POLYTECHNIQUE MONTRÉAL

WORLD-CLASS ENGINEERING

Polytechnique Montréal

ELE8704 : Transmission de données et réseaux internet

Projet De Session

Réalisé par :
AFIF Yassine **2403652**
BOXUS Florent **2411548**
LOPEZ Ines **2404168**

Professeur :
Brunilde Sanso

Année académique : 2024-2025

Table des matières

Table des figures

Glossaire

MTU Maximum Transmission Unit. 8

1 Introduction

Dans un contexte où les réseaux de télécommunications doivent répondre à une demande croissante en données massives et en temps réel, ce projet explore et analyse le trafic IP généré par trois activités : regarder une vidéo YouTube, passer un appel vidéo sur Teams et télécharger une bibliothèque Lapack. En étudiant les processus d'arrivée des paquets, leur taille et la gigue, l'objectif est de mieux comprendre les caractéristiques du trafic réseau. Des modèles d'apprentissage machine seront également développés pour classifier les paquets selon leur type ou protocole. Ce projet mêle théorie et pratique pour contribuer à l'optimisation des réseaux modernes.

2 Expériences

Les expériences ont été réalisées dans des conditions rigoureuses, en utilisant le même routeur, ordinateur et navigateur, avec des captures sur *Wireshark* via Wifi et Ethernet. Aucun filtre n'a été appliqué pendant la capture, le filtrage étant effectué après pour plus de flexibilité.

Les fichiers ont été sauvegardés aux formats `pcap` et `csv`, permettant une analyse en *dataframes* sous Python tout en conservant toutes les informations des paquets. Pour éviter le trafic non destiné à notre machine, seuls les paquets contenant les adresses IP de notre PC (`192.168.2.35` pour le Wifi et `192.168.2.68` pour l'Ethernet) ont été conservés. La commande `./whois` a été utilisée pour analyser la provenance des paquets ([2]).

2.1 Capture des paquets

2.1.1 Trafic Vidéo

Pour cette expérience, nous avons capturé le trafic pendant que notre ordinateur lisait une vidéo youtube. Nous avons commencé la vidéo aux mêmes instants pour le wifi et ethernet et terminé aux mêmes instants. Nous avons utilisé chrome pour cette expérience.

2.1.2 Dialogue

Pour cette expérience, nous avons écrit un dialogue et nous sommes placés dans des pièces différentes. Les deux personnes impliquées dans le dialogue étaient les mêmes et nous avons utilisé Zoom dans les deux expériences. Nous avons essayé de reproduire les mêmes intonations dans les deux expériences pour changer le contenu des paquets le moins possible.

2.1.3 Téléchargement

La difficulté pour cette partie est que nous avons du être réactifs car le téléchargement était assez rapide (de l'ordre de la seconde) et nous avons donc essayé de commencer celui-ci le plus vite possible après le début de la capture et arrêté la capture le plus vite possible à la fin du téléchargement. Nous avons utilisé FireFox comme navigateur pour ces expériences.

2.2 Protocoles Rencontrés

Cette section vise à expliquer brièvement les différents protocoles rencontrés au cours des différentes expériences. En effet, nous n'avons jamais rencontré une grande partie d'entre eux et il est fondamental de comprendre leur rôle dans le transfert de données afin de réaliser des analyses pertinentes. Wireshark permet d'avoir le protocole correspondant au protocole de plus haut niveau dans le modèle en couche trouvé dans l'en-tête du paquet.

On peut expliciter ça avec un exemple, si le paquet a les protocoles suivants affichés dans la structure : [Protocols in frame : eth :ethertype :ip :tcp], ce sera TCP qui sera affiché dans la colonne protocole "Protocole" ([11]).

2.2.1 Trafic Vidéo

Au cours de cette expérience, l'immense majorité des paquets capturés sont des paquets de protocole **UDP** ou **Quic**

- **UDP** :Le protocole UDP (User Datagram Protocol) est un protocole de transport léger, rapide et sans connexion. Contrairement à TCP, il ne garantit ni la livraison, ni l'ordre des paquets, ni la correction des erreurs. UDP est souvent utilisé pour les applications en temps réel comme le streaming vidéo, les jeux en ligne et la VoIP, où la vitesse prime sur la fiabilité [7].
- Le protocole **QUIC** (Quick UDP Internet Connections) est un protocole de transport moderne conçu pour remplacer TCP, optimisé pour la latence et la sécurité. Il fonctionne au-dessus d'UDP et intègre des fonctionnalités comme le chiffrement (par défaut), le multiplexage de flux (similaire à HTTP/2), et la gestion rapide des connexions. Il est utilisé notamment pour HTTP/3, assurant des performances accrues sur les réseaux instables comme le Wi-Fi ou les mobiles. Il présente un mécanisme d'Acknowledgement même si il est basé sur UDP pour augmenter la fiabilité de la connexion [4].

On voit en analysant le trafic que l'on reçoit majoritairement du trafic UDP ou QUIC qui arrive "en rafale", à la suite de ces paquets reçus on a un message de controle de protocole UDP ou Quic qui est renvoyés à la source. Cela peut être pour QUIC un acknowledgement ou une demande de transfert particulière pour une partie de la vidéo. Le même phénomène s'observe avec UDP, alors que celui-ci n'est pas sensé envoyé de message d'acknowledgement ou de demande de retransmission. cela peut s'expliquer par le fait qu'un protocole d'application qui tourne sur ces paquets UDP puisse le faire.

2.2.2 Téléchargement

Au cours de cette expérience, l'immense majorité des paquets capturés proviennent des protocoles **TCP**, **TLSv1.2**, et **TLSv1.3**, qui sont souvent utilisés pour garantir la fiabilité, la sécurité et la gestion des connexions.

- **TCP (Transmission Control Protocol)** : Le protocole TCP est un protocole de transport fiable, orienté connexion, garantissant l'ordre des paquets, leur livraison et la correction d'éventuelles erreurs. Il est largement utilisé pour des applications où la fiabilité prime, comme le transfert de fichiers. TCP fonctionne en établissant une connexion entre le client et le serveur avant l'échange des données, en utilisant un mécanisme de *handshake* et de gestion des fenêtres de séquence. [3]
- **TLSv1.2 et TLSv1.3 (Transport Layer Security)** : Ces protocoles sont des protocoles cryptographiques utilisés pour sécuriser les échanges de données sur Internet. TLSv1.2 et TLSv1.3 assurent la confidentialité, l'intégrité et l'authentification des communications, notamment dans des applications telles que le HTTPS. TLSv1.3 est plus performant et sécurisé, en réduisant le nombre de tours de négociation lors de l'établissement de la connexion par rapport à TLSv1.2. Les paquets de données chiffrées transportent les informations sensibles tandis que des paquets de contrôle assurent la négociation des clés et la gestion des sessions. [8]

2.3 Conversation Teams

Dans le cadre de cette expérience, nous avons principalement observé des paquets issus des protocoles **STUN**, **UDP**, **RTCP** et **STP**, qui sont fréquemment utilisés dans des applications nécessitant des échanges rapides et efficaces, souvent en temps réel, et sans garantie de fiabilité pour certains d'entre eux.

- **STUN (Session Traversal Utilities for NAT)** : Le protocole STUN permet à une application de découvrir son adresse IP publique et de déterminer la manière dont elle peut traverser les dispositifs NAT (Network Address Translation). Ce protocole est particulièrement utilisé dans les applications de communication en temps réel, telles que la VoIP ou la vidéoconférence, pour résoudre les problèmes de connectivité associés aux routeurs et pare-feu, facilitant ainsi la mise en place de connexions directes entre les clients. Ce dernier opère sur la couche Application.[1]
- **STP (Spanning Tree Protocol)** : Le protocole STP est essentiel pour éviter les boucles de réseau dans les infrastructures de réseaux locaux (LAN). Il permet de déterminer un chemin de communication sans boucle, assurant ainsi la stabilité et l'efficacité des réseaux de commutateurs. STP désactive les chemins redondants pour garantir que les données suivent un chemin unique, empêchant ainsi les problèmes de congestion ou de répétition des paquets.[5]
- **UDP (User Datagram Protocol)** : Voir 2.2.1
- **RTCP (Real-time Transport Control Protocol)** : RTCP complète le protocole RTP (Real-time Transport Protocol) pour fournir un contrôle de la qualité de service dans les applications de communication en temps réel. Il permet de collecter des informations sur les performances du réseau, comme la perte de paquets et les délais de transmission, et aide à ajuster la qualité des flux multimédia en fonction des conditions du réseau. RTCP est essentiel pour maintenir une expérience utilisateur optimale dans les applications sensibles au temps.[6]

3 Partie Statistiques

3.1 Analyse des temps d'arrivée

Pour cette partie, il nous est demandé d'analyser les temps d'arrivée des paquets de données. On choisit d'abord de conserver les paquets de données.

3.1.1 Analyse des paquets de la vidéo Youtube

On représente sur les histogrammes ci-dessous les différents temps d'arrivée des paquets.

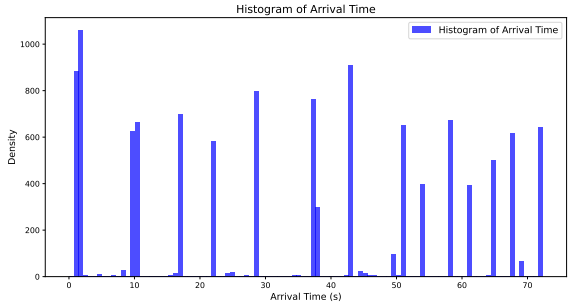


FIGURE 1 – Histogramme des temps d'arrivée pour la vidéo wifi

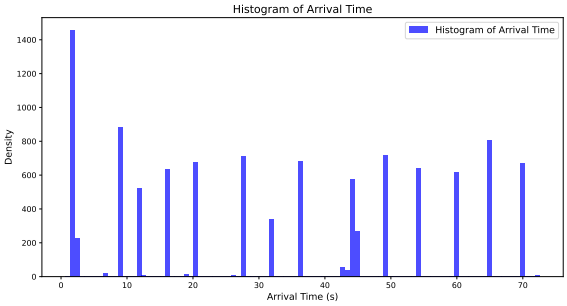


FIGURE 2 – Histogramme des temps d'arrivée pour la vidéo ethernet

On va ensuite donner les statistiques principales des temps d'arrivées :

	Wifi	**Ethernet**
Premier Moment (s)	34.3808	32.6779
Deuxième Moment (s^2)	1748.5529	1582.1908
Variance (s^2)	566.5568	514.3889
Minimum (s)	0.000	0.6927
Maximum (s)	72.3479	72.5347
Médiane (s)	37.534625	32.172

TABLE 1 – Statistiques des temps d'arrivées

On se rend facilement compte qu'il n'est pas possible d'ajuster de distribution connue à ce trafic en rafale. On peut cependant essayer de comprendre pourquoi le trafic a cette forme. Cela peut s'expliquer par la mise en cache des paquets. Ce système de tampon permet une lecture plus fluide de la vidéo et une meilleure qualité de service. Noter que la vidéo capturée dure plus d'une heure, donc tout au long de la capture, on reçoit des paquets pour la lecture de la vidéo y compris après notre extrait.

3.1.2 Analyse des paquets du téléchargement

Pour garder les paquets de données, on garde uniquement ici les paquets des protocoles TLSv1.3 (Ethernet), TLSv1.2 (Wifi) et TCP arrivant à notre ordinateur.

On commence avec wifi comme protocole de liaison, les histogrammes représentatifs des temps d'inter-arrivée sont données ci-dessous.

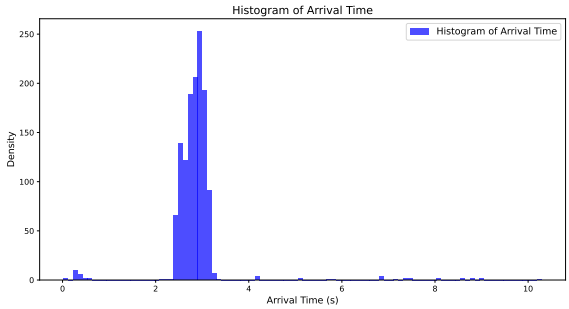


FIGURE 3 – Histogramme des temps d'arrivée pour le téléchargement en wifi

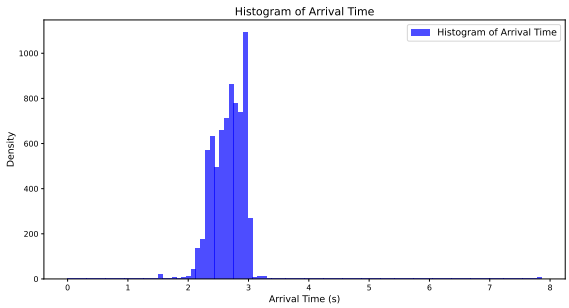


FIGURE 4 – Histogramme des temps d'arrivée pour le téléchargement en Ethernet

On va ensuite donner les statistiques principales des temps d'arrivées :

	Wifi	**Ethernet**
Premier Moment (s)	2.8751	2.6646
Deuxième Moment (s^2)	8.8086	7.2254
Variance (s^2)	0.5424	0.1249
Minimum (s)	0.021	0.000
Maximum (s)	10.291	7.858
Médiane (s)	2.8612	2.692

TABLE 2 – Statistiques des temps d'arrivées

On remarque rapidement a quel moment le téléchargement a démarré et a terminé avec

l'allure en 'pic' autour de 3 sec. On observe bien une augmentation de trafic soudaine due au téléchargement, que ce soit en Wifi ou en Ethernet.

3.1.3 Analyse des paquets de la Conversation Teams

On représente sur les histogrammes ci-dessous les différents temps d'arrivée des paquets.

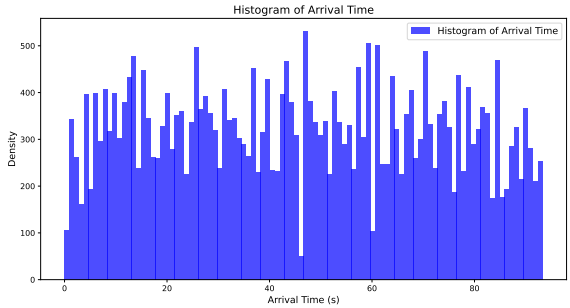


FIGURE 5 – Histogramme des temps d'arrivée pour la conversation teams wifi

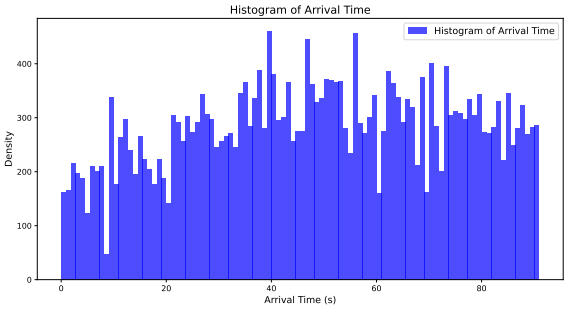


FIGURE 6 – Histogramme des temps d'arrivée pour la conversation teams ethernet

On va ensuite, comme pour les deux sous-parties du dessus, donner les statistiques principales des temps d'arrivées :

	Wifi	**Ethernet**
Premier Moment (s)	46.0123	48.3193
Deuxième Moment (s^2)	2806.8950	2944.7059
Variance (s^2)	689.7814	609.9677
Minimum (s)	0.005	0.0000
Maximum (s)	93.2973	90.9838
Médiane (s)	45.5631	48.8246

TABLE 3 – Statistiques des temps d'arrivées

Il est évident qu'aucune distribution connue ne peut être ajustée à ce trafic en rafale. Toutefois, il est intéressant de chercher à comprendre les raisons derrière cette forme particulière de trafic. On voit ici une distribution plutôt homogène des temps d'arrivée pour les deux types de communications possibles, ce qui semble cohérent compte tenu de la situation. En effet, comme nous sommes sur une conversation Teams, les paquets ont besoin d'arriver à l'ordinateur en continu, car l'information est reçue et envoyée en temps réel, ce qui permet de comprendre les différences avec les deux situations précédentes.

3.2 Analyse de la taille des paquets

Dans cette partie, nous devons analyser conformément aux consignes la taille de **tous** les paquets qui partent ou arrivent à notre pc. Nous filtrons donc les paquets ayant pour source ou pour destination notre adresse Ip dans les expériences suivantes. On note qu'on traite les temps d'inter-arrivée comme des paquets différents apès une analyse de la trace, car les numéros d'offset sont tous les mêmes,0, indiquant que ce sont des paquets différents.

3.2.1 Analyse du trafic vidéo

On représente sur les histogrammes suivants la longueur des paquets avec le protocole de liaison wifi et ethernet respectivement.

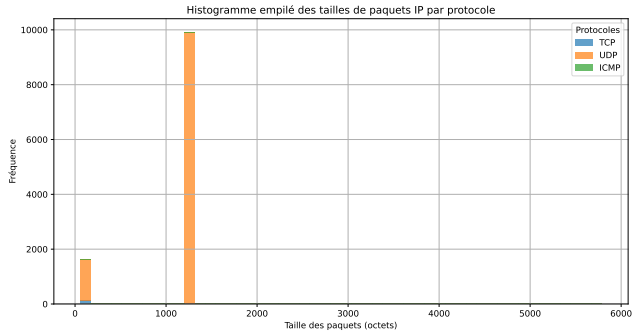


FIGURE 7 – Taille des paquets par Wifi

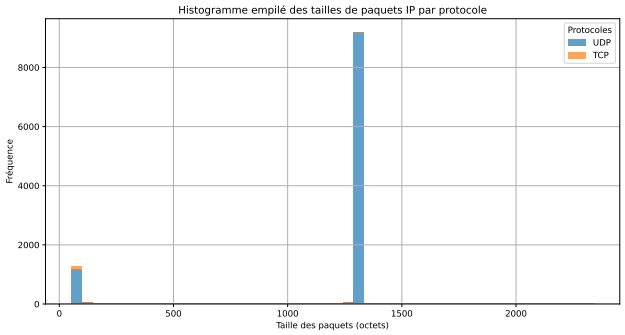


FIGURE 8 – Taille des paquets par ethernet

On peut clairement voir qu'on a deux zones bien distinctes pour la taille des paquets. En fait, le protocole Quic et UDP ont une taille standard de paquets, qui vaut exactement 1292 octets, soit 1250 octets de données par paquet. La grande majorité des paquets que nous analysons ont cette taille comme le montre l'histogramme. Des paquets plus petits semblent également arriver à notre ordinateur à la suite d'une série de ces paquets de taille standard, comme pour livrer le "reste des données". On va alors calculer les premiers et deuxièmes moments de la distribution des paquets.

	Wifi	**Ethernet**
Premier Moment	1113.88	1130.956
Deuxième Moment	1428062.24	1446491.527
Variance	187343.72	167445.3843
Minimum	54	54
Maximum	5790	2340
Médiane	1292.0	1292.0

TABLE 4 – Statistiques de longueurs des paquets

On voit dans le cadre de cette expérience des résultats assez comparables. On note cependant que la taille maximale des paquets ethernet est limitée par le Maximum Transmission Unit (MTU) de 1500 octets, donc la longueur maximale de paquet que nous avons pour wifi est plus élevée. On peut noter qu'on voit que la MTU a été dépassée pour ethernet, cela peut s'expliquer par le fait que le paquet a été défragmenté puis refragmenté par Wireshark après son passage en couche liaison.

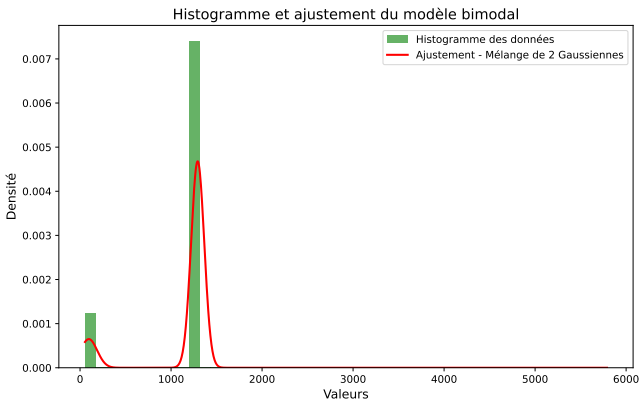


FIGURE 9 – Taille des paquets par Wifi

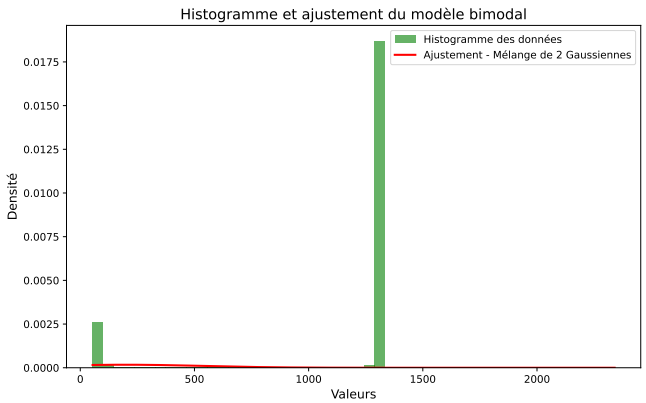


FIGURE 10 – Taille des paquets ethernet

Dans les deux cas, la p-value est proche de 0. On ne peut donc pas considérer que ces histogrammes suivent une normale. C'est cohérent car on a plusieurs paquets qui suivent un standard de taille comme expliqué auparavant. Cela ne peut donc pas suivre une loi normale.

3.2.2 Analyse du téléchargement

On représente dans le cas du téléchargement, les histogrammes de la longueur des paquets avec le protocole de liaison wifi et ethernet respectivement.

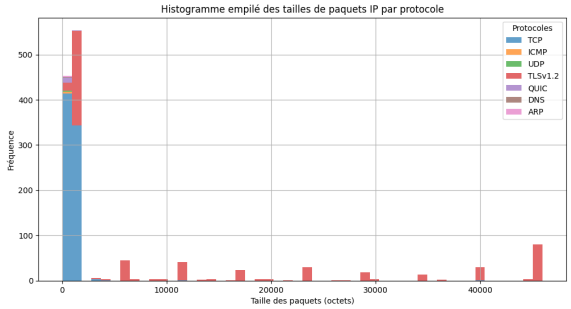


FIGURE 11 – Taille des paquets par Wifi pour le téléchargement

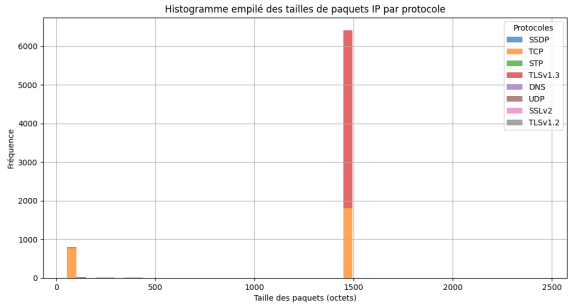


FIGURE 12 – Taille des paquets par ethernet pour le téléchargement

On observe que dans le cas du téléchargement par Wifi, la majorité des petits paquets sont en TCP, ce qui pourrait correspondre à des données de contrôle. En revanche, on remarque que la taille des paquets de données en TLSv1.2 est assez variable et dispersé. Au contraire, en Ethernet on observe bien 2 pics distincts, avec une taille fixe pour la transmission des données proche de 1500 bytes, et un pic de petits paquets TCP, que l'on attribue au contrôle.

On va alors calculer les premiers et deuxièmes moments de la distribution des paquets.

	Wifi	**Ethernet**
Premier Moment	6982.88	1322.33
Deuxième Moment	224315612.76	1956546.02
Variance	175687025.87	208015.92
Minimum	54	54
Maximum	45942	2456
Médiane	1488.0	1488.0

TABLE 5 – Statistiques de longueurs des paquets pour le téléchargement

Ces résultats confirment la grande variation de tailles de paquet dans le cas du Wifi.

3.2.3 Analyse de la Conversation Teams

On trace les histogrammes représentant la longueur des paquets pour la liaison wifi et ethernet :

Contrairement aux cas précédents, la longueur des paquets est ici beaucoup plus variable, et on n'observe pas deux longueurs bien définies. Cela s'explique par la présence significative du protocole STUN, qui opère au niveau de la couche application et utilise UDP comme protocole de transport. STUN, conçu pour gérer des communications à travers des NAT et pare-feu, génère des paquets dont les longueurs varient fortement en fonction des requêtes et réponses échangées, ce qui se reflète dans la distribution des tailles de paquets observée.

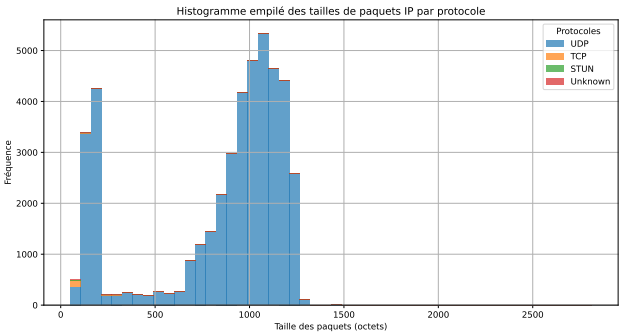


FIGURE 13 – Taille des paquets par Wifi

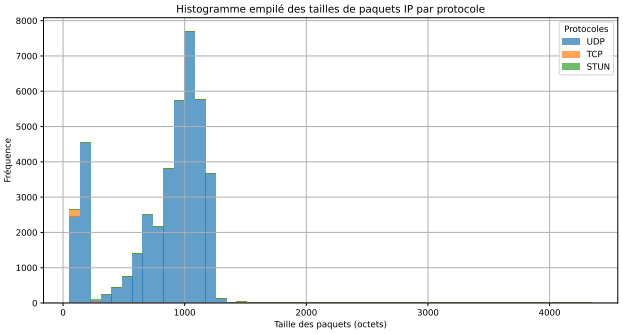


FIGURE 14 – Taille des paquets par ethernet

	Wifi	**Ethernet**
Premier Moment	843.22	820.35
Deuxième Moment	845274.52	795960.22
Variance	134253.55	122983.89
Minimum	49	54
Maximum	2814	4346
Médiane	983.0	957.0

TABLE 6 – Statistiques de longueurs des paquets

On va alors calculer les premiers et deuxièmes moments de la distribution des paquets.

On voit dans le cadre de cette expérience des résultats assez comparables. On note cependant qu'ici, la taille maximale des paquets ethernet est supérieur à celle en wifi, ce qui devrait être l'inverse normalement. Cependant, avec la valeur moyenne, on voit quand même que ceux en ethernet sont plus court que ceux en wifi, ce qui nous rassure quant aux attentes (liées à la limitation par MTU).

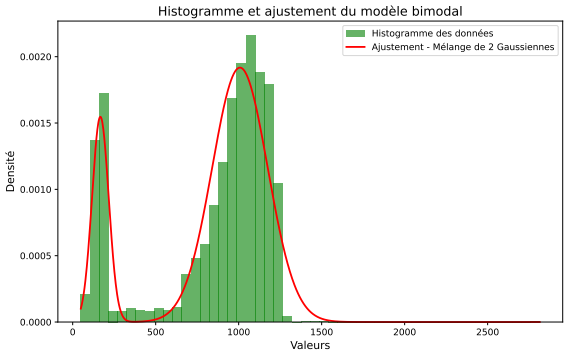


FIGURE 15 – Taille des paquets par Wifi

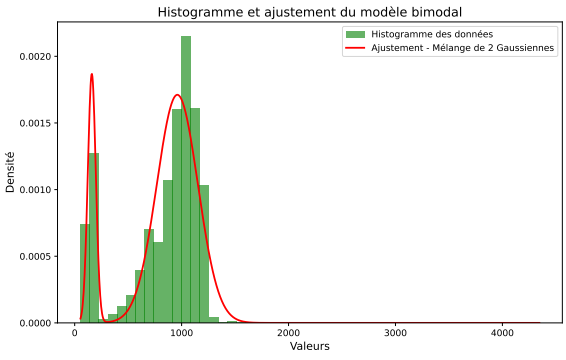


FIGURE 16 – Taille des paquets par Ethernet

Dans les deux cas, la p-value est proche de 0. On ne peut donc pas considérer que ces histogrammes suivent une normale, malgré que la courbe fit pas si mal avec les histogrammes.

3.3 Analyse de la gigue

Dans cette partie, nous ne gardons que les paquets de données.

3.3.1 Analyse des paquets du trafic vidéo

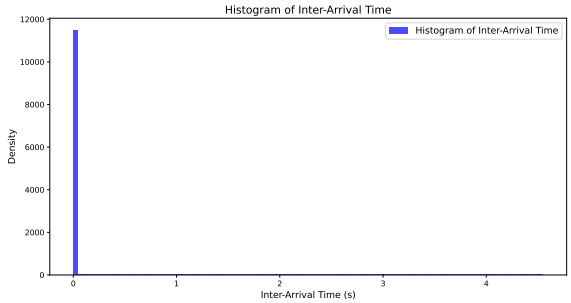


FIGURE 17 – Histogramme des temps d'inter-arrivée

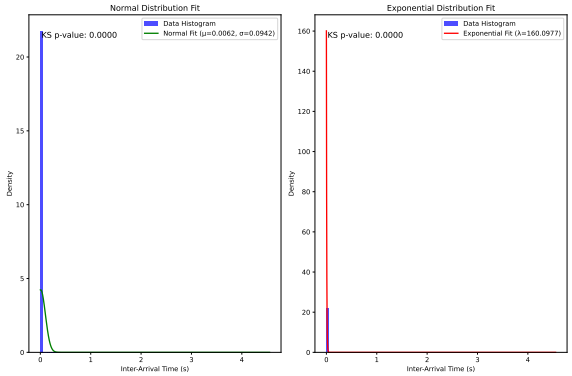


FIGURE 18 – Histogramme ajusté à des distributions connues

Sur la figure 17, on voit que l'histogramme brute ne nous renseigne pas sur la distribution des intervalles de temps. En effet, comme nous avons certains temps d'inter-arrivée de quelques secondes, ils forcent l'histogramme à devoir représenter ces valeurs, ce qui met la majorité des données dans le premier bin. Nous avons lors de cet essai essayé d'ajuster l'histogramme à une distribution connue, l'exponentielle qui nous semblait la plus appropriée. On note cependant que la p-value correspondante extraite du test de Kolmogorov-Smirnov (KS) est nulle pour ce test, ce qui permet de rejeter l'hypothèse nulle selon laquelle les temps d'inter-arrivée suivent une distribution exponentielle.

On remarque aisément que beaucoup de paquets ont un temps d'inter-arrivée nul. Une analyse plus approfondie nous a permis de constater que $\frac{87}{100}$ des paquets ont un temps d'inter-arrivée nul, ce qu'il faut expliquer.

Si nous retirons ces paquets nous avons les figures suivantes :

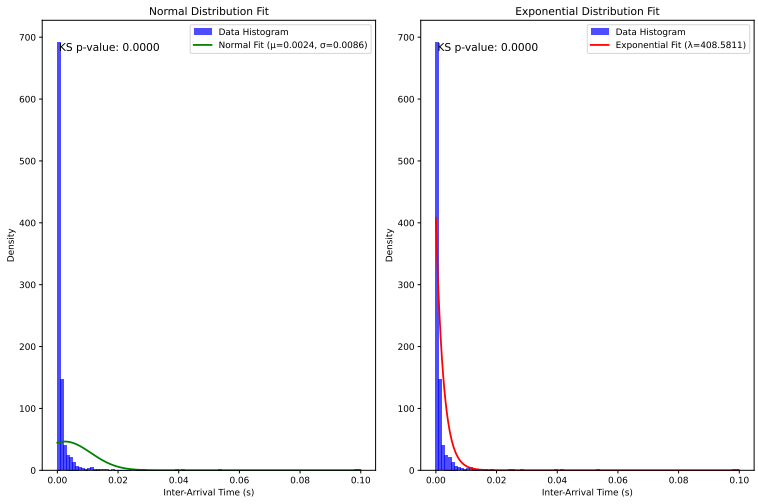


FIGURE 19 – Histogramme où les temps d'inter-arrivée nuls sont retirés

On note qu'on aperçoit deux pics dans la distributions des intervalles d'arrivée. Cependant, aucune distribution connue n'a permis d'obtenir une p-value significative.

Analysons maintenant la même expérience mais où nous avons utilisé le cable ethernet comme couche liaison. On répète la même procédure pour représenter les histogrammes. On a

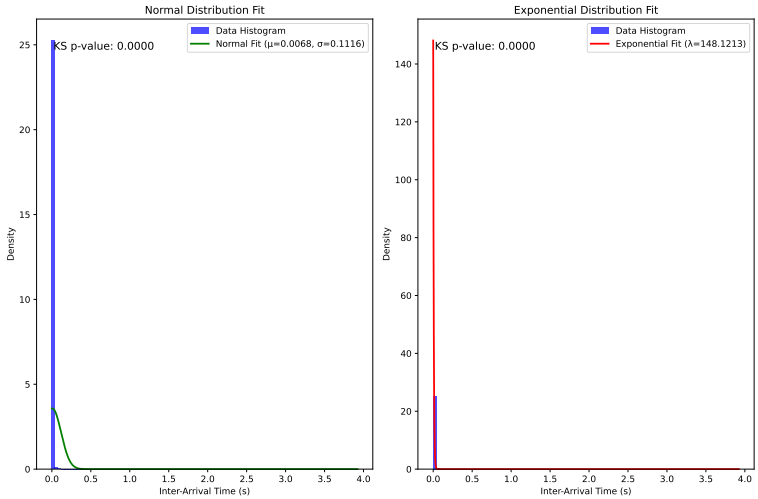


FIGURE 20 – Histogramme des temps d'inter-arrivée inférieurs à 0.1 s

des résultats très proches du cas avec liaison wifi. Si on décide d'enlever les paquets avec un temps d'inter-arrivée de 0 (qui représentent ici $\frac{88}{100}$ des paquets), on a la figure suivante, cela revient dont à assumer que les paquets des mêmes temps d'arrivée sont les mêmes paquets.

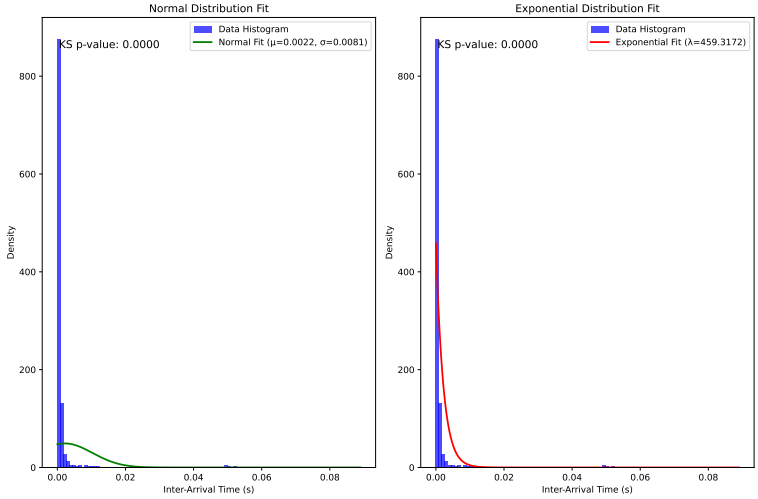


FIGURE 21 – Histogramme où les temps d'inter-arrivée nuls sont retirés

On remarque qu'encore une fois qu'aucune des distributions ne peut s'ajuster avec une bonne p-value aux histogrammes.

On peut s'intéresser au fait que la majorité des paquets ont un temps d'inter-arrivée de 0. Plusieurs raisons peuvent expliquer cela :

— **Problèmes de précision de l'horodatage :** Il est physiquement impossible que des paquets arrivant à la suite les uns des autres arrivent exactement au même instant. Une fréquence d'horloge trop basse du processus de capture de wireshark peut expliquer ce résultat.

— **trafic en Rafale :** On a remarqué que les hypothèses faites dans le cours pour modéliser une MM1 ne sont plus valides dans ce cas. Les temps d'arrivées des paquets ne sont pas indépendants les uns des autres.

Cela peut expliquer pourquoi nous ne pouvons retrouver des résultats similaires au premier laboratoire.

	Wifi	**Ethernet**
Premier Moment (s)	0.006246	0.006751
Deuxième Moment (s^2)	0.0089	0.012497
Variance (s^2)	0.008868	0.012453
Minimum (s)	0.0	0.0
Maximum (s)	4.550831	3.922867
Médiane (s)	0.0	0.0

TABLE 7 – Statistiques de gigue

La gigue semble acceptable pour l'analyse vidéo, car la variance est relativement faible dans les deux cas (0.008868 pour le Wifi et 0.012453 pour l'Ethernet), indiquant une stabilité des temps d'inter-arrivées.

3.3.2 Analyse des paquets du téléchargement

On n'effectuera pas d'analyse de la gigue pour le téléchargement. En effet, ce n'est pas un critère majeur de qualité de service dans cette situation, mais plutôt pour les applications de voix sur IP. Toutefois, nous pouvons analyser les temps d'inter-arrivés et commenter sur la nature du flux d'arrivée.

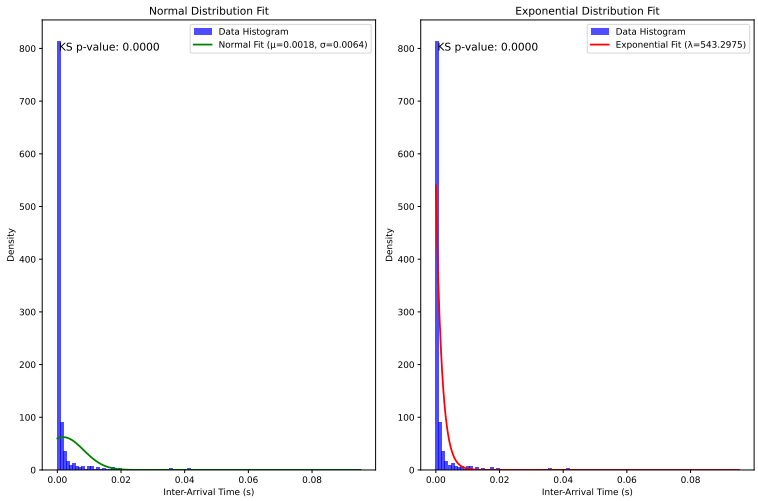


FIGURE 22 – Histogramme où les temps d'inter-arrivée nuls sont retirés, pour le téléchargement en Wifi

On remarque qu'encore une fois qu'aucune des distributions ne peut s'ajuster avec une bonne p-value aux histogrammes. Cela suggère que les temps d'inter-arrivée ne suivent pas une loi exponentielle et le processus d'arrivée n'est pas Poisson. Plusieurs explications peuvent en être la cause :

— Dépendances temporelles : La loi exponentielle repose sur l'indépendance des événements, ce qui peut être invalide dans un réseau où des paquets sont liés par la congestion, la synchronisation ou les messages de contrôle, comme les ACK dans TCP ou TLS.
— Autres distributions : Les données pourraient suivre une distribution différente, qui modélise mieux les phénomènes réseaux avec des taux d'arrivée variables ou des périodes de calme.
— Caractéristiques du réseau : Les processus réseau peuvent inclure des pics de trafic ou des périodes d'inactivité, entraînant des variations qui dévient des attentes d'une loi exponentielle.

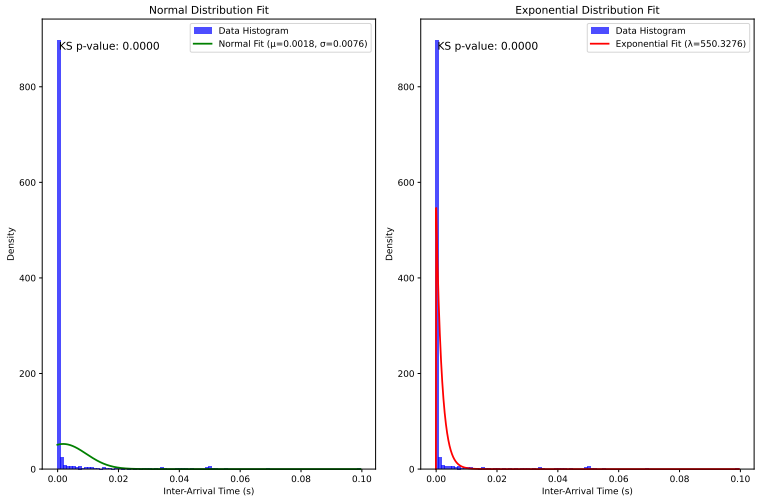


FIGURE 23 – Histogramme où les temps d'inter-arrivée nuls sont retirés, pour le téléchargement en Ethernet

3.3.3 Analyse des paquets de la Conversation Teams

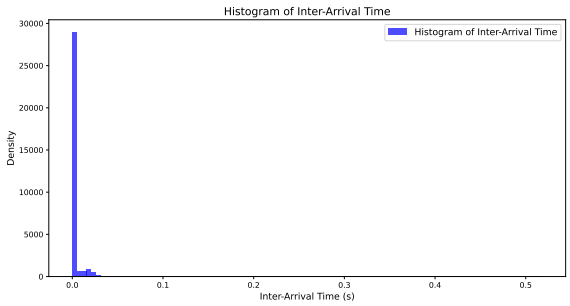


FIGURE 24 – Histogramme des temps d'inter-arrivée

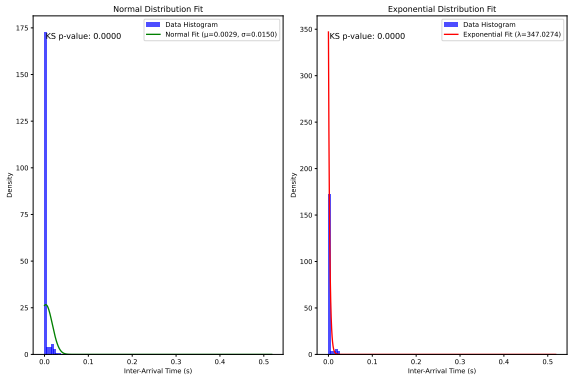


FIGURE 25 – Histogramme ajusté à des distributions connues

Les résultats sont similaire à ceux de la partie sur la vidéo Youtube, ce qui nous permet de faire les mêmes conclusions : L'histogramme brut ne permet pas d'observer la distribution des intervalles de temps. Les quelques temps d'inter-arrivée de plusieurs secondes étendent l'échelle, concentrant la majorité des données dans le premier bin. Nous avons tenté d'ajuster l'histogramme à une distribution exponentielle, mais la p-value du test de Kolmogorov-Smirnov étant nulle, l'hypothèse d'une distribution exponentielle est rejetée.

Si nous retirons ces paquets (ce qui représente $\frac{86}{100}$ des paquets), nous avons les figures suivantes 26.

On note qu'on aperçoit deux pics dans la distributions des intervalles d'arrivée, ce qui n'est pas représentatif des deux distributions testées. De plus, la p-value ($= 0$) nous montre aussi que les distributions ne peuvent pas être liées aux données acquisitionnées.

Analysons maintenant la même expérience mais où nous avons utilisé le cable ethernet comme couche liaison. On répète la même procédure pour représenter les histogrammes. On a des résultats très proches du cas avec liaison wifi. Si on décide d'enlever les paquets avec un temps d'inter-arrivée de 0 (qui représentent ici $\frac{79}{100}$ des paquets), on a la figure suivante.

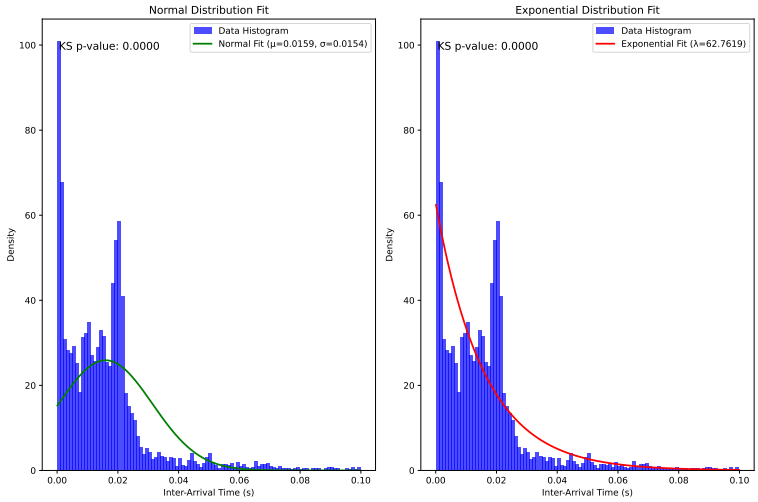


FIGURE 26 – Histogramme où les temps d'inter-arrivée nuls sont retirés

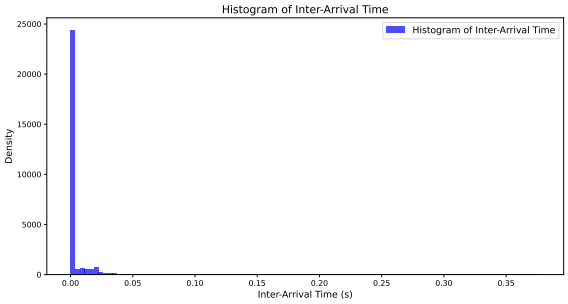


FIGURE 27 – Histogramme des temps d'inter-arrivée pour l'ethernet

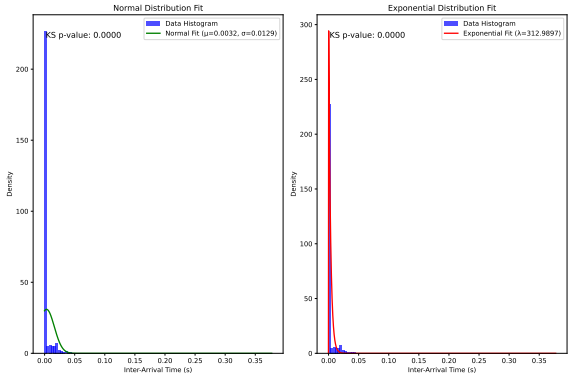


FIGURE 28 – Histogramme des temps d'inter-arrivée inférieurs à 0.1 s pour l'ethernet

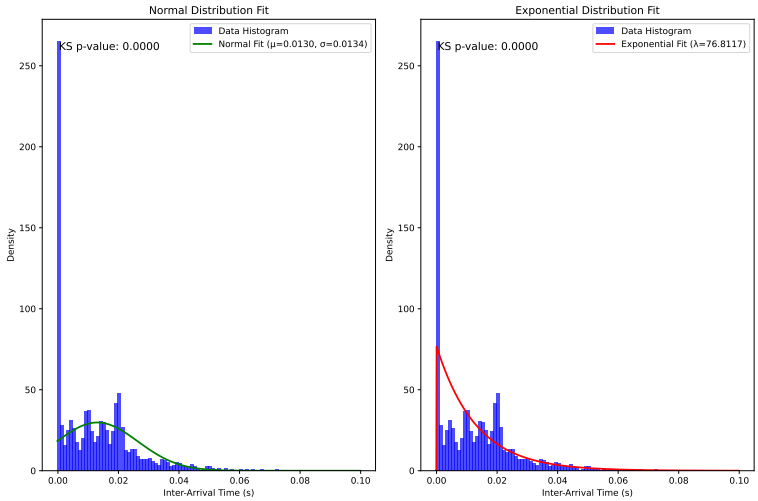


FIGURE 29 – Histogramme où les temps d'inter-arrivée nuls sont retirés pour l'ethernet

On remarque qu'encore une fois qu'aucune des distributions ne peut s'ajuster avec une bonne p-value aux histogrammes.

Cela peut expliquer pourquoi nous ne pouvons retrouver des résultats similaires au premier laboratoire.

	Wifi	**Ethernet**
Premier Moment (s)	0.002881	0.003195
Deuxième Moment (s^2)	0.000232	0.00017
Variance (s^2)	0.000223	0.000166
Minimum (s)	0.0	0.0
Maximum (s)	0.5181	0.3773
Médiane (s)	0.0	0.0

TABLE 8 – Statistiques de gigue

La gigue semble acceptable pour l'analyse vidéo, car la variance est relativement faible dans les deux cas (0.000223 pour le Wifi et 0.000166 pour l'Ethernet), indiquant une stabilité des temps d'inter-arrivées.

4 Partie Apprentissage Machine

Dans cette partie, nous allons développer plusieurs modèle de machine learning visant à classifier les paquets selon les critères suivants :

— Paquets de contrôle ou données
— Protocole d'application
— Liaison par Wifi ou Ethernet

4.1 Traitement et Nettoyage des données

Dans cette partie, nous expliquons brièvement comment rassembler les données en un seul dataframe pour pouvoir mener des analyses ayant du sens à partir de ce set de données.

Pour cela, nous importons tous les datasets de chacune des 6 expériences. Chacun d'entre eux doit être traité. Nous appliquons donc à ceux-ci des fonctions permettant de déterminer les protocoles d'application correspondants grâce aux numéros de port et à la documentation des protocoles [10]. Nous créons à partir du fichier pcap notre propre dataframe contenant les informations suivantes :

1. La longueur des paquets en octets
2. Le Temps d'arrivée des paquets
3. Le temps d'inter-arrivée des paquets
4. Le protocole de transport utilisé par les paquets
5. Le protocole d'application des paquets
6. L'adresse IP de la destination et de la source pour les paquets
7. Une colonne de booléens indiquant si le paquet est de contrôle ou non

Ensuite, on fusionne les 6 datasets correspondants aux 6 expériences. Nous nous débarassons de la colonne contenant les temps d'arrivée. En effet, comme il y a un temps de référence différent pour chaque expérience, celle-ci n'est plus pertinente.

Pour appliquer des algorithmes d'apprentissage machine, toutes les colonnes doivent contenir des nombres, pas de chaînes de caractères. Dès lors, nous utilisons un encodeur pour transformer les chaines de caractères des protocoles et les adresses IP en entier et en float de manière bijective. On enlève alors toutes les colonnes ne comportant pas de valeurs numériques.

Le dataset est maintenant prêt pour les expériences de machine learning.

4.2 Division en Set de données et Set de validation

En machine learning, diviser le jeu de données en *données d'entraînement* et *données de validation* est essentiel pour évaluer la performance d'un modèle. Les données d'entraînement servent à ajuster les paramètres du modèle, tandis que les données de validation permettent de mesurer sa capacité à généraliser sur des données non vues. Cela aide à détecter des problèmes tels que le surapprentissage (*overfitting*) ou le sous-apprentissage (*underfitting*) [9].

Utiliser **30% des données pour la validation** est une pratique courante, car :
— cela garantit une taille d'échantillon suffisamment grande pour une évaluation fiable du modèle ;
— cela laisse encore 70% des données pour l'entraînement, ce qui est généralement suffisant pour bien ajuster le modèle.

Ainsi, ce compromis entre évaluation et apprentissage permet une analyse robuste sans gaspiller les données disponibles.

4.3 Classification et Interprétation

4.3.1 Paquets de contrôle ou de données

Ici, nous allons chercher à développer un modèle performant pour prédire si nous avons un paquet de contrôle ou de données. Nous utilisons *LogisticRegression()* qui utilise la régression pour prédire la valeur d'un attribut avec une cardinalité de 2.

On représente la matrice de corrélation des données considérées dans ce problème sur la figure 30.

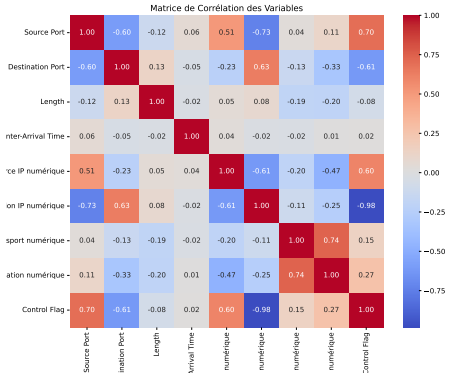


FIGURE 30 – Matrice de corrélation Contrôle

Nous comparons les prédictions avec un modèle de base qui prédit la valeur de la classe en prenant la moyenne des données.

Nous comparons les rapports de classification à l'aide des tableaux suivants :

On se rend compte que la classification naive attribue tous les paquets à la classe "données" car la majorité des paquets sont de la classe données. Par contre, on constate que les prédictions pour la régression sont excellentes, quelques erreurs sont évidemment commises, comme on peut le voir sur la matrice de confusion (31)

	Precision	**Recall**	**f1-score**	**Support**
Données	1.	1.	1.	27507
Contrôle	0.99	0.99	0.99	7728
Accuracy			1.	35235
Macro Avg	0.99	0.99	0.99	35235
Weighted Avg	1.00	1.00	1.00	35235

TABLE 9 – Rapport de classification pour la régression

	Precision	**Recall**	**f1-score**	**Support**
Données	0.78	1.	0.88	27507
Contrôle	0.	0.	0.	7728
Accuracy			0.78	35235
Macro Avg	0.39	0.5	0.44	35235
Weighted Avg	0.61	0.78	0.68	35235

TABLE 10 – Rapport de classification pour la classification naïve

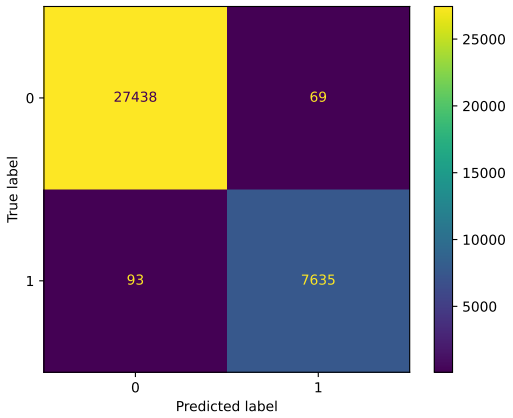


FIGURE 31 – Matrice de confusion pour la prédiction des paquets de contrôle ou de données pour la régression

On peut maintenant essayer de chercher parmis les nombreux attributs utilisés pour la prédiction, lesquels sont les plus significatifs pour prédire correctement la nature du paquet. Nous avons alors procédé par essai-erreur et comparaison des différents rapport de classification. Nous sommes alors arrivés à la conclusion que c'est la combinaison des ports de destination et des protocoles d'application qui nous permettent d'avoir les résultats obtenus ci-dessus.

On a également observé qu'on a une bonne performance en prenant l'adresse Ip de la destination, en effet, le modèle a tendance alors à classer tous les paquets arrivant comme des paquets de données et repartant comme des paquets de contrôle. Sans cela, la source de l'adresse IP joue également un rôle significatif. On a une précision moyenne avec uniquement la longueur des paquets. Ce modèle suggère que les paquets les plus courts sont considérés comme paquets de contrôle, ce qui a du sens. Ces résultats sont cohérents avec la figure 30.

4.3.2 Prédiction du protocole d'application

On va maitenant chercher un algorithme qui permet de déterminer le protocole d'application de chaque paquet. On choisit ici de retirer du dataset en plus de ce qu'on a déjà retiré auparavant,

les ports d'arrivée et de destination. En effet, cette information nous a permis de déterminer le protocole d'application de façon déterministe.

On représente la matrice de corrélation du dataset :

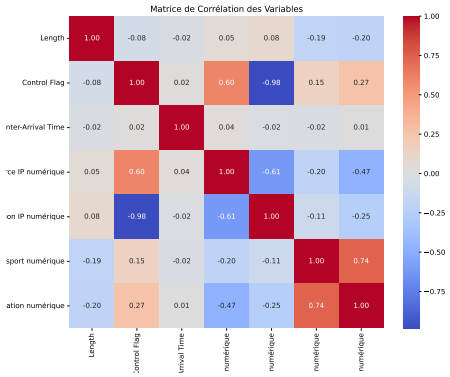


FIGURE 32 – Matrice de corrélation Application

On va alors comme il y a plus que deux valeurs pour la classe à prédire, utiliser un decision tree dont les performances sont représentés par le rapport de classification et la matrice de confusion suivants :

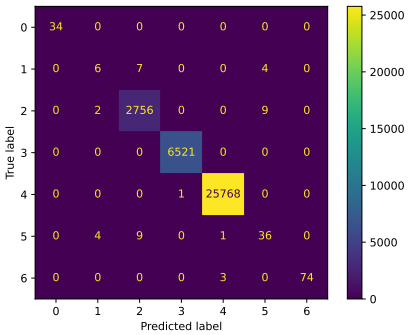


FIGURE 33 – Matrice de confusion Application Decision Tree

	Precision	**Recall**	**f1-score**	**Support**
DNS	1.	1.	1.	34
HTTP	0.50	0.35	0.41	17
HTTPS	0.99	1.00	1.00	2767
Quic	1.00	1.00	1.00	6521
STUN	1.00	1.00	1.00	25769
xTCP	0.73	0.72	0.73	50
xUDP	1.00	0.96	0.98	77
Accuracy			1.00	35235
Macro Avg	0.89	0.86	0.87	35235
Weighted Avg	1.	1.	1.	35235

TABLE 11 – Rapport de classification pour la classification avec decision tree

On voit que les résultats sont également très concluants. Cependant, comme vu dans la matrice de correlation, on utilise beaucoup de features alors que certaines sont inutiles aux

prédictions. On peut utiliser la matrice de correlation (37) pour déterminer les features les plus significatives comme on a fait au point précédent. Supposons maintenant que nous n'avons accès qu'à la longueur et au temps d'inter-arrivée des paquets, les features les moins corrélées à la longueur pour analyser les performances de l'algorithme de machine learning et comment optimiser celui-ci.

En utilisant le même algorithme qu'avant, on obtient la matrice de confusion et le rapport de classification suivant :

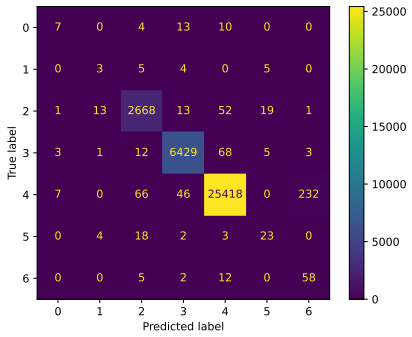


FIGURE 34 – Matrice de confusion Application Decision Tree avec moins de features

	Precision	**Recall**	**f1-score**	**Support**
DNS	0.39	0.21	0.27	34
HTTP	0.14	0.18	0.16	17
HTTPS	0.96	0.96	0.96	2767
Quic	0.99	0.99	0.99	6521
STUN	0.99	0.99	0.99	25769
xTCP	0.44	0.46	0.45	50
xUDP	0.20	0.75	0.31	77
Accuracy			0.98	35235
Macro Avg	0.59	0.65	0.59	35235
Weighted Avg	0.99	0.98	0.98	35235

TABLE 12 – Rapport de classification pour la classification avec decision tree et moins de features

Les résultats sont moins bons comme on voulait l'illustrer. Optimisons maintenant grâce à un gridsearch les paramètres suivants :

— *max depth* :$[3, 5, 10, 15, None]$
— *min samples split* :$[2, 5, 10]$
— *min samples leaf* :$[1, 2, 5]$
— *class weight* :$[balanced, None]$

On note que class weight en *balanced* sert de dire à l'algorithme que certaines classes sont sous-représentées par rapport à d'autres, et donc de donner plus de poids aux classes sous-représentées. Le grid search nous donne alors les paramètres optimaux :

$$'class_weight' : None, 'maxdepth' : None, 'minsamplesleaf' : 2, 'minsamplessplit' : 10$$

avec lesquels on entraine un nouvel arbre qui classe les données comme ci-dessous :

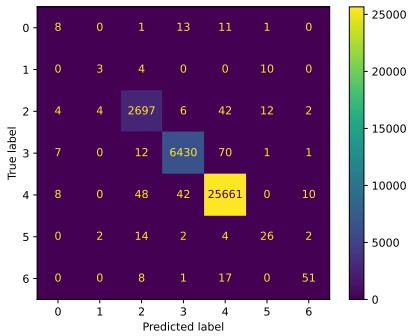


FIGURE 35 – Matrice de confusion

	Precision	**Recall**	**f1-score**	**Support**
DNS	0.3	0.24	0.26	34
HTTP	0.33	0.18	0.23	17
HTTPS	0.97	0.97	0.97	2767
Quic	0.99	0.99	0.99	6521
STUN	0.99	1.	1.	25769
xTCP	0.52	0.52	0.52	50
xUDP	0.77	0.66	0.71	77
Accuracy			0.99	35235
Macro Avg	0.7	0.65	0.67	35235
Weighted Avg	0.99	0.99	0.99	35235

TABLE 13 – Rapport de classification pour la classification avec decision tree, moins de features et optimisation

Les résultats sont meilleurs que pour l'arbre précédent, indiquant que celui faisait surement de l'overfitting et se généralisait moins bien.

Enfin, on représente la courbe d'apprentissage du modèle, qui montre bien que la performance du modèle sur les données de test augmente avec la taille de l'échantillon.

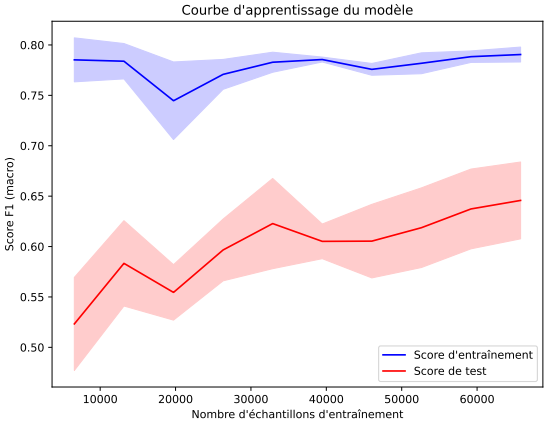


FIGURE 36 – Courbe d'apprentissage

4.3.3 Prédiction du moyen de liaison

Dans cette dernière section, on cherche à déterminer si la connexion se faisait en sans fil (Wifi) ou par Ethernet.

On représente alors la matrice de corrélation avec les features conservées pour cette dernière expérience.

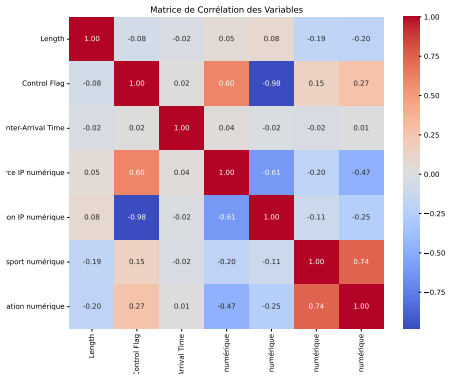
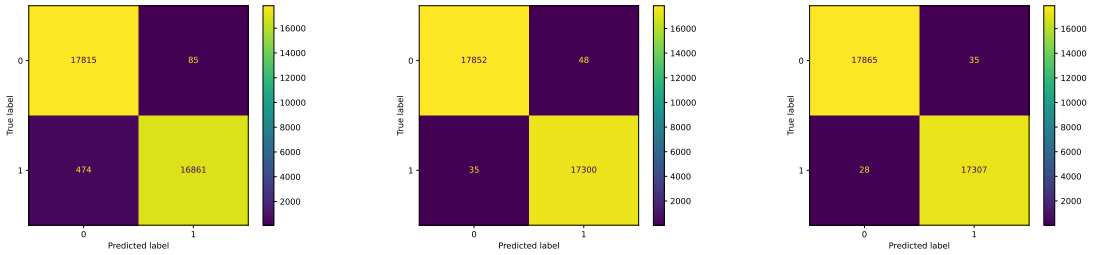


FIGURE 37 – Matrice de corrélation Liaison

Pour déterminer le moyen de liaison, nous avons essayé trois types de famille d'algorithme : la régression, K-Nearest-Neighbours et le decision Tree.

On va comparer ici leur confusion matrix sur la figure 38



Régression Knn Decision Tree

FIGURE 38 – Matrices de confusion pour liaison

On peut voir grâce à ces matrices et aux rapport de classification que c'est l'arbre de décision qui réalise la meilleure performance, juste avant knn et moins bon pour la régression qui est moins complexe mais aussi moins efficace. Le rapport de classification du decision tree est le suivant :

	Precision	**Recall**	**f1-score**	**Support**
Wifi	1.	1.	1.	27507
Ethernet	1.	1.	1.	7728
Accuracy			1.	35235
Macro Avg	1.	1.	1.	35235
Weighted Avg	1.	1.	1.	35235

TABLE 14 – Rapport de classification pour la régression avec la liaison

Noter que la classification n'est pas parfaite comme on peut le voir sur la matrice de confusion même si le rapport de classification est parfait.

Finalement, on essaie de déterminer les attributs qui contribuent à faire cette si bonne prédiction. Avec la matrice de corrélation et nos essais, nous avons déterminé qu'il s'agissait surtout du port de destination et de la source. En ne prenant que ces informations, nous entrainons notre decision tree sur ces features uniquement et nous obtenons le même rapport de classification (13) et la matrice de confusion suivante :

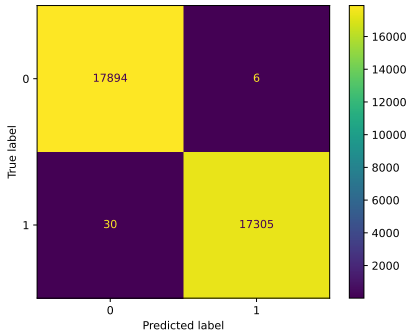


FIGURE 39 – Matrice de confusion avec decision tree pour la liaison

On peut voir que les prédictions sont même meilleures que quand on entraînait le modèle avec encore plus d'attributs.

En effet, lorsqu'un modèle est entraîné sur un sous-ensemble de caractéristiques présentant les plus fortes corrélations avec la variable cible, il bénéficie généralement d'une meilleure performance par rapport à un modèle utilisant l'ensemble complet des attributs. Cette amélioration peut être expliquée par plusieurs raisons :

- **Réduction du bruit :** En sélectionnant uniquement les caractéristiques ayant une forte corrélation avec la variable cible, on élimine les attributs qui apportent peu ou pas d'information utile pour la prédiction. Ces attributs non pertinents peuvent introduire du bruit dans le modèle, ce qui peut le rendre moins performant. En réduisant le nombre de variables, on simplifie le modèle, ce qui peut améliorer sa capacité à généraliser sur des données non vues.
- **Réduction du surapprentissage (Overfitting) :** L'ajout de nombreuses variables sans pertinence peut conduire à un surapprentissage, où le modèle s'adapte trop précisément aux données d'entraînement et perd de sa capacité à généraliser. En se concentrant sur les caractéristiques les plus informatives, le modèle devient moins susceptible de s'adapter aux fluctuations aléatoires des données d'entraînement, améliorant ainsi sa performance sur les données de test.
- **Moins de redondance :** Si certaines caractéristiques sont fortement corrélées entre elles, elles peuvent introduire de la redondance dans le modèle, rendant la formation moins efficace et augmentant la complexité computationnelle. En sélectionnant les caractéristiques les plus pertinentes, on réduit la redondance, ce qui permet au modèle de se concentrer sur les informations réellement significatives.
- **Simplicité du modèle :** En réduisant le nombre d'attributs, le modèle devient plus simple et moins susceptible de s'enliser dans des interactions complexes entre des variables peu pertinentes. Cela peut conduire à une meilleure interprétabilité et à des performances améliorées.

En résumé, la sélection des attributs ayant une forte corrélation avec la variable cible permet de réduire la complexité du modèle, de limiter le bruit et le surapprentissage, et de favoriser une meilleure généralisation, conduisant ainsi à des performances améliorées par rapport à l'entraînement sur un ensemble de données complet et potentiellement redondant.

5 Conclusion

Dans ce rapport, nous avons abordé plusieurs étapes essentielles pour l'analyse des paquets réseau et leur classification. Les expériences ont été conçues de manière à maximiser la fiabilité des conclusions en prenant en compte la diversité des paquets capturés et en excluant les paquets de contrôle, afin de se concentrer sur ceux contenant des informations utiles. Nous avons utilisé l'outil de capture de paquets pour sauvegarder les données sous forme de fichiers pcap, garantissant ainsi une traçabilité et une reproductibilité des résultats.

Les statistiques sur les processus d'arrivée des paquets ont été calculées, ce qui nous a permis d'obtenir une vue d'ensemble sur la distribution temporelle des paquets, excluant ceux qui ne participaient pas au transport d'information. De plus, des statistiques sur la taille des paquets IP ont été effectuées, nous fournissant des informations sur la répartition des tailles et leur impact potentiel sur les performances du réseau.

Une analyse plus approfondie a été menée sur la gigue, conformément à la définition de l'IETF, permettant d'évaluer la variabilité du délai d'arrivée des paquets et d'identifier d'éventuelles anomalies dans le flux de données.

Enfin, nous avons mis en place des modèles d'apprentissage machine pour la classification des paquets, en utilisant des techniques adaptées pour distinguer les différents types de paquets. Ces modèles ont permis d'obtenir des résultats prometteurs, offrant ainsi des perspectives intéressantes pour automatiser l'analyse des paquets réseau et détecter des anomalies ou des comportements suspects.

En conclusion, ce travail a permis d'acquérir des connaissances précieuses sur les paquets IP, leur taille, leur arrivée et la gigue, tout en mettant en place des outils de classification basés sur l'apprentissage machine. Ces méthodes ouvrent la voie à des analyses plus poussées et à une surveillance proactive des réseaux pour garantir leur performance et leur sécurité.

Références

[1] FrameIP. Entête STUN : Comprendre le protocole STUN. URL : https://www.frameip.com/entete-stun/.

[2] Equipe G-force. Localiser une adresse IP. Accessed : 2024-12-02. 2024. URL : https://www.g-force.ca/hebergement/ip-whois#:~:text=Le%20Whois%20est%20un%20service%20fourni%20par%20les,sur%20%22qui%20est%22%20le%20propri%C3%A9taire%20d%27une%20adresse%20IP..

[3] IBM. Transmission Control Protocol. Accessed : 2024-12-2. URL : https://www.ibm.com/docs/en/aix/7.2?topic=protocols-transmission-control-protocol.

[4] Ionos. QUIC : qu'est-ce qui se cache derrière le protocole expérimental de Google ? URL : https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/quic/.

[5] OpenSpaceCourse. Le protocole STP : Comprendre le fonctionnement de Spanning Tree Protoco URL : https://openspacecourse.com/le-protocole-stp/.

[6] Brunilde Sanso. Cours sur la Voix sur IP : Explication du protocole RTCP. Polytechnique Montréal, 2024.

[7] Brunilde Sanso. Précisions sur UDP et Transport multimédia tiré du Peterson et Davie. Polytechnique Montréal, 2024.

[8] Internet Society. TLS Basics. Accessed : 2024-12-2. URL : https://www.internetsociety.org/deploy360/tls/basics/.

[9] Statorials. Ensemble de validation et ensemble de test : quelle est la différence ? Accessed : 2024-12-2. URL : https://statorials.org/jeu-de-validation-vs-jeu-de-test/.

[10] Joe Touch. Recommendations on Using Assigned Transport Port Numbers. IETF, août 2015.

[11] Wireshark. Wireshark Documentation. Accessed : 2024-12-02. 2023. URL : https://www.wireshark.org/docs/.