Sommaire

Ce document exprime toutes les erreurs auxquelles j’ai pu faire face lors de cette expérimentation.

Sommaire :

Trixie

Pourquoi ne pas utiliser Trixie !

La licence Trixie et la dernière mise à jour de Debian et cela mène forcément plein de problèmes. Cette licence ne fonctionne pas avec les boot option proposé par l’Imager Raspberry suite à cela, il n’y as pas eu d’étude plus approfondie .

Ubuntu

Pourquoi ne pas utiliser Ubuntu !

L’outils Ubuntu comme l'outil Trixie ne collecte pas les informations de boot correctement et ducoup ne ce lance pas correctement. J’ai donc trouvé des solutions pour pallier certains problèmes comme le boot. Car sans information de boot il est impossible de se connecter à la machine. Afin de pallier ce problème, il suffit de se rendre dans les fichiers de boot et de changer dans le /etc/shadow la ligne de l'utilisateur ubuntu et de mettre cette clef exactement.
ubuntu:yj9T$ChHK5rnPqupCbB4LSb8gf.$sH30N1AQ34dkM7k.ss6wiafwWYVAQnw0geNM3ZwPJq0:20472:0:99999:7:::
Le tout en une seule ligne, pour changer le mot de passe en ‘**password**’.
Après exploration, le ssh ne marchait pas il fallait faire plein de modifications et d’activation désactivation de processus.

Bookworm

Pourquoi utiliser Bookworm !!!

Bookworm boot au démarrage avec les informations du Imager, il se connecte donc automatiquement au réseau, et le fichier de configuration à était travaillé pour fonctionner seulement sur cette licence pour le moment. Le choix s'est porté pour la licence Raspberry PI OS (LEGACY, 32-BIT) Lite. Cette OS est utilisée pour mettre en place la topologie sur toutes les puces sauf sur la Raspberry pi 3 car elle va devoir afficher le flux vidéo. Afin de mettre en place cela on vas utiliser Raspberry PI OS (LEGACY, 32-BIT) afin d’avoir une image graphique.

Fichier.sh

Comment faire marché convenablement le fichier maj.sh et add.sh

Pour faire fonctionner le maj.sh il faut simplement mettre la date du jour après le flash de la puce sur votre fichier, puis le lancer avec internet pour se connecter. Il faut aussi absolument avoir déposer côte à côte dans le répertoire /home les fichier 95proxy et le dossier OONF.

oldmaj.sh

Mise en marche oldmaj.sh

Projet abandonné suite au problème d’exécution du script. Le script réussissait à se lancer jusqu’au bout mais peu importe les modifications que je pouvais effectuer, impossible de créer le réseau ad hoc et de se connecter.

logmaj.sh

Mise en marche logmaj.sh

Projet d’amélioration mise en place afin de logger les informations qui sont exécutés durant le script, suite à ce code j’ai créé d’autre version qui sont des déclinaisons plus vulgaires afin de continuellement améliorer ce fichier.

Alpha.sh

Mise en marche Alpha.sh

Projet d’amélioration d’avant logmaj.sh,

Beta.sh

Mise en marche beta.sh

Projet d’amélioration concrète du script, mise en place des log et mise à jour des étapes d’exécution, développement de solution concrète au retour en fonctionnement, remise en fonctionnement du système, création du réseaux adhoc, mise en up de l’interface wlan0 systématique et même après plusieur redémarrage, Problème restant, OLSRv2 ne démarre pas ce qui empêche les puces de contacter les autres lors de leur création, et ne permet pas de scanner le réseau efficacement. Les puces ne prennent évidemment pas les bonnes adresses, elle s’équipe d'adresse en 169.254.x.x .

Charlie.sh

Mise en marche de Charlie.sh

Projet de mise en marche de OLSRv2 et de mise en place d'une solution afin de remettre en place les adresses demandées dans le script.

Afin de résoudre les problèmes liés au address, mise en place d’un système de log bien plus strict. Et, lors des étapes 8 et 9 le service ne redémarre pas à redémarrer ce qui inclut que l'adresse n’est pas ajouté au redémarrage, c’est le sujet de la recherche futur

Dans le moment où j'en suis, le code n'installe pas les librairi OLSRv2 dans les bon répertoire ce qui empêche le service de fonctionner correctement.

J’avais donc cette erreur quand je consultai le journalctl -u olsrv2 -b

raspberrypi systemd[1]: WARN: OLSRv2 lancé sans IP 10.0.0x raspberrypi systemd[1]: olsrv2.service: Failed with result 'exit-code'. raspberrypi systemd[1]: Failed to start olsrv2.service - OLSRv2 Routing Daemon (OONF)
raspberrypi systemd[1]: Stopped olsrv2.service - OLSRv2 Routing Daemon (OONF).
raspberrypi systemd[1]: Starting olsrv2.service - OLSRv2 Routing Daemon (OONF)...
raspberrypi olsrv2_dynamic[7482]: Unknown parameter: 'f' (102) raspberrypi systemd[1]: olsrv2.service: Main process exited, code=exited, status=1/FAILURE

Afin de pallier cette erreur j’ai donc changé le répertoire d’installation afin de voir ce que cela donnerait. Cela à donc donné cela, une nouvelle erreur.

Les lignes d'erreur que l'on retrouve quand je fais un journalctl -u olsrv2 -b sont,
bash[2965]: WARN: OLSRv2 lancé sans IP 10.0.0.x
systemd[1]: olsrv2.service: Failed with result 'exit-code'.
systemd[1]: Failed to start olsrv2.service - OLSRv2 Routing Daemon (OONF).
systemd[1]: olsrv2.service: Scheduled restart job, restart counter is at 1.
systemd[1]: Stopped olsrv2.service - OLSRv2 Routing Daemon (OONF).
(_dynamic)[3066]: olsrv2.service: Failed to locate executable /usr/local/sbin/olsrd2_dynamic: No such file or directory
(_dynamic)[3066]: olsrv2.service: Failed at step EXEC spawning /usr/local/sbin/olsrd2_dynamic: No such file or directory
systemd[1]: Starting olsrv2.service - OLSRv2 Routing Daemon (OONF) ...
systemd[1]: olsrv2.service: Main process exited, code=exited, status=203/EXEC

Pour changer cette erreur en solution, j'ai modifié mon fichier car lors de mon make install j'avais un problème make install

```
echo "/usr/lib/oonf" > /etc/ld.so.conf.d/oonf.oonf
ldconfig

echo "Vérifiaction"
ls -l /usr/sbin/olsrd2_dynamic || step_fail
ldd /usr/sbin/olsrd2_dynamic | grep not && step_fail
```

(mon fichier)

```
make install || step_fail

# Déclarer les libs OONF au linker
echo "/usr/lib/oonf" > /etc/ld.so.conf.d/oonf.conf
ldconfig

echo "Vérification binaire OLSRv2"

ls -l /usr/sbin/olsrd2_dynamic || step_fail

# Vérification différée : warning seulement
if ldd /usr/sbin/olsrd2_dynamic | grep -q "not found"; then
  log "Libs non résolues AVANT reboot (normal à ce stade)"
else
  log "Toutes les libs sont correctement résolues"
fi
```

C'est la partie 5, lors de la compilation de OLSR j'ai complètement modifier mes fichier et cela m'as redonnée mon erreur unknown 'f'
J'ai donc retiré ma ligne
-f /etc/olsrd2/olsrd2.conf -d 1
et je me retrouve avec un olsrv2 qui fonctionne !!!!

Pour voir les routes disponible une fois que au moins deux puces sont allumer,et qu'elles se sont déjà ping il faut faire : *ip neigh show dev wlan0*

Pour bien interpréter cette commande, il faut déjà connaître les résultats possibles.
10.0.0.1 lladdr ipv6 REACHABLE

l’ip, c’est classique, tout comme pour l’ipv6, cependant, il est cool de savoir à quoi correspond le REACHABLE et tous les autres état possible, afin de diagnostiquer l’état des relations réseaux. J’ai laissé l’explication chatgpt, c’est pour ca les smileys et les texte en gras/couleur, il as bien expliqué selon moi.

🟢 REACHABLE

- Le voisin a **répondu récemment**
- Communication OK
- C’est l’état idéal en mesh IBSS

🟡 STALE

- L’entrée est connue, mais **pas vérifiée récemment**
- Elle sera testée au prochain envoi de paquet

🔵 DELAY

- Le noyau attend avant de relancer une requête ARP

🟠 PROBE

- Le noyau **envoie activement des ARP** pour vérifier

🔴 FAILED

- Aucune réponse ARP
- Le voisin est considéré **injoignable**

⚪ **INCOMPLETE**

- Requête ARP envoyée
- Pas encore de réponse

Delta.sh

Mise en marche Delta.sh

L’objectif de Delta.sh et de faire marché l’addressage classique afin que l’on puisse commencer à utiliser les services olsrv2. Pour cela, j’essaye de désactiver le service kernel qui force l’attribution automatique d’une ip à une interface afin de permettre à l’interface Wlan0 d’obtenir une adresse en 10.0.0.x .
Pour cela, j’ai amélioré le fichier Chalie.sh en ajoutant une étape, et quelques commandes, le problème actuel et que je me retrouve avec les deux adresse sur mon interface.
Pour exclure l'attribution automatique d'adresse, j’ai d'abord exclu Avahi mais je n’ai cela n’as pas suffi, ce n’était pas lui le problème principal.
J’ai donc ensuite essayé de désactivé IPv4LL.
Cela n’as pas suffi, cependant, vu que l’interface réussissait à prendre les deux adresse, j’ai abandonnée et est rajoutée une étape afin de mettre en place un script simple qui envoyé des ping en broadcast dès le redémarrage. Cela à bien marché alors j’ai décidé de partir sur une nouveau document pour sauver le dernier.

Echo.sh

Mise en marche Echo.sh

Mise à jour de Delta.sh, ajout d'un script visant à lire dynamiquement les requêtes ICMP (ping) afin de tester la configuration réseaux et commencé à évaluer les distances maximum entre les noeux ainsi que de tracer les routes de Olsr.
Pour afficher ces logs il faut faire la commande :
journalctl -k -f

Tableau request journalctl

Pour aller piochée dans le code qui s'occupe de filtrer les requests
sudo mesh-ping-live

Suite au mise en place de ce script, il as était possible de lire en continue les ping qui on était transmis sur le réseaux via un affichage en IN —> OUT

Foxtrot.sh

Mise en marche Foxtrot.sh

Mise en place d’une serveur qui contient la date et l’heure afin d’unifier toute la flotte de raspberry. Ensuite mise en place de log dans un fichier.json afin de collecter toute les informations et mise en place d’un script qui s’exécute après réception d’un message du nœud maître.

Pour commencer, mise en place d'une boucle dans un nouvelle étape qui passe la puce 10.0.0.1 uniquement comme détentrice du service de date et de d’heure centralisé. Dans la même étape, mise en place d’une boucle qui dès lors le nœud comprenant le serveur de date, envoie d’un message les noeux à proximité prenne les même information que le serveur. Mise en place du serveur `chrony.`

Suite à cela mise en place de la journalisation dans le .json mais un problème est survenu sur la route, lorsque le nœud maître cherche à récupérer toutes les informations des autres nœuds.
Quand le programme se lancé, il proposé un coup de partir sur le réseau 10.0.0.x et une autre fois il cherché des liaisons sur le réseau 169.254.x.x .
J’ai donc connecté mes deux neurones et me suis rappelé de comment faire pour créer un service exécutable au démarrage, j’ai donc mise en place qui supprimer toute les ip de wlan0 15 secondes après le redémarrage. Enfin, on attend 10 secondes puis on applique l'adresse contenue dans $BOARD.

Gamma.sh

Mise en marche Gamma.sh

Mise en place de mise à jour sur la manière de logger suite à quel erreur rencontrée dans l'ancien code. Modification d'une ligne visant à forcer le log .json à utilisé l'adresse en 10.0.0.1 .

Utilisation de la commande
sudo bash -x /usr/local/bin/mesh-icmp-json.sh
Pour lancer le script de journalisation et de ping toute les secondes,
cat /var/log/mesh-icmp.json | jq .
Ensuite, la commande ci-dessus pour consulter le résultat.
Les commandes indiquées ci-dessus sont présentées pour mettre en marche le service en local.

Pour lancer la séquence sur tous les nœuds via un nœud maître il faut faire *sudo mesh-run-icmp*