Compte rendu 5 semaine de stage en mise en place d’un réseaux ad-hoc et service de journalisation

Sommaire

I. Présentation de l’entreprise
II. Outils
III. Objectifs
IV. Travail réalisé
    A. Choix des OS
    B. Construction du script
    C. Mise en place de topologie teste
        1.maj.sh
        2.Logmaj.sh
        3.Chalie.sh
        4.Delta.sh
        5.Foxtrot.sh
        6-Gamma.sh
    D. Commande utile
V. Difficultés rencontrées
    A.Choix d’OS
    B.Mise en place du script
VI. Biblio
VII. Annexe

I-Présentation de l’ONERA Toulouse

Introduction

Dans le cadre de mon stage, j’ai intégré le centre de Toulouse de l’ONERA (Office national d’études et de recherches aérospatiales), principal organisme public français de recherche dans le domaine aéronautique, spatial et de défense. Il contribue au développement scientifique et technologique du secteur et accompagne les industriels dans leurs projets d’innovation.

Présentation générale de l’ONERA

Créé en 1946, l’ONERA est un établissement public à caractère industriel et commercial (EPIC) placé sous la tutelle du ministère des Armées. Il mène des recherches amont et appliquées pour la conception et l’amélioration des aéronefs, satellites et systèmes de défense. Ses missions principales sont de développer les connaissances scientifiques, concevoir et exploiter des moyens expérimentaux, accompagner les industriels et former des doctorants et ingénieurs.

Le centre ONERA de Toulouse

Situé sur le campus scientifique de Rangueil, le centre de Toulouse se trouve au cœur d’un environnement marqué par l’industrie aéronautique et spatiale, favorisant les collaborations académiques et industrielles. Les équipes travaillent sur le traitement de l’information et les systèmes embarqués, l’électromagnétisme et les technologies radar, l’optique, la physique de l’environnement spatial et la modélisation multiphysique.

Activités et rôle

L’ONERA Toulouse conduit des recherches fondamentales et appliquées, élabore des modèles théoriques, conçoit des outils numériques et réalise des expérimentations pour améliorer la compréhension et la performance des systèmes aérospatiaux. Le centre participe à des projets nationaux et européens et accueille des doctorants et stagiaires, renforçant le lien entre recherche et formation.

II-Outils et choix matériels

Pour réaliser mon expérimentation sur un réseau mesh ad-hoc, j'ai choisi d'utiliser des **Raspberry Pi Zero W2 (voir annexe 1)** comme nœuds du réseau et une **Raspberry Pi 3 (voir annexe 2)** comme nœud maître. Ce choix a été motivé par plusieurs critères : accessibilité, flexibilité et faible consommation énergétique. Les Raspberry Pi offrent une interface simple pour développer et tester des protocoles de communication, tout en restant proches des contraintes matérielles que l'on pourrait rencontrer sur des systèmes embarqués réels.

D'autres solutions matérielles, comme les cartes Arduino ou les modules ESP32, ont été envisagées. Cependant, un ESP32 ne peut pas faire fonctionner nativement un véritable réseau ad hoc 802.11 IBSS car sa pile Wi-Fi, fournie par ESP-IDF, ne prend en charge que les modes station et softAP et n'expose pas le contrôle MAC de bas niveau nécessaire au routage ad hoc conforme aux normes. Cependant, il peut toujours former des réseaux dynamiques multi-sauts en utilisant des cadres Espressif comme ESP-MESH ou des communications entre pairs sans connexion via ESP-NOW. Dans ces configurations, les développeurs peuvent mettre en œuvre une logique de routage ou de transfert personnalisée au niveau de l'application ou de la couche maillée, ce qui permet une communication multi-nœuds à autorégénération, mais avec des limites de ressources plus strictes et moins de souplesse de protocole que les véritables mises en œuvre de MANET IBSS. Pour avoir accès à ces fonctions, nous avons besoin d'un environnement Linux.

L'utilisation de Raspberry Pi a permis de disposer d'un environnement complet, capable d'exécuter le logiciel de gestion du réseau et de simuler des scénarios réalistes, tout en restant modulaire pour l'ajout ou le retrait de nœuds. Ainsi, ce choix permet de justifier l'expérimentation sur un réseau fonctionnel et reproductible, tout en offrant un cadre proche des applications industrielles pour lesquelles la scalabilité et la robustesse des communications sont des facteurs clés.

III- Contenu du stage

L'objectif principal de mon stage a été de développer et tester des applications fonctionnant sur des **Raspberry Pi Zero 2 W** au sein d'un réseau mesh ad-hoc. Ce réseau repose sur des nœuds autonomes communiquant entre eux et coordonnés par un **nœud maître** basé sur une Raspberry Pi 3. L'expérimentation visait à étudier à la fois la gestion du trafic et la génération de données au sein du réseau.

Le stage s'est articulé autour de deux applications distinctes mais complémentaires. La première application a été conçue pour **surveiller le réseau**. Elle collecte et affiche des informations sur l'état des nœuds, la disponibilité des liaisons et le trafic passant par chaque nœud. Ces données permettent de visualiser le fonctionnement du réseau en temps réel et d'identifier d'éventuels goulots d'étranglement ou pertes de paquets. Cette application a été développée avec un souci de légèreté et de compatibilité avec les ressources limitées des Raspberry Pi Zero 2 W.

La seconde application a pour rôle de **générer du trafic applicatif** afin de tester le réseau. Elle envoie différents types de données, allant de simples messages texte à des flux vidéo ou des pings entre nœuds. L'objectif est de simuler des scénarios d'utilisation réalistes et de mesurer les performances du réseau en conditions variées. La combinaison des deux applications permet d'analyser la robustesse, la réactivité et l'efficacité du réseau mesh.

Le choix des Raspberry Pi Zero 2 W s'est avéré pertinent pour ce stage, car ces cartes offrent un compromis entre puissance de calcul et faible consommation, tout en permettant de simuler un réseau de nœuds relativement dense. Le nœud maître sur Raspberry Pi 3 centralise la collecte de données et la coordination du réseau, garantissant une expérimentation fiable et reproductible.

Au cours du stage, le développement a inclus la programmation des applications, la configuration des nœuds et la mise en place d'un protocole de communication simple mais efficace. Les tests ont permis de valider le bon fonctionnement du réseau, de mesurer la latence et le débit des communications, et de générer des rapports détaillés sur le comportement du réseau selon différentes charges applicatives.

Cette expérience m'a permis de mettre en pratique des compétences en **programmation embarquée**, en **réseaux ad-hoc** et en **analyse de performance**, tout en travaillant dans un environnement proche des contraintes industrielles. Le stage a également renforcé ma compréhension des défis liés aux systèmes distribués et à la communication entre multiples nœuds dans un réseau dynamique.

IV-Travaux réalisé

A-Choix des OS

Le choix du système d'exploitation constitue une étape clé pour le développement et le déploiement des applications sur un réseau de Raspberry Pi. Dans le cadre de mon stage, j'ai étudié plusieurs distributions afin de sélectionner celle qui offrirait la meilleure stabilité, compatibilité matérielle et facilité de configuration pour les nœuds du réseau et le nœud maître.

Plusieurs options ont été évaluées. La distribution Trixie a été envisagée en raison de ses fonctionnalités modernes et de son support pour les dernières versions des logiciels. Cependant, elle a été écartée car il s'agit d'une licence récente, encore peu utilisée dans les communautés techniques. La documentation disponible reste limitée, ce qui aurait pu poser des difficultés pour l'installation, la configuration et le débogage des applications développées pour le stage.

La distribution Ubuntu Server a également été considérée. Elle bénéficie d'une large base d'utilisateurs et d'un suivi actif. Toutefois, son utilisation sur les Raspberry Pi a été rendue compliquée par des problèmes liés à l'outil Imager, entraînant des erreurs d'installation et des incompatibilités avec certains périphériques. Ces difficultés auraient allongé le temps de mise en place et introduit des risques pour la stabilité des expérimentations.

Finalement, le choix s'est porté sur la distribution Debian Bookworm, adaptée aux Raspberry Pi et relativement stable. Bien que cette licence soit plus ancienne, elle présente plusieurs avantages décisifs : la documentation est abondante, les outils de paramétrage sont simples à utiliser et la compatibilité avec le matériel Raspberry Pi est éprouvée. Cette distribution permet d'installer rapidement les bibliothèques nécessaires et de déployer les scripts expérimentaux sans rencontrer de blocages techniques majeurs.

L'utilisation de Debian Bookworm a également facilité l'intégration avec le nœud maître basé sur une Raspberry Pi 3, garantissant une uniformité dans l'environnement logiciel et une meilleure gestion des communications au sein du réseau mesh. Ce choix contribue à assurer la reproductibilité des tests, la fiabilité des mesures de trafic et la simplicité des mises à jour logicielles au cours du stage.

En résumé, Debian Bookworm a été retenu pour sa stabilité, sa compatibilité matérielle, sa documentation complète et sa facilité de configuration, répondant ainsi aux besoins spécifiques du projet. Ce choix s'inscrit dans une logique de minimisation des risques techniques et de maximisation de l'efficacité lors de la mise en œuvre des applications sur les nœuds du réseau.

B-Construction d’un script

1-maj.sh

Ce document représente la première mise à jour du scripte d’origine qui m'avait été donnée par mon doctorant en chef, Hugo. Il avait cherché à développer un script qui lors de son exécution allait installer des paquets de manière automatique tels que OLSR et des dépendances tiers de manière automatique afin de mettre en place un réseau ad-hoc en IBSS avec des adresses définies. Ce script était opérant sur Trixie, mais étant trop complexe à soutenir, j’ai tenté de l'adapter pour Bookworm.

2-Logmaj.sh

Suite à des tentatives d’adaptations du script j’ai commencé à me rendre compte que sans la journalisation des tests que j’effectué je ne pouvais pas vraiment comprendre les erreurs que je faisais. J’ai donc fait la mise à jour majeure d'ajouter un système de logs dans le code. Grâce à celui-ci j’ai pu identifier plus rapidement les informations qui venaient à me porter préjudice.
Après vraiment beaucoup d'essai sur cette version, j’ai pu faire obtenir à l’interface Wlan0 une double interface 169.254.x.x et 10.0.0.x mais grâce à la magie, les ping pouvaient marcher même sans fonctionnement de olsr.

3-Charlie.sh

Dans ce code, j’ai fait en sorte de mettre en route OLSR en route et suite à cela j’ai pu découvrir les routes voisines.

4-Delta.sh

Cette évolution m'a fait mettre en place un système de ping automatique afin de créer les routes entre les différents nœuds de manière passive et automatique.

5-Foxtrot.sh

Mise en place d’un serveur NTP (date et heure synchronisée) sur un nœud maître, (10.0.0.1 et Pi 3). Passage en simple Ip, juste 10.0.0.x qui posait de vrai problème avec la synchro .json .

6-Gamma.sh

Version final à ce jour avec olsrv2, qui contient toute les évolutions précédemment évoqué, ce script permet de mettre n'importe quel puce Raspberry Pi en mode réseaux mesh + IBSS avec la bonne adresse 10.0.0.x, crée les routes via des ping automatiques, et il possède un système de log qui est récupéré par le noeud maître à la fin de l'exécution

7-Hotel.sh

Version évolutive de Gamma.sh avec un changement majeur, changement de protocole passage de Olsr vers BATMAN

C- Mise en place de topologie teste

Mise en place de topologie afin de tester dans la nature les cartes et la robustesse du script.

Topologie 1 : Bureau

Mise en place de cette topologie **(Voir annexe 3)** afin de prendre des valeurs de référence et vérifier que les puces soient bien logées dans les fichiers .json .

Topologie 2 : Sous sol

Mise en place de cette topologie **(Voir annexe 4)** afin de visualiser une formation en bus et la solidité des routes.

Topologie 3 : BATMAN-adv

Mise en place de cette topologie **(Voir annexe 5)** afin de visualiser une formation en ligne droite avec une cible mouvante (Entourer en noir) qui ce déplace dans le périmètre des autre puce, afin de tester la souplesse des routes.

D-Commande utile

Pour voir l'état des routes: *ip neigh show dev wlan0*
Pour voir l'état d'un service: *systemctl status olsrv2.service*
Pour lancer le service de journalisation pendant 10 min: *sudo mesh-run-icmp*
Pour voir les commandes utilisable dans cette version de Olsr: */usr/sbin/olsrd2_dynamic --schema*
Uniquement pour batman : *mesh-info*

V-Difficultés rencontrées

A-Choix d'OS

Dans la première partie du stage j'ai commencé à chercher quel OS utilisé afin de mettre en place l'infrastructure IBSS + ad-hoc et pour cela j'ai essayé différente licence différente, et j'ai eu un mal fou à mettre en route le développement du script car j'ai avancé à l'aveugle quelque jour, a cause d'un manque d'interface visuel. Et même après, j'ai eu énormément de mal avec cette décision. Je ne citerai qu'un seul énorme problème auquel j'ai fait face, l'identification sur Ubuntu 22.04. J'ai du chercher jusqu'à crypté directement un mot de passe en dehors de ubuntu car il y avait un bug lors du boot qui empêchait le changement de mots de passe avec l'interface user classique. Tout cela pour ne même pas utiliser Ubuntu.

B-Mise en place du script

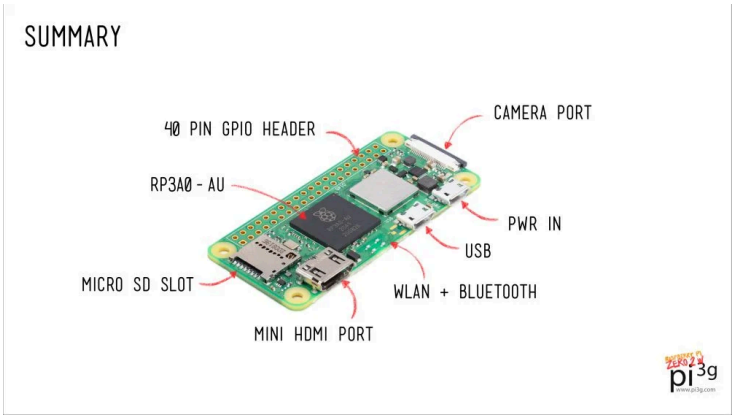
Lors de la mise en place du script j'ai fait plein d'erreur, comme oublié de changer la date, oublier de mettre des numéro lors du lancement du script mais j'ai fais un erreur qui m'as couté presque une semaine de galère infini, un retour à la ligne dans un bloc, lors l'écriture d'un .sh j'ai mis sans faire exprès un retour à la ligne et je n'ai trouvé l'erreur qu'une semaine après ce qui m'as value des recherches approfondie pour rien.

VI-Biblio

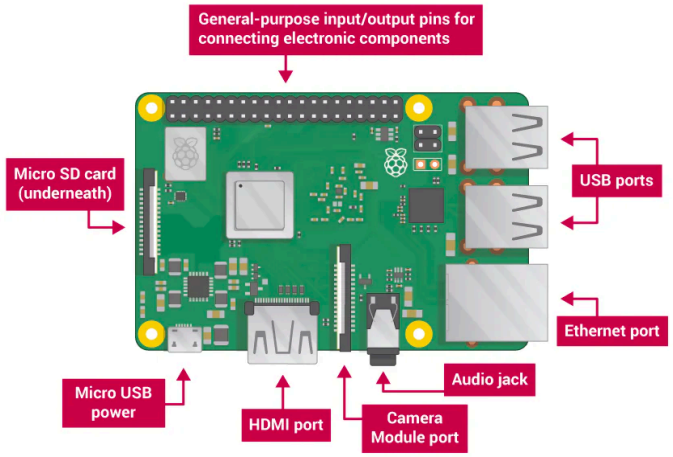
Petit résumé des sources utilisés lors de mes recherches ou, utile pour la suite,
Site de doc de BATMAN : https://www.open-mesh.org/projects/batman-adv/wiki/Wiki
Site de doc OLSRv2 : https://openwrt.org/docs/guide-user/start
Site de téléchargement img: https://www.raspberrypi.com/software/operating-systems/
Site de recherche le plus utilisé : https://chatgpt.com/
Site de recherche le plus pertinent : https://claude.ai

VII- Annexe

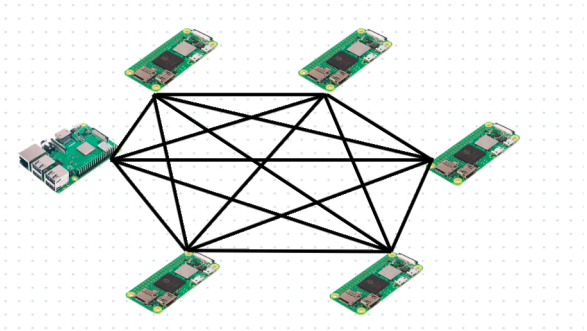
Annexe 1: Raspberry Pi Zero 2W



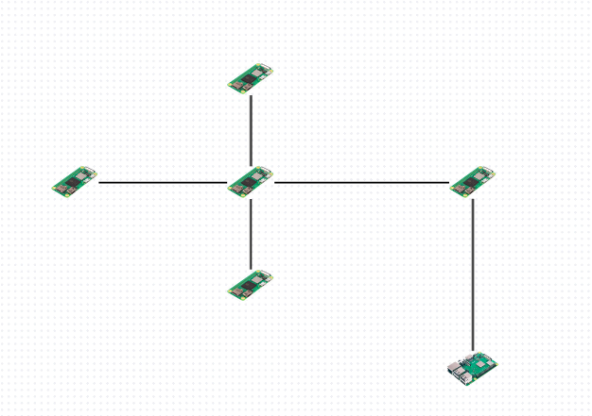
Annexe 2: Raspberry Pi 3



Annexe 3: Topologie Bureau



Annexe 4: Topologie sous-sol



Annexe 5: Topologie testé BATMAN-adv