flux pré-existant. Soit $f \in \mathbb{N}^n$ le flux en question, défini par les n attributs utilisés par le contrôleur pour l'identifier de manière unique (adresses, protocoles, ports). En acceptant ce flux, le contrôleur a établi et maintient donc un chemin en fonction de la topologie actuelle G, entre la source a et le nœud de destination b. Soit P ce chemin, tel que :

$$\begin{cases} P = \mu[a, b] = (u_0, \dots u_l), \forall i, u_i \in V, \\ u_0 = a, u_l = b, \\ \forall i \in [1..l], (u_{i-1}, u_i) \in E \end{cases}$$

Lorsqu'un paquet est reçu et correspond à un flux, en plus d'être traité en suivant les directives SDN, un compteur de paquet et un compteur d'octets sont incrémentés. En conséquence, chaque nœud sur le chemin P possède un tel compteur pour le flux f.

Soit $c_i(t) \in \mathbb{N}$ le compteur de paquets ou d'octets reçus par le nœud u_i du chemin P à l'instant t. Le temps de parcours (traitement, retransmission et propagation) n'étant pas nul, un nœud en aval ne peut recevoir et incrémenter le compteur avant que le nœud amont ne l'ait transmis. Nous avons donc :

$$\forall i, j \in [0..l], i < j \Rightarrow c_i(t) \geq c_j(t) \tag{4.1}$$

Cet invariant requiert que les compteurs de tous les nœuds d'un flux soient lus au même temps t et ensuite centralisés vers le contrôleur afin qu'il fasse la vérification. Ceci n'implique pas nécessairement une synchronisation horaire de l'ensemble des nœuds, mais seulement que la lecture soit effectuée de manière synchrone.

On peut exprimer un variant équivalent qui permet de se passer de cette contrainte de synchronisation des lectures en prenant en compte la nature croissante monotone des compteurs de paquets ou d'octets. Nous avons alors :

$$\forall \Delta t > 0, c_i(t + \Delta t) \geq c_i(t) \tag{4.2}$$

Soit t_i le temps auquel la mesure $c_i(t)$ a été faite et t_j, celui pour la mesure au

niveau du nœud c_j. De (4.1) et (4.2) on obtient :

$$\begin{cases} \forall t_i, t_j \in \mathbb{R}, t_i \geq t_j, \\ \forall i, j \in [0..l], \\ i < j \Rightarrow c_i(t_i) \geq c_j(t_j) \end{cases}$$

On peut exprimer cela ainsi : à l'instant t_i, pour tout nœud c_i, son compteur sera toujours supérieur à ceux obtenus précédemment sur les nœuds en aval. Lorsque le contrôleur reçoit une lecture du compteur, il est alors aisé de vérifier cet invariant pour déterminer s'il y a eu injection de trafic et où elle a eu lieu. Bien que la mise en œuvre de cet invariant paraisse simple, deux facteurs limitent toutefois sa mise en œuvre.

Tout d'abord, l'exigence de lecture simultanée dans 4.1 a été remplacée par une exigence de synchronisation horaire pour permettre la comparaison des dates de lecture dans 4.2. Cette synchronisation est possible compte tenu de la relative courte durée d'une mission à condition d'avoir une source horaire avec une dérive limitée, ou au travers d'une synchronisation avec les signaux des systèmes de navigation par satellite GNSS.

Ensuite, la perte de paquets diminue la précision de cet invariant et la perte de paquets est un évènement probable dans un réseau sans fil. De plus, la mobilité engendre des changements topologiques qui se traduisent par la mise à jour des entrées dans les tables de flux SDN. En conséquence, la mobilité engendre des remises à zéro des compteurs, empêchant toute comparaison. C'est pourquoi la comparaison ne peut être effectuée de manière fiable seulement à l'entrée et à la sortie du flux, là où les entrées ne devraient pas être modifiées et où, en conséquence, les compteurs sont effectivement croissant monotone.

La lecture des compteurs peut se faire dans OpenFlow avec la primitive Multipart : la réponse contient l'ensemble des compteurs demandés dans la requête. Ce faisant, le contrôleur peut obtenir de manière régulière la valeur des compteurs à surveiller, tout en s'assurant que le nœud est bien présent. Par ailleurs, cette activité en continue permet de maintenir la route AODV active, ce qui évite d'avoir recours à un mécanisme de maintien de vie. En alternative à ce mode sur requête, une solution asynchrone peut être envisagée sur la base d'un chronomètre d'émission automatique depuis chaque drone.

Dans les deux cas, la lecture et l'émission de ces compteurs sont consommateurs de ressources. Un juste équilibre entre la précision du mécanisme de détection et la durée de vie de la batterie devra être trouvé. En effet, augmenter la fréquence d'interrogation accroît la consommation de ressources de ce mécanisme, tout en diminuant le temps de détection.

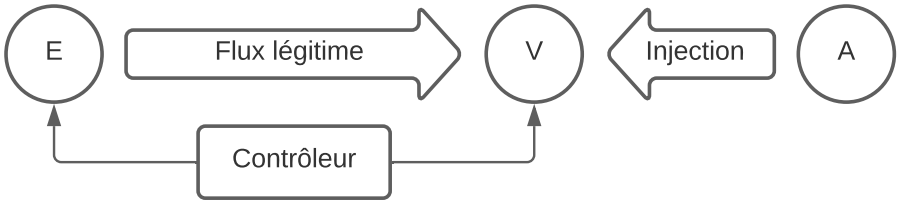


FIGURE 4.1 – Scénario de test d'injection de trafic

Il est à noter que le format du message de réponse MultipartReply est très modulaire et souple, et par là même plus verbeux et donc moins efficace qu'un message au format adapté. Par exemple, l'émission d'un compteur pour un seul flux d'un commutateur peut nécessiter 242 octets. Un format bien plus spécifique ne prendrait qu'une centaine d'octets pour décrire un flux et transporter les compteurs : 82 octets sont dûs aux entêtes de protocole, chaque flux nécessite environ une vingtaine d'octets de données.

Performances Dans le but de mettre en évidence la performance d'une détection d'injection par vérification d'invariant, nous avons mis en place une plateforme de test avec deux nœuds, un contrôleur et un attaquant. L'émetteur E établit un flux UDP vers la victime V et l'attaquant A injecte des paquets sur ce flux préalablement établi en usurpant l'adresse de la source, comme représenté dans la figure 4.1. Le trafic ajouté est équivalent en volume au trafic légitime.

Nous avons pour cela implémenté l'émission automatique de l'état des compteurs à raison d'une transmission toutes les secondes, puis implémenté la vérification de l'invariant au niveau du contrôleur. Comme contre-mesure, le contrôleur émet de nouvelles entrées dans la table de flux des drones en cas d'injection détectée de trafic, afin de modifier à la volée les numéros de ports source et destination des paquets du flux. Il en résulte que les paquets injectés se retrouvent filtrés par le nœud victime, puisqu'ils ne correspondent plus aux caractéristiques du flux attendu. Les courbes du nombre de paquets reçus par seconde avant et après filtrage sont données dans la figure 4.2. La courbe en tirets représente le nombre de paquets reçus par seconde au niveau de l'interface réseau de la victime ; celle en ligne continue est la courbe du nombre de paquets reçus par seconde après filtrage.

Le scénario de cette attaque est le suivant :

— à t=40s, le nœud émetteur débute la transmission d'environ 6 paquets par seconde, simulant ainsi une application de contrôle et de commande du drone, l'attaquant est en écoute mais reste muet ;

— à t=47s, l'attaquant émet à son tour un volume de données équivalent pour

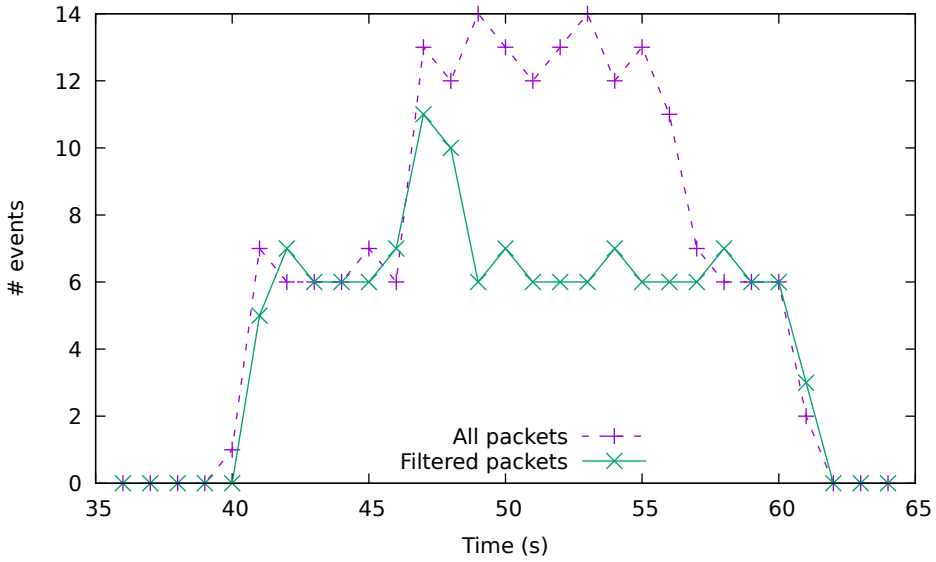


FIGURE 4.2 – Comparaison entre le nombre de paquets émis et reçus lors d'une injection de trafic

perturber ou prendre le contrôle du drone ;

— à t=49s, l'attaquant continue à émettre comme le montre la courbe en tirets qui ne descend pas en dessous de douze paquets par seconde ; mais le contrôleur a déjà détecté et mis en œuvre la contre-mesure. Le nœud victime filtre donc maintenant les paquets de l'attaquant, comme le montre la ligne en continue qui redescend au niveau qu'elle avait avant l'attaque ;

— à t=57, l'attaquant arrête d'injecter du trafic, et à t=62 le scénario se termine.

Le résultat obtenu permet de valider la détection et le filtrage des paquets injectés. L'émission des mesures de compteurs étant configurée à une par seconde, sans que les émissions de la source et de l'émetteur ne soient synchronisées, et le délai entre paquet étant de l'ordre de 150ms, le délai de détection de deux secondes est cohérent. Le trafic induit par l'émission des mesures est de l'ordre de cinq cent octets par seconde pour deux nœuds.

La contre-mesure mise en œuvre ici, qui consiste à modifier à la volée les numéros de ports des paquets du flux uniquement lorsque l'injection est détectée, reste efficace jusqu'à ce que l'attaquant soit capable de s'adapter dynamiquement aux nouvelles valeurs choisies aléatoirement. Dans le contexte d'un réseau sans fil, cette hypothèse est réaliste : une injection à l'aveugle par rejeu de tout trafic sur la fréquence est possible mais son effet dépend de la sensibilité à ce type d'attaque de l'application considérée. Sinon il faudrait à l'attaquant être capable d'identifier l'application visée par un autre moyen que les numéros de port. Toutefois, le simple fait de détecter l'injection et d'en informer l'opérateur humain constitue un progrès significatif.

Cette technique pourrait être poussée plus avant, à l'image des méthodes par

évasion de fréquence dans les communications radios, en changeant régulièrement les numéros de ports source et destination de tout ou partie des flux sur le réseau. Cette technique d'obfuscation, puisque le même protocole applicatif se retrouve disséminé sur plusieurs flux, pourrait rendre les attaques par injection de trafic plus complexes à mettre en œuvre et moins efficaces. Toutefois, elle nécessiterait soit un processus local ad hoc, soit un nombre plus important de transmissions sur le plan de contrôle afin de mettre à jour les tables de flux, au détriment de la longévité de la batterie du drone. D'autres pistes, comme la stéganographie pour transporter des données sensibles pourraient être explorées [93] [94], mais elles sont hors du champ de cette thèse.

On notera enfin que l'attaque simulée ici n'aurait sans doute pas été suffisante pour prendre le contrôle du drone qui aurait nécessité d'émettre plus de commandes frauduleuses pour prendre le pas sur les ordres légitimes. Elle semble donc plus représentative d'une attaque par rejeu. Si le flux injecté était plus important, la détection n'en serait que plus évidente. Le délai de détection ne serait toutefois pas réduit, car il dépend entièrement de la fréquence d'émission des mesures de compteurs.

4.3 Détection d'attaques au sein du réseau

Comme indiqué dans la section précédente, de par son rôle central dans une architecture SDN, le contrôleur capte les signaux de l'activité sur le réseau et il parait pertinent d'exploiter ces signaux afin d'identifier des comportements agressifs. Cette détection devra être la plus rapide possible afin de protéger les drones et la mission.

Contrairement à une approche par inspection de paquets, le contrôleur n'a accès qu'au premier paquet, car ce dernier lui est transmis. Il n'est donc pas question de rechercher des signatures sur les données, mais plutôt de considérer les différents évènements PacketIn d'Openflow dans leurs caractéristiques réseau (adresses, protocoles, ports, etc) et dans leur temporalité.

Cette détection en utilisant des méthodes d'apprentissage automatique est un problème classique que l'on peut aborder de deux manières :

— Soit en considérant le problème comme une détection d'anomalie, où l'algorithme modélise le comportement normal des applications du réseau au fur et à mesure, identifiant alors tout écart significatif comme une anomalie. Cette approche met en œuvre des modèles non supervisés (on parle de unsupervised learning).

— Soit en cherchant à identifier des comportement connus comme étant ceux d'une attaque en soumettant un jeu de données d'entraînement à l'algorithme,

le modèle ainsi créé permettant par la suite une classification par ressemblance aux données apprises. Cette approche met en œuvre des modèles supervisés (on parle alors de supervised learning).

Il existe différents algorithmes concernant l'apprentissage supervisé. L'entraînement se déroulant hors ligne, c'est à dire avant la mission, le temps et les ressources pour cette étape ne sont pas bornés. La phase de détection est rapide en comparaison de l'apprentissage.

Les méthodes non supervisées nécessitent quant à elles la construction d'un modèle au fur et à mesure des échantillons. Malgré tout, il existe des algorithmes de classification qui permettent un calcul en temps réel [95] sur un flux de données.

L'apprentissage non supervisé présente plusieurs aspects intéressants dans le cadre de notre étude. D'une part, il suppose que le modèle est capable d'apprendre le comportement normal des drones durant la mission. Nous parlons ici de comportement sur le réseau : la fréquence des échanges, leur topologie avec éventuellement des nœuds centraux, des nœuds plus actifs que d'autres... Il n'est donc pas nécessaire de connaitre le type de mission à l'avance. D'autre part, cette méthode est supposée détecter les écarts par rapport à cette normale, qui seront alors déclarés comme des anomalies. On comprend par là que cette approche est capable de s'adapter à toutes attaques, même celles que nous ne connaissons pas encore. Mais il s'ensuit également un risque pour qu'un comportement normal mais rare soit classifié comme une anomalie.

À l'opposé, l'apprentissage supervisé consiste à classifier des comportements préalablement connus. Bien que l'apprentissage automatique soit basé sur une certaine généralisation à partir des échantillons du jeu de données d'entraînement, il est peu probable qu'un tel modèle détecte un nouveau comportement significativement différent sans devoir entraîner un modèle avec un nouveau jeu de données. De même, il faut que le jeu de données soit le plus large possible pour que l'algorithme puisse apprendre les comportements bénins.

On comprend donc que la plasticité apportée par l'apprentissage non supervisé est aux dépens d'un risque accru de faux positifs, là où la spécialisation de l'apprentissage supervisé a pour conséquence la non détection de nouvelles formes d'attaques. Dans les deux cas, et comme dans tout le domaine de l'apprentissage automatique, la qualité du jeu de données, que ce soit en quantité comme en diversité, est primordial.

Il n'existe pas de jeu de données disponible actuellement sur le trafic engendré sur un réseau d'un essaim de drone. Il n'est donc pas possible de valider et d'évaluer un modèle d'apprentissage automatique sur un jeu de données dans un environnement proche de la réalité.

Par ailleurs, l'apprentissage non supervisé est basé sur l'idée que le trafic normal est significativement plus présent que celui que l'on qualifiera d'anormal. On peut

donc s'attendre à ce que la technique soit moins efficace si le trafic normal n'est pas suffisamment présent, par exemple si l'attaque arrive très tôt après le début de la mission. A ce moment là, le modèle n'aura pas capturé et modélisé le trafic normal. Or, certains types de missions peuvent conduire à une relative rareté des évènements au niveau du contrôleur. Par exemple, un flux vidéo n'engendrera qu'un seul évènement au début de la mission, puis plus aucun. Il en est de même avec les données de commande et contrôle de la mission avec des flux très stables. Cette stabilité fait que les retours sur les comportement sont assez rares : un évènement PacketIn au début du flux seulement, puis plus rien (les événements liés à la mobilité n'étant pas pris en compte ici). Seule la lecture des compteurs de paquet des tables de flux permettraient de compenser ce manque d'information sur le comportement des nœuds. Toutefois, ces données ne peuvent être remontées vers le contrôleur de manière continue et en temps réel, sans peser significativement sur le trafic et sur la durée de vie de la batterie. Par ailleurs, la détection ne peut se faire au moment de la réception d'un nouveau flux, mais seulement lorsque les compteurs auront été mis à jour.

C'est pourquoi, en l'absence d'un jeu de données de qualité représentatif des différentes missions concernant un réseau d'un essaim de drone, et compte tenu des réserves sur son efficacité dans certains scénarios, nous avons décidé de mettre de côté l'apprentissage non supervisé et explorer l'utilisation de méthodes d'apprentissage automatique supervisé.

4.3.1 Jeu de données

Comme indiqué précédemment, le choix du jeu de données est particulièrement important.

Nous avons fait l'hypothèse que l'on pouvait entraîner un modèle sur un jeu de données reposant sur une topologie réseau et qu'il resterait efficace sur une topologie cible différente. En effet, le comportement que nous cherchons à identifier ne dépend que très peu de la topologie d'un point de vue IP, et nous cherchons à identifier l'attaque plus que le trafic normal.

On peut en effet décrire des schémas spécifiques aux attaques visées, l'unicité de la cible étant une caractéristique commune :

— Un parcours de ports part d'un attaquant unique et explore les ports transport de la victime, il y a donc une grande diversité sur le port destination ;

— Une attaque en force brute ou un déni de service cible une application unique ou un nombre limité de ports ouverts et engendre une répétition de tentatives de connexions ;

— Une attaque en déni de service distribué part d'un ensemble de sources, vers un destinataire unique, sur une durée limitée.

Sources et transformation

Un nombre important de travaux liés à la sécurité et à la détection d'attaques se basant sur une signature utilisent le jeu de données proposé en 1999 pour une compétition de découverte de connaissance et d'exploration de données. Celui-ci est communément nommé KDD99 [78] en référence au nom de la compétition en question. Une version améliorée a été proposée pour éliminer certains biais et est désignée sous le nom de NSL-KDD99 [79]). Ils sont issus d'un ensemble de captures TCP/IP et de données supplémentaires sur le fonctionnement et le système des machines. KDD99 propose un ensemble d'échantillons labellisés avec 41 attributs, directement utilisable pour de l'apprentissage automatique. Ce jeu de données et les résultats obtenus par différentes techniques, représentent donc un référentiel permettant la comparaison de nouvelles techniques sur la base d'un jeu de données commun.

Outre le fait qu'il s'agit d'un jeu de données assez ancien dont la représentativité est remise en cause depuis de nombreuses années [96], il contient des attributs purement liés au système (par exemple le nombre d'accès super-utilisateur, nombre d'opérations de création de fichiers, nombre de shell ouverts) ou disponibles uniquement à la fin du flux, par exemple la durée de ce dernier. Ces caractéristiques sont hors de portée d'un commutateur SDN et ne semblent donc pas applicables à notre cas d'étude. L'exigence de détection au plus tôt nécessite de ne prendre en compte que les caractéristiques connues au début d'un flux. C'est pourquoi nous n'avons pas retenu ces jeux de données et nous avons donc cherché une source de données récente qui nous permette de mettre en œuvre un modèle d'apprentissage automatique.

Nous avons ainsi choisi d'utiliser le jeu de données proposé par l'Université du New-Brunswick en collaboration avec l'institut canadien pour la cybersécurité pour des travaux sur les sondes de détection d'intrusion (IDS) datant de 2018 [97]. Outre le jeu de données précalculé qui contient donc des données prêtes à l'emploi, il propose l'ensemble de leurs sources sous la forme de captures en fichiers PCAP. Il regroupe ainsi l'historique des transmissions sur un réseau simulé avec plusieurs scénarios d'attaque.

Ce jeu de données a été créé à partir d'une plateforme de calcul AWS simulant un réseau de cinq sous-réseaux : recherche et développement, direction, secrétariat et production, et un département informatique. Chaque département contient une centaine de machines. Il y a enfin un sous réseau de serveurs. Les machines du département informatique tournent sur un système d'exploitation Linux Ubuntu alors que les autres tournent sous Windows (8.1 et 10 pour les utilisateurs finaux et 2012 ou 2016 pour les serveurs). Le trafic bénin est simulé à partir d'agents

spécifiques exécutés sur chaque machine terminale, alors que les attaques ont été menées depuis un sous-réseau extérieur de 50 machines avec les logiciels appropriés. Les attaques qui ont été menées contre le réseau sont listées ci-dessous :

— infiltration du réseau depuis l'intérieur (portscan) ;

— déni de service HTTP en Denial of Service (DoS) et Distributed Denial of Service (DDoS) ;

— attaque en force brute ;

— injection SQL ;

— botnet (capture d'écran et enregistreur de frappe).

La disponibilité des fichiers de capture nous a permis de construire le jeu de données propre à notre cas d'étude puisqu'il s'est agi d'extraire des paquets capturés l'ensemble des événements reçus par le contrôleur et de calculer des métriques liées à l'activité toujours du point de vue du contrôleur SDN. Nous avons donc tout d'abord extrait de toutes ces captures l'ensemble des informations suivantes :

— début et fin de chaque flux, en considérant un délai d'inactivité de 20 secondes au-delà duquel le flux est automatiquement détruit ;

— certains champs de l'entête IP : adresses source et destination ainsi que protocole ;

— entêtes TCP, UDP ou ICMP : ports, fanions, raisons, etc. ;

— la longueur du premier paquet.

Nous avons donc développé et validé les outils nécessaires pour produire ces données à partir des fichiers PCAP.

Validation Nous avons tout d'abord validé le contenu des captures en comparant leurs données avec la description des attaques attendues. Nous avons globalement retrouvé l'ensemble des attaques prévues. Toutefois, certains écarts y ont été identifiés et ont donc été exclus de la liste des données potentielles.

Par ailleurs, la validation des scénarios d'attaque a permis de collecter les données nécessaires pour labelliser les flux en vue de la phase d'apprentissage.

Le scénario d'infiltration consiste en un mail contenant un fichier vérolé exploitant des vulnérabilités du système permettant à l'attaquant de lancer un script. Cette attaque n'a pas été utilisée telle quelle puisque nous n'avions pas envisagé ce scénario. Mais le script ainsi exécuté contenant des parcours de ports TCP, ces derniers ont été gardés pour notre jeu de données. La liste des scénarios retenus est donnée en table 4.3.

Nous avons également étudié la base de données d'un point de vue qualitatif. Une analyse des types de flux montre qu'un tiers de ceux-ci sont liés à l'application DNS

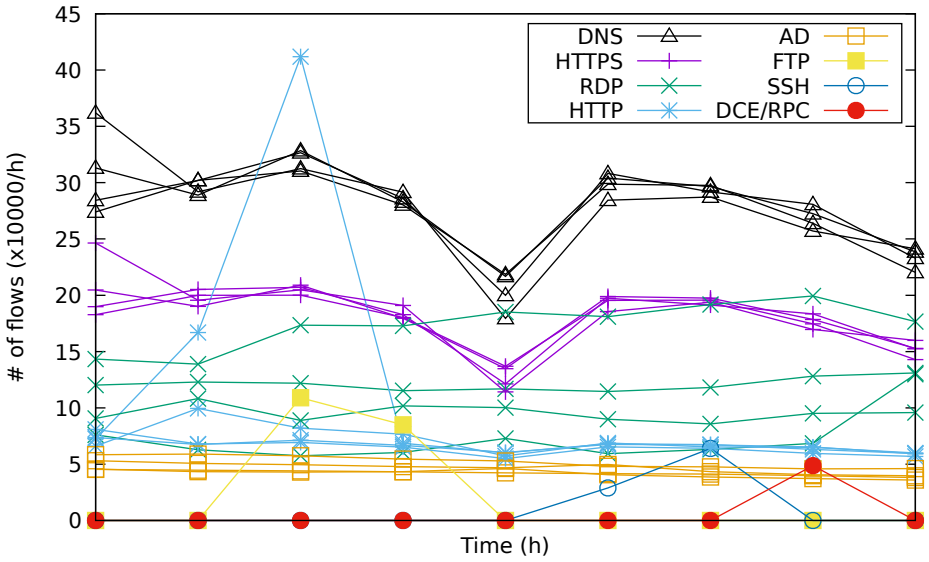


FIGURE 4.3 – Nombre de flux par application dans le jeu de données

Date	Attaque
14/02/18	Force Brute
15/02/18	DoS
20/02/18	DDoS
28/02/18	Scan

TABLE 4.3 – Scénarios sélectionnés pour le jeu de données

mais que cette dernière n'est jamais la cible d'attaques. C'est donc une application d'usage intensif qui représente pourtant un trafic bénin et constitue donc a priori une bonne base pour la phase d'apprentissage. Viennent ensuite des données web HTTPS pour un quart, suivi de Remote Desktop Protocol (RDP) et enfin HTTP. La figure 4.3 montre l'évolution du nombre de flux par application durant les différents scénarios. En dehors d'une attaque en déni de service HTTP clairement visible, les autres attaques ne semblent pas facilement identifiables en terme de nombre de flux induits.

Calcul et sélection des features

A partir de l'ensemble des flux identifiés dans les captures, nous avons envisagé plusieurs mesures issues de la théorie des graphes, ainsi que des statistiques basées sur ces mêmes mesures : degré entrant, sortant et total des sommets du graphe ; fréquence de création de ces flux ainsi que leurs statistiques associées. Toutefois, ces mesures ont sans doute plus de signification sur un réseau plus important où l'on retrouverait des topologies remarquables comme des parties centralisées avec notamment les serveurs comme points centraux. Ces mesures nous ont semblé moins

pertinentes sur de plus petits réseaux de nœuds homogènes, où les attaques qu'elles mettraient en évidence ne sont que peu représentées dans le jeu de données. En effet, en dehors du déni de service distribué, les trois attaques retenues mettent en œuvre des communications directes entre un attaquant et une victime.

Nous avons également considéré le comptage d'évènements anormaux comme le nombre d'erreurs reportées via le protocole ICMP, en particulier l'erreur *destinataire inaccessible - port non joignable*, symptôme lorsqu'il se répète, d'une tentative d'exploration ou de parcours de ports sur une victime. Toutefois, ce symptôme dépend fortement du système d'exploitation et éventuellement du protocole transport utilisé. De plus, ce paquet ne peut arriver qu'une fois le flux accepté et installé par le contrôleur. L'attaquant aura déjà eu le temps d'émettre un nombre important de paquets. Nous avons donc écarté cet indicateur.

Nous avons finalement développé deux mesures liées au comportement d'un nœud : son activité en terme de création de flux avec une vision quantitative et une approche plus qualitative avec la dispersion temporelle des numéros de ports.

Expression de l'activité Le premier attribut qui semble pertinent pour rendre compte du comportement d'un nœud est l'expression de l'activité du point de vue du contrôleur SDN. Pour cela, le contrôleur ne recevra comme témoin de l'activité d'un nœud que le premier paquet d'un flux sous la forme d'un événement OpenFlow PacketIn. Par définition, un flux de données a une durée, des compteurs de paquets et d'octets, etc. Toutefois, nous considérerons ici l'activité comme la mesure du nombre de flux créés et non la quantité de données générées par un nœud du réseau.

Il peut arriver, lorsque la source d'un flux produit beaucoup de paquets dès le début que des événements PacketIn supplémentaires arrivent au niveau du contrôleur. Nous les ignorerons et nous considérerons que seul le premier paquet est soumis à détection. De plus, nous ne comptabilisons pour un t-uplet (adresse source, adresse destination, protocol, port source, port destination) que le premier flux. S'il existe déjà un flux dans l'autre direction, ce dernier est considéré comme la seconde demi connexion ou la réponse à la requête et ne sera donc pas examiné.

Le nombre de flux d'un nœud existant à un instant t n'est que peu représentatif de son activité ; ces flux peuvent avoir été créés bien avant. L'utilisation du degré de ce nœud dans le graphe de notre réseau n'est donc pas pertinent. Seul le nombre d'événements PacketIn par unité de temps nous renseigne sur son activité courante.

Ce calcul est relativement consommateur de ressources. Le comptage d'événements sur une période glissante nous impose d'en conserver l'historique. Bien qu'ayant émis l'hypothèse d'un essaim de drones de taille limitée, ceci peut représenter une quantité importante de données selon la durée de la période. Par ailleurs, cela pourrait constituer une faille de sécurité, la quantité de mémoire utilisée étant

directement liée au nombre de flux générés.

Une autre approche serait d'utiliser des mesures statistiques sur les temps entre événements communément appelé Inter-Arrival Time (IAT) ou bien son inverse, la fréquence d'arrivée des événements. On peut par exemple prendre la moyenne mobile exponentielle ou Exponentially Weighted Moving Average (EWMA) qui présente l'avantage de ne pas nécessiter de conserver l'historique.

Les événements PacketIn sont par définition des événements discrets dont nous noterons t_i les dates d'arrivée au niveau du contrôleur.

$$\overline{\delta t}_n = \alpha \cdot \delta t_{n-1} + (1-\alpha) \cdot \overline{\delta t}_{n-1} = \overline{\delta t}_{n-1} + \alpha \cdot (\delta t_{n-1} - \overline{\delta t}_{n-1})$$

Nous proposons ici une autre mesure qui garde les propriétés de décroissance exponentielle de l'EWMA, mais présente des avantages quant à la représentativité de l'activité. En effet, nous cherchons une fonction qui ne nécessite pas de mémoriser l'historique, et qui permette en même temps de lisser les pics ou les creux par un effet mémoire, tout en permettant l'oubli en cas d'arrêt des événements durant une longue période.

Soit $E = \{t_i \in \mathbb{R}\}$ l'ensemble des dates d'arrivée des événements PacketIn sélectionnés. Nous détaillerons plus loin la notion de sélection et les critères de cette sélection : il s'agit, dans une première approche, d'une caractéristique commune à plusieurs événements PacketIn qui relie ces flux entre eux (par exemple, le même nœud source) et dont nous souhaitons une mesure de l'activité.

Soit $F = \{t_i \in E : t_i \leq t\}$ le sous-ensemble de E des événements antérieurs à l'instant t, et $n = |F|$ le nombre d'éléments de F. L'activité A en fonction du temps t peut alors être exprimée sous la forme :

$$A(t) = \sum_{i=1}^{|F|} \alpha^{t-t_i} \tag{4.3}$$

avec $\alpha \in \mathbb{R}$ et $0 < \alpha < 1$.

Nous ne calculerons $A(t)$ que lorsqu'un événement est reçu, c'est à dire à chaque $t_i \in E$. Nous pouvons alors écrire $A(t)$ à l'instant t_n de la manière suivante :

$$A(t_n) = 1 + \sum_{i=1}^{n-1} \alpha^{t_n - t_i} = 1 + \alpha^{t_n - t_{n-1}} \sum_{i=1}^{n-1} \alpha^{t_{n-1} - t_i}$$

Il vient alors l'expression suivante :

$$S_n = A(t_n) = 1 + \alpha^{t_n - t_{n-1}} A(t_{n-1}) = 1 + \alpha^{\delta t_n} S_{n-1} \tag{4.4}$$

où $\delta t_n = t_n - t_{n-1}$ est donc égal au temps écoulé depuis l'événement précédent (IAT).

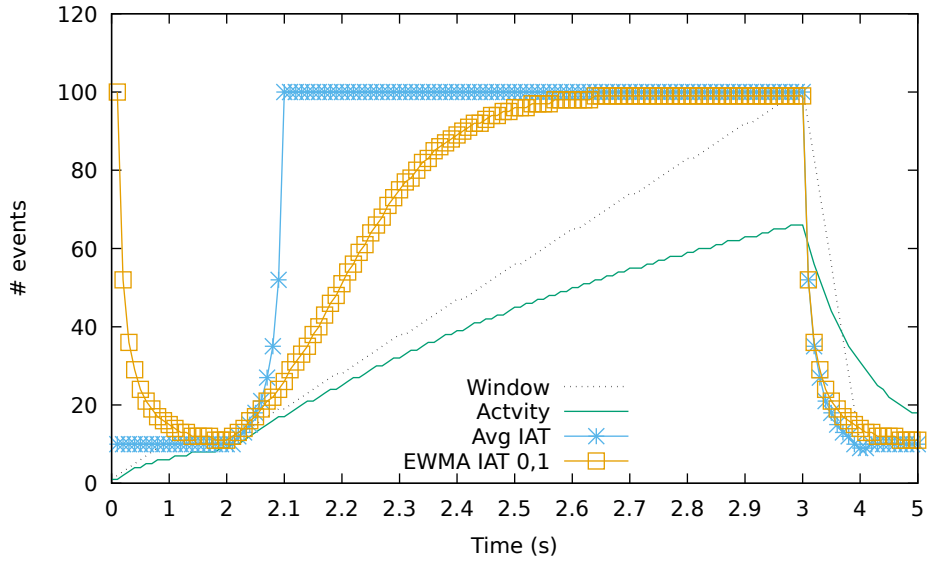


FIGURE 4.4 – Moyenne, moyenne mobile exponentielle et activité

Notons que si $\alpha = 1$ alors $A(t)$ est égal au nombre total d'événements antérieurs à t. S_n doit être comparé à un nombre d'événements par unité de temps, c'est à dire à une fréquence, plutôt qu'à un délai.

La figure 4.4 montre la variation de différentes expressions de l'activité proposées précédemment, dans le scénario suivant : les événements surviennent toutes les dixième de seconde depuis $t = 0$ et pendant deux secondes. Ensuite, les événements arrivent tous les centièmes de seconde durant une seconde, pour revenir à l'écart initial durant deux secondes. La figure montre la valeur pour chaque événement et il en résulte que l'axe des x n'est pas linéaire (il y a plus d'échantillons au milieu).

Les valeurs représentées sont :

— En pointillés, le nombre d'événements durant la seconde précédent l'événement. Il n'y a eu aucun événement précédent $t = 0$. Il n'y a pas d'effet de mémoire.

— La ligne marquée d'étoiles est l'inverse de la moyenne glissante des temps entre événements sur les 10 derniers événements.

— La ligne marquée de carrés est l'inverse de la moyenne mobile exponentielle, avec $\alpha = 0.1$, avec une valeur nulle au démarrage (d'où la valeur élevée à gauche sur le graphique).

— La ligne continue sans marque représente l'activité telle qu'exprimée dans 4.4 avec $\alpha = 0.366$.

On constate un effet mémoire assez important sur la courbe de l'activité. Cet effet mémoire peut également être obtenu et contrôlé par la taille de la fenêtre mobile dans le cas du calcul du nombre d'événements, ou par le choix de la valeur de α dans le calcul de l'EWMA. Outre les difficultés liées au choix de cette valeur,

Δ T	2Hz	10Hz	100Hz	1000Hz
1s	0.25	0.348678	0.366032	0.367695
10s	0.9025	0.904382	0.904792	0.904833
100s	0.990025	0.990045	0.990049	0.990050

TABLE 4.4 – α pour différentes valeurs des paramètres

aucune moyenne statistique n'est toutefois satisfaisante. Seule l'équation 4.4 fournit un comportement satisfaisant bien que sa croissance soit asymptotique et donc plus lente :

— Le calcul par fenêtre glissante est consommateur de mémoire.

— Le calcul de l'EWMA sur la fréquence instantanée entre deux événements, c'est à dire l'inverse de l'IAT, pose le problème de deux événements distants dans le temps. Dans ce cas en effet, la valeur négligeable de la fréquence instantanée ne fera pratiquement pas varier le résultat si α est petit et il n'y aura pas d'effet mémoire si α est proche de 1.

— Le calcul de l'EWMA sur l'IAT donne l'effet inverse : lorsque la valeur courante est proche de 0, l'EWMA aura une croissance très lente si α est trop proche de 0, mais une valeur proche de 1 diminue également l'effet mémoire.

Afin de calculer le paramètre α de l'activité, nous avons préféré une méthode pragmatique à une approche basée sur la demi-vie. Nous avons calculé la valeur de α pour laquelle l'activité S_n tend vers la valeur exacte du nombre d'événements par unité de temps ΔT pour une fréquence f donnée, c'est à dire à IAT constant.

Ainsi, pour une fréquence f sur une période de ΔT, nous avons $\Delta T \times f$ événements espacés de $\delta t = \frac{1}{f}$ et nous cherchons α pour lequel :

$$\begin{cases} S_n &= 1 + \alpha^{\frac{1}{f}} S_n \\ S_n &= \Delta T \times f \end{cases}$$

D'où nous déduisons que :

$$S_n = \frac{1}{1 - \alpha^{\frac{1}{f}}} \tag{4.5}$$

Et :

$$\alpha(f, \Delta T) = \left(1 - \frac{1}{f \times \Delta T}\right)^f \tag{4.6}$$

Tout d'abord, nous remarquons que α n'est pas constant en fonction de la fréquence. Toutefois, nous ne cherchons pas ici à obtenir une valeur juste quelle que soit la fréquence des événements. La table 4.4 fournit des valeurs dans différents cas. Une analyse empirique de l'erreur montre que celle-ci est relativement limitée : l'erreur

	L=226		L=1334	
B (bps)	Eq. 4.4	Eq. 4.7	Eq. 4.4	Eq. 4.7
10^4	1.97x10^1	3.57	3.78x10^0	4.04
10^5	1.92x10^2	3.48	3.30x10^1	3.53
10^6	1.92x10^3	3.48	3.26x10^2	3.48
10^7	1.92x10^4	3.48	3.25x10^3	3.48
10^8	1.92x10^5	3.48	3.25x10^4	3.48

TABLE 4.5 – Activité maximale et activité corrigée pour différentes valeurs du débit et de la longueur du paquet pour $\alpha = 0.75$

pour $\alpha(1000Hz, 10s)$ est plus élevée pour les événements de basse fréquence, d'environ 5% pour $f = 1Hz$ ($M_n(t_n) = 10.5$) et de moins de 1% pour toute fréquence supérieure à 4Hz. On constate par ailleurs que c'est ΔT qui est prépondérant dans la valeur de α.

On peut remarquer que la valeur de l'activité n'est pas indépendante du réseau considéré, ni même du type de trafic émis. En effet, le débit du canal de transmission ainsi que la longueur des paquets émis influera directement sur la valeur obtenue de l'activité. En fait, celle-ci est nécessairement limitée par le nombre maximum d'événement PacketIn qu'il est possible de générer, et par conséquent au débit de la transmission et à la longueur du paquet

Si B est le débit du réseau et L la longueur de chaque paquet, alors la fréquence maximale f_M des événements sur le réseau est de : $f_M = \frac{B}{L}$. En utilisant cette expression en lieu et place de la fréquence f dans 4.5, il en résulte que pour des longueurs de transmission différentes ou pour des débits différents, l'activité maximale résultante sera également différente.

Cette caractéristique rend caduque la capacité du modèle à généraliser entre le jeu de données d'entraînement, créé à partir d'un réseau filaire, et une situation réelle utilisant un canal de transmission au débit plus faible de plusieurs ordres de grandeur. Nous avons donc étudié la variation de la valeur du maximum de l'activité pour différents débits et longueurs de paquets. La table 4.5 montre les valeurs obtenues pour deux longueurs de paquets et différents débits. On constate que l'évolution du maximum d'activité est proportionnelle au débit et inversement proportionnelle à la longueur.

Pour des valeurs de B et de L réalistes pour un réseau IP, il apparaît que la multiplication de S_n par $K = \frac{L}{B}$ donne des valeurs comparables, voir très proches lorsque la longueur du paquet est inférieur de plusieurs ordres de grandeurs au débit du canal.

Ainsi l'équation (4.4) se transforme en :

$$S_n = \frac{L_i}{B} + \alpha^{\delta t} \cdot S_{n-1} \tag{4.7}$$

Variations du numéro de port Une fois exprimée l'activité d'un nœud, il nous semblait important de caractériser l'usage qu'un utilisateur fait du réseau. Étant donné la corrélation existant entre le numéro de port et l'application considérée, il semble évident d'utiliser cette information pour évaluer la versatilité d'un hôte comme témoin d'un comportement anormal. Notons que cette dernière peut s'appliquer au port source comme au port de destination, mais qu'une versatilité élevée pour chacune de ces deux valeurs ne relève pas d'un même comportement. Un parcours de port donnera nécessairement une grande variation du port de destination mais pourrait être réalisé avec un port source constant, alors qu'une attaque en force brute sur un mot de passe impliquera une constance du port de destination mais nécessitera normalement un port source changeant.

La dispersion d'une série est généralement exprimée par l'écart type de sa distribution. Au delà de la problématique de mémorisation desdites distributions, qui peut être résolue par l'utilisation d'algorithmes spécifiques, le calcul de la dispersion doit se faire sur une fenêtre glissante, au même titre que l'activité. En effet, dans le cas contraire, la détection d'un changement de comportement après une longue période de calme en serait négativement affectée.

L'utilisation de la connaissance du graphe par le contrôleur SDN est potentiellement une solution, en calculant la différence entre le minimum et le maximum des flux récents. Toutefois, ceci nécessiterait un parcours des différents flux, ce qui est consommateur en ressources et inefficace s'il y a beaucoup de flux anciens.

Nous avons donc préféré calculer deux valeurs M et m comme étant le maximum (respectivement le minimum) entre le port considéré du PacketIn (source ou destination) et la valeur d'une fonction dépendante du temps et de la valeur précédente.

Cette fonction a pour but de fournir un effet mémoire en maintenant artificiellement la valeur courante durant une période ΔT puis en laissant décroître linéairement jusqu'à 0 pour le maximum, et à l'inverse, de croître jusqu'à un maximum pour le minimum. Nous utilisons la même fonction mais avec un coefficient k de signe opposé : négatif pour M et positif pour m.

Soient T_m la date de l'événement ayant conduit à la mise à jour du minimum m et T_M son équivalent pour le maximum M. Soit g la fonction sur $\mathbb{R}^2$, de paramètres k et ΔT telle que :

$$g(p, \delta t) = \begin{cases} p & \forall 0 \leq \delta t \leq \Delta T \\ p + k\ (\delta t - \Delta T) & \forall \delta t > \Delta T \end{cases} \tag{4.8}$$

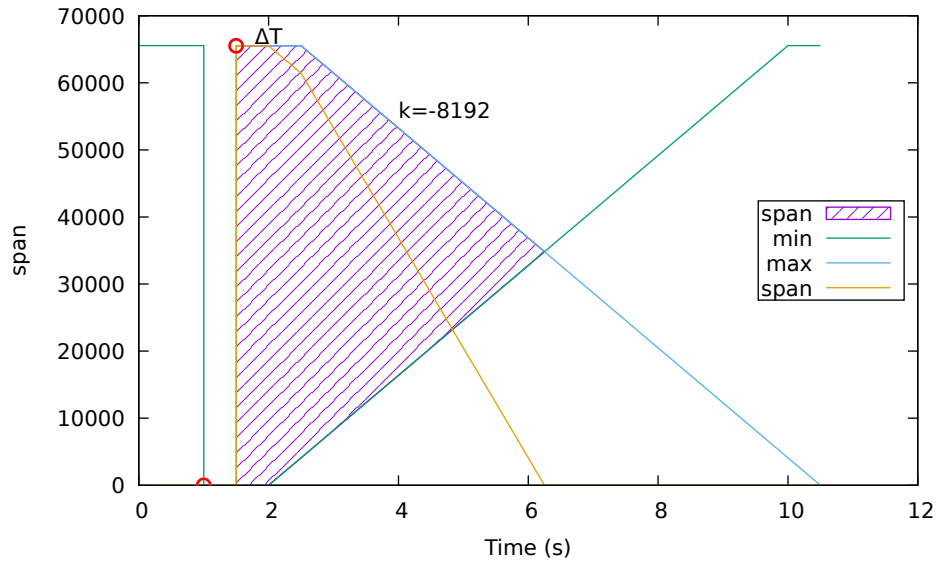


FIGURE 4.5 – Variation du numéro de port

Si le port courant est inférieur à $g(m, t - t_m)$, alors m et T_m sont remis à jour. De même, si le port courant est supérieur à $g(M, t - t_M)$, alors M et T_M sont remis à jour. La valeur retenue pour rendre compte de la dispersion en fonction du temps est alors :

$$S(t) = min(0, g(M, t - t_M) - g(m, t - t_m))$$

Cette expression nécessite uniquement la mémorisation de la dernière valeur mise à jour ainsi que la date de cette dernière, et s'obtient sans parcours de graphe.

Prenons un exemple pour illustrer la variation de cette caractéristique au cours du temps. La figure 4.5 représente les valeurs calculées pour le scénario suivant : un *PacketIn* est reçu à $t = 1$ avec un numéro de port de 0, un second flux est créé à $t = 1.5$ sur le port 65535. Ces deux événements sont représentés par un cercle. Les courbes représentées sont le minimum et le maximum comme calculé à partir de l'équation 4.8. L'écart représente donc la hauteur de la partie hachurée, sa valeur étant également représentée sous la forme d'une courbe.

À partir de $t = 1$, le minimum et le maximum sont à 0 et l'écart (*span*) est nul. À $t = 1.5$, le maximum est maintenant de 65535 et l'écart a la même valeur. Durant une période $\Delta T = 1s$, le min et le max ne varient par. À $t = 2$, l'écart diminue à cause du minimum qui croit au rythme de k. Enfin, à partir de $t = 2.5$, le maximum diminue à son tour et l'écart diminue donc en $2k$ jusqu'à atteindre 0 après $t = 6.25s$.

Calcul des attributs des échantillons Notons que ces deux métriques sont définies de manière générique (l'activité d'un nœud) là où la détection des attaques se base principalement sur l'identification d'un attaquant et d'une victime (l'activité de l'attaquant vers sa victime). Nous avons donc appliqué les formules précédentes,

non pas uniquement sur chaque nœud du réseau (activité en tant que source, activité en tant que destinataire), mais également sur des groupes définis par plusieurs critères. Ces critères sont issus des champs présents dans l'événement PacketIn et correspondent aux entêtes protocolaires :

— adresses source et destination IP ;

— champ protocole IP, qui différencie notamment entre flux TCP et UDP ;

— ports source et destination TCP ou UDP ;

— champs type et code pour ICMP.

Nous avons ainsi formé différents groupes à partir de ces critères de sélection et calculé les attributs correspondants sur ceux-ci à partir des expressions mathématiques. En effet, les critères de sélection influent sur le choix des événements PacketIn et donc sur les temps d'arrivée t_i correspondants.

Il est à noter que le nombre de critères a un fort impact sur la quantité de mémoire nécessaire pour le calcul d'un attribut et conséquemment peut influer sur la performance du contrôleur. En effet, le contrôleur doit retrouver la valeur et la date d'arrivée précédentes pour le groupe considéré. Plus le nombre de critères est important, plus y aura de groupes et de valeurs à mémoriser.

Le calcul des métriques étant réalisé dans le contrôleur, le risque de saturation de mémoire est moins prégnant que sur un drone. Il peut être traité, par exemple par un récupérateur d'espace mémoire dont le critère sera lié à l'âge du dernier événement du groupe. Ce délai d'expiration peut être calculé directement à partir de l'équation 4.8 en prenant :

$$\delta t_{max} = \frac{65536}{|k|} + \Delta T$$

Quant au délai maximal pour l'activité, il peut être basé sur la valeurs de ΔT dans l'expression 4.6, ou calculé à partir d'un seuil ϵ :

$$\alpha^{\delta t_{max}} < \epsilon \; ou \; \delta t_{max} = log_{\alpha}(\epsilon)$$

4.3.2 Choix du modèle et des métriques de performance

Comme exposé précédemment, la détection des comportements agressifs sera réalisée au niveau du contrôleur, sur réception d'évènements PacketIn. Nous avons fait l'hypothèse que le contrôleur serait hébergé sur la station sol car nous nous libérons ainsi de contraintes matérielles en terme de puissance de calcul et de mémoire, par comparaison à une plateforme embarquée. Toutefois, nous ne souhaitions pas exclure cette possibilité et l'algorithme de détection ne devait pas nécessiter de calculs

Jeu de données	Bénin	Force Brute	DoS	DDoS	Scan
Binaire	50%	12.5%	12.5%	12.5%	12.5%
Multiclasses	20%	20%	20%	20%	20%

TABLE 4.6 – Répartition des classes dans le jeu de données d'entraînement

importants et rester réactif y compris sur une machine avec une puissance de calcul limitée.

Bien que les deux métriques exposées précédemment soient des valeurs numériques, certains champs protocolaires sont purement symboliques, notamment les fanions TCP, les adresses IP, les numéros de port, etc. L'algorithme d'apprentissage automatique doit donc être capable de gérer ces deux types d'attributs. Par ailleurs, la détection d'un comportement agressif devrait être reportée à l'opérateur humain et il serait souhaitable que le modèle soit interprétable afin de fournir des informations circonstancielles sur ce comportement. C'est pour cette raison que nous avons écarté les réseaux de neurones qui ne permettent pas une interprétation aisée.

Nous nous somme finalement tournés vers l'algorithme dit "Random Forest Classifier" [98]. En effet, il montre généralement de bonnes performances sur des classifications d'anomalies liées à la sécurité [99] [77], il répond à nos contraintes (interprétation et capacité à utiliser des attributs symboliques), et ne nécessite pas de normalisation des attributs.

Le jeu de données initial extrait des captures UNB IDS contient plusieurs centaines de milliers d'événements PacketIn. Nous avons donc étudié la courbe d'apprentissage du modèle pour estimer à quel moment celui-ci devient suffisamment précis, tout en évitant le sur-apprentissage. Au delà de 10000 échantillons, la courbe montre les signes d'un sur-apprentissage. En dessous, le jeu de données n'a pas assez de diversité et le modèle reste peu performant.

Nous avons également créé deux étiquetages différents. Le premier est binaire (bénin ou attaque), le second contient plusieurs classes (bénin et détail de chaque type d'attaque). Nous souhaitions en effet évaluer la capacité du modèle à distinguer les différentes attaques et ainsi envisager des contre-mesures différenciées.

Lors de la création des jeux de données, les ratios d'échantillons ont été contrôlés pour chaque classe. La répartition des différentes classes est donnée dans la table 4.6. Elle garantie une quantité égale pour chaque classe et dans le cas du jeu de données binaire, un nombre égal d'échantillon de chaque attaque.

La phase d'entraînement du modèle a été réalisée avec 10000 échantillons, dont 20% étaient mis de côtés pour la validation.

Nous avons pu vérifier que le modèle était sensible à la répartition des différentes classes dans le jeu de données, en augmentant le nombre d'échantillons bénins. Nous

avons pu constater une diminution du taux de faux positifs.

Nos critères de performance correspondent à ceux habituellement choisis pour ce types d'utilisation [77] : précision, taux de rappel et score F1. Le taux de faux positifs doit rester le plus bas possible. Nous utiliserons pour cela le score F1 qui combine taux de rappel (*recall*) et précision de la manière suivante :

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Mais au delà de ce score, nous souhaitons également évaluer le délai avant détection, c'est-à-dire le nombre d'échantillons nécessaires avant que le modèle n'identifie l'agression. Ce temps de réaction est primordial pour pouvoir mettre en œuvre une contre-mesure et ainsi éviter un effondrement des performances du réseau.

4.4 Performances

Nous avons entraîné deux modèles à partir des choix explicités précédemment, et avec les attributs les plus significatifs : un modèle de classification binaire (attaque vs. bénin) et un modèle multiclasse dont le but est de différencier et donc d'identifier le type d'attaque en cours.

4.4.1 Sélection des attributs du modèle

Après avoir généré les différents attributs du modèle, nous nous sommes attachés à en réduire le nombre. Cette réduction a pour objectif de limiter la complexité du modèle, et par conséquent, de limiter le durée de calcul nécessaire pour une prédiction. Ceci permet également de limiter les risques de sur-apprentissage.

La matrice de covariance des différents attributs de nos échantillons montre de fortes corrélations en cas de critères de groupage semblables. Par exemple, l'activité groupée par paire d'adresses (source et destination) est fortement corrélée avec l'activité groupée par couple d'adresse et par protocole IP. En effet, les courbes de ces attributs sont souvent semblables.

En conséquence, la technique de la sélection par élimination récursive des attributs, qui part de l'ensemble des attributs et élimine les attributs les moins significatifs de manière récursive, n'a pas montré de bons résultats. En effet, lorsqu'un attribut était retiré pour tester son importance, il était remplacé par un autre attribut fortement corrélé avec ce dernier. Cet attribut était donc considéré comme peu important de manière souvent erronée ce qui rendait la sélection difficile.

Nous avons donc appliqué la méthode inverse, consistant à sélectionner le meilleur attribut de manière récursive. Ici, meilleur veut dire que la sélection de cet attribut

Caractéristique	Paramètre	Valeur
Activité *(10 Hz, 1s)*	α	0.348678
Variation du n° de port	ΔT	1.0s
(3s d'effet mémoire)	k	$\pm$16384

TABLE 4.7 – Paramètres de calcul des équations 4.4 et 4.8

améliore le plus le score F1. Pour chaque attribut non encore sélectionné, il s'agissait donc d'entraîner le modèle avec ce nouvel attribut et de voir celui qui apporterait le plus grand gain, comme cela est décrit dans [100].

Nous avons toutefois écarté certains attributs afin de garder un modèle relativement agnostique concernant le protocole transport. En effet, utiliser le champ protocole de l'entête IP ou les fanions TCP comme attribut donne au modèle un moyen de discriminer les attaques TCP des attaques réalisées via un autre protocole transport. Bien que ces attributs amélioraient parfois significativement le score F1, ils auraient rendu le modèle incapable de généraliser à UDP.

Il est à noter aussi que l'utilisation du port TCP/UDP est directement liée à la présence d'une couche transport et qu'une attaque ICMP par exemple pourrait ne pas être détectée par notre modèle.

Avec les méthodes décrites précédemment, nous avons sélectionné les quatre attributs suivants :

— l'activité du nœud de destination ;

— l'activité par paire source/destination, par protocole transport ;

— la disparité du port source, groupée par adresse destination et protocole IP ;

— la disparité du port destination, groupée par adresse destination et protocole IP.

Les paramètres des équations 4.4 et 4.8 utilisées pour le calcul de ces caractéristiques dans les différents jeux de données sont donnés dans le tableau 4.7.

La table 4.8 liste les paramètres de l'algorithme de classification Random Forest utilisés lors de l'entraînement de notre modèle et tels que retournés par la fonction get_params de SciKitLearn v0.22.2.post1.

4.4.2 Scénarios de test

L'entraînement et la validation du modèle a été faite sur la base du jeu de données contenant du trafic capturé sur un réseau filaire. Afin d'affiner encore la validation, nous avons également effectué une simulation à partir d'application spécifiques aux drones afin d'obtenir des données plus réalistes. C'est pourquoi nous avons construit le scénario de test suivant : 4 drones et une station sol, impliquant du trafic de

Paramètres	Valeur
n_estimators	100
min_weight_fraction_leaf	0.0
class_weight	None
min_samples_leaf	1
max_leaf_nodes	None
random_state	None
max_depth	None
bootstrap	True
max_samples	None
ccp_alpha	0.0
min_samples_split	2
max_features	'auto'
criterion	'gini'
warm_start	False
min_impurity_decrease	0.0

TABLE 4.8 – Paramètres du Random Forest Classifier SciKitLearn

Scénario	Attaque	Paramètres
trafic bénin	Sans	iperf + web + ping
patator	Force Brute	par défaut
scan_normal	NMAP	parcours normale
scan_polite	NMAP	parcours poli
scan_sneaky	NMAP	parcours discret
synflood	DoS	
OS fingerprint	NMAP	
UDPscan	NMAP	normal speed
explore	NMAP	polite speed

TABLE 4.9 – Scénarios de test et leurs paramètres

commande et de contrôle de la mission, la télémétrie pour chaque drone et des flux vidéo pour trois d'entre eux. De plus, les drones échangent quelques données entre eux, ce trafic simulant des données de mission et de coordination. Ces données ont été générées au travers de l'outil iperf, de requêtes HTTP et de pings.

A partir de ce trafic normal, nous avons simulé diverses attaques depuis un des drones et à destination d'un second, la victime. Nous avons utilisé pour cela différentes techniques, dont certaines n'étaient pas dans le jeu de données d'entraînement

		Bénin		Attaque		
Scénario	Exactitude	préc.	rappel	préc.	rappel	Δt
no_attack	1.0	1.00	1.00	-	-	N/A
finger	0.88	0.70	1.00	1.00	0.84	<1ms
polite scan	0.94	0.81	1.00	1.00	0.92	1ms
synflood	0.99	0.88	1.00	1.00	1.00	10ms
normal scan	0.90	0.74	1.00	1.00	0.87	14ms
patator	0.84	0.51	0.96	0.99	0.83	27ms
UDP scan	0.98	0.97	1.00	1.00	0.98	1.2s
sneaky scan	0.84	0.84	1.00	0.00	0.00	-
explore	0.18	0.18	1.00	0.00	0.00	-

TABLE 4.10 – Scores pour la détection binaire à base de RFC et délai de détection

du modèle, et différents paramètres : attaque en force brute, parcours de ports avec différents niveaux d'agressivité et différents protocoles transport, déni de service par inondation SYN, signature numérique du système d'exploitation ou découverte du réseau. La table 4.9 reprend les différents scénarios et leurs paramètres.

4.4.3 Détection des attaques

A partir des scénarios présentés ci-avant, nous avons capturé le trafic et extrait un jeu de données de test de la même manière que nous l'avions fait avec le jeu de données d'entraînement. Nous avons labellisé les données en séparant les flux nominaux des flux d'attaques. Enfin, nous avons appliqué le modèle et comparé les prédictions. Le délai entre le premier flux de l'attaque et la première détection qui correspond au temps que le modèle met pour détecter l'attaque et informer l'opérateur et éventuellement appliquer une contre mesure, a été mesuré.

Classification binaire

Les résultats de ces tests sont repris, en détaillant pour chaque scénario l'exactitude ainsi que la précision et le taux de rappel du trafic bénin et des flux d'attaque, dans la table 4.10. La dernière colonne donne le délai de détection.

On remarque que le taux de rappel pour le trafic bénin et de manière équivalente la précision de la détection d'attaque, sont très élevés et proches de 100% dans la plupart des cas. Ces deux mesures donnent une indication sur le taux de faux positifs, c'est à dire le nombre de flux bénins classifiés comme des attaques. Des vérifications ciblées ont montré qu'il s'agit dans ces cas là de communications bénignes initiées par l'attaquant au milieu de l'attaque. Il semble dans ce cas là assez diffi-

cile, même pour un expert, d'identifier un flux bénin au milieu d'une foule d'autres, sans avoir recours à des signatures plus précises. Ceci ne remet donc pas en cause l'efficacité du modèle ni son utilisation. Le modèle semble simplement considérer l'ensemble des transmissions d'un nœud agressif comme potentiellement dangereux. Les contre mesures que l'on peut envisager ici iront d'ailleurs dans le même sens : l'attaquant démasqué ou supposé sera alors simplement empêché de transmettre et l'échec de transmissions de données légitimes ne sera qu'un effet de bord acceptable en comparaison du risque que fait courir la présence d'un agresseur. Il pourra être intéressant toutefois de garder les flux préalablement établis, dans l'espoir de conserver des flux vitaux pour le drone ou son contrôle (par exemple commande et contrôle, télémétrie).

Nous avons également vérifié que le modèle identifiait bien un seul attaquant et la victime concernée (lorsque ceci est applicable), c'est à dire qu'il n'y avait pas de confusion et que les autres nœuds ne seraient pas impactés par une éventuelle contre mesure.

En contre partie, il existe des flux d'attaque mal classifiés mais le taux de détection reste élevé. Ces faux négatifs ne pénalisent d'ailleurs pas le délai de détection de l'attaque qui est, dans la plupart des cas, de l'ordre de quelques dizaines de millisecondes.

On remarque enfin trois exceptions à ces premières constatations. Deux scénarios n'étaient pas dans le jeu de données initial : le scénario de parcours de ports UDP et l'exploration du réseau. Cette dernière n'est pas détectée par le modèle et illustre l'inconvénient majeur de l'apprentissage supervisé, c'est-à-dire sa mauvaise adaptation à la nouveauté. Le parcours de ports en UDP est quant à lui bien détecté mais requiert un temps significativement plus long que pour le parcours en TCP. Enfin, on constate également qu'un parcours suffisamment lent permet de rendre la détection inopérante. Ce ralentissement entrave l'attaquant dans l'atteinte de son but : imposer une cadence de parcours de un port par seconde augmenterait le temps nécessaire à l'exploration des «well known ports» d'environ 17 minutes, celle des ports enregistrés d'environ 13 heures.

Classification multiclasses

Nous avons ensuite effectué le même test, mais cette fois-ci avec le modèle à plusieurs classes. La table 4.11 reprend tous ces résultats. Nous n'avons pas repris le détail par classe mais fait un résumé global pour faciliter la comparaison avec le cas binaire précédent. Ce faisant, nous avons considéré que le cas d'une détection positive mais avec une classe erronée constituait une mauvaise classification.

On retrouve ici des résultats relativement proches du cas binaire pour certains scénarios. Toutefois, le modèle ne donne pas de résultats exploitables quant à la

		Bénin		Attaque		
Scénario	Exactitude	préc.	rappel	préc.	rappel	Δt
no_attack	1.0	1.0	1.0	-	-	N/A
finger	0.78	0.55	1.0	1.0	0.70	2ms
polite scan	0.95	0.82	1.0	1.0	0.93	3s
synflood	0.84	0.93	1.0	1.0	0.84	23ms[1]
normal scan	0.8	0.58	1.0	1.0	0.72	<1ms
patator	0.15	0.68	0.96	0.0	0.0	- [2]
UDP scan	0.99	0.98	1.0	1.0	0.99	<1ms
sneaky scan	0.84	0.84	1.0	0.0	0.0	-
explore	0.18	0.18	1.0	0.0	0.0	-

[1] Cette valeur correspond au temps écoulé entre le début de l'attaque et la première prédiction correcte. Le modèle a identifié une attaque à 19ms, mais elle ne correspondait pas à un déni de service.

[2] Le modèle a bien détecté un comportement anormal au bout de 1ms, mais n'a jamais prédit un attaque en force brute.

TABLE 4.11 – Scores pour la détection multiclasses à base de RFC et délai de détection

séparation entre les classes : il lui arrive de détecter un comportement anormal, mais se montre incapable de le classifier correctement. Par exemple, une attaque de force brute n'est pas identifiée comme telle, alors même que le modèle classifie les flux comme non bénins au bout de 1 milliseconde. De même, une attaque synflood est détectée au bout de 19 millisecondes dans notre scénario, mais il faut quelques flux supplémentaires pour que le modèle converge vers la bonne classe.

Cette incapacité ou ce manque de sélectivité est sans doute liée au nombre réduit d'attributs que nous avons retenus. Ce manque de sélectivité nous conduit, en l'état, à ne pas pouvoir envisager de contre mesure adaptée au type d'attaque. Il n'est toutefois pas surprenant car nous attendons de ce second modèle d'être plus sélectif que le premier, sans lui apporter plus d'information. Ainsi, l'attaque SYN flood et l'attaque en force brute impliquent un seul et même comportement, du point de vue des attributs sélectionnés : un attaquant ouvre de manière répétitive et intensive, un nombre important de flux sur la même victime et sur le même port. Sans information sur le déroulé normal de l'établissement de la connexion TCP (threeway handshake), il paraît peu probable qu'un expert puisse faire la différence ici.

La différence de topologie entre les jeux de données d'entraînement et de vérification nous donne une confiance raisonnable sur l'absence de fuites de données entre elles, qui pourraient expliquer les bonnes performances du modèle. Il est à noter

toutefois que l'attaque en force brute a été réalisée avec le même outil.

Enfin, en terme de performances, l'exécution d'une simple prédiction a une durée moyenne située entre 4 et 45 microsecondes sur un Intel® Core™ i7-4710MQ CPU @ 2.50GHz en utilisant SciKitLearn v0.22.2.post1.

Chapitre 5

Conclusion

Ce chapitre reprend l'ensemble des travaux et contributions de cette thèse. Nous présentons également les perspectives et futurs travaux.

Sommaire

5.1 Travaux réalisés

L'usage de drones, d'abord principalement militaire, s'est largement répandue ces dernières années autant pour le loisir que pour des besoins industriels. Ces aéronefs miniatures et sans pilote permettent notamment des prises de vue aériennes et un transport facile et rapide à moindre coût. Toutefois, les performances d'un drone seul restent limitées, l'autonomie en particulier. C'est pourquoi certaines missions nécessitent d'utiliser une flotte de drones.

Cela pose toutefois la question de leur sécurité alors que le nombre d'attaques est en constante augmentation sur Internet. Cette question est d'autant plus importante que la liaison entre la flotte de drones et son opérateur est sans fil ; que les aéronefs peuvent évoluer dans des espaces aériens proches des avions de transport et que des attaques sur des drones seuls ont déjà été menées avec succès.

La particularité d'un essaim de drones, en comparaison avec des missions assurées par un aéronef unique, est sa forte dépendance aux échanges qui ont lieu entre ces systèmes. Il existe des exemples d'attaques sur les drones ciblant les différents éléments d'un tel système : le guidage au travers des signaux GPS, la durée de vie de la batterie et les applications embarquées. Dans cette thèse, nous nous attachons à étudier les risques et à explorer des pistes permettant d'améliorer la sécurité et de protéger le réseau et les systèmes contre des attaques.

Comme sur tout réseau sans fil, les systèmes communiquant ainsi mis en relation sont vulnérables face à un nombre important d'attaques rencontrées communément sur les réseaux filaire, en plus d'attaques spécifiques. Ce champ de recherche a donné lieu à de nombreux travaux sur la sécurisation des protocoles de routage ou sur le brouillage. Notre approche, en plus de permettre une sécurisation du routage, apporte des barrières supplémentaires.

5.1.1 Une architecture orientée vers la sécurité

Nous avons tout d'abord étudié l'utilisation des réseaux définis par logiciel dans notre cas d'étude. Cette technologie apporte des outils pour collecter des mesures statistiques sur le trafic dans le réseau ainsi qu'un contrôle fin sur les communications dans le réseau. Nous avons considéré son utilisation dans un réseau sans-fil multi-sauts sans infrastructure au sol et mis en œuvre cette architecture par émulation sur un système Linux.

Nous avons proposé l'utilisation d'une version sécurisée d'AODV comme protocole de routage sur le plan de contrôle pour les faibles ressources qu'il mobilise et pour ses bonnes performances dans le cas d'un réseau ad hoc de drones. Le routage sur le plan de données est quant à lui protégé par l'utilisation de protocoles cryptés. Notons que cette architecture ne nécessite la définition d'aucun protocole supplé-

mentaire. Cette signalisation du plan de contrôle est de plus réalisée dans la bande et donc avec une seule carte réseau sans fil, ce qui en facilite l'intégration.

Une architecture de type réseau défini par logiciel est par nature centralisée. Cette centralisation, bien que présentant des menaces spécifiques, apporte une vision globale et la possibilité de disposer de données en dehors du domaine réseau, ce qui offre de nouvelles opportunités pour la détection d'attaques ou pour le routage.

Les délais de relayage hors transmission sur le réseau sans fil de notre architecture, de l'ordre de la milliseconde, sont comparables à celles obtenues avec une architecture plus classique basée uniquement sur AODV. L'indisponibilité sur une route du réseau en cas de changement topologique dépend avant tout du temps de détection de la perte de voisinage. Elle est toutefois dégradée par le temps de traitement de l'application SDN gérant la mobilité ainsi que par des délais éventuels de retransmission sur le plan de contrôle : d'environ 1s en moyenne en AODV, elle est d'un peu moins de 1,6s en SDN/AODV. Ces indisponibilités engendrent également des pertes de paquets supplémentaires. Le coût indirect lié à l'utilisation de SDN au travers du protocole OpenFlow est estimé à moins de 1 pourcent pour un scénario de capture vidéo. Des améliorations possibles ont été identifiées, notamment en fonction de l'utilisation de retours d'information depuis la couche liaison de données et de la fréquence d'émission des messages de vie du protocole de routage.

En terme de sécurité, le fait que le commutateur SDN soit sur le drone lui même permet de manière certaine d'associer certains ports du switch aux applications du drone : on définit ainsi la bordure du réseau et le contrôleur est capable d'identifier les flux qui en sont issus. Par cette approche il est alors aisé d'exclure les flux issus de nœuds en dehors du réseau et d'ignorer dans une certaine mesure les transmissions non légitimes. Des attaques communément rencontrées sur un réseau sans fil, seules l'injection de trafic avec usurpation d'identité et l'écoute passive restent pertinentes.

5.1.2 Détection d'attaques

Les statistiques retournées par SDN dans le traitement des paquets permettent justement d'identifier une injection de trafic au niveau d'un nœud. Mieux, il est possible de modifier les flux dans le réseau afin de contrer de telles injections. Cette méthode nécessite toutefois la transmission des statistiques pour chaque nœud, y compris lorsqu'aucune attaque n'a lieu. Par ailleurs, la contre-mesure peut être contournée par l'attaquant s'il est capable d'identifier le flux qu'il souhaite attaquer autrement que simplement par les numéros de port. Mais il semble difficile d'aller plus loin sans considérer la sécurisation des applications embarquées.

Dans la première partie de nos travaux, nous avons ainsi mis en place et validé une architecture SDN comme base de la sécurisation du réseau de l'essaim de

drone. Elle place une première barrière contre des nœuds extérieurs au réseau en permettant d'identifier les transmissions venant d'une application. Nous avons donc considéré dans la seconde partie les scénarios d'attaque depuis des nœuds faisant partie du réseau, autrement dit l'éventualité d'avoir un nœud infecté par un logiciel malveillant.

La détection de ces attaques, comme dans un réseau filaire, passe par la mise en œuvre de techniques de détection d'intrusion. Cette dernière se place naturellement sur le contrôleur SDN, compte tenu de son rôle central et de sa connaissance du réseau. Pour profiter de la capacité de SDN de mettre en place des contre mesures suite à une attaque il est essentiel que la détection soit rapide. Enfin, nous avons cherché à limiter les échanges de données supplémentaires.

Ainsi, le calcul de statistiques à partir des informations de réception de paquets pour chaque nouveau flux (*PacketIn*) nous permet d'identifier des comportements de nœuds spécifiques à certaines attaques. En particulier, le parcours des ports pour détecter les ports ouverts, certaines attaques en déni de service ou encore la découverte de mots de passe par attaque en force brute sont détectées avec un niveau de précision élevé.

Cette détection est basée sur des techniques en apprentissage automatique, en utilisant un modèle *Random Forest*. Ce dernier présente en effet des propriétés intéressantes en particulier dans une phase exploratoire : il accepte des caractéristiques non numériques, ne nécessite aucune normalisation de celles-ci et il est possible d'interpréter le modèle obtenu ainsi que les raisons qui ont mené à une détection d'anomalie.

Pour le scénario testé, le modèle a été capable de détecter les attaques SYN flood (déni de service), scan de ports, l'identification de système d'exploitation et force brute avec un précision supérieure à 80% et un taux de faux positifs extrêmement bas. Seules les attaques menées de façon très lentes et donc peu efficaces, n'ont pas été identifiées par le modèle. Cette détection intervient de plus très rapidement, généralement de l'ordre de 10ms. Les modèles n'ont toutefois pas été capables de différencier les attaques les unes des autres.

5.2 Futurs travaux

Les travaux présentés dans ce mémoire ont permis de montrer la possibilité pour une application SDN de détecter des attaques réseau sur la base des PacketIn reçus au niveau du contrôleur. Nous avons proposé un ensemble de caractéristiques et un modèle d'apprentissage automatique pour ce faire. Il reste à mettre en œuvre l'ensemble sur une plate forme de démonstration, que ce soit sur des drones réels ou bien sur des nano-ordinateurs comme les Raspberry Pi.

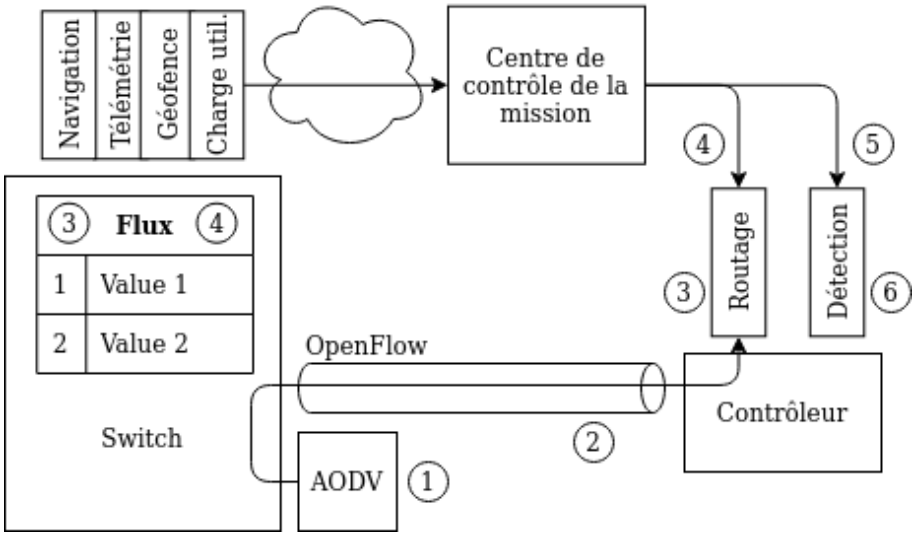


FIGURE 5.1 – Futurs travaux : éléments d'architecture

Au delà de cette plateforme de démonstration, les différents éléments mis en œuvre pourraient être étudiés pour améliorer les performances globales comme indiqué dans la figure 5.1 :

1. le choix d'une version sécurisée du protocole AODV sur le plan de contrôle peut être étudié pour en améliorer l'efficacité dans le contexte particulier d'une route unique vers le contrôleur. Il s'agit alors d'envisager des stratégies de routage alternatifs en secours pour maintenir la connectivité sur le plan de contrôle lors de changements topologiques et de diminuer le temps de rétablissement de la route optimale afin de revenir à une situation nominale le plus rapidement possible ;
2. l'utilisation d'un protocole transport sécurisé sans connexion comme DTLS ou Quick UDP Internet Connections (QUIC) [101] au lieu de TCP/TLS pour la connexion entre les commutateurs et le contrôleur SDN afin de réduire les délais de rétablissement lors de changements de topologie entraînant la rupture de la route vers le contrôleur ;
3. l'application SDN développée dans le cadre de cette thèse exploite uniquement l'information de visibilité radio remontée par les drones vers le contrôleur mais ne garantit pas la redondance des routes. Une version plus élaborée pourrait donc combiner le routage réactif implémenté ici avec un calcul de route alternative pour maintenir la connectivité sur le plan de données en cas de changement de topologie ;
4. la définition des routes n'exploite aucune autre donnée que la visibilité radio de chaque nœud. Les performances de routage pourraient profiter d'autres données si elles étaient rendues disponibles au niveau du contrôleur comme la position des drones (issue du centre de contrôle de mission), ou la qualité du

lien entre deux nœuds (issue de l'entité AODV) ;

5. la détection devrait être étendue à d'autres attaques comme les vers ou les botnets et pourrait prendre en compte des informations en dehors du domaine du réseau ;
6. d'autres algorithmes d'apprentissage automatique pour la détection d'anomalies pourraient être évalués, y compris l'apprentissage par renforcement.

Mais au-delà de l'approfondissement des solutions étudiées ici, l'ajout de données de mission ou de télémétrie peuvent également entrer dans les caractéristiques exploitables par un système de détection d'intrusion : la position des drones et le niveau de charge de leur batterie.

Enfin, les contre mesures envisagées ici étaient limitées aux capacités du contrôleur SDN lui même. D'autres réponses aux attaques sont envisageables. Si l'on considère le système de communication dans son ensemble par exemple, la modification des caractéristiques physiques de la transmission autoriserait des tentatives d'évasion en fréquence ou en modulation.

Une autre option est de considérer les autres éléments du système de drones. En particulier, les applications de navigation où la détection d'attaques extérieures au réseau pourraient amener à la définition de zones géographiques d'exclusion afin de limiter les risques d'attaques.

Bibliographie

[1] J. C. Correa Chica, J. C. Imbachi, and J. F. Botero Vega, "Security in SDN : A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, 2020.

[2] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "Ub-anc planner : Energy efficient coverage path planning with multiple drones," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6182–6189, 2017.

[3] C. Ju and H. I. Son, "Multiple uav systems for agricultural applications : Control, implementation, and evaluation," *Electronics*, vol. 7, no. 9, 2018.

[4] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications : A communications viewpoint," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016.

[5] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[6] Y. Mualla, A. Najjar, A. Daoud, S. Galland, C. Nicolle, A.-U.-H. Yasar, and E. Shakshuki, "Agent-based simulation of unmanned aerial vehicles in civilian applications : A systematic literature review and research directions," *Future Generation Computer Systems*, vol. 100, pp. 344–364, 2019.

[7] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in *2012 IEEE Conference on Technologies for Homeland Security (HST)*, pp. 585–590, Nov 2012.

[8] "Tiphon release 4 ; service independent requirements definition ; threat analysis," Tech. Rep. TR-101-771-V1.1.1, ETSI, 2001.

[9] E. A. Oladimeji, S. Supakkul, and L. Chung, "Security threat modeling and analysis : A goal-oriented approach," in *Proc. of the 10th IASTED International Conference on Software Engineering and Applications (SEA 2006)*, pp. 13–15, Nov 2006.

[10] C. G. L. Krishna and R. R. Murphy, "A review on cybersecurity vulnerabilities for unmanned aerial vehicles," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pp. 194–199, Oct 2017.

[11] V. L. L. Thing and J. Wu, "Autonomous vehicle security : A taxonomy of attacks and defences," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 164–170, Dec 2016.

[12] F. Fayollas and A. Vacher, "Drone predator, one drone to rule them all," Master's thesis, TLS-SEC, 2019. Accessed on October 2020.

[13] K. Ledieu, F. Pennel, and A. Pernot, "Attaques sur un ar drone 2.0 parrot," Master's thesis, TLS-SEC, 2019.

[14] B. Harris and R. Hunt, "Tcp/ip security threats and attack methods," *Computer Communications*, vol. 22, no. 10, pp. 885–897, 1999.

[15] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *Knowledge Engineering Review*, vol. 19, no. 4, pp. 281–316, 2004.

[16] R. G. Smith, "The contract net protocol : High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.

[17] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems : A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[18] Y. Jung, M. Kim, A. Masoumzadeh, and J. B. D. Joshi, "A survey of security issue in multi-agent systems," *Artificial Intelligence Review*, vol. 37, pp. 239–260, Mar 2012.

[19] S. Wang, J. Hu, A. Liu, and J. Wang, "Security frame and evaluation in mobile agent system," in *2005 2nd Asia Pacific Conference on Mobile Technology, Applications and Systems*, pp. 1–6, 2005.

[20] O. W. A. S. Project, "Owasp top ten." `https://owasp.org/www-project-top-ten/`. Accessed on February 2021.

[21] I. Bekmezci, O. K. Sahingoz, and S. Tamal, "Flying Ad-Hoc Networks (FANETs) : A survey," *Ad Hoc Networks*, vol. 11, no. 3, p. 1254–1270, 2013.

[22] J.-A. Maxa, M.-S. Ben Mahmoud, and N. Larrieu, "Survey on UAANET Routing Protocols and Network Security Challenges," *Ad Hoc & Sensor Wireless Networks*, Mar. 2017.

[23] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *SIGCOMM Comput. Commun. Rev.*, vol. 24, p. 234–244, Oct. 1994.

[24] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Proceedings. IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century.*, pp. 62–68, 2001.

[25] D. Johnson, Y. Hu, and D. Maltz, "Rfc4728 : The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4," 2007.

[26] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing." RFC 3561, July 2003.

[27] B. Karp and H. T. Kung, "Gpsr : Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, (New York, NY, USA), p. 243–254, Association for Computing Machinery, 2000.

[28] J.-A. Maxa, G. Roudière, and N. Larrieu, "Emulation-Based Performance Evaluation of Routing Protocols for Uaanets," in *Nets4Aircraft 2015*, vol. LNCS of *Nets4Cars/Nets4Trains/Nets4Aircraft 2015*, (Sousse, Tunisia), pp. 227–240, Springer, May 2015.

[29] A. Nadeem and M. P. Howarth, "A survey of manet intrusion detection and prevention approaches for network layer attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2027–2045, 2013.

[30] V. Mahajan, M. Natu, and A. Sethi, "Analysis of wormhole intrusion attacks in manets," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pp. 1–7, 2008.

[31] M. G. Zapata, "Secure ad hoc on-demand distance vector routing," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 3, p. 106–107, 2002.

[32] Q. Li, M. Zhao, J. Walker, Y.-C. Hu, A. Perrig, and W. Trappe, "Sear : a secure efficient ad hoc on demand routing protocol for wireless networks," *Security and Communication Networks*, vol. 2, no. 4, pp. 325–340, 2009.

[33] J. Maxa, M. S. Ben Mahmoud, and N. Larrieu, "Extended verification of secure uaanet routing protocol," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–16, 2016.

[34] N. Battat, H. Seba, and H. Kheddouci, "Monitoring in mobile ad hoc networks : A survey," *Computer Networks*, vol. 69, pp. 82 – 100, 2014.

[35] J. Case and R. Mundy and D. Partain and B. Stewart, "Introduction and Applicability Statements for Internet Standard Management Framework." RFC 3410, 2002.

[36] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith, "Rfc3084 : Cops usage for policy provisioning (cops-pr)," 2001.

[37] K. S. Phanse and L. A. Dasilva, "Protocol support for policy-based management of mobile ad hoc networks," in *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No. 04CH37507)*, vol. 1, pp. 3–16, IEEE, 2004.

[38] C.-C. Shen, C. Jaikaeo, C. Srisathapornphat, and Z. Huang, "The guerrilla management architecture for ad hoc networks," in *MILCOM 2002. Proceedings*, vol. 1, pp. 467–472, IEEE, 2002.

[39] S. Ghannay, S. M. Gammar, F. Kamoun, and D. Males, "The monitoring of ad hoc networks based on routing," *IFIP Med-Hoc-Net*, 2004.

[40] H. Kazemi, G. Hadjichristofi, and L. A. DaSilva, "Mman-a monitor for mobile ad hoc networks : design, implementation, and experimental evaluation," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pp. 57–64, 2008.

[41] A. Munaretto, N. Agoulmine, and M. S. Fonseca, "Policy-based management of ad-hoc enterprise networks," 2002.

[42] M. K. Rafsanjani and A. Movaghar, "Identifying monitoring nodes with selection of authorized nodes in mobile ad hoc networks," *World Applied Sciences Journal*, vol. 4, no. 3, pp. 444–449, 2008.

[43] K. Gopalakrishnan and V. R. Uthariaraj, "Neighborhood monitoring based collaborative alert mechanism to thwart the misbehaving nodes in mobile ad hoc networks," *Eur. J. Sci. Res. v57 i3*, pp. 411–425, 2011.

[44] J. A. Alvarez Aldana, *A novel online functional testing methodology based on a fully distributed continuous monitoring approach applied to communicating systems*. Theses, Université Paris-Saclay, Sept. 2018.

[45] "SDN Architecture," Tech. Rep. TR-SDN-ARCH-V1.0, Open Networking Foundaion, 2014.

[46] "Open vSwitch (OvS)." `http://www.openvswitch.org/`.

[47] "Openflow® Switch Specification Ver 1.3.5," Specification ONF TS-023, Open Networking Foundation, Apr. 2015.

[48] "Ryu component-based software." `https://ryu-sdn.org/`.

[49] "Opendaylight project." `https://www.opendaylight.org/`.

[50] "Floodlight openflow controller." `https://github.com/floodlight/floodlight`.

[51] "Open network operation system." `https://opennetworking.org/onos/`.

[52] "RESTCONF Protocol." RFC 8040, Jan. 2017.

[53] C. Trois, M. D. Del Fabro, L. C. E. de Bona, and M. Martinello, "A survey on sdn programming languages : Toward a taxonomy," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 2687–2712, Fourthquarter 2016.

[54] T. Nelson, A. D. Ferguson, M. J. Scheer, and S. Krishnamurthi, "Tierless programming and reasoning for software-defined networks," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pp. 519–531, 2014.

[55] N. Foster, M. J. Freedman, R. Harrison, J. Rexford, M. L. Meola, and D. Walker, "Frenetic : a high-level language for openflow networks," in *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, pp. 1–6, 2010.

[56] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular sdn programming with pyretic," *Technical Report of USENIX*, p. 30, 2013.

[57] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker, "Netkat : Semantic foundations for networks," *Acm sigplan notices*, vol. 49, no. 1, pp. 113–126, 2014.

[58] L. Mamushiane, A. Lysko, and S. Dlamini, "A comparative evaluation of the performance of popular sdn controllers," in *2018 Wireless Days (WD)*, pp. 54–59, 2018.

[59] M. T.BAH, A. Azzouni, M. Nguyen, and G. Pujolle, "Topology discovery performance evaluation of opendaylight and onos controllers," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 285–291, 2019.

[60] "Station and Media Access Control Connectivity Discovery." IEEE 802.1AB, Sept. 2009.

[61] A. Azzouni, R. Boutaba, T. M. T. Nguyen, and G. Pujolle, "softdp : Secure and efficient topology discovery protocol for SDN," *CoRR*, vol. abs/1705.04527, 2017.

[62] G. Bianchi and A. Capone, "From dumb to smarter switches in software defined networks : an overview of data plane evolution," 2014.

[63] I. Ku, Y. Lu, M. Gerla, F. Ongaro, R. L. Gomes, and E. Cerqueira, "Towards software-defined vanet : Architecture and services," 2014.

[64] M. Abolhasan, J. Lipman, W. Ni, and B. Hagelstein, "Software-defined wireless networking : Centralized, distributed, or hybrid ?," vol. 29, 07 2015.

[65] H. C. Yu, G. Quer, and R. R. Rao, "Wireless sdn mobile ad hoc network : From theory to practice," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2017.

[66] J. Wang, Y. Miao, P. Zhou, M. S. Hossain, and S. M. M. Rahman, "A software defined network routing in wireless multihop network," *Journal of Network and Computer Applications*, vol. 85, pp. 76 – 83, 2017. Intelligent Systems for Heterogeneous Networks.

[67] K. Poularakis, Q. Qin, E. M. Nahum, M. Rio, and L. Tassiulas, "Flexible sdn control in tactical ad hoc networks," *Ad Hoc Networks*, vol. 85, pp. 71 – 80, 2019.

[68] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing network security through software defined networking (SDN)," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, 2016.

[69] M. Monshizadeh, V. Khatri, and R. Kantola, "Detection as a service : An sdn application," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pp. 285–290, 2017.

[70] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible network-based intrusion detection and prevention system on software-defined networks," in *2015 International Conference on Advanced Computing and Applications (ACOMP)*, pp. 106–111, IEEE, 2015.

[71] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense : Monitoring network utilization with zero measurement cost," in *Passive and active measurement (PAM)*, pp. 31–41, 2013.

[72] O. Oubbati, M. Atiquzzaman, T. Ahanger, and A. Ibrahim, "Softwarization of uav networks : A survey of applications and future trends," *IEEE Access*, vol. PP, 05 2020.

[73] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito, and S. Palazzo, "OPERETTA : An OPEnflow-based REmedy to mitigate TCP SYNFLOOD Attacks against web servers," *Computer Networks*, vol. 92, pp. 89 – 100, 2015.

[74] R. J. R. Mohammadi and M. Conti, "SLICOTS : An SDN-Based Lightweight Countermeasure for TCP SYN Flooding Attacks," *IEEE Transactions on Network and Service Management*, vol. 14, pp. 487–497, June 2017.

[75] D. Q. Le, T. Jeong, H. E. Roman, and J. W. Hong, "Traffic dispersion graph based anomaly detection," in *Proceedings of the 2011 Symposium on Information and Communication Technology, SoICT 2011, Hanoi, Viet Nam, October 13-14, 2011*, pp. 36–41, 2011.

[76] J. McCoy and D. B. Rawat, "Software-defined networking for unmanned aerial vehicular networking and security : A survey," *Electronics*, vol. 8, p. 1468, 12 2019.

[77] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking : evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, Jun 2018.

[78] U. of California, "Knowledge discovery and datamining cup 1999 data." `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 1999. Accessed on July 2020.

[79] "NSL-KDD dataset." `https://www.unb.ca/cic/datasets/nsl.html`, 2009. Accessed on July 2020.

[80] L. Boero, M. Marchese, and S. Zappatore, "Support vector machine meets software defined networking in ids domain," in *29th International Teletraffic Congress, Genoa, Italy*, pp. 25 – 30, 2017.

[81] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227 – 2235, 2011.

[82] P. S. Bithas, E. T. Michailidis, N. Nomikos, D. Vouyioukas, and A. G. Kanatas, "A survey on machine-learning techniques for uav-based communications," *Sensors*, vol. 19, no. 23, 2019.

[83] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "A hierarchical detection and response system to enhance security against lethal cyber-attacks in uav networks," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, vol. 48, no. 9, pp. 1594–1606, 2018.

[84] L. Tamilselvan and V. Sankaranarayanan, "Prevention of blackhole attack in manet," in *The 2nd International Conference on Wireless Broadband and Ultra Wideband Communications (AusWireless 2007)*, pp. 21–21, 2007.

[85] D. Ye, M. Zhang, and A. V. Vasilakos, "A survey of self-organization mechanisms in multiagent systems," *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, vol. 47, no. 3, pp. 441–461, 2017.

[86] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in software defined networks," in *2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pp. 1–8, 2014.

[87] I. D. Chakeres and E. M. Belding-Royer, "Aodv implementation design and performance evaluation," *International Journal of Wireless and Mobile Computing*, vol. 2, no. 3, p. 42, 2005.

[88] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "Sdn controllers : A comparative study," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–6, 2016.

[89] L. Berc, W. Fenner, R. Frederick, S. McCanne, and P. Stewart, "Rtp payload format for jpeg-compressed video." RFC 2435, Oct. 1998.

[90] A. Nayyar, "Flying adhoc network (fanets) : Simulation based performance comparison of routing protocols : Aodv, dsdv, dsr, olsr, aomdv and hwmp," in *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–9, 2018.

[91] J.-A. Maxa, M.-S. Ben Mahmoud, and N. Larrieu, "Joint model-driven design and real experiment-based validation for a secure UAV ad hoc network routing protocol," in *ICNS 2016, 2016 Integrated Communications Navigation and Surveillance Conference* , 2016 Integrated Communications Navigation and Surveillance (ICNS), (Herndon, United States), pp. pp 1E2–1 – 1E2–16 /, AIAA/IEEE, Apr. 2016.

[92] L. Joncheray, "A simple active attack against TCP," in *Proceedings of the 5th Conference on USENIX UNIX Security Symposium - Volume 5*, SSYM'95, (USA), p. 2, USENIX Association, 1995.

[93] K. Szczypiorski, "Steganography in TCP/IP networks. state of the art and a proposal of a new system-hiccups," *Warsaw University of Technology, Poland Institute of Telecommunications, Warsaw, Poland*, 2003.

[94] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski, *Information hiding in communication networks : fundamentals, mechanisms, applications, and countermeasures*. John Wiley & Sons, 2016.

[95] M. Carnein, D. Assenmacher, and H. Trautmann, "An empirical comparison of stream clustering algorithms," in *Proceedings of the Computing Frontiers Conference*, p. 361–366, 2017.

[96] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, "Kdd cup 99 data sets : A perspective on the role of data sets in network intrusion detection research," *Computer*, vol. 52, no. 2, pp. 41–51, 2019.

[97] U. of New Brunswick, "Cse-cic-ids2018 on aws." `https://www.unb.ca/cic/datasets/ids-2018.html`, 2018. Accessed on July 2020.

[98] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5 – 32, Oct. 2001.

[99] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–8, 2020.

[100] T. Hamed, R. Dara, and S. C. Kremer, "Network intrusion detection system based on recursive feature addition and bigram technique," *Computers & Security*, vol. 73, pp. 137 – 155, 2018.

[101] J. Iyengar and M. Thomson, “QUIC : A UDP-Based Multiplexed and Secure Transport.” RFC 9000, May 2021.

Liste des acronymes

AODV Ad hoc On-demand Distance Vector.

API Application Programming Interface.

ARP Address Resolution Protocol.

BGP Border Gateway Protocol.

C2 Command and Control.

CDF Cumulative Distribution Function.

CPU Central Processing Unit.

DDoS Distributed Denial of Service.

DNS Domain Name System.

DoS Denial of Service.

DPI Deep Packet Inspection.

DSDV Destination-Sequenced Distance Vector.

DSR Dynamic Source Routing.

DTLS Datagram Transport Layer Security.

EWMA Exponentially Weighted Moving Average.

FANET Flying Ad Hoc Network.

FIN Final.

GCS Ground Control Station.

GNSS Global Navigation Satellite System.

GPS Global Positionning System.

GPSR Greedy Perimeter Stateless Routing.

HIDS Host Intrusion Detection System.

HTTP Hypertext Transfer Protocol.

HTTPS HTTP Secure.

IAT Inter-Arrival Time.

ICMP Internet Control Message Protocol.

IDS Intruction Detection System.

IMNSAN Identifying Monitoring Nodes with Selection of Authorized Nodes.

IP Internet Protocol.

JSON JavaScript Object Notation.

LLDP Link Layer Discovery Protocol.

MAC Medium Access Control.

MANET Mobile Ad Hoc Network.

MPR Multi Point Relay.

NFV Network Function Virtualization.

NIDS Network Intrusion Detection System.

NMCAM Neighborhood Monitoring based Collaborative Alert Mechanism.

NTP Network Time Protocol.

OFDP OpenFlow Discovery Protocol.

OLSR Optimized Link State Routing.

OS Operating System.

OSI Open Systems Interconnection.

OSPF Open Shortest Path First.

PDU Protocol Data Unit.

PKI Public Key Infrastructure.

QUIC Quick UDP Internet Connections.

RAM Random Access Memory.

RDP Remote Desktop Protocol.

RERR Route Error.

REST Representational state transfer.

RFC Random Forest Classifier.

RREP Route Reply.

RREQ Route Request.

RST Reset.

RTP Real Time Protocol.

SAODV Secure AODV.

SDN Software Defined Network.

SEAR Secure Efficient Ad hoc Routing.

SNMP Simple Network Management Protocol.

SQL Structured Query Language.

SSH Secured Shell.

SUAP Secure UAV Ad hoc routing Protocol.

SUMO Small Unmanned Meteorological Observer.

SVM Support Vector Machine.

SYN Synchronize.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TTL Time to Live.

UAANET Unmanned Aerial Ad Hoc Network.

UAS Unmanned Aerial System.

UAV Unmanned Aerial Vehicle.

UDP User Datagram Protocol.

UNB Université du New Brunswick.

VANET Vehicular Ad Hoc Network.

VTOL Vertical Take Off and Landing.