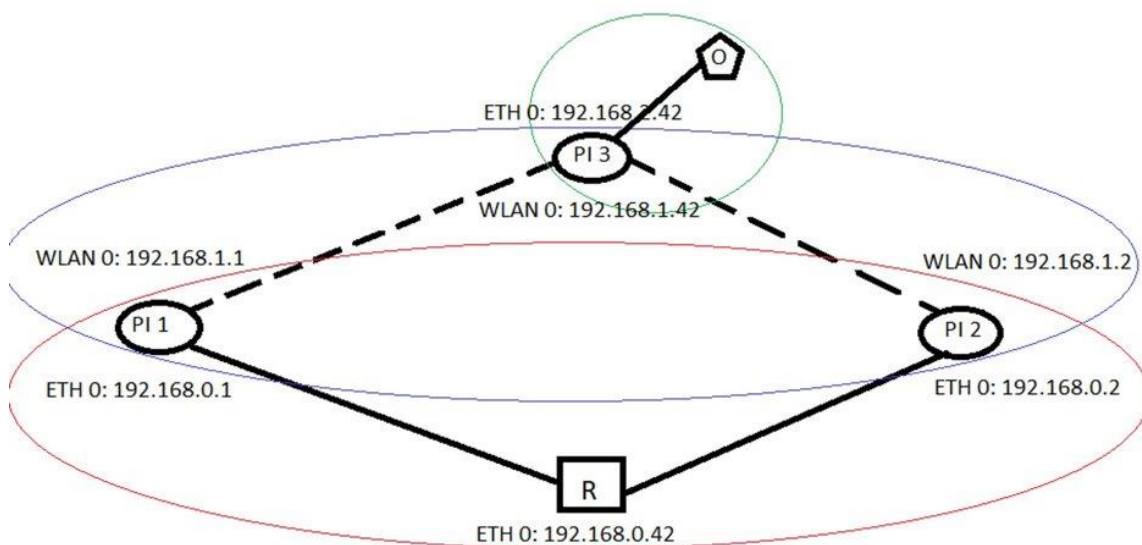


Projet redondance dans les réseaux IP

Objectif du projet :

Réaliser une architecture réseau IP permettant d'illustrer le principe et l'intérêt de la redondance dans le cadre d'un échange de données.

Description :



Matériel : 3x raspberry pi 3b+ sous Raspbian

1x routeur ethernet 8 ports MOXA EDS-611

3x câble ethernet

1x PC avec port ethernet et client SSH installé

3 réseaux séparés :

- réseau câblé ethernet avec routeur
- réseau HOSTAPD avec point d'accès WIFI
- réseau câblé entre PI3 et PC (pour accès SSH)

Toutes les machines sont configurées avec une adresse IP statique prédéfinie.

Il est nécessaire d'utiliser HOSTAPD pour configurer un point d'accès WIFI sur les pi 3b+. Cela facilite aussi l'attribution des IP statiques.

Principe de fonctionnement :

Ce projet prend la forme d'un système de messagerie instantanée envoyant des données de type string entre clients (PI 1 et PI 2). Le message est envoyé simultanément sur les réseaux ethernet et wifi.

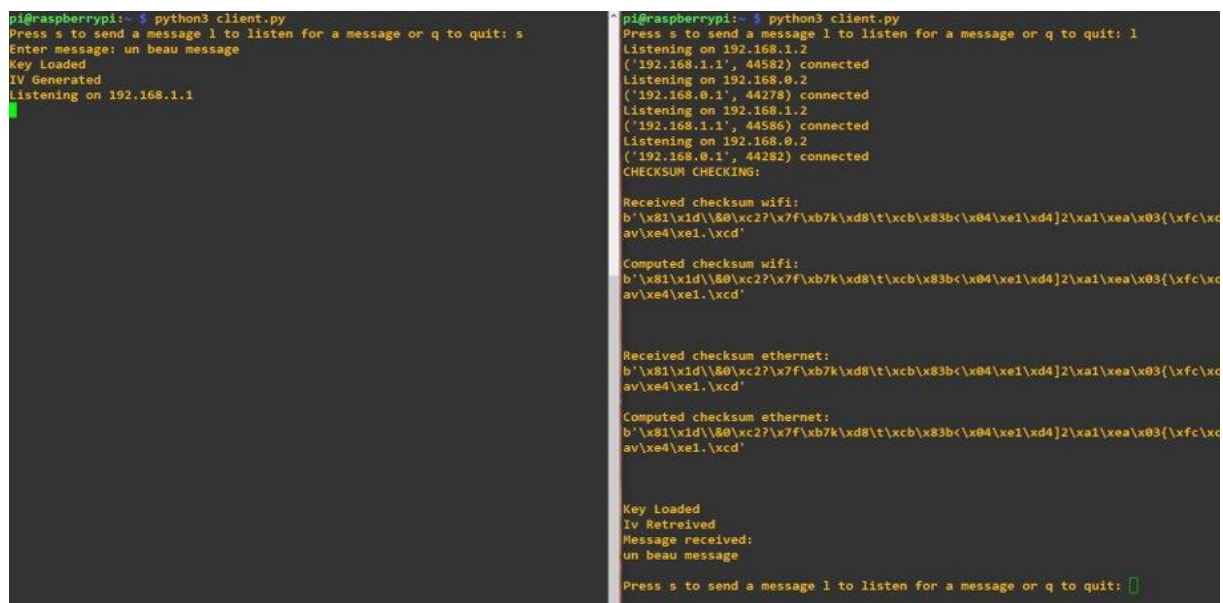
Le message est chiffré en AES (CFB) et on lui applique un checksum en SHA256. Chaque message de l'utilisateur fait l'objet de deux envois sur chaque réseau. Le premier paquet contenant le checksum calculé par la machine d'envoi et le vecteur d'initialisation aléatoire nécessaire au chiffrement et déchiffrement du message. Le second paquet contient le message chiffré.

Chaque machine client est dotée d'un script en python « client.py » permettant la gestion de toutes les méthodes décrites précédemment. La machine PI 3 est dotée d'un script en bash permettant le verrouillage du système en terminant les processus d'envoi sur les machines clients. Le script est portable et OS indépendant.

Les échanges IP sont gérés par la librairie Socket. Chaque socket est paramétré avec un délai de timeout ainsi que la possibilité de réutiliser les mêmes adresses plusieurs fois (indispensable pour garder le même canal de communication ouvert !)

Chaque machine client conserve une trace des messages envoyés et des cas d'erreur dans un fichier de log.

Exemple d'utilisation client :



```
pi@raspberrypi:~$ python3 client.py
Press s to send a message l to listen for a message or q to quit: s
Enter message: un beau message
Key Loaded
IV Generated
Listening on 192.168.1.1

pi@raspberrypi:~$ python3 client.py
Press s to send a message l to listen for a message or q to quit: l
Listening on 192.168.1.2
('192.168.1.1', 44582) connected
Listening on 192.168.0.2
('192.168.0.1', 44278) connected
Listening on 192.168.1.2
('192.168.1.1', 44586) connected
Listening on 192.168.0.2
('192.168.0.1', 44282) connected
CHECKSUM CHECKING:
Received checksum wifi:
b'\x81\xd1\x80\xc2?\x7f\xb7k\xd8\t\xcb\x83b\x04\xe1\xd4]2\xa1\xea\x03{\xfc\xca\x04\xe1.\xcd'
Computed checksum wifi:
b'\x81\xd1\x80\xc2?\x7f\xb7k\xd8\t\xcb\x83b\x04\xe1\xd4]2\xa1\xea\x03{\xfc\xca\x04\xe1.\xcd'
Received checksum ethernet:
b'\x81\xd1\x80\xc2?\x7f\xb7k\xd8\t\xcb\x83b\x04\xe1\xd4]2\xa1\xea\x03{\xfc\xca\x04\xe1.\xcd'
Computed checksum ethernet:
b'\x81\xd1\x80\xc2?\x7f\xb7k\xd8\t\xcb\x83b\x04\xe1\xd4]2\xa1\xea\x03{\xfc\xca\x04\xe1.\xcd'
Key Loaded
IV Retrieved
Message received:
un beau message
Press s to send a message l to listen for a message or q to quit: [
```

L'utilisateur dialogue par un terminal. Il choisit un mode (envoi ou écoute).

Lors de l'envoi, on récupère le message, on le chiffre et on calcule le checksum associé. On envoi les paquets sur les deux réseaux, on les récupère puis on vérifie les checksum reçus avec ceux calculés par la machine de réception. Enfin, on déchiffre le message, on l'affiche et on inscrit une trace dans le fichier log. Lorsqu'une machine a fini un envoi, elle passe automatiquement en mode écoute afin de faciliter un échange de messages régulier entre les deux clients.