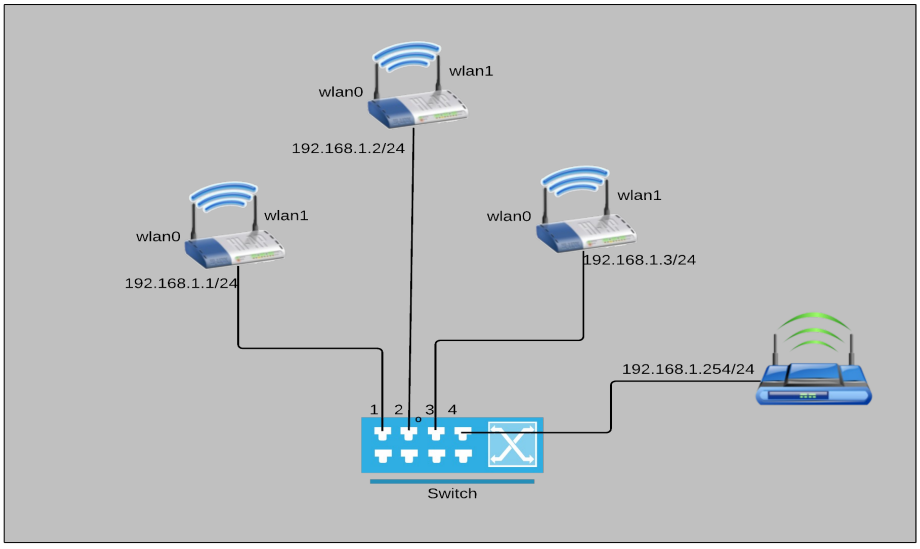
Memòria Final de TFM

Drizzle: consola de gestió de punts d'accés per a una xarxa mallada.



Alumne: Jordi Garcia Soler (jordialcoister@gmail.com)

Tutor UOC: Gregorio Robles Martínez (grobles@uoc.edu)

Tutor d'empresa: Víctor Oncins (victor@routek.net)

Índex de continguts

1 Introducció i motivació

Les xarxes obertes són un fenomen en expansió. Solen tenir una comunitat forta al darrere, incloent empreses que hi dediquen molts esforços.

Aquestes xarxes es regeixen per unes lleis similars a les del programari lliure, establertes a la llicència «Comuns de XOLN»[1]. Aquesta llicència garanteix el dret del usuaris d'aquestes xarxes a:

- Utilitzar-les per a qualsevol propòsit.
- Conèixer com estan construïdes i la seua estructura.
- Afegir-hi serveis i continguts amb qualsevol tipus de condicions.
- Incorporar-se a elles sota les tres condicions anteriors.

Un exemple molt conegut el tenim a casa nostra, el 2008 la Fundació privada per a la xarxa oberta, lliure i neutral guifi.net[2], ONG de cooperació al desenvolupament i entitat de voluntariat, es va inscriure com a Operador de Telecomunicacions al registre d'operadors amb la intenció de ser un instrument al servei de la xarxa de comuns sense alterar la seva forma original. I així nasqué la xarxa Lliure, Oberta i Neutral «guifi.net»[3].

Guifi.net és una xarxa oberta, mallada, desplegada majoritàriament a Catalunya i en menor mesura al País Valencià i a altres punts de l'estat.

Aquesta mena de xarxes depenen de la comunitat d'usuaris, pel que es construeixen amb pocs recursos i els serveis i els continguts depenen completament de la comunitat, això crea una sèrie de mancances per falta de recursos.

Routek, S.L. col·labora en un projecte anomenat qmp (quick mesh project) que persegueix la creació d'un sistema per al desplegament fàcil de xarxes mallades mitjançant la tecnologia Wi-Fi. Aquest «sistema» consisteix en un «firmware» per a sistemes embeguts basats en el sistema operatiu OpenWRT, proporcionant una manera senzilla de construir xarxes mallades, pel que és particularment útil en desplegaments ràpids de xarxes (per exemple en events curts o demostracions) i xarxes lliures comunitàries, com «guifi.net».

Aquí es on entra en escena el present TFM, consistent en el desenvolupament des de zero d'una consola de gestió de punts d'accés aprofitant les facilitats proporcionades per la distribució OpenWRT (com el sistema ubus) per a integrar-la al projecte qmp.

Si el resultat del TFM és satisfactori, se l'alliberarà amb llicència GPL sota l'auspici de Routek, S.L. al projecte qmp per a que la comunitat puga continuar amb el seu desenvolupament.

2 Objectius, metodologia i planificació:

L'objectiu d'aquest projecte és passar a formar part de manera temporal del planter de desenvolupadors de «Routek, S.L.» per a iniciar el desenvolupament des de zero d'un controlador de punts d'accés per a una xarxa mallada i alliberar-lo posteriorment amb llicència GPL.

La idea que es buscava era que fos bàsic, alhora que robust, i que servís de base per al desenvolupament, una vegada alliberat, d'una consola de gestió més completa que es pogués utilitzar en xarxes obertes reals, com ara «guifi.net».

Inicialment, es va proposar crear un programari seguint el paradigma agent/controlador, on els dispositius gestionats executessin un «daemon agent» encarregat de recopilar informació de configuració i aplicar els canvis requerits pel «daemon controlador» associat.

Aquest «daemon controlador» s'executaria en un encaminador i seria accedit via http per una interfície web servida des del mateix dispositiu.

Les sol·licituds de informació per part del «controlador» es farien via ubus sobre http mentres que les peticions de canvis de configuració es farien sobre SSH mitjançant autenticació de clau pública/privada.

Aquest model, que anomenarem ideal, implicava una complexitat alta, i hagués requerit moltes més hores de les que s'han dedicat ací per a assolir uns mínims requeriments, pel que es va optar per una solució més senzilla basada només en dos components: un «daemon» i una interfície web.

En aquest model, una mica més senzill, un encaminador executa el «daemon» i serveix la interfície web.

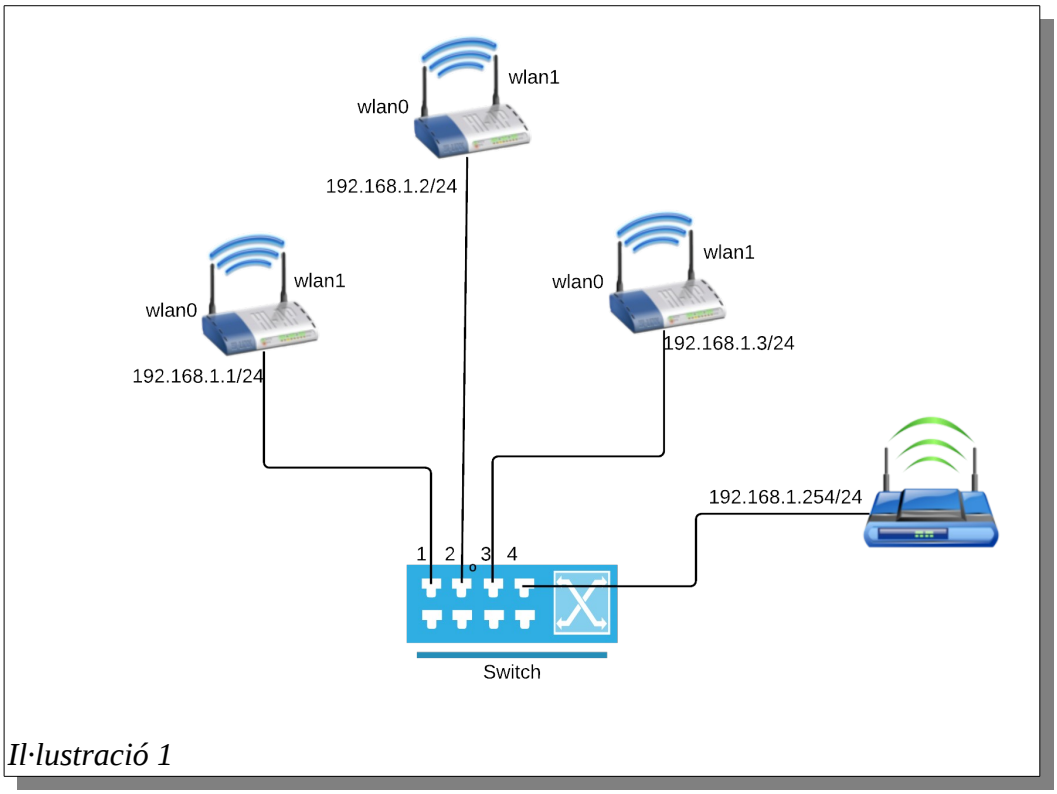
El «daemon» fa tota la feina fora d'escena, recupera informació de gestió dels dispositius gestionats si així se li demana des de la interfície web o aplica canvis en la configuració d'aquests que li arriben pel mateix canal, en aquest cas fent servir connexions SSH.

Routek, S.L. s'encarrega de proporcionar els elements de xarxa necessaris per a la construcció del laboratori sobre el que el programa ha de demostrar ser capaç de funcionar raonablement bé. Consisteix en:

Dispositiu	Descripció
Encaminador	Encaminador amb arquitectura Alix(x86)/OpenWRT
Switch	Switch D-Link Easy Smart Switch DGS-1100-08P amb capacitat PoE
AP	Kit de tres Access Point Unifi Pro amb OpenWRT preinstal·lat
Material addicional	Cables Ethernet i conversor USB-Flash per a recuperació de desastres a l'encaminador.

Taula 1: Elements de xarxa proporcionats per Routek, S.L.

I l'arquitectura de xarxa requerida es pot veure a la següent il·lustració:



Il·lustració 1

Els subobjectius generals inicialment establers consistien a

- Estudiar la distribució Linux *OpenWRT*, dissenyada específicament per a dispositius de xarxa (encaminadors, AP, tallafocs...) ja que tot i tractar-se d'una distribució de Linux, presenta una sèrie de peculiaritats que la diferencien de la resta, això inclou:
  - Estudiar les peculiaritats de configuració de la xarxa.
  - Estudiar la comanda *uci* per a la gestió de la configuració de la xarxa a *OpenWRT*.
  - Estudiar el funcionament del subsistema *ubus* d'interconnexió de comunicació entre *daemons* propi d'*OpenWRT* .
  - Estudiar de quina manera d'accedir a *ubus* via *http*.
- Desenvolupar un *daemon* capaç de recuperar periòdicament la adreça IP així com altres informacions estàtiques dels dispositius gestionats des d'un fitxer de configuració preestablert i paral·lelament enregistrar-se al sistema *ubus* per a poder rebre les peticions des de la interfície *web*.

- El *daemon*, una vegada enregistrat a *ubus*, hauria d'oferir uns procediments adequats als requeriments de la interfície web, sent capaç d'obrir connexions SSH paral·leles mitjançant fils per a recuperar informació de gestió dels dispositius o canviar configuracions massives de paràmetres dels dispositius de xarxa.
- Desenvolupar una interfície web capaç de recuperar la informació d'estat de tots els dispositius configurats al fitxer de configuració abans esmentat, així com permetre modificar un paràmetre de configuració del/s dispositiu/s dispositiu *wireless* dels mateixos. Finalment, hauria de ser capaç de modificar la configuració d'un d'aquests paràmetres a tots els dispositius.
- Crear un guió d'inici del sistema per a que el *daemon* es llanci a l'engegar l'encaminador.
- Crear els paquets d'instal·lació estàndard d'*OpenWRT* escaients.

A continuació els presente la planificació original, amb els objectius concrets:

Tasca		Durada
Desenvolupament d'un controlador bàsic		44d
	Crear una interfície web amb JavaScript capaç d'obtindre la informació de sistema del propi controlador.	7d
	Desenvolupar un *daemon* en C al controlador que recuperi les adreces IP estàtiques *hard-coded* dels 3 AP, desades a un fitxer	15d
	Fer que aquest agent obri una connexió SSH amb cadascun d'aquests AP.	7d
	Fer que el *daemon* consulti les dades del fitxer periòdicament.	4d
	Registrar el *daemon* a la interfície ubus del controlador.	2d
	Fer que el *daemon*, enviï la ordre "I'm alive" i recupere la informació de sistema de tots els AP, via *ubus*	7d
	Emmagatzemar la informació dels AP en una estructura de dades en memòria i mostrar-la en la web.	2d
Desenvolupament d'un controlador avançat		22d
	Fer que el *daemon* s'iniciï amb els scripts d'inici del sistema	1d
	Fer que el *daemon* capture els senyals dels sistema per a tancar-se elegantment, alliberant els recursos.	2d
	Crear un procediment ubus al daemon per a recuperar les dades de configuració *wireless* d'un dispositiu o tots.	3d
	Crear una interfície web que ens permeta recuperar el canal pel que transmet el primer dispositiu *wireless* d'un o de tots els dispositius.	3d
	Crear un procediment *ubus* al *daemon* per a alterar el canal de transmissió del primer dispositiu *wireless* d'un dispositiu o tots.	3d
	Crear una interfície web que ens permeta modificar el canal pel que transmet el primer dispositiu *wireless* d'un o de tots els dispositius.	3d
	Crear una interfície web que ens permeta fer modificacions a tots els AP (p. ex. Canviar els canals de totes les AP)	7d

Taula 2: Planificació inicial (I)

Tasca		Durada
Refinaments i tasques finals		2d
	Crear un paquet *OpenWRT* estàndard per al *daemon*.	1d
	Crear un paquet *Operant* estàndard per a la interfície web.	1d
Total		68d

Taula 3: Planificació inicial (II)

3 Estat de l'art

A continuació es pot veure un llistat de les tecnologies en que es recolza el treball:

- **Wireless**: El projecte tracta de la gestió de xarxes sense fils mallades, pel que tot el projecte gira al voltant d'aquesta tecnologia de xarxa.
- **C**: Per a la programació de l'agent, existien dues possibilitats, utilitzar el llenguatge **C** o el llenguatge «**lua**», finalment s'ha triat **C** pel fer que em sento còmode programant-hi.
- **HTML/css/Javascript**: Per al desenvolupament de la plana web, s'han utilitzat aquestes dues tecnologies, amb la intenció de descarregar al màxim al dispositiu controlador.
- **Ajax[4]**: Ajax són les sigles d'Asynchronous JavaScript XML, no és un nou llenguatge, si no una nova manera d'utilitzar els estàndards ja existents per a intercanviar dades amb un servidor i actualitzar parts d'una web sense haver de recarregar-la.
- **JSON[5]**: JSON (JavaScript Object Notation) és una sintaxi per a l'intercanvi de dades alternatiu a XML i més fàcil d'utilitzar.
- **Git**: Com a dipòsit de programari, control de versions, etc... s'ha utilitzat la tecnologia git.
- **OpenWRT[6]**: **OpenWRT** és una distribució de Linux optimitzada per ser encastada en Punts d'Accés i Encaminadors sense fils. En el cas que ens ocupa, utilitzarem un encaminador com a controlador i tres Punts d'Accés com a dispositius gestionables (dels comunament utilitzats a la xarxa guifi.net), als quals se'ls instal·la aquest «*firmware*». Per aquesta raó tot el desenvolupament s'ha realitzat per a aquesta plataforma. De la mateixa manera, tant els guions d'inici com l'empaquetat del programari s'ha adaptat a les seves peculiaritats.
- **Ubus[7]**: **Ubus** és una interfície de comunicació entre varis «*daemons*» i aplicacions presents al sistema **OpenWRT** i que utilitzar **JSON** per a passar missatges entre aquests. Drizzle el que fa és enregistrar el daemon a ubus, per a poder-nos comunicar amb ell mitjançant aquesta interfície homogènia. **Ubus** és accessible a la línia d'ordres i via SSH o HTTP, pel que resulta idoni per a la intefície web.
- **UCI[8]**: El sistema **UCI** (*Universal Configuration Interface*) és la principal interfície de configuració d'usuari per als paràmetres principals del sistema.

4 Desenvolupament i implementació

4.1 L'arquitectura ubus

Ubus és una arquitectura de micro bus pròpia del sistema OpenWRT que comunica diversos daemons i aplicacions al sistema. Aquesta arquitectura consisteix bàsicament en un dimoni (ubusd) que proporciona un mecanisme per a que altres dimonis del sistema s'hi puguen registrar i enviar-s'hi missatges. Aquests dimonis especifiquen una sèrie de camins, baix un determinat espai de noms, els quals poden proporcionar diversos procediments amb varis arguments que al seu torn poden contestar amb un missatge.

En altres paraules, es crea un micro-bus on tots els *daemons* registrats poden intercanviar missatges. Cada dimoni especifica uns camins que defineixen els procediments de què disposa i els paràmetres que admeten i pot contestar amb un altre missatge.

Per a la crida de procediments amb paràmetres i per a les respostes, s'utilitza el format JSON.

Ubus proporciona, entre altres coses, un binari per a comunicar-se amb *ubusd* a guions o a la línia d'ordres, per a la crida de procediments i per als missatges de resposta utilitza el format JSON vejam alguns exemples senzills:

- *ubus list*
  - Aquesta opció mostra tots els espais de noms registrats amb el servidor RPC:

```
root@OpenWrt:~# ubus list
dhcp
hostapd.wlan1
log
network
network.device
network.interface
network.interface.WLAN_Casa
network.interface.WLAN_Feina
network.interface.lan
network.interface.loopback
network.interface.wan
network.interface.wan6
network.interface.wwan
network.wireless
service
session
system
uci
root@OpenWrt:~#
```

Il·lustració 2: Mostrant els namespaces registrats a ubus

- *ubus call*
  - Aquesta ordre crida a un procediment d'un dimoni registrat, cal passar-li el camí sencer i el nom del procediment (una llista dels procediment i els paràmetres acceptats així com diversos exemples es poden consultar a la documentació d'ubus[7]).
  - Un exemple de crida a procediment, en aquest cas sense paràmetres, podria ser la a següent, que serveix per a obtindre la informació del sistema:

```
root@OpenWrt:~# ubus call system info
{
        "uptime": 165435,
        "localtime": 1427040023,
        "load": [
                21792,
                17856,
                18432
        ],
        "memory": {
                "total": 261791744,
                "free": 240893952,
                "shared": 0,
                "buffered": 569344
        },
        "swap": {
                "total": 0,
                "free": 0
        }
}
root@OpenWrt:~#
```

Il·lustració 3: Executant un procediment ubus

Per a poder accedir-hi via web, què és el nostre objectiu, OpenWRT proporciona un mòdul per a uhttpd, anomenat uhttpd-mod-ubus que ens permet fer crides ubus via HTTP.

Les crides s'han d'enviar a la URL «/ubus» utilitzant el mètode POST.

Aquest mòdul utilitza l'estàndard jsonrpc 2.0, pel que un dels requisits imprescindibles és tenir instal·lat el paquet RPC a l'OpenWRT, si no es satisfà aquesta dependència mai no podrem accedir-hi via HTTP.

L'uhttpd utilitza llistes de control d'accés (ACL), el que permet definir amb bastant nivell de detall a quins mòduls i procediments es permet l'accés. A l'instal·lar rpcd, se'n creen dues per defecte, anomenades «unauthenticated» i «superuser», la primera defineix els mètodes d'inici de

sessió (però no permet fer res més que logar-se), mentre que la segona defineix una ACL molt insegura que permet l'accés sense restriccions a tot. Pel moment s'ha utilitzat aquesta per a eliminar problemes en el desenvolupament inicial, però se n'hauria de crear de més restrictives per al projecte.

En darrera instància, és l'RPCD qui està manejant-ho, així que cal afegir una entrada al fitxer /etc/config/rpcd com la següent:

```
config login
        option username 'root'
        option password '$p$root'
        list read '*'
        list write '*'
```

Il·lustració 4: Configuració RPCD

En aquest punt, el sistema està preparat per a rebre crides ubus via HTTP.

4.2 Crida a procediments via HTTP:

Per a poder fer crides, primer s'ha d'iniciar sessió, per a aconseguir-ho s'ha d'enviar un missatge com el següent (a la url /ubus)[7]:

{ "jsonrpc": "2.0", "id": 1, "method": "call", "params": ["00000000000000000000000000000000", "session", "login", { "username": "root", "password": "contrasenya" }] }

El missatge fa una crida («call») al mètode «login» del mòdul «session» amb dos paràmetres, l'usuari i la contrasenya. Aquest és un cas especial en què s'utilitza com a identificador de sessió 23 zeros, un identificador estàndard que només s'utilitza per a iniciar sessió.

Si la resposta és positiva, es rebrà una missatge de resposta similar al següent:

{"jsonrpc":"2.0","id":1,"result":[0,
{"ubus_rpc_session":"8afa69468f18af4e648d1e900614c2bf","timeout":300,"expires":300,"acls":
{"access-group":{"superuser":["read","write"],"unauthenticated":["read"]},"ubus":{"*":
["*"],"session":["access","login"]},"uci":{"*":["read","write"]}},"data":{"username":"root"}}]}

D'aquest missatge s'obté l'identificador de sessió (*ubus_rpc_session*), el *timeout* i el temps d'espiració de la sessió, així com la ACL i el nom d'usuari que ha iniciat la sessió.

A JavaScript, les respostes a peticions HTTP fetes mitjançant l'objecte *XMLHttpRequest[4]* es poden manejar de manera síncrona o asíncrona.

El maneig asíncron funciona de la següent manera, creem una instància de *XMLHttpRequest* i definim el mètode *onreadystatechange*:

```
var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {
     if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
     var myArr = JSON.parse(xmlhttp.responseText);
     myFunction(myArr);
     }
}
```

El mètode *onreadystatechange* és l'encarregat de manejar la resposta, el definim com una funció que comprova si el *readyState* és 4 i si *l'state* és 200 (resposta correcta), en cas afirmatiu, converteix la resposta en un objecte JavaScript i crida a una funció que s'encarregarà de processar la resposta en un fil propi per a no bloquejar la resta del guió mentre s'espera la resposta.

4.3 Codi JavaScript per a fer peticions *ubus* sobre http.

Al codi JavaScript, quan la pàgina es carrega, es crida a la funció *getSysInfo*(), la qual crea una petició mitjançant l'objecte ***XMLHttpRequest***, i defineix una funció de maneig de la resposta que romandrà esperant la resposta en un fil d'execució propi, pel que funciona de manera asíncrona.

Aquest enfocament asíncron, té diversos avantatges, principalment, que la pàgina no es bloqueja esperant la resposta. En el nostre cas, no és ideal, ja que fins que no rebem l'identificador de sessió no podrem sol·licitar la informació que volem, així que la funció que processa la resposta és la encarregada de crear una altra petició, enviar-la i establir una funció que s'encarregue de manejar la resposta en un segon fil, en una mena de cascada de fils, cosa que seria molt més senzilla amb un enfocament síncron, però aquest mètode està «deprecated», pel que he optat per no utilitzar-lo.

Així, quan es rep una resposta satisfactòria del missatge d'inici de sessió, es crida a la funció getData() que recupera el ubus_rpc_session i l'emmagatzema en una caixa de text oculta en la web per a que siga accessible per a la resta de funcions.

A continuació crida a la funció querySysInfo(), que recupera el ubus_rpc_session de la caixa de text i crea i envia una nova request al mètode «info» del mòdul «system» d'ubus, amb un missatge com el següent, que inclou el identificar de sessió recollit abans (recuperat de la caixa de text i emmagatzemat en la variable «sid»:

```
{ "jsonrpc": "2.0", "id": 1, "method": "call", "params": [
    "' + sid + '", "system", "info", { } ] }
```

La funció paintSysInfo(arr), rep el missatge de resposta a la crida de dalt, convertit en un

objecte de JavaScript mitjançant la funció «*JSON.parse*» per a facilitar-ne l'accés als camps i s'encarrega de formatar adequadament i mostrar en la plana web les dades rebudes.

S'han creat funcions per a formatar adequadament el temps de funcionament («uptime»), ja que el valor retornat és en segons, així com per a l'hora del sistema que ve també en un format no llegible pels humans, i comprova si hi ha memòria d'intercanvi («swap»), ja que en aquests casos és molt probable que no n'hi haja d'instal·lada, i en eixe cas mostra un missatge i no 0/0.

4.4 Creació d'un *daemon* bàsic al controlador capaç d'obtindre les adreces IP establertes en un fitxer de configuració predefinit.

El dimoni s'ha desenvolupat en C, per a crear-lo s'ha utilitzat l'esquelet disponible al *daemon-howto* del projecte TLDP, una referència bàsica sobre *daemons* en Linux, on s'expliquen molts conceptes bàsics sobre *daemons*.

El *daemon*, ha de permetre el pas de paràmetres de l'ínia d'ordres, amb la qual cosa, el primer que fa és cridar a una funció anomenada «*readParameters*», que s'encarrega de recuperar els paràmetres d'entrada mitjançant la funció *getopt*().

Inicialment, s'han programat dos opcions que es poden passar per l'ínia d'ordres:

- -v: si s'executa amb aquesta opció, el dimoni mostrarà per pantalla la informació de la versió i sortirà.
- -h: si s'executa amb aquesta opció, el dimoni mostra l'ajuda per pantalla i surt.
- -t: si es passa aquest paràmetre, aquest substituirà al període per defecte de descobriment de la xarxa, que està fixat en 300 segons.

Una vegada comprovats els paràmetres i modificada la configuració si s'escau, es bifurca en dos processos mitjançant la funció *fork()* i finalitza el procés pare, amb un codi de sortida adient. Si tot ha anat bé, el procés fill seguirà en marxa, però necessitarà una nou SID per a no passar a convertir-se en un procés zombi (ja que el procés líder de la sessió ha finalitzat), pel que a continuació es crida a la funció *setsid*(), si es produeix un error es sortirà amb un codi adient.

A continuació, crida a *umask*(0), canvia de ruta al directori arrel i tanca els descriptors STDIN, STDOUT i STDERR, ja que un dimoni no hauria d'interactuar mai de manera directa amb l'usuari, si no mitjançant els *logs* del sistema.

Després, estableix una funció s'encarregarà de manejar les senyals d'interrupció del procés:

SIGKILL, SIGINT i SIGTERM per a realitzar un tancament elegant, anomenada *handleSignals*().

Fins ací tota la inicialització del *daemon*, si tot ha anat bé, inicia un bucle «while» controlat per una variable anomenada *bucle_on*, que val u i que canviarà a zero si es rep una senyal d'interrupció. Aquest bucle, crida a una funció anomenada *getIPs*(), que s'encarrega de recollir del fitxer de configuració les IP predefinides de tres AP i les emmagatzema en una matriu de registres com el següent:

```
typedef struct{
    char name[128];
    char address[15];
} device_t;
```

Per al fitxer de configuració s'ha optat per la llibreria «libconfig» que defineix un format de fitxer i proporciona funcions per a recuperar dades d'ells, així com per a afegir-ne i modificar-ne.

El fitxer de configuració que s'ha definit és el següent i s'ha de situar a /etc/drizzle.conf:

```
# preliminar configuration file that lists the IP adresses of the
# maneageable APs.
version ="0.1";

network:
{
        devices = ( {
                name = "localhost";
                IPv4Address = "192.168.2.1";
                },
                {
                name = "AP1"
                IPv4Address = "192.168.2.2";
                },
                {
                name = "AP2"
                IPv4Address = "192.168.2.3";
                },
                {
                name = "AP3"
                IPv4Address = "192.168.2.4";
                } ); #devices

}; #network;
```

Il·lustració 5: El fitxer /etc/drizzle.conf

La funció recupera totes les dades que hi haja i les emmagatzema en la matriu de *device_t* passada com a paràmetre. Recupera tant els noms com les adreces.

4.5 Compilació creuada i creació dels paquets OpenWRT:

Per a realitzar aquest punt, la documentació del projecte *OpenWRT* no ha estat suficient, pel que s'ha hagut de recórrer a Internet. Per a construir el paquet calen, bàsicament tres coses:

- El *buildroot* o l'SDK *d'OpenWRT*. El primer conté al segon i permet la construcció d'una imatge de la distribució a partir del codi font per a l'arquitectura desitjada i, per tant, la compilació creuada (*cross-compile*) per a les diferents arquitectures suportades per *OpenWRT*. Amb aquesta sistema, es poden compilar paquets individuals, però una alternativa és l'SDK, (que es compila al construir la imatge si utilitzem el *buildroot*) que conté la *toolchain* necessària per a la compilació creuada de paquets individuals.
- Un fitxer *Makefile* estàndard per a compilar correctament.
- Un fitxer *Makefile* especial, específic *d'OpenWRT*, amb instruccions relatives al paquet, la seua compilació, instal·lació, desinstal·lació, URL de descàrrega, etc...

Tots els passos s'han de realitzar amb un usuari sense privilegis.

Per a la compilació de *drizzle*, jo he optat pel *buildroot[9]* i el primer pas per a compilar amb ell consisteix a obtenir el codi font de la distribució del git oficial:

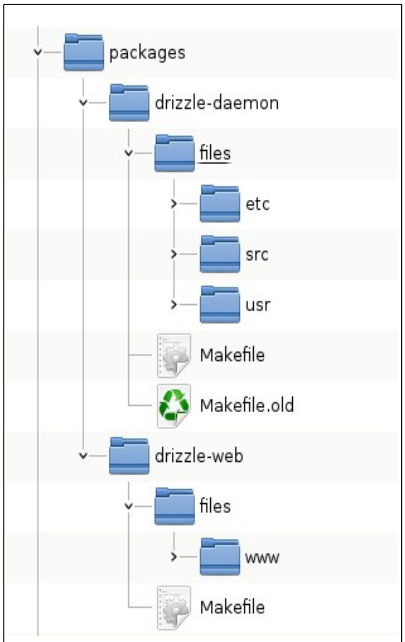
```
$git clone git://git.openwrt.org/openwrt.git
```

Aquesta ordre, crearà una directori *openwrt* al directori on siga executat, i contindrà tots els fitxers necessaris per a construir la imatge.

A continuació, el que he fet ha estat crear una estructura de directoris per al programari similar a la que utilitzen altres paquets de la distribució:

Aquest arbre consta d'una directori per a cada paquet, en aquest cas «drizzle-daemon» i «drizzle-web», què contenen un fitxer anomenat *Makefile*, corresponent al *Makefile d'OpenWRT* i un directori *files* on ja sí que hi ha els els fitxers corresponents.

Aquests fitxers estan organitzats en carpetes, en el cas del *daemon*, el codi font hi és a la carpeta *src* amb el seu *Makefile* típic i els fitxers de codi font, de capçaleres i demés, mentre que la resta de directoris, es corresponen amb directoris típics d'un sistema Linux,



Il·lustració 6: Estructura de directoris de desenvolupament

llestos per a ser copiats en el procés d'instal·lació.

En aquest cas, *files/etc* conté el fitxer de configuració del *daemon* i *files/usr/bin* conté el fitxer executable *drizzled* què s'obté al fer la compilació creuada, mitjançant make i que serà copiat a la carpeta homònima de l'encaminador.

Per la seua banda, la plana web del controlador, conté una carpeta /files/www, que conté els fitxers html, css i js que formen el web, llestos per a ser copiats al directori homònim de l'encaminador.

Els arxius *Makefile d'OpenWRT*

A continuació mostraré els punts d'interès dels fitxers Makefile d'OpenWRT utilitzats per a la creació dels paquets, fets a partir de plantilla:

- Drizzle-daemon

El fitxer està dividit en varis blocs, el primer què es fa és incloure un fitxer de macros predefinides que facilita enormement la compilació. A continuació, s'introdueix la informació del paquet, com nom del paquet, versió, llicència i fitxer de llicència, en el nostres cas GPL 3 o posterior.

```
include $(TOPDIR)/rules.mk

PKG_NAME:=drizzle-daemon
PKG_VERSION:=0.0.1
PKG_RELEASE:=$(PKG_SOURCE_VERSION)
PKG_LICENSE:=GPL-3.0+
PKG_LICENSE_FILE:=../../COPYING


PKG_BUILD_DIR:=$(BUILD_DIR)/$(PKG_NAME)-$(PKG_VERSION)


include $(INCLUDE_DIR)/package.mk


define Package/drizzle-daemon
    SECTION:=Network
    CATEGORY:=Drizzle
    TITLE:=Drizzle daemon
    URL:=http://dev.qmp.cat/projects/drizzel
    DEPENDS:=+uhttpd +ubus +libc +libconfig +libssh2 +libblobmsg-
ubus +libpthread
endef


define Package/drizzle-daemon/description
        An attempt to create a network of centrally managed acces
sed on batman advanced. This package provides the daemon.
endef
```

Il·lustració 7: OpenWRT Makefile per a drizzle-daemon (I)

La variable PKG_BUILD_DIR, es deixa per defecte, i s'inclou un altre fitxer de macros.

La compilació es fa mitjançant un sistema de menús tipus *ncurses*, molt similar a la compilació del nucli Linux, i per a poder compilar els nostres paquets, aquests han d'aparèixer a aquest menú, el menú on apareguin, el nom que tindran, així com la url des de la que es pot descarregar i les dependències d'altres paquets es configuren al següent bloc *define*.

El següent pas és establir la descripció curta que acompanya a cada paquet. S'ha d'intentar que càpiga en una línia.

A continuació ve la part més complexa del fitxer, la compilació del paquet.

A la clàusula Build/Prepare, el que es fa és crear un directori on compilar el nostre paquet, i els subdirectoris necessaris, així com copiar tot allò que necessitem copiar, en aquest cas creem la estructura de directoris $(PKG_BUILD_DIR)/etc (recordem que aquesta variable es defineix al començament del fitxer) i es copien el codi font i el fitxer de configuració a aquesta ruta.

```
define Build/Prepare
        mkdir -p $(PKG_BUILD_DIR)/etc
        $(CP) ./files/src/* $(PKG_BUILD_DIR)/
        $(CP) ./files/etc/drizzle.cfg $(PKG_BUILD_DIR)/etc/
endef


define Build/Configure

endef

#define Build/Compile
#
#endef

define Package/drizzle-daemon/install
        $(INSTALL_DIR) $(1)/usr/bin
        $(INSTALL_DIR) $(1)/etc
        $(INSTALL_CONF) $(PKG_BUILD_DIR)/etc/drizzle.cfg $(1)/etc/
        $(INSTALL_BIN) $(PKG_BUILD_DIR)/drizzled $(1)/usr/bin/
endef


define Package/drizzle-daemon/preinst
endef


define Package/drizzle-daemon/postinst
endef


define Package/drizzle-daemon/prerm
endef


define Package/drizzle-daemon/postrm
endef

$(eval $(call BuildPackage,drizzle-daemon))
```

Il·lustració 8: OpenWRT Makefile per al drizzle-daemon (II)

$(CP) és una macro que fa algunes comprovacions abans de copiar per a evitar errors i es recomana el seu ús malgrat que es pot utilitzar *cp* de la manera habitual.

El següent pas és «preparar» els directoris destí on s'han de copiar els fitxers, això es defineix al *define* «Package/drizzle-daemon/install». Per a simplificar aquest pas, s'han predefinit unes macros que realitzen una sèrie de comprovacions per nosaltres, com $(INSTALL_DIR), $(INSTALL_CONF) i $(INSTALL_BIN) que s'utilitzen per a crear el directoris d'instal·lació si aquests no existeixen, i copiar els fitxer de configuració i binaris respectivament d'una manera robusta i sense complicacions.

La darrera línia és estàndard i s'ha d'executar sempre per a la correcta construcció del paquet.

- Drizzle-web

El Makefile segueix els mateixos criteris que el del daemon, la següent imatge mostra els camps sensibles d'analitzar:

```
define Build/Prepare
	mkdir -p $(PKG_BUILD_DIR)
endef

define Build/Configure
endef

define Build/Compile
	true
endef

define Package/drizzle-web/install
	$(CP) ./files/* $(1)/
endef

define Package/drizzle-web/preinst
endef

define Package/drizzle-web/postinst
endef

define Package/drizzle-web/prerm
endef

define Package/drizzle-web/postrm
endef

$(eval $(call BuildPackage,drizzle-web))
```

Il·lustració 9: OpenWRT Makefile per a drizzle-web

Després d'establir la informació del paquet de la mateixa manera que al daemon, s'arriba al define Build/Prepare, que en aquest cas, senzillament crea el directori de compilació de la mateixa

manera que abans, i el «define Package/drizzle-web/install» senzillament copia la estructura de directoris amb els seus fitxers al directori arrel del router, representat als Makefiles d'OpenWRT amb la variable $(1).

Finalment, s'executa la línia de construcció del paquet.

La compilació de la distribució[10]

Ara hi ha dues opcions, compilar la distribució completa, incloent els dos paquets de drizzle o compilar paquets individuals.

A la distribució hi ha una sèrie de paquets estàndard a escollir, però n'hi ha molts més (estem parlant d'una distribució de Linux completa.

Per a recuperar els fitxers Makefile de tots ells, haurem de descarregar els anomenats «feeds», el que s'aconsegueix d'aquesta manera:

```
~$ cd openwrt

openwrt$ scripts/feeds update -a && scripts/feeds install -a
```

Això farà que es descarreguen tots els *makefiles* de tots els «feeds» configurats i que s'instal·lin.

Per a que els nostres paquets apareguin al menú de selecció, haurem d'afegir el nostre propi «*feed*», el que es fa afegint una línia nova al fitxer feeds.conf, amb la ruta del meus paquets, què es /home/jordi/drizzle/packages:

```
src-git packages https://github.com/openwrt/packages.git;for-14.07
src-git luci https://github.com/openwrt/luci.git;luci-0.12
src-git routing https://github.com/openwrt-routing/packages.git;for-14.07
src-git telephony https://github.com/openwrt/telephony.git;for-14.07
src-git management https://github.com/openwrt-management/packages.git;for-14.07
src-git oldpackages http://git.openwrt.org/14.07/packages.git
#src-svn xwrt http://x-wrt.googlecode.com/svn/trunk/package
#src-svn phone svn://svn.openwrt.org/openwrt/feeds/phone
#src-svn efl svn://svn.openwrt.org/openwrt/feeds/efl
#src-svn xorg svn://svn.openwrt.org/openwrt/feeds/xorg
#src-svn desktop svn://svn.openwrt.org/openwrt/feeds/desktop
#src-svn xfce svn://svn.openwrt.org/openwrt/feeds/xfce
#src-svn lxde svn://svn.openwrt.org/openwrt/feeds/lxde
#src-link custom /usr/src/openwrt/custom-feed
src-link drizzle /home/jordi/drizzle/packages
feeds.conf (END)
```

Il·lustració 10: Feeds de paquets d'OpenWRT

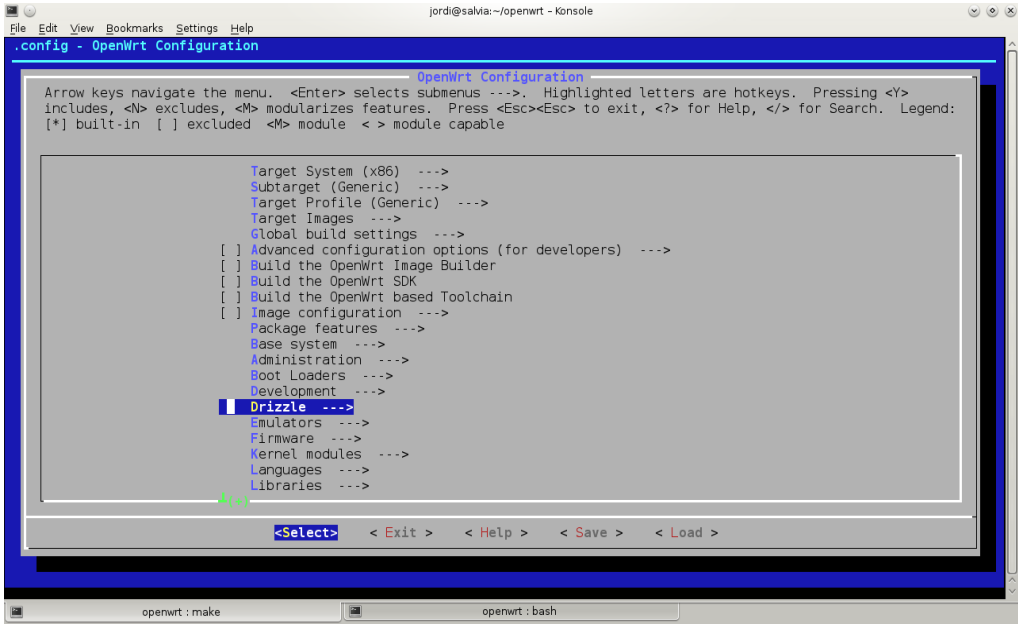
En aquest cas és un directori del disc dur, pel que utilitzarem el tipus de *feed* «src-link» i una etiqueta per a identificar-lo, en el meu cas *drizzle*.

Una vegada ja tenim això fet, caldrà que executem «make menuconfig» i ens apareixerà el menú de selecció de paquets d'OpenWRT:

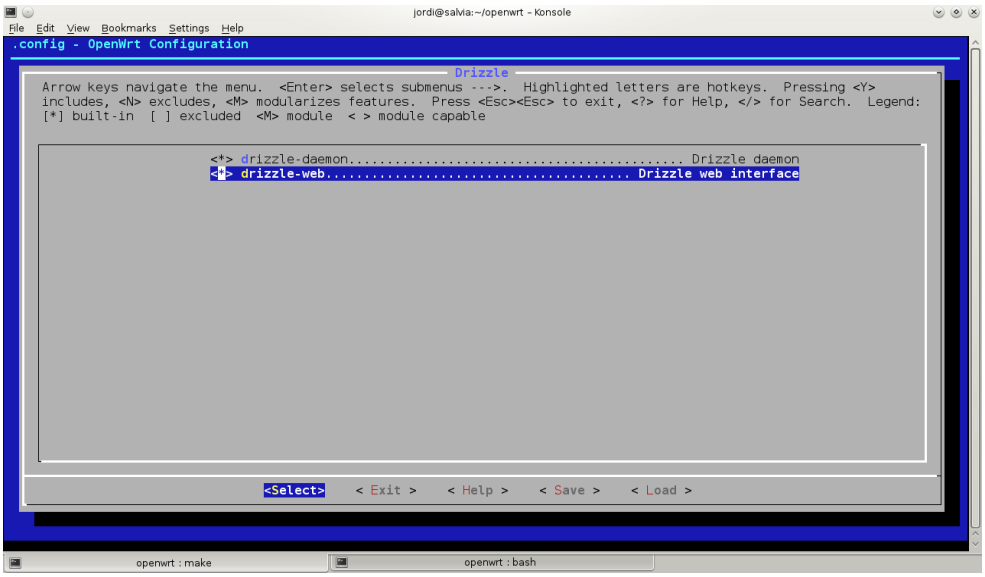
```
[openwrt]$ make menuconfig
```

L'aparença es similar a la del menú de configuració del nucli, com ja havíem dit abans, haurem d'escollir l'arquitectura per a la que volem compilar al menú *Target System*, en el nostre cas *x86*.

Si abaixem trobarem un menú anomenat *Drizzle*, i si hi accedim, dues entrades, una per a drizzle-daemon i una altra per a drizzle-web, polsant la barra espaiadora dues vegades els seleccionarem per a ser construïts i instal·lats a la imatge de la distribució.



Il·lustració 11: Menú de compilació buildroot OpenWRT (I)



Il·lustració 12: Menú de sel·lecció de paquets buildroot OpenWRT

Els paquets corresponents a les dependències que hàgim establert al *Makefile*, tal i com *libssh2* o *libconfig*, apareixen marcats automàticament per a la seva instal·lació una vegada marquem «drizzle-daemon». Les llibreries solen estar dins del menú «*Libraries*»

Podem continuar personalitzant tot allò que vulguem a la imatge i una vegada acabem, premerem «*exit*» fins a sortir del tot. Ens demanarà si volem alçar la configuració, li diem que sí.

Per a començar la construcció de la imatge, i per tant la compilació del «drizzle-daemon», executem «*make*» al *shell* per a començar el procés de compilat. Si tenim un processador de més d'un nucli, convé utilitzar el paràmetre -jn, sent «n» el nombre de nuclis que tinguem més un, per a que utilitze varis processos alhora i el procés vaja més ràpid, ja que tarda una bona estona :

```
[openwrt]$ make -j3
```

Apareixeran una cascada de textos en color marró corresponents als paquets que es van compilant amb *make*.

Si hi ha problemes compilant algun paquet, podem utilitzar la següent ordre per a obtenir una sortida extensa:

```
[openwrt]$ make -j3 V=s
```

Per a resoldre problemes de compilació resulta de molta utilitat poder construir paquets individuals. Per a poder compilar, per exemple, el «drizzle-daemon» individualment , podem utilitzarem la següent ordre:

```
[openwrt]$ make package/drizzle-daemon/{clean,compile,install} V=s
```

Si la compilació de la imatge finalitza amb èxit, obtindrem una imatge fresca llesta per ser copiada a la *flash* al dispositiu.

També podem trobar els paquets individuals (fitxer amb extensió .ipx) a la següent ruta, suposant que *openwrt* és la ruta on vàrem clonar el diposit git:

openwrt/bin/x86/packages

Només ens queda copiar-los a l'encaminador mitjançant scp, per exemple, i instal·lar-lo:

```
[openwrt]$ scp bin/x86/packages/drizzle/drizzle-daemon_0.0.1_x86.ipk
root@192.168.1.254:drizzle-daemon_0.0.1_x86.ipk

Enter passphrase for key '/home/jordi/.ssh/id_rsa':

root@192.168.1.254's password:

drizzle-daemon_0.0.1_x86.ipk
100% 8210     8.0KB/s   00:00

[openwrt]$ ssh root@192.168.1.254
```

```
Enter passphrase for key '/home/jordi/.ssh/id_rsa':
root@192.168.1.254's password:
BusyBox v1.22.1 (2014-09-21 14:57:34 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.


  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 BARRIER BREAKER (14.07, r42625)
 -----------------------------------------------------
  * 1/2 oz Galliano         Pour all ingredients into
  * 4 oz cold Coffee        an irish coffee mug filled
  * 1 1/2 oz Dark Rum       with crushed ice. Stir.
  * 2 tsp. Creme de Cacao
 -----------------------------------------------------
root@OpenWrt:~# opkg install drizzle-daemon_0.0.1_x86.ipk
Installing drizzle-daemon (0.0.1) to root...
Configuring zlib.
//usr/lib/opkg/info/zlib.postinst: line 4: default_postinst: not found
Configuring libopenssl.
//usr/lib/opkg/info/libopenssl.postinst: line 4: default_postinst: not found
Configuring libssh2.
//usr/lib/opkg/info/libssh2.postinst: line 4: default_postinst: not found
Configuring libconfig.
//usr/lib/opkg/info/libconfig.postinst: line 4: default_postinst: not found
Configuring drizzle-web.
//usr/lib/opkg/info/drizzle-web.postinst: line 4: default_postinst: not found
Configuring drizzle-daemon.
```

```
Collected errors:
 * pkg_run_script: package "zlib" postinst script returned status 127.
 * opkg_configure: zlib.postinst returned 127.
 * pkg_run_script: package "libopenssl" postinst script returned status 127.
 * opkg_configure: libopenssl.postinst returned 127.
 * pkg_run_script: package "libssh2" postinst script returned status 127.
 * opkg_configure: libssh2.postinst returned 127.
 * pkg_run_script: package "libconfig" postinst script returned status 127.
 * opkg_configure: libconfig.postinst returned 127.
 * pkg_run_script: package "drizzle-web" postinst script returned status 127.
 * opkg_configure: drizzle-web.postinst returned 127.
root@OpenWrt:~#
```

Com es pot veure a la transcripció, apareixen alguns missatges, però, el paquet s'ha instal·lat correctament.

A la majoria de documentació es fiquen les instruccions de compilació a «Build/Compile», i la documentació no excel·leix, per aquesta raó vaig perdre moltíssim temps amb errades d'enllaçat intentant escriure correctament les instruccions de compilat, però finalment me'n vaig adonar que estava compilant el *daemon* amb el *toolchain* del meu Linux, enlloc del *toolchain* del *buildroot*.

Al tractar-se d'un sistema embegut, utilitza una llibreria de C alternativa a la *glibc* de GNU, atès que aquesta genera binaris «unflats», així que utilitza una anomenada *ulibc*. Així, el *daemon* es compilava, però al fer les comprovacions posteriors abans de donar per bo el fitxer (tot això ho fa el make d'OpenWRT, el *builroot* se n'adonava de que s'estava utilitzant la glibc.so.6 (glibc 2) enlloc de la glibc.so.0 (ulibc).

Finalment vaig llegir que si no es defineix, s'utilitza la macro de compilació predeterminada, que funciona de meravella. I així es com ho estic fent des d'aleshores, he comprovat que si definisc la clàusula, compila amb la llibreria de C equivocada.

4.6 Consulta periòdica de les adreces IP dels dispositius a gestionar del fitxer de configuració.

El bucle principal del *daemon*, s'encarrega de recuperar les adreces, emmagatzemar-les en memòria en una variable global anomenada «*devices*», així com el nombre de dispositius trobats al fitxer en una altra variable global anomenada «*count*».

Per a recuperar-les utilitza la funció get_ips() analitzada anteriorment, que té la següent

capçalera:

```
int get_ips(device_t *devices);
```

On la que se li passa un vector els elements del qual son del tipus «device_t» i que retorna el nombre de dispositius trobats. A continuació, senzillament fa una crida a la funció estàndard sleep, amb el nombre de segons a esperar, abans de tornar a llegir el fitxer.

4.7 Registre del *daemon* a la interfície *ubus* del controlador i definició dels procediments associats.

Per a aquest apartat, s'ha hagut d'acudir directament als exemples[11] subministrats amb la llibreria, donada la escassetat de documentació trobada. En realitat no s'ha trobat cap documentació més que aquesta.

Així, el *daemon*, abans d'entrar en bucle, crea un fil d'execució en què s'enregistra amb *ubus* i queda escoltant, en paral·lel. L'objectiu és que més endavant, des de la pàgina web, es puguen enviar odres al *daemon* via *ubus*.

La funció que realitza aquesta tasca es diu ubus_register(), i el que fa és:

- Inicialitzar el bucle ubus, mitjançant la funció uloop_init()
- Crear un *socket* d'ubus per a connectar-s'hi
- Si la connexió amb *ubus* es correcta, afegir la connexió al bucle *d'ubus*

Si tot això ha anat correctament, ja tenim un bucle *d'ubus* propi connectat amb *l'ubus* del sistema. A continuació es crida a la funció server_main(), que hereta del nom dels exemples, i què és la que s'encarregarà d'enregistrar el *daemon* a *ubus* amb les seues accions, així com definir les funcions que les implementaran.

Per a definir el nostre objecte a *ubus*, necessitarem tres variables, una que defineixi l'objecte, una altra que defineixi el tipus i una altra que defineixi els mètodes i els associe amb les funcions que els implementaran.

En aquest cas són: **drizzle_object**, **dizzle_object_type** i **drizzle_methods[]**.

A la captura es pot analitzar amb més detall :

```
/* Next, we define the methods for the name drizzle,
   we do this with an array of struct ubus_method's */


static const struct ubus_method drizzle_methods[] = {

  UBUS_METHOD("set_config", drizzle_set_config, set_config_policy),
  UBUS_METHOD("get_config", drizzle_get_config, get_config_policy),
  /* as im_alive doesn't get any parameter, we don't need a policy */
  UBUS_METHOD("im_alive", drizzle_im_alive, im_alive_policy),
};

/* We define the type of our ubus object, so when we execute ubus
   list, it appears there. This is necessary before we especify its
   attributes */


static struct ubus_object_type drizzle_object_type =
   UBUS_OBJECT_TYPE("drizzled", drizzle_methods);

 /* We especify the attributes of the object */

static struct ubus_object drizzle_object = {
   .name = "drizzled",
   .type = &drizzle_object_type,
   .methods = drizzle_methods,
   .n_methods = ARRAY_SIZE(drizzle_methods),
 };
```

Il·lustració 13: Variables de definició de l'objecte drizzled a ubus

Bàsicament, a dizzle_object es defineix el nom de l'objecte que apareixerà quan es cride a «ubus list», en aquest cas drizzled, quins mètodes té, i el nombre de mètodes. Són funcions estàndard, totes definides a libubus.h.

Així, server_main(), intenta afegir aquest objecte a *ubus*, i registrar un vigilant per a capturar els successos, i si tot va bé, inicia el bucle *ubus*:

```
static void server_main(void)
{
  int ret;

  /* If we are here is because we successfilly connected to ubus, then
     we can try to add the object, so it could be accessed via ubus */

  ret = ubus_add_object(ctx, &drizzle_object);
  if (ret)
    syslog(LOG_ERR, "failed to connect!");

  syslog(LOG_INFO,"Object drizzled added successfully!");
  /* then it's time to register a subscriber to catch the events */
  ret = ubus_register_subscriber(ctx, &drizzle_event);
  if (ret)
    syslog(LOG_ERR, "failed to add watch handler: %s\n",ubus_strerror(ret));
;
  else
    syslog(LOG_INFO,"Watch handler added successfully!");
  uloop_run();
}
```

Il·lustració 14: La funció server_main()

El *daemon* admet tres mètodes distints:

- «im_alive» : Aquest mètode no accepta paràmetres i retorna la informació d'estat de tots els dispositius gestionats
- «get_config» : Aquest mètode s'utilitza per a obtindre la configuració de la primer radio del dispositiu amb l'adreça IP passada com a paràmetre o la de totes si enlloc d'una adreça IP rep la cadena «All»
- «set_config» : Aquest mètode s'utilitza per a aplicar canvis en la configuració de la primer radio del dispositiu especificat o de tots. Rep obligatòriament tres paràmetres, l'adreça IP, el paràmetre a configurar i el nou valor que volem fixar.

Cadascun d'aquests mètodes per a ser vàlid, ha de definir tres coses, declarades al seu UBUS_METHOD:

- Un nom únic en el seu *namespace* (en aquest cas drizzled)
- El punter a funció establert al primer paràmetre d'UBUS_METHOD (la tasca que realitza).
- La *policy*, que estableix els noms dels paràmetres que accepta i el seu tipus.

També resulta convenient crear enumeracions per a identificar els paràmetres.

No entrarem en detall en el funcionament de la llibreria blobmsg, només diré que per a poder manejar les peticions, recuperar els paràmetres i enviar la respost correctament codificada s'ha de fer mitjançant el que s'anomenen *blob's*. OpenWRT inclou biblioteques per al treball amb aquests i codificar-los en JSON.

Per a poder-les utilitzar hi ha que afegir les següents línies #include:

```
#include <libubox/blobmsg_json.h>
#include <libubox/blobmsg.h>
#include <libubus.h>
```

```
enum {
  IM_ALIVE_ID,
  __IM_ALIVE_MAX,
};

enum {
    GET_CONFIG_IP,
    __GET_CONFIG_MAX,
};

enum {
    SET_CONFIG_IP,
    SET_CONFIG_PARAM,
    SET_CONFIG_VALUE,
    __SET_CONFIG_MAX,
};


static const struct blobmsg_policy im_alive_policy[] = {
  [IM_ALIVE_ID] = { .name = "id", .type = BLOBMSG_TYPE_INT32 },
};


static const struct blobmsg_policy get_config_policy[__GET_CONFIG_MAX] = {
    [GET_CONFIG_IP] = { .name = "Address", .type = BLOBMSG_TYPE_STRING },
};

static const struct blobmsg_policy set_config_policy[__SET_CONFIG_MAX] = {
    [SET_CONFIG_IP] = { .name = "Address", .type = BLOBMSG_TYPE_STRING },
    [SET_CONFIG_PARAM] = { .name = "Parameter", .type = BLOBMSG_TYPE_STRING },
    [SET_CONFIG_VALUE] = { .name = "Value", .type = BLOBMSG_TYPE_STRING },
};
```

Il·lustració 15: Definició de procediments ubus

4.8 Obertura de connexions SSH amb els AP recuperats.

Els procediments abans esmentats utilitzen la llibreria libssh2 per a crear les connexions ssh necessàries.

Quan el *daemon*, està en execució, aquest té dos fils d'execució, un que llig periòdicament el fitxer de configuració i un altre que s'enregistra amb *ubus* i queda esperant ordres. A partir d'eixe moment, si s'emet la comanda «im_alive» mitjançant l'ordre *ubus* de la línia d'ordres, aquest obre una connexió SSH (en el seu propi fil) amb cadascuna de les adreces IP i executa la comanda adient, reunint tota la informació en un únic JSON.

La part de l'inici de sessió s'ha pres dels exemples[12] del projecte libssh2, analitzem una mica de quina manera funciona l'establiment d'una sessió amb ssh.

Per a aquesta tasca he creat un punter a funció anomenat ssh_connect(), que s'encarrega de crear un *socket* sobre el qual establir la connexió *ssh*, així com inicialitzar la llibreria *libssh* dins del seu propi fil d'execució:

```
rc = libssh2_init(0);
if (rc != 0) {
  syslog(LOG_ERR, "libssh2 initialization failed (%d)",rc);
  return (void *)-1;
}

/* The first thing to do is create a socket and connect it to
   the server*/

hostaddr = inet_addr(conn_params->address);

sock = socket(AF_INET, SOCK_STREAM, 0);

sin.sin_family = AF_INET;
sin.sin_port = htons(22);
sin.sin_addr.s_addr = hostaddr;

if (connect(sock, (struct sockaddr*)(&sin),
            sizeof(struct sockaddr_in)) != 0) {
  syslog(LOG_ERR, "failed to connect!");
  return (void *)((uintptr_t)-2);
}
```

Il·lustració 16: Establiment de connexions amb libssh2 (I)

```
/* Create a session instance and start it up. This will trade welcome
 * banners, exchange keys, and setup crypto, compression, and MAC layers
 */
session = libssh2_session_init();
if (!session){
  syslog(LOG_ERR, "Failure creating the SSH session");
  return (void *)((uintptr_t)-3);
}

/* tell libssh2 we want it all done non-blocking */
libssh2_session_set_blocking(session, 0);

if (libssh2_session_handshake(session, sock)) {
  syslog(LOG_ERR, "Failure establishing SSH session");
  return (void *)((uintptr_t)-4);
}
```

Il·lustració 17: Establiment de connexions amb libssh2 (II)

Aquesta part no té més complicació, a continuació, s'ha de crear una instància de sessió i iniciar-la, d'aquesta manera, s'informa al servidor *ssh* que volem establir una connexió i s'intercanvien la informació de xifrat, compressió, *banners*, s'intercanvien les claus, etc...

```
  nh = libssh2_knownhost_init(session);

  if(!nh) {
    /* eeek, do cleanup here */
        return (void *)((uintptr_t)-5);
  }

  /* read all hosts from here */
  libssh2_knownhost_readfile(nh, "known_hosts",

                             LIBSSH2_KNOWNHOST_FILE_OPENSSH);

  /* store all known hosts to here */
  libssh2_knownhost_writefile(nh, "dumpfile",

                              LIBSSH2_KNOWNHOST_FILE_OPENSSH);

  fingerprint = libssh2_session_hostkey(session, &len, &type);

  if(fingerprint) {
    struct libssh2_knownhost *host;
#if LIBSSH2_VERSION_NUM >= 0x010206
    /* introduced in 1.2.6 */
    int check = libssh2_knownhost_checkp(nh, conn_params->address, 22,

                                         fingerprint, len,
                                         LIBSSH2_KNOWNHOST_TYPE_PLAIN|
                                         LIBSSH2_KNOWNHOST_KEYENC_RAW,
                                         &host);
```

```
    /* 1.2.5 or older */
    int check = libssh2_knownhost_check(nh, conn_params->address,

                                        fingerprint, len,
                                        LIBSSH2_KNOWNHOST_TYPE_PLAIN|
                                        LIBSSH2_KNOWNHOST_KEYENC_RAW,
                                        &host);
#endif
    fprintf(stderr, "Host check: %d, key: %s\n", check,
            (check <= LIBSSH2_KNOWNHOST_CHECK_MISMATCH)?
            host->key:"<none>");

    /*****
     * At this point, we could verify that 'check' tells us the key is
     * fine or bail out.
     *****/
  }
  else {
    /* eeek, do cleanup here */
    return (void *)((uintptr_t)-6);
    }
  libssh2_knownhost_free(nh);
```

Il·lustració 18: Establiment de connexions amb libssh2 (III)

En aquest punt, el programa encara no s'ha autenticat contra el servidor, només s'ha posat d'acord amb el servidor sobre els termes en què es produïra la connexió. El següent pas és

recuperar la llista de hosts «coneguts» del fitxer «known_hosts» i veure si s'hi troba una coincidència amb el *fingerprint* del servidor al què estem connectant.

Ara, tocaria comprovar quins mètodes d'autenticació ofereix el servidor, però per simplicitat s'ha optat per utilitzar el mecanisme d'usuari/contrasenya:

```
/* On drizzle we have to authenticate with the devices via passsword, so we do it directly */
/* We could authenticate via password */
while ((rc = libssh2_userauth_password(session,conn_params->username, conn_params->password)) ==
       LIBSSH2_ERROR_EAGAIN);
if (rc) {
  syslog(LOG_WARNING, "Authetication by password failed!");
  ssh_shutdown(session, channel, sock);
  libssh2_exit();
  return (void *)((uintptr_t)-7);
} else {
  syslog(LOG_INFO,"\tAuthentication by password succeeded.");
}

/* Exec non-blocking on the remove host */
while( (channel = libssh2_channel_open_session(session)) == NULL &&
       libssh2_session_last_error(session,NULL,NULL,0) ==
       LIBSSH2_ERROR_EAGAIN ){
  waitsocket(sock, session);
}
if( channel == NULL ){
  syslog(LOG_ERR, "Unable to open a session.");
  ssh_shutdown(session, channel, sock);
  libssh2_exit();
  return (void *)((uintptr_t)-9);
}


/* Exec non-blocking on the remote host */
while( (rc = libssh2_channel_exec(channel, conn_params->command)) ==
       LIBSSH2_ERROR_EAGAIN )
  waitsocket(sock, session);

if( rc != 0 ) {
  syslog(LOG_ERR, "Exec error");
  return (void *)((uintptr_t)-10);
}
```

Il·lustració 19: Execució de comandes remotes amb ssh

La funció rep com a paràmetre un *struct* del següent tipus:

```
typedef struct connection_data{
  char address[15];
  char username[64];
  char password[64];
  char command[128];
  char response[BUFFER_LEN];
  int r_value;
} t_con_data;
```

Il·lustració 20: Struct connection_data

Així que formalitzem l'autenticació amb l'usuari i contrasenya rebuts i si tot funciona correctament, finalment iniciem una nova sessió ssh.

A continuació, es crida a la funció libsssh2_channel_exec definida a libssh2 que executa la ordre emmagatzemada a *command* al servidor, i si tot a funcionat bé, crida a la funció ssh_get_answer() que implementa una funció típica de recepció de dades en sockets.

Aquesta funció proporciona el servei de connexió, enviament i recepció de les dades a la resta de funcions definides, anàlogues a les d'ubus, però a nivell ssh:

- int ssh_im_alive();
- int ssh_set_config(char radio, const char *addr, const char *param, const char *value);
- int ssh_get_config(int wireless, const char *addr);

Aquestes funcions són cridades directament per les seves anàlogues, enviant la ordre adient i recuperant la resposta per que aquelles la converteixin en blob i la envien per ubus formatades en JSON.

4.9 Desenvolupament del web

4.9.1 Recuperació de la informació dels dispositius

A l'apartat 4.3, s'explicava el codi necessari per a realitzar peticions a ubus mitjançant http, vejam com s'ha aplicat això a l'hora de desenvolupar la interfície web

Tot es desencadena amb l'event onload() de la pàgina principal, quan succeeix s'utilitza el mètode explicat abans per a emetre una ordre «im_alive» al daemon. Aquest fa la seva feina utilitzant els mecanismes explicats i retorna un JSON.

Aquest JSON es converteix en un objecte de JavaScript mitjançant el mètode JSON.parse(), d'aquesta manera resulta molt senzill accedir a les dades, a través de variable.result[1].

Una vegada s'ha recuperat aquesta informació, es crida a la funció queryDevicesConfig() que repeteix el procés però emetent la ordre «drizzle_get_config» passant com a paràmetre el comodí «All» per a rebre tota la informació disponible i, aprofitant l'asincronia dibuixa en pantalla la taula amb la informació d'estat dels dispositius.

Uns segons després, quan es rep la contestació a la sol·licitud de informació de configuració, es repeteix el procés, s'emmagatzema mitjançant JSON.parse a una variable i s'hi accedeix als seus membres mitjançant variable.result[1].

Aquesta manera és especialment útil, ja que al tractar-se d'una matriu, conté una propietat

length que ens diu el nombre de dispositius dels que s'ha rebut informació.

A la següent figura podem veure la funció que recupera la informació de l'estat i crida a la de recuperació de la informació de configuració mentres mostra la informació d'estat.

```
function querySysInfo(){
    /* Once we have a session id, we query the device for
       its system information, by calling the method
       system info without parameters, just the session id
       we got */
    var sid;
    var request;

    request = new XMLHttpRequest();
    request.open("POST",url,true);
    request.setRequestHeader("Content-type","application/json");
    request.onreadystatechange = function() {
        if (request.readyState == 4 && request.status == 200){
            sessionStorage.status=request.responseText;
            /* for debugging
               console.log(sessionStorage.status);
            */
            var msg = JSON.parse(request.responseText);
            queryDevicesConfig();
            paintSysInfo(msg);
        }// else if (request.readyState == 4 && request.status == 400) {
    }
    sid = document.getElementById("outside").value;
    console.log("SID: " + sid);
    query ='{ "jsonrpc": "2.0", "id": 1, "method": "call", "params": [ "' + sid + '", "drizzled", "im_alive", { } ] }';
    request.send(query);
}
```

Il·lustració 21: Crida a un procediment de drizzled via ubus des de la web

4.9.2 Canvis de configuració

Totes les pantalles estan situades dins d'àrees que s'oculten o es mostren a conveniència segons el menú sobre el que es faça clic.

De la mateixa manera, les «fitxes» de configuració de cada dispositiu són àrees que «floten» a l'esquerra.

Quan accedim al menú «devices setup», s'amaga el div de l'estat i es mostra el de configuració amb tots els dispositius visibles.

Al formulari de dalt, es pot triar sobre quin dispositiu es vol aplicar la configuració, identificant-lo per la seva IP, i només es mostra aquells dispositius sobre els que es vulga treballar.

Així, per a modificar una configuració, es tria l'adreça IP del dispositiu o «All», es tria el paràmetre a modificar (només funciona amb certesa *channel*) i del desplegable es tria el nou valor. En aquest moment, si es fa clic al botó «Submit changes», es crida la funció querySetConfig().

Aquesta funció recull les dades del formulari i construeix una sol·licitud per al daemon, amb l'adreça, el paràmetre i el valor, repetint de nou el procés.

Una vegada contesta, s'emmagatzema la informació a la sessió i es crida a la funció que mostra les fitxes de configuració.

Per a emmagatzemar la informació de configuració i no haver-la de passar entre funcions, s'utilita l'objecte sessionsStorage[13].

Per a acabar amb el bloc de desenvolupament, només puntualitzar que s'han creat unes funcions per a mostrar d'una manera més adient la informació d'estat ja que *ubus* retorna les mesures de temps en un format illegible pels humans.

També m'he recolzat en fulls d'estil bootstrap[14] per a les barres de la memòria lliure/ocupada.

5 Resultats

5.1 Instal·lació.

El resultat obtingut consisteix en dos paquets OpenWRT estàndard:

- drizzle-daemon_0.0.3_x86.ipx
- drizzle-web_0.0.3_x86.ipx

La operativa a seguir per a instal·lar és copiar els dos fitxers al dispositiu mitjançant scp:

```
[jordi@salvia ~]$ scp drizzle-web_0.0.3_x86.ipk root@192.168.1.254:drizzle-
web_0.0.3_x86.ipk
[jordi@salvia ~]$ scp bin/x86/packages/drizzle/drizzle-daemon_0.0.3_x86.ipk
root@192.168.1.254:drizzle-daemon_0.0.3_x86.ipk
```

A continuació es connecta mitjançant ssh:

```
[jordi@salvia ~]$ ssh root@192.168.1.254
Enter passphrase for key '/home/jordi/.ssh/id_rsa':
root@192.168.1.254's password:


BusyBox v1.22.1 (2014-09-21 14:57:34 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.


  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 BARRIER BREAKER (14.07, r42625)
 -----------------------------------------------------
  * 1/2 oz Galliano         Pour all ingredients into
  * 4 oz cold Coffee        an irish coffee mug filled
  * 1 1/2 oz Dark Rum       with crushed ice. Stir.
```

```
  * 2 tsp. Creme de Cacao

 -----------------------------------------------------

root@OpenWrt:~#
```

i s'executea la següents ordre:

```
root@OpenWrt:~# opkg install drizzle-daemon_0.0.3_x86.ipk drizzle-
web_0.0.3_x86.ipk
```

Apareixen uns avisos, que podem ignorar. I si hi han dependències sense complir es satisfan instal·lant els paquets adients si hi ha connexió a Internet o donarà error i no s'instal·larà.

En aquest punt, només queda configurar el fitxer /etc/drizzle.conf segons les nostres necessitats i executar *drizzled*, el binari del *daemon*, que resideix a /usr/bin, i per tant hi és al PATH del sistema.

Alternativament es pot reiniciar el dispositiu i deixar que s'engegui amb el sistema.

5.2 Funcionament de la interfície web

En aquest punt, la interfície web està copiada a /www/drizzle i es pot accedir mitjançant la següent URL:

`http://Device_IP_Address/drizzle/main.html`

Al carregar la pàgina, aquesta sol·licita tant la informació d'estat de tots els dispositius com els paràmetres essencials de la configuració del primer dispositiu wireless d'aquests.

Passats uns segons veurem aparèixer a una taula la següent informació de tots els dispositius:

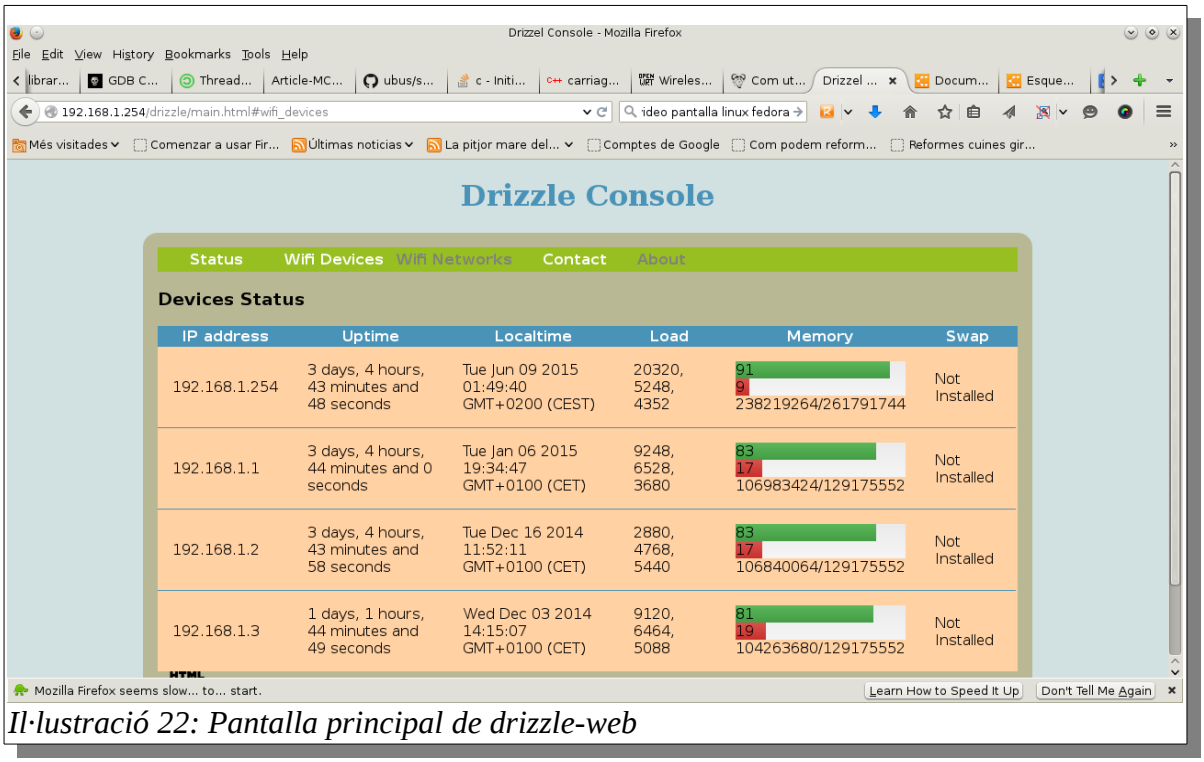
- Adreça IP
- uptime
- Data local
- Càrrega
- Memòria Lliure/Memòria ocupada
- Swap Lliure / Swap Ocupada

Uns segons més tard, tindrem disponible també les dades de configuració abans esmentades, i podrem accedir-hi mitjançant el menú «Wifi Devices».

En cas de voler modificar alguna d'aquestes configuracions, utilitzarem les llistes desplegables de dalt per a seleccionar les dades necessàries:

- Adreça IP del dispositiu a configurar (o «All»)
- Paràmetre a modificar
- Nou valor d'aquest

i finalment prémer «Submit changes» per a fer-los efectius, en un parell de segons hauríem de veure les dades actualitzades.



Il·lustració 22: Pantalla principal de drizzle-web

Il·lustració 23: Pantalla de configuració de dispositius wifi a drizzle-web

6 Conclusions

Els objectius genèrics s'han assolit, és adir, s'ha obtingut un *daemon* plenament funcional que llig la informació dels dispositius gestionats des d'un fitxer i és capaç d'obrir connexions ssh amb aquests si cal per a realitzat les tasques de gestió que se li transmeten des d'una interfície web d'estil agradable que, al seu torn, permet visualitzar la informació d'estat dels dispositius, així com mostrar per quin canal transmeten i canviar aquest paràmetre a un dispositiu individual o a tots alhora.

Si fem una ullada als objectius específics llistats a la taula següent, veurem que la majoria s'han assolit, a excepció del guió per a que el «daemon» s'inicïi amb el sistema, que malgrat estar fet cal depurar-lo i el pas de paràmetres al «daemon» per línia d'ordres, que malgrat estar implementat, al codi es fa cas omís.

Tasca		Assolit
Desenvolupament d'un controlador bàsic		
	Crear una interfície web amb JavaScript capaç d'obtindre la informació de sistema del propi controlador.	✔
	Desenvolupar un «daemon» en C al controlador que recuperi les adreces IP estàtiques hard-coded dels 3 AP, desades a un fitxer	✔
	Parametritzar el «daemon» mitjançant paràmetres de la línia d'ordres	✗
	Fer que aquest agent obri una connexió SSH amb cadascun d'aquests AP.	✔
	Fer que el «daemon» consulti les dades del fitxer periòdicament.	✔
	Registrar el «daemon» a la interfície ubus del controlador.	✔
	Fer que el «daemon», enviï la ordre "I'm alive" i recupere la informació de sistema de tots els AP, via ubus	✔
	Emmagatzemar la informació dels AP en una estructura de dades en memòria i mostrar-la en la web.	✔
Desenvolupament d'un controlador avançat		
	Fer que el «daemon» capture els senyals dels sistema per a tancar-se elegantment, alliberant els recursos.	✔
	Crear una interfície web que ens permeta recuperar el canal pel que transmet el primer dispositiu wireless d'un o de tots els dispositius.	✔
	Crear una interfície web que ens permeta recuperar una dada de configuració wireless (p. ex. el canal) a tots els AP.	✔
	Crear un procediment ubus al daemon per a alterar el canal de transmissió del primer dispositiu wireless d'un dispositiu o tots.	✔
	Crear una interfície web que ens permeta modificar un paràmetre de configuració wireless (p. ex. el canal)	✔
	Crear una interfície web que ens permeta fer modificacions a tots els AP (p. ex. Canviar els canals de totes les AP)	✔

Tasca		Assolit
Refinaments i tasques finals		
	Fer que el «daemon» s'iniciï amb els scripts d'inici del sistema	✗
	Crear dos paquet OpenWRT estàndard, un per al daemon i un per al web	✔

Malgrat no estar al llistat d'objectius específics, hi ha alguns objectiu intrínsecs que no s'han assolit: la interfície web té un deficient control d'errors i presenta alguns problemes d'usabilitat.

A continuació hi ha una llista de les mancances detectades al resultat:

- La interfície web no té un control d'errors adequat.
- La interfície web hauria de mostrar una finestra modal al carregar-se informant a l'usuari de que s'estan recuperant dades.
- El *daemon* es limita a treballar amb un màxim de quatre dispositius, queda pendent un algoritme que controle el funcionament per a un nombre indeterminat de dispositius, limitant el nombre de fils que s'executen alhora i reservant dinàmicament la memòria.

6.1 Coneixements aplicats apresos al Màster.

Aquest projecte m'ha permès aplicar la majoria dels coneixements apresos al Màster de manera transversal, ja que un desenvolupat d'aquestes característiques avarca moltes matèries.

A continuació detallo els coneixements què he aplicat, per assignatures:

Assignatura	Coneixement
Desenvolupament web	A aquesta assignatura vaig aprendre a programar webs tant de la banda client com de la banda de servidor, pel que els coneixement obtinguts sobretot de css, m'han resultat d'inestimable ajuda, doncs css no és senzill.
Implantació de sistemes de programari lliure	D'aquesta assignatura vaig aprendre que per a portar a terme un projecte IT, comporta moltíssima feina i requereix la presa de moltes decisions. A mi personalment em va aportar una certa maduresa en la presa de decisions què he hagut d'aplicar ací intensament.
Sistema Operatiu GNU/Linux bàsic	Aquesta assignatura em va aportar uns coneixements específics sobre el sistema operatiu GNU/Linux, així com de programació de guions bash indispensables per a treballar còmodament amb Linux.
Introducció al programari lliure	Gràcies a la capacitat d'analitzar llicències de programari lliure obtinguts, vaig triar la llicència sobre la que alliberar aquest programari.
Xarxes Obertes	Aquesta assignatura em va fer descobrir les xarxes obertes i és una de les raons per les que vaig triar aquest projecte i no un altre.

Taula 4: Coneixements aplicats apresos al Màster (I)

Assignatura	Coneixement
Desenvolupament de programari	Malgrat abandonar l'assignatura a la meitat, vaig aprendre molt sobre aspectes avançat de la programació en C, com processos i fils, que no solen aparèixer als llibres i he aplicat intensivament al projecte.

Taula 5: Coneixements aplicats apresos al Màster (II)

6.2 Lliçons apreses no relacionades amb l'àmbit acadèmic.

La principal lliçó què he aprés és que el desenvolupament d'aplicacions no és una feina gens fàcil, sovint hi ha molt a fer i molt poc temps per a dedicar-li, a més a més sempre apareixen problemes inesperats que trenquen les planificacions, que per una altra banda no es solen complir.

A més a més, sovint es fa necessari saber llegir codi per la pobresa o manca de documentació, així que també he après que la bona documentació té un valor incalculable, pel temps que ens pot arribar a estalviar.

6.3 La meva visió personal

Finalitze el TFM amb bones sensacions, havent gaudit de la oportunitat, potser única, d'aplicar la majoria dels coneixements adquirits de forma transversal, així com conèixer noves tecnologies o ampliar-ne els coneixements.

El fet d'haver tingut la possibilitat de realitzar un desenvolupament web i un de baix nivell en paral·lel per a acabar fent-los entendre's entre ells m'ha resultat d'allò més interessant.

Només una inconvenient què és alhora un avantatge: fer-ho a distància. Hagués sigut ideal poder-me integrar personalment en l'equip i veure l'ambient que es respira en un entorn laboral real, però probablement si no hagués sigut d'aquesta manera no ho podria haver realitzat.

Això sí, tinc una queixa, quan vaig anar a triar TFM no n'hi havia de sistemes que era la meva preferència i no fer-lo de desenvolupament, però malgrat tot estic satisfet de la decisió i del resultat.

7 Bibliografia

1: , , , https://guifi.net/ComunsXOLN#ComunsXOLN
2: , , , http://fundacio.guifi.net/index.php/Fundaci%C3%B3
3: , , , https://guifi.net/ca
4: Ajax - Getting Started, , 2015, https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started
5: W3Schools.com, , , www.w3schools.com/json/json_http.asp
6: , , , https://openwrt.org/
7: OpenWRT.org, , , http://wiki.openwrt.org/doc/techref/ubus
8: , , , http://wiki.openwrt.org/doc/uci
9: OpenWRT.org, , , wiki.openwrt.org/doc/howto/buildroot.exigence
10: OpenWRT.org, , , http://wiki.openwrt.org/doc/howto/build
11: , Libubus examples, 2015, https://github.com/commodo/ubus/tree/master/examples
12: , libssh2 Examples, , http://www.libssh2.org/examples/
13: Nicholas C. Zakas, , 2009, http://www.nczonline.net/blog/2009/07/21/introduction-to-sessionstorage/
14: W3Schools.com, , ,

8 Apèndixs

8.1 Llicència

Drizzle es distribueix sota llicència GPL3+. A continuació es transcriu el text complet del fitxer COPYING que acompanya la distribució del programari:

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for
software and other kinds of works.

The licenses for most software and other practical works are designed
to take away your freedom to share and change the works. By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users. We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors. You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component

(kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or

similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to

produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program,

in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no

more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or

specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this

License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law. If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it. (Additional permissions may be written to require their own
removal in certain cases when you modify the work.) You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the
terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or
author attributions in that material or in the Appropriate Legal
Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or

additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly
provided under this License. Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after

your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an

organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version,

but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have

actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,

```
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
```

into proprietary programs. If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License. But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.

9 Índexos de taules i il·lustracions

9.1 Índex d'il·lustracions

Índex d'il·lustracions

9.2 Índex de taules