WiBed, A Platform for Commodity Wireless Testbeds

Pau Escrich
Guifi.net Foundation, Catalonia
{pau.escrich}@guifi.net

***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 FOSS platform for WiFi testbeds based on OpenWRT Linux made to run on commodity IEEE802.11 WiFi routers part of the Community-lab.net project, a global testbed for Community networks. WiBed has been designed to support realistic low layer network experiments (according to the OSI model). This work recolects the details of the architecture, design and implementation of WiBed consolidated during its operation as a testbed. In addition to a set of routing experimentation results obtained during the Wireless Battlemesh v7 where WiBed was used as testbed platform.**

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

I. Introduction

Current standard research facilities such as PlanetLab[1] or Community-lab[2][3] (developed by the Community Networks Testbed for the Future Internet (CONFINE)[4] project), integrate virtualisation techniques to allow running experiments in parallel. Nonetheless, the adoption of virtualisation has the drawbacks of higher node costs due to increased requirements for computing resources, such as CPU and memory, and for ensuring proper isolation of experimentation resources, restricting access to the lower layers of the operating system and communication stack. As a result, cost increase entails a reduction of the number of experimentation nodes being deployed and 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 diminish the hardware restrictions: the capability of running a GNU/Linux system and having two ath9k[1] 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) wireless routers available in the retail market for less than 100€, allowing the deployment of WiBed-like testbed of tenths of nodes for a few thousand Euros. Thanks to a minimised management system, WiBed allows experiments from link-layer to application-layer.

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

Currently, a WiBed testbed of 20 nodes has been deployed over two buildings of Universitat Politècnica de Catalunya (UPC) Campus Nord, Barcelona. The resulting testbed is available to the researchers as part of the CONFINE project. It has been successfully used as basis platform for the Wireless Battle of the Mesh 2014[2], where several routing experiments were executed.

The remainder of this paper is organised as follows. Section II reviews the related work and section III introduces the necessary background. Section IV presents the platform design. In section V we provide a more detailed view of the platform implementation and in section VI we describe the testbed we built at UPC based on the WiBed platform. Section VII introduces the preliminary results of the routing experiments performed during the Wireless Battle Mesh v7 using WiBed as basis platform. In section VIII we discuss the current development status of the platform, the results gathered during the BattleMesh event, the work pending, the replicability of the solution presented and costs entailed. Finally, section IX presents the conclusions and the lessons learnt.

Fig. 1. WiBed logo

II. Related work

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 the future federation of it as a directive. 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

[2]http://battlemesh.org/BattleMeshV7

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 April 2014 Community-lab has around 80 operational nodes distributed among three community networks. The advantage of Community-Lab is to be the first testbed that is embedded inside of production community network (Guifi.net, AWMN, FreiFunk), providing thus an experimentation environment within a real used network with real traffic. Community-Lab however is limited regarding conducting L1-L3 experiments, due to potential service interruptions that these experiments may produce. With regards to this feature, WiBed extends Community-Lab by enabling L1-L3 experiments.

B. Roofnet

The Roofnet testbed presented by Bicket and Aguayo [5] in 2003 provides one of the first works on characterizing the implementation of a WiFi network testbed. It is physically located on Cambridge/Massachusetts in a total approximate area of 1.25 Miles (2 KM). The nodes (37) are mainly placed in houses of MIT students and volunteer citizens houses, except from the three gateways which are placed in university buildings and connect the testbed with the MIT wired network. The nodes are indoor x86 (500MHz) Mini-ITX computers with a a single 802.11b omnidirectional antenna and a single Ethernet port. The software is based on RedHat 9 (Linux Kernel 2.4.20) and it is fully customized and ready to operate inside the network. A proactive, link-state, source-routing protocol called Srcr, integrated in each node is responsible for routing the traffic in the flat and fully meshed network topology. Further analysis provides in-depth measurements and simulation results on the link and end-to-end performance and about the topological characteristics of this network. The value of Roofnet today is mainly that of having been a path maker for wireless network testbeds. The WiBed testbed also follows Roofnet's approach, being deployed in a relatively small area.

C. ORBIT

The Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT)[3] [6] testbed is centred around the "radio grid emulator". It provides 400 programmable radio nodes for at-scale and reproducible emulation of next-generation wireless network protocols and applications located in a 460m^2 room at Rutgers University, New Jersey, USA. It can be accessed via an Internet portal, which provides a variety of services to assist users with setting up a network topology, programming the radio nodes, executing the experimental code, and collecting measurements. The testbed also supports end-to-end wired and wireless experiments using a combination of ORBIT and OpenFlow switch/router nodes under the same experimental execution framework. The radio grid is also supplemented by a number of outdoor and vehicular nodes (both WiFi and WiMAX) deployed on or around the university campus, to be used for real-world validation of results or for application trials. The ORBIT testbed has also been federated into the Global Environment for Network Innovation future Internet (GENI)[4] research network and can be used for integrated global-scale Internet experiments involving wireless access networks. It has been operative since 2005. While Orbit offers a very comprehensive solution for wireless network experiments, it is not really open to be used by the researchers which are targeted with WiBed, i.e. developers related to community networks and not necessarily linked the academic community.

D. NITOS

Network Implementation Testbed using Open Source code (NITOS)[5] offered by NITLab, is a wireless experimental testbed that is designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real-world settings. It has been developed in the city of Volos, Greece by OneLab partner CERTH, in association with NITLab, the Network Implementation Testbed Laboratory of the Computer and Communication Engineering Department at the University of Thessaly. NITOS consists of nodes based on commercial Wi-Fi cards and Linux-based open-source platforms, which are deployed both inside and outside of the University of Thessaly's campus building. Currently, three kinds of nodes are supported: ORBIT-like nodes, diskless Alix2c2 PCEngines nodes, and GNU/MIMO nodes. NITOS is remotely accessible and gives users the opportunity to implement their protocols and study their behaviour in a real-case environment. Users can perform their experiments by reserving slices of the testbed though the NITOS scheduler. The control and management of the testbed is achieved by using control Management Framework (OMF) open-source software. OMF simplifies the procedure of experiment defining and offers a more centralized way of deploying experiments and retrieving measurements. CERTH, as part of the OneLab[6] project, collaborates with other European institutes and it is in the process of federating NITOS with other testbed facilities, in particular PlanetLab Europe, providing in this way access to a unified European experimental infrastructure. Currently the testbed has 40 nodes. NITOS as part of PlanetLab and Fed4Fire, is an open testbed, to be used by external researchers, and in this sense is similar to WiBed. WiBed, however, uses less-powerful hardware for its nodes, reflecting more realistically the typical hardware of ad-hoc networks and community network routers.

[3]http://www.orbit-lab.org/

[4]http://www.geni.net/

[5]http://nitlab.inf.uth.gr/NITlab/index.php/testbed

[6]http://www.onelab.eu/

E. w-iLab.t

The w-iLab.t [7] testbed is an office indoor testbed covering three floors of a building owned by iMinds in Belgium. In addition to the WiFi mesh/ad-hoc experiments it also includes support for sensors. The node's hardware is based on Alix boards [7] with two 802.11a/b/g radios, dual band antennas and a set of sensors attached to the USB port. The nodes run a customized version of Voyage Linux[8], a Debian based distribution for embedded devices. The management infrastructure is provided by a wired Ethernet network which connects all the nodes to a central control server. The users of the testbed access the controller to prepare, deploy, monitor and obtain the results of the experiments through a web based interface based on the MoteLab[9] software. To allow full experimentation (including changes in the Kernel) the w-iLab nodes use Preboot Execution Environment (PXE) and Network File System (NFS) to select which disk partition boots (management or experimentation) and to transfer the experimentation filesystem. After a WiFi experiment completes, the node automatically reboots to be load with the initial management partition. W-iLab.t shares many characteristics with the WiBed testbed platform presented on this work with the main difference in the management network. Wibed can use either Ethernet or WiFi for this purpose becoming this way a more flexible and less expensive solution for testbed deployments.

F. BOWL

Berlin Open Wireless Lab Network (BOWL[8]) is a hybrid testbed/hotspot WiFi network with more than 40 outdoor nodes which allows experimentation and Internet broadband user access at same time. The complexity of ensuring the privacy and reliability of broadband users while allowing experimentation makes the project very challenging. On the other hand, this feature gives an extra value to the research since the experiments can be performed in a completely real environment with real traffic. BOWL is built using three different kind of hardware (ARM, MIPS and x86) with multiple 802.11/a/b/g/n radio cards and at least one Ethernet interface which is attached to the TUB campus wired network for management purposes. One of the WiFi radios is used as Access Point to bring Internet connectivity to plain users, the others can be used for research purposes. The nodes run OpenWRT Linux, a very small Operating System for embedded devices (used in this work too). The central controller implements a web based front-end to a configuration database which can be pushed to the nodes (UCI[10] style) by a researcher and which is applied in the first boot throw an auto-configuration mechanism. Custom filesystem binary images are also allowed, in this case the node is flashed from scratch with the new system. The BOWL testbed offers already many features WiBed aims to support. BOWL however is not conceived as an open testbed, and cannot be used by external people. WiBed, as part of Community-Lab, will be open to external developers and researchers.

G. QuRiNet

The Quail Ridge Wireless Mesh Network (QuRiNet)[11] [9] is an experimental wireless mesh and environmental sensor network, established in 2004 at the Quail Ridge Reserve in Napa County, California, USA. Set in the wildlands, free of electromagnetic noise and spectral interference, QuRiNet provides a wireless mesh testbed for the finer understanding and development of protocols for medium access control, efficient routing, efficient mobility support, and experimental validation of designs proposed by computer scientists. It also provides both infrastructure for the transmission and sensors for the collection of environmental, ecological and physiological data in real time. QuRiNet is a joint project with the UC Davis Natural Reserve System and the Networks Lab at the Department of Computer Science, UC Davis. Currently it has 34 operational nodes. All the nodes use IEEE802.11n, while most of them use solar powered due to the lack of connectivity to the power grid. Different to WiBed, QuRiNet is an outdoor testbed. While from a system perspective it is similar to WiBed, however it's not open for external research.

H. Wireless Battle Mesh

The Wireless Battle Mesh (WBM), also known as the Battlemesh[12], is a totally horizontal event organised and driven by the participants 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 (WiBed). Both the code and the data sets of that edition are publicly available[13]. The node's hardware used is TPlink 4900, a 680MHz MIPS device, dual band, dual radio and 802.11n/MiMo compatible (same used in WiBed). The environment of the testbed is in general in extreme bad conditions, meaning very week links, very noisy spectrum and a permanent instability of the nodes (people moving them, switching them off, reinstalling them, etc.). However this is presented as a feature, since the objective of the event is to compare the routing protocols and for such purpose the more challenging the environment the better. On the other hand, collecting information in this environment is a hard task or even impossible for some cases. WiBed aims to be very close in terms of features to the Wireless Battle Mesh testbed. The Wireless Battle Mesh testbed, however, is a temporal testbed, only mounted and available during the Wireless Battle Mesh event. WiBed aims to offer permanent availability to researchers and developers.

[7]http://pcengines.ch/alix.htm

[8]http://linux.voyage.hk

[9]http://motelab.eecs.harvard.edu

[10]http://wiki.openwrt.org/doc/uci

[11]http://qurinet.ucdavis.edu/

[12]http://battlemesh.org/

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

I. Comparison

Table I and table II summarises the main characteristics of the analysed testbed. While the WiBed testbed shares some of its features with other testbeds, it is the only WiFi testbed for L1-L3 experiments which is truly open to external developers and researchers (due its combination and integration to Community-lab). In addition, according to its design, the WiBed platform is the only one which can be deployed using low cost hardware (commodity routers) becoming thus the easiest solution for any research group that needs to perform network experiments.

III. Background

A. 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.

B. OpenWrt

OpenWrt[14] is a Linux distribution that has become the de-facto standard[15] 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[16] and Unified Configuration Interface (UCI)[17]. OpenWrt also provides a feeds system, to integrate additional packages (over 3.500 available by default) into the firmware[18].

C. Wireless Battle Mesh

In the sixth edition of the WBM (referenced in section II-H) 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[19]. In the seventh edition (Leipzig, Germany 2014) the battle testbed was deployed using the WiBed platform. In this edition the project was announced to the international community and several experiments were executed. At the time of this writing the final results are being processed and will be published in the next months. In this work we have included some initial results which can be found in section VII.

[14]https://openwrt.org/

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

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

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

[18]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.

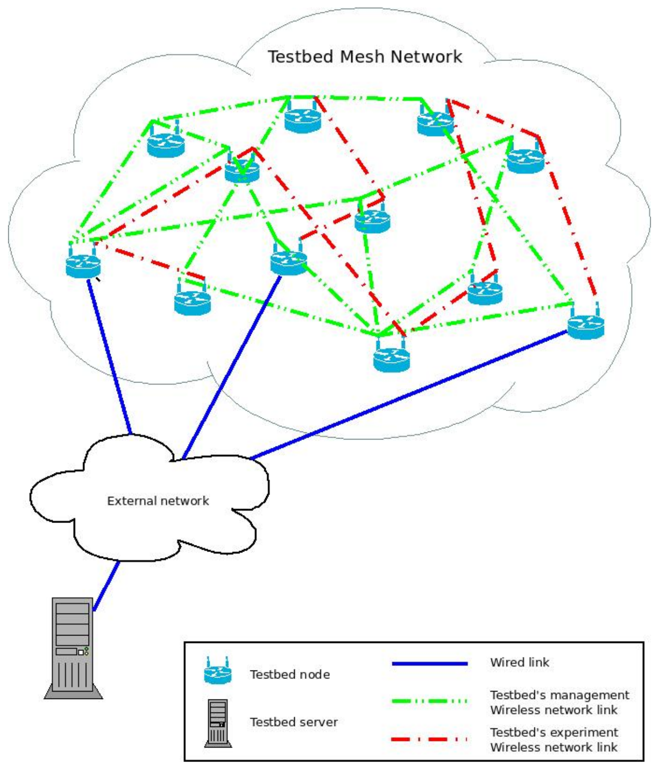

Testbed Mesh Network
External network
Testbed node
Testbed server
Wired link
Testbed's management Wireless network link
Testbed's experiment Wireless network link

Fig. 2. System architecture.

IV. Design

As shown in Fig.2, 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*. 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* the controller issues for the node's requests. Controller orders are node-specific, making it possible, for instance, to stop the experiment execution on a single specific node but also in a set of nodes.

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

TABLE I
TESTBEDS COMPARISON (1/2)

	Community-lab	**RoofNet**	**Orbit**	**Nitos**
Location	Europe	Massachusetts	New Jersey	Volos/Greece
Administration	CONFINE project	MIT	Rutgers UNI.	NITLab
Kind	EU research testbed	Mesh network	Research testbed	Research testbed
Year	2011	2003	2005	2009
Number of nodes	80	37	400	40
Environment	Indoor	Indoor/Outdoor	Indoor	Indoor/Outdoor
Node's place	Volunteer households	Volunteer households	Laboratory	University
Base hardware	x86 (atom)	x86 (500MHz)	x86 (core i7)	x86 (core i2)
WiFi Radios	0/1/2 (ABGN)	1 (B)	1/2 (ABG/ABGN)	2 (ABG/ABGN)
Base software	OpenWRT	RedHat 9	Linux based	Linux based
Main purpose	Research L3-L7	Internet and Research	Research L1-L7 / Sensors	Research in L1-L7 / Sensors
Controller	Web-UI	none	Web-UI	Web-UI
Management	VPN Overlay	Raw over the network	Wired/OpenFlow	Wired/OMF

TABLE II
TESBEDS COMPARISON (2/2)

	QuRiNet	**WBM**	**w-iLab.t**	**BOWL**	**WiBed**
Location	California	Nomad	Ghent/Belgium	Berlin	Catalonia
Administration	UNI. of CAL.	WBM participants	iMinds	TUB	UPC
Kind	Mesh network	Temporal testbed	Research testbed	Research / Internet	Research testbed
Year	2011	2003	2005	2009	2014
Number of nodes	34	10-80	200	58	20-40
Environment	Outdoor	Indoor	Indoor	Indoor/Outdoor	Indoor
Node's place	Natural Reserve	Any	University	University	University
Base hardware	x86 (266MHz)	MIPS	x86 (alix2)	ARM/MIPS/x86	MIPS (680MHz)
WiFi Radios	2 (BG)	1/2 (BGN/ABGN)	2 (ABG)	2 (ABGN)	2 (ABGN)
Base software	Linux Based	OpenWRT	VoyageLinux	OpenWRT	OpenWRT
Main purpose	Research L3	Competition / Research	Research L1-L3 / Sensors	Research L1-L3 / real traffic	Research L1-L3
Controller	none	none	Web-UI	Web-UI	Web-UI
Management	Raw over the network	Isolated MANET	Wired/PXE	Wired	Isolated MANET

Experiments, as explained in V-B, 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 when an experiment is running.

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.

V. 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[20].

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

All messages exchanged are in JSON[21] standard text format.

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

TABLE III
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

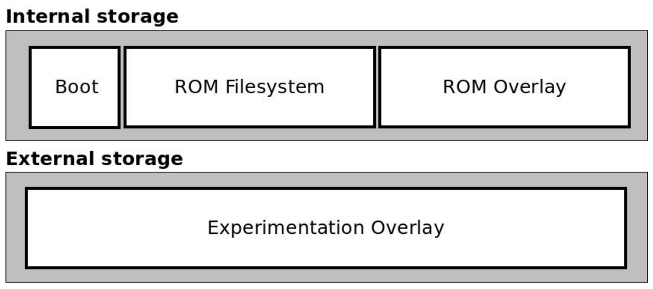


Fig. 3. Nodes firmware filesystem architecture.

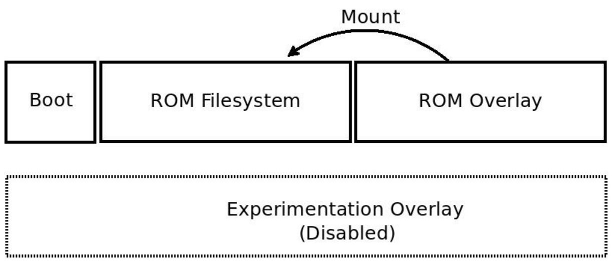


Fig. 4. Node firmware filesystem in the IDLE status.

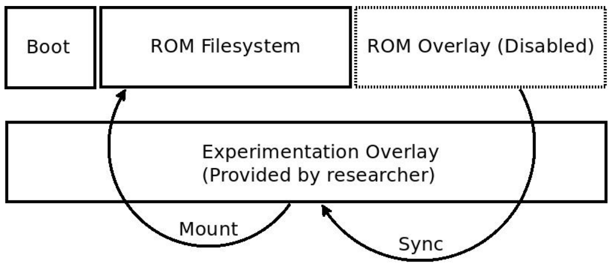


Fig. 5. Node firmware filesystem in the RUNNING status.

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 system (kernel, a minimal root filesystem and the testbed management software) and a JFFS2 mounted as OverlayFS[22] over the read-only partition to store filesystem changes.

The standard boot process of OpenWRT is as follows[23]:

1) The kernel boots from ROM and executes */etc/preinit*
2) */etc/preinit* executes */sbin/mount_root*
3) *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.3 WiBed extends this approach by adding a third component: a second overlay aimed at allocating the experiment placed in an external storage device[24] (such as a USB stick) . The *experiment overlay* is only mounted during the execution of an experiment. As Fig.4 and Fig.5 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.

[22]Included in Linux Kernel mainline 3.11
[23]For further information see http://wiki.openwrt.org/doc/techref/filesystems.
[24]The additional storage device overcomes the common space limitation of the internal storage in current COTS routers.

C. Nodes firmware overview

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 [25].
A single generic firmware image is used for installation in all testbed nodes. During the first boot, each node configures itself writing the changes to the internal 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 (MGMT-NET) 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[26].
- 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 IV. Fig.6 shows the node finite-state machine with transitions resulting from a

[25]https://wiki.confine-project.eu/wibed:config
[26]X are 8bits predefined and shared by all testbed nodes. R1 and R2 are 16bits from the CRC16-HASH

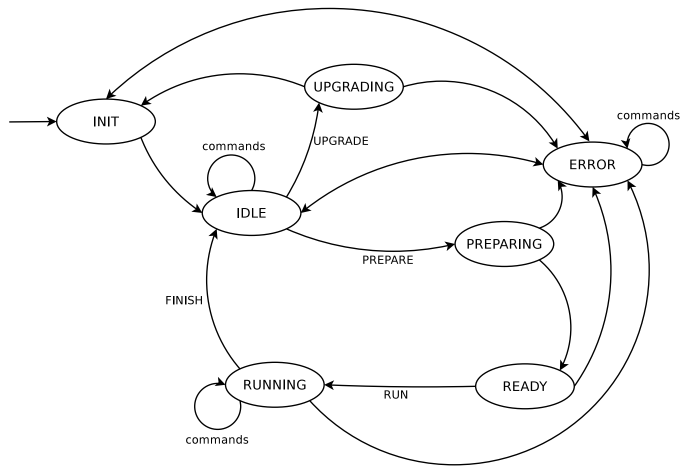


Fig. 6. Node finite-state machine.

TABLE IV
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	DEPLOYING	Installing the overlay and rebooting the node
5	RUNNING	Experiment running
6	RESETTING	Resetting the node to its default configuration
7	UPGRADING	Upgrading firmware
8	ERROR	Error detected

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 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 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 needed information and/or commands in response to a node request. Consequently, the network bandwidth required is optimized.

The registration of new nodes on the controller is made in a totally reactive 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 attached to the testbed in the INIT state and 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. Updated node information on the server side allows the delivery of compatible firmware to testbed nodes and the researchers to choose nodes with similar hardware and up-to-date firmware, among others.

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.

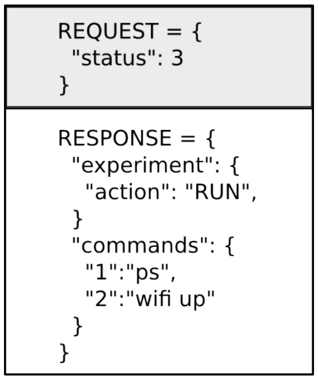
F. Communication API

To communicate the nodes with the controller a REST-API has been implemented. Nodes are constantly pulling the controller throw HTTP to receive the orders and send information to it. The basis API-URL pulled is *http://[controllerIP]/api/wibednode/[nodeID]*. If a node with the specified node ID did not exist, it is automatically added to the controller's database with this call. Every node's request contains the status code defined in section V-D. The controller replies according the finit state machine depending on what the researcher or the administrator wants to do with the node.

The text protocol used for communication is JSON (simple, well known and human-readable) as shows the example API call of figure 7

G. Remote management

In order to manage the nodes remotely a centralized management system has been implemented. Attached to a controller's response, a set of commands can be added as shows the figure 7. Commands are group in two categories, the experimentation ones (executed during an experiment by a researcher) and the administration ones (executed by the testbed administrators at any time). The output of the executed commands (both stdout and stderr) are attached to the next communication message to the controller. This way all nodes can be managed from a single point, making unnecessary the individual remote access through SSH[27].



```
REQUEST = {
  "status": 3
}
```

```
RESPONSE = {
  "experiment": {
    "action": "RUN",
  }
  "commands": {
    "1":"ps",
    "2":"wifi up"
  }
}
```

Fig. 7. API request/reply example with commands attached

H. Management network

The MGMT-NET is used to connect the research devices between the and with the testbed controller. 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 MGMT-NET 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[28] 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).

I. Data storage

To save the experiment results, a special directory placed on */save* is provided. Its content is synchronised with the controller server once the experiment is finished. Afterwards the researcher can access the stored data through the controller's web interface.

[27]Secure Shell
[28]http://open-mesh.org

J. Repository structure

The WiBed platform software has been divided into four source repositories (using GIT-SCM[29]). All of them are open to the public and released under a free licence.

- wibed-packages: the set of OpenWRT compatible packages implemented to run a WiBed node
- wibed-controller: the central web controller
- openwrt: a frozen version of OpenWRT based on the current trunk (Barrier Braker)
- openwrt-packages: a frozen version of the main OpenWRT packages repository

All of them can be found under the master project named WiBed[30]. The node's firmware has been structured as an OpenWRT feed[31]. A feed is a collection of packages which share the same location. It can be imported from any standard OpenWrt buildroot[32].

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.

K. Functional diagram

The figure 8 shows a diagram summarizes the executed steps of a WiBed node from its initial boot to the end of an experiment.

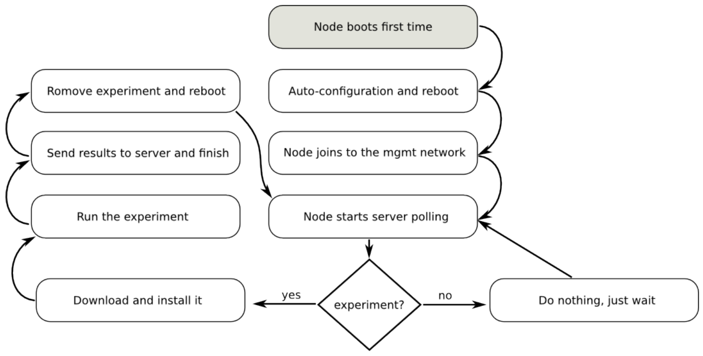

Node boots first time
Romove experiment and reboot
Auto-configuration and reboot
Send results to server and finish
Node joins to the mgmt network
Run the experiment
Node starts server polling
Download and install it
yes
experiment?
no
Do nothing, just wait

Fig. 8. Functional diagram for a WiBed node

VI. UPC A6 Testbed

A. Location

The UPC-A6 Testbed consists of 20 nodes that are deployed over three buildings of Campus Nord of UPC, Barcelona. The 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. Three gateways are deployed to connect with 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

[29]http://git-scm.com/
[30]https://redmine.confine-project.eu/projects/wibed
[31]http://wiki.openwrt.org/doc/devel/feeds
[32]http://wiki.openwrt.org/about/toolchain

TABLE V
TESTBED NODES CHARACTERISTICS

Vendor	TP-LINK
model	TL-WDR4300
CPU	Atheros AR9344@560MHz
OpenWRT target	ar71xx
FLASH	8MB
RAM	128MB
Wireless NICs	2
Wireless NIC 1	Atheros AR9341 (2.4 GHz, 2T2R, b/g/n)
Wireless NIC 2	Atheros AR9580 (5GHz, 3T3R), a/n)
Detachable antennas	3
Lan Ethernet ports	4x1 GigE
Wan Ethernet ports	1x1 GigE
USB ports	2
JTAG	yes
Serial port	yes
PoE	No
Power range	12VDC / 1.5A

network of UPC-A6 testbed is expected have not less than six hops of network diameter.

B. Hardware selection

The specific hardware selected for the UPC-A6 testbed nodes is the TL-WDR4300[33] from the TP-LINK vendor that is available for around 60€, excluding VAT and shipping costs. Its main characteristics are summarised in Table V. 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 analyser 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. The controller can be reach throw the url http://wibed.ac.upc.edu.

VII. BATTLEMESH V7 EXPERIMENTS

A. Motivation

The routing protocol is probably the most important piece of a Mesh network. It has the responsibility of decide how the packets flow among the multiple paths conforming a network. To this end, a good analysis and further comparison are very valuables tools for network communities to decide which protocol to use. The following work recollects a set of results which were taken during the Battlemesh v7 event in Leipzig, Germany.

In addition to the achievement of the experimental results, the usage of the WiBed platform pursuits to test the usability of the system to deploy a low cost mesh testing environment, to get feedback from the users and to fix potential problems.

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

B. Deployment

During the first days of the event a total of 20 WiBed nodes have been deployed. 16 WiBed nodes have been spread over 3 different floors in the main event building. About 10 of these 16 nodes were located in the main event hall (approximately 300 square-meters workshop room) with highest node density in a particular corner of this room and the 6 in the below and above floor of the event hall. Three more nodes have been placed in a neighbouring building, all belonging to the same WiFi cloud. One node was battery powered for allowing mobile-node scenarios. In fact not all node positions were always exactly known as nodes were sometimes moved to fulfil specific experimentation-scenario requirements. In each building 1 of the WiBed-nodes were configured as GW nodes and blocked for experimental usage. The remaining 18 nodes were shared between three different experimentation groups for running tests and different scenarios (each node was used by at most one experimentation group at any time).

All experiments were performed in a single 5Ghz channel. However, due to the presence of around 50 participants with wireless laptops and several other actively used wireless equipment, also the used 5GHz channel was likely affected by non-testbed related interference.

C. WiBed integration

Following the WiBed approach and requirements, an overlay filesystem packet in tar.gz format has been provided. It contains all the needed files to perform the experiments in addition to the protocol binaries and other tools which are participating in the experiment[34].

The overlay includes two special packets named *wbm-testbed* and *wbm-test-scripts* which can be found in the Battlemesh repository [35]. The first one contains a set of scripts which automatic configures the node and leaves it ready to participate in the experiments. The second one includes a set of scripts which will be used to perform the experiments and recollects the output.

As result, once the overlay is uploaded and installed to the WiBed nodes, all scripts will be automatically executed and the node will became ready to start the experimentation which can be initiated using the remote management interface (API commands described in section V-F).

D. Protocol Configurations and Assumptions

The highly dynamic and uncontrollable interference in the measurement environment made it impossible to ensure equal conditions for sequential experiment executions. Therefore, to ensure equal (fair) environment conditions for all tested protocols, all routing protocols were running and observed in parallel on all nodes, thus all being always exposed to the exactly same environmental conditions.

The accepted downside of this approach is of course that protocol overhead introduced by one protocol or by protocol-observing tools like ping (causing total overhead in the order

[34]http://wibed.ac.upc.edu/static/overlays/wbm-exp-axn-13.tar.gz
[35]https://github.com/battlemesh/battlemesh-packages

of few KB/s, as can be seen from later measurements) slightly affects the maximum achievable end-to-end throughput of other protocols (in the order of several MB/s).

Only netperf-tcp-based throughput probes, seeking to measure the achievable tcp performance of the end-to-end routing paths established by the individual protocols, were performed sequentially. This decision has been made because each netperf test tries to load the capacity of a given end-to-end path with a maximum of traffic, thus introducing maximum probing-traffic overhead and interference while on the other hand (given the tcp-inherent exponential backoff approach) drastically lowering the currently offered load in the presence of packet loss, leading to highly randomized results when running in parallel and making the comparison of parallel executions difficult.

All protocols were configured for routing IPv6 traffic using an individual ULA address prefix per protocol. All protocols were configured with default parametrizations, thus no environment specific customizations have been made apart from ensuring the routing of the given IPv6 address range. The exact configuration can be accessed via *wbm-config*.

To avoid protocol-bootstrapping effects (e.g. unfinished neighbour-, path- or topology-discovery), all routing protocol daemons were started at least 200 seconds before any measurement.

E. Measurement Configuration and Assumptions

Follow up measurements were executed from a singe selected node (src-node) by launching a pre-deployed test script *wbm-test* and given the id of a single other node (dst-node) for probing end-to-end path characteristics.

Each follow up measurement last 200 seconds during which the following additional tools were used to observe protocol performance and overhead:

- ping6 (unix) command to dst-node ipv6 address with one-second interval and 1000 bytes icmp user data. The output of the ping6 command got logged for later end-to-end packet loss, hop-count, and round-trip time (RTT) over time analysis.
- top (unix) command for logging protocol-specific CPU and memory consumption at 1 second intervals.
- mtr (my trace route, unix) command at 1 second interval for tracing full src-to-dst protocol-established path information. Due to the difficulty to correlate or graphically represent these traces, the obtained log files were not processed further.
- netperf, executed in repeating rounds (4 rounds), each probing sequentially the maximum achievable end-to-end throughput to always the same destination node for 10 seconds via each routing protocol.
- tcpdump, passively capturing the present routing-protocol overhead of each protocol as received on the wireless channel by the source node.

F. Experiments

The experiment focused on measuring the overhead and performance of 4 different mesh routing protocol implementations in static and mobile scenarios. The five tested protocols were batman-advanced[10] (batadv), bmx6[11], olsr and olsr2[12]. Experiments are grouped in two different scenarios: stationary and mobile.

1) Stationary scenario: During the stationary scenario the involved WiBed nodes were not moved, however still affected by uncontrollable interference conditions from the environment itself, other parallel experiments executed on other nodes, and the traffic created by the experiments itself. For the measurement a couple of source-node and destination-node was selected intuitively with the objective to select, in terms of network-topology, rather distant (so non-neighbouring) nodes.

2) Mobile scenario: Apart from the presence of a single mobile node, always serving as destination node, the mobile scenario was performed in the same way as the stationary scenario. During this scenario the mobile node was moved (carried) manually at slow-walking speed (approximately 1m/second) the about 100 meters back and forth along the main event hall, downstairs to the lower floor, and along the lower hall.

G. Protocol-traffic overhead

The protocol network traffic overhead in terms of bytes per second (figures 12 18). The continuous lines are presenting the measured overhead over time with a one-second resolution and the dashed lines the average overhead captured by the source node over the 200-seconds measurement period. It should be noted that the captured traffic includes all received protocol traffic, the traffic created by the capturing node itself as well as the traffic created by neighbouring nodes (being in transmission range of the capturing node). Only pure protocol-traffic was considered (not user-data related traffic) as created by each protocol to detect and establish the routing inside the network. For batman-advanced, encapsulating user traffic inside the same Ethernet frame type as used for protocol traffic, special filtering rules were used for differentiation.

H. Memory and CPU consumption

Memory and CPU consumption over time per protocol (figures 11 17 10 16). Both characteristics are not captured for batman advanced, which, running in kernel space, does not allow an easy profiling of this data. It can be seen that none of the captured protocols changed its memory requirements during this (nor any later) measurement.

I. RRT and hops

Round trip time (RTT) analysis shows the result of the ping6 tool from the source node to destination node (figures 14 20 13 19). Meaning the time (in milliseconds) needed for a single packet to go and return. The hops analysis shows the RTT for each protocol split by the number of hops (nodes between source and destination in the mesh network). Hops given for batman-advanced are incorrect due to its layer-2 routing characteristic. As expected, the RTT shows smaller values for less hops and higher latencies for more hops.

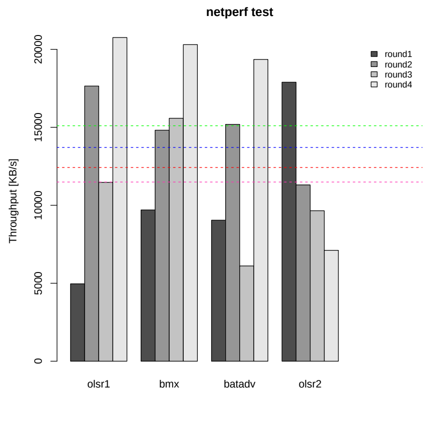


Fig. 9. Netperf throughput (stationary).

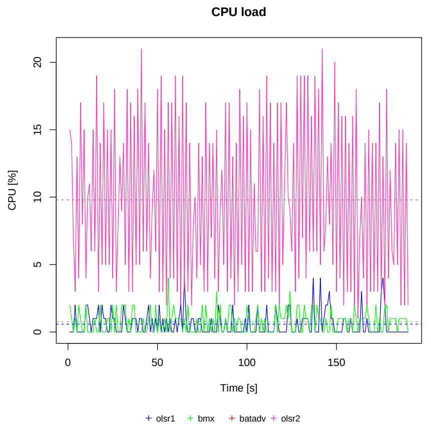


Fig. 10. CPU load (stationary).

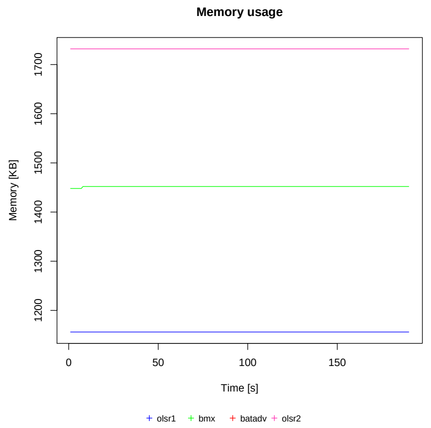


Fig. 11. Memory consumption (stationary).

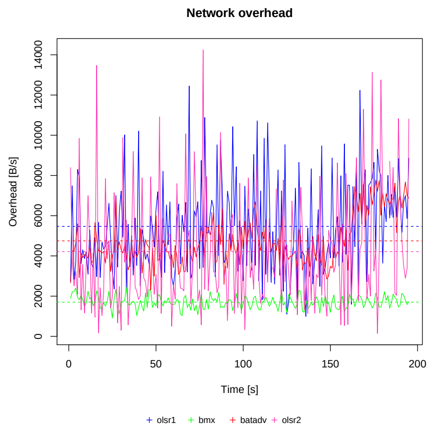


Fig. 12. Network overhead (stationary).

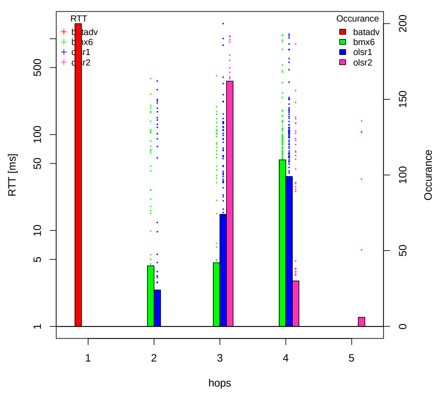


Fig. 13. Number of hops (stationary).

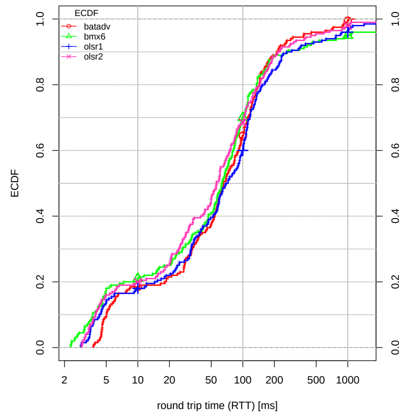


Fig. 14. Round trip time (ECDF) (stationary).

VIII. Discussion

A. *Experiment results*

Given the few number of measurements and the high amount of randomness involved, the results gathered during this measurement campaign are by no means representative. A general ranking of protocols can only be done regarding some few and particular characteristics. For other, if at all, at most some vague tendencies may be guessed. Main conclusion here is that much more measurements would be needed for concluding representative results. Thus, observations summarized in the following must be taken carefully.

OLSR showed in average medium results for measured path throughput, RTT, and total packet loss. Its observed CPU and memory consumption was always notable low. Regarding protocol-traffic overhead, it typically ranged within the most expensive (bandwidth consuming) protocols.

OLSR2 also showed in average medium results for measured path throughput, RTT, and total packet loss (OLSR2 driven path throughput was typically lowest but once notable high). Its observed CPU and memory consumption was always notably higher than any other protocol. Regarding protocol-traffic overhead, it showed average cost (bandwidth consuming more than bmx6 but less than olsr and batman advanced).

BMX6 showed rather good results for measured path throughput, RTT, and total packet loss (all four throughput measurements resulted in first or second position). Its observed CPU and memory consumption showed average cost. Regarding protocol-traffic overhead, it always showed least cost.

Batman advance also showed varying and in average medium results for measured path throughput, RTT, and total packet loss. Memory and CPU usage could not be measured. Regarding protocol-traffic overhead, it typically ranged within the most expensive (bandwidth consuming) protocols.

B. *Limitations and problems of the platform*

The main current limitation for the WiBed platform is the mandatory usage of a specific Linux Kernel (the one included in the base system). So researches cannot upload their own Kernel (they might add some pre-compiled modules). This issue might be solved using kexec[36], a tool which can be used to load a new Kernel in memory without the need of rebooting. However it would be very dangerous for the entire testbed due the lack of control on the new loaded Kernel and

[36]https://www.kernel.org/pub/linux/utils/kernel/kexec

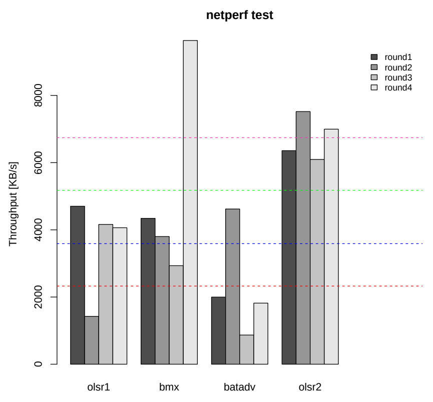


Fig. 15. Netperf throughput (mobile).

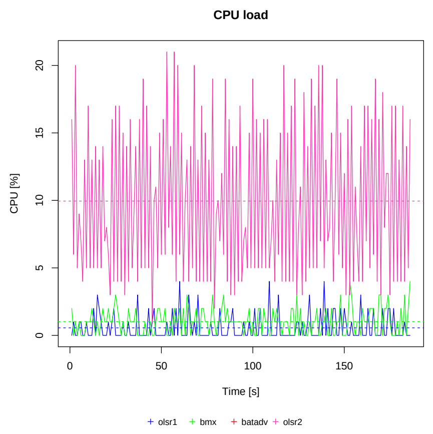


Fig. 16. CPU load (mobile).

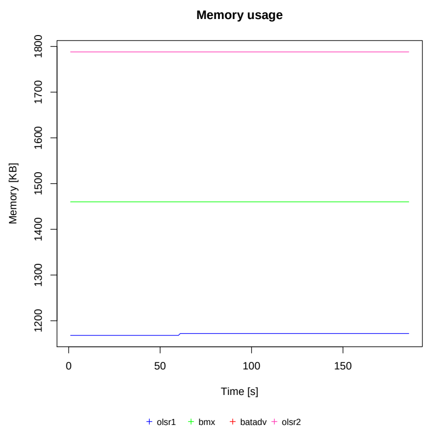


Fig. 17. Memory consumption (mobile).

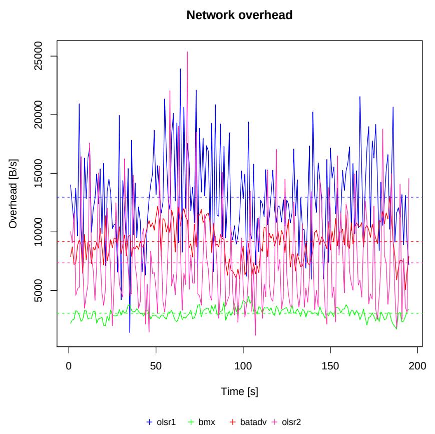


Fig. 18. Network overhead (mobile).

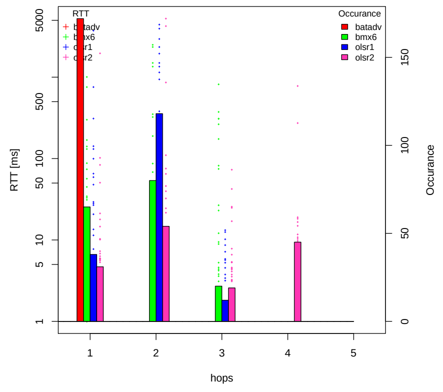


Fig. 19. Number of hops (mobile).

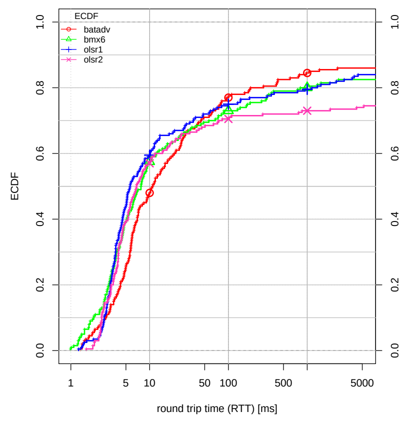


Fig. 20. Round trip time (ECDF) (mobile).

the possibility of breaking the management system (as a result the node would be lost and only a manual intervention could recover it).

Another problem on which we are currently working is the control on the required functions to guarantee the good operation of the management system. For instance, if the researcher rewrites the entire file */etc/config/wireless* which contains the configuration of the WiFi management network, the connection with the controller would be lost and the node would become unreachable. To this end, a script executed every five minutes has been implemented to check if the connection with the central controller is working. In case of failure it should reset the node to remove the overlay and start from scratch with the factory system. However this feature is not active because the whole platform is not yet stable enough and it might cause several problems in the entire system (i.e if the controller has an internal problem, all nodes would be reset). Currently the nodes just enter into ERROR state in case of three consecutive failures.

C. *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 further testing, bug fixing and synchronising the documentation with the final implementation.

On the testbed deployment side, twenty nodes are already deployed and available for experimentation. However the first plan was to deploy forty in the A1-A3 buildings. It has been not possible due the not expected problems to deploy the WiFi management network. The UPC buildings of the lower floors have reinforced concrete walls, which makes the WiFi signal very difficult to penetrate.

There have been already some experiments executed such as the INESC WiFiX[37] as part of the CONFINE first Open Call. Twenty more nodes are already ready to be installed in the next buildings in order to extend the testbed to forty nodes during the next months.

The experiments executed during the WBM in Leipzig demonstrate that WiBed is currently a useful platform for routing experimentation. Furthermore it was accepted by the community and now the main code is hosted under the official battlemesh repository in GitHub[38].

[37]http://win.inescporto.pt/Publications
[38]http://github.com/battlemesh

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

D. Costs and replicability

As already mentioned, the entire WiBed platform is a free/libre-software project, and thus available to everybody. The total cost of the hardware of the presented 20-nodes UPC testbed is below 1.500€. 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.

IX. 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.

We have also presented details regarding an ongoing deployment of this testbed in a real-world scenario, encompassing 20 nodes spread throughout six buildings at the North campus at UPC. 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.

According to the comparison made in this work, it has been seen that the WiBed UPC testbed is the only WiFi testbed for L1-L3 experiments which is truly open to external developers and researchers. In addition the implemented platform is free/libre open source, thus any individual or group of researchers can use it to quick deploy a low cost WiFi testbed.

Finally we have presented a dynamic routing protocol comparison experiment as a proof of concept for the usage of the WiBed platform. Network throughout, memory and CPU consumptions, round trip time and network overhead results for stationary and mobile node environments have been obtained and summarized in this work.

Acknowledgment

Special thanks to Roger Baig, Axel Neumann, Eloi Carbo and Emmanouil Dimogerontakis for their support and contributions on the implementation. Professors Felix Freitag and Leandro Navarro for making this work possible. To the Wireless Battle Mesh and OpenWRT folks for their constant work on the FOSS platforms which are basis for this work. To the Guifi.net Foundation for supporting the community networks. To Anna for her constant support in the real life.

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.

CONNECTED COMMUNITIES
CONFINE

This work has been presented as final thesis for the master *Programari lliure* imparted by *Universitat Oberta de Catalunya*.

UOC
Universitat Oberta de Catalunya

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] 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

[3] 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.

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

[5] D. Aguayo, J. Bicket, S. Biswas, D. S. De Couto, and R. Morris, "Mit roofnet implementation," 2003.

[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] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, "The w-ilab. t testbed."

[8] T. Fischer, T. Hühn, R. Kuck, R. Merz, J. Schulz-Zander, and C. Sengul, "Experiences with bowl: Managing an outdoor wifi network (or how to keep both internet users and researchers happy?)," in *Proceedings of the 25th Large Installation System Administration Conference (LISA11)*, 2011.

[9] 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

[10] "Batman Advanced L2 routing protocol," http://www.open-mesh.org.

[11] "BMX6 – a loop-free routing protocol for IP-based mesh networks," Jun. 2011, http://www.bmx6.net.

[12] "OLSRd – a routing protocol for IP-based mesh networks," Jun. 2011, http://www.olsr.org.