Setting up *relayd* to create a bridge between Ethernet and 802.11 in OCB mode

1 Introduction

OpenWrt allows users to set up, among the possible options, the *relayd* package.

As reported on the OpenWrt website, thanks to this daemon, it is possible to create a bridge between an Ethernet and a wireless interface even when the device is in client mode, including, as we were able to experience, when it is set in `ocb` (Outside Context of BSS) mode and not only in `sta` mode. This enables the creation of a Wi-Fi Extender which is using 802.11p or, in general, any OCB mode wireless connection.

Two pages can be taken as reference, in the OpenWrt documentation:

1. openwrt.org/docs/guide-user/network/wifi/relay_configuration
2. oldwiki.archive.openwrt.org/doc/recipes/relayclient (from the old wiki, but containing very useful information)

Thanks to this package, it is possible to use embedded devices running OpenWrt, such as the *PC Engines* APU1 boards[1], as Wi-Fi dongles for other devices, such as normal laptops connected with the boards through Ethernet.

The following instructions refer to a setup in which two laptops are connected with two APU1D boards, through Ethernet, and they are talking with each other passing though the boards, realizing a 802.11p connection between them; as OpenWrt version, OpenWrt-V2X[2], based on OpenWrt 18.06.1, is used.

[1] `https://www.pcengines.ch/apu1d.htm`

[2] `https://github.com/francescoraves483/OpenWrt-V2X`

These instructions should be easily ported to other hardware platforms and setups too, if OCB mode support is provided.

2 Instructions

The following instructions refers to OpenWrt 18.06.1, but they should work also in newer versions of OpenWrt.

It can be useful, before proceeding, to pinpoint that we successfully tested and consequently described a configuration foreseeing static IP addresses.

All we need to do is to edit few configuration files, after installing *relayd* with:

```
opkg update
opkg install relayd
```

2.1 network

Edit the `/etc/config/network` file.

First of all, setup the Ethernet `lan` interface:

```
config interface 'lan'
#       option type 'bridge'
        option ifname 'eth2'
        option proto 'static'
        option ipaddr '192.168.1.182'
        option netmask '255.255.255.0'
        option gateway '192.168.1.1'
#       option ip6assign '60'
```

As shown in the sample above:

- Comment out or delete the lines related to `option type` and, only if not required, `option ip6assign`. As we will work with *relayd*, there should be no need to configure a "bridge" interface.
- Set up a single interface name after `option ifname` (if there are more than one, otherwise, just skip this point).
- Set up an IP address for the Ethernet interface, including a netmask and a default gateway. All these addresses should belong to the same subnet, which should be the same wired subnet to which the laptop connected to this device belongs. As the IP addresses are static, pay attention not to create any conflict in the network. The default gateway can be omitted if this device won't connect to the Internet through Ethernet.

Then, create a new interface for the wireless station, after the `'lan'` one mentioned before.

```
config interface 'wwan'
        option proto 'static'
        option ipaddr '10.10.6.102'
        option netmask '255.255.0.0'
        option gateway '192.168.1.1'
```

- The interface should be `static`.
- It should then have an IP address belonging to a **different subnet** with respect to the one defined for the wired interface.
- As gateway, two options are possible. If this is a device that can be connected to the Internet through Ethernet, thanks to an external default gateway, use the same address as the one defined before for the `lan` interface, trying to avoid multiple gateway definitions (i.e. set the external default gateway address). If this device will simply forward all the traffic towards another device that can be connected to the Internet through Ethernet, simply omit the `gateway` line.

The last step requires the creation of a relay interface, using the special `relay` protocol:

```
config interface 'stabridge'
        option proto 'relay'
        option network 'lan wwan'
#       option ipaddr '10.10.6.102'
```

- As network, list the interface names which you wish to connect (they should be a wired and wireless interfaces, in our case).
- An IP address (the same as the wireless interface) can be specified, but the bridge seems to work even when no address is specified.

2.2 dhcp

Edit the `/etc/config/dhcp` file.

Disable the internal DHCP server by adding the line `option ignore '1'` for the `lan` interface, like in the following example:

```
config dhcp 'lan'
        option interface 'lan'
        option start '100'
        option limit '150'
```

```
        option leasetime '12h'
        option dhcpv6 'server'
        option ra 'server'
        option ignore '1'
```

2.3 wireless

Edit the `/etc/config/wireless` file.

In the `config wifi-iface` section, after the desired wireless interface (on the APU1D boards only one interface should be present, under the name `'default_radio0'`), replace the `network` option with the newly created `'wwan'` interface, as in the following example:

```
config wifi-iface 'default_radio0'
        option device 'radio0'
        option network 'wwan'
        # other options .......
```

This will attach the wireless to the `'wwan'` interface.

2.4 firewall

Edit the `/etc/config/firewall` file.

You should then edit few settings in the firewall configuration file, in order to let it accept the traffic forwarding. Under the default options, set `forward` to `ACCEPT`, like in the following example:

```
config defaults
        option syn_flood        1
        option input            ACCEPT
        option output           ACCEPT
        option forward          ACCEPT
```

Then, in the `lan` zone, add the newly created `wwan` network, adding this line after "`list network 'lan'`":

```
        list    network         'wwan'
```

Set then `option forward` to `ACCEPT` to enable the traffic forwarding. You should obtain something like the following example:

```
config zone
        option name             lan
        list    network         'lan'
        list    network         'wwan'
        option input            ACCEPT
        option output           ACCEPT
        option forward          ACCEPT
```

2.5 iw_startup

Then, you can setup your OCB mode wireless interface.

You can rely on `iw` and create a custom script to be launched at startup, by launching it inside */etc/rc.local*.

If you have a patched kernel/driver, you can access the 10 MHz-wide channels foreseen by the 802.11p standard. The presented script refers to a patched kernel, enabling OCB 10 MHz-wide channels with *ath9k*-based WNICs, like the one included in OpenWrt-V2X[3].

If you have a patched kernel and a compatible WNIC, you can create a script like this one (you can call it, for instance, `iw_startup`):

```
#!/bin/sh

echo "Waiting 5 seconds..."
sleep 5
echo "Pacthed Italian Frequency Register Set"
iw reg set IT
sleep 1
echo "Turn off wlan0"
ifconfig wlan0 down
sleep 1
echo "Set Mode OCB"
iw dev wlan0 set type ocb
echo "Turn on wlan0"
ifconfig wlan0 up
sleep 1
echo "Leave possibile OCB..."
iw dev wlan0 ocb leave
echo "Set Frequency 5890 MHz (CCH) and channel width 10MHz (802.11p)"
iw dev wlan0 ocb join 5890 10MHz
echo "Set Rate 3M and Power 15 dBm, using iw"
iw dev wlan0 set bitrates legacy-5 6
iw dev wlan0 set txpower fixed 1500
echo "Completing configuration, waiting 8 seconds..."
sleep 8
echo "Configured as: "
iw dev
```

Remember not to set any IP for the wireless interface, as it is already configured inside `/etc/config/network`.

Then, you can launch it at boot by saving it to `/root/` and adding the line:

```
/root/iw_startup
```

[3] `https://github.com/francescoraves483/OpenWrt-V2X`

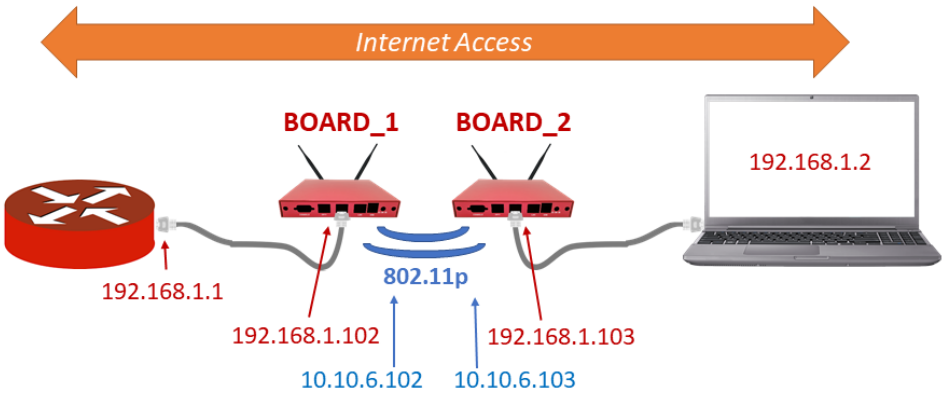

Figure 1: Example setup with Internet access, using *relayd* and 802.11p (OCB mode), as described in Section 2.7

to */etc/rc.local*, before `exit 0`.

Please remember that a patched kernel (or set of drivers), enabling OCB mode, are currently needed.

A script very similar to this one is **already provided** in OpenWrt-V2X, even though removing two lines, setting an IP address, is required in order to properly work with *relayd*:

```
echo "IP address set to 10.10.6.102"
ifconfig wlan0 10.10.6.102 netmask 255.255.0.0
```

2.6 Starting the relayd service

The *relayd* service, at least under certain configurations, seems to be unable to start at boot, even when `/etc/init.d/relayd enable` was previously launched. So, keep in mind that, after each reboot, you will have to launch `service relayd start` to properly start the service and activate the bridge. You can, however, automatize this process to make this service start at boot, by relying on the *rc.local* script, adding the lines:

```
sleep 1
/etc/init.d/relayd start
```

to the `/etc/rc.local` file, before `exit 0`.

2.7 Enabling Internet access

In order to enable Internet access through the bridge, an additional step is required.

If the current device will access Internet though Ethernet, acting as a sort of "gateway", no additional configuration steps are needed.

If the current device, instead, will bridge all the traffic towards the "gateway", mentioned before, to let all the connected wired devices gain Internet access, a specific static route has to be manually added. We tried inserting it inside `/etc/config/network`, but, at the time of writing, it wasn't configured and we could not obtain the desired result, as if the lines inside the configuration file were completely ignored after setting up the *relayd* bridge. Therefore, an ad-hoc command need to be issued, **after completing all the previous steps**:

```
route add -net 0.0.0.0 gw <gw_ip_addr>
```

where `<gw_ip_addr>` should be substituted with the wireless IP address of the *relayd* device acting as "gateway" towards the Internet.

For instance, if one board (let's call it *BOARD_1*) has IP `10.10.6.102` for the `wwan` wireless interface and it has Internet access through Ethernet (i.e. its default gateway setting matches the address of an external default gateway, such as `192.168.1.1`), and a second board *BOARD_2* shall bridge all the Internet traffic to *BOARD_1*, that will then forward it to the external default gateway, the following command must be issued on ***BOARD_2***:

```
route add -net 0.0.0.0 gw 10.10.6.102
```

This procedure can be automatically executed by adding the previously mentioned command to to the `/etc/rc.local` file, before `exit 0` but after all the other *relayd* configuration commands.

To summarize, your `/etc/rc.local` file should contain the following logical commands, in this order:

```
<other non-relayd related commands...>

<launch wireless/OCB mode configuration script>
<perform /etc/init.d/relayd start>
<set the static route if necessary>

<other non-relayd related commands...>

exit 0
```

`sleep` commands may be needed between each command to make everything work properly.

This an example of `/etc/rc.local` from a device that does not have direct Internet access through Ethernet, connecting to a device like the *BOARD_1* mentioned before:

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.

/root/iw_startup
```

```
sleep 1
# Additional command to set txpower
iw dev wlan0 set txpower fixed 900
sleep 1
/etc/init.d/relayd start
sleep 1
route add -net 0.0.0.0 gw 10.10.6.102
exit 0
```

2.8 Final steps

After configuring the system as described before, reboot at least once, with `reboot`, and test the bridge realized through *relayd*.

When two devices are relaying network traffic between each other, using *relayd* and OCB mode, their wired interfaces should never be connected together or to a common switch, as this will create a switching loop, which can easily bring all the network down.

If the devices shall be connected to the same switch (for instance for a different application with respect to relaying traffic), first connect one device, then stop the *relayd* service with `service relayd stop` and finally connect the second device.