

***T**able des matières*

<i>Liste des Figures</i>	<i>3</i>
<i>Liste des Tableaux</i>	<i>5</i>
<i>Glossaire</i>	<i>6</i>
<i>Introduction</i>	<i>7</i>
<i>Chapitre 1 : Concepts Généraux</i>	<i>8</i>
I. Consommation des nœuds	9
II. Classification du trafic	9
1) Définition	9
2) Les méthodes conventionnelles de classification du trafic	10
3) Les Limites	10
4) Cas d'un trafic chiffré	11
III. Classification et intelligence artificielle	12
1. Vue d'ensemble sur l'intelligence artificielle	12
2. Les avantages de l'utilisation de l'IA dans la classification du trafic réseau	13
IV. Architecture SDN	14
1. Définition	14
2. Fonctionnement de SDN	15
3. Avantage de l'utilisation de SDN	16
V. Synthèse	17
<i>Chapitre 2 : Etude de l'existant</i>	<i>18</i>
I. Mémoire De Mr Bessala [1]	19
II. Publication de recherche de Feifei Hu ¹ , Situo Zhang ¹ , Xubin Lin ¹ , Liu Wu ¹ , Niandong Liao ^{2*} and Yanqi Song sur la thématique : Network traffic classification model based on attention mechanism and spatiotemporal features	19
<i>Chapitre 3 : Expérimentation et Présentation des résultats</i>	<i>20</i>
I. Classification du trafic	21
1. Architecture du système	21
2. Le DataSet	22
3. Les algorithmes utilisés	27
4. Outils utilisés	28
5. Environnement de développement	29
6. Structuration du dossier	29
7. Lancement du projet	29
a) Configuration du fichier wifi_classifier2.py	29
8. Mesures d'évaluation	32
a) Accuracy Score (Score de précision)	32
b) F1 Score	33
9. Résultats et Discussion	34
II. Consommation des nœuds	37
1) Power top	37
2) Contiki-NG / Cooja	37
3) CPU Energy Meter	38
4) Procédure	39
5) Résultats	39

*Simulation de la consommation des noeuds et Classification du trafic dans un réseau maillé
sans fils programmable*

III. Interface de monitoring	39
Conclusion	41
Annexe	42
I. Annexe 1 : Power Top	42
II. Annexe 2 : Contiki - NG / Cooja	44
References	48

Liste des Figures

Figure 1 : Visualisation des différentes méthodes de classification	12
Figure 2 : Architecture classique vs SDN	14
Figure 3 : Vue d'ensemble topologie physique de l'architecture SDN	15
Figure 4 : Architecture SDN vu d'ensemble topologie logique	15
Figure 5 : Vue globale de l'architecture du système	21
Figure 6 : Caractéristiques retenues sur les Datasets	22
Figure 7 : Vue sur le dataset non normalisé	23
Figure 8 : Vue sur le code de normalisation du Dataset	25
Figure 9 : Vue sur le Dataset normalisé	26
Figure 10 : Vue sur le code de sectionnement du dataset	27
Figure 11 : Vue du résultat de l'arbre de décision	28
Figure 12 : Vue de la localisation des chemins dans le code	30
Figure 13 : Vue du tableau des résultats de la classification du trafic	32
Figure 14 : Matrice de Confusion XGBoost	
Figure 15 : Matrice de Confusion AdaBoost	34
Figure 16 : Matrice de Confusion CatBoost	
Figure 17 : Matrice de Confusion Random Forest	34
Figure 18 : Matrice de Confusion Logistic regression	
Figure 19 : Matrice de Confusion DecisionTree	35
Figure 20 : Matrice de Confusion GaussianNb	
Figure 21 : Matrice de Confusion MLP 15 HL	35
Figure 22 : Matrice de Confusion MLP 50 HL	
Figure 23 : Matrice de Confusion MLP 20 HL	35
Figure 24 : Présentation de l'interface de monitoring	39
Figure 25 : Vue du code de rafraîchissement	40
Figure 26 : Vue d'ensemble des statistiques	43

Figure 27 : Vue d'ensemble du Fichier CSV	43
Figure 28 : Aperçue d'un fichier COOJA.testlog.....	46
Figure 29 : Résultats de compilation	47

Liste des Tableaux

Table 1 : Tableau des commandes de D-ITG	31
Table 2 : Présentation des scores des modèles suivant différentes métriques.	36
Table 3 : Présentation des résultats de quelques classificateurs en situation réelle.	36

Glossaire

Réseau informatique: Un réseau informatique est un ensemble d'ordinateurs et d'autres périphériques connectés entre eux pour partager des ressources, des données et des services. Les réseaux informatiques permettent la communication et l'échange d'informations entre les différents appareils connectés.

Classifier le trafic réseau: Le fait de classifier le trafic réseau consiste à analyser et à étiqueter les flux de données qui traversent un réseau en fonction de différents critères tels que le protocole utilisé, le type de service, l'adresse source ou de destination, etc. Cela permet de comprendre et de contrôler le trafic réseau, de mettre en place des politiques de sécurité et de prioriser les flux de données en fonction de leurs caractéristiques.

Nœud: Dans le contexte des réseaux informatiques, un nœud fait référence à tout appareil connecté à un réseau, tel qu'un ordinateur, un commutateur, un routeur, un point d'accès sans fil, etc. Les nœuds jouent un rôle essentiel dans la transmission, le routage et le traitement des données au sein du réseau.

Trafic chiffré: Le trafic chiffré est le trafic réseau qui est crypté pour des raisons de sécurité et de confidentialité. Lorsque les données sont chiffrées, elles sont converties en une forme illisible à l'aide d'un algorithme de chiffrement, de sorte que seules les personnes ou les systèmes disposant de la clé de déchiffrement appropriée peuvent les lire. Le chiffrement est couramment utilisé pour sécuriser les communications en ligne, telles que les transactions bancaires, les échanges d'informations sensibles, etc.

Intelligence artificielle: L'intelligence artificielle (IA) est un domaine de l'informatique qui vise à développer des systèmes qui peuvent effectuer des tâches qui nécessitent normalement une intelligence humaine. L'IA comprend des techniques telles que l'apprentissage automatique (machine learning), le traitement du langage naturel, la vision par ordinateur, la logique floue, etc. L'objectif de l'IA est de permettre aux machines de comprendre, d'apprendre, de raisonner et d'agir de manière autonome pour résoudre des problèmes et prendre des décisions.

Introduction

De nos jours, avec l'augmentation du nombre d'utilisateurs sur les différentes plates-formes numériques, les réseaux classiques informatiques ont pris un coup, que ce soit au niveau de la sécurité ou de la performance. Dans cette optique, il devient important de trouver des méthodes afin d'avoir une meilleure gestion du réseau à une échelle industrielle, afin de satisfaire toutes les parties. Ces méthodes engloberaient des façons nouvelles de classifier le trafic réseau. Une approche nouvelle a été proposée dans le rapport de mémoire de M.Bessala [1]. Cette approche a pour thème : "Qualification Intelligente du Trafic dans un réseau maillé sans fil programmable contraint". Il est question ici d'intégrer les outils d'intelligence artificielle pour mieux classifier le trafic réseau. C'est dans ce contexte que nous intervenons à travers le thème : "Routage multi-chemins intelligent dans les réseaux maillés sans fil programmables". Pour ces travaux, nous nous limiterons à la simulation de la consommation des nœuds et à la classification du trafic.



Chapitre 1 : Concepts Généraux



I. Consommation des nœuds

Entendons par nœud un point de connexion dans un réseau capable d'envoyer, de recevoir, de transmettre ou de stocker des données. Il s'agit ainsi des dispositifs tels que : un ordinateur, un routeur, un commutateur, un serveur ou tout autre appareil pouvant réaliser au moins l'une des fonctionnalités précédemment citées. Ainsi, pour assurer l'effectivité de toutes ces fonctions, il est nécessaire d'évaluer les besoins de chacun des nœuds pour une meilleure appréhension des sollicitations du réseau dans sa globalité : C'est dans ce contexte qu'on parlera de consommation des nœuds. L'étude de la consommation des nœuds va permettre d'identifier les besoins du réseau de part en part afin d'envisager des solutions pour une meilleure gestion des ressources disponibles, des prévisions sur les besoins, et la longévité. Ceci dans l'optique de garantir que le réseau en entier puisse satisfaire la demande des utilisateurs. On distinguera plusieurs critères sur lesquels s'appuyer pour évaluer la consommation [2] d'un réseau à savoir :

Consommation de la bande passante : quantité de données transmise ou reçue Par un nœud pendant une période donnée. Certains équipements en fonction de Leurs activités peuvent consommer une quantité considérable de la bande passante. En l'occurrence, un serveur de Streaming (netflix, youtube, spotify) transmet en Continu de grandes quantités de données à de nombreux utilisateurs.

Consommation des ressources systèmes : certains nœuds tels que les serveurs Ou ordinateurs hébergeant des applications pourront consommer une quantité Importante de ressources système comme la puissance de calcul, la mémoire, le Stockage, etc. Par exemple, un serveur exécutant des bases de données complexes Ou des algorithmes d'apprentissage automatique consommera beaucoup de Ressources système.

Consommation d'énergie : certains équipements réseaux tels que les Commutateurs, les routeurs et serveurs peuvent consommer beaucoup d'énergie, Ce qui peut devenir important à prendre en compte en termes de coût opérationnel Et de durabilité environnementale. Cet aspect est souvent négligé lors de L'évaluation d'un réseau.

II. Classification du trafic

1) Définition

La classification du trafic réseau est un processus permettant de catégoriser les différents types de données qui circulent sur un réseau informatique. Cela est généralement fait en analysant les caractéristiques des paquets de données, tels que les adresses IP source et destination, les ports utilisés, les protocoles de communication, les motifs de trafic, etc. La classification du trafic réseau est importante pour plusieurs raisons. Elle permet de surveiller

et d'analyser le trafic réseau pour des raisons de sécurité, de gestion du réseau et d'optimisation des performances. En classifiant le trafic, les administrateurs réseau peuvent identifier les applications, les protocoles ou les utilisateurs qui consomment le plus de bande passante, détecter des activités suspectes ou non autorisées, et mettre en place des politiques de gestion du trafic adaptées.

2) Les méthodes conventionnelles de classification du trafic

Il existe différentes méthodes de classification du trafic réseau, dont voici quelques exemples :

- **Classification basée sur les ports** : Cette méthode consiste à classer le trafic en fonction des numéros de port utilisés par les applications. Par exemple, le trafic utilisant le port 80 est généralement associé au trafic web, tandis que le trafic utilisant le port 25 est généralement associé au trafic de messagerie électronique.
- **Classification basée sur les protocoles** : Cette méthode repose sur l'analyse des protocoles de communication utilisés par les paquets de données. Par exemple, le trafic utilisant le protocole HTTP est généralement associé au trafic web, tandis que le trafic utilisant le protocole SMTP est généralement associé au trafic de messagerie électronique.
- **Classification basée sur les signatures** : Cette méthode utilise des signatures préétablies pour identifier des types spécifiques de trafic, tels que les signatures de logiciels de partage de fichiers peer-to-peer, les signatures de logiciels malveillants, etc.
- **Classification basée sur l'analyse comportementale** : Cette méthode repose sur l'observation des schémas de trafic et des caractéristiques de comportement pour classer le trafic. Par exemple, le trafic qui présente des pics d'activité réguliers peut être associé à une application de streaming vidéo.

Il est important de noter que la classification du trafic réseau est un domaine en constante évolution, car de nouvelles applications et de nouveaux protocoles apparaissent régulièrement. Par conséquent, les méthodes de classification doivent être mises à jour régulièrement pour rester efficaces.

3) Les Limites

Les limites de la classification conventionnelle :

Classification basée sur les ports : Certains protocoles peuvent utiliser des ports non standards ou peuvent même utiliser des ports dynamiques qui changent à chaque session. De plus, les applications peuvent utiliser des techniques d'évitement de port pour masquer leur trafic réel.

Classification basée sur les protocoles : Certains protocoles peuvent être encapsulés dans d'autres protocoles, ce qui rend difficile leur identification. Des techniques d'offuscation peuvent également être utilisées pour masquer le protocole réel utilisé.

Classification basée sur les signatures : Les signatures doivent être constamment mises à jour pour prendre en compte les nouvelles applications et les nouvelles variantes de logiciels malveillants. Les signatures peuvent également être contournées par des techniques d'offuscation.

Classification basée sur l'analyse comportementale : Certains types de trafic peuvent avoir des comportements similaires, ce qui rend difficile leur distinction. De plus, les schémas de trafic peuvent évoluer avec le temps, ce qui nécessite une mise à jour régulière des modèles de comportement.

4) Cas d'un trafic chiffré

Cas d'un trafic chiffré :

Lorsque le trafic réseau est chiffré, les données sont encodées de manière à ce qu'elles ne puissent pas être lues par des tiers non autorisés. Cela peut rendre la classification du trafic plus difficile, car les informations contenues dans les paquets de données sont masquées. Voici comment le chiffrement du trafic peut affecter les méthodes de classification traditionnelles :

Classification basée sur les ports : Lorsque le trafic est chiffré, il devient plus difficile d'identifier les applications en se basant uniquement sur les numéros de port. Le chiffrement peut masquer les informations sur les ports utilisés, rendant difficile la distinction entre différents types de trafic.

Classification basée sur les protocoles : Lorsque le trafic est chiffré, il devient également difficile d'analyser les protocoles utilisés dans les paquets de données. Les en-têtes de protocole sont chiffrés, ce qui rend difficile la détermination du type de protocole utilisé.

Classification basée sur les signatures : Lorsque le trafic est chiffré, les signatures préétablies peuvent ne pas être détectées, car elles sont masquées par le chiffrement. Les techniques d'offuscation peuvent également être utilisées pour modifier les signatures et éviter leur détection.

Classification basée sur l'analyse comportementale : Le chiffrement du trafic peut également rendre difficile l'analyse comportementale du trafic, car les schémas de trafic sont masqués. Les modèles de comportement peuvent être plus difficiles à détecter lorsque les données sont chiffrées.

Pour surmonter cette limitation, les méthodes de classification du trafic réseau doivent s'appuyer sur des techniques d'inspection plus avancées, telles que l'inspection approfondie des paquets (DPI) ou l'analyse du trafic basée sur le comportement et les caractéristiques

statistiques. L'utilisation de l'intelligence artificielle (IA) peut également être bénéfique pour analyser les données chiffrées et détecter des schémas et des comportements spécifiques, même lorsque le trafic est chiffré. Les techniques d'apprentissage automatique peuvent être utilisées pour apprendre à partir de modèles de trafic chiffré et effectuer une classification plus précise.

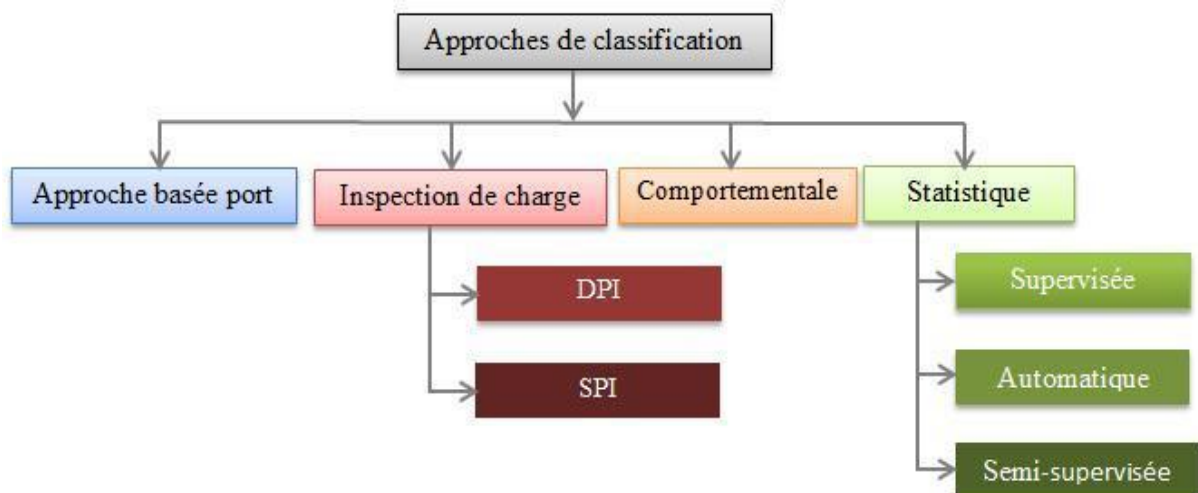


Figure 1: Visualisation des différentes méthodes de classification

III. Classification et intelligence artificielle

1. Vue d'ensemble sur l'intelligence artificielle

L'intelligence artificielle (IA) est un domaine de l'informatique qui vise à créer des systèmes capables de simuler ou de reproduire certaines capacités intellectuelles humaines. L'objectif de l'IA est de permettre aux ordinateurs et aux machines d'effectuer des tâches qui nécessitent normalement l'intelligence humaine, telles que la perception, le raisonnement, l'apprentissage, la résolution de problèmes, la compréhension du langage naturel et la prise de décision. Elle se compose de diverses techniques et méthodes qui permettent aux machines d'acquérir, de traiter et d'interpréter des informations, et de prendre des décisions en fonction de ces informations. Voici quelques-unes des techniques couramment utilisées en IA :

- **Apprentissage automatique (machine learning)** : L'apprentissage automatique est une approche de l'IA qui permet aux machines d'apprendre à partir de données sans être explicitement programmées. Les algorithmes d'apprentissage automatique analysent les données, identifient des schémas et des relations, et utilisent ces connaissances pour effectuer des tâches spécifiques ou prendre des décisions. [3]
- **Réseaux de neurones artificiels (deep learning)** : Les réseaux de neurones artificiels sont des modèles inspirés du fonctionnement du cerveau humain. Ils

sont composés de couches de neurones interconnectés et sont utilisés pour des tâches telles que la reconnaissance d'images, la traduction automatique et la prédiction.

- **Logique floue** : La logique floue est une approche de l'IA qui permet de traiter des concepts ou des variables qui peuvent avoir des valeurs intermédiaires ou incertaines. Elle permet une représentation plus flexible et nuancée de la connaissance, ce qui est particulièrement utile pour des domaines où des degrés de certitude différents sont nécessaires.
- **Traitement du langage naturel (NLP)** : Le traitement du langage naturel vise à permettre aux machines de comprendre, d'analyser et de générer du langage humain. Cela comprend des tâches telles que la traduction automatique, la reconnaissance vocale, la compréhension du langage naturel et la génération de texte.
- **Vision par ordinateur** : La vision par ordinateur concerne l'analyse et l'interprétation des images et des vidéos par les machines. Elle comprend des tâches telles que la reconnaissance faciale, la détection d'objets, la segmentation d'images et la reconnaissance de gestes.

L'intelligence artificielle est utilisée dans de nombreux domaines, dont dans l'industrie du réseau. Elle continue d'évoluer rapidement et a un potentiel considérable pour résoudre des problèmes complexes et apporter des améliorations significatives dans de nombreux aspects de notre vie quotidienne.

2. Les avantages de l'utilisation de l'IA dans la classification du trafic réseau

Apprentissage automatique supervisé : En utilisant des techniques d'apprentissage automatique supervisé, des modèles peuvent être entraînés à partir de données de trafic préalablement étiquetées pour prédire la classe ou la catégorie à laquelle appartient un nouveau trafic non étiqueté. Cela permet d'automatiser le processus de classification et de fournir des résultats plus précis.

Apprentissage automatique non supervisé : L'apprentissage automatique non supervisé peut être utilisé pour découvrir des structures et des motifs cachés dans le trafic réseau sans étiquettes préalables. Par exemple, des algorithmes de regroupement (clustering) peuvent être utilisés pour identifier des groupes de trafic similaires, ce qui peut aider à la classification en regroupant les flux de trafic similaires dans des catégories distinctes.

Apprentissage profond : L'apprentissage profond, une branche de l'IA basée sur les réseaux de neurones artificiels profonds, peut être utilisé pour extraire des caractéristiques complexes et hiérarchiques à partir des données de trafic. Cela peut permettre une classification plus précise en identifiant des schémas de trafic subtils qui peuvent être difficiles à détecter avec d'autres méthodes.

En combinant ces techniques d'IA avec d'autres méthodes de classification traditionnelles, il est possible d'améliorer la précision et l'efficacité de la classification du trafic réseau. L'IA

permet également une adaptation plus rapide aux nouvelles formes de trafic et aux évolutions du réseau, ce qui est particulièrement important compte tenu de l'évolution constante des applications et des protocoles.

IV. Architecture SDN

1. Définition

L'architecture de réseau défini par logiciel (SDN - Software-Defined Networking) [4] est un paradigme émergent dans le domaine des réseaux informatiques. Elle vise à simplifier la gestion et le contrôle des réseaux en séparant le plan de contrôle qui gère les décisions de routage et de commutation (control plane) , du plan de données qui transfère les paquets de données (data plane).

Le contrôle est centralisé dans un contrôleur SDN, tandis que les commutateurs et les routeurs sont simplifiés et deviennent des entités passives, exécutant les instructions du contrôleur. Contrairement aux architectures traditionnelles, où les commutateurs et les routeurs prennent des décisions de manière autonome, l'architecture SDN centralise le contrôle du réseau, ce qui permet une gestion plus flexible et dynamique.

L'architecture SDN a été initialement proposée par des chercheurs de l'Université Stanford, notamment Nick McKeown, Scott Shenker, Martin Casado et Guido Appenzeller. Ils ont publié un article intitulé "Software-Defined Networking: A Comprehensive Survey" en 2011, qui a contribué à populariser le concept de SDN.

Cependant, il est important de noter que l'architecture SDN est le résultat des efforts collectifs de la communauté de recherche et de l'industrie des réseaux. Depuis sa proposition initiale, de nombreux chercheurs, ingénieurs et entreprises ont contribué au développement et à l'adoption de l'architecture SDN, en créant des normes, des protocoles et des technologies associées.

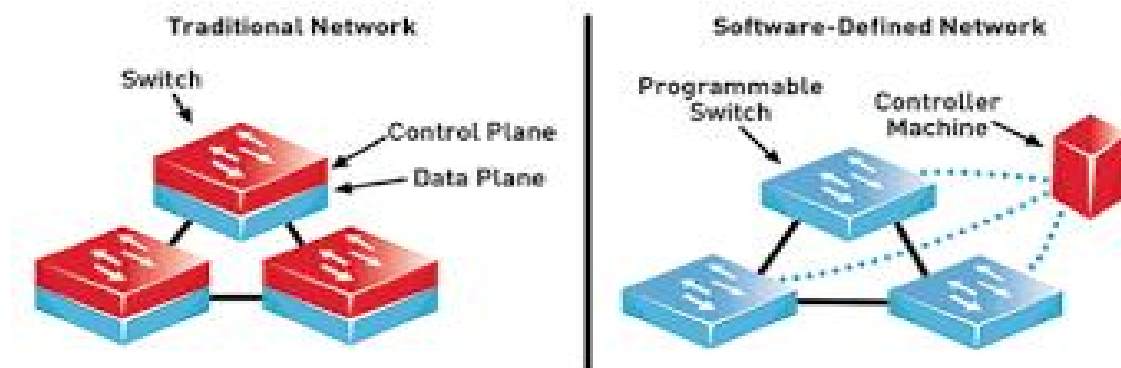


Figure 2 : Architecture classique vs SDN

2. Fonctionnement de SDN

Dans une architecture SDN, les commutateurs et les routeurs sont programmables via des interfaces ouvertes, telles qu'OpenFlow. Le contrôleur SDN communique avec les commutateurs pour établir des règles de routage, de commutation et de sécurité. Les décisions de contrôle sont prises par le contrôleur et transmises aux commutateurs pour le transfert des données.

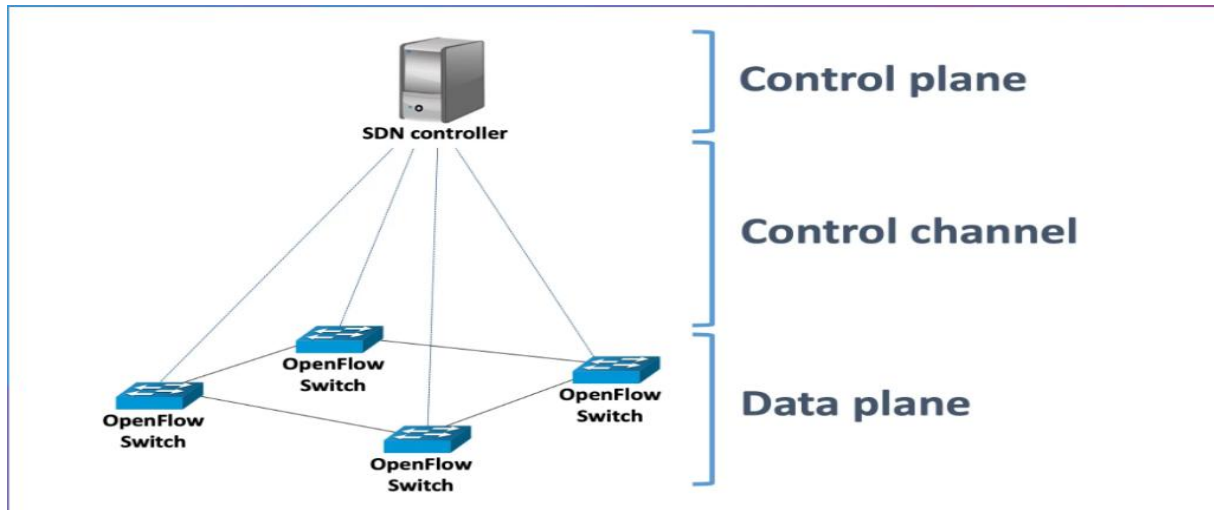


Figure 3 : Vue d'ensemble topologie physique de l'architecture SDN

En ce qui concerne les couches, nous en distinguons 3 : la couche application, contrôle et donnée.

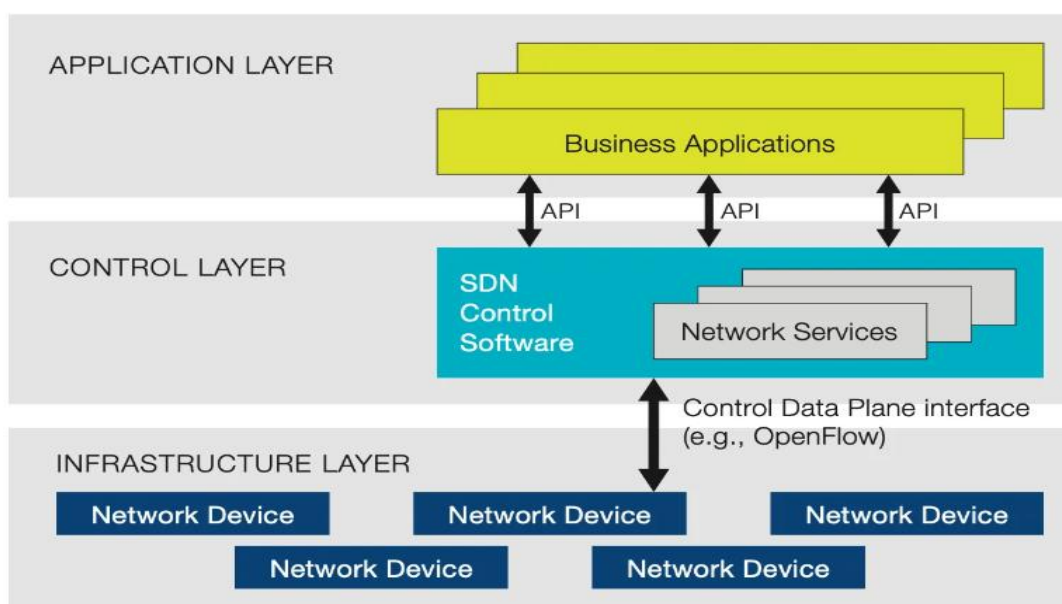


Figure 4 : Architecture SDN vu d'ensemble topologie logique

- **Couche d'application (APPLICATION LAYER) :**

Cette couche comprend les applications ou fonctions réseau typiques utilisées par les organisations, telles que les systèmes de détection d'intrusion, l'équilibrage de charge ou les pare-feu. Les applications de cette couche utilisent un contrôleur pour gérer le comportement du plan de données.

- **Couche de contrôle (CONTROLE LAYER) :**

Cette couche représente le logiciel de contrôleur SDN centralisé qui agit comme le cerveau du réseau défini par logiciel. Le contrôleur gère les politiques et les flux de trafic dans l'ensemble du réseau.

- **Couche d'infrastructure (DATA, INFRASTRUCTURE LAYER) :**

Cette couche est composée des commutateurs physiques du réseau. Les commutateurs acheminent le trafic réseau vers leurs destinations.

Plusieurs protocoles interviennent pour la communication dans SDN :

Pour la couche de contrôle, nous avons le contrôleur **RYU**. Pour la couche data, nous avons **mininet** qui nous permet de générer une topologie. Le protocole **openflow** nous permet de faire le pont entre la couche contrôle et data

3. Avantage de l'utilisation de SDN

L'architecture SDN permet une gestion centralisée du réseau, offrant une vue globale et une programmabilité étendue. Le contrôleur SDN gère les politiques de routage, les règles de sécurité et les services réseau, facilitant ainsi la configuration et l'optimisation du réseau.

- **Flexibilité** : L'architecture SDN permet une programmabilité et une configuration dynamique du réseau, facilitant l'adaptation aux besoins changeants.
- **Gestion centralisée** : La centralisation du contrôle facilite la gestion du réseau, la détection des problèmes et la mise en œuvre de politiques de sécurité cohérentes.
- **Séparation des fonctions** : La séparation du plan de contrôle et du plan de données permet d'innover plus facilement dans le contrôle du réseau sans perturber le transfert des données.
- **Automatisation** : L'architecture SDN facilite l'automatisation des tâches de gestion du réseau, réduisant ainsi les erreurs humaines et les délais de déploiement.

V. Synthèse

Dans cette partie, il était question de présenter les concepts généraux nécessaire à la bonne compréhension de nos travaux à savoir : la notion de consommation de nœud, classification du trafic, apprentissage automatique, architecture SDN.



Chapitre 2 : Etude de l'existant



La thématique de la consommation et la classification du trafic réseau n'est pas nouvelle dans le monde scientifique. Elle est d'autant plus importante que plusieurs travaux dans ce sens ont été menés. Nous citerons ici quelques travaux en vue de voir ce qui s'est fait autour de nous.

I. Mémoire De Mr Bessala [1]


Il a mené une étude sur qualification intelligente du trafic dans un réseau maillé sans fil programmable contraint. Il a réussi à intégrer à l'architecture SDN, en utilisant Ryu comme contrôleur, mininet wi-fi pour la topologie et D-itg pour la génération du trafic réseau, un algorithme de machine Learning de classification qui a produit un taux d'erreur relativement faible. Mais en situation réelle, ce modèle était peu performant. D'où la nécessité d'approfondir l'entraînement.

II. Publication de recherche de Feifei Hu¹ , Situo Zhang¹ , Xubin Lin¹ , Liu Wu¹ , Niandong Liao^{2*} and Yanqi Song sur la thématique : Network traffic classification model based on attention mechanism and spatiotemporal features


Ce papier nous dévoile l'utilisation de technique hybrides pour une meilleure classification . Ici ils couplent les LSTM (long short-term memory), Puis CNN (the convolutional neural network) et SE (the squeeze and excitation) [5]

Ce résumé présente une étude sur la classification du trafic réseau, qui est largement utilisée dans la sécurité et la gestion des réseaux. Les recherches antérieures se sont principalement concentrées sur la classification du trafic réseau des applications non chiffrées, mais peu de recherches ont été menées sur la classification du trafic réseau des applications chiffrées, en particulier le trafic sous-jacent de ces applications. Pour résoudre ces problèmes, cet article propose un modèle de classification du trafic réseau chiffré qui combine des mécanismes d'attention et des caractéristiques spatio-temporelles. Le modèle utilise d'abord la méthode du réseau de neurones à mémoire à court terme (LSTM) pour analyser les flux continus du réseau et trouver les caractéristiques de corrélation temporelle entre ces flux. Ensuite, la méthode du réseau de neurones convolutifs (CNN) est utilisée pour extraire les caractéristiques spatiales d'ordre supérieur du flux réseau. Enfin, le module "squeeze and excitation" (SE) est utilisé pour pondérer et redistribuer les caractéristiques spatiales d'ordre supérieur afin d'obtenir les caractéristiques spatiales clés du flux réseau. Grâce à ces trois étapes d'entraînement et d'apprentissage, une classification rapide des flux réseau est réalisée.

Les principaux avantages de ce modèle sont les suivants : (1) la relation de correspondance entre le flux réseau et l'étiquette est automatiquement construite par le modèle sans intervention manuelle et décision basée sur les caractéristiques du réseau, (2) il possède une forte capacité de généralisation et peut s'adapter rapidement à différents ensembles de données de trafic réseau, et (3) il peut traiter les applications chiffrées et leur trafic sous-jacent avec une grande précision. Les résultats expérimentaux montrent que le modèle peut être utilisé pour classer simultanément le trafic réseau des applications chiffrées et non chiffrées, et en particulier, la précision de classification du trafic sous-jacent des applications chiffrées est améliorée. Dans la plupart des cas, la précision dépasse généralement 90%.



Chapitre 3 : Expérimentation et Présentation des résultats

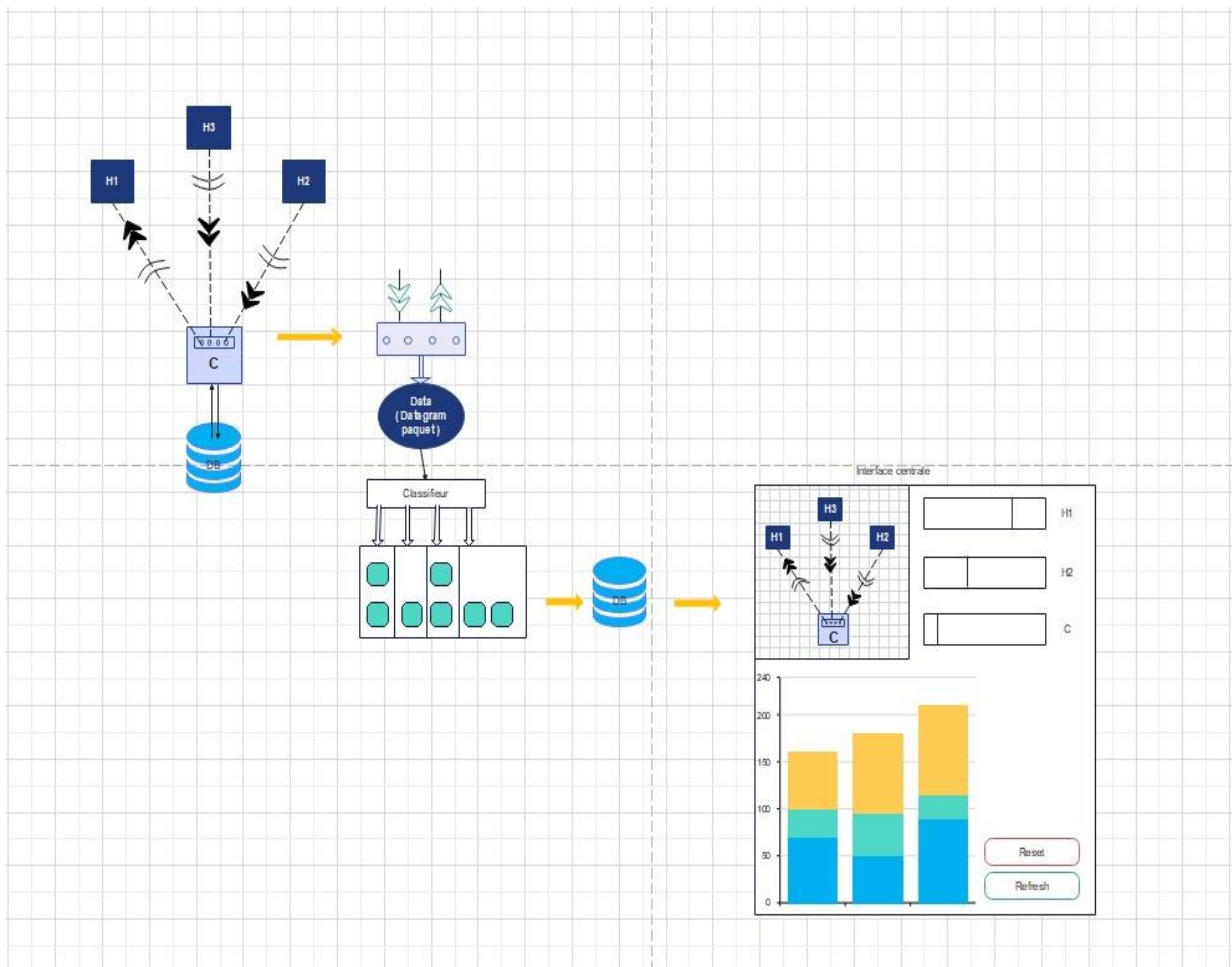


Notre travail avait pour but de simuler la consommation des nœuds, et de faire une classification du trafic dans un réseau maillé sans fil programmable, qui suit l'architecture SDN.

I. Classification du trafic

1. Architecture du système

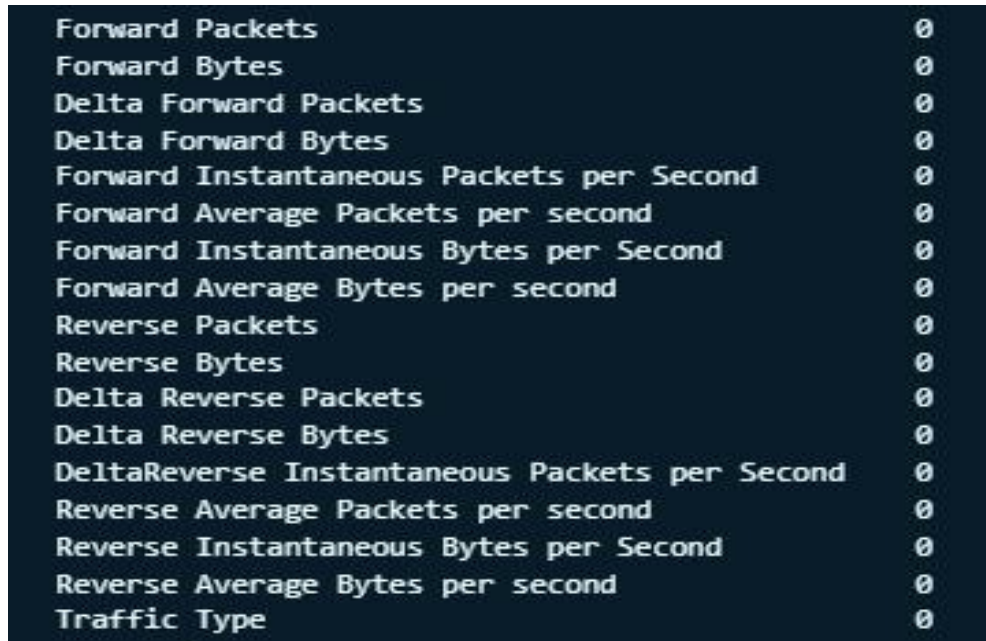
Nous utilisons l'architecture SDN dans notre réseau. Nous générons la topologie avec mininet wi-fi, nous lançons le contrôleur Ryu qui intègre en lui un classifieur (algorithme de machine Learning) qui nous dit si le paquet reçu est : DNS, TELNET, PING, QUAKE, VOIP, CSA, CSI, unknown, ...



2. Le DataSet

Notre data set a été obtenu après génération du trafic sur D-itg et enregistrer dans des fichiers .csv .Nous avons 7 dataset pour chaque nature de trafic : CSA , CSI , DNS,PING,QUAKE3,TELNET,VoIP. Chaque dataset a été généré pendant 60 minutes. On a environ 42 500 paquets par datasets.

Les caractéristiques retenues sont :



Forward Packets	0
Forward Bytes	0
Delta Forward Packets	0
Delta Forward Bytes	0
Forward Instantaneous Packets per Second	0
Forward Average Packets per second	0
Forward Instantaneous Bytes per Second	0
Forward Average Bytes per second	0
Reverse Packets	0
Reverse Bytes	0
Delta Reverse Packets	0
Delta Reverse Bytes	0
DeltaReverse Instantaneous Packets per Second	0
Reverse Average Packets per second	0
Reverse Instantaneous Bytes per Second	0
Reverse Average Bytes per second	0
Traffic Type	0

Figure 6 : Caractéristiques retenues sur les Datasets

Ils se trouvent dans le fichier dataset de notre dossier.

Prenons le cas de VoIp. Le data set enregistré avec pour nom Voice training est de cette forme :

	A	B	C	D
1	Forward Packets	Forward Bytes	Delta Forward Packets	Delta Forward Bytes
2	151470000.00.00.00.000000.00.00.00.0PING			
3	151470000.00.00.00.000000.00.00.00.0PING			
4	151470000.00.00.00.000000.00.00.00.0PING			
5	151470000.00.00.00.000000.00.00.00.0PING			
6	151470000.00.00.00.000000.00.00.00.0PING			
7	151470000.00.00.00.000000.00.00.00.0PING			
8	151470000.00.00.00.000000.00.00.00.0PING			
9	151470000.00.00.00.000000.00.00.00.0PING			
10	151470000.00.00.00.000000.00.00.00.0PING			
11	151470000.00.00.00.000000.00.00.00.0PING			
12	151470000.00.00.00.000000.00.00.00.0PING			
13	151470000.00.00.00.000000.00.00.00.0PING			
14	151470000.00.00.00.000000.00.00.00.0PING			
15	151470000.00.00.00.000000.00.00.00.0PING			
16	151470000.00.00.00.000000.00.00.00.0PING			
17	151470000.00.00.00.000000.00.00.00.0PING			
18	151470000.00.00.00.000000.00.00.00.0PING			
19	151470000.00.00.00.000000.00.00.00.0PING			
20	151470000.00.00.00.000000.00.00.00.0PING			
21	151470000.00.00.00.000000.00.00.00.0PING			
22	151470000.00.00.00.000000.00.00.00.0PING			
23	151470000.00.00.00.000000.00.00.00.0PING			
24	151470000.00.00.00.000000.00.00.00.0PING			
25	151470000.00.00.00.000000.00.00.00.0PING			
26	151470000.00.00.00.000000.00.00.00.0PING			
27	151470000.00.00.00.000000.00.00.00.0PING			
28	151470000.00.00.00.000000.00.00.00.0PING			
29	151470000.00.00.00.000000.00.00.00.0PING			
30	151470000.00.00.00.000000.00.00.00.0PING			
31	151470000.00.00.00.000000.00.00.00.0PING			
32	151470000.00.00.00.000000.00.00.00.0PING			

Figure 7 : Vue sur le dataset non normalisé

Le Dataset ici présenté n'est pas directement exploitable, il faut le normaliser.

```
ping_df = pd.read_csv('PING_training_data.csv', delimiter='\t')
voice_df = pd.read_csv('VOICE_training_data.csv', delimiter='\t')
quakes3_df = pd.read_csv('QUAKE3_training_data.csv', delimiter='\t')
dns_df = pd.read_csv('DNS_training_data.csv', delimiter='\t')
csi_df = pd.read_csv('CSI_training_data.csv', delimiter='\t')
csa_df = pd.read_csv('CSA_training_data.csv', delimiter='\t')
telnet_df = pd.read_csv('TELNET_training_data.csv', delimiter='\t')
dataset = pd.concat([ping_df[3000:45940], voice_df, quakes3_df, dns_df, telnet_df, csa_df, csi_df], ignore_index=True)
```


Figure 8 : Vue sur le code de normalisation du Dataset

Voici le résultat obtenu suite à la normalisation

dataset.head()

	Forward Packets	Forward Bytes	Delta Forward Packets	Delta Forward Bytes	Forward Instantaneous Packets per Second	Forward Average Packets per second	Forward Instantaneous Bytes per Second	Forward Average Bytes per second	Reverse Packets	Reverse Bytes	Delta Reverse Packets	Delta Reverse Bytes	Reverse Instantaneous Packets per Second	Reverse Average Packets per second	Reverse Instantaneous Bytes per Second	Reverse Average Bytes per second	Traffic Type
0	15	1470	0	0	0.0	15.0	0.0	1470.0	15	1470	0	0	0.0	15.0	0.0	1470.0	PNG
1	15	1470	0	0	0.0	15.0	0.0	1470.0	15	1470	0	0	0.0	15.0	0.0	1470.0	PNG
2	15	1470	0	0	0.0	15.0	0.0	1470.0	15	1470	15	1470	0.0	0.0	0.0	0.0	PNG
3	15	1470	0	0	0.0	15.0	0.0	1470.0	15	1470	15	1470	0.0	0.0	0.0	0.0	PNG
4	15	1470	0	0	0.0	0.0	0.0	0.0	15	1470	15	1470	0.0	0.0	0.0	0.0	PNG

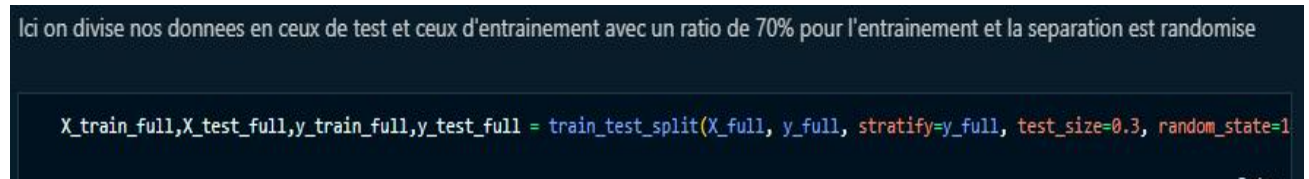
python

5 14 17 0 0 0

Figure 9 : Vue sur le Dataset normalisé

Une fois normalisé, nous pouvons mener une étude exploratoire pour apprendre de nos données. Cette dernière nous permet de voir les points forts entre certaines variables de notre Dataset, d'enlever les valeurs aberrantes et non significatives. Une fois tout ceci fait, nous avons un Dataset final prêt à être utilisé.

On va diviser notre data set en deux, 70% pour l'entraînement et 30% pour le test.



```

Ici on divise nos donnees en ceux de test et ceux d'entrainement avec un ratio de 70% pour l'entrainement et la separation est randomise

X_train_full,X_test_full,y_train_full,y_test_full = train_test_split(X_full, y_full, stratify=y_full, test_size=0.3, random_state=1

```

Figure 10 : Vue sur le code de sectionnement du dataset

3. Les algorithmes utilisés

L'étude exploratoire nous a montré une très forte corrélation entre les caractéristiques Forward entre eux et Reverse entre eux. Ceci nous ne permettant pas de faire un choix clair sur quel algorithme de machine Learning utiliser, nous avons opté pour différents algorithmes afin de les comparer : Nous avons utilisé

En machine Learning :

- Les arbres de décision(décision Tree)
- Les Forêts aléatoires (Random Forest)
- Le classifieur gaussien (Gaussian NB)
- La Régression logistique (Logistique régression)

En Deep Learning:

- Le perceptron multi-couches (MLP)
- Les réseaux de neurones récurrents (RNN)
- Les réseaux de neurone rapide (FNN)

Dans Jupiter notebook, nous utilisons la bibliothèque **sklearn** car elle implémente déjà tous ces algorithmes. Il suffit juste de l'appeler.

Exemple avec les arbres de décision :

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()

model.fit(X_train_full,y_train_full)

predictions = model.predict(X_test_full)

print('Accuracy full: %.2f%%' % (accuracy_score(y_test_full, predictions)*100))

Accuracy full: 99.99%
```

Figure 11 : Vue du résultat de l'arbre de décision

On commence par importer le model avec : `from sklearn.tree import DecisionTreeClassifier`. Puis on fait l'entraînement avec la fonction `.fit` , ensuite on passe à la prédiction , et on obtient une accuracy (exactitude de 99.99%)Pour plus de détail voir les notebooks commenté, ils se trouvent dans le dossier notebook du projet.

4. Outils utilisés

Nous avons utilisé :

- **Github** : pour le développement collaboratif et le versioning
- **Google docs** : pour la rédactions des différents rapports
- **Google colab** : pour l'entraînement de nos modèles.
- **Jupyter Notebook:** pour la définition des différents algorithmes de classification
- **Ryu** : comme contrôleur SDN
- **Mininet-wifi:** pour la génération de notre topologie
- **D-ITG:** pour la génération des trafic .

5. Environnement de développement

Nous utiliserons le système d'exploitation Linux ubuntu dont les caractéristiques sont les suivants

Distribution ID : ubuntu

Description : ubuntu 18.04.6 LTS

Release : 18.04

Codename : bionic

Pré-requis: python2, Python3, Ryu , D_itg , mininet , mininet_wi-fi installer sur notre ubuntu

6. Structuration du dossier

Le dossier s'appelle : **PROJET_RESEAU**

Il est constitué de 5 dossiers :

binary : comporte le fichier binaire du modèle de machine Learning utilisé

dataset : comporte tous les datasets utilisés pour l'entraînement du modèle

Documents : comporte tous les documents important relatif au projet (rapports, livres utiles ...)

Interface : ce fichier comporte le code de l'interface de monitoring.

Notebook : comporte les différents notebook utilisés lors de la phase d'entraînement des algorithmes de machine Learning.

src : comporte les fichiers de lancement du projet (Le wi-fi_AP_ST_Topo.py, wi-fi_classifier2.py, wi-fi_monitor_v2.py)

7. Lancement du projet

a) Configuration du fichier wi-fi_classifier2.py

Avant de lancer le projet, il faut déjà copier le dossier et le coller dans Document de votre machine Ubuntu.

```
proj_location_Src = "/home/bessala/PROJET_RESEAU/src/"
proj_location_dataset = "/home/bessala/PROJET_RESEAU/dataset/"
proj_location_Binary = "/home/bessala/PROJET_RESEAU/binary/"
```

Figure 12 : Vue de la localisation des chemins dans le code

Ensuite aller dans /PROJET_RESEAU/src/wi-fi_classifier2.py, modifier : « Bessala » par le nom de votre machine.

b) Configuration des terminaux

- Ouvrez deux terminaux ubuntu avec la commande : `crtl+alt+t`
- Passez en mode sudo : `sudo su` , puis entrez votre mot de passe .
- Dans chacun des terminaux , naviguez : `cd Document/PROJET_RESEAU/src`
- Dans le terminal 1 , lancez le classifier : `python3 wi-fi_classifier2.py`
« unsupervised » ou « supervised » , selon votre model de classification choisi.
- Puis Dans le terminal 2, lancez la topologie mininet : `python2 wi-fi_AP_ST_TOPO.py`
- **Nb** : servez-vous de la tabulation pour aller plus vite.

c) Génération du trafic

- Une fois le réseau virtuel mis en place avec mininet , on peut commencer à générer le trafic grâce à d_itg.
- Enter dans le terminal 2, taper « `xterm h1 h2` » .vous pouvez changer d'hôte ou de station à votre convenance en entrant la commande « `dump` » dans mininet pour visualiser tous les équipements créer par notre topologie. pour une bonne prise en main de mininet ,Ryu , openflow, ouvrez le document:/PROJET_RESEAU/documents/TP_4GI_SD_WMN.pdf
- Une fois les deux invites de commandes ouvertes, on va générer un trafic de h1 (10.0.0.1) vers h2 (10.0.0.2). h1 sera l'émetteur, et h2 le récepteur.
- Par exemple on veut générer le trafic DNS,
- Dans le xterm de h2 entrez la commande : `ITGRecv -l recv.log`
- Dans le xterm de h1 entrez la commande : `ITGSend -t 15000 -a 10.0.0.2 -rp 10003 DNS`

Dans nos travaux, nous nous sommes focalisés sur les trafic que D_itg permet de générer simplement.

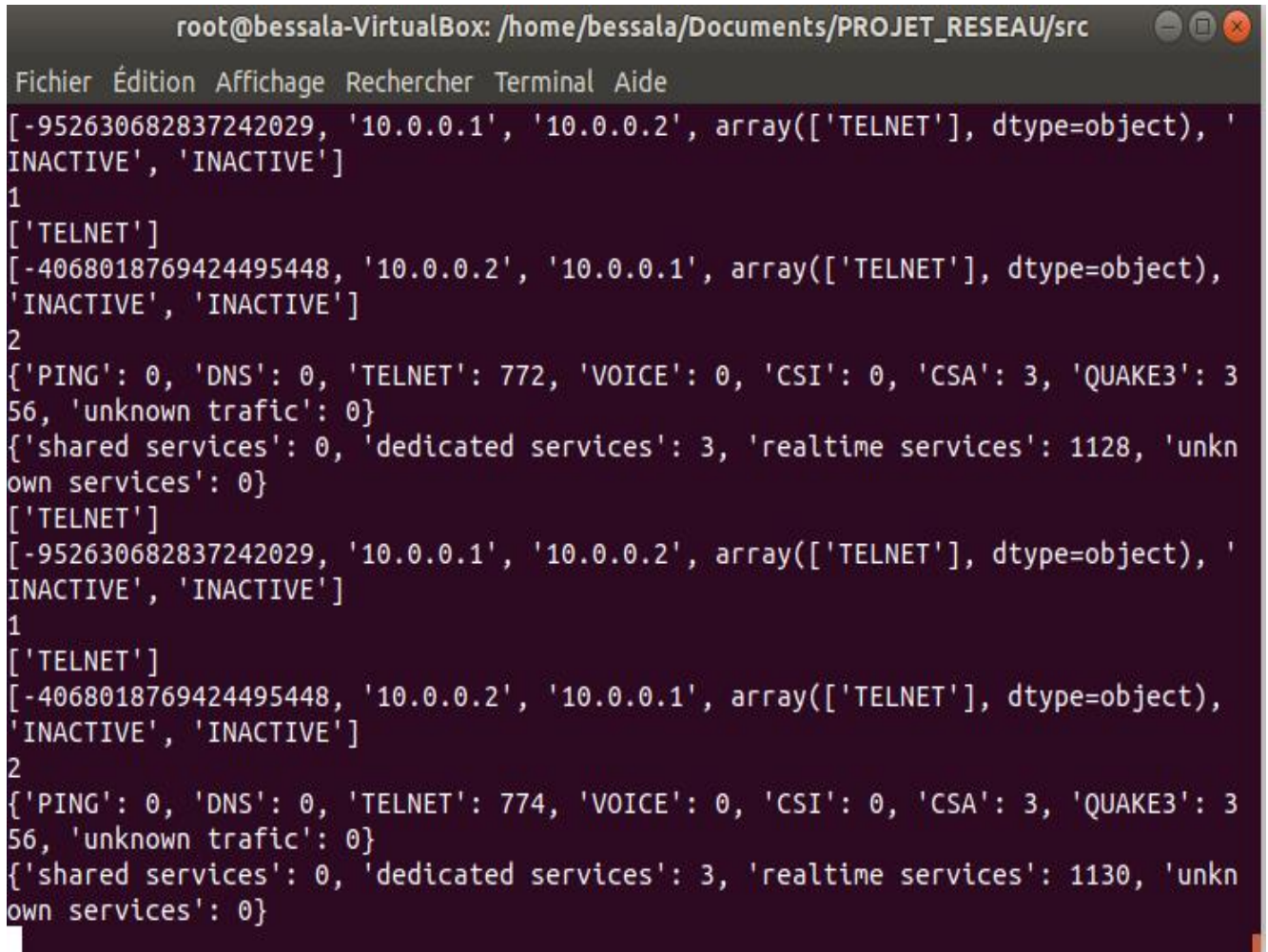
NOM TRAFIC	CODE EMETTEUR	CODE RECEPTEUR	COMMENTAIRE
PING	Ping « ip machine à ping »	Ping « ip machine à ping »	Il suffit de lancer un ping all dans la console de mininet wi-fi
DNS	ITGSend -t 15000 -a 10.0.0.2 -rp 10003 DNS	ITGRecv -l recv.log	L'émetteur envoie pendant 15000 millisecondes des paquets DNS au récepteur d'adresse ip 10.0.0.2
TELNET	ITGSend -t 15000 -a 10.0.0.2 -rp 10002 Telnet		
QUAKE	ITGSend -t 15000 -a 10.0.0.2 -rp 10002 Quake3		
VOIP	ITGSend -t 15000 -a 10.0.0.2 -rp 10001 VoIP -x G.711.2 -h RTP -VAD		
CSI	ITGSend -t 15000 -a 10.0.0.2 -rp 10002 Csi		
CSA	ITGSend -t 15000 -a 10.0.0.2 -rp 10002 Csa		

Table 1 : Tableau des commandes de D-ITG

Pour en savoir plus sur ces trafic voir le document : D_ITG-2.8.1-manual.pdf dans PROJET_RESEAU/documents/

d) Observation des résultats

Une fois le trafic lancé, on observe sur le terminal où on a lancé la `wi-fi_classifier2.py` supervised une classification en temps réel.



```
root@bessala-VirtualBox: /home/bessala/Documents/PROJET_RESEAU/src
Fichier Édition Affichage Rechercher Terminal Aide
[-952630682837242029, '10.0.0.1', '10.0.0.2', array(['TELNET'], dtype=object), '
INACTIVE', 'INACTIVE']
1
['TELNET']
[-4068018769424495448, '10.0.0.2', '10.0.0.1', array(['TELNET'], dtype=object),
'INACTIVE', 'INACTIVE']
2
{'PING': 0, 'DNS': 0, 'TELNET': 772, 'VOICE': 0, 'CSI': 0, 'CSA': 3, 'QUAKE3': 3
56, 'unknown traffic': 0}
{'shared services': 0, 'dedicated services': 3, 'realtime services': 1128, 'unkn
own services': 0}
['TELNET']
[-952630682837242029, '10.0.0.1', '10.0.0.2', array(['TELNET'], dtype=object), '
INACTIVE', 'INACTIVE']
1
['TELNET']
[-4068018769424495448, '10.0.0.2', '10.0.0.1', array(['TELNET'], dtype=object),
'INACTIVE', 'INACTIVE']
2
{'PING': 0, 'DNS': 0, 'TELNET': 774, 'VOICE': 0, 'CSI': 0, 'CSA': 3, 'QUAKE3': 3
56, 'unknown traffic': 0}
{'shared services': 0, 'dedicated services': 3, 'realtime services': 1130, 'unkn
own services': 0}
```

Figure 13 : Vue du tableau des résultats de la classification du trafic

Une fois le trafic terminé, on observe le tableau qui nous donne aussi la classification du nombre de service dédié rencontré. (shared services , dedicated services , real-time services) . Une fois fini, on ferme mininet_wi-fi avec la commande « quit » , on ferme le classifier avec `crtl+c`

8. Mesures d'évaluation

Nous avons utilisé les métriques accuracy score, f1 score, recall et le MSE .

a) Accuracy Score (Score de précision)

L'accuracy score mesure la précision globale d'un modèle de classification en calculant le pourcentage de prédictions correctes par rapport au nombre total d'échantillons. C'est la

mesure la plus simple, mais elle peut être trompeuse lorsque les classes sont déséquilibrées. Par exemple, si 95% des échantillons appartiennent à la classe A et que le modèle prédit toujours la classe A, L'accuracy score peut sembler élevé, mais le modèle n'est pas réellement performant pour détecter la classe B.

b) F1 Score

Le F1 score est une mesure qui combine la précision et le recall (rappel) d'un modèle de classification. Il est particulièrement utile lorsque les classes sont déséquilibrées. Le F1 score est la moyenne harmonique de la précision et du recall, et il donne un chiffre unique pour évaluer la performance globale du modèle. Il est souvent utilisé lorsque les faux positifs et les faux négatifs ont des conséquences similaires.

c) Recall (Rappel)

Le recall est une mesure qui évalue la capacité d'un modèle de classification à identifier tous les échantillons positifs. C'est le rapport entre le nombre d'échantillons positifs correctement identifiés (vrais positifs) et le nombre total d'échantillons positifs réels. Le recall est important lorsque la détection des échantillons positifs est cruciale, même au détriment de quelques faux positifs (par exemple, dans la détection de fraudes).

d) Mean Squared Error (MSE)

Le MSE est une mesure couramment utilisée pour évaluer les modèles de régression. Il calcule la moyenne des carrés des différences entre les valeurs prédites et les valeurs réelles. Le MSE pénalise davantage les erreurs plus importantes, car il s'agit d'une mesure quadratique. Une valeur de MSE plus faible indique une meilleure adéquation entre les prédictions du modèle et les valeurs réelles.

Il est important de choisir les mesures d'évaluation appropriées en fonction du problème spécifique que vous essayez de résoudre. Par exemple, L'accuracy score peut être suffisant lorsque les classes sont équilibrées, mais le F1 score ou le recall sont souvent préférés lorsque les classes sont déséquilibrées. De même, le MSE est généralement utilisé pour évaluer les modèles de régression, tandis que les mesures de classification sont plus adaptées pour les problèmes de classification.

9. Résultats et Discussion

a) Observation

Les matrices de confusion nous permettent d'avoir un aperçu global de la situation.

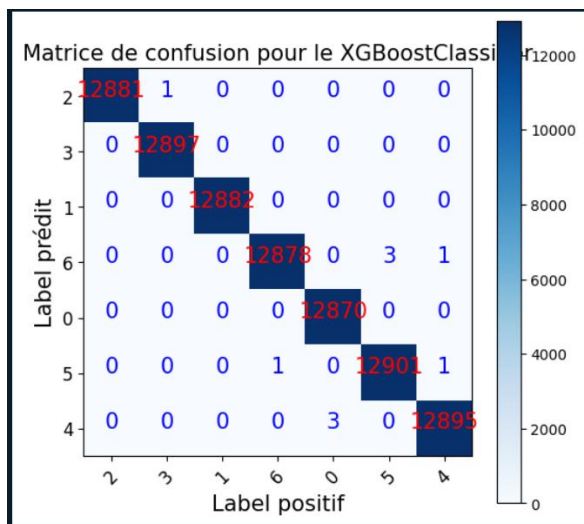


Figure 14 : Matrice de Confusion XGBoost

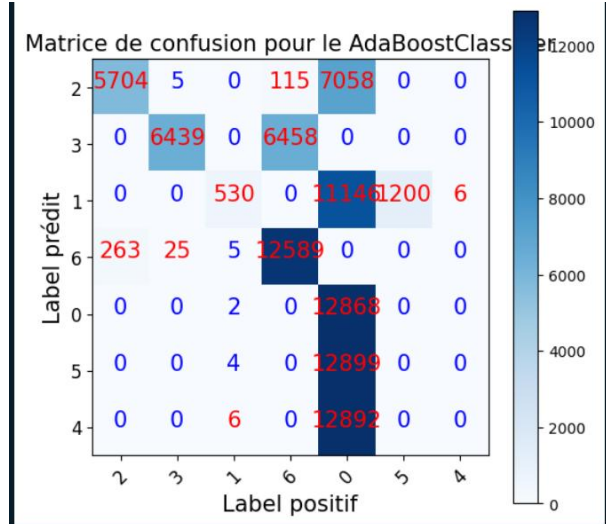


Figure 15 : Matrice de Confusion AdaBoost

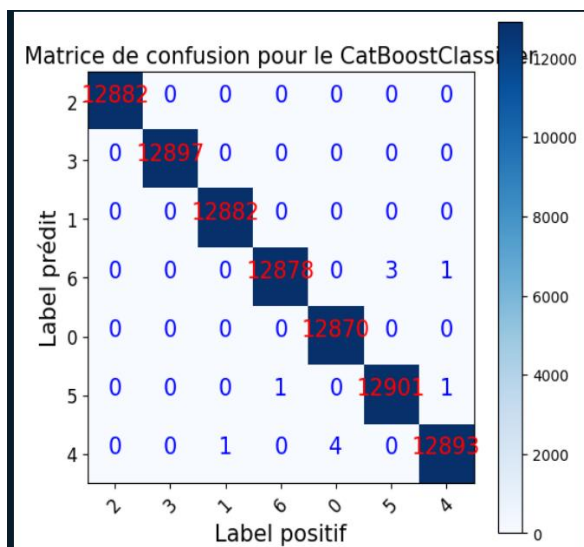


Figure 16 : Matrice de Confusion CatBoost

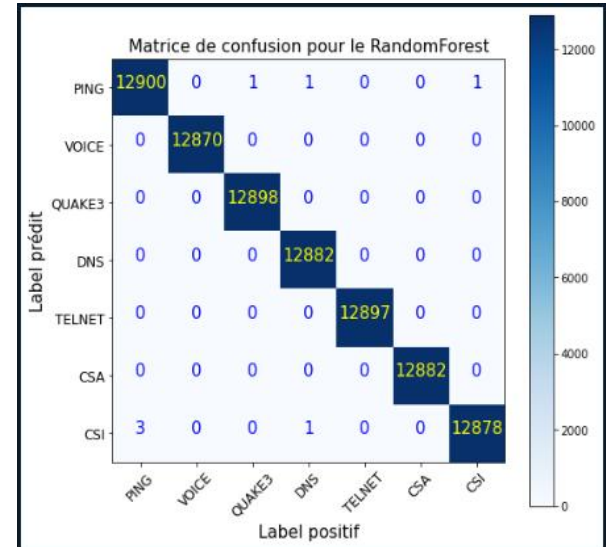


Figure 17 : Matrice de Confusion Random Forest

Simulation de la consommation des noeuds et Classification du trafic dans un réseau maillé sans fils programmable

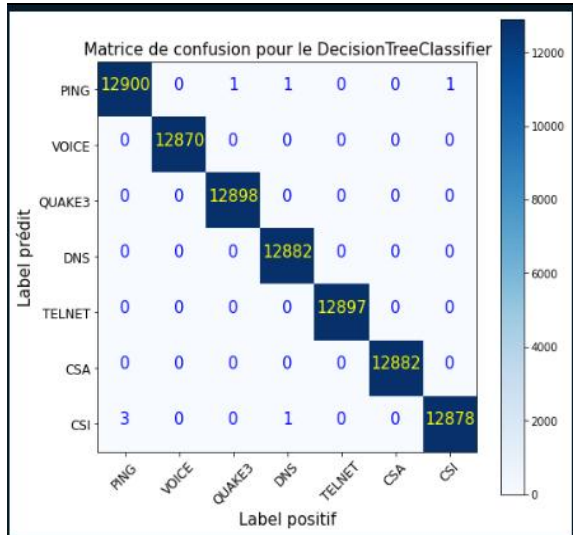


Figure 18 : Matrice de Confusion Logistic regression

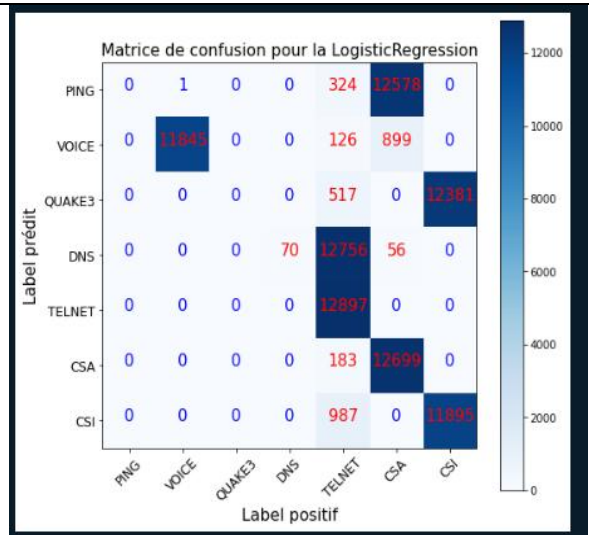


Figure 19 : Matrice de Confusion DecisionTree

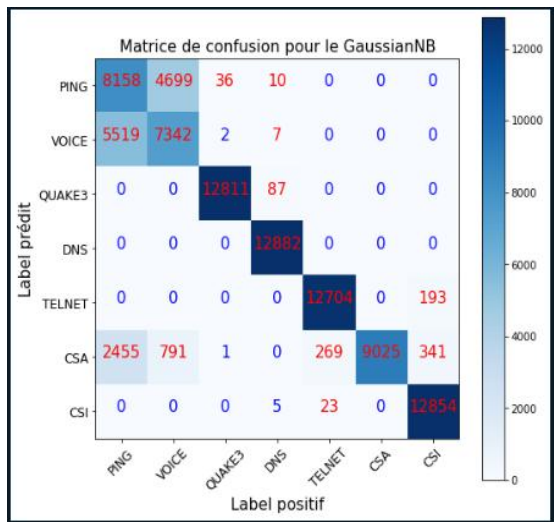


Figure 20 : Matrice de Confusion GaussianNb

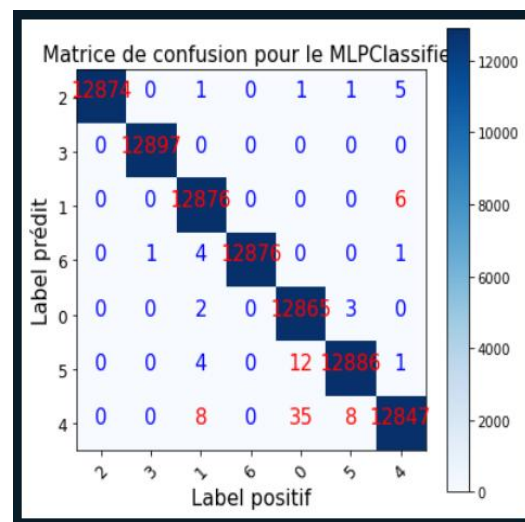


Figure 21 : Matrice de Confusion MLP 15 HL

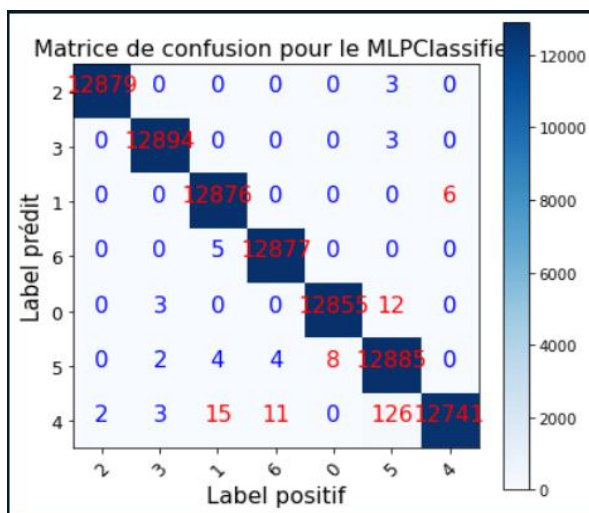


Figure 22 : Matrice de Confusion MLP 50 HL

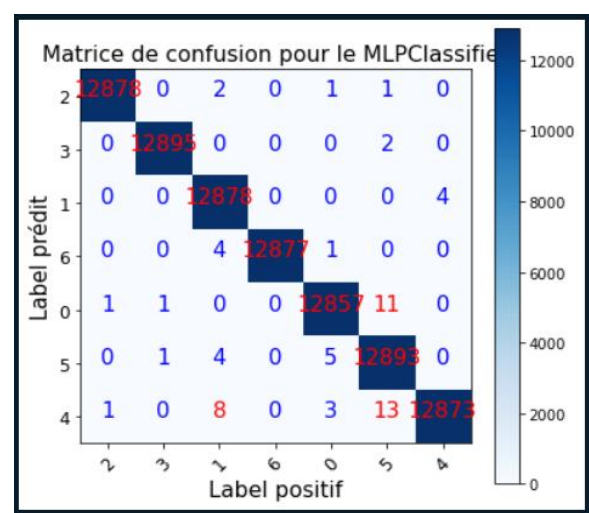


Figure 23 : Matrice de Confusion MLP 20 HL

Nous avons choisi de mettre les résultats de nos modèles dans un tableau.

	models	accuracy score	f1 score	recall	MSE
Machine Learning	XgBoost	0.999889	0.999889	0.999889	0.000432
	AdaBoost	0.422661	0.357609	0.422661	2.908185
	CatBoost	0.999878	0.999878	0.999878	0.000643
	Random Forest	0.999878	0.999878	0.999878	0.000443
	Decision Tree	0.999900	0.999900	0.999900	0.000321
	Logistic Regression	0.551090	0.450002	0.551090	4.699060
	GaussianNB	0.836333	0.840653	0.836333	1.029341
Deep Learning	MLP 15 Hidden Layers	0.998969	0.998969	0.998969	0.007471
	MLP 20 Hidden Layers	0.997705	0.997704	0.997705	0.010718
	MLP 50 Hidden Layers	0.999301	0.999301	0.999301	0.004789
	FNN 5 epochs	0.998614	-	-	12.289973
	FNN 10 epochs	0.712927	-	-	12.249119
	RNN 5 epochs	0.609018	-	-	12.22199
	RNN 10 epochs	0.610404	-	-	12.217326

Table 2 : Présentation des scores des modèles suivant différentes métriques.

e) Interprétation :

Pour les algorithmes de machine Learning :

- Le MSE de la régression Logistique supérieure à 4, donc elle n'est pas recommandée
- Vu que les classes de notre dataset ne sont pas déséquilibré, on peut se servir de la métrique accuracy pour exclure le AdaBoost , le Logistic Régression , et le Gaussian NB.
- Le CatBoost et le Random Forest obtiennent les même résultats , meilleur que le decision Tree.

Pour les algorithmes de deep Learning :

- Le perceptron multi-couches de 50 Hidden layer à de meilleurs résultats dans l'ensemble.

Nous recommandons donc l'utilisation du Random Forest et CatBoost pour les classificateurs de machine learning , et le MLP en ce qui concerne le deep learning .

Présentation des résultats des modèles en situation réelle.

	Ping	DNS	Telnet	Voice	Csi	Csa	Quake3
Random Forest Classifier	99,99%	99,99%	99,99%	99,85%	99,70%	99,65%	99,90%
MLP 50 Hidden layers	99,99%	97,00%	99,50%	99,45%	92,40%	92,85%	98,50%
Decision Tree Classifier	99,99%	99,95%	99,95%	99,70%	99,65%	99,25%	99,70%

Table 3 : Présentation des résultats de quelques classificateurs en situation réelle.

II. Consommation des nœuds

La modélisation de la consommation d'énergie des nœuds d'un réseau maillé sans fil est un problème important pour la conception et l'optimisation de ces réseaux, qui sont utilisés pour des applications variées. Il est essentiel de minimiser la consommation d'énergie des nœuds, afin de prolonger la durée de vie du réseau et d'assurer la qualité de service des applications.

La modélisation de la consommation d'énergie des nœuds d'un réseau maillé sans fil peut se faire à différents niveaux d'abstraction, selon les objectifs et les contraintes de la conception. On peut utiliser une approche par fonction ou une approche par réseau.

Dans **l'approche par fonction**, on identifie les différents états possibles du nœud: Sleep, Idle, Receive, et Transmit. En évaluant la consommation de chaque état et en la mesurant sur des fractions de temps, on peut retrouver la consommation énergétique moyenne d'un nœud:

$$P_m = t_{sl} * P_{sl} + t_{Id} * P_{Id} + t_{Rx} * P_{Rx} + t_{Tx} * P_{Tx}$$

avec t_i la fraction de temps passée par le nœud dans chaque état i et P_i l'énergie consommée dans l'état i .

Dans **l'approche par réseau**, on modélise le comportement du réseau dans son ensemble, en prenant en compte les interactions entre les nœuds, la topologie du réseau, le protocole de routage, le canal de communication, etc. Cette approche permet d'évaluer la consommation d'énergie du réseau en fonction du type de trafic et de la qualité de service. En analysant la consommation pour un type de trafic donné on peut modéliser la consommation énergétique des nœuds pour chaque flux d'informations envoyé ou reçu. [6]

Plusieurs pistes ont été observées :

1) Power top

PowerTOP est un outil logiciel open source conçu pour surveiller et optimiser la consommation d'énergie des systèmes Linux. Il fournit des informations détaillées sur la consommation d'énergie des différents composants matériels et logiciels d'un système, permettant aux utilisateurs d'identifier les sources de consommation d'énergie élevée et de prendre des mesures pour améliorer l'efficacité énergétique. Il fonctionne en collectant des données en temps réel sur la consommation d'énergie et en fournissant des estimations et des statistiques détaillées. Il fournit des mesures détaillées sur la consommation d'énergie, les états de veille, les réveils, et d'autres paramètres pertinents. Ces données pourront être utilisées pour une analyse approfondie de la consommation énergétique. Voir Annexe 1.[10]

2) Contiki-NG / Cooja

Contiki-NG comprend le module Energest qui peut être utilisé pour mettre en œuvre une approche légère d'estimation de l'énergie basée sur les logiciels pour les appareils IoT à ressources limitées. Le module Energest tient compte du temps passé par un système dans différents états. En utilisant ces informations conjointement avec le modèle de consommation d'énergie matérielle d'un système, le développeur peut estimer la consommation d'énergie du système.

Le module Energest de Contiki-NG permet de suivre cinq états prédéfinis liés à la consommation d'énergie, tels que l'activité du processeur, les modes de faible consommation d'énergie et les opérations de transmission et d'écoute de la radio. Pour activer le service simple-energest, il suffit d'ajouter une ligne spécifique dans le Makefile de l'application. La plupart des plates-formes Contiki-NG prennent en charge le suivi de ces états, mais il peut y avoir des exceptions en fonction du matériel, comme dans le cas des produits Texas Instruments. [7] Voir annexe 2

3) CPU Energy Meter

CPU Energy Meter est un outil Linux qui permet de surveiller la consommation d'énergie des processeurs Intel avec une granularité temporelle fine (quelques dizaines de millisecondes). La surveillance de la consommation d'énergie est disponible pour les domaines de puissance suivants :

- **Domaine par paquet (socket CPU) :** consommation d'énergie pour chaque paquet de CPU.
- **Domaine par cœur (tous les cœurs CPU sur un paquet) :** consommation d'énergie pour tous les cœurs CPU sur un paquet.
- **Domaine par unicore (composants uncore, par exemple, la carte graphique intégrée sur les CPU client) :** consommation d'énergie pour les composants uncore.
- **Domaine par nœud mémoire (mémoire locale à un paquet, uniquement pour les CPU serveur) :** consommation d'énergie pour chaque nœud mémoire.
- **Domaine par plate-forme :** consommation d'énergie pour tous les périphériques de la plate-forme qui reçoivent de l'énergie du mécanisme d'alimentation intégré (par exemple, les cœurs de processeur, SOC, mémoire, périphériques supplémentaires ou périphériques).

Cet outil permet d'obtenir des informations détaillées sur la consommation d'énergie des différents domaines de puissance des processeurs Intel, ce qui peut être utile pour surveiller et optimiser l'efficacité énergétique des systèmes informatiques.[8]

D'autres outils sont aussi explorés comme **Joulesmeter** ,**Pyjoules**. Pour voir celui qui correspondra le mieux à notre modèle [11] aussi certains outils d'administration réseau comme *snmp* peuvent aussi nous permettre d'avoir la consommation des nœuds au sein d'un réseau.

4) Procédure

A partir de ces outils, nous récupérerons la consommation de l'équipement à tout instant dans un fichier .csv, en fonction de la nature du trafic entrant. Ayant ce dataset , nous pourrons dans un futur proche , nous pourrons , soit estimer par la moyenne , la consommation moyenne de l'équipement , ou alors faire un algorithme de prédiction de la consommation des nœuds ,en fonction de la nature , la taille du paquet entrant , ainsi que la consommation du réseau à l'instant t-1. Ceci nous permettra de pouvoir donc quantifier efficacement la consommation des nœuds à tout instant.

5) Résultats

Les différents outils nous permettent chacun à leur façon de calculer la consommation énergétique sur les différents nœuds. Dans le cadre de nos travaux, nous approfondissons les codes pour voir comment les intégrer à l'architecture de notre système et notamment l'introduire au contrôleur Ryu.

III. Interface de monitoring

Afin de mieux visualiser l'état du réseau à tout instant, nous avons fait cette interface de monitoring.

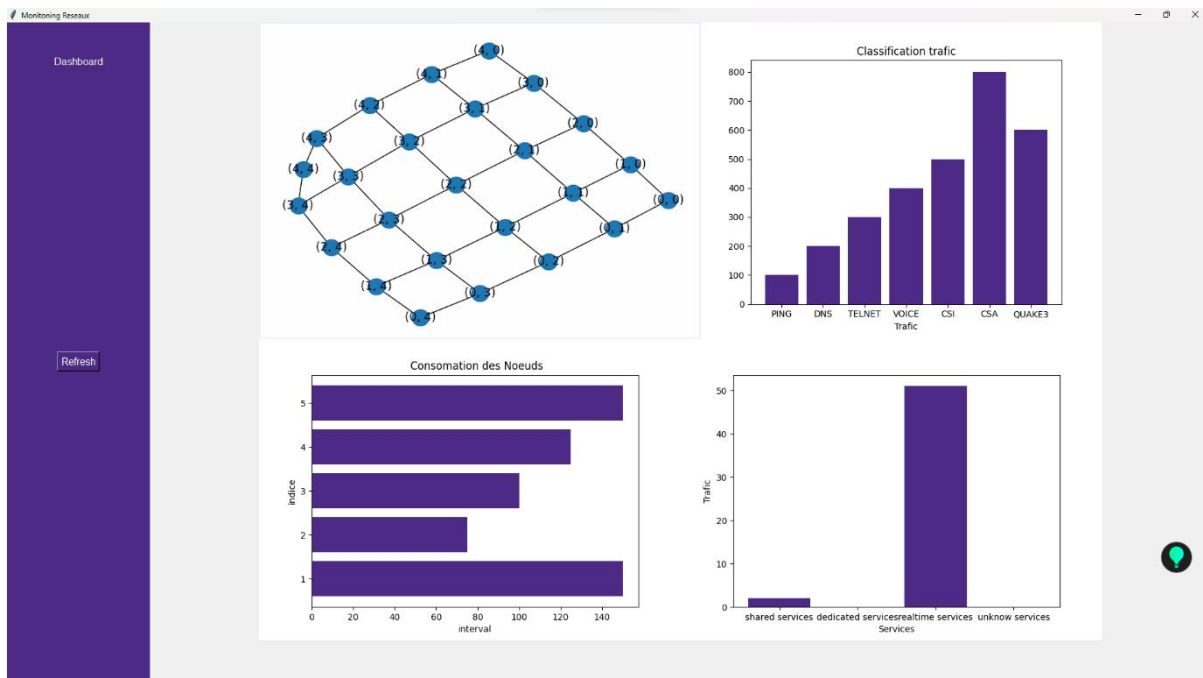


Figure 24 : Présentation de l'interface de monitoring

Le bouton Refresh permet de rafraîchir les graphes. Ceci n'est qu'un prototype.

L'interface de surveillance réseau présentée offre une expérience visuelle riche et diversifiée en combinant des graphiques à barres, à secteurs et à lignes. Sa conception ergonomique intègre des fonctionnalités de rafraîchissement en temps réel, permettant une surveillance constante des performances du réseau avec des données réelles provenant du fichier `data.py`.

A screenshot of a code editor with a dark background and light-colored text. The code is written in Python and defines five functions to generate random data for different network metrics. The functions are: `get_latest_sales_data()`, `get_latest_inventory_data()`, `get_latest_product_data()`, `get_latest_sales_year_data()`, and `get_latest_inventory_month_data()`. Each function uses `random.randint` to generate random values for various products and years. The code is numbered from 1 to 33.

```
1
2 import random
3 import datetime
4
5 def get_latest_sales_data():
6
7     products = ['PING', 'DNS', 'TELNET', 'VOICE', 'CSI', 'CSA', 'QUAKE3' ]
8     sales_data = {product: random.randint(10, 100) for product in products}
9     return sales_data
10
11 def get_latest_inventory_data():
12
13     products = ['1', '2', '3', '4', '5']
14     inventory_data = {product: random.randint(50, 200) for product in products}
15     return inventory_data
16
17 def get_latest_product_data():
18
19     products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']
20     product_data = {product: random.randint(5, 50) for product in products}
21     return product_data
22
23 def get_latest_sales_year_data():
24
25     years = [str(year) for year in range(2020, 2023)]
26     sales_year_data = {year: random.randint(100, 1000) for year in years}
27     return sales_year_data
28
29 def get_latest_inventory_month_data():
30
31     products = ['shared service', 'dedicated services', 'realtime', 'unknow services']
32     inventory_month_data = {product: random.randint(0, 100) for product in products}
33     return inventory_month_data
```

Figure 25 : Vue du code de rafraîchissement

La disposition intuitive des graphiques facilite la comparaison entre différentes métriques de trafic, allant des données aux flux et à la consommation des nœuds. L'utilisation de couleurs vives, d'une image du réseau et de fonctionnalités avancées de Matplotlib contribue à une expérience utilisateur conviviale et informatique, tandis que les boutons de rafraîchissement offrent une interaction directe pour mettre à jour les graphiques en fonction des données les plus récentes, facilitant ainsi la prise de décision rapide.

Conclusion

Nous sommes arrivés au terme de nos travaux. Nous avons pour thème : la simulation de la consommation des nœuds, et la classification du trafic au sein d'un réseau maillé sans fil programmable. Pour ce faire, nous avons commencé par faire une vue globale du concept, ensuite nous avons présenté l'existant et enfin avons présenté nos résultats. Par rapport à ces derniers, il en ressort, commençant par la classification, que les algorithmes de machine Learning tel que : CatBoost et Random Forest, obtiennent de meilleur résultat, et en Deep learning, le perceptron multi-couches de 50 Hidden layer. En ce qui concerne la consommation des nœuds, il en ressort que évalué la consommation d'un équipement réseau peut être plus ou moins complexe. Dans notre cas, nous avons implémenté Pyjoules, contiki, Powertop. Dès lors, nous nous proposons de pousser la réflexion plus loin avec un algorithme de prédiction de la consommation des nœuds en fonction de la nature du trafic entrant et d'autres caractéristiques.

Annexe

I. Annexe 1 : Power Top

Pour ce qui est de la consommation énergétique, On peut se tourner vers un outil permettant d'évaluer cette consommation pour un nœud terminal donné de façon instantanée en l'occurrence Powertop.

En effet, PowerTOP est un outil logiciel open source conçu pour surveiller et optimiser la consommation d'énergie des systèmes Linux. Il fournit des informations détaillées sur la consommation d'énergie des différents composants matériels et logiciels d'un système, permettant aux utilisateurs d'identifier les sources de consommation d'énergie élevée et de prendre des mesures pour améliorer l'efficacité énergétique. Il fonctionne en collectant des données en temps réel sur la consommation d'énergie et en fournissant des estimations et des statistiques détaillées. Il fournit des mesures détaillées sur la consommation d'énergie, les états de veille, les réveils, et d'autres paramètres pertinents. Ces données pourront être utilisées pour une analyse approfondie de la consommation énergétique.

Son équivalent sous Windows est : "**PowerCfg**".

Pour démarrer powertop, on exécute la commande **sudo powertop**. S'il n'est pas préalablement installé, utiliser la commande '**sudo apt install PowerTOP**'.

On pourra, grâce à powertop générer des fichiers de sortie tels que '**powertop.csv**' pour enregistrer les données collectées.

Le fichier '**powertop.csv**' généré par PowerTOP contient plusieurs mesures et champs dont les principaux sont:

- '**Timestamp**' : L'horodatage de la mesure, indiquant le moment où la mesure a été effectuée.
- '**Power**' : La consommation d'énergie actuelle du système, généralement exprimée en Watts (W).
- '**Power est.**' : Une estimation de la consommation d'énergie actuelle du système, basée sur les mesures précédentes et les variations de la charge de travail. Cela fournit une estimation plus stable de la consommation d'énergie.
- '**RAM**' : Les mesures liées à la mémoire RAM du système, telles que la consommation d'énergie, les temps de résidence dans différents états de veille, etc.
- '**Wakeups/s**' : Le nombre de réveils (wakeups) par seconde générés par les différents composants matériels ou logiciels du système. Les réveils sont des événements qui

sortent le système de son état de veille et peuvent augmenter la consommation d'énergie.

- 'Power est. usage' : Une estimation de la consommation d'énergie totale du système, basée sur les estimations individuelles des différents composants.
- 'Wakeups/s est.' : Une estimation du nombre de réveils par seconde générés par les différents composants du système.

Voici en image le lancement et le fonctionnement de Powertop

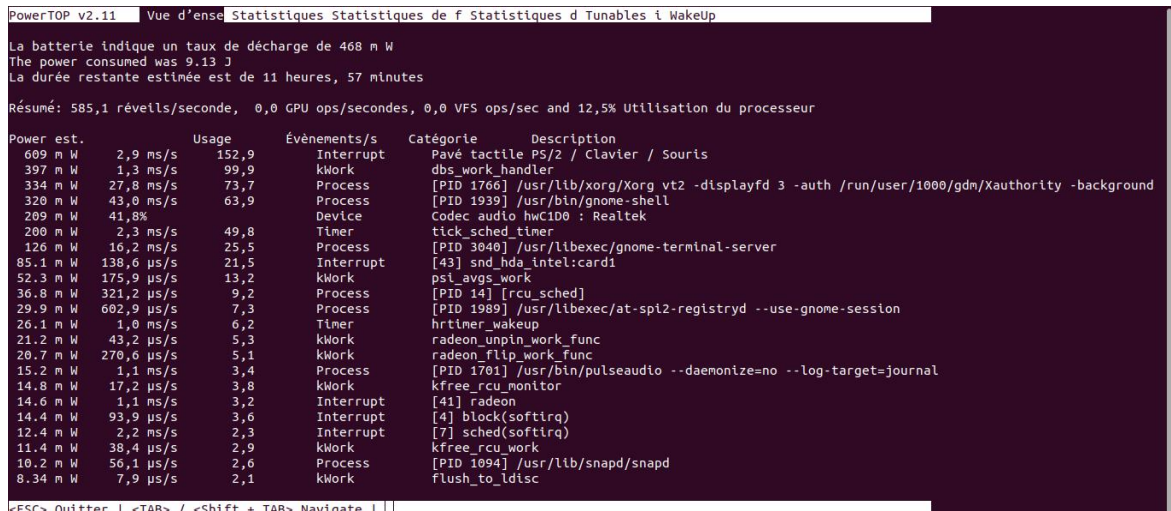


Figure 26 : Vue d'ensemble des statistiques

Vous verrez sur l' image ci-dessus un extrait des informations fournies par Powertop sur la consommation énergétique de l'appareil avec des champs et leur valeurs; on aura ici une estimation de l'énergie consommée en fonction du type d'événements qui se réalise.

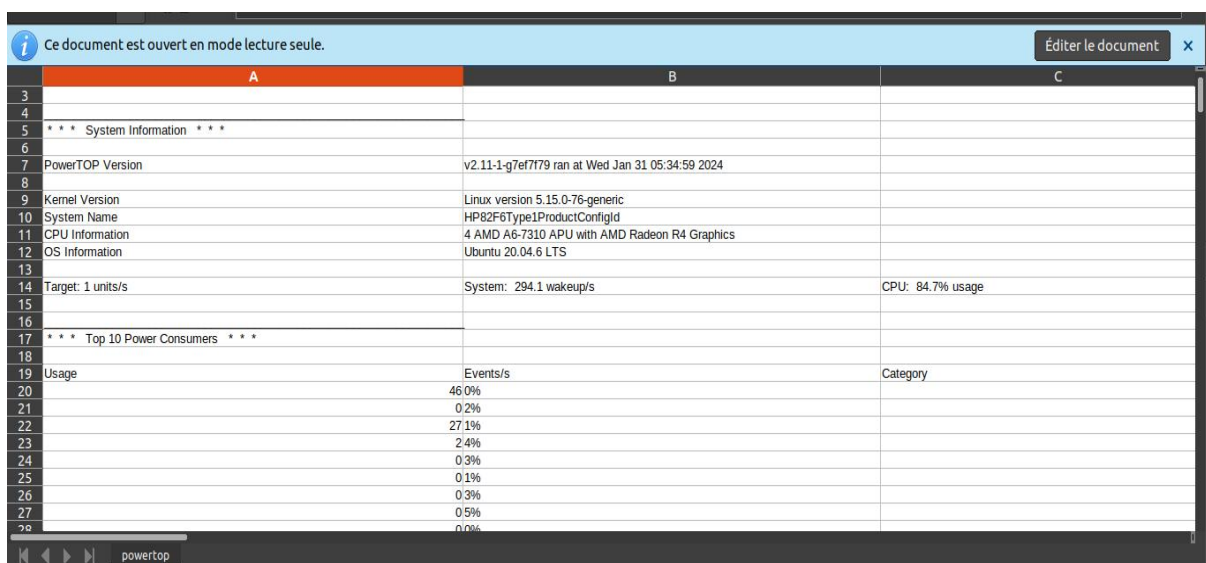


Figure 27 : Vue d'ensemble du Fichier CSV

Dans l'image ci-dessus, on a un extrait du fichier CSV de récupération des données enregistrées par Powertop avec mention de l'horodatage et des informations sur le système qu'on évalue; avec une sélection du top 10 des consommateurs d'énergie.

II. Annexe 2 : Contiki - NG / Cooja

Contiki-NG comprend le module Energest qui peut être utilisé pour mettre en œuvre une approche légère d'estimation de l'énergie basée sur les logiciels pour les appareils IoT à ressources limitées. Le module Energest tient compte du temps passé par un système dans différents états. En utilisant ces informations conjointement avec le modèle de consommation d'énergie matérielle d'un système, le développeur peut estimer la consommation d'énergie du système.

Le module Energest de Contiki-NG permet de suivre cinq états prédéfinis liés à la consommation d'énergie, tels que l'activité du processeur, les modes de faible consommation d'énergie et les opérations de transmission et d'écoute de la radio. Pour activer le service simple-energgest, il suffit d'ajouter une ligne spécifique dans le Makefile de l'application. La plupart des plates-formes Contiki-NG prennent en charge le suivi de ces états, mais il peut y avoir des exceptions en fonction du matériel, comme dans le cas des produits Texas Instruments.

Une fois que le service Simple Energest est activé en définissant la variable MODULES dans le fichier Makefile, il affichera un message de synthèse une fois par minute. Voici un exemple de message sur un nœud Zolertia Z1 émulé :

[INFO: Energest] --- Résumé de la période #2 (60 secondes)

[INFO: Energest] Temps total : 1966080

[INFO: Energest] CPU : 10374/ 1966080 (5 permil)

[INFO: Energest] LPM : 1955706/ 1966080 (994 permil)

[INFO: Energest] Sommeil profond : 0/ 1966080 (0 permil)

[INFO: Energest] Radio Tx : 106/ 1966080 (0 permil)

[INFO: Energest] Radio Rx : 104802/ 1966080 (53 permil)

[INFO: Energest] Total radio : 104908/ 1966080 (53 permil)

Les champs sont :

Résumé de la période #2 (60 secondes) - #2 est le numéro de séquence de la période comptable, c'est-à-dire depuis le dernier rapport. 60 secondes est la durée de la période.

Temps total : 1966080 - le nombre ici fait référence au nombre total de ticks dans la période comptable (60 secondes). Comme la valeur de la constante RTIMER_ARCH_SECOND sur la plate-forme Z1 est égale à 32768, on peut s'attendre à ce que 32768 multiplié par le nombre de secondes. $32768 * 60 = 1966080$.

CPU : 10374/ 1966080 (5 permil) - le premier nombre 10374 ici fait référence au nombre de ticks passés en mode actif du CPU. 5 permil est la proportion approximative du temps passé dans cet état. Il est égal à 10374 divisé par 1966080, sous réserve d'erreurs d'arrondi. 1 permil est égal à 0,1 pour cent.

Estimation de la consommation de charge et de la consommation d'énergie

Supposons que les variables suivantes sont déjà connues :

ticks - le nombre de ticks passés dans un état

RTIMER_ARCH_SECOND - le nombre de ticks par seconde

current_mA - la consommation actuelle dans cet état en mA

tension - la tension fournie par le système au composant (radio ou CPU)

période_sec - la durée de la période de comptabilité en secondes

période_ticks - la durée de la période de comptabilité en ticks

Les métriques suivantes peuvent maintenant être calculées pour l'état pendant la période de mesure :

consommation moyenne de courant (en milliampères, mA)

$$\text{state_avg_current_mA} = (\text{ticks} * \text{current_mA}) / (\text{RTIMER_ARCH_SECOND} * \text{période_sec}) = (\text{ticks} * \text{current_mA}) / \text{période_ticks}$$

consommation de charge (en millicoulombs, mC)

$$\text{state_charge_mC} = (\text{ticks} * \text{current_mA}) / \text{RTIMER_ARCH_SECOND}$$

consommation moyenne de puissance (en milliwatts, mW)

$$\text{state_power_mW} = \text{avg_current_mA} * \text{tension}$$

consommation d'énergie (en millijoules, mJ)

$$\text{state_energy_mJ} = \text{state_charge_mC} * \text{tension}$$

ou, alternativement :

$$\text{state_energy_mJ} = \text{state_power_mW} * \text{période_sec}$$

Les valeurs totales pour l'ensemble du système peuvent être obtenues en additionnant simplement les valeurs de tous les états suivis.

Supposons que vous ayez un fichier COOJA.testlog formaté comme ceci :

Simulation de la consommation des noeuds et Classification du trafic dans un réseau maillé sans fils programmable

```
1107305 2 [INFO: Main      ] Starting Contiki-NG-release/v4.5-149-g1c0b472-dirty
1110081 2 [INFO: Main      ] - Routing: RPL Lite
1112715 2 [INFO: Main      ] - Net: sicslowpan
1114921 2 [INFO: Main      ] - MAC: TSCH
1118219 2 [INFO: Main      ] - 802.15.4 PANID: 0x81a5
1123248 2 [INFO: Main      ] - 802.15.4 TSCH default hopping sequence length: 4
1125408 2 [INFO: Main      ] Node ID: 2
```

Figure 28 : Aperçu d'un fichier COOJA.testlog

On a donc un aperçu du code de lecture et des calcul des consommations

Simulation de la consommation des noeuds et Classification du trafic dans un réseau maillé sans fils programmable

```
INPUT_FILE = "COOJA.testlog"
# From Z1 node datasheetCURRENT_MA = {
    "CPU" : 10,
    "LPM" : 0.023,
    "Deep LPM" : 0, # not used by Z1 nodes
    "Radio Rx" : 18.8,
    "Radio Tx" : 17.4,}
STATES = list(CURRENT_MA.keys())
VOLTAGE = 3.0 # assume 3 volt batteriesRTIMER_ARCH_SECOND = 32768
def main():
    node_ticks = {}
    node_total_ticks = {}
    with open(INPUT_FILE, "r") as f:
        for line in f:
            if "INFO: Energgest" not in line:
                continue
            fields = line.split()
            try:
                node = int(fields[1])
            except:
                continue
            if node not in node_ticks:
                # initialize to zero
                node_ticks[node] = { u : 0 for u in STATES }
                node_total_ticks[node] = 0
            try:
                state_index = 5
                state = fields[state_index]
                tick_index = state_index + 2
                if state not in STATES:
                    state = fields[state_index] + " " + fields[state_index+1]
                    tick_index += 1
                    if state not in STATES:
                        # add to the total time
                        if state == "Total time":
                            node_total_ticks[node] += int(fields[tick_index])
                        continue
                    # add to the time spent in specific state
                    ticks = int(fields[tick_index][-1])
                    node_ticks[node][state] += ticks
            except Exception as ex:
                print("Failed to process line '{}': {}".format(line, ex))

    nodes = sorted(node_ticks.keys())
    for node in nodes:
        total_avg_current_mA = 0
        period_ticks = node_total_ticks[node]
        period_seconds = period_ticks / RTIMER_ARCH_SECOND
        for state in STATES:
            ticks = node_ticks[node].get(state, 0)
            current_mA = CURRENT_MA[state]
            state_avg_current_mA = ticks * current_mA / period_ticks
            total_avg_current_mA += state_avg_current_mA
        total_charge_mC = period_ticks * total_avg_current_mA / RTIMER_ARCH_SECOND
        total_energy_mJ = total_charge_mC * VOLTAGE
        print("Node {}: {:.2f} mC ( {:.3f} mAh) charge consumption, {:.2f} mJ energy consumption in {:.2f} seconds".format(
            node, total_charge_mC, total_charge_mC / 3600.0, total_energy_mJ, period_seconds))
if __name__ == "__main__":
    main()
```

Le résultat de compilation est :

```
Node 1: 3917.93 mC (1.088 mAh) charge consumption, 11753.79 mJ energy consumption in 3599.99 seconds
Node 2: 5699.00 mC (1.583 mAh) charge consumption, 17097.01 mJ energy consumption in 3599.99 seconds
```

Figure 29 : Résultats de compilation

References

- [1] : Rapport de master, université de Yaoundé 1, Mr André Bessala ; thème : classification intelligente du trafic dans un réseau maillé sans fil programmable contraint.
- [2] : Evaluation of énergie efficiency in mobile cellular networks using a fluid modeling framework Thèse de doctorat de l'université Paris-Saclay
- [3] : BENRAZEK HAKIM Sujet CLASSIFICATION DU TRAFIC INTERNET A L'AIDE DE L'APPRENTISSAGE AUTOMATIQUE
- [4] : SEBASTIAN ORLOWSKI MICHAL PIORO ARTUR TOMASZEWSKI ROLAND WESSALY SNDlib 1.0–Survivable Network Design Library
- [5] Publication de recherche de Feifei Hu¹ , Situo Zhang¹ , Xubin Lin¹ , Liu Wu¹ , Niandong Liao^{2*} and Yanqi Song sur la thématique : Network traffic classification model based on attention mechanism and spatiotemporal features
- [6] [Energy Consumption in Wireless Sensor Network \(researchgate.net\)](#): EnergyConsumptionin Wireless Sensor Network, John T. Ogbiti, Henry C. Ukwuoma, Salome Danjuma, and Mohammed Ibrahim
- [7] contiki documentation <https://docs.Contiki-NG.org/>
- [8] cpu energy Meter : <https://github.com/sosy-lab/CPU-energy-meter/tree/main>
- [9] site web regroupant différents outils open source d'estimation de la consommation énergétique des équipements. lien : <https://sustainableit-tools.isit-europe.org/?hashtag=86>
- [10] powertop <https://doc.ubuntu-fr.org/powertop#powertop>
- [11] <https://sustainableit-tools.isit-europe.org/?hashtag=86>

