Access Point Reconfiguration Using OpenWrt

Dalton M. Tavares[1], Maicon J. Lima[1], Rafael V. Aroca[2],
Glauco A. P. Caurin[3], Antônio Carlos de Oliveira Jr[1],
Tércio A. Santos Filho[1], Stella J. Bachega[4],
Marcos A. Batista[1], and Sérgio F. da Silva[1]
[1]Computer Science Department, Federal University of Goiás, Catalão, Goiás, Brazil
[2]Mechanical Engineering Department / Federal University of São Carlos, São Carlos, São Paulo, Brazil
[3]Mechanical Engineering Department / University of São Paulo, São Carlos, São Paulo, Brazil
[4]Production Engineering Department / Federal University of Goiás, Catalão, Goiás, Brazil

Abstract— *The research project Mobile mEsh Network to Aid in CountEring drug TRAffiCKing (M.E.N.A.C.E-TRACK) proposes the creation of a dynamic mesh network, intended to interconnect field personnel to a base of operations whenever possible. This type of network accepts the dynamic disconnection and reconnection of nodes. To configure a mesh node using, for instance, an access point, usually a modified firmware is needed. In this paper we present the first steps to build the M.E.N.A.C.E-TRACK infrastructure concerning the configuration of the access point, the chosen firmware and some configuration scenarios on an infra-structured network in order to demonstrate its flexibility.*

Keywords: Mesh Network, Ad hoc, OpenWrt

1. Introduction

The research project Mobile mEsh Network to Aid in CountEring drug TRAffiCKing (M.E.N.A.C.E-TRACK) proposes the creation of a dynamic mesh network, intended to interconnect field personnel (e.g. in vehicles or on foot) to a base of operations (e.g. a police station) whenever possible. This type of network accepts the dynamic disconnection and reconnection of a node or group of nodes leaving or returning to the main base network range. The use of day to day low cost and off the shelf devices, like access points, notebooks or smartphones and open source software is one of the main attractors to our approach.

To configure a mesh node using, for instance, a wireless router, usually a modified firmware is needed. This firmware allows the creation of a dynamic route between the base station (the one that have the Internet connection and access to the main systems) and the client nodes in a mesh environment. The dynamic route can be stablished directly between the base station and mesh nodes, or using each mesh node as a "bridge" (in-between nodes) to amplify the range of the original access point. Therefore, each mesh node receives a data connection from a given node and conveys data to the next node, extending the range of communication for each passing node [1].

The steps to build the M.E.N.A.C.E-TRACK infrastructure involve the definition of the devices that can be used as nodes, the choice of the operating system for such devices and the selection of the best routing protocol(s) to provide routing adjustments considering mobile nodes with varying ranges. The chosen devices are off the shelf with the ability to use wireless communication, i.e. IEEE 802.11 b/g/n, and the TCP/IP protocol suite. Specifically, we will cover the adaptation of access points (APs) so they can be used as mesh nodes in the future. This procedure addresses the selection of a firmware, preferably open source, which is flexible enough to be modified according to the choice of the wireless environment to be implemented (e.g. infra-structured or ad hoc network).

2. Infra-structured versus Ad hoc networks

An IEEE 802.11 W-LAN can be implemented either with infrastructure or without infrastructure (i.e. ad hoc). In an infrastructure based network, there is a centralized controller for each cell. This cell represents the fundamental building block of the 802.11 architecture, known as the basic service set (BSS). A BSS typically contains one or more wireless stations and a central base station, known as access point (AP). The wireless stations, which may be either fixed or mobile, and the central base station communicate among themselves using the IEEE 802.11 wireless MAC protocol. Multiple APs may be connected together (e.g. using a wired Ethernet or another wireless channel) to form a distribution system (DS). The DS appears to upper-level protocols (e.g. IP) as a single 802.11 network, in much the same way that a bridged wired 802.3 Ethernet network appears as a single network to the upper-layer protocols. We can use the same analogy considering that an AP, which is normally connected to a wireline backbone (either wired or wireless) is also considered a DS, thus providing Internet access to mobile devices. All traffic goes through the AP, even when this is sent to a destination that belongs to the same cell [2], [3].

In an ad hoc network there is no central control with

connections to the Internet. Here, the network is a temporary arrangement as and when required. The benefit is that no infrastructure is needed and the users themselves may extend the area of coverage. Hence, mobile devices that have found themselves in proximity to each other, that have a need to communicate and find no pre-existing network infrastructure in the location (e.g. a pre-existing 802.11 BSS with an AP) may communicate using each other as the current medium [2], [4].

The focus of this paper will reside in extending the range of an infra-structured network using multiple APs. The main objective to be accomplish with this course of action is to gain the necessary expertise concerning the firmware operation and configuration, which will be invaluable in future experiments involving mesh networks.

2.1 OpenWrt Features

OpenWrt is described as a GNU/Linux distribution for embedded devices, which provides a fully writable filesystem with package management. In that sense, by definition, it is not strictly a firmware but a complete and modifiable operating system. This frees the developer from the static application profile provided by a vendor and allows the customization of the device through the use of packages that suit any particular need [5].

Compared to other distributions OpenWrt may also not be regarded as a true end-user or user friendly firmware. Nevertheless, it can be used as such sometimes, depending on the feature set provided in addition to the main package [5]. Therefore, OpenWrt was chosen as our base architecture to implement M.E.N.A.C.E-TRACK. Section 3 presents a set of experiments devised to study OpenWrt flexibility regarding a simple research question: how can I extend the range of an infra-structured IEEE 802.11 wireless network?

3. Experimental Testbed

We chose as our experimental testbed a TP-LINK TL-WDR 4300 router. According to [6] this router provides a simultaneous dual band (concurrent 2.4 GHz and 5 GHz) and is advertised as 750 Mbps in dual-stream mode on the 2.4 GHz band, and triple-stream on the 5 GHz band. It's hardware can be summarized as an Atheros AR9344 CPU operating at 560MHz, 8MB internal flash, 128 MB RAM, 4 Gigabit Ethernet ports, 1 Gigabit Ethernet WAN port, 2 USB 2.0 ports, Serial and JTag interfaces and support up to 12 VLANs.

To simplify the presentation of the experimental results, we organized this section to discuss the device preparation (section 3.1) and the case studies involving the two APs so they can expand the indoor range of the Wi-Fi connection shared by a desktop computer (section 3.2). The experimental testbed is presented in Figure 1.

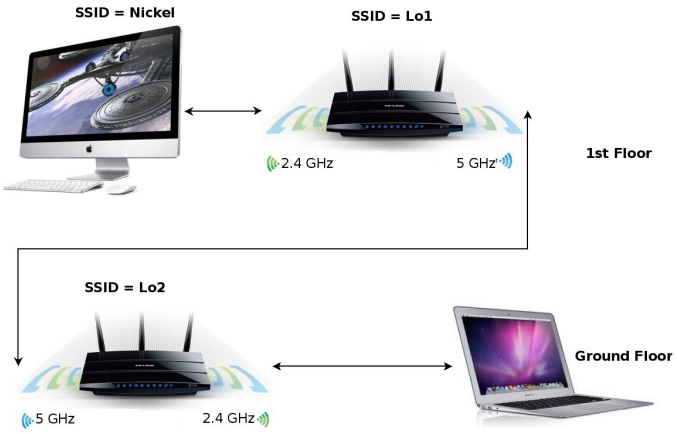

Fig. 1: Proposed testbed for the case study.

3.1 Preparing the AP Devices

The first steps taken to install OpenWrt on the TP-LINK TL-WDR 4300 router are described at [7]. They involve consulting the Buyers' Guide [8], and verifying how much a given router is compatible with OpenWrt using the table of hardware [9].

The first installation of the OpenWrt firmware is done using the original web interface of the TP-LINK firmware. It is performed just like a flash update of some new version of the original TP-LINK firmware. According to [6], it is recommended to turn off the Wi-Fi interfaces manually (there is a switch behind the device to disable them). Therefore, at the first boot, we will not have any wireless interface enabled by default. The first login test is done using an Ethernet cable plugged into any Ethernet port (not the WAN port). The computer interface must be set to any address in the range 192.168.1.0/24, exception made to 192.168.1.1, which is the default address of the OpenWrt interface [10].

After the first connection it is recommended to setup the root password (the only account). After the password is set, the telnet daemon is disabled and all future accesses are done using the ssh service [10]. Considering our testbed is essentially experimental and is intended to test multiple mesh routing protocols, implying in a frequent change in its configuration files, we still did not consider the procedures described at [11] for network and system hardening.

3.1.1 USB Storage

Considering the size of the internal flash memory (8 MB), it is recommended to increase this size as soon as possible. After the base installation, we have up to 4 MB available, which runs out relatively fast depending on the number of packages the administrator chooses to install. Fortunately, OpenWrt allows the extension of this available size by means of an overlay file system. This means we can extend the root file system (stored at the internal flash memory) to an external storage (stored in a USB stick) [12].

This configuration is called pivot overlay on version 12.09 of OpenWrt. To ensure pivot overlay will work, it is recommended some packages are installed for the correct USB support and mounting of the external device (i.e. `kmod-usb-core`, `kmod-usb-ohci`, `kmod-usb-storage`, `kmod-usb-uhci`, `kmod-usb2`, `libusb-1.0`, `usbutils`, `kmod-fs-ext4`, `e2fsprogs`) [13].

Support to pivot overlay is granted by the package `block-mount`. This package allows the mounting of all block devices by just calling the commands `block mount` and `block umount` [14]. To create the pivot overlay on the external USB device, one can either use an empty new rootfs or copy the contents of the current overlay (JFFS2) to the new rootfs. Assuming the filesystem for the new external rootfs is mounted, for example on `/mnt/sda1`, one could issue `tar -C /overlay -cvf - . | tar -C /mnt/sda1 -xf -` [12].

Considering the load of the overlay file system at boot time, it is necessary to create the file `/etc/config/fstab`. This can be done by simply issuing `block detect > /etc/config/fstab` [15]. After enabling the pivot overlay at boot time, we must ensure the external file system is being mounted correctly. Special attention to the target and device options. If after a system reboot a command `mount /mnt` issues no error, than everything is correct. The last remaining step is to change `option target /mnt` to `option target /overlay` and in the next boot, the file system size will be the size of the USB stick plugged in the USB port (minus the space already in use).

3.2 Case Studies

Our experimental case study considers the range extension of a connection shared by a desktop computer (SSID = Nickel) to a notebook. When in an indoor environment in the first floor of a building, we found it would not go further than approximately 10 m. The next step was to establish a simple connection to the two APs as client/server and the connection of a notebook on their respective SSIDs (i.e. Nickel ↔ Lo1 ↔ Lo2 – the notebook can access either of the SSIDs). As each antennae has its own frequency (2.4 GHz and 5 GHz) we had to match them as illustrated on Figure 1. In OpenWrt, this configuration is done at `/etc/config/wireless` (Listing 1).

Listing 1: Excerpt from `/etc/config/wireless`.

```
...
config wifi-device  radio0
        option hwmode    11ng ←
...
#radio0 is connected to the Desktop
config wifi-iface
        option device   radio0
        option mode     sta
        option ssid     'Nickel'
        option encryption psk2
        option key       password
        option network  wwan
...
config wifi-device  radio1
        option hwmode   11na ←
...
config wifi-iface
        option device   radio1
        option network  lan
        option mode     ap
        option ssid     Lo1
        option encryption psk2
        option key       password
```

Listing 1 shows that `radio0` operates in the 2.4 GHz band (`hwmode 11ng`) while `radio1` operates in the 5 GHz band (`hwmode 11na`). `radio0` is in mode `sta`, meaning it is configured to be a client of `Nickel`. It also shows that `Nickel` uses encryption (WPA2) and is related to a network called `wwan`. The configuration for `radio1` is analogous, but it is operating in AP mode (`option mode ap`), with `SSID Lo1` and is related to network `lan`. The definitions for networks `wwan` and `lan` are showed on Listing 2.

Listing 2: Excerpt from `/etc/config/network`.

```
...
config interface 'lan'
        option ifname 'eth0.1'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.0.1'
        option netmask '255.255.255.0'
...
config interface 'wwan'
        option ifname 'wlan0'
        option proto 'dhcp'

config interface 'stabridge'
        option 'proto'      'relay'
        option 'network'    'lan wwan'

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

Listing 2 shows the settings for network `lan` (Ethernet ports) and `wwan` (wireless – `radio0`). Network `lan` is configured in bridge mode and its static IP address is 192.168.0.1, which ensures access to the AP via Ethernet cable (or wireless via `radio1`). Any computer connected via Ethernet cable receives an IP address via DHCP, in the range 192.168.0.100/192.168.0.150 (Listing 3). The `wwan` DHCP config ignores the `lan` pool and gets addresses outside this pool.

A bridge is defined between `lan` and `wwan`. Therefore, any computer connected to the wireless network (on `radio1`) or via Ethernet cable (via `lan`) will receive an

IP address automatically. Instead of a true "bridge", in this configuration, the traffic forwarded is affected by firewall rules, such that the `wwan` network and the `lan` network should be configured according to the same "zone" created by a firewall with the "forward" policy adjusted to "accept", so that all the traffic flows between both interfaces [19].

The insertion of firewall rules inside `/etc/config/network` is not recommended, but it was implemented as such for the sake of simplicity in this first test. According to the official documentation [20], this configuration should not work. It is explicitly mentioned that "...STA and AP at the same time is not yet supported...". Fortunately, this was true for trunk version up to r22989. We are using trunk version r40555.

Listing 3: Excerpt from `/etc/config/dhcp`.

```
...
config dhcp 'lan'
        option interface 'lan'
        option start '100'
        option limit '150'
        option leasetime '12h'
...
config dhcp 'wwan'
        option interface 'wwan'
        option ignore '1'
...
```

This configuration demonstrates how we can connect each AP in series to Nickel. The configuration of the second access point (Lo2) is analogous. Attention to the frequency of the interfaces. In Lo2, we must connect `radio1` (`hwmode 11na - 5 GHz` in `mode sta`) to `radio 1` in Lo1 (`hwmode 11na - 5 GHz` in `mode ap`) and `radio0` (`hwmode 11ng` - 2.4 GHz in `mode ap`) must be available to the clients (e.g. notebooks or other APs).

The objective is to set each AP in different floors in a building (i.e. first floor and ground floor) in order to extend the range of Nickel. To enable Internet access from either AP, it is necessary to set a packet forwarding service on both APs so that each could route traffic appropriately. We will also discuss a Wireless Distribution System (WDS) implementation in order to extend the original SSID beyond its original range. The specifics of each approach will be further discussed in sections 3.2.1 and 3.2.2.

3.2.1 IP Forwarding

According to [21], netfilter is used for packet filtering, NAT and mangling. This firewall is configured by means of a proper syntax called Unified Configuration Interface (UCI) [22]. This "language" is intended to centralize the configuration of OpenWrt. The problem with UCI is the fact one needs to learn a new syntax for something that is already well known (i.e. `iptables` syntax). Although UCI is intended to simplify the configuration of OpenWrt, depending on the complexity of the intended scenario, the use of UCI can lead to some confusion. For instance, in the beginning of this section, we saw how to configure bridging according to OpenWrt documentation [19]. The configuration uses the firewall to establish the packet transport from the `lan` to `wwan` as discussed before. Therefore, this can be assessed as an application of the UCI firewall. This section will perform some minor modifications on the discussion regarding the connection of both APs and display a configuration profile easier to follow using the `iptables` scripting allowed on OpenWrt.

In order to obtain Internet access from Lo1 and Lo2, the easiest way is to perform source network address translation (SNAT). SNAT translates an outgoing packet, which may come from Lo1 or Lo2 clients, so that each intermediary OpenWrt system looks like the source. Considering the intent to use Lo1 and Lo2 as extensions of Nickel, a SNAT rule must be created on both. In Lo1 each outgoing packet will have its source IP changed to the Lo1 outgoing IP address (interface in `mode sta`). From Lo2, the outgoing packets coming from its clients will suffer SNAT so that they seem to come directly from Lo2. Then, in Lo1, these packets will suffer another SNAT so the clients connected to Lo2 can access the Internet. Therefore, a client must be able to access the Internet connecting directly through Nickel, through Lo1 (going further in the same floor) or through Lo2 (covering the ground floor). The modified configuration for `/etc/config/wireless` and `/etc/config/network` is presented on Listings 4 and 5.

The main differences from the version presented before are the removal of the firewall rules originally inserted to establish the bridge rules and the creation of two distinct interfaces for each radio (`wlan0` and `wlan1`) (on `/etc/config/network` – see Listing 2). We renamed interface `wwan` to make it simple to correlate the "virtual" interface names (`interface` parameter) to the "real" interface names (`ifname` parameter). Therefore, `wwan` will be renamed to `wlan0` and we will create a new interface called `wlan1` which will be in `mode 'ap'`. We also modified accordingly file `/etc/config/dhcp`. There are no modifications to file `/etc/config/wireless`.

Listing 4: Excerpt from modified `/etc/config/network`.

```
...
config interface 'wlan1'
        option ifname 'wlan1'
        option proto 'dhcp'

config interface 'wlan0'
        option ifname 'wlan0'
        option proto 'dhcp'
...
```

Listing 5: Excerpt from modified `/etc/config/dhcp`.

```
...
config dhcp 'wlan0'
        option interface 'wlan0'
```

```
        option ignore '1'

config dhcp 'wlan1'
        option interface 'wlan1'
        option ignore '1'
...
```

This scenario seems simple enough, however we have some operational problems. First, considering the UCI syntax, it is not clear how to specify a rule that considers any source IP to be translated to a given destination IP. Second, considering Lo1 and Lo2, we have to detect the destination IP address every time the packet is translated. The outgoing interface is being configured via DHCP so, the address is very likely to change in each boot (or every 12 h as described in Listing 3).

The best way to address this issues was to use the UCI firewall interface to `iptables` commands. Therefore, we achieved a simplified configuration model that is fit for the purposes of our case study. We created a user script that is processed by the UCI firewall, after the firewall rules are loaded. This script is stored in `/etc/firewall.user` (Listing 6).

Listing 6: Excerpt from `/etc/firewall.user`.

```
...
WANIF="wlan0"
WANIP="`/sbin/ifconfig $WANIF | grep
    'inet addr' | awk '{print $2}' |
    sed -e 's/.*://'`"
iptables -t nat -I POSTROUTING 1 -o $WANIF
    -j SNAT --to $WANIP
```

The main part of the user script of Listing 6 is the deduction of the IP address of the outgoing interface. We create a filter using `grep` and `awk` in oder to extract from the `ifconfig` command output exactly the IP address of the outgoing interface (defined by the user in the `WANIF` parameter). This is used as the input for the SNAT rule (in the `iptables` rule). Therefore, for any IP address coming from the `WANIF` interface we have the translation to the new outgoing `WANIP` address.

One last quirk of OpenWrt is that the processing of the `firewall.user` is not deterministic in our system. It sometimes worked (i.e. the `firewall.user` file is processed) and it sometimes didn't. Therefore, as a workaround, we used the init script `/etc/rc.local` (Listing 7). The commands appearing inside this file are executed once the system init is finished.

Listing 7: Excerpt from `/etc/rc.local`.

```
# Workaround to load the SNAT firewall rule
# correctly
sleep 10
/etc/init.d/firewall stop # clean up the system
                          # firewall rules
sleep 10
/bin/sh /etc/firewall.user # insert SNAT firewall
                           # rule
```

3.2.2 Wireless Distribution System

According to [23], WDS is a misunderstood concept. WDS is usually referred to as a "wireless DS" or a "DS" that operates over a WLAN. A WDS (as defined by [24]) is neither. This confusion perhaps results from an extremely poor choice in naming the WDS capability. The "WDS" capability actually has nothing to do with either of those terms.

Still according to [23], WDS is a mechanism for constructing 802.11 frames using a 4-address format. The content of the data frame address fields are dependent upon the values of To DS and From DS bits and is defined in Table 1. If the content of a field is shown as not applicable (N/A), the field is omitted. Note that Addr. 1 always holds the receiver address (RA) of the intended receiver and Address 2 always holds the address of the station that is transmitting the frame (TA). Addr. 3 and 4 refers to the usual destination address (DA) and source address (SA).

Table 1: WDS 4-address format [23].

To DS	From DS	Addr. 1	Addr. 2	Addr. 3	Addr. 4
1	1	RA	TA	DA	SA

OpenWrt implements WDS between a client AP and a master AP using the 4-address format, which enables transparent bridging on the client side. In this scenario, a bridged host (e.g. computer A) sends a packet to a target host (e.g. computer B). The frame is relayed via the client AP (i.e. Lo2) and the sender MAC (i.e. computer A) is preserved. The master AP (i.e. Lo1) receives the frame and redirects it to the target (i.e. computer B) using the original sender source MAC (computer A). The target (computer B) receives the frame and generates a response, using the given source MAC (computer A) as destination. The master AP relays the frame to the client AP with the right destination MAC as target (computer A). The client AP receives the frame and redirects it to the final destination using the computer A MAC as target. Computer A receives the response frame and the connection is established [25].

The aforementioned scenario was inspected using a network sniffer (Wireshark) on both computer A and Computer B. We used `ping` to send an ICMP Echo Request from computer A and inspected the received packet on computer B. It was possible to verify that the MAC address was really from computer A instead of the master AP where the traffic is relayed. We also had a Wireshark running on computer A and we also inspected the return message (ICMP Echo Reply) in computer B. The source MAC address from the packet was the one from computer A and not from the relay station (client AP).

To configure WDS, on the master AP (Lo1), we need to add the line `option wds '1'` on the `config wifi-iface` section for `radio 1` (the one configured

in mode 'ap'), at /etc/config/wireless [26] (see Listing 1). That's all there is to it. If a client AP connects to this master AP, the WDS interface is created as wlan1.sta1. This can be verified using the ifconfig command.

The client AP (Lo2) configuration is a little bit more complicated. First, all the firewall configurations described on section 3.2.1 must be removed. WDS approach is a level 2 bridge, not a level 3 translation via netfilter. For all purposes, all the firewall rules can be disabled on the client AP. Second, at /etc/config/network, interface 'lan' and interface 'wlan0' are bridges. Care must be taken considering the configuration of the static IP addresses in both cases. Both bridges must not be in the same IP address range (see Listing 8). We must also observe that the bridge configuration for interface 'lan' is not done inside /etc/config/network. This will be done afterwards manually using the brctl command (Listing 10) [27]. This is done because UCI, in our test case, for some reason, did not allow the configuration of two bridges simultaneously.

Listing 8: Excerpt from /etc/config/network.

```
...
config interface 'lan'
        option ifname 'eth0.1'
        option proto 'static'
# Quirk: invalid IP address to avoid conflict with
# br-wlan0 address
        option ipaddr '192.168.1.199' ←
        option netmask '255.255.255.0'
        option ip6assign '60'

config interface 'wlan0'
        option ifname 'wlan0'
        option type 'bridge' ←
        option proto 'static'
        option ipaddr '192.168.0.200' ←
        option netmask '255.255.255.0'
...
```

The br-wlan0 bridge is a wireless to wireless bridge (radio 1 ↔ radio 1 at 5 GHz). Its definition becomes clear in the specification presented at /etc/config/wireless, for the radio 1 (Listing 9). One particular quirk that must be observed is the use of both Wi-Fi interfaces (radio 0 and radio 1). radio 0 is configured as the local AP for Lo2, but its SSID is identical to the SSID used by Lo1 (including the password). radio 1 is configured as a client (mode 'sta') of the Lo1. This configuration is necessary to assure a seamless integration between Lo1 and Lo2. That way, for example, a client that is moving across the boundaries of Lo1 and Lo2 cells will not realize the transition from one AP to the other. One particular feature in this case study is that the APs Lo1 and Lo2 incidentally operate in different frequencies (Lo1 uses 5GHz for its clients and Lo2 uses 2.4. GHz). Therefore, the client must also have the ability to operate in both frequencies.

Listing 9: Excerpt from /etc/config/wireless.

```
...
config wifi-iface
        option device 'radio0'
        option ssid 'Lo1'  ←
        option encryption 'psk2'
        option key 'password' ←
        option network 'wlan0'
        option mode 'ap' ←
...
config wifi-iface
        option device 'radio1'
        option mode 'sta' ←
        option ssid 'Lo1' ←
        option encryption 'psk2'
        option key 'password'
        option network 'wlan0' ←
# This line is important to establish the WDS
        option wds '1'  ←
...
```

The next step is to add manually a second network interface to the established bridge, in this case, the br-wlan0. As this procedure must be done on every boot of Lo2, we recommend putting it in /etc/rc.local. Another problem is that even though all the clients of Lo2 correctly access the Internet, Lo2 itself cannot do it. This is important mainly to keep the ability to install new packages from Lo2. The problem is its routing table needs a default gateway, therefore we use Lo1 address and create a static route [28]. We put all this in /etc/rc.local, so the complete procedure is executed for every boot (Listing 10).

Listing 10: Excerpt from /etc/rc.local.

```
...
# ### Quirks to load static route and local DNS
# (192.168.0.1)
rm /etc/resolv.conf
ln -sf /etc/config/resolv.conf.auto /etc/resolv.
    conf
# ### Create the static route for the default
# gateway
sleep 5
route add default gw 192.168.0.1 br-wlan0 ←
# ### Add Ethernet interfaces to the wifi bridge
# br-wlan0
sleep 5 # Important to give time for the WDS
        # connection establishment
# ### Add eth0.1 to the br-wlan0 bridge using
    brctl
brctl addif br-wlan0 eth0.1 ←
exit 0
...
```

4. Final Remarks

The proposed research question for this paper regards which methods could be used to extend the range of an infra-structured wireless network. The concealed objective was to find an open firmware, flexible enough so it could be used in the context of project Mobile mEsh Network to Aid in CountEring drug TRAffiCKing (M.E.N.A.C.E-TRACK). In that sense, this paper succeeded considering the scenarios tested on OpenWrt provided a rich environment for

interaction and customization of the firmware. Therefore, we tested the configuration of APs used to extend the original reach of a wireless network using a level 3 solution (i.e. a firewall) and a level 2 solution (bridging via WDS).

The main advantage of using packet forwarding on the APs is the solution is hardware independent. We can use packet forwarding in virtually any device (considering it has at least up to the network layer of TCP/IP protocol stack). The main drawback is transparency. The user must choose explicitly the SSID of the network he/she is using. This limits the kind of application one can implement considering the need for a seamless integration of APs in order to provide a unified SSID for a bigger range (e.g. in the transit of a mobile robot which exchanges information with an external computer in the same Wi-Fi network).

Considering transparency of integration among devices, WDS is the best choice between the test cases. It provides the possibility for using multiple AP devices without the need to choose between SSIDs. There is only one SSID and the integration is seamless. The drawbacks, on the other hand, can be more extreme when we compare this solution to packet forwarding. WDS is not a certified IEEE standard and, therefore, every vendor (e.g. Ralink, Atheros, Broadcom etc) can have its own implementation, resulting in incompatibility among devices [29].

One design problem we managed to solve with our configuration concerns the loss of bandwidth in WDS Repeater mode. According to [29], WDS Repeater mode will sacrifice half of the bandwidth available from the primary router for clients wirelessly connected to the repeater. This is a result of the repeater taking turns talking to not just one partner but two, and having to relay the traffic between them. As the AP used for the WDS use case has two independent antennae, this will not occur (i.e. it does not have to take turns between communicating pairs).

Considering the coverage area in both cases, we achieved roughly 40 m when compared to the original 10 m of the desktop computer sharing its Internet connection. When using the packet forwarding implementation, we had to exchange SSIDs on the 1st floor (Lo1) and ground floor (Lo2) explicitly. When using WDS, we could roam both floors without connection loss using only Lo1.

As a future work, we will devise the case study for implementing a mesh network using TP-LINK TL-WDR 4300. Therefore, OpenWrt will be the basis to configure part of our mobile ad-hoc network (MANET), which will be implemented in the context of the M.E.N.A.C.E-TRACK system.

Acknowledgment

The authors would like to thank the Research Assistance Foundation of the State of Goiás (FAPEG – Fundação de Amparo a Pesquisa do Estado de Goiás) for the full sponsorship of this research – edict number 006/2012.

References

[1] D. Johnson, K. Matthee, D. Sokoya, L. Mboweni, A. Makan, and H. Kotze, *Building a Rural Wireless Mesh Network A do-it-yourself guide to planning and building a Freifunk based mesh network, version 0.8*, Meraka Institute, South Africa, 2007.

[2] J. F. Kurose and K. W. Ross, *Computer Networking - A Top-Down Approach*, 6th ed. Pearson Education, 2012.

[3] G. Aggelou, *Wireless Mesh Networking*. McGraw-Hill Communications, 2009.

[4] S. Methley, *Essentials of Wireless Mesh Networking*, ser. Cambridge Wireless Essentials Series. Cambridge University Press, 2009.

[5] OpenWrt, "About openwrt." [Online]. Available: http://wiki.openwrt.org/about/start

[6] ——, "Tp-link tl-wdr4300." [Online]. Available: http://wiki.openwrt.org/toh/tp-link/tl-wdr4300

[7] ——, "Beginners' guide to openwrt." [Online]. Available: http://wiki.openwrt.org/doc/howto/user.beginner

[8] ——, "Buyers' guide." [Online]. Available: http://wiki.openwrt.org/toh/buyerguide

[9] ——, "Table of hardware." [Online]. Available: http://wiki.openwrt.org/toh/start

[10] ——, "Openwrt – first login." [Online]. Available: http://wiki.openwrt.org/doc/howto/firstlogin

[11] ——, "Secure your router's access." [Online]. Available: http://wiki.openwrt.org/doc/howto/secure.access

[12] ——, "Rootfs on external storage (extroot)." [Online]. Available: http://wiki.openwrt.org/doc/howto/extroot

[13] ——, "Usb basic support." [Online]. Available: http://wiki.openwrt.org/doc/howto/usb.essentials

[14] ——, "Mounting block devices." [Online]. Available: http://wiki.openwrt.org/doc/techref/block_mount

[15] ——, "Fstab configuration." [Online]. Available: http://wiki.openwrt.org/doc/uci/fstab

[16] subsignal.org, "Luci," 2014. [Online]. Available: http://luci.subsignal.org/trac

[17] OpenWrt, "Luci essentials." [Online]. Available: http://wiki.openwrt.org/doc/howto/luci.essentials

[18] ——, "Openwrt sysupgrade." [Online]. Available: http://wiki.openwrt.org/doc/howto/generic.sysupgrade

[19] ——, "Network configuration." [Online]. Available: http://wiki.openwrt.org/doc/uci/network

[20] ——, "Wireless configuration." [Online]. Available: http://wiki.openwrt.org/doc/uci/wireless

[21] ——, "Firewall configuration." [Online]. Available: http://wiki.openwrt.org/doc/uci/firewall

[22] ——, "The uci system." [Online]. Available: http://wiki.openwrt.org/doc/uci

[23] D. Engwer, "Ieee p802.11 wireless lans - "wds" clarifications," Nortel, July 2005. [Online]. Available: http://www.ieee802.org/1/files/public/802_architecture_group/802-11/4-address-format.doc

[24] IEEE, "Ieee standard for information technology - telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements- part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," ANSI/IEEE," Standard, 2003.

[25] OpenWrt, "Client mode wireless (solution using wds)." [Online]. Available: http://wiki.openwrt.org/doc/howto/clientmode#solution.using.wds

[26] ——, "Atheros and mac80211 wds to implement a wireless network bridge." [Online]. Available: http://wiki.openwrt.org/doc/recipes/atheroswds

[27] "Building bridges with linux." [Online]. Available: http://bwachter.lart.info/linux/bridges.html

[28] N. Craft, "Linux setup default gateway with route command," August 2006. [Online]. Available: http://www.cyberciti.biz/faq/linux-setup-default-gateway-with-route-command/

[29] dd-wrt.com, "Wds linked router network," July 2012. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/WDS_Linked_router_network