

MASTER INFORMATIQUE RÉSEAUX INFORMATIQUES ET SYSTÈMES  
EMBARQUÉS (RISE)  
INGÉNIEUR RÉSEAUX, TÉLÉCOMS - INFRASTRUCTURES NUMÉRIQUES  
ET OBJETS COMMUNICANTS (RT-INOC)

# Mobilité et réseaux sans fil

Groupe :  
Stina GERARD

Le 11 Janvier 2018

# 1 Partie 1 : Projet Réseau Sans Fil Sécurisé et Monitoré

L'objectif de ce projet est de mettre en place un système appelé "Réseaux Sans Fil Sécurisé et Monitoré". Ce système consiste à fournir une configuration IPv4 et IPv6 aux clients WIFI mais à bloquer tout le trafic IP émis tant que celui-ci n'est pas passé par une authentification réalisée à l'aide d'un portail Web Captif. Le but de ce projet est de mettre en place une telle solution et de permettre à un client WIFI de disposer d'une connectivité IPv4 et IPv6 une fois que l'authentification est réalisée.

## 1.1 Mise en place du hotspot

### 1.1.1 Sur le PC

Les interfaces seront utilisées comme ceci : internet sur eth1, RaspberryPi sur eth0. Le PC doit disposer d'une adresse de sous-réseau sur l'interface qui le relie au Pi (dans tout ce projet le sous-réseau choisi est 192.168.2.0/24). Pour se faire nous allons procéder comme suit :

Désactiver le service network-manager et entrer la commande `ifconfig eth0 192.168.2.1/24`

Ensuite, afin notamment faciliter les installations d'outils nécessaires sur le Raspberry, il est conseillé de créer un nat sur le PC pour permettre au Raspberry de se connecter à l'internet :

```
1 echo 1 > /proc/sys/net/ipv4/ip_forward
2 iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
3 iptables -A FORWARD -i eth1 -j ACCEPT
```

### 1.1.2 Sur le RaspberryPi (en root)

Après la première connexion ssh il est nécessaire de configurer l'interface Ethernet en statique, en éditant le fichier `/etc/network/interfaces` comme ceci :

```
1 auto eth0
2 iface eth0 inet static
3     address 192.168.2.3
4     netmask 255.255.255.0
5     gateway 192.168.2.254
```

On en profite alors pour configurer l'interface Wi-Fi que l'on va mettre en place sur le RaspberryPi. Il faut définir un sous-réseau :

```
1 auto wlan0
2 iface wlan0 inet static
3     hostapd /etc/hostapd/hostapd.conf
4     address 192.168.5.1
5     netmask 255.255.255.0
```

Pour créer le réseau Wi-Fi il faut configurer et utiliser hostapd.

Il faut ensuite configurer hostapd en éditant le fichier `/etc/hostapd/hostapd.conf` comme ceci :

```
1 \# Basic configuration
2 interface=wlan0
3 ssid=Stina-AP
4 channel=1
5 \# WPA and WPA2 configuration
6 macaddr_acl=0
7 auth_algs=1
8 wmm_enabled=0
9 max_num_sta=10
10 wpa=1
11 wpa_passphrase=raspberry
12 wpa_key_mgmt=WPA-PSK
13 wpa_pairwise=TKIP
14 rsn_pairwise=CCMP
15 \# Hardware configuration
16 driver=rtl871xdrv
```

Et ajouter la ligne suivante dans `/etc/default/hostapd` pour que hostapd se lance automatiquement au démarrage du Raspberry :

```
1 DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Par la suite, on passe à la configuration de DNSMASQ : Pour cela, on modifie le fichier : `/etc/dnsmasq.conf` On remarque aussi, que l'on a les lignes à ajouter relatives à IPV6 et qui correspondent aux données fournies par netassist.

```
1 interface=wlan0 # Use interface wlan0
2 listen-address=192.168.5.1 # Explicitly specify the address to listen on
3 bind-interfaces # Bind to the interface to make sure we aren't sending
   ↪ things elsewhere
4 server=8.8.8.8 # Forward DNS requests to Google DNS
5 domain-needed # Don't forward short names
6 bogus-priv # Never forward addresses in the non-routed address spaces.
7 dhcp-range=192.168.5.50,192.168.5.150,12h # Assign IP addresses
8
9 dhcp-range=2a01:d0:e2e9::, ra-stateless
10 enable-ra
11 dhcp-option=option6:dns-server,[2a01:d0::1],[2001:4860:4860::8888]
```

Nous devons aussi activer le forwarding IPV4. Pour cela, nous décommentons la ligne `net.ipv4.ip_forward=1` dans le fichier `/etc/sysctl.conf`. Nous devons également partager la connexion Internet de notre Pi à nos appareils connectés via le WiFi en configurant un NAT entre notre interface `wlan0` et notre interface `eth0`. Nous pouvons le faire en utilisant les commandes suivantes :

```

1 sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
2 sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,
  ↳ ESTABLISHED -j ACCEPT
3 sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT

```

Il n'y a plus qu'à démarrer les services :

```

1 sudo service hostapd start
2 sudo service dnsmasq start

```

Il ne reste plus qu'à redémarrer le RaspberryPi (reboot) et la configuration est terminée.

## 1.2 Tunnel Broker IPV6

Avant la création du Tunnel, il faut activer le support IPV6 sur la Raspberry Pi et modifier le fichier `/etc/network/interfaces` en ajoutant une adresse IPv6 statique faisant partie de la plage routable. Afin de créer un Tunnel Broker IPV6, j'ai utilisé Netassist et obtenu les adresses suivantes :

Your NetAssist IPv6 Tunnel Broker details:

Server IPv4 address	62.205.132.12
Client IPv4 address	130.79.92.11
Server IPv6 address	2a01:d0:ffff:62e9::1/64
Client IPv6 address	2a01:d0:ffff:62e9::2/64
Routed /48 IPv6 network	2a01:d0:e2e9::/48
IPv6 DNS server	2a01:d0::1
Your e-mail (login)	stina-gerard@hotmail.fr
Change	

FIGURE 1 – Configuration Netassist

Une fois le tunnel Broker mis en place, on ajoute au fichier `/etc/network/interfaces`, une nouvelle interface "netassist" avec les adresses fournies dans le fichier ci-dessus et on vérifie qu'on a bien des adresses IPv6 pour eth0 et wlan0.

```

auto eth0
iface eth0 inet static
    address 192.168.2.3
    netmask 255.255.255.0
    network 192.168.2.0
    gateway 192.168.2.1

iface eth0 inet6 static
    address 2a01:d0:e2e9::1
    netmask 48

auto netassist
iface netassist inet6 v4tunnel
    address 2a01:d0:ffff:62e9::2
    netmask 64
    endpoint 62.205.132.12
    local 192.168.2.3
    gateway 2a01:d0:ffff:62e9::1

ttl 255
mtu 1472

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.5.1
    netmask 255.255.255.0
    network 192.168.5.0
    broadcast 192.168.5.255
iface wlan0 inet6 static
    address 2a01:d0:e2e9::3
    netmask 48

```

FIGURE 2 – Fichier "interfaces"

Ensuite, on ajoute au fichier `/etc/resolv.conf`, les lignes suivantes :

```

1 nameserver 8.8.8.8
2 nameserver 8.8.4.4

```

Finalement, on doit activer le forwarding IPv6 en éditant le fichier `/etc/sysctl.conf` et en décommentant la ligne :

```

1 net.ipv6.conf.all.forwarding=1

```

Après cette étape, il était possible de vérifier que nous avions bien la connectivité en IPv6 en lançant un ping6 vers `ipv6.google.com` par exemple.

## 1.3 Authentification et portail Web Captif

### 1.3.1 Freeradius

Pour cette partie du projet, j'ai utilisé FreeRadius (version OpenSource de RADIUS) qui permet d'avoir sur son serveur/machine un protocole réseau qui est utilisé pour gérer les authentifications et les comptes d'utilisateurs. Il permet de contrôler les accès (authentification) mais aussi d'en surveiller les usages et d'y appliquer des règles d'autorisation ou de rejet sur base d'attributs comme par exemple l'heure, la durée, le volume de données, etc...

On commence par créer la base de données dans MySQL dédiée à FreeRadius et installer le schéma de base radius que nous venons de créer. On installe ensuite la partie d'administration afin d'avoir un utilisateur d'administration et lui donner tous les droits nécessaires et on installe les tables supplémentaires pour le NAS.

On édite pour cela, le fichier `/etc/freeradius/3.0/mods-available/sql`, notamment les lignes suivantes :

```

1 server = "localhost"
2 port = 3306
3 login = "radius"
4 password = "raspberry"
5 radius_db= "radius"

```

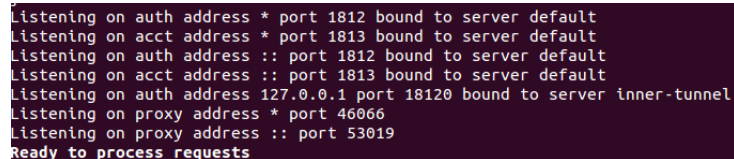
De plus, il faut éditer le fichier `/etc/freeradius/radiusd.conf` pour charger le module SQL et activer l'authentification par la base MySQL en éditant le fichier `/etc/freeradius/sites-enabled` en décommentant toutes les lignes où "sql" apparaît.

Finalement on peut tester la configuration en arrêtant FreeRadius et en le relançant en mode **debug**.

```

1 service freeradius stop
2 freeradius -X

```



```

Listening on auth address * port 1812 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on auth address :: port 1812 bound to server default
Listening on acct address :: port 1813 bound to server default
Listening on auth address 127.0.0.1 port 18120 bound to server inner-tunnel
Listening on proxy address * port 46066
Listening on proxy address :: port 53019
Ready to process requests

```

FIGURE 3 – Serveur Freeradius

### 1.3.2 Coovachilli

CoovaChilli est la version OpenSource du projet ChilliSpot. Il propose une interface utilisateur pour authentifier les utilisateurs qui se connectent à un hotspot (pas forcément Wifi). C'est grâce à ce projet que l'on pourra gérer le hotspot et notamment authentification. Dans le fichier `/etc/chilli/config` se trouve la configuration principale de Chilli. Là où on a pu définir quelles interfaces sont utilisées, quel réseau, etc .. J'ai notamment modifié les lignes suivantes :

```

1 HS_WANIF=eth0
2 HS_LANIF=wlan0
3 HS_NETWORK=192.168.5.0
4 HS_UAMLISTEN=192.168.5.1
5 HS_UAMALLOW=192.168.5.0/24
6 HS_SSID=Stina-AP

```

Avec : HS\_WANIF est l'interface reliée à Internet HS\_LANIF est l'interface du Wifi/-Hotspot HS\_NETWORK le réseau du hotspot HS\_UAMLISTEN la gateway du réseau du hotspot HS\_UAMALLOW les IP du réseau du hotspot autorisées à se connecter

Finalement, on peut démarrer le service avec la commande :

```

1 service chilli start

```

Avec la commande `ifconfig` on voit apparaître une interface `tun0` qui confirme que Coova-Chilli est bien exécuté.

```
tun0: flags=81<UP,POINTOPOINT,RUNNING> mtu 1500
inet 192.168.5.1 netmask 255.255.255.0 destination 192.168.5.1
inet6 fe80::845e:98aa:10cf:861a prefixlen 64 scopeid 0x20<link>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 100
```

FIGURE 4 – Interface tun0

### 1.3.3 Daloradius

De plus, j'ai installé Daloradius afin de gérer les utilisateurs, gérer leurs droits, heures de connexion, débits autorisés.

Users Listing +

SELECT: **ALL** NONE

Delete Disable Enable CSV Export

1

ID	Name	Username	Password
<input type="checkbox"/> 7		test	123
<input type="checkbox"/> 8		user	azerty

FIGURE 5 – Daloradius - Liste des utilisateurs

### 1.3.4 NTOP-NG

Pour finir, il nous était demandé d'installer un outil de monitoring. J'ai choisi NTOP-NG. Grâce à cet outil, il nous est possible de consulter des informations concernant le trafic sur notre hotspot. On peut par exemple avoir accès au contenu consulté par tel ou tel utilisateur. Ci-dessous, on retrouve les informations relatif à un utilisateur précis que l'on repère à son adresse MAC et son adresse IP.

ntop

Flows Hosts Interfaces Settings Users Alerts Search Host

Host: 192.168.5.125 Traffic Packets Ports Peers Protocols DNS HTTP Flows Talkers Similarity Alerts Settings Traffic

(Router) MAC Address	98:01:A7:B3:76:6D	<input type="checkbox"/> Dump Traffic
IP Address	192.168.5.125 [ 192.168.5.0/24 ]	
OS	Apple Intel Mac OS X	
Name	MBP-de-GERARD-2 Local Private IP	192.168.5.125 Save Custom Name
Alerts	57	
First / Last Seen	21/12/2017 09:36:18 [1 hour, 19 min, 52 sec ago]	21/12/2017 10:55:54 [16 sec ago]
Sent vs Received Traffic Breakdown	Sent Rcvd	
Traffic Sent / Received	12,227 Pkts / 1.70 MB	12,514 Pkts / 12.03 MB
Active Flows / Total Active / Low Goodput	'As Client' 45 ↑ / 865 ↑ / 2	'As Server' 0 - / 17 - / 0
TCP Packets Sent Analysis	Retransmissions	475 Pkts
	Out of Order	26 Pkts
	Lost	90 Pkts

FIGURE 6 – NTOPNG pour cet utilisateur

## 1.4 Interface Web - Suivi utilisateurs

Une fois le portail Web captif fonctionnel, il nous était demandé de créer une page où seraient présentes plusieurs informations relatives aux utilisateurs. Ce qui est présent sur l'interface web codée en PHP et reliée à la base de données regroupant les utilisateurs. Les informations présentes sont les suivantes :

- 1 Nom de l'utilisateur
- 2 Adresses IPV4 et IPV6
- 3 Horaire et date de début de connexion
- 4 Lien vers la page NTOPNG (lien différent pour chaque user)

Afin d'obtenir les différentes informations, j'ai utilisé la base de données **radius** et la table **radacct**. Ci-dessous, un aperçu de la page Web pour un utilisateur connecté.

### Liste des utilisateurs

USER	MAC ADDRESS	IPV4 ADDRESS	IPV6 ADDRESS	LOGTIME	LOGTIME	NTOPNG LINK
user	98-01-A7-B3-76-6D	192.168.5.125	2a01:d0:e2e9:0:b5fa:52d3:8007:8dc0	2017-12-21 08:38:52	2017-12-21 08:38:52	<a href="#">Ntop link</a>

FIGURE 7 – Interface des utilisateurs



## 2 Partie 2 : Support de la mobilité IPv6

L'objectif de cette partie du projet est de mettre en œuvre le support de la mobilité IPv6. Il conviendra de supporter les fonctionnalités de "Mobile Node". La fonctionnalité de "Home Agent" n'est finalement pas à considérer, on nous donne par contre l'adresse IPv6 de ce dernier. Dans un scénario type, le "Mobile Node" devra se connecter sur le portail Web Captif d'une autre personne et y être considéré comme équipement mobile IPv6 venant de s'attacher à un réseau visité.

### 2.1 Partie : Recompile du kernel

Afin de pouvoir utiliser la mobilité IPV6, il faut recompiler le kernel de l'ordinateur car dans celui-ci la mobilité IPV6 est désactivée. Après avoir télécharger le code source, j'ai suivi les différentes étapes nécessaires afin de compiler un nouveau kernel :

```
1 apt-get source linux-image-$(uname -r)
```

Le fichier de configuration à modifier est présent via le chemin `/usr/src/linux-3.13.0/.config`  
Les modules à vérifier/ajouter sont les suivants :

```
1 CONFIG_IPV6_MIP6=y
2 CONFIG_XFRM_USER=y
3 CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION=y
4 CONFIG_IPV6_TUNNEL=y
5 CONFIG_INET6_ESP=y
6 CONFIG_NET_KEY=y
```

Puis j'ai suivi le guide présent sur le lien suivant :

<https://help.ubuntu.com/lts/installation-guide/i386/ch08s06.html>

Les différentes commandes utiles pour la recompilation du noyau sont :

```
1 sudo apt-get install kernel-package
2 fakeroot make-kpkg --initrd --revision=1.0.custom kernel_image. -j8
3 dpkg -i ../linux-image-4.4-subarchitecture_1.0.custom_i386.deb
4 make
5 make modules_install
6 make install
7 make headers_install
8 shutdown -r now
```

## 2.2 Partie : UMIP

Pour cette partie, la première étape est de télécharger UMIP sur le site officiel. Dans la racine du dossier de UMIP, j'ai suivi le guide install.KERNEL. Finalement j'ai fini l'installation avec le fichier INSTALL de UMIP.

J'ai installé les paquets nécessaires à la compilation puis ai procédé à la compilation et l'installation :

```
1 sudo apt-get install make build-essential indent linux-headers-amd64
   ↳ autoreconf bison flex
2 autoreconf -i
3 CPPFLAGS='-isystem /usr/include/' ./configure --enable-vt
4 make
5 sudo make install
```

Il a fallu aussi créer le fichier `mip6d.conf` ci-dessous et notamment préciser la Home Address et l'adresse du Home Agent ainsi qu'ajouter l'option "Tunnel enabled" :

```
1 # Sample UMIP configuration file for a MIPv6 Mobile Node
2 NodeConfig MN;
3
4 # Set DebugLevel to 0 if you do not want debug messages
5 DebugLevel 10;
6
7 # Enable the optimistic handovers
8 OptimisticHandoff enabled;
9
10 # Disable RO with other MNs (it is not compatible
11 # with IPsec Tunnel Payload)
12 DoRouteOptimizationMN disabled;
13
14 # The Binding Lifetime (in sec.)
15 MnMaxHaBindingLife 60;
16
17 # List here the interfaces that you will use
18 # on your mobile node. The available one with
19 # the smallest preference number will be used.
20 Interface "wlan0" {
21     MnIfPreference 2;
22     Tunnel enabled;
23 }
24 # Replace eth0 with one of your interface used on
25 # your mobile node
26 MnHomeLink "wlan0" {
27     HomeAgentAddress 2001:660:4701:f055:ffff::1;
28     HomeAddress 2001:660:4701:f055:ffff::1016/64;
29 }
30 # Enable IPsec static keying
31 UseMnHaIPsec disabled;
32 KeyMngMobCapability disabled;
```

## 2.3 Test de la mobilité IPv6

Afin de tester la mobilité IPv6, je me suis tout d'abord connecté avec le PC sur un premier HotSpot. Ensuite, j'ai lancé umip en tant que mobile node avec la commande suivante lancée en root :

```
1 sudo mip6d -c /usr/local/etc/mip6d.conf
```

J'ai d'abord lancé une capture Wireshark sur toutes les interfaces et j'ai lancé un ping6 sur `ipv6.google.com`. Ensuite je change de HotSpot et observe si le ping continue bien (après un petit laps de temps). Sur la capture Wireshark, j'ai constaté que l'initialisation s'effectue correctement : le mobile node s'enregistre auprès de son home agent (« Binding Update » puis « Binding Acknowledgement »).

282	16.607224000	2001:660:4701:f055:ffff::1016	2001:660:4701:f055:ffff::1	MIPv6	112 Binding Update
297	16.640119000	2001:660:4701:f055:ffff::1	2001:660:4701:f055:ffff::1016	MIPv6	96 Binding Acknowledgement
792	38.611844000	2001:660:4701:f055:ffff::1016	2001:660:4701:f055:ffff::1	MIPv6	112 Binding Update
804	38.643608000	2001:660:4701:f055:ffff::1	2001:660:4701:f055:ffff::1016	MIPv6	96 Binding Acknowledgement

FIGURE 8 – Capture Wireshark sur le protocole MIPv6

En conclusion : J'ai obtenu un portail Web captif opérationnel et qui supporte IPv4 et IPv6. A cela s'ajoute l'interface Web afin d'avoir des informations sur les utilisateurs connectés. Pour la partie mobilité IPv6, j'ai finalement pu supporter les fonctionnalités de "Mobile Node".

En définitive, je pense avoir passé 35 heures sur la partie 1 du projet (dont l'interface Web), 5h sur la partie 2 et 5h sur le rapport.

Sources :

```
1 https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with
  ↪ -hostapd/
2 http://www.dickson.me.uk/2013/03/15/setting-up-a-raspberry-pi-as-an-ipv6-
  ↪ gateway-using-hurricane-electric/
3 https://citricks.net/install-freeradius-server/
4 http://wiki.freeradius.org/guide/SQL-HOWTO-for-freeradius-3.x-on-Debian-
  ↪ Ubuntu#setting-up-the-radius-database
5 http://www.pihomeserver.fr/2015/08/05/raspberry-pi-coovachilli-et-
  ↪ freeradius-pour-un-hotspot-wifi-avec-portail-captif/
6 https://help.ubuntu.com/lts/installation-guide/i386/ch08s06.html
7 http://www.umip.org/docs/umip-install.html
```