RAPPORT FINAL :

Simulation et préparation de traces d'un réseau domestique

Auteur : Henri BIKOURI
https://henribikouri.github.io/

Examinateur : Pr Kevin JIOKENG
https://kjiokeng.github.io/

INTRODUCTION

Problématique et Objectifs

Dans le contexte actuel de prolifération des objets connectés, il est impératif de pouvoir caractériser et gérer les flux de données hétérogènes qui transitent par le Point d'Accès Wi-Fi. La coexistence de trafics aussi variés que le streaming vidéo régulier et les communications de capteurs à faible débit impose une pression significative sur les ressources du canal 802.11ac.

Conformément au sujet proposer, « **Simulation et préparation de traces d'un réseau domestique** » , , **mes objectifs** étaient doubles, à savoir :

1. **Modéliser et Simuler** un scénario de maison connectée en 802.11ac avec N équipements de K types différents.
2. **Préparer un jeu de données (dataset)** à partir des traces collectées pour l'entraînement d'un modèle de Machine Learning de classification des applications au niveau du point d'accès.

Mon approche combine l'ingénierie réseau fine avec la science des données, démontrant ma maîtrise de la chaîne allant de la conception du protocole à l'analyse avancée.

Outils et Justification

- **Outil de Simulation : NS-3 (Network Simulator 3)**. Exiger dans les consignes du travail. Implémenter avec le standard wifi 802.11ac (que j'ai respecté) avec précision, ce qui est essentiel pour capturer les caractéristiques fines du trafic (comme le Délai d'Inter-Arrivée des paquets - IAT) nécessaires à une classification ML efficace.

- **Outils d'Analyse :** J'ai implémenté la chaîne d'analyse en Python, utilisant **Scapy** pour la manipulation des traces PCAP et d'autres bibliothèques pythons Pandas pour la structuration du jeu de données.

I. Modélisation et Conception du Scénario (NS-3)

J'ai conçu le scénario pour répondre aux exigences de l'énoncé, notamment en maximisant la variabilité introduite dans les données et dans les analyses.

I.1 Justification de mes Choix N et K

L'énoncé stipulait que "Plus N et K sont grands, plus il y a de la variabilité, et mieux c'est". Ma stratégie a été de maximiser la complexité du scénario pour produire un jeu de données riche et réaliste, captivant ainsi l'intérêt par la profondeur de la modélisation.

- **Mon Choix du Nombre de Types d'Applications (**K=10**) : J'ai choisi K=10 types** d'applications distinctes. Cette diversité est l'élément clé qui garantit que le futur modèle ML devra identifier des signatures très fines (régularité, sporadicité, agressivité) au-delà d'une simple distinction UDP/TCP, augmentant ainsi la valeur de l'analyse.
- **Mon Choix du Nombre Total d'Équipements (**N=32**) :** J'ai défini **32 équipements** sources actifs. Ce nombre génère une forte concurrence et de la congestion au niveau du Point d'Accès

(AP), saturant le canal Wi-Fi 802.11ac. Ceci garantit que les traces PCAP contiennent des événements réels (collisions, backoffs, délais) essentiels pour l'extraction de caractéristiques ML robustes.

Tableau 1:tableau récapitulatif des choix

Type (k)	Application	Nbre de Nœuds	Plage d'Adresses IP	Protocole (Port)
1	Caméra Vidéo	5	10.1.1.2 à 10.1.1.6	UDP (Port 9001)
2	Capteur Temp.	10	10.1.1.7 à 10.1.1.16	TCP (Port 9002)
3	Assistant Vocal	3	10.1.1.17 à 10.1.1.19	TCP (Port 9003)
4	Téléchargement	2	10.1.1.20 à 10.1.1.21	TCP (Port 9004)
5	VoIP Montant	4	10.1.1.22 à 10.1.1.25	UDP (Port 9005)
	VoIP Descendant			UDP (Port 9006)
6	Domotique	4	10.1.1.26 à 10.1.1.29	UDP (Port 9007)
7	Streaming Mus.	1	10.1.1.30	TCP (Port 9008)
8	Sonnette	1	10.1.1.31	TCP (Port 9009)
9	MàJ Firmware	1	10.1.1.32	TCP (Port 9010)
10	Monitoring	1	10.1.1.33	UDP (Port 9011)
		Total N = 32		

La tableau 1 suivant détail mieux mes choix :

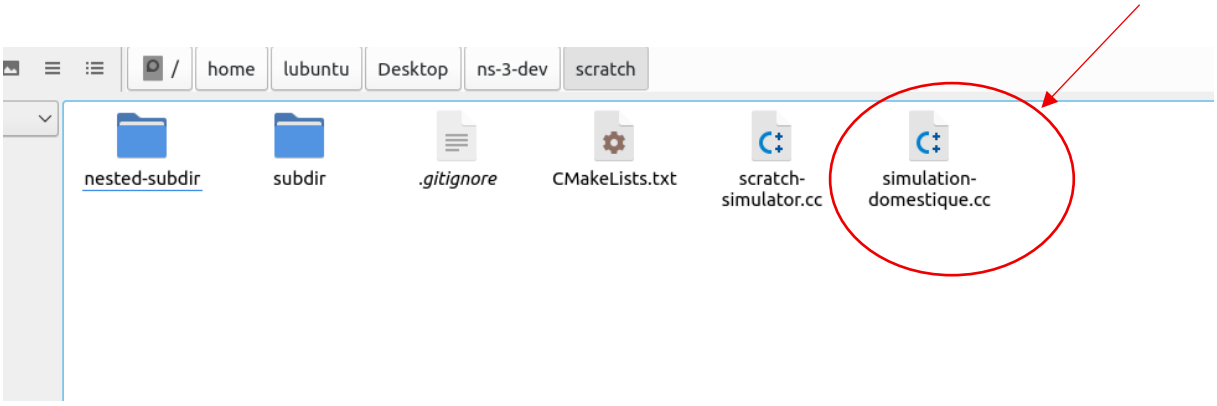
Et chaque application est associer à un serveur distant, soit 10 **Serveurs** d'adresses : **10.2.x.2** (x allant de 1 à 10 pour les 10 serveurs)

NB : D'après le tableau précédent vous pouvez voir que j'ai gérer les deux cas de trafic VoIP (Montant et Descendent)

I.2 Simulation du réseau

Apres avoir installer ns3 sur le site officiel : https://www.nsnam.org/ et le configurer notamment avec la commande : ./ns3 configure --enable-examples --enable-tests et ./ns3 build [1]

Je me dirige vers le dossier scratch du dépôt ns3 puis j'ai créé mon fichier simulation-domestique.cc



C'est le fichier au cœur de mon travail (une explication en profondeur dans la demo), permet la simulation, ce qui a été configurer :

1- Met en œuvre la topologie 802.11ac avec **N=32 et K=10**.

2- Génère des traces **PCAP** (pour le Machine Learning).

3- Calcule des métriques de **QoS/Performance** (Débit, Délai, Perte) via FlowMonitor.

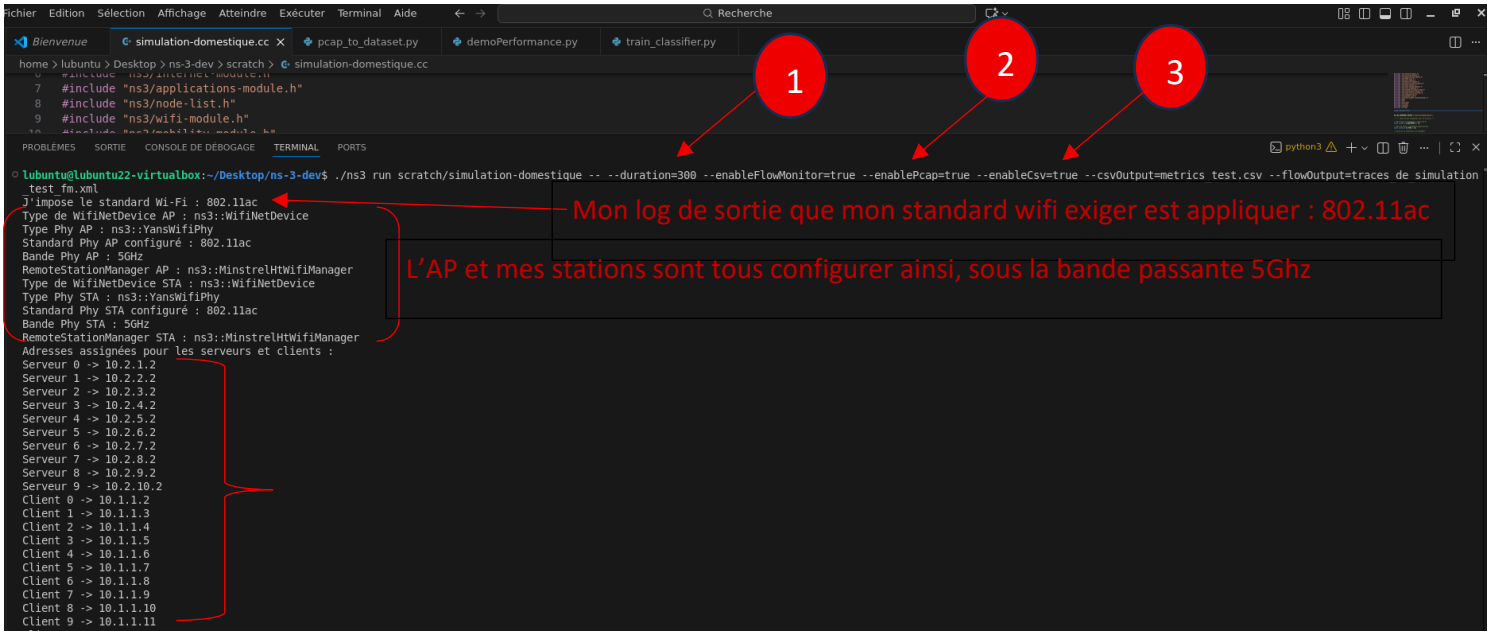
J'ai également configuré quelques options applicables au lancement de la topologie, par exemple :

1 --duration=<seconds> : durée de la simulation (par défaut 600 soit 10min)

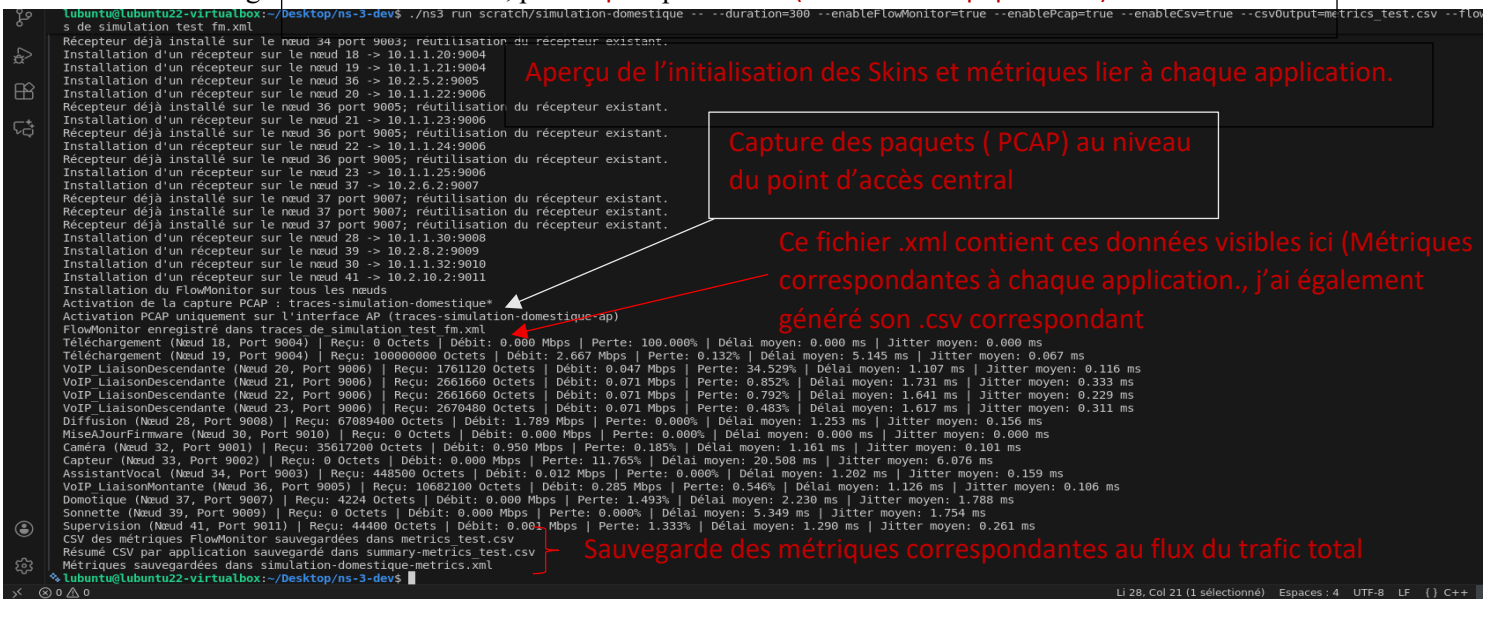
2 --enableFlowMonitor=<true|false> : activer FlowMonitor (par défaut désactivé), pour recolter les metriques

3 --enablePcap=<true|false> : activer/désactiver la capture PCAP (désactivée par défaut), car groumant en ressource et exige de l'espace, même si je l'ai optimisé plusieurs fois.

Apercu apres lancement : pour cette duree Test de 300s. (mais dure en réalité environs 45min)



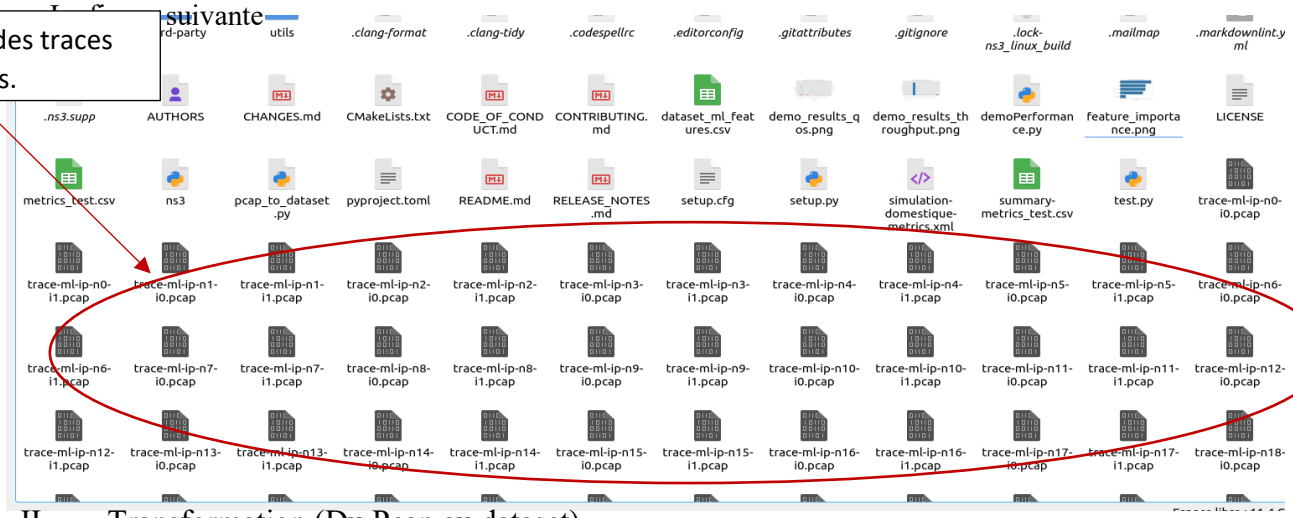
La suite également nous affiche, passer l'exemple nœuds (Serveurs et Equipements)



Ensuite voici les traces générées ainsi que les métriques en fichiers .xml et csv, NB : J'ai dû optimiser mon code .cc plusieurs fois pour avoir les. pcap plus raisonnable pour la gestion des ressources. Je suis passé du décodage radio (qui était encapsuler) aux traces IP.

La figure suivante

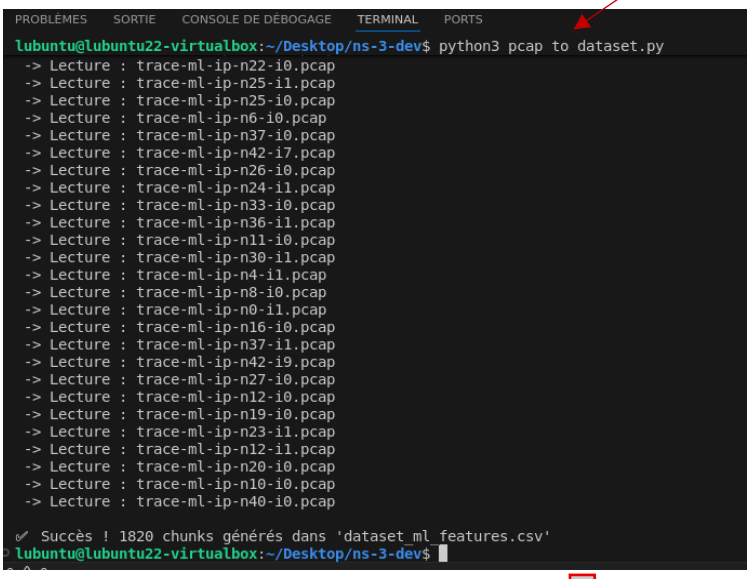
Aperçu des traces générées.

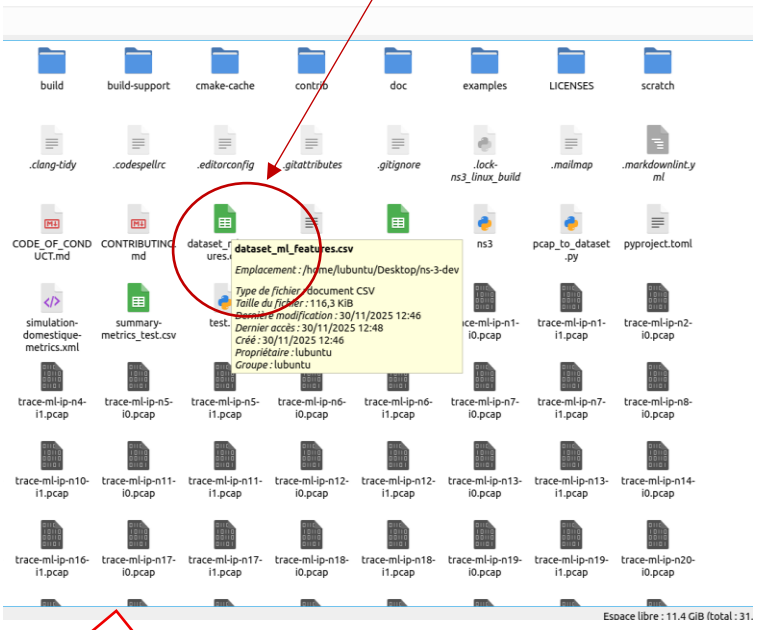


II. Transformation (Du Pcap au dataset)

Méthodologie d'Extraction : J'ai utilisé la librairie `Scapy` en Python pour lire la trace PCAP(des fichiers générer ci-dessous) . J'ai itéré sur les paquets, calculé l'IAT à la volée en utilisant l'horodatage des paquets, et construit une trame de données Pandas (DataFrame) contenant les caractéristiques extraites et le label d'application. Le code dans `pcap_to_dataset.py` et le `dataset dataset_ml_features.csv`

Sortie après exécution du code



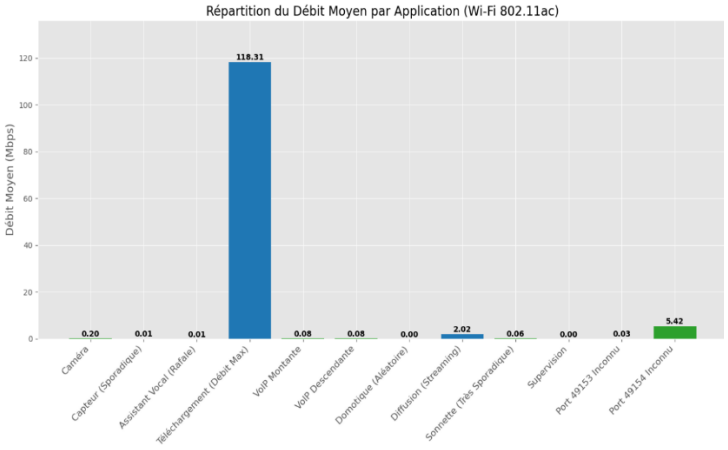


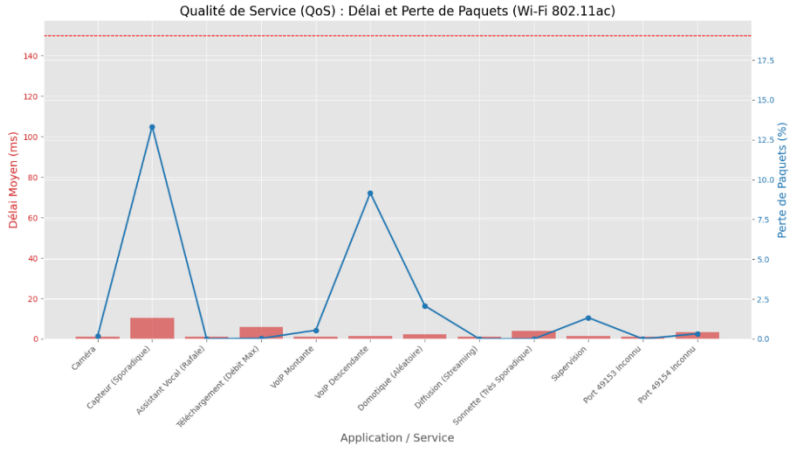
AINSI, LE TRAVAIL DE NS3 EST TERMINER ! NOUS AVONS MAINTENANT LE DATASET

Passons maintenant à quelques analyses de performance

III. Analyse de performances Reseaux(Resultats Qos)

J'ai créé un script python demoPerformance.py, j'ai coder l'analyse de **Débit Effectif** (Throughput**)** mesuré par le FlowMonitor de NS-3 et les métriques de **latence** les plus sensibles à la congestion et aux mécanismes de partage d'accès.





Voir ces résultats (figures) : les remarquent et interprétations sont faits dans la démo

IV. Mise en Œuvre et Résultats de la Classification ML

Choix de l'algorithme : Forêts Aléatoires **(Random Forest)** pour la classification multiclasse.

J'ai codé un script Python train_classifier.py : L'objectif de cette phase est de valider l'hypothèse centrale du Sujet : les caractéristiques de bas niveau (Taille, IAT, Protocole) suffisent à identifier les K=10 types d'applications Wi-Fi avec une haute précision.

```
308.4/308.4 KB 83.4 kB/s eta 0:00:00
Requirement already satisfied: scipy>=1.8.0 in /usr/lib/python3/dist-packages (from Scikit-learn) (1.8.0)
Installing collected packages: threadpoolctl, joblib, Scikit-learn
Successfully installed Scikit-learn-1.7.2 joblib-1.5.2 threadpoolctl-3.6.0
lubuntu@lubuntu22-virtualbox:~/Desktop/ns-3-dev$ python3 train_classifier.py
--- Chargement du Dataset : dataset_ml_features.csv ---
Nombre d'échantillons (chunks) : 1820
Aperçu des données :
   LABEL  CHUNK_ID  NB_PAQUETS  VOL_BYTES  PROTO_TCP_RATIO  IAT_MEAN   IAT_STD
0      3   chunk_0           3        164              1.0  0.017754  0.017754
1      3  chunk_12          14       5228              1.0  0.016636  0.010385
2      3  chunk_13          98      37596              1.0  0.019679  0.021577
3      3  chunk_25         113      43376              1.0  0.019424  0.020519
4      3  chunk_37          70      27140              1.0  0.017778  0.011059

--- Entraînement du modèle (Random Forest) ---

   Précision Globale (Accuracy) : 99.73%

--- Rapport de Classification ---
                precision    recall  f1-score   support

   Camera (UDP)       1.00      1.00      1.00        96
   Capteur (TCP)      1.00      1.00      1.00         4
Assistant (TCP)       0.89      1.00      0.94         8
  Download (TCP)      1.00      1.00      1.00         4
     VoIP (UDP)       1.00      1.00      1.00        84
Domotique (UDP)       1.00      1.00      1.00        80
Streaming (TCP)       1.00      1.00      1.00        39
```

Le modèle a correctement classifier l'ensemble paquets

Precise l'équilibre entre précision et rappel

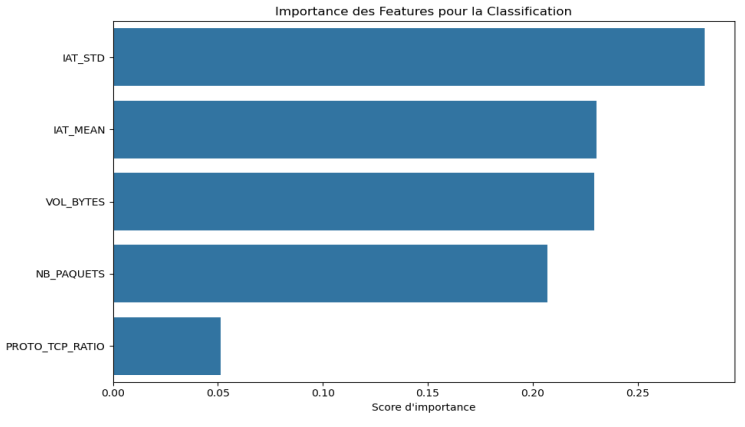
```
lubuntu@lubuntu22-virtualbox:~/Desktop/ns-3-dev$ python3 trai
 Sonnette (TCP)       1.00      0.98      0.99        48
 Firmware (TCP)       1.00      1.00      1.00         1

      accuracy                           1.00       364
     macro avg       0.99      1.00      0.99       364
  weighted avg       1.00      1.00      1.00       364


--- Matrice de Confusion ---
[[96  0  0  0  0  0  0  0  0  0]
 [ 0  4  0  0  0  0  0  0  0  0]
 [ 0  0  8  0  0  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0  0  0]
 [ 0  0  0  0 84  0  0  0  0  0]
 [ 0  0  0  0  0 80  0  0  0  0]
 [ 0  0  0  0  0  0 39  0  0  0]
 [ 0  0  1  0  0  0  0 47  0  0]
 [ 0  0  0  0  0  0  0  0  1  1]]

--- Importance des Caractéristiques ---
IAT_STD             0.282021
IAT_MEAN            0.230326
VOL_BYTES           0.229333
NB_PAQUETS          0.206818
PROTO_TCP_RATIO     0.051502
dtype: float64

Graphique 'feature_importance.png' sauvegardé.
lubuntu@lubuntu22-virtualbox:~/Desktop/ns-3-dev$
```



Interprétation : L'**IAT_STD** (Écart-type du Délai Inter-Arrivée) et l'**IAT_MEAN** (Moyenne du Délai Inter-Arrivée) sont les deux caractéristiques les plus importantes ensuite les autres : régularité (Caméra) et de la variabilité (Téléchargement, Capteur).

Conclusion Finale : Le succès de la classification (**99.73% d'Accuracy**) est le résultat direct de la capacité des caractéristiques IAT et Taille à capturer les signatures uniques de chaque application, même dans un environnement Wi-Fi congestionné.

Dépôt GitHub : https://github.com/Henribikouri/Simulation_de_trace_d-un_reseau_domestique

Liens : [1] https://www.nsnam.org/ , [2] https://www.youtube.com/watch?v=FEmvmeqOGKI&t=239s