OpenPattern project: a comprehensive modular routing platform

Florian Fainelli[1], Xavier Carcelle[2], Etienne Flesch[3], Gwenaël Saint-Genest[4], Hossam Afifi[5]

[1] Florian Fainelli, Institut National des Télécommunications (INT), Paris, France, florian.fainelli@telecomint.eu

[2] Xavier Carcelle, Open Pattern Foundation, xavier.carcelle@openpattern.org

[3] Etienne Flesch, Open Pattern Foundation, etienne.flesch@openpattern.org

[4] Gwenaël Saint-Genest, Open Pattern Foundation, gwenael.saint-genest@openpattern.org

[5] Hossam Afifi, Institut National des Télécommunications (INT), Paris, France, hossam.afifi@int-edu.eu

Abstract. The last developments of the home gateways, DSL-boxes, off-the-shelf wireless routers, wireless communities initiatives have brought a great number of network hardware to the home. These different developments have been followed by different initiatives of "hacking the box" to be able to flash this specific hardware with open-source firmware resulting from the contribution of several open-source developers (OpenWrt, Freifunk...). Due to the lack of "open hardware" off-the-shelf available to adjust the platform to the home-networking (and networking) applications, OpenPattern is a project to develop such a hardware based on the inputs from the past initiatives and bringing a box based on an open hardware (i.e. open specifications of the targeted hardware), using a motherboard and able to receive daughterboard to expand the functionalities in conjunction with the current open source project to be using the box once this one is ready to be shipped.

Keywords: Wireless, Home Networking, open hardware, electronic, boards, routing, open source, network interfaces.

1. Introduction

This paper introduces the OpenPattern modular routing platform (OMRP). We will explain what current problems this platform ad-dresses, as well as its innovative characteristics to explain the architectural choices we have made accordingly. Finally we will explain the set of modules and various interfaces that will be shipped within the base board.

OpenPattern is a modular routing platform, composed of a powerful single board computer, on top of which you plug network interface modules. The software running on this board is fully open source from the kernel and user-space applications on the CPU to the bit stream of the FPGA. You also benefit from the mailing-lists and code repository to contribute and talk with people interested in the project.

The SBC supports a wide variety of network interfaces such as Wi-Fi 802.11n, WiMAX, Bluetooth 2.0, ZiBee and WiBree, as well as ADSL2+ and PLC. The modularity of the motherboard will allow any new network interfaces to be adapted to become a daughter-board for the OMRP. The first chapter will address a state of art of the past and current concept of "free electronic", the second part will explain the objectives of the OpenPattern project and finally the third part will go deeper into the targeted architecture of the OMRP (hardware and software).

2. The context of "free electronic" and state of the art

The concept of "Free electronic" is vast and had been in the past a hot topic within the open source (software mainly) community worldwide. This concept is trying to extend the notion of open source of code (along with the right licensing of the work done such as the GPL license) to the development of hardware platform but this extension is harder than simply opening a work done on a software project to the public by the fact that hardware means third and fourth parties such as manufacturers and OEMs. Having said that the concept of "free electronic" is more and more important nowadays to allow people to be able to develop their own applications (with open source) based on open specifications of the hardware platform. In this part we will describe the current situation of such project and the relations between major open source projects and their implementations on "to-soon-open" hardware platform.

2.1 Open and closed electronic components

A hardware platform such as a cell-phone, a routing board or a home gateway is composed of an electronic board and several key components along with discreet components to fine-tune the impedance, current and voltage levels needed for the components to remain in their manufacturer specifications.

In the past years, several open source communities have developed such projects targeting several applications where the hardware had been "closed" for several months along with a huge development of radio communications and home networking applications. Table I gives a list of several "open hardware" projects that aim to open up the platform, the drivers and the applications source code.

TABLE I
DIFFERENT "OPEN HARDWARE" PROJECTS

Project	Field of interest
OpenMoko	Cellphones (closed GSM stack)
OpenWrt	Wireless Routers
Gecko3	Fully open FPGA
GnuRadio	Radio communications
GSM Cracking project	Open GSM receivers
OpenGraphics	Graphic adapters
OpenSPARC	SPARC arhictecture
Atheros Driver / Linux kernel	Atheros Wi-Fi chipsets
Leox	System on Chip core and busses

One of the most known project nowadays is *OpenMoko* aiming (with the support of certain hardware manufacturers), at developing an "open cell phone" with several radio interfaces (GSM, Wi-Fi…) allowing the end-users to develop their own applications on top of the hardware. The GSM stack is not yet open as the GSM Telcos are not prone to open up the GSM stack for now.

2.2 Home gateways boards

With the huge development of triple and quadruple play offers from the worldwide ISPs (IP-browsing, VoIP, IPTV, IP-phone), end-users are now using on a daily basis home gateways embedded in proprietary hardware from the different manufacturers. For instance, French ISPs are now delivering so-called *Internetbox* (Livebox, Freebox, 9box …) which are hardware platforms able to implement Linux for routing, VLC client [1], video decoding, web-GUI, SIP-client, filesharing server. This home gateway boards may be in a near future generic hardware with open source applications and proprietary software stack for the identification on the ISP network for secure connexions. Moreover, end-users are now holding others embedded boxes for their home-networking applications such as wireless LAN, PLC (Power Line Communications) interfaces or file sharing based on proprietary hardware that may be open too in a close future.

2.3 Routing boards and platforms

From a software point of view, several open source projects have been active nowadays based on the famous Linksys WRT serie (WRT54G and others) router board trying to find the most appropriate implementation of open source firmware. These implementations have been using the hardware specifications that Linksys had been giving to the public.

Among these firmwares, OpenWrt is one of most widespread and had been used even natively in certain commercial applications available.Here is a list of certain boards running OpenWrt natively:

- Neuf/Cegetel EasyGate
- Mindspeed Comcerto
- Meraki Mini
- Fon La Fonera
- Ubiquity LiteStations
- Infineon Amazon

This list shows that open source firmwares have now reached a level of quality (along with the right implementations based on the targeted hardware) enough for wide commercial applications.

2.4 Wireless communities boards

Another big area of development of open hardware and software have been the different wireless communities around the world implementing different firmwares based on the IETF and IEEE standards around the Wi-Fi (802.11flavors, IP-mesh standards such as OLSR). Table II is giving a non-exhaustive list of the different boards and firmwares used by these communities.

TABLE II
WIRELESS COMMUNITIES BOARDS

Community	Boards	Firmware
Freifunk	Meshcube / WRT54G	Nylon / OpenWrt
France-Wireless	WRT54G / Fonera	OpenWrt / OpenWrt
Meraki	Meraki	OpenWrt
Fon	La Fonera	OpenWrt
Seattle-Wireless	Various	Original firmwares
TIER Project (UC Berkeley)	Gateworks Avila 2348	Custom linux

For instance, the Freifunk wireless community in Berlin and Leipzig (Germany) are using two different hardwares (Meshcube and Linksys WRT) to implement very large ad-hoc networks. These networks are based on 802.11 MAC layer along with an OLSR-based IP-layer embedded in the OpenWrt firmware. This implementation allows anyone in the world aiming at launching a new wireless community to do the same by getting some Linksys boards and flashing the hardware with the Freifunk flavor. So far several Freifunk wireless implementation had been done and the Berlin network is now reaching several hundreds OLSR nodes (up-to 5 hops from the wired IP feed) to allow shared wireless Internet access. The TIER project from UC Berkeley is also aiming at developing wireless network in developing countries by using standard hardware with a custom Linux distribution to implement long distance wireless based on a modified 802.11 MAC layer with TDMA medium access for a better QoS (Quality of Service).

3. Project motivations and objectives

3.1 Is Free as in free software electronics possible?

Nowadays, the core of electronics and microelectronics is what is called an Intellectual Property block (IP-block). This block is nothing more than a piece of code, written in a Hardware Description Language (HDL), which describes common electronics subsystems: processors, memory controllers, Ethernet controllers and more as the layout that a bunch of programmable transistors should have. This small piece of code is part of a library, in which electronic component designers pick their items to design a subsystem and then put this together to finally produce an electronic component.

Ten years ago, some electronic engineers realized that every time they developed a new dedicated electronic component such as processor or a microcontroller, they were reinventing the wheel, even for the smallest part of the system like doing a logical and between two bits. The goal of the *OpenCores* project was to offer to electronic designers a common place to share, improve and discuss Intellectual Property blocks, and reduce the time to test and market. Though this project has been hosting a wide range of IP-block of a good quality, there is another problem which is the software used in simulating, testing and designing electronic components.

Most FPGA vendors also supply a development tool, which allows electronic designers to place analyze, change their IP-blocks, to design the whole electronic component. This tool is also in charge of doing the routing and placement of the different IP blocks on the FPGA, and finally produces the "bitstream" and netlist that will be sent to the FPGA to make it be the component freshly designed. These tools are not released under a Free Software compatible license, but can are available freely online. There is an ongoing effort to release open source tools that will be able to generate a bitstream. The good point of these tools is that they can run on a GNU/Linux system and therefore you can use them in combination of proprietary software running on open source software.

Since we cannot have free software to develop freely available IP-blocks, can we really do *Free Electronic* as in Free Software? Well, the answer is yes we can, as long as we keep the whole process open in terms of tools being used, documents and schematics produced as well as the project management and the capability for people to produce their own board based on that design. The project remains free because everything that has been produced or discussed is freely available under licenses that will protect the materials, but not prevent people from contributing. We can see this as the logical extension of free software concepts being applied to software programmable components including the "make it yourself" part of it that hardly exists with hardware developments.

By offering schematics, source code and tips to produce their own Printed Circuit Boards (PCB), people will really be able to produce their own cards and implement their own applications on-top of it.

3.2 Who would use this design?

Since the design is completely open, quite a lot of people can be interested in using it, especially:

- academic researchers
- industrial researchers
- consumer electrics companies
- wireless communities
- core routers for emerging countries

Academic and industrial researchers really need to have an open design to work on because otherwise they will spend their time in getting information they can possibly never have from the manufacturers or distributors because they prefer to keep it secret for commercial reasons. Since designing hardware from scratch is risky and time-consuming, we provide a working hardware platform so researchers can focus on developing code or making the existing project evolve instead of dealing with a new design every time they want to test something.

The *OpenPattern* platform is also ideal for anyone needing a board that can host a wide range of network interfaces, would it be for rapid deployment, development or testing. Since we have been designing and validating the most complex parts of the system, such as the memory and the DMA controller, people can use the design to even validate their own hardware.

Quite a lot of people in the world are looking for ready-to- use devices they can deploy in a meshed network, which is what devices like the Linksys WRT54G currently do. More and more countries will use cheap wireless equipments to build mesh networks for their cities, and therefore the *OpenPattern* board can be ideal because it can act as gateway for multiple network interfaces, mainly Wi-Fi and WiMAX. As it was described, getting decent open source support, for a given hardware, takes from one month to several years after the hardware public relase depending on the Board Support Package that was shipped with.

3.3 Where is the innovation?

The *OpenPattern* project can be seen as innovative in several manners including: project management, consumer electrics design process, hardware and software features. Regarding the problems explained before, it was clear that a modular platform was needed, open and upgradeable hardware to help on-going efforts to focus their energy on developing or designing software instead of hardware.

This project aims to have contributions from the beginning unlike some open source related projects that were first developed and then open up to the public. The *OpenPattern* project aims at sharing the information to allow the maximum of contributions such as the design of new modules, new applications and better drivers for the open hardware. Finally it seems important that end-users can be able to use the PCB to build their own boards or even add modifications and add-ons to the original PCB.

4. Hardware and software architecture

The *OpenPattern* architecture has been designed to provide an effective software and hardware architecture. Since modularity is important and the lifetime of this product chosen is a combination of dedicated hardware providing good performances, and modular hardware to host future modules.

From a software point of view, it had been providing *paravirtualisation* for this embedded system with the same usage as desktop or server *paravirtualisation* software.
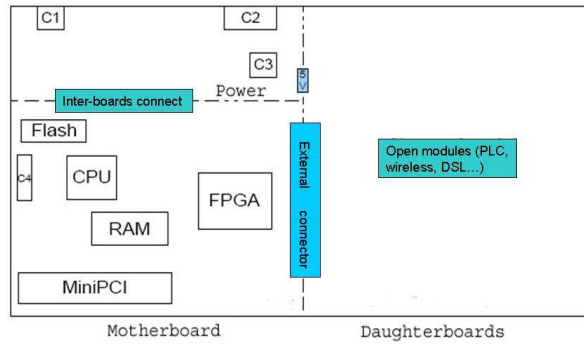
4.1 Hardware architecture

Embedded devices are designed around a central and dedicated component (ASIC) which is called a System-on-chip (SoC) in which functional subsystems such as a CPU, memory controller and Ethernet controller was put in with a high rate of integration and cost reduction. The main problem with this approach is that there is a dedicated SoC every time a new device is marketed, cheap with a high level of integration. Also, the SoC takes place in a specific design method called, co-design, where both the system features and software programming is done to reduce the time to market. This approach fits very well to a mass-market production and selling but is unaffordable for custom designs.

This kind of System-on-Chip is widely used to reduce the cost and physical space required by the equivalent and standalone chips. For instance, most hardware manufacturers integrate a CPU, memory controller, Ethernet controller and Wi-Fi MAC processor inside the SoC for a occupied space of less than 5 square centimeter. The SoC manufacturing cost can be reduce up to few dollars for a million pieces.

Taking that into account, the design is done by thinking of a chip that is capable of evolving simply by making software upgrades on a programmable flash, and which could be powerful enough to even replace a SoC in later design revisions. Such functionality is achieved by a Field Programmable Gate Array (FPGA) while storing its bitstream code is done in a programmable Flash.

FIGURE 1
HARDWARE ORGANIZATION OF THE OMRP

The figure I is showing the hardware organization targeted on the board with the motherboard on one side and the place for the modules (daughterboards) on the other side. Table II shows the targeted hardware component to be used for the OMRP.

TABLE III
OMRP HARDWARE FEATURES

CPU	400 Mhz ARM
FPGA	Xilinx Spartan-3 family
CPU Flash	16 Mo NAND
RAM	128 Mo SDRAM

Unlike SoC that does everything needed, the FPGA allows the designer to choose a SoC with less features (cheaper and smaller), and do the rest in the software running on the FPGA. Combining this architecture design with a good placement and interconnection with the network interfaces will lead to a fully modular design.

These arguments gave the reason to place the FPGA as an intermediate component for the microcontroller to access hardware such as Wireless and Ethernet interfaces. The FPGA placement and feature can be compared to what is done on the PC architecture with the so-called "chipsets" which are basically allowing the
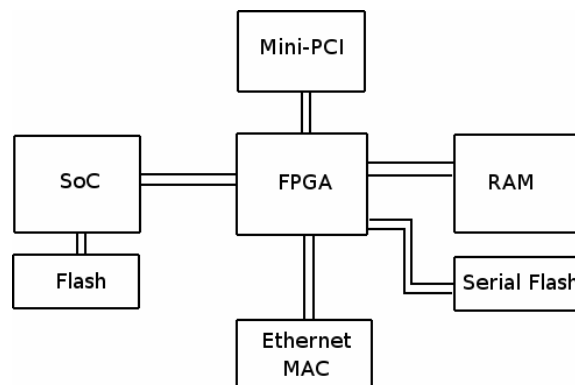
CPU to access hardware resources without the need to have specific analog/digital front-end circuitry. The advantages of this architecture are very clear:

- The FPGA is more tolerant to electric surges and temporary designs than the CPU
- The CPU is not loaded with the hardware interruption handling the FPGA does this
- There is a clear separation between the hardware accesses and the software, allowing virtualization techniques
- The FPGA bitstream can be easily changed to handle different network interfaces
- Reduction of the front-end components doing hardware interfacing.

Since the SoC has fewer features, it gets smaller and cheaper, as well as the FPGA, because there is no need for either highly integrated and/or powerful components to handle the same functionalities. The cost of the two components makes less than either a powerful FPGA or a full-featured SoC.

The figure II shows the organization of the components on the OMRP and the data-bus connections between the two core components (SoC and FPGA), the different network interfaces and the memory chips.

FIGURE II
ORGANIZATION OF THE COMPONENTS

Because of its central role in the hardware interfacing with the CPU, the FPGA implements the following IP-blocks:

- RAM memory controller for the CPU
- memory segmentation and/or paging
- Direct Memory Access controller
- Layer-2 and/or Layer-3 networking
- hypervisor for paravirtualization

To fully benefit from modularity, it has been left intentionally unpopulated pins so that one can either plug a custom daughterboard or a networking module for instance.

Since the design had been thought to be modular, the hardware will use several connectors that can host a wide range of network interfaces. The default layout features are:

- 2 x Ethernet MAC, 1x Ethernet PHY
- 2 x Mini-PCI connectors
- 2 x USB host 2.0 connectors
- SPI and GPIO connectors
- 1 x UTOPIA2 connector

One Ethernet MAC controller is not connected to any PHY controller because one can choose to plug a Power Line Communication PHY module. This PLC module can be designed using either a HomePlug-based chip or a Opera-based chip, or any other standard. Mini-PCI is the standard for connecting Wi-Fi and WiMAX cards when not built in a System-on-Chip. Having two MiniPCI slots allows one to setup the board as a WiMAX/Wi-Fi router for instance. Finally, one can plug a xDSL module on the UTOPIA2 high-speed digital bus to be used by any ISP along with the proprietary radius stack.

It has been clear that using a CPU coupled to a FPGA will improve the overall performance of the system because the CPU will not be loaded with all the interrupt handling routines while scheduling for over tasks. This design is very near to what has been done recently by some hardware manufacturers while marketing "Network processors.
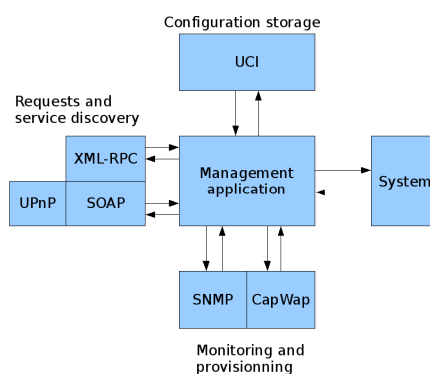
It can be found two fairly closed designs to the OMRP design. The first one is the *Realtek* RTL8651B design [1], which integrates the *Realtek* RE856x network

processor, capable of doing up to Layer-4 networking (NAT, IPsec, VPN...) and hardware cryptography. This solution is fully open source but lacks decent support in the Linux kernel. Also, the network chip is an ASIC, which is partly software controlled, thus making it hard-to-change or hard-to-optimize for different task. Another design having similar capabilities is the *Ikanos Fusiv* DSL platform [2], which includes a routing processor to which the Linux kernel directly passes its routing table. This approach is very interesting because the chip handles up to Layer-3 networking based on a formatted routing table. The main problem with this platform is that it is currently only available to integrators and therefore only under Non Disclosure Agreements. Both lack decent Open Source support and were designed for two particular targets: SOHO routers for *Realtek*, and DSL CPE for *Ikanos*, which does not make them suitable for evolving networking applications.

4.2 Software architecture

Nowadays, most research and industrial funding on embedded system target virtualization features, comparable to what is done for servers and desktops. Virtualization has been for years, the key to delivering fast, flexible and secure to deploy systems in most internet hosting companies and service providers. Virtualization intends to separate an operating system from the hardware or another operating system by putting it in a software jail and by defining traps mechanism to access the hardware when required. This task is done by a computer program which is responsible for securing every operating system instance and hardware accesses.

FIGURE 1II
SOFTWARE ORGANIZATION OF THE OMRP

The figure III indicates an example of targeted software architecture for the OMRP for network applications.

Since the operating system is responsible for isolating and providing mechanism to programs to access the hardware, it runs in a privileged space so-called kernel space, while programs run in a less privileged space, the user space. This isolation is very important because otherwise, either we should be running only one application, then quickly becoming a kernel by itself, or programs should have mutual exclusions and protection mechanism, which makes them non standard and compatible. All processor architectures include at least two protection mechanisms (privileged and unprivileged modes) to allow such memory and resources separation. Some architecture like the Intel x86, allows more levels of protection, which are called rings, up to four.

4.3 Virtualization

Virtualization techniques use this hardware and operating system architecture to provide memory boundaries and isolation between either user-space programs or different instances of the kernel. The main problem is that different techniques require more or less modification of the existing operating system to be virtualized, and sometimes with a lot of performance cost.
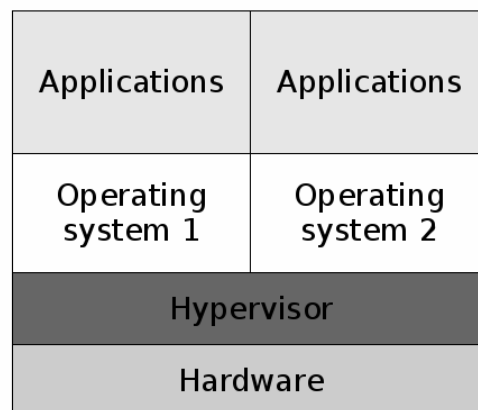
One can make the distinction between **5 kinds of virtualization techniques**. The first one is what we call an **isolator**, which is a piece of software interfacing an existing and unmodified operating system with a user-space application (Linux VServer [3], OpenVZ [4]). The second one is **user-space kernel**, which an operating system running in user-space, then running programs in user-space. This solution is one of the least effective solutions, but is very convenient for kernel development (User-mode Linux [5], Cooperative Linux [6]). The third one is a **virtual machine** running on top of an existing kernel and providing hardware resources and access. The main advantage of this solution is that the guest operating system, running in the virtual machine "sees" the hardware like if it was the real operating system, but with a huge performance cost (VMWare [7], Qemu [8], and Parallels [9]). The fourth requires the use and design of a **paravirtualizer or hypervisor**, which interfaces the hardware with a domain 0 kernel. Most of the time, the domain 0 kernel must be modified to run on-top of the operating system. The domain 0 kernel then can start guest OS on top of it, but they will interact with the hypervisor so that the domain 0 kernel is only here for control. This solution is a good compromise between low-level development which is architecture specific and performance. Recent works now allow hypervisor to use the hardware virtualization instructions provided by some processors like *Xen* [10] or the jails in FreeBSD [11]. The last one is **hardware virtualization**, which more or less uses the same design as the paravirtualization with a hypervisor, except that the hypervisor is a dedicated component or even part of the processor. This solution is

widely used in the business class servers such as AIX or Sun servers, and more recently for desktops with *Intel Vanderpool* [11] and *AMD Pacifica* [12] technologies.

Virtualization, as most industrial actors want to use it for embedded targets, should bring a secure way to make online payments handled by an application running on its own operating system copy, while other applications such as messaging, internet browsing would run in a separate operating system copy. The main drawback with this approach is that it will only bring virtualization features to a mass-marketed business, exclusively for security reasons while other features are available. Most of the works on embedded targets focus on using paravirtualization instead of hardware virtualization because it allows software upgrades to gain virtualization whereas a hardware solution would require a lot of changes.

Most embedded systems, especially in networking environnement run with no or few management and software upgrades, because of a high availability needed. Virtualization can help in reducing the downtime while upgrading the software. Since the hypervisor will hardly be changed, it can easily start another operating instance, because there is no need to reinitialize all the hardware and go to any recovery procedure or such. This is also very useful because you might want to test development versions of your operating system, while keeping a production instance of it running. It becomes convenient to decide when to switch from a production release to another with a minimum downtime as allowed by the hypervisor. By promoting *Free Electronic*, *OpenPattern* promotes free software running on open designs by using standard and carrier-class operating systems such as Linux or *BSD. The software stack that will be running on the OpenPattern design is shown above. The figure IV show an example of software layers in the case of virtualization on top of the OMRP.

FIGURE 1V

SOFTWARE LAYERS OF THE OMRP

Applications	Applications
Operating system 1	Operating system 2
Hypervisor	
Hardware	

The great advantage of the FPGA handling RAM accesses, is that the FPGA can provide a hardware separation of the memory for the processor, and even doing context switching for it when it needs to run either one instance of the operating system of the other. Starting an architecture and design from scratch allows this project to benefit from the development done on the *Xen* hypervisor and on the recent version of jails in the FreeBSD environment, while preventing too much software work by having hardware mechanisms. This also allows unmodified kernels to run on the *OpenPattern* design, while porting the *Xen* hypervisor to the ARM architecture would require a lot of changes [13].

4.4 Managing the board

Most network and system administrators, whether they work for large scale or community networks face the same problem with an increasing rate of heterogeneous equipments being part of the system to manage. Nowadays, most network equipment manufacturers will ship their custom software solution based on the SNMP protocol [14]. These systems can manage other manufacturer's networks because of the SNMP design which uses a Management Information Base. You just need to provide a per-equipment MIB [15] and a SNMP program to read and set values for the system.

This solution has been accepted as *de-facto* standard for managing network equipments and server, brought together on a central administration console. Another standard that is being worked by the IEEE is the CapWap [16], which is mainly targeted to managing wireless access points. Though a lot of work has been done to get it mature, there seems to be few implementations of it, except OpenCapWap [17. Using CapWap for wireless home gateways makes a lot of sense, CapWap can be a great software implementation on top of the OMRP.

5. Conclusion and perspectives

OpenPattern aims not to be "yet another open source project" but to gather hardware and software people around an open platform able to be duplicate, modified, improved. Hardware is now a key point for the home gateways as well as for the wireless communities (in developed and developing countries as it has been described earlier) and giving it fully open will allow people to choose between designing their own hardware with the public PCBs, getting a standard OMRP or buying a proprietary hardware from an OEMs. Certainly the latest success of the *OpenMoko* project and the achievement of the "*One Laptop-Per-Child*" (OLPC) initiative had show that opening the hardware is now possible. Ideally the OMRP can be seen as a cheap, upgradeable and open routing board for wireless networks in developing along with open hardware for the end-users such as *OpenMoko* phones or OLPC computers making a telecommunications infrastructure fully open from the PCB of the board to the software implemented in the core of the network to the user interface.

References

[1] http://www.videolan.org/vlc/

[2] http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=11&Level=4&Conn=3&ProdID=70

[3] http://www.ikanos.com/solutions/security/index.cfm

[4] http://linux-vserver.org/Welcome_to_Linux-VServer.org

[5] http://openvz.org/

[6] http://user-mode-linux.sourceforge.net/

[7] http://www.colinux.org/

[8] http://www.vmware.com/fr/

[9] http://fabrice.bellard.free.fr/qemu/

[10] http://www.parallels.com/

[11] http://www.xensource.com/Pages/default.aspx

[12] "Jails in FreeBSD", Isaac Levy, EuroBSD Conference September 2007, Copenhagen, Denmark.

[13] http://www.intel.com/technology/platform-technology/virtualization/index.htm

[14] http://enterprise.amd.com/us-en/AMD-Business/Business-Solutions/Consolidation/Virtualization.aspx

[15] http://www.cs.uiuc.edu/class/sp06/cs523/lectures/26/XenARM-Final.ppt

[16] http://tools.ietf.org/html/rfc1157

[17] http://tools.ietf.org/html/rfc1156

18

[18] http://www.ietf.org/html.charters/capwap-charter.html

[19] http://opencapwap.org/