# Impact of TP-Link WN 722N and Sniffer placement on trace completeness

Nabil Bencherif, Mohamed Chabane, Lisa Mehidi, Lizeth Paredes, Mohammad Imran Syed

# Impact of TP-Link WN 722N and Sniffer placement on trace completeness

Encadrant: **Mohammad Imran SYED**

Etudiants: **Nabil BENCHERIF, Mohamed CHABANE, Lisa MEHIDI, Lizeth PAREDES**

(Chaque auteur a contribué de manière équivalente au travail.)

# Table des matières

# 1 Cahier des charges

## 1.1 Introduction

Les interférences dans un réseau sans fil, lorsqu'un signal Wi-Fi est perturbé ou bloqué par des obstacles, est l'une des principales causes de connexions Internet lentes ou instables, faisant de la moindre petite tâche en ligne une montagne de travail et d'attente à la suite de la perte de paquets ou de réception de paquets erronés. Avec un nombre croissant de lieux utilisant de plus en plus la technologie Wi-Fi, il n'a jamais été aussi important de comprendre l'impact de ces interférences et comment les éviter au maximum.

## 1.2 L'objectif du projet

Le but de ce projet est de voir comment pouvons-nous améliorer au maximum les performances Wi-Fi, notamment en comparant la complétude des trames, en trouvant la méthode optimale d'utilisation de ces antennes installées sur un Raspberry Pi 4. Nous allons donc étudier l'impact du TP-Link WN 722N et de l'emplacement des récepteurs sur la complétude des traces Wi-Fi.

## 1.3 Définition du projet

La méthode choisie pour ce projet a été la combinaison d'antenne afin de former un super-sniffer. Un super-sniffer est composé de plusieurs sniffers individuels. Cette redondance aide à capturer une trace plus complète, car les renifleurs individuels sont voués à manquer certains paquets en raison des caractéristiques inhérentes au milieu sans fil comme les interférences, le multiple-trajets et l'évanouissement.

Un sniffer ou en français renifleur réseau (aussi appelé analyseur réseau ou analyseur de paquets) est un logiciel ou un appareil capable d'intercepter et d'enregistrer le trafic d'un réseau, il capte chacun des paquets qui transitent par ce réseau. Dans notre projet, ce qui fera office de sniffer, est le logiciel de tshark installé sur chaque Raspberry Pi 4 connecté à un adaptateur Wi-Fi «TP-Link WN 722 N ».

L'adaptateur Wi-Fi «TP-Link WN 722 N » est composé d'une carte Wi-Fi et d'antennes lui permettant l'envoie et la réception des données, à la réception, il convertit les signaux Wi-Fi en données électroniques et les transmit aux Raspberry Pi 4.

Le Raspberry Pi 4 est un mini-ordinateur composé d'un processeur, de mémoire vive, de divers ports et d'un système d'alimentation. Sa très faible taille et son coût réduit nous ont permis de mener à bien ce projet.

Le logiciel tshark s'occupe de la capture des trames Wi-Fi sur l'interface nouvellement créée par suite de la connexion de l'adaptateur Wi-Fi «TP-Link WN 722 N » au Raspberry Pi 4.

## 1.4 Les résultats attendus

Nous devons mettre en place plusieurs scénarios de capture de trace Wi-Fi qui se base sur plusieurs critères tels que la distance entre l'antenne émettrice et les antennes réceptrices qui capturent le trafic émis et leurs comportements lorsqu'on les combine, l'impact des interférences selon l'endroit de simulation du projet.

Voici la liste des tâches à effectuer :

- Installation de système d'exploitation sur chaque Raspberry Pi 4 et configuration des antennes.
- La programmation des scripts permettant la génération des traces émises, leur capture et le traitement de celles reçues.
- Réalisation des tests de simulations des antennes en considérant différents critères.
- Récupération et analyse des données générés après la simulation.
- Extraction des traces avec les champs d'information qui seront utilisés pour l'analyse.
- Synchronisation des traces extraites.
- Analyser le comportement des antennes individuellement et en groupe.
- Obtenir les statistiques et des graphiques après l'analyse de comportement des antennes.
- Enfin, l'analyse globale en comparant les résultats et donner notre conclusion.

Le résultat attendu c'es de pouvoir avoir une idée claire de comportement des antennes Wi-Fi à travers de nos résultats statistiques selon notre analyse et pouvoir définir un scenario où les conditions Wi-Fi par ces antennes seront optimales.

## 1.5 Contraintes

Nous avons des contraintes temporelles à respecter :

- 24 janvier : remise du rapport intermédiaire.
- 09 mai : remise du rapport final.
- 25 mai : soutenance.

Concernant les contraintes matérielles, nous avons à dispositions cinq adaptateurs Wi-Fi « TP-Link WN 722 N », cinq Raspberry Pi 4, que nous allons configurer, un petit écran pour afficher le bureau du RPI, câble Ethernet ou HDMI pour un affichage sur un grand écran et des batteries externe pour alimenter les Raspberry Pi 4 surtout au moment de la réalisation des simulations des tests.

# 2 Plan de développement

Le plan de développement a pour but de définir et planifier les différentes étapes de la réalisation du projet. Ainsi, ce projet se décompose en trois axes principaux, la rédaction des documents, la partie développement et la documentation finale.

Cependant, avant cela, il y a tout un travail d'appropriation et de recherche documentaire à effectuer. Plusieurs articles parlant des problématiques dans les réseaux sans fil, sélectionnés par nous-même et l'encadrant, ont dû être lus, compris et résumés.

Avec les tutorats de la MIR (Mathématiques, Informatique et Recherche), nous avons affiné et amélioré nos recherches. Nous avons pu avoir accès à d'autres sources qui nous ont permis d'en comprendre davantage sur le fonctionnement des réseaux sans fil. Nous avons rédigé un carnet de bord expliquant notre sujet et les démarches effectuées pour remplir notre bibliographie.

Ensuite, il y a divers autres documents à concevoir : le cahier des charges, l'analyse, la conception détaillée, l'état d'avancement du projet ou encore les diagrammes et les études statistiques.

La rédaction de l'ensemble des documents comprenant également le carnet de bord et la bibliographie, ont été équitablement répartis entre tous les membres du groupe.

Pour optimiser l'organisation du travail, nous avons partagé le matériel reçu et chacun a configuré le matériel qui lui est attribué puis nous avons décidé de séparer le groupe en deux sous-groupes, l'un s'occupera de coder le comportement d'une antenne émettrice Wi-Fi  et le script qui permet de traiter les donner capturer, et l'autre se chargera de coder le script de capture des traces et d'interpréter les différents traitements sous forme de diagramme. De plus, nous ferons un point régulièrement sur l'état d'avancement de chaque sous-groupe.

La planification de notre développement suivra le diagramme de Gantt, il décrit l'ordre et l'avancement de chaque tâche de notre projet.

**Projet 8**
**Impact TP-Link-Sniffer-Trace completeness**

N. Bencherif
M. Chabane
L. Mehidi
L. Paredes

Voici le diagramme de Gantt :

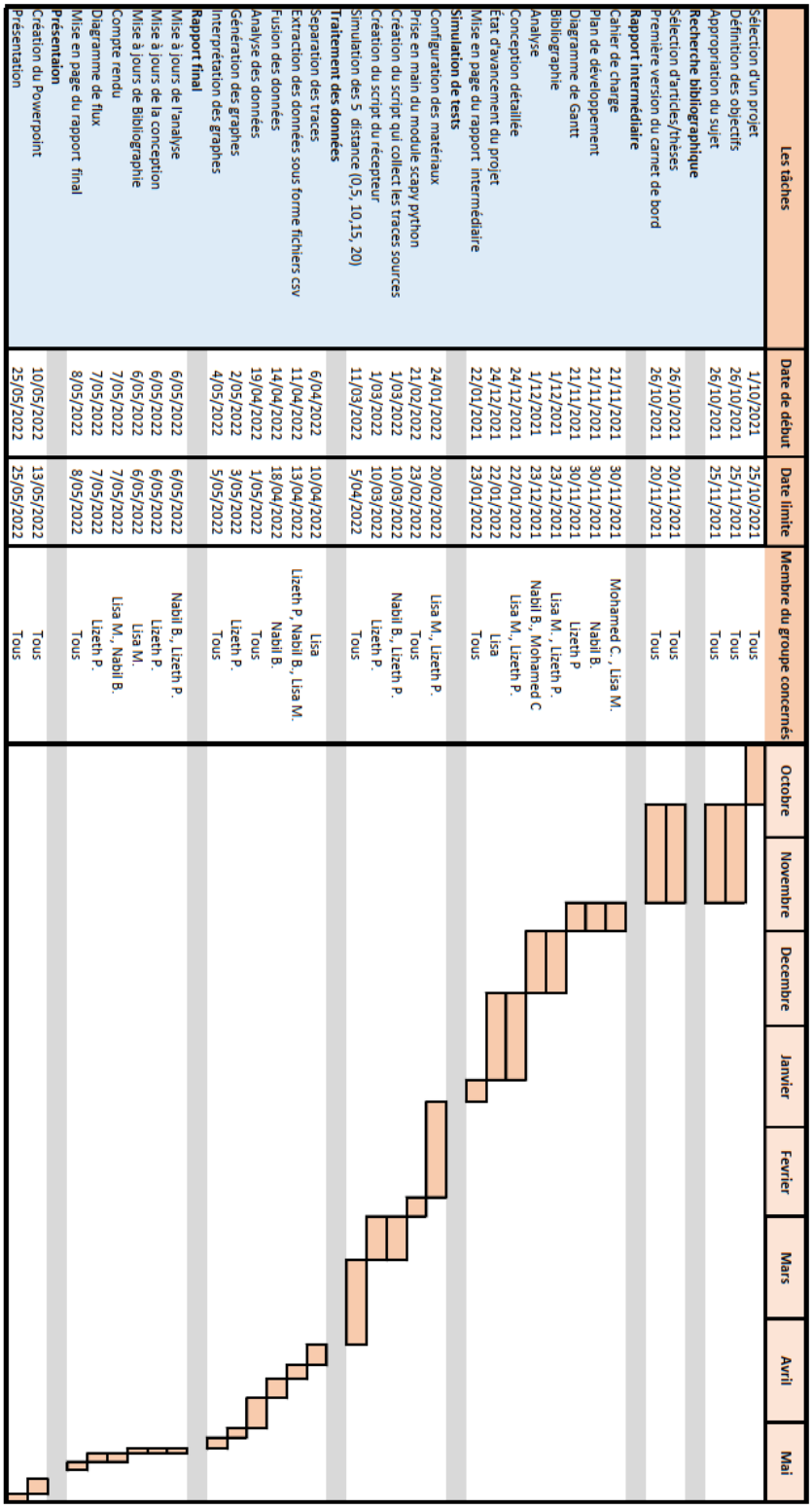


*Figure 1. Diagramme Gantt du projet*

# 3 Bibliographie

## 3.1 Descriptif de la recherche documentaire

Lors de notre tutorat de la MIR, nous avons appris à utiliser les outils adéquats à la réalisation d'une bibliographie.

Nous avons utilisé principalement des bases de données comme Association for Computing Machinery (ACM), Springer et ar Xiv, nous avons aussi utilisé google Scholar Et le site « https:/dblp.Org/ ».

Lors de la recherche documentaire sur ses différentes bases de données, nous avons cherché par union de mots-clés, auteurs et dates. De plus, en lisant la bibliographie de certains articles, nous avons pu trouver des sources supplémentaires intéressantes.

Ils nous ont permis d'accéder à des articles de revues et de conférences. Toutes nos sources correspondent à de la littérature académique, des thèses, des articles et des livres.

## 3.2 Carte heuristique

Une carte heuristique a été réalisée pour aider à la recherche documentaire. Elle permet de réfléchir à la problématique et d'en extraire les principaux mots clés.

Cette carte se trouve à l'annexe A.

## 3.3 Articles

### 3.3.1. Article [1]

Dans cet article, les auteurs présentent leur création, une boîte à outils qui permet l'enregistrement des mesures détaillées du canal sans fil ainsi que des traces Wi-Fi reçus (paquets 802.11).

Comparer aux autres outils tels que l'indicateur d'intensité du signal de réception (RSSI), qui ne fait que capturer la puissance totale reçue par l'auditeur, cet outil quant à lui fournit des informations sur le canal entre l'émetteur et le récepteur au niveau des sous-porteuses de données individuelles pour chaque paire d'antennes d'émission et de réception.

Il nous offre aussi une fonctionnalité de point d'accès pour contrôler les deux extrémités du lien et des scripts MATLAB pour analyser les données reçues.

### 3.3.2. Article [2]

Cette thèse nous présente les antennes configurables, une antenne configurable possède différents diagrammes de rayonnement codant chacun pour un symbole binaire particulier.

Dans le but d'accroître la compacité des antennes, ils ont conçu une antenne à anneaux fendus générant 8 différents diagrammes de rayonnement décorrélés à la fréquence de 2.45 G. Hz, ces dernières présentes des résultats prometteurs dans une communication numérique en environnement indoor.

### 3.3.3. Article [3]

Cet article porte sur l'évaluation de l'efficacité de deux technologies radio Wi-Fi et RFID dans les applications sans fil intérieur. Il expose une recherche dans laquelle un modèle de propagation radio réaliste en 3D est étudié afin d'approximer la propagation radio dans un environnement intérieur, puis d'effectuer des mesures sur le terrain en utilisant des radios Wi-Fi et RFID dans un bâtiment universitaire à deux étages.

La recherche aboutie au fait que la radio Wi-Fi est plus stable que la RFID et qu'elle transmet sur une portée plus grande. Elle nous apprend aussi que l'atténuation des signaux radio par des cloisons multiples n'est pas linéaire par conséquent, il n'existe pas de relation simple entre les murs (les obstacles) et l'intensité du signal dans un environnement intérieur.

### 3.3.4. Article [4]

Cet article traite le problème des interférences inter-technologique (CTI), c'est-à-dire le chevauchement temporel et fréquentiel de transmissions simultanées.

Ces interférences se produisent en raison de la nature des transmissions sans fil et des diffusions produites par des dispositifs situés au même endroit et dont les radios sont basées sur des techniques différentes, telles que le Wi-Fi (IEEE802.11), Bluetooth (IEEE802.15), Etc.

Ils ont étudié la coexistence des WSN basés sur le Wi-Fi et l'IEEE802.15.4, mais les principes fondamentaux sont applicables à d'autres technologies et aussi Ils ont proposé une solution pour modéliser avec précision le trafic Wi-Fi agrégé et pour prédire le niveau d'interférence.

Le modèle de trafic Wi-Fi qu'ils ont utilisé a été validé contre des traces réelles et un modèle Pareto de pointe et le mécanisme de prédiction a été évalué par rapport à la méthode d'accès aléatoire 0,5-persistant, à la fois pour le trafic saturé et non saturé.

### 3.3.5. Article [5]

Cet article présente les recherches sur les systèmes de positionnement à l'intérieur des bâtiments et la problématique des coûts d'installation élevé pour couvrir toute un environnement avec une technologie spéciale.

Pour diminuer les coûts, Ils ont proposé l'utilisation des points d'accès Wi-Fi, l'empreinte Wi-Fi est devenue prometteuse car elle ne nécessite pas l'installation de dispositifs supplémentaires dans l'environnement intérieur, elle utilise les points d'accès Wi-Fi existants, cependant elle nécessite toujours des phases de calibrage hors ligne pour caractériser l'environnement intérieur à utiliser.

Ils ont présenté l'algorithme de positionnement intérieur basé sur le Wi-Fi tout en montrant ces performances de positionnement avec des simulations. Puis ils ont présenté la simulation des performances de l'IMU et de l'INS. Enfin, ils ont présenté l'algorithme de fusion de capteurs qui permet d'améliorer les performances globales du système.

Après avoir fusionné le positionnement Wi-Fi RSSI et le positionnement INS en prenant en compte différents chemins et un mouvement de 1 minute, les résultats de la simulation montrent une erreur de positionnement prometteuse inférieure à 1 mètre ce qui peut également être amélioré en utilisant une estimation avancée comme le Kálmán étendu (EKF), le Kálmán non centré (UKF).

### 3.3.6. Article [6]

Cet article a pour objectif de proposer de nouveaux mécanismes et protocoles qui visent à "améliorer l'utilisation des ressources dans les réseaux de paquets sans fil".

Ces améliorations peuvent être réalisées à deux niveaux complémentaires : le niveau du paquet et le niveau de la connexion.

Au niveau de la connexion, la principale préoccupation était de maintenir la qualité de service expérimentée alors que les utilisateurs souhaitent se déplacer de manière transparente sur les divers réseaux de paquets sans fil disponibles ; tandis qu'au niveau des paquets, la question abordée était d'améliorer la qualité de service expérimentée par les flux de paquets générés par les nouveaux services de communication de données et multimédia.

### 3.3.7. Article [7]

Cet article porte sur les modules de laboratoire basés sur Scapy, qui fournissent une exposition pratique à la manipulation des en-têtes de réseau.

Ils sont utiles pour les campus sans installations de laboratoires réseau dédié, car leur capacité, à expérimenter des protocoles sur un réseau institutionnel existant est limitées.

Le module Scapy basé sur Python fournit un contrôle explicite et détaillé du contenu des champs d'en-tête et inclut des fonctionnalités de visualisation graphique qui offrent une rétroaction facile.

L'environnement Python interactif permet une exploration pas à pas et guidée des différents protocoles. L'utilisation efficace de Scapy nécessite des privilèges Root (administrateur) ; un environnement de machines virtuelles tel que celui fourni par Oracle Virtualbox qui permet un contrôle et un accès complets au système d'exploitation.

## 3.4 Livres

Pour ce projet, nous avons étudié le cas de plusieurs livres tels que :

Le livre « Programming with Raspberry Pi » [9] : Ce livre nous parle des fonctionnalités du Raspberry Pi, comment configurer le système d'exploitation d'un Raspberry Pi, configuration des ports et les bases de python, on s'est inspiré pour mettre les interfaces en mode moniteur.

Le livre « Raspberry Pi By Example » [10] : Ce livre nous parle de la configurer d'un Raspberry Pi, préparation manuelle de la carte MicroSD, Minecraft Pi, Construire des jeux avec PyGame, Programmation réseau en python sur Raspberry Pi et comment travailler avec une webcam. Dans ce dernier chapitre, il aborde le Cron qui est un planificateur de tâches basées sur le temps dans les systèmes d'exploitation Unix, on s'est inspiré de leur exemple pour automatiser et synchroniser le lancement de nos scripts sur les Raspberry Pi.

Le livre « Mastering Python for Networking and Security » [11] : Ce livre aborde la programmation python et la programmation réseau avec python sous Linux, parle de l'analyse du trafic réseau avec le scanner Nmap, balayage synchrone en utilisant Numpy dons en c'est inspiré pour automatiser le lancement de nos scripts et la synchronisation des résultats des captures.

## 3.5 Librairies

Pour ce projet, nous avons utilisé plusieurs des librairies et modules telles que :

- **La librairie matplotlib** :

  C'est une bibliothèque du langage de programmation Python destinée à tracer et à visualiser des données sous forme de graphiques et elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy.

- **La librairie numpy :**

  C'est une librairie qui fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynôme.

- **La librairie pandas :**

  C'est une librairie python qui permet de manipuler facilement des données à analyser. On peut notamment manipuler des tableaux de données avec des étiquettes de variables qui correspondent aux colonnes et aux lignes.

  Ces tableaux sont appelés DataFrames, on peut facilement lire et écrire leurs données à partir ou vers un fichier tabulé et on peut tracer des graphes à partir de ces DataFrames grâce à matplotlib.

- **Le module Subprocess :**

  C'est un module de python permettant d'exécuter des programmes externes et de lire leurs sorties dans nos codes Python. Il comprend plusieurs classes et fonctions qui nous permettent exécuter des commandes shell et des binaires exécutables.

- **Le module time :**

  Ce module fournit différentes fonctions liées au temps, nous l'utilisons principalement pour obtenir le temps retourné par la fonction time.time() qui renvoie le temps en secondes depuis epoch sous forme de nombre à virgule flottante. Pour les systèmes Unix, epoch est le 1er janvier 1970, 00:00:00 (UTC).

- **Le module csv :**

  Le module csv implémente des classes pour lire et écrire des données tabulaires au format CSV (Comma Separated Values, valeurs séparées par des virgules). Ce format est plus commun dans l'importation et dans l'exportation des feuilles de calculs et des bases de données.

- **Le module struct** :

  Ce module effectue des conversions entre des valeurs Python et des structures C représentées sous la forme de bytes (séquences d'octets) Python. Cela permet, entre autres, de manipuler des données agrégées sous forme binaire dans des fichiers ou à travers des connecteurs réseau. Il utilise les chaînes de spécification de format comme description de l'agencement des structures afin de réaliser les conversions depuis et vers les valeurs Python.

- **Le module Scikit-learn:**

  Aussi appelé sklearn, est la bibliothèque la plus puissante et la plus robuste pour le machine learning en Python. Elle fournit une sélection d'outils efficaces pour l'apprentissage automatique et la modélisation statistique, notamment la classification, la régression et le clustering via une interface cohérente en Python. Elle est en grande partie écrite en Python, s'appuie sur NumPy, SciPy et Matplotlib.

## 3.6 Outils

Pour ce projet, nous allons utiliser différents outils tels que :

- gitlab (https://gitlabsu.sorbonne-universite.fr/users/sign_in) : pour la gestion du répertoire de notre projet.
- Raspberry Pi (https://raspberry-pi.fr/) : pour trouver le système d'exploitation à installer dans les Raspberry-Pi 4.
- Miro (https://miro.com/app/board/) : pour la création de diagrammes de flux.
- Pypal (https://gitlab.lip6.fr/syed/pypal) : pour synchroniser et fusionner les traces capturées.

## 3.7 Références bibliographiques

[1-12]

[1] Susanne Albers, Stefan Eilts, Eyal Even-Dar, Yishay Mansour, and Liam Roditty. 2014. On Nash Equilibria for a Network Creation Game. *ACM Trans. Econ. Comput.* 2, 1 (March 2014), 2:1-2:27. DOI:https://doi.org/10.1145/2560767

[2] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2011. Tool Re- lease: Gathering 802.11n Traces with Channel State Information. ACM SIGCOMM Comp. Commun. Rev. 41, 1 (January 2011),53–53. DOI: https ://doi.org/10.1145/1925861.1925870

[3] Kammel Rachedi. 2019. Antennes compactes reconfiguration en diagramme de rayonnement pour la modulation spatiale MIMO et introduction aux communications numériques par rétrodiffuseurs. phdthesis. Sorbonne Université. Retrieved from https ://tel.archives- ouvertes.fr/tel-03139838

[4] Yiming Ji and Robert Player. 2011. A 3-D indoor radio propagation model for WiFi and RFID. In Proceedings of the 9th ACM international symposium on Mobility management and wireless access (MobiWac '11), Association for Computing Machinery, New York, NY, USA, 27–34. DOI :https ://doi.org/10.1145/2069131.2069137

[5] Indika S. A. Dhanapala, Ramona Marfievici, Sameera Palipana, Piyush Agrawal, and Dirk Pesch. 2017. Modeling WiFi Traffic for White Space Prediction in Wireless Sen-sor Networks. arXiv :1709.08950 [cs] (September 2017). from http ://arxiv.org/abs/1709.08950

[6] Abdelrahman El-Naggar, Amr Wassal, and Khaled Sharaf. 2019. Indoor Positioning Using WiFi RSSI Trilateration and INS Sensor Fusion System Simulation. In Pro ceedings of the 2019 2nd International Conference on Sensors, Signal and Image Proces- sing (SSIP 2019), Association for Computing Machinery, New York, NY, USA, 21–26. DOI: https ://doi.org/10.1145/3365245.3365261

[7] Yacine Ghamri-Doudane. 2010. Contributions à l'amélioration de l'utilisation des ressources dans les réseaux de paquets sans fil. Thésis. Université Paris-Est. Retrieved January 6, 2022 from https ://tel.archives-ouvertes.fr/tel-00848592

[8] Robert Montante. 2018. Using Scapy in Teaching Network Header Formats: Programming Network Headers for Non-Programmers (Abstract Only). In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18), Association for Computing Machinery, New York, NY, USA,1106. DOI:https ://doi.org/10.1145/3159450.3162228

[9] Sai Yamanoor and Srihari Yamanoor. 2017. Python Programming with Raspberry Pi. Packt Publishing. from http ://univ.scholarvox.com.accesdistant.sorbonne-universite.fr/catalog/book/docid/88842797

[10] Ashwin Pajankar and Arush Kakkar. 2016. Raspberry Pi By Example. Packt Publishing. From http://univ.scholarvox.com.accesdistant.sorbonne- universite.fr/catalog/book/doc

[11] Jose Manuel Ortega. 2018. Mastering Python for Networking and Security. Packt Publishing. From http://univ.scholarvox.com.accesdistant.sorbonne-universite.fr/catalog/book/docid/88863251?searchterm=20scapy20python

[12] Mohammad Imran Syed, Anne Flandenmuller, and Marcelo Dias De Amorim. 2022. PyPal: Wi-Fi Trace Synchronization and Merging Python Tool. LIP6 UMR 7606, UPMC Sorbonne Université, France. Retrieved May 7, 2022 From: https://hal.archivesouvertes.fr/hal-03618014

# 4 Analyse

## 4.1 Analyse du contexte

Nous avons décidé de faire une analyse basée sur différents tests qui mettent en relation les distances entre les antennes de réception et la quantité de trames envoyées afin d'évaluer l'intégrité des trames reçues par ceux-ci, pour cela nous avons utilisé comme matériel 5 antennes TPLINK WN 722 N version 3 et 5 dispositifs Raspberry. Parmi les 5 antennes, l'une d'entre elles agira comme un point d'accès (émetteur)qui enverra des trames aux 4 autres récepteurs.

Nous avons décidé de réaliser les tests à la faculté et nous avons choisi de faire 5 tests de 5 minutes chacun en considérant 5 distances (0cm, 5cm, 10cm, 15cm et 20cm) de séparation entre les 4 récepteurs.

Au cours du premier semestre, nous avons déjà effectué des tests avec 3 antennes, dans lesquelles nous avons installé les pilotes dans les équipements Raspberry et les avons configurés pour qu'ils puissent fonctionner. Au cours de ce second semestre a suivi la suite du projet, nous avons commencé par la mise en œuvre des tests avec les 5 antennes et les équipements Raspberry. Cependant, lors de l'installation des pilotes dans les antennes restantes, un problème a été mis en évidence, le nouveau système d'exploitation de Raspberry a présenté des incompatibilités avec les pilotes, malgré cela nous avons trouvé un moyen d'installer les pilotes en programmant automatiquement certaines commandes qui ont permis de configurer les antennes en mode moniteur, mode qui permet de donner la fonctionnalité à l'antenne d'envoyer et de recevoir des paquets. Or, après avoir exécuté ces commandes, il a été nécessaire de déconnecter et de connecter l'antenne pour que l'équipement Raspberry puisse la reconnaître, en bref, le bug du driver a été résolu de cette manière.

Après avoir résolu ce problème, un second problème est apparu, celui de synchronisation entre l'antenne de l'émetteur et les récepteurs, puisqu’ après la déconnexion entre le Raspberry et les antennes, elles n'arrivaient pas à se synchroniser en même temps. Cela aurait pu impliquer que les tests effectués ne seraient pas pertinents et ne pourraient pas être analysés correctement.

Après avoir montré à notre responsable tout ce que nous avons découvert avec les antennes, nous avons convenu avec lui qu'il était impossible de continuer les tests de ces antennes tant que ce bug n'était pas résolu. Nous avons également découvert que ce problème était récurrent dans les antennes TP-Link WN 722N v3, mais pas dans la version 2, car dans cette version il

n'est pas nécessaire d'installer de pilote. Suite à cela et avec l'approbation de notre professeur encadrant responsable, il a été décidé de travailler avec les antennes Alfa Networks, ne nécessitant pas d'installer de pilotes, en suivant la même analyse déjà évoquée précédemment.

La structure de l'analyse sera la suivante

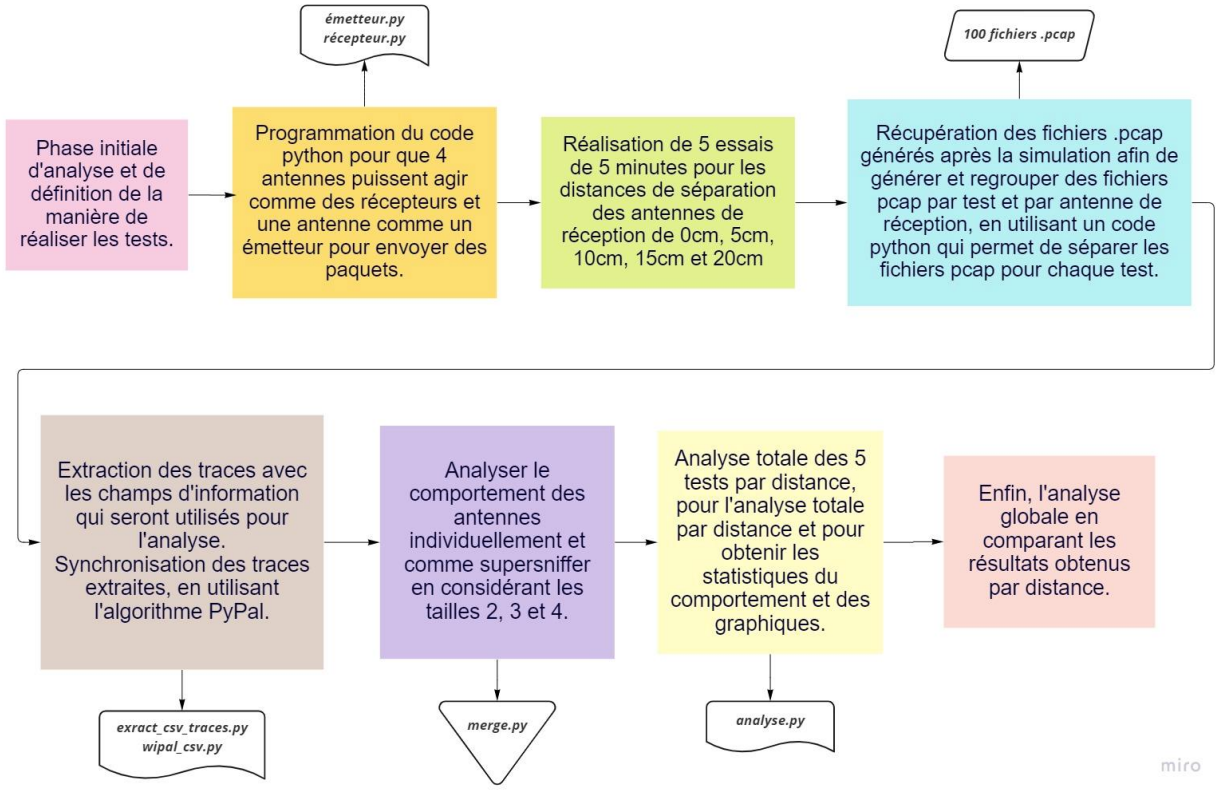


*Figure 2. Structure d'analyse du projet.*

## 4.2 Phase initiale d'analyse des tests

Programmer une antenne pour qu'elle puisse envoyer des paquets, que nous appellerons l'émetteur et 4 antennes qui agiront comme récepteurs, ensuite nous devrons effectuer la programmation de deux codes/script dans ce cas en Python pour que l'équipement puisse envoyer des trames et recevoir des trames à la suite d'un démarrage (BOOT).

Nous prendrons en compte une distance fixe de 20 mètres, entre une antenne qui servira d'émetteur de trames et les antennes réceptrices.

Pour chaque test, 5 distances différentes seront prises en compte, l'analyse par distance se fera donc comme suit :

| Distance 0cm | Distance 5cm | Distance 10cm | Distance 15cm | Distance 20cm |
|---|---|---|---|---|
| Test 1 | Test 1 | Test 1 | Test 1 | Test 1 |
| Test 2 | Test 2 | Test 2 | Test 2 | Test 2 |
| Test 3 | Test 3 | Test 3 | Test 3 | Test 3 |
| Test 4 | Test 4 | Test 4 | Test 4 | Test 4 |
| Test 5 | Test 5 | Test 5 | Test 5 | Test 5 |

*Tableau 1. Structure de l'analyse des tests.*

Chaque Sniffer procédera à 25 tests au total pour toutes les distances, c'est-à-dire que l'objectif est de pouvoir analyser 5 fois les trames envoyées par l'antenne émettrice.

Chaque test aura une durée de 5 minutes et entre chaque test. Il y aura un temps mort de 2 minutes avant de passer au test suivant plus 30 secondes de synchronisation pour la configuration de ces antennes en mode moniteur. En d'autres termes, chaque test aura une durée de 7 minutes et 30 secondes, chaque distance une durée de 37,5 minutes. Au total pour les 5 distances, nous aurons une durée totale de 187,5 minutes.

Voici un schéma de la représentation de la simulation de test :

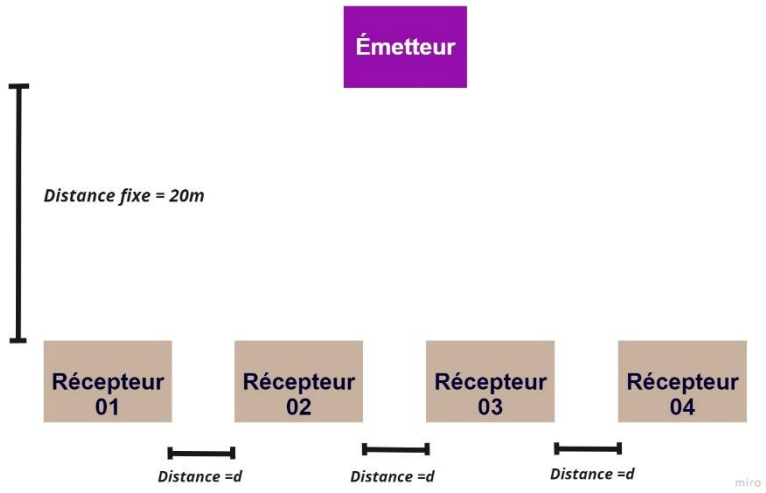


*Figure 3. Schéma de la représentation de la simulation des tests.*

L'endroit où tous les tests ont été effectués est la faculté. Par conséquent, dans ce scénario, l'analyse des résultats du projet sera effectuée en tenant compte des facteurs impliqués.

### 4.2.1. Planification de la réalisation des tests :

Nous avons trouvé que la meilleure façon d'effectuer les tests était par paires, c'est-à-dire que nous avons commencé avec les simulations des distances de 0cm et 5cm, puis avec les distances de 10cm et 15cm pour enfin finir avec la distance de 20cm. Ainsi trois tests ont été effectués dans le même environnement : le premier et deuxième test ont eu une durée de 75 min chacun et le dernier de 37,5minutes.

#### 4.2.3. Planification de l'exécution des codes/scripts par les dispositifs Raspberry Pi :

Pour cette étape, on a pensé que la meilleure façon de travailler était de planifier les tâches en utilisant le code Python et certaines fonctionnalités du système d'exploitation du Raspberry, dans notre cas Debian, afin que celles-ci soient automatisées, car pour effectuer les simulations, les dispositifs ne seront pas connectés à une source d'alimentation, mais fonctionneront complètement sans fil et avec des batteries externes.

La fonctionnalité crontab du système d'exploitation Debian Linux a été utilisée pour qu'après la mise sous tension des appareils, elle puisse être exécutée automatiquement, sachant qu'il est très important que le récepteur et l'émetteur soient synchronisés.

## 4.3 Analyse de tests - Séparation des tests

Après avoir effectué les trois tests séparément, on a obtenu trois fichiers pcap contenant les trames envoyées par l'antenne émettrice. Pour effectuer l'analyse des tests, il faut pouvoir séparer chaque test de 5 minutes et obtenir un fichier pcap pour chacun d'eux.

C'est-à-dire que l'objectif est de pouvoir disposer de 25 fichiers pcap. Nous avons programmer un code Python qui permet la séparation de chaque test, en tenant compte de leurs numéros de séquence et en nous guidant avec le temps de séparation entre chaque test. En bref, la raison de la séparation de chaque test est de vérifier les trames envoyées toutes les 7 minutes (5 minutes de simulation et deux minutes de timeout entre chaque test). Pour faciliter l'usage futur de ces données, elles ont été stockées dans un tableau Excel organisant tous les numéros de séquence et les heures de chaque test pour chaque sniffer.

## 4.4 Analyse des tests - Extraction des champs pertinentes de las trames.

Une fois la séparation des 25 fichiers pcap, nous devions analyser les champs présents dans ce fichier pour décider lesquels utiliser, nous avons donc extrait à partir de tous les fichiers uniquement les informations avec les champs nécessaires sans compromettre les données capturées pendant les tests. Nous avons donc décidé de programmer un code Python qui nous permet d'anonymiser et de tronquer les données capturées pour ne capturer que les champs qui nous permettront d'effectuer l'analyse en toute légalité.

## 4.5 Analyse des test - Vérification de la Synchronisation des temps de test

Une fois que nous avons analysé les données de chaque test pour chaque distance, l'étape suivante consiste à s'assurer que tous les tests ont des temps synchronisés, car il y a un décalage entre chaque antenne en termes de fréquence de trame. Durant notre phase de recherche et de lecture bibliographique, nous avons pu trouver un algorithme de synchronisation existant appelé PyPal qui utilise des régressions linéaires sur des fenêtres glissantes de trames de référence, c'est-à-dire un outil de fusion hors ligne pour les traces IEEE 802.11. C'est un outil fondamental pour notre analyse, l'objectif de son utilisation

est de pouvoir synchroniser toutes les trames de chaque test. En tenant compte des éléments suivants.

## 4.6 Analyse des tests - Phase de fusion des tests

Une fois que nous avons obtenu tous les fichiers pcap de chaque trame synchronisées, l'objectif est maintenant de pouvoir analyser la complétude selon la taille du super-sniffer.

Nous avons donc décidé de travailler en analysant la complétude des trames venant d'abords d'un unique sniffer, puis en combinant 2, 3 et 4.

Pour la réalisation de toute ces analyses, nous avons programmé un code "Merge" qui nous permettra d'analyser le comportement des antennes sur la base de l'analyse décrite ci-dessus. Ce code nous permettra d'obtenir 4 fichiers, dans chacun d'eux nous obtiendrons la quantité de trames reçues pour chaque comportement.

L'objectif final de cette analyse est de pouvoir comprendre comment les antennes de réception fonctionnent individuellement et dans leur ensemble, en analysant la quantité de trames reçues.

Une fois que nous avons obtenu tous les fichiers « .pcap » de chaque trame synchronisées, l'objectif est maintenant de pouvoir analyser la complétude selon la taille du super-sniffer.

## 4.7 Phase d'analyse de graphiques pour les tests de chaque distance

Après avoir obtenu la fusion de trames pour chaque comportement selon l'analyse précédente, l'objectif est maintenant de pouvoir visualiser toute ces résultats à l'aide de graphiques et de pourcentages. Cela nous permettra de discerner quel est le meilleur scénario de travail avec ce type d'antenne, en tenant compte de tous les facteurs externes. Afin de réaliser cette visualisation graphique, un code Python a été programmé pour prendre les données des fichiers obtenus dans l'analyse précédente et calculer le pourcentage de variation entre eux et les représenter à l'aide d'un graphique.

### 4.7.1. Analyse du graphe lorsque les antennes de réception sont séparées de 0 cm

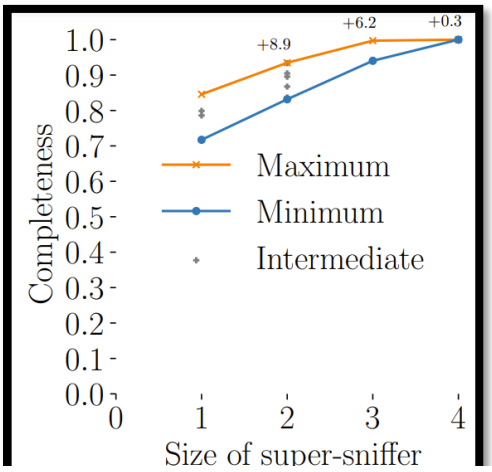


*Figure 4. Graphique représentant la complétude selon la taille du super-sniffer pour une distance de séparation de 0 cm.*

Pour comprendre tout d'abord la Figure 4 puis les suivantes, voici les outils permettant de mieux décrypter le graphique.

Axe X : la taille d'un super sniffer.
Axe Y : la complétude de paquets.

Les points bleus correspondent au minimum de complétude capturer par l'un des sniffers qui compose le super-sniffer.

De même, les points orange correspondent au maximum de complétude capturer par l'un des sniffers qui compose le super-sniffer.

Les points gris sont donc assignés aux sniffers restants que nous appelons intermédiaires.

Donc pour la taille de super-sniffer égale à 1, nous avons donc 4 super-sniffers dont 1 parmis les 4 correspondant à la complétude minimale et de même pour la complétude maximale.

Lorsque la taille de super-sniffer est égale à 4, cela signifie que nous n'avons plus qu'un seul super-sniffer : Les 4 sniffers composent alors un super-sniffer.Dans ce cas , la complétude minimale et maximale sont équivalentes.

D'après la Figure 4, nous pouvons observer pour les tailles de super-sniffer de 1 à 3 que si nous choisissons d'utiliser un seul sniffer au hasard, il est possible que nous choisissions celui qui a la valeur maximale ou même minimale de complétude. Dans ce cas, la part d'aléatoire rend l'utilisation de ces tailles non optimales puisqu'un pire cas existe : celui où nous choisissons le sniffer avec la valeur minimale de complétude.

De plus, même avec la valeur maximale, nous ne recevons que 80% des paquets dans le cas où la taille égale à 1, mais si nous augmentons la taille et que l'on passe à 2, nous améliorons le pourcentage de paquets capturés. Dans le cas minimum, nous passons de 70% à 80%, mais dans le cas maximum, nous passons de 80% à 90% (+8,9%).

De même, si nous augmentons la taille et que l'on passe à 3. Dans le cas minimum, nous passons de 80% à 90%, mais dans le cas maximum, nous passons de 90% à 96,2% (+6,2%).

Nous pouvons encore améliorer cette complétude et prendre une taille égale à 4 : nous combinons alors tous nos sniffers pour ne former qu'un unique super-sniffer. Les résultats obtenus dans ce cas, semblent optimaux puisque nous atteignons une unique complétude garantit qui converge vers une valeur très proche de 100%.

Finalement, si l'on souhaite obtenir la complétude maximale dans le cas où la distance de séparation est de 0 cm entre les sniffers, il faut choisir la taille du super-sniffer la plus grand possible : dans notre cas, la taille 4 est la plus optimal puisque nous évitons alors la part d'aléatoire, et obtenons la plus grande valeur possible de complétude.

### 4.7.1. Analyse du graphe lorsque les antennes de réception sont séparées de 5 cm

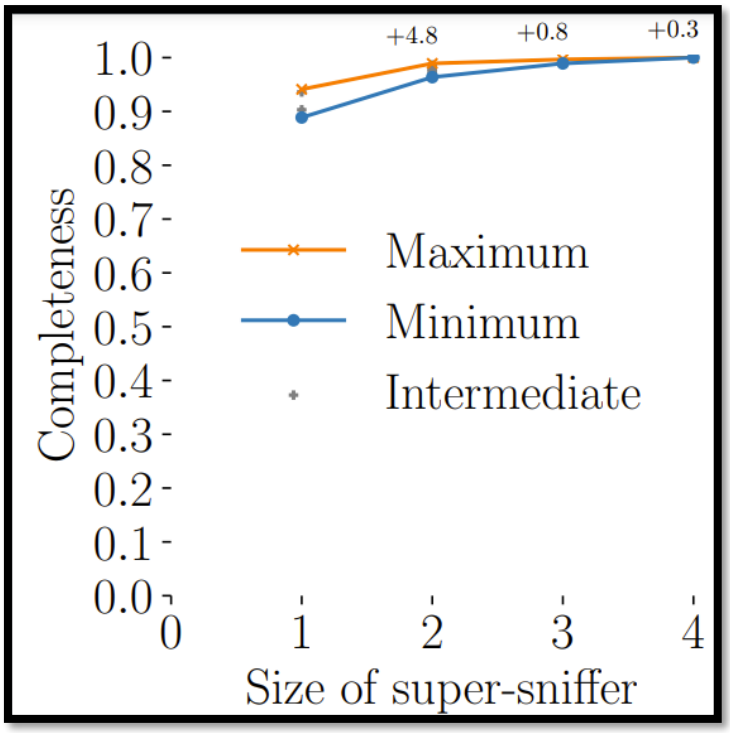


*Figure 5. Graphique représentant la complétude selon la taille du super-sniffer pour une distance de séparation de 5 cm.*

D'après la Figure 5, nous pouvons observer pour les tailles de super-sniffer de 1 à 3 que si nous choisissons d'utiliser un seul sniffer au hasard, il est possible que nous choisissions celui qui a la valeur maximale ou même minimale de complétude. Dans ce cas, la part d'aléatoire rend l'utilisation de ces tailles non optimales puisqu'un pire cas existe : celui où nous choisissons le sniffer avec la valeur minimale de complétude.

De plus, même avec la valeur maximale, nous ne recevons que 94% des paquets dans le cas où la taille égale à 1, mais si nous augmentons la taille et que l'on passe à 2, nous améliorons le pourcentage de paquets capturés. Dans le cas minimum, nous passons 88% à 96%, mais dans le cas maximum, nous passons de 94% à 98,8% (+4,8%).

De même, si nous augmentons la taille et que l'on passe à 3. Dans le cas minimum, nous passons de 96% à 98%, mais dans le cas maximum, nous passons de 98,8% à 99,6% (+0,8%).

Nous pouvons encore améliorer cette complétude et prendre une taille égale à 4 : nous combinons alors tous nos récepteurs pour ne former qu'un unique super-sniffer. Les résultats obtenus dans ce cas, semblent optimaux puisque nous atteignons une unique complétude garantit qui converge vers une valeur très proche de 100%.

Finalement, si l'on souhaite obtenir la complétude maximale dans le cas où la distance de séparation est de 5 cm entre les antennes de réception, il faut choisir la taille du super-sniffer la plus grande possible : dans notre cas, la taille 4 est la plus optimal puisque nous évitons alors la part d'aléatoire, et obtenons la plus grande valeur possible de complétude.

### 4.7.2. Analyse du graphe lorsque les antennes de réception sont séparées de 10 cm

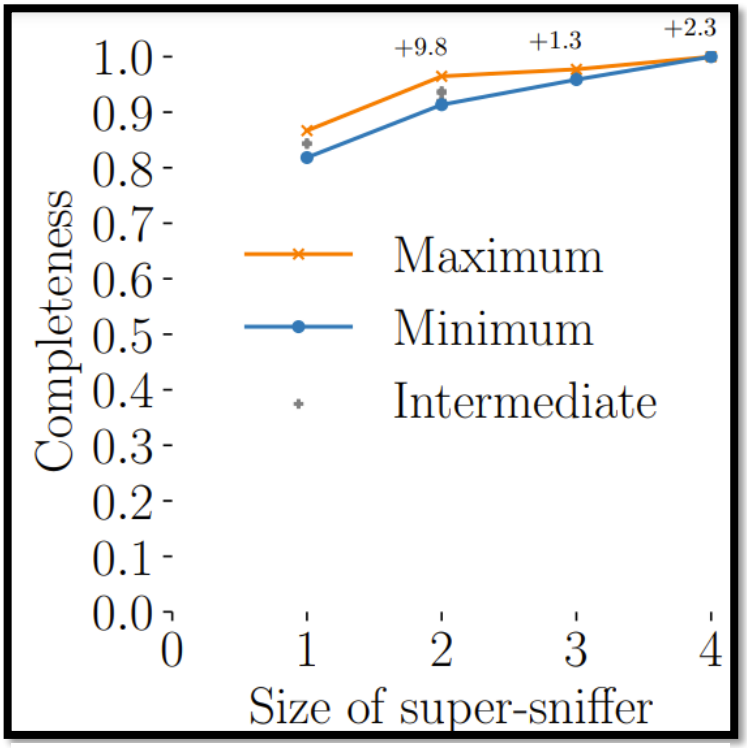


*Figure 6. Graphique représentant la complétude selon la taille du super-sniffer pour une distance de séparation de 10 cm.*

D'après la Figure 6, nous pouvons observer pour les tailles de super-sniffer de 1 à 3 que si nous choisissons d'utiliser un seul sniffer au hasard, il est possible que nous choisissions celui qui a la valeur maximale ou même minimale de complétude. Dans ce cas, la part d'aléatoire rend l'utilisation de ces tailles non optimales puisqu'un pire cas existe : celui où nous choisissons le sniffer avec la valeur minimale de complétude.

De plus, même avec la valeur maximale, nous ne recevons que 86% des paquets dans le cas où la taille égale à 1, mais si nous augmentons la taille et que l'on passe à 2, nous améliorons le pourcentage de paquets capturés. Dans le cas minimum, nous passons de 81% à 91%, mais dans le cas maximum, nous passons de 86% à 93,8% (+9,8%).

De même, si nous augmentons la taille et que l'on passe à 3. Dans le cas minimum, nous passons de 91% à 93%, mais dans le cas maximum, nous passons de 93,8% à 95,1% (+1,3%).

Nous pouvons encore améliorer cette complétude et prendre une taille égale à 4 : nous combinons alors tous nos antennes de réception pour ne former qu'un unique super-sniffer. Les résultats obtenus dans ce cas, semblent optimaux puisque nous atteignons une unique complétude garantit qui converge vers une valeur très proche de 100%.

Finalement, si l'on souhaite obtenir la complétude maximale dans le cas où la distance de séparation est de 10 cm entre les antennes de réception, il faut choisir la taille du super-sniffer la plus grande possible : dans notre cas, la taille 4 est la plus optimal puisque nous évitons alors la part d'aléatoire, et obtenons la plus grande valeur possible de complétude.

### 4.7.3. Analyse du graphe lorsque les antennes de réception sont séparées de 15 cm

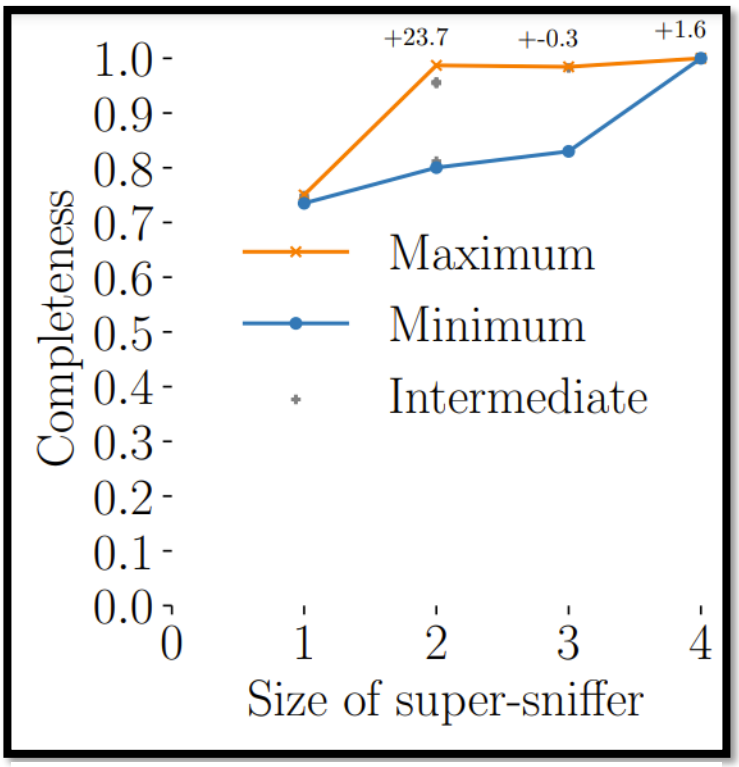


*Figure 7. Graphique représentant la complétude selon la taille du super-sniffer pour une distance de séparation de 15 cm.*

D'après la Figure 7, nous pouvons observer pour les tailles de super-sniffer de 1 à 3 que si nous choisissons d'utiliser un seul sniffer au hasard, il est possible que nous choisissions celui qui a la valeur maximale ou même minimale de complétude. Dans ce cas, la part d'aléatoire rend l'utilisation de ces tailles non optimales puisqu'un pire cas existe : celui où nous choisissons le sniffer avec la valeur minimale de complétude.

De plus, même avec la valeur maximale, nous ne recevons que 75% des paquets dans le cas où la taille égale à 1, mais si nous augmentons la taille et que l'on passe à 2, nous améliorons le pourcentage de paquets capturés. Dans le cas minimum, nous passons de

72% à 80%, mais dans le cas maximum, nous passons de 74% à 97,7% (+23,7%).
De même, si nous augmentons la taille et que l'on passe à 3. Dans le cas minimum, nous passons de 80% à 82%, mais dans le cas maximum, nous passons de 97,7% à 97,5% (-0,3%). Cette légère diminution est sûrement due à des perturbations ou interférences dans le milieu.

Nous pouvons encore améliorer cette complétude et prendre une taille égale à 4 : nous combinons alors tous nos antennes de réception pour ne former qu'un unique super-sniffer. Les résultats obtenus dans ce cas, semblent optimaux puisque nous atteignons une unique complétude garantit qui converge vers une valeur très proche de 100%.

Finalement, si l'on souhaite obtenir la complétude maximale dans le cas où la distance de séparation est de 15 cm entre les antennes de réception, il faut choisir la taille du super-sniffer la plus grande possible : dans notre cas, la taille 4 est la plus optimal puisque nous évitons alors la part d'aléatoire, et obtenons la plus grande valeur possible de complétude.

### 4.7.4. Analyse du graphe lorsque les antennes de réception sont séparées de 20 cm

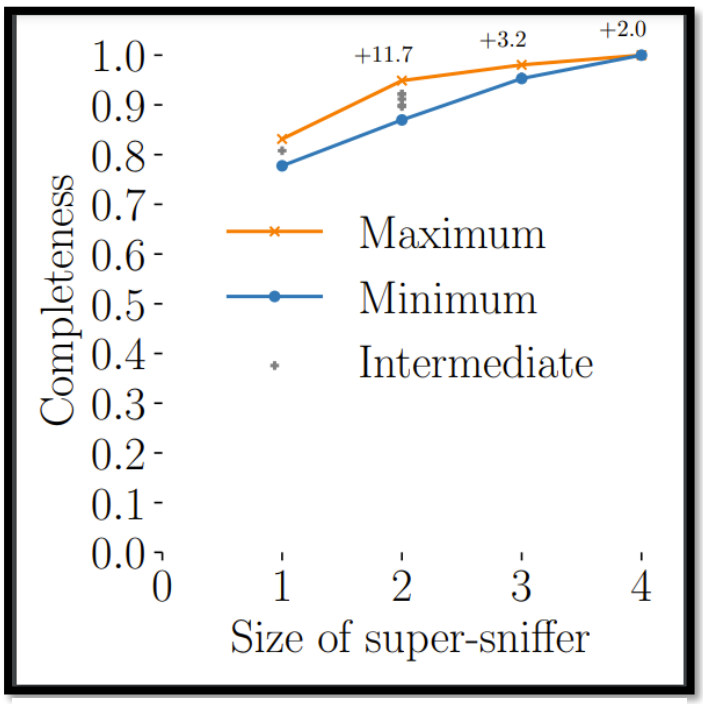


*Figure 8. Graphique représentant la complétude selon la taille du super-sniffer pour une distance de séparation de 20 cm.*

D'après la Figure 8, nous pouvons observer pour les tailles de super-sniffer de 1 à 3 que si nous choisissons d'utiliser un seul sniffer au hasard, il est possible que nous choisissions celui qui a la valeur maximale ou même minimale de complétude. Dans ce cas, la part d'aléatoire rend l'utilisation de ces tailles non optimales puisqu'un pire cas existe : celui où nous choisissons le sniffer avec la valeur minimale de complétude.

De plus, même avec la valeur maximale, nous ne recevons que 81% des paquets dans le cas où la taille égale à 1, mais si nous augmentons la taille et que l'on passe à 2, nous améliorons le pourcentage de paquets capturés. Dans le cas minimum, nous passons de 77% à 86%, mais dans le cas maximum, nous passons de 81% à 92,7% (+11,7%).

De même, si nous augmentons la taille et que l'on passe à 3. Dans le cas minimum, nous passons de 86% à 94%, mais dans le cas maximum, nous passons de 92,7% à 95,9% (+3,2%).

Nous pouvons encore améliorer cette complétude et prendre une taille égale à 4 : nous combinons alors tous nos antennes de réception pour ne former qu'un unique super-sniffer. Les résultats obtenus dans ce cas, semblent optimaux puisque nous atteignons une unique complétude garantit qui converge vers une valeur très proche de 100%.

Finalement, si l'on souhaite obtenir la complétude maximale dans le cas où la distance de séparation est de 20 cm entre les antennes de réception, il faut choisir la taille du super-sniffer la plus grande possible : dans notre cas, la taille 4 est la plus optimal puisque nous évitons alors la part d'aléatoire, et obtenons la plus grande valeur possible de complétude.

## 4.8 Analyse finale

Après les simulations et les analyses individuelles, nous pouvons maintenant faire une analyse globale de notre projet.

La complétude de la trame pour 0 cm est de 71% minimum et 84,5% maximum pour un seul sniffer, alors que pour la distance de 5cm elle est de 88,8% minimum et 94% maximum, pour la distance de 10cm la complétude minimum est de 81,8% et 86,6% maximum, pour 15cm 73,5% minimum et 75% maximum et enfin pour 20cm 77,77% minimum et 83,1% maximum.

Nous pouvons donc voir que malgré la courte distance de 0 cm pour le premier cas, bien qu'il soit exposé à plus d'interférences, le pourcentage de complétude est plus élevé que pour les distances de 15 cm et 20 cm, et en général nous nous attendions à avoir de moins bons résultats pour cette distance compte tenu de l'impact des interférences.

Pour le cas de 5 cm nous pouvons voir qu'il présente un meilleur comportement en obtenant 94% de complétude maximale en comparaison avec les distances de 10 cm qui présente 86.6% maximum, la distance de 15 cm présente 75% maximum et finalement pour 20 cm une complétude de 83.1% maximum.

De même, lorsque nous évaluons la complétude des trames pour un super-sniffer de taille 2, la complétude maximale de trame pour la distance 0 cm est de 93,4%, pour 5 cm 94%, pour 10 cm 93,8%, pour 15 cm 93% maximum et pour 20 cm 92% maximum. Nous pouvons donc constater à nouveau que le pourcentage de complétude pour 0 cm est encore élevé.

Dans le cas d'un super-sniffer de taille 3, la complétude maximale de trames pour la distance 0 cm est de 97%, pour 5 cm 98%, pour 10 cm 97,7%, pour 15 cm 98% et pour 20 cm 98,04%. Ici la différence de pourcentage de complétude des trames entre les distances est minime, la distance dans ce cas n'a pas un grand impact.
Ensuite, notre expérimentation, nous indique que quelle que soit la distance de séparation lorsque la taille du super-sniffer est de 4 (combinaison des 4 antennes de réception), la complétude obtenue reste toujours la plus optimale : une valeur très proche de 100%.

Nous pouvons également remarquer, que nos résultats indiquent qu'une distance de séparation se détache des autres à travers ses valeurs de complétude. En effet, les valeurs obtenues pour la complétude restent les plus élevés quelle que soit la taille choisie pour le sniffer en comparaison aux autres distances de séparation. Nous pouvons alors conclure que cette distance parmi celle étudié se retrouve être la plus optimal.

Nos résultats ont été aussi, une fois très clairement, impactés par l'impact de perturbation et d'interférence sans fil dans le milieu comme indiqué dans l'analyse de la distance 15 cm.

# 5 Conception

## 5.1 Phase de programmation des antennes

Nous avons choisi de réaliser la programmation en utilisant le langage de programmation Python et en utilisant les bibliothèques suivantes : scapy, subprocess et time, qui permettront aux antennes d'envoyer et de recevoir des trames.
Dans le cas des antennes de transmission, nous avons programmé une fonction Bacon, qui permet d'envoyer des trames balise, trames de gestion dans les WLAN basés sur IEEE 802.11 qui contient toutes les informations sur le réseau. Ainsi, les trames de balise seront transmises périodiquement pendant la période de simulation.
Tout d'abord, nous importons tous les modules requis, puis nous définissons certaines variables. En utilisant Scapy, nous allons programmer la couche Dot11 aves le paramètre type = 0. Cela signifie simplement que c'est une trame de gestion et le sous-type = 8 signifie que c'est une trame de balise, aussi bien en tant qu'adresse source qu'en tant qu'adresse de destination de diffusion.
Scapy fonctionne avec des "couches", nous avons défini donc la couche Dot11 avec quelques paramètres. Le paramètre type = 0 signifie simplement que c'est une trame de gestion et le sous-type = 8 signifie que c'est une trame de balise.
Ensuite, nous avons créé la couche Dot11Elt qui prend en entrée le SSID de l'antenne réceptrice ainsi que d'autres entrées liées au SSID. Enfin, nous empilons les couches dans l'ordre et ajoutons une couche RadioTap à la fin.
La logique de notre code programmé est présentée dans les diagrammes de flux suivants:

**Diagramme de flux pour la programmation de l'émetteur :**

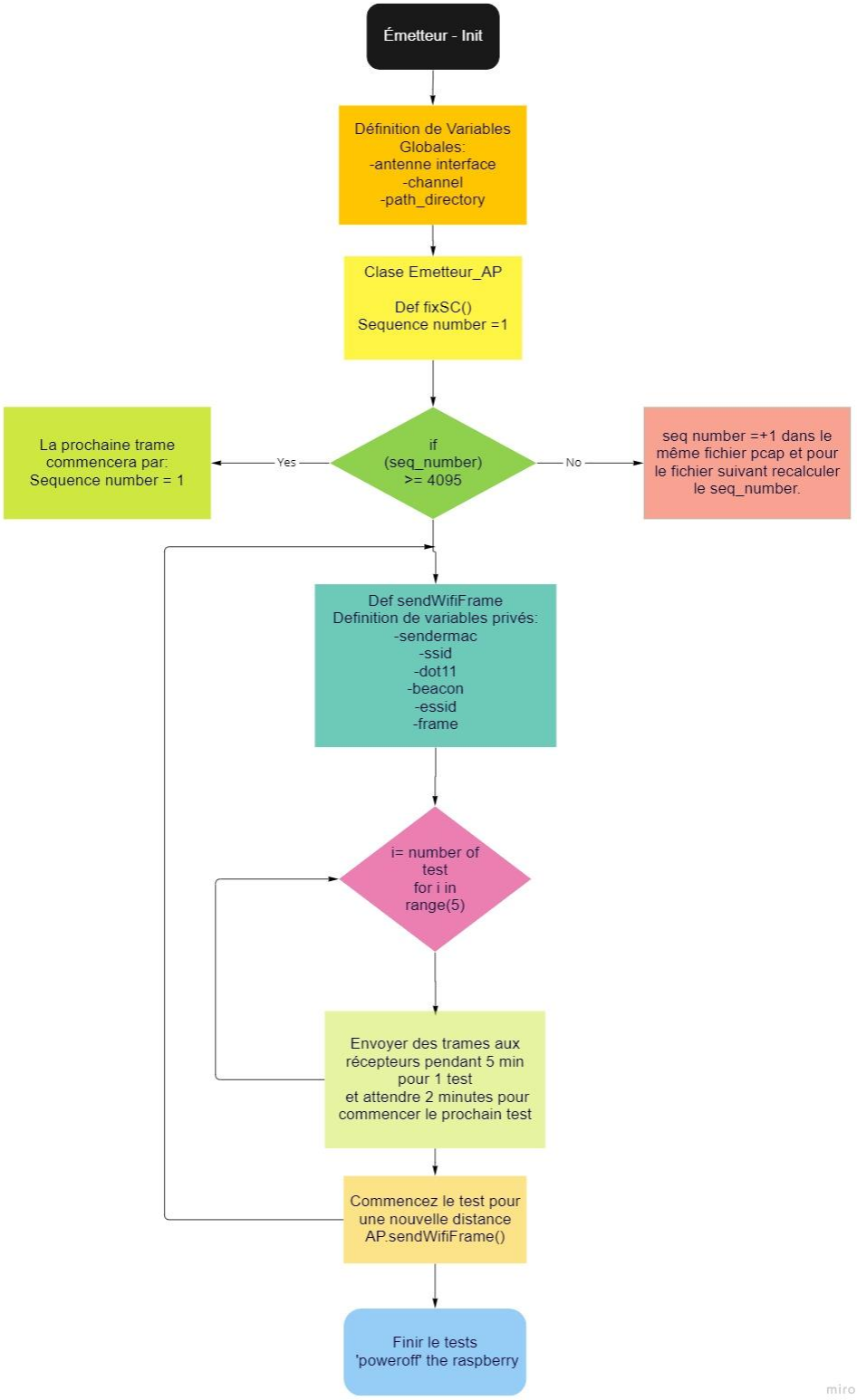


*Figure 9. Diagramme de flux pour la programmation de l'émetteur.*

**Diagramme de flux pour la programmation du récepteur :**

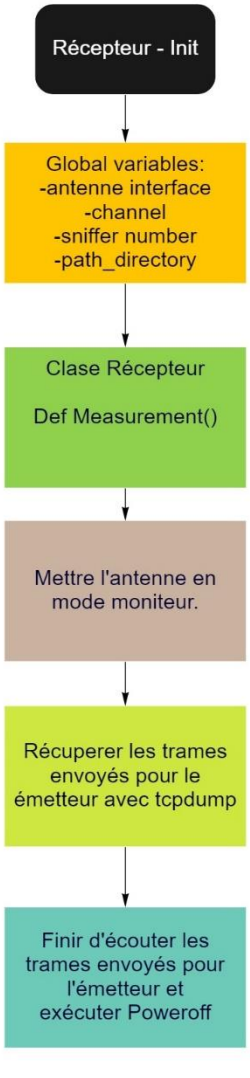


*Figure 10. Diagramme de flux récepteur.*

## 5.2 Phase de réalisation des expériences :

Après avoir programmé en Python le fait que les 4 antennes fonctionnent comme des récepteurs et une comme un émetteur, nous avons commencé par la simulation des tests, pour des distances de 0cm, 5cm, 10cm, 15cm et 20cm. Dans l'image suivante, nous pouvons visualiser une simulation effectuée pour une distance de 10cm entre les antennes, alors que l'antenne de transmission était située à 20m.
L'image suivante montre un exemple de simulation effectuée pour une distance de 10 cm.

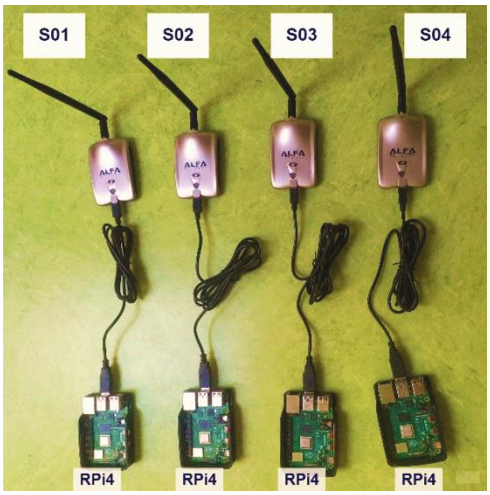


*Figure 11. Simulation de test pour la distance de 10cm.*

Pour l'exécution automatique des codes programmés en Python, nous avons utilisé crontab, une fonctionnalité du système d'exploitation Linux qui permet de programmer des tâches automatiquement avec une heure spécifique après le redémarrage de l'équipement.
Il a été programmé de manière que le code puisse être exécuté 90 secondes après le démarrage pour l'antenne émettrice, tandis que pour les antennes réceptrices après 60 secondes. Pendant la phase de simulation, nous avons synchronisé les deux temps, en allumant d'abord l'antenne d'émission puis l'antenne de réception.

## 5.3 Phase de récupération des données

Une fois la simulation des 25 tests terminée, l'objectif est d'obtenir 100 fichiers pcap qui représentent les 5 répétitions de test effectuées pendant 5 minutes, afin de pouvoir les regrouper par distances et par antenne de réception.
Pour ce faire, nous avons regroupé tous les tests effectués dans un tableau Excel, en les triant par leur numéro de séquence initial, leur numéro de séquence final, le temps de test initial et le temps de test final. Nous avons fini par programmer un code Python qui nous permet de les séparer pour chaque test. Afin de réaliser tout cela, nous avons utilisé la bibliothèque Python subprocess et time.

## 5.4 Phase de récupération et synchronisation des trames

Une fois toutes les données organisées, l'étape suivante consiste à extraire les données qui seront utilisées pour l'analyse, en ne considérant que les champs utiles, tels que : l'heure, le numéro de séquence, le canal utilisé et la source mac. Nous avons obtenu 4 fichiers pour chaque antenne de transmission et cela pour chaque test.

Pour la phase de synchronisation, nous avons utilisé l'algorithme Pypal issue de Wypal comme référence et nous avons utilisé le code Python existant développer dans notre faculté.[1]

Ce code nous permettra de synchroniser toutes les traces obtenues. Pour cette dernière analyse, nous avons programmé un code Python utilisant les bibliothèques Python time, panda et csv.

## 5.5 Phase d'analyse

Une fois les trames fusionnées pour évaluer leur comportement, nous pouvons enfin les analyser. Pour ce faire, nous avons programmé un code Python qui nous permet de récupérer les données obtenues précédemment, de calculer le nombre de paquets minimum et maximum reçus par chaque antenne de réception et en considérant les antennes de réception comme des super-antennes de réception, ainsi qu'un graphique qui nous permet de visualiser les pourcentages delta entre chacune d'entre elles, pour lequel nous avons principalement utilisé les bibliothèques suivantes : matplotlib, numpy et pandas.
Le diagramme de flux suivant explique la logique utilisée pour la programmation Python :

[1] Mohammad Imran Syed, Anne Flandenmuller, and Marcelo Dias De Amorim. 2022. PyPal: Wi-Fi Trace Synchronization and Merging Python Tool. LIP6 UMR 7606, UPMC Sorbonne Université, France. Retrieved May 7, 2022 from: https://hal.archivesouvertes.fr/hal-03618014

**Diagramme de flux pour la programmation de la partie analyse :**

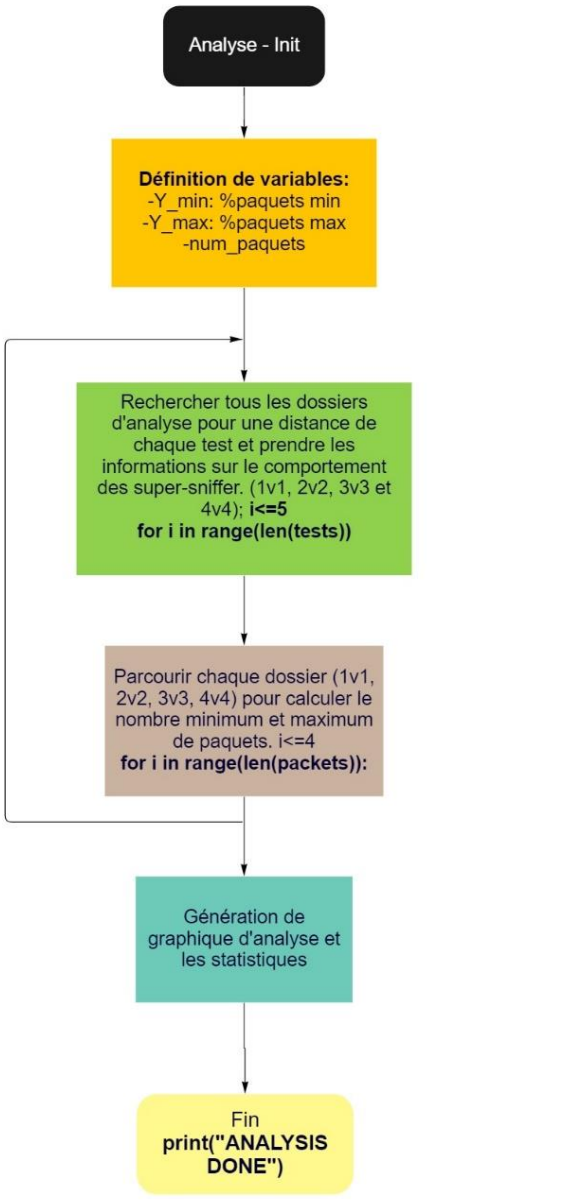


*Figure 12. Diagramme de flux pour la partie analyse.*

# 6 Compte rendu

**Le déroulement :**

Le déroulement de notre projet a tout d'abord débuté par un élargissement de nos connaissances des réseaux sans-fil pour finalement nous concentrer principalement sur les transmissions Wi-Fi. Nos recherches bibliographiques, quant à elles nous ont aidé à mieux comprendre les problèmes que rencontrent les transmissions Wi-Fi.

À la suite de ces recherches, nous avons cerné le but principal de notre projet qui est d'étudier les transmissions Wi-Fi à travers une simulation qui met en place la notion de distance entre les terminaux qui capture le trafic Wi-Fi et le type d'adaptateur Wi-Fi utilisé.

**La réalisation :**

Notre rôle consistait à réaliser cette simulation puis étudier ces résultats afin d'identifier l'impact de distance entre les sniffers sur le nombre de trames capturer et définir le meilleur cas pour arriver à la complétude des trames.

L'étape suivante était de se familiariser avec les équipements mis à notre disposition par l'encadrant pour réaliser la simulation d'une transmission Wi-Fi.

Grâce aux livres trouvés pendant la recherche bibliographique et aux tutoriels sur Internet nous avons pu configurer et installer les systèmes d'exploitation des Raspberry Pi 4.

Ensuite, nous avons essayé d'installer la version open source du pilote nécessaire pour la gestion d'adaptateurs Wi-Fi «TP Link » mais cela a créé des problèmes de compatibilité entre la dernière version du système d'exploitation et la dernière version du pilote présente en open source. Après plusieurs tentatives veines, afin de régler le problème avec notre encadrant, il a été convenu de changer d'adaptateur Wi-Fi.

Pour la suite du projet nous avons utilisé les adaptateurs « ALFA-NETWORK-AWUS051 NH » qui ne nécessitent pas l'installation de pilotes.

S'en suit la phase de programmation, où nous avons choisi d'utiliser le langage de programmation Python, car riche en librairies et modules traitant les différentes couches de la pile de protocoles.

Nous avons commencé par programmer une antenne émettrice Wi-Fi qui génère et envoie des trames de Wi-Fi frames avec des numéros de séquence ordonner tout en capturant le trafic qu'il a émis. Ce dernier fera office d'émetteur dans la simulation et il s'exécute sur un Raspberry Pi 4 à la suite d'un démarrage.

Puis nous avons programmé un script qui permet de capturer les trames émises par l'antenne émettrice et de les sauvegarder au format «. pcap ». Ce script s'exécute également à la suite du démarrage sur chacun des quatre Raspberry Pi qui feront office de sniffer/récepteur dans la simulation.

La simulation regroupe alors un émetteur et quatre sniffer/récepteurs séparés entre eux par différentes distances.

Au moment de la rédaction de ce rapport, nous avions déjà fini la simulation, le traitement des données récoltées : Le découpage des fichiers «. pcap » , la synchronisation, la sauvegarde sous forme de fichier csv, la concaténation ,la fusion , la suppression des trames dupliquées pour assurer la complétude, l'analyse, la comparaison des différentes complétude des trame selon les combinaisons entre les sniffers et la génération des graphes qui indique les résultats finaux de la simulation.

**La validation :**

L'objectif est maintenant de pouvoir valider tous ces résultats obtenus en les confrontant à nos connaissances théoriques. Cela nous a permis de discerner quel est le meilleur scénario de travail avec ce type d'antenne.

Nos résultats issus de l'expérimentation, nous indiquaient donc que quelle que soit la distance de séparation lorsque la taille du super-sniffer est de 4 (combinaison des 4 antennes de réception), la complétude obtenue reste toujours la plus optimale : une valeur très proche de 100%. Cela reste donc cohérent par rapport à ce qui était attendu théoriquement et valide donc bien nos résultats.

Ensuite, pour a complétude des trames pour 0 cm, le pourcentage de complétude était plus élevé que pour les distances de 15 cm et 20 cm, et en général nous nous attendions à avoir de moins bons résultats pour cette distance compte tenu de l'impact des interférences.

Pour le cas de 5 cm nous avons pu voir qu'il présente un meilleur comportement en obtenant 94% de complétude maximale en comparaison avec les distances de 10 cm qui présente 86.6% maximum, la distance de 15 cm présente 75% maximum et finalement pour 20 cm une complétude de 83.1% maximum.

De même, lorsque nous avons evalué la complétude des trames pour un super-sniffer de taille 2, la complétude maximale de trame pour la distance 0 cm est de 93,4%, pour 5 cm 94%, pour 10 cm 93,8%, pour 15 cm 93% maximum et pour 20 cm 92% maximum. Nous avons pu constater à nouveau que le pourcentage de complétude pour 0 cm est encore élevé.

Dans le cas d'un super-sniffer de taille 3, la complétude maximale de trames pour la distance 0 cm était de 97%, pour 5 cm 98%, pour 10 cm 97,7%, pour 15 cm 98% et pour 20 cm 98,04%. Ici la différence de pourcentage de complétude des trames entre les distances était minimale, la distance dans ce cas n'a pas eu d'impact majeur.

Nous avons pu également remarquer, que nos résultats indiquent qu'une distance de séparation se détache des autres à travers ses valeurs de complétude. Il s'agit de la distance de séparation 5 cm. En effet, les valeurs obtenues pour la complétude restent les plus élevés quelle que soit la taille choisie pour le sniffer en comparaison aux autres distances de séparation. Nous pouvons alors conclure que cette distance parmi celle étudié se retrouve être la plus optimal.

Nos résultats ont été aussi, une fois très clairement, impactés par l'impact de perturbation et d'interférence sans fil dans le milieu comme indiqué dans l'analyse de la distance 15 cm.

Ce qui encore une fois valide un peu plus nos résultats puisque l'on observe des interférences et perturbations dans notre milieu expérimentale qui n'en n'est pas exclu.

**La livraison :**

Notre implication dans ce projet, nous a permis de mieux comprendre les comportements des réseaux sans-fils et en particulier les réseaux wifi. Le développement de ce projet, nous a donné l'opportunité d'approfondir nos connaissances sur le langage python et de nous familiarisés avec divers matériels typique réseau comme les Raspberry et les antennes.

Les différents codes utilisés, nous a aussi permis de maitriser de nouvelles librairies et modules python jusqu'ici inconnu.

Dans la 1ère partie de ce projet, nous avons grandement améliorer nos compétences en matière de recherches de ressources bibliographique.

Durant, la phase d'analyse, la plus longue de notre projet, nous avons bien discernés comment les contraintes dans un milieu sans-fil peuvent impacter la performance de notre réseau Wi-Fi crée.

Pour le traitement de nos données, nous avons appris à mieux les retranscrire en résultats statistiques cohérents afin d'avoir une représentation graphique la plus valide possible.

Avant de pouvoir mener à bien nos simulations de test, nous avons rencontrés énormément de problèmes liés au dysfonctionnement du matériel et de problèmes de compatibilité entre les pilotes et l'OS, ce qui nous a grandement retardé par rapport à notre planning initialement prévu. Cela nous a permis de nous rendre compte de la dure réalité expérimentale et que celle-ci n'est pas totalement acquises et sans embuche. En effet, désormais nous sommes conscient qu'il faut toujours prévoir une marge d'erreur afin de s'adapter à tout aléas qui pourrait affectés notre expérimentation.

# 7 Annexe A

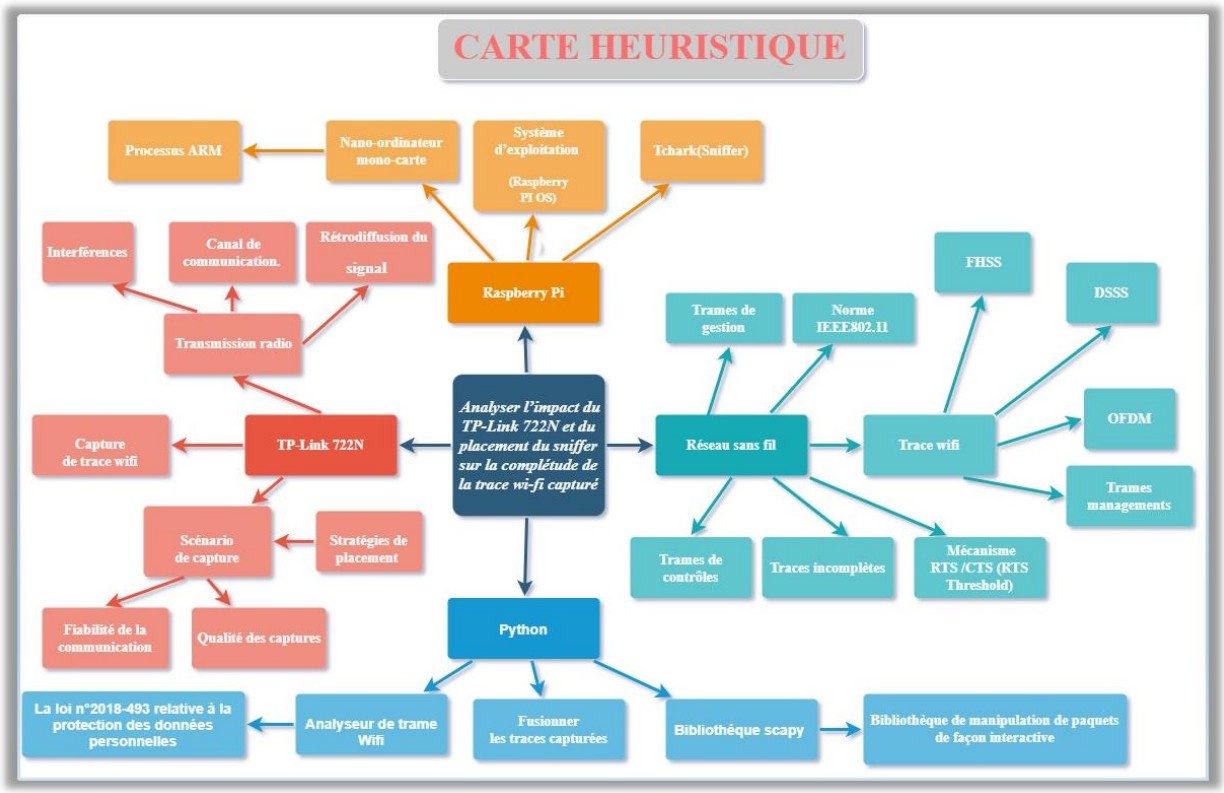


*Figure 13 Annexe A - Carte Heuristique*