Wibed, A Platform for Commodity Wireless Testbeds

Pau Escrich, Roger Baig
Fundació Privada per la Xarxa Lliure,
Oberta i Neural guifi.net
Mas l'Esperança, 08503 Gurb, Catalonia
{pau.escrich, roger.baig}@guifi.net

Axel Neumann, Alexandre Fonseca,
Felix Freitag, Leandro Navarro
Universitat Politècnica de Catalunya
Barcelona, Spain
{axel, afonseca, felix, leandro}@ac.upc.edu

Abstract—**Testbeds are a stage between the simulation and the production stages. To this end they must be as close as possible to production environments (i.e. real hardware, on the field deployments) while also keeping the traits of experimentation facilities (i.e. fault tolerance, ease of deployment, testing and data collection).**

This paper presents Wibed, a platform for facilitating the quick and cost-efficient acquisition, deployment, and management of testbeds based on commodity IEEE802.11 routers and enabling experimentation with wireless technology including the modification of low-level system components such as physical and link layer mechanisms, and network and transport layer protocols. A key functionality of the Wibed system lies in the automatic configuration and operation of a meshed testbed-management network designed for coping also with unstable and dynamically changing wireless links and topologies. This way, in contrast to the typical experimentation approach of identifying and reproducing relevant characteristics of a target environment in an existing and stationary testbed, our approach will allow to deploy a testbed infrastructure inside the target environment within only few hours and at a very low cost.

As a proof of concept, the Wibed platform has been used to deploy the UPC CN-A testbed with currently 50 nodes over six campus buildings and which is being federated with the Community-lab testbed.

Index Terms—**COTS IEEE802.11 routers; quick deployment, wireless testbed; mesh networks; community networks.**

I. Introduction

Following the current standard research facilities (e.g. PlanetLab[1]), the experimentation nodes in the Community-lab testbed, developed by the Community Networks Testbed for the Future Internet (CONFINE)[2] project, integrate virtualisation techniques[1] to allow running experiments in parallel, a user-friendly deployment of experiments and management environment and a robust recovery system.

Nonetheless, the adoption of virtualisation has the drawbacks of higher node costs due to increased requirements on computing resources such as CPU and memory. The need to ensure proper isolation of experimentation resources also implies restricting the access to the lower layers of the operating system and communication stack. While a cost increase entails a reduction of the number of experimentation nodes being deployed, restricted access to the low layers restrains the range of supported experiments. Wibed, the testbed platform presented in this paper, has been envisaged as a complement to the Community-Lab testbed facility to cope with these two problems at the cost of foregoing the virtualisation support.

The Wibed architecture has been conceived to keep the hardware restrictions as open as possible: the capability of running a GNU/Linux system and having at least one (see Subsection V-F) ath9k[2] supported Wireless Network Interface Cards (WNICs) are the minimum conditions set by design. Currently these conditions are broadly fulfilled by many of the Commercial off-the-shelf (COTS) dual-radio routers available in the retail market for less than 100€, allowing the acquisition of the whole testbed hardware, including tenths of nodes, for a few thousand Euro. Thanks to the self-configuration system and wireless-mesh based management network, the node deployment time is significantly reduced. Wibed has been designed to admit from link-layer to application-layer experiments.

At the time of this writing, the platform is being developed and, in parallel, a testbed of 50 nodes is being deployed over six buildings of Universitat Politècnica de Catalunya (UPC) Campus Nord, Barcelona. The resulting testbed will be made available to the researchers as part of the CONFINE Community-lab[3][3][4] facilities. The software and documentation[4] are publicly available and maintained as part of the CONFINE project.

The remainder of this paper is organised as follows. Section II discusses the movements and opportunities, given by recently available commodity wireless routers and open system developments allowing the operation of such hardware, and that inspired the design of the Wibed platform. Section III reviews the related work. Section IV presents the platform design. Section V discusses the platform implementation. Section VI analyses the usage of the platform to set up a testbed at UPC. Section VII discusses the current development status of the platform, the work pending, the replicability of the solution presented and costs entailed. Finally Section VIII presents the conclusions and the lessons learnt.

[1]Using Linux Containers. http://lxc.sourceforge.net/.

[2]http://wireless.kernel.org/en/users/Drivers/ath9k

[3]http://community-lab.net/

[4]http://git.confine-project.eu/wibed, http://wiki.confine-project.eu/wibed:

II. Background

A. Community-lab

Community-lab is a testbed being built under the CONFINE project and inspired by PlanetLab. It is, therefore, designed to be Slice Federation Architecture (SFA) and cOntrol, Management and Measurement Framework (OMF) compatible, having federation in the agenda. PlanetLab concepts have been ported and adapted to the community networks environment and are referred to with the same terms. Thanks to the interconnection of the three community networks involved in CONFINE all nodes can reach each one another using the Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures (FEDERICA) overlay network. There are indoor and outdoor nodes. Indoor nodes consist of computers based on low-power standard PC technology (Intel Atom) meant to run application level tests. Outdoor nodes consist of embedded nodes with wireless equipment that offers access to the wireless link-layer to a certain degree allowing lower level experiments. Conceptually, the access is limited to prevent any experiment interfering with any another running in parallel, but in practice currently it is also limited as a result of some kernel contextualisation limitations. All research devices are linked to a community device via an Ethernet connection. As of July 2013 Community-lab has around 50 operational nodes distributed among three community networks.

B. COTS IEEE802.11 routers

The router market is offering routers made from COTS components. These COTS routers (also know as commodity routers, Customer-premises equipments (CPEs), home gateways, home routers, etc.) are fast, inexpensive, and equipped with bleeding-edge technology. Compared to routers offered by the industry's largest manufacturers, these routers cost a fraction of that price. Nonetheless, the strategy of offering the latest innovations while keeping the prices as low as possible has its downsides such as a limited number of production series, extremely short release intervals between new models and a high component variability even between very similar models. Nowadays, several vendors are offering in the market various device models sporting dual radios with IEEE802.11n support, 8MB of flash and 64MB of RAM memory, fast CPUs, USB sockets, all below the 100€ level.

C. OpenWrt

OpenWrt[5] is a Linux distribution that has become the defacto standard[6] for embedded devices thanks to its accurate design, frequent updates and small size. OpenWrt provides a complete Buildroot to easily generate both a cross-compilation toolchain and a firmware (root filesystem plus a kernel) for the target architectures. The toolchain consists of gcc as the compiler, binutils as the assembler and linker, and μClibc as the C standard library. The base system of the firmware consists of the Linux kernel as the operating system kernel and the following root filesystem components: BusyBox, mac80211, opkg[7] and Unified Configuration Interface (UCI)[8]. OpenWrt also provides a feeds system, to integrate additional packages (over 3.500 available by default) into the firmware[9].

D. Wireless Battle Mesh

The Wireless Battle Mesh (WBM), also known as the Battlemesh[10], is a totally horizontal event organised and driven by the participants (the authors of this paper included) who get together to test dynamic routing protocols on temporary testbeds deployed by themselves to that end. In the sixth edition (Aalborg, Denmark, April 2013) the experiment deployment system was redesigned from scratch. The resulting design partially set the precedents of the work presented here. Both the code and the data sets of the that edition are publicly available[11].

III. Related work

Indoor testbeds like the 49 nodes grid wireless testbed used in [5] or the 400 nodes ORBIT testbed [6] represent one type of testbed where experimentats are executed over physical links but under laboratory conditions. The latter also offers the possibility of introducing artificial noise as an additional means to replicate environmental characteristics.

Several outdoor testbeds have been set up with a focus on wireless network experimentation in particular environments.

QuRiNet [7] is an outdoor IEEE802.11-based wireless network deployed in a National Park in the US with currently 41 nodes. Different to testbeds deployed in urban or laboratory environments, QuRiNet operates in an environment free of wireless interference and electromagnetic noise.

The DES-testbed [8], [9] developed by the Freie Universität Berlin, is presented as a campus testbed of 95 nodes for research on wireless and sensor networks. It offers a framework for carrying out protocol experiments.

NITOS [10], [11] is another wireless experimentation infrastructure. It is created and operated by CERTH and NITLab and deployed in university-campus areas with a focus on evaluation of protocols and applications in real-world settings. Federation with other global-scale testbeds like PlanetLab is foreseen. Similar to Wibed, the testbed nodes are based on commercial Wi-Fi cards and Linux-based open-source platforms.

The BOWL testbed [12] is another campus testbed with around 50 outdoor nodes. BOWL reports that by means of reconfigurable software architecture, the testbed allows to perform routing protocol testing, e.g. switch between different routing protocols, such as OLSR and DSR. The wireless

[5]https://openwrt.org/

[6]Over 200 specific devices are currently considered supported.

[7]Opkg: a package management system: http://code.google.com/p/opkg/

[8]UCI: a configuration command-line interface: http://wiki.openwrt.org/doc/techref/uci

[9]For instance, Lua Unified Configuration Interface (LuCI) (http://luci.subsignal.org/), the OpenWrt standard de-facto web configuration interface is provided as an external feed.

[10]http://battlemesh.org/

[11]The code is available as at https://github.com/battlemesh and the results at http://downloads.battlemesh.org/WBMv6/test_data/.

network testbed is also a production network used by student and therefore offers the conditions of live traffic for research experiments.

In summary (and to the best ouf our knowledge) the following differences between existing experimentation facilities and the Wibed approach presented in this work can be summarized:

- Existing tesbed facilities are made for long-term stationary deployments while the Wibed platform is designed for also allowing quick and temporary deployments in target experimentation evironments.
- Compared to our deployment, which is part of CONFINE community-lab testbed, most other existing testbeds[12] are stand-alone systems that are not foreseen to be federated with larger (global-scale) testbeds.
- Also, through the integration of Wibed testbeds into the management infrastructure of the CONFINE system, access to Wibed deployments will be made available for external researchers, an opportunity which is yet only clearly defined and offered by the ORBIT and NITOS testbed.
- Finally, the fact that the platform is made available at no cost as free software together with the low cost of the hardware supported facilitates the replication and the customisation of Wibed-like testbeds.

IV. Platform design

Testbeds based on the Wibed platform are composed by a set of COTS routers, the *testbed nodes*, forming mesh networks with access to an external *testbed server*, as depicted in Fig. 1. The testbed management system follows a server-client model with the *testbed controller* (the server software) being the only means of external interaction with the whole testbed (thus, neither sysadmins nor researchers should ever log in to the nodes). The nodes receive *orders* from the controller (e.g. ”install a new experiment”) in a pull-based manner, by periodically sending it a *request*. The orders are embedded in the *replies* to the node's requests. Controller orders are node-specific, making it possible, for instance, to stop the experiment execution on a single specific node.

Experiments are filesystem overlays which are attached to the nodes firmware during the experiment execution. Nodes can run only a single experiment at a time but different nodes can run different experiments in parallel. The management system also allows the execution of commands in the nodes.

Aside from the experiment deployment tools, Wibed includes a centralised storage system to ease data collection from experiments.

In order to relax deployment restrictions, the testbed management and related nodes-controller communication is established over a wireless mesh network, operating independent of the wireless experimentation network. Although it is recommended to include more wired nodes in order to increase the testbed resilience, this approach allows to significantly reduce the costly and time-consuming task of deploying wired network connection in the target experimentation zone. Therefore, if experiments demand low-level WNICs access, at least two WNICss are needed to fully isolate the management from the experimentation network.

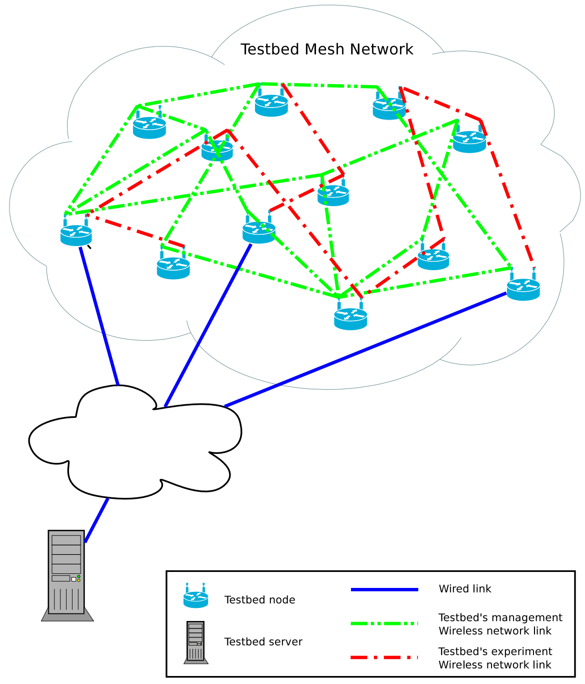
Testbed Mesh Network
Testbed node
Testbed server
Wired link
Testbed's management Wireless network link
Testbed's experiment Wireless network link

Fig. 1. System architecture.

V. Platform implementation

A. Nodes-controller communication

As already mentioned, the nodes are responsible for periodically pulling the server for new orders by sending requests. Requests are also made in every transition (loop-back transitions included). Requests contain the node status. Some statuses entail additional information such as the standard and the error output in the case of commands executed and mechanisms to prevent resending data[13].

The controller responds to each request with a reply containing an order. Orders are detailed in Table I. The *experiment* and *upgrade* orders contain *objects*. The experiment object comprises: `Experiment ID`, `Action ID (0 FINISH, 1 PREPARE, 2 RUN)`, `Overlay URL`, `Overlay HASH`; the upgrade object: `Firmware ID`, `UNIX time to upgrade the node`, `Firmware URL`, `Firmware HASH`.

All messages exchanged are in JSON[14] standard text format.

B. Nodes filesystem architecture

Following the OpenWRT approach, the testbed nodes filesystem is composed by two parts: a SquashFS read-only LZMA compressed ROM containing the basic operating

[12]The recent federation approaches of PlanetLab and NITOS present a pioneering exception here

[13]For specific details see https://wiki.confine-project.eu/wibed:unified-api.
[14]http://www.json.org/

TABLE I
ORDERS

Order	Content
experiment	Object describing experiment details
upgrade	Object describing firmware upgrade details
commands	List of pairs (cmdID, cmdStr)
resultAck	cmdID of the last result received from the node
	Empty order, nothing to do

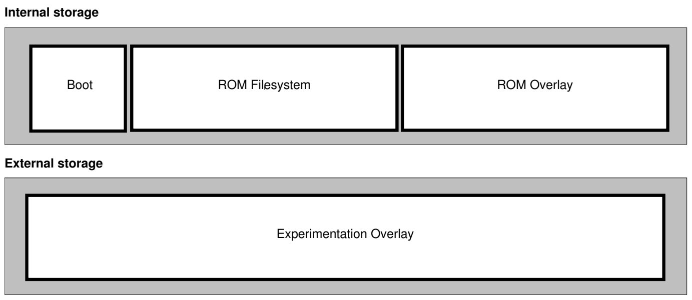

Fig. 2. Nodes firmware filesystem architecture.

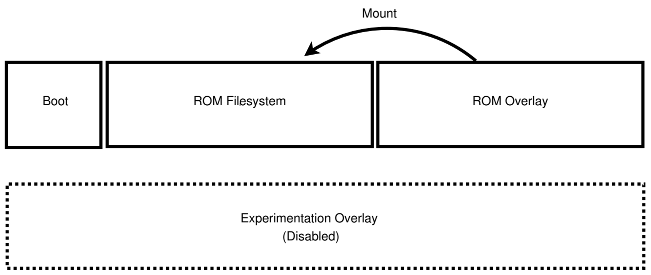

Fig. 3. Node firmware filesystem in the IDLE status.

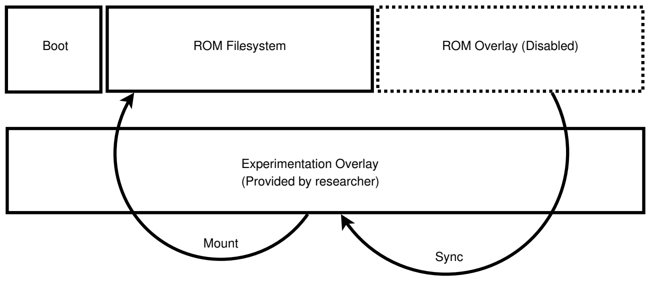

Fig. 4. Node firmware filesystem in the RUNNING status.

system (kernel, a minimal root filesystem and the testbed management software) and a JFFS2 mounted as OverlayFS[15] over the read-only partition to store filesystem changes.

The standard boot process of OpenWRT is as follows[16]:

1) The kernel boots from ROM and runs */etc/preinit*
2) */etc/preinit* runs */sbin/mount_root*
3) */sbin/mount_root* mounts the RW partition and combines it with the RO partition (*/rom*) to create a new virtual root filesystem
4) The bootup continues with */sbin/init*

As depicted in Fig. 2, Wibed extends this approach by adding a third component: a second overlay aimed at allocating the experiment placed in an external storage device[17] (such as a USB stick) . The *experiment overlay* is only mounted during the execution of an experiment. As Fig. 3 and Fig. 4 show, the process to launch an experiment is as follows:

1) Copy the filesystem provided by the researcher to the experimentation overlay.
2) Synchronise the files from the standard overlay to the experimentation one.
3) Configure system to boot with the experimentation overlay.
4) Reboot.

C. Overview of the node firmware

To coordinate all the processes running in the node, a local UCI database is used. It is placed on */etc/config/wibed* and contains some static predefined fields such as WiFi channel, BSSID or node hardware model, a set runtime dynamic fields such as status, last command executed or current experiment identifier[18].

A single generic firmware image is used for installation in all testbed nodes. During the first boot, each node configures itself writing the changes in the ROM overlay. Firstly, it generates a CRC16-HASH based on the first network interface MAC address. This hash will be used to identify the node. Secondly, it configures the management network according to the parameters specified in predefined UCI sections. The IP assignment is organised as follows:

- Static private IPv4 192.168.1.1 for rescue purposes.
- Static private IPv4 10.X.R1.R2/16 being X (8bits) predefined and shared by all testbed nodes, and R1 and R2 (8bits each) from CRC16-HASH.
- Static ULA IPv6 fdba:X:R1R2::1/64.
- Dynamic DHCP IPv4 request to get automatic gateway network configuration.

Finally, the node starts the pulling process from the controller, announcing current status 0 (INIT) and immediately changing to status 1 (IDLE) upon receiving a valid response from the controller (ensuring that the controller correctly registered the node).

D. Nodes management system

The node states are detailed in Table II. Fig. 5 shows the node finite-state machine with transitions resulting from a controller order tagged. The remaining transitions are the result of node's local operations. INIT-IDLE transition triggers a special request including the device model and the firmware version. ERROR-IDLE and ERROR-INIT are formally an internal transition but can only be triggered externally via the

[15]Included in Linux Kernel mainline 3.11
[16]For further details see: http://wiki.openwrt.org/doc/techref/filesystems
[17]The additional storage device overcomes the common space limitation of the internal storage in current COTS routers.
[18]https://wiki.confine-project.eu/wibed:config

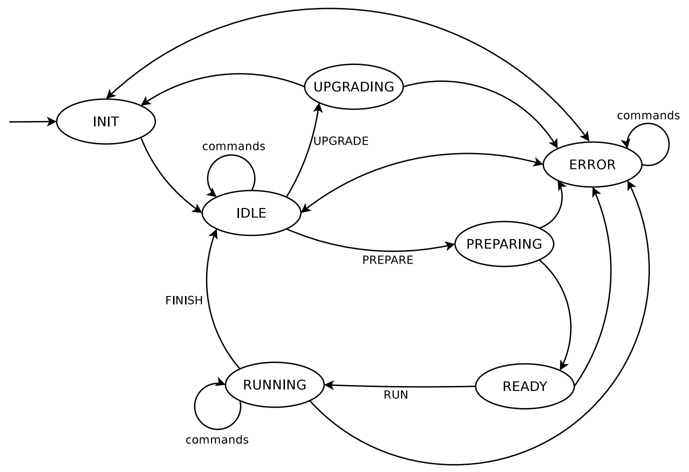

Fig. 5. Node finite-state machine.

TABLE II
NODE STATES

stateID	Name	Meaning
0	INIT	Booting
1	IDLE	Idle (waiting for action)
2	PREPARING	Downloading overlay
3	READY	Overlay ready to be installed
4	RUNNING	Experiment running
5	UPGRADING	Upgrading firmware
6	ERROR	Error detected

execution of a command. All transitions to INIT and to IDLE imply the unmount of the experiment overlay. RUNNING is the only state where the experiment overlay is mounted. It must be noted that UPGRADING is the only state in which node-server communication is expected to be lost. In any other state the interruption of the communications leads to the ERROR state directly or after a given number subsequent attempts.

E. Controller architecture

The Wibed controller works as a standard web server (implemented in Python with the Flask framework) providing an API endpoint (`api/wibednode/<nodeId>`) to which the nodes of the testbed send their periodical requests. The controller parses these requests and stores information about the nodes on a local database, sending only relevant information and/or commands in response to a node request. This is an attempt to minimise bandwidth usage.

The registration of new nodes on the controller is made in a totally ad-hoc manner. When the controller receives an API request containing a new node id, it will consider that request as coming from a new node and will add it to the management database. Nodes are initially installed in the INIT state or they revert back to that state after an upgrade. As described in Subsection V-D, when nodes are in the INIT state they send their device model and firmware version along with the request. This allows the controller to always know updated hardware details of each node in this ad-hoc management operation. This knowledge is important, among other things, for ensuring the delivery of compatible firmware to testbed nodes or allowing researchers to choose nodes with similar hardware and up-to-date firmware.

Currently, the servicing of experiment and firmware images is done via a simple HTTP response to a GET request (to `static/overlays/<overlayId>` or `static/firmwares/<firmwareId>`). However, the system is being designed in a flexible manner to allow future experimentation with other delivery mechanisms such as Bittorrent or wireless-mesh-optimized P2P technologies.

A front-end is also provided for researchers and administrators where the state of the nodes can be checked, experiments started/finished, commands issued and experiment or firmware images uploaded. An API is planned for these interactions so as to allow future integration with common management interfaces in the CONFINE project.

The testbed controller will also be running an NTP server with which the nodes can synchronise their local clocks so as to keep them loosely in sync. This is important for coordinating the order of firmware upgrades (which should be done from the outside to the inside of the mesh due to possible incompatibility between versions) and for log analysis after an experiment.

F. Management network

The management network is used to reach the research devices remotely. Wired connectivity between research nodes is not a requirement for all Wibed nodes but for at least one (identified as *border node*). Consequently, to ensure the controller-node communication, the management network has to be built using WiFi 802.11 standards. Thus the primary WNICs radio is used to create a AD-HOC (IBSS) network between all deployed nodes.

The routing protocol BATMAN-ADV[19] handles the layer 2 routing by encapsulating Ethernet over Ethernet. The border nodes make the interconnection between the WiFi testbed and the controller. As a result of this configuration, from a networking point of view, the controller and the nodes are in the same collision domain. This facilitates the management and administration of the nodes, since standard auto-configuration and node access techniques via IPv6 link-local addresses are possible.

A proper operation of the nodes and experiments is essential for the usability of the testbed. Therefore, to clearly identify and handle the cases of correct and abnormal experiment execution while coping with potential instabilities of the wireless and multi-hop management network, the system must combine robustness against temporary connectivity-failures with restrictive checks and recovery procedures. Only in case of long-term disconnection and unrecoverable failure, the node automatically returns to the initial state (even if there is an experiment running).

[19]http://open-mesh.org

G. Data storage

To save the experiment results, a special directory placed on */data* is provided. Its content is constantly synchronised with the controller server using the Rsync protocol[20] over SSH and the management network.

The Rsync command is launched periodically by the controller. It does not include the option *–delete*, so once a file is synchronised, it can be removed in the node (it will persist in the server). The result of this process is a directory in the server with the following structure:

- root / nodeA / experimentX / files
- root / nodeA / experimentY / files
- root / nodeB / experimentX / files
- ...

H. Repository structure

The Wibed platform software has been divided into four source repositories (using GIT-SCM[21]):

- wibed-packages
- wibed-controller
- openwrt
- openwrt-packages

All of them can be found under the master project named Wibed[22]. The node's firmware has been structured as an OpenWRT feed[23]. A feed is a collection of packages which share the same location. It can be imported from any standard OpenWrt buildroot[24].

To preserve the coherence between parts (as the OpenWRT source is changing quite fast), a frozen clone of the OpenWRT buildroot and main packages feed is used with some minor modifications. Finally, the Wibed controller has its own repository which is not directly related with the node's repository.

VI. UPC CN-A Testbed

A. Location

The UPC CN-A Testbed consists of 50 nodes that are being deployed over six buildings of Campus Nord (CN) of UPC, Barcelona. The buildings, known as the "A" buildings, are in a 260 meters long and 20 metres wide area. Four of them are four storeys tall and the other two are five storeys tall. It is planned to deploy two nodes per storey and to connect one node per building to the campus wired network. The outer walls and slab floors of the buildings, made out of rather thick reinforced concrete, are known to have strong WiFi attenuation effects. Thus the mesh network of UPC CN-A testbed is expected have not less than six hops of network diameter.

[20]http://rsync.samba.org

[21]http://git-scm.com/

[22]https://redmine.confine-project.eu/projects/wibed

[23]http://wiki.openwrt.org/doc/devel/feeds

[24]http://wiki.openwrt.org/about/toolchain

TABLE III
Testbed nodes characteristics

Vendor / model	TP-LINK / TL-WDR3900
CPU OpenWRT target	Atheros AR9344@560MHz / ar7xxx/ar9xxx
RAM / FLASH / External	128MB / 8MB / 16GB
WNIC 1	Atheros AR9341 (2.4 GHz, 2T2R, b/g/n)
WNIC 2	Atheros AR9580 (5GHz, 3T3R), a/n)
Lan Eth. / Wan Eth. ports	4x1 GigE / 1x1 GigE
USB / JTAG / Serial ports	2 / yes / yes
Volt. / Max .Cur. / PoE	12VDC / 1.5A / No

B. Hardware selection

The specific hardware selected for the UPC CN-A testbed nodes is the dual radio TL-WDR4300[25] from the TP-LINK vendor that is available for around 60€, excluding VAT and shipping costs. Its main characteristics are summarised in Table III. Each node has a USB stick of 16GB of storage capacity, available for less than 10€. In addition, the second USB port could be used for connecting another WIFI WNIC or USB-based spectrum analyzer to further extend experimentation capabilities.

C. Testbed controller

The controller can be installed in any standard x86 compatible PC running any Linux distribution. In our case it has been allocated in a virtual machine (1GB of RAM, 200GB of storage capacity in Intel i7 shared processor) of the CONFINE project facilities.

VII. Discussion

A. Current status and further work

Regarding the platform, although development is still a work in progress, the main functionalities are implemented and tested. The main tasks pending are rewriting some parts of the code, further testing of the whole platform, fixing detected bugs, and synchronising the documentation with the final implementation.

On the testbed deployment side, half of the nodes have already been purchased and potential deployment locations have been identified. The deployment is scheduled to start in September and will extend until November.

The CONFINE integration at the controller level is a topic that has not yet been addressed because it requires some modifications to the CONFINE's controller. This topic is planned to be dealt with after the UPC CN-A testbed is fully deployed and operational.

B. Proof of concept

As part of the development process a small testbed (seven nodes with two wired connections to the server) has been setup in our laboratory. On this testbed most of the functionalities of the Wibed platform (e.g. the auto-configuration of the management network, the experiment overlay system, the storage system) have been independently tested and validated.

[25]http://www.tp-link.com/en/products/details/?model=TL-WDR4300

Additionally, a complete simple experiment (each node stores the output of 100 ping6 to multicast address[26]) using all Wibed functionalities has been run successfully.

C. Costs and replicability

As already mentioned, the entire Wibed platform is a free/libre-software project, and thus available to everybody at no cost. The total cost of the hardware of the presented 50-nodes testbed is below 4.000 €. The skills required for designing and implementing wireless experiments may suffice to install and operate a Wibed-based testbed. Thus, the solution here presented allows the deployment and execution of a fully operational wireless testbed at a fraction of the cost required by most other available testbeds.

VIII. Conclusion

In this work we have presented Wibed, a platform for deploying and managing testbeds for experimenting on mesh networks built on top of COTS IEEE802.11 routers. We have presented its design, and how nodes evolve throughout the execution of an experiment and react to commands given by a central controller. We have also described how these nodes interconnect to one another and, eventually, to the controller server. By focusing on a very pragmatic and simple ad-hoc operation and management we have achieved to reduce both the budget and effort requirements for the setting up of link-layer to application-layer experiments over these wireless testbeds.

In addition, we have also presented details regarding an ongoing deployment of this testbed in a real-world scenario, encompassing 50 nodes spread throughout six buildings at the Campus Nord of UPC, Barcelona. Once this deployment is complete and the platform matures, it is our objective to open it to other researchers, providing a physical testbed on which novel algorithms and systems designed for wireless mesh networks may be tested and verified.

Acknowledgment

This work is partially supported by the European Community Framework Programme 7 (FP7) within the Future Internet Research and Experimentation Initiative (FIRE), Community Networks Testbed for the Future Internet (CONFINE), contract FP7-288535.

References

[1] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, Jan. 2003. [Online]. Available: http://doi.acm.org/10.1145/774763.774772

[2] "Community networks testbed for future internet (confine)," European Commission FP7, Future Internet Research and Experimentation Initiative (FIRE), http://confine-project.eu.

[3] L. Navarro, P. Escrich, R. Baig, and A. Neumann, "Community-lab: Overview and invitation to the research community," in *IEEE International Conference on Peer-to-Peer Computing*, IEEE Press. Tarragona: IEEE Press, 09/2012 2012. [Online]. Available: http://p2p12.org/program?view=article&id=101

[4] A. Neumann, I. Vilata, X. León, P. Escrich, L. Navarro, and E. López, "Community-lab: Architecture of a community networking testbed for the future internet," in *International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2012)*, IEEE Press. IEEE Press, 10/2012 2012.

[5] D. Johnson, N. Ntlatlapa, and C. Aichele, "A simple pragmatic approach to mesh routing using BATMAN," in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, 2008.

[6] D. Raychaudhuri, M. Ott, and I. Secker, "Orbit radio grid tested for evaluation of next-generation wireless network protocols," in *Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, ser. TRIDENTCOM '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 308–309. [Online]. Available: http://dx.doi.org/10.1109/TRIDNT.2005.26

[7] D. Wu, D. Gupta, and P. Mohapatra, "Qurinet: A wide-area wireless mesh testbed for research and experimental evaluations," *Ad Hoc Netw.*, vol. 9, no. 7, pp. 1221–1237, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2011.02.001

[8] M. Günes, B. Blywis, and F. Juraschek, "Concept and Design of the Hybrid Distributed Embedded Systems Testbed," Freie Universität, Tech. Rep. TR-B-08-10, 2008.

[9] B. Blywis, M. Güneş, F. Juraschek, and J. H. Schiller, "Trends, advances, and challenges in testbed-based wireless mesh network research," *Mob. Netw. Appl.*, vol. 15, no. 3, pp. 315–329, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1007/s11036-010-0227-9

[10] S. Keranidis, D. Giatsios, T. Korakis, I. Koutsopoulos, L. Tassiulas, T. Rakotoarivelo, and T. Parmentelat, "Experimentation in heterogeneous european testbeds through the onelab facility: The case of planetlab federation with the wireless nitos testbed." in *TRIDENTCOM*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, T. Korakis, M. Zink, and M. Ott, Eds., vol. 44. Springer, 2012, pp. 338–354. [Online]. Available: http://dblp.uni-trier.de/db/conf/tridentcom/tridentcom2012.html#KeranidisGKKTRP12

[11] D. Giatsios, A. Apostolaras, T. Korakis, and L. Tassiulas, "Methodology and Tools for Measurements on Wireless Testbeds: The NITOS Approach," Book chapter in Springer Lecture Notes in Computer Science (LNCS), vol. 7586, 2013.

[12] R. Merz, H. Schiberg, and C. Sengul, "Design of a configurable wireless network testbed with live traffic," in *Testbeds and Research Infrastructures. Development of Networks and Communities*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, T. Magedanz, A. Gavras, N. Thanh, and J. Chase, Eds. Springer Berlin Heidelberg, 2011, vol. 46, pp. 189–198. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17851-1_15

[26]https://wiki.confine-project.eu/wibed:example