OpenTestBed: Poor Man's IoT Testbed

Jonathan Munoz, Fabian Rincon, Tengfei Chang, Xavier Vilajosana, Brecht Vermeulen, Thijs Walcarius, Wim van de Meerssche, Thomas Watteyne

OpenTestBed: Poor Man's IoT Testbed

Jonathan Muñoz[1], Fabian Rincon[2,1], Tengfei Chang[1], Xavier Vilajosana[3], Brecht Vermeulen[4], Thijs Walcarius[4], Wim Van de Meerssche[4], Thomas Watteyne[1]
[1] Inria, Paris, France. first.last@inria.fr
[2] ENSTA ParisTech, France. fabian-antonio.rincon-vija@ensta-paristech.fr
[3] Open University of Catalunya, Barcelona, Catalunya. xvilajosana@uoc.edu
[4] IDLAB, Ghent, Belgium. first.last@ugent.be

***Abstract*—Testbeds are a key tool for evaluating and benchmarking IoT solutions. Several public testbeds are being run by institutions around the world. These are built with a variety of tools, and are typically "heavy" installations with dedicated wiring, hard installations, switches, servers, and a reservation and experiment management back-end. To complement those, we have taken the opposite, minimalistic, approach in designing the OpenTestBed. The OpenTestBed features all the tools necessary to build a testbed from off-the-shelf components such as Raspberry Pi single-board computers, OpenMote B low-power wireless devices, and glass domes. Each TestBox in the testbed connects to an MQTT broker over WiFi, no dedicated wiring or back-end is needed. The Inria-Paris OpenTestBed testbed of 80 motes has cost only 9,480 euros, and is open-access. The OpenTestBed is a fully open-source and open-hardware project, which several institutions have already adopted.**

I. Introduction

Like any networking technology, IoT solutions must to be extensively tested and validated before being applied to real-world problems. Running a high-level simulation of the general behavior of a solution is generally the first step. The second step is usually to run the firmware that fully implements the solution on a testbed, a collection of low-power wireless devices deployed in a controlled environment.

The key service a testbed offers is to be able to load new firmware onto the devices, and observe their behavior when that firmware runs. More advanced services include user and testbed reservation management, toolchains to compile the firmware, energy measurement, and storing logs. Testbeds are usually built as part of a large research project, and represent a significant effort. This can go as far as building dedicated hardware, installing dedicated wiring for powering and networking the testbed, and running a large number of management servers. The danger is that the resulting testbeds take a long time to build, are quickly outdated, and are deployed in dedicated rooms, which does not represent the wireless environment of most real-world deployments. That being said, these large institutional testbeds have made an enormous contribution in pushing the IoT research to making deployment-ready solutions.

In this paper, we explore ways of creating a testbed using a minimalistic approach very complementary to larger institutional testbeds. We ask ourselves the following questions. *What are the minimal services a low-power wireless testbed should offer? Can this be built using only off-the-shelf components? How can we make sure a testbed as easy to install and operate as possible?*

Fig. 1. The Inria-Paris OpenTestBed before deployment. 20 OtBox contain a total of 80 OpenMote B boards.

The contributions of this paper are three-fold:

- We describe the architecture of the OpenTestBed in enough detail to allow interested readers to replicate it.
- We provide all the source code under an open-source license. The hardware required is widely available and cheap; the most important elements are open-hardware.
- We give examples of how the OpenTestBed is being used by different institutions.

The remainder of this paper is organized as follows. Section II surveys how different testbeds have been built and operated, focusing on low-power wireless testbeds. Section III presents the OpenTestBed platform, including how its architecture, and detailing hardware, software and installation. Section IV gives two use cases of how the OpenTestBed is being used by different institutions. Finally, Section V concludes this paper.

II. Related Work

Several institutional testbeds have been built over the last decade, and are available to the research community to run experiments. We are in particular interested in low-power wireless testbeds, which are composed of 10's to 100's of low-power wireless devices and used to develop IoT applications.

This section details the three testbeds we believe are the most closely related to the OpenTestBed. For a more exhaustive survey of testbeds, we refer the interested reader to the excellent work by Tonneau *et al.* [1].

One of the most advanced testbeds is the SmartSantander project [2], a 20,000 node, city-scale testbed which is installed in indoor and outdoor areas in the cities of Santander (Spain), Guildford (UK), Lubeck (Germany) and Belgrade (Serbia). The nodes include IEEE802.15.4 devices and GPRS modules. Only nodes which are mains powered are available to be (re-)programmed over-the-air through a second IEEE802.15.4 transceiver. The strength of the SmartSantander testbed is that it is deployed in an actual smart city environment, increasing the confidence one can have in the result it yields.

FIT IoT-lab [3] is a federation of open-source testbeds located across 6 cities in France, and composed of 2728 low-power wireless devices. A user can request an account, then reserve an arbitrary number of nodes for an arbitrary amount of time to conduct an experiment. Using that account, a user can log into a central Linux machine, in which she can recompile her binary. When an experiment is running, the user has bare-metal access to the low-power wireless devices, and she can load any arbitrary on any node. In the back-end, each low-power wireless device is connected to a single-board computer, which itself is wired into the testbed network over a dedicated Ethernet network with Power-over-Ethernet capabilities. The back-end consist of a series of servers, some local to each deployment site, and interconnected to a central set of servers in Paris. The filesystem of the single-board computers is mapped over NFS to the user's Linux account, resulting in very powerful logging capabilities. The user also has the option of doing in-circuit debugging on each low-power wireless device over JTAG. Moreover, each device is equipped with dedicated hardware to monitor instantaneous power consumption; at the heart of the system is an Analog-to-Digital Converter chip connected to a series resistor. FIT IoT-lab is arguably the most full-featured IoT testbed available today. But that comes at a price. First, because of the dedicated wiring, Power-over-Ethernet and NFS mapping, each of the 6 testbeds requires a dedicated Ethernet network to be put up across the deployment site. Because of the NFS mapping, the amount of data transitioning over that network is high, and piggy-backing the testbed traffic over the already existing Ethernet or WiFi network in the building was not an option. A side-effect of that is that, in most deployments, all devices are deployment in a single room, with the unfortunate side-effect that the wireless environment is very stable and not generally representative of a deployment done across an entire building. Finally, because of the feature-rich hardware needs (e.g. JTAG to all boards, power consumption measurement), FIT IoT-lab nodes are custom-made hardware. The unfortunate side-effect is that the low-power devices are not off-the-shelf, so a researcher outside of the FIT IoT-lab consortium cannot buy a handful of the same boards for local development.

The EWSN conference has featured a competition over the past 4 editions, organized by Boano *et al.* [4]. This competition has been the catalyst for creating and maintaining a testbed, which is evolving at each edition. The testbed consists of 51 TelosB low-power wireless devices deployed across a building at TU Graz, in Austria. Each TelosB is connected to a Raspberry Pi which runs the management software. The team has developed an open-hardware interface board between the Raspberry Pi and the TelosB to monitor energy consumption. The back-end solution consists of a very complete set of services custom made for the competition. Competitors submit a binary image they have developed outside of the testbed. That image is then loaded into the boards and an experiment runs for a pre-set duration. After the experiment, the testbed outputs the key performance indicators (latency, reliability, power consumption) that are used to rank the competitors.

A drawback that is often raised about testbeds is that the connectivity between the nodes does not represent that of a real-world deployment. That is because the deployment is done in a single room, and/or far from any source of interference. This means the wireless links exhibit a very good quality and very little variation over time. Brun *et al.* [5] have recorded the quality of the links between nodes both in testbeds and real-world deployment. From the analysis of the latter, they identify the three phenomena that often appear: external interference, multi-path fading and dynamism in the connectivity between nodes. They then develop a tool that verifies whether those as present in a testbed, and thereby quantify the "realism" of different testbed deployments.

This related work has served as a basis for defining the requirements for the OpenTestBed, as detailed in Section III.

III. The OpenTestBed

This sections details the OpenTestBed. We start by reviewing the requirements for the OpenTestBed, and the approach we have taken, for example to ensure the testbed is accepted by the occupants of the building in which it is deployed (Section III-A). From that, we detail the hardware (Section III-B) and software (Section III-C) used.

A. Requirements and Approach

As detailed in Section II, a testbed such as FIT IoT-lab offers countless features. In our experience of implementing protocol stacks for low-power wireless IoT network, some of these features are not strictly necessary. Our approach when developing this type of firmware has been to first develop it on a handful of boards on our desk. Having the hardware in front of us allows the use of in-circuit debuggers, logic analyzer and oscilloscopes, do the bulk of the development while verifying all works as expect on a network of 2-5 boards. The result of this work is a firmware image, which we now want to test at scale. Only then we do need a testbed, and the only thing we need from that testbed is to be able to load the firmware on all the devices, let an experiment run, and verify the performance of the network. Because each firmware is different and each developer wants to log different information, the most generic approach is to send and receive serial bytes to each of the nodes in the testbed.

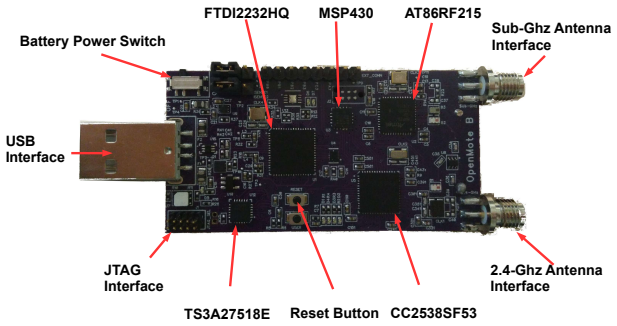

Fig. 2. The Openmote B sensor node.

This translates into the following (minimalistic) set of user stories for a firmware developer using the testbed:

- As a developer, I want the testbed to be composed of devices which are well-known, commercially available and state-of-the-art.
- As a developer, I want the testbed to be deployed in an environment which is representative of the environment of my final deployment.
- As a developer, I want to be able to load arbitrary binary images on any device at any time during an experiment.
- As a developer, I want to be able to reset/disable any device at any time during an experiment.
- As a developer, I want to be able to send and receive serial bytes with any device at any time during an experiment.

The operator is the person who builds and maintains the testbed. Since the testbed needs to be deployed in a building, the operator wants a solution that it easy to install, and which is accepted by the occupants of the building. Furthermore, she doesn't want to have to maintain a complex back-end system.

B. Hardware

We chose the OpenMote B (fig:openmote-b) as the low-power wireless device for the OpenTestBed. The OpenMote B is a state-of-the-art open-hardware low-power wireless device. It features two radios: the CC2538[1] IEEE802.15.4 compliant radio communicating at 2.4 GHz and the AT86RF215 [2] IEEE802.15.4g compliant radio communicating in the sub-GHz bands. The latter radio implements all IEEE802.15.4g-2012 modes (FSK, OQPSK, OFDM). The CC2538 also contains an ARM Cortex-M3 micro-controller, and is connected to the AT86RF215 over SPI. The OpenMote B has been designed with testbeds in mind.

The OpenTestBed consists of a number of "OtBoxes", shown in Fig. 3. Each OtBox contains:

[1] http://www.ti.com/lit/ds/symlink/cc2538.pdf

[2] http://ww1.microchip.com/downloads/en/devicedoc/atmel-42415-wireless-at86rf215_datasheet.pdf

Fig. 3. At OtBox: a Raspberry Pi, 4 OpenMote B boards, a screen and a QR code in a glass dome.

Component	Cost
4× OpenMote B	90 €
1× Raspberry Pi 3B +	50 €
1× IKEA glass dome	15 €
1× LED screen	30 €
1× USB extensions	14 €
1× engraved wood	5 €
Total	**474 €**

TABLE I
COST BREAKDOWN OF AN OTBOX.

- A **Raspberry Pi**. This single-board computer runs the OpenTestBed software connects to the back-end over WiFi. We use a 5 GHz WiFi (available on the Raspberry Pi 3B+) in order not to interfere with the OpenMote B board communicating at 2.4 GHz.
- 4 **OpenMote B** motes.
- A **screen**.
- A **QR code** pointing to an explanation of the OpenTestBed.

The overall aesthetics of the OtBox ("barbershop" looking glass dome and laser engraves dark wood) are designed increase acceptability of the devices.

Table I details the cost of an OtBox.

C. Software

Each OtBox runs the `otbox.py` single-file Python program[3]. This program connects to each OpenMote B over its serial port, and offers the following services to a user, over a simple API:

[3] As an online addition to this paper, the OtBox software is available under a BSD open-source license at https://github.com/openwsn-berkeley/opentestbed/.

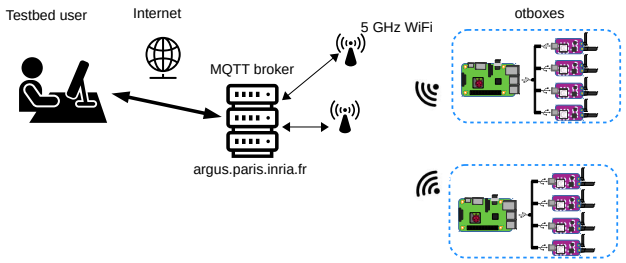

Fig. 4. OpenTestBed Infrastructure.

- Mote management: reprogramming any mote with any firmware, reset any mote, disable any mote.
- OtBox management: retrieve the status of the OtBox, discover the MAC address of the motes connected, upgrade the software, disply an image on the screen.
- Serial port forwarding: publish the bytes sent by any mote, send bytes to a motes.

This API is transported over MQTT. That is, each OtBox connects to a central MQTT broker, a popular publish-subscribe solution. To run an experiment, a user connects to the same broker and thereby can receive any notification sent by any OtBox, and issue commands. As such there is *no* "testbed server". Fig. 4 shows the overall architecture.

Purely for ease of use, we developed the OpenTestBed dashboard. This dashboard connects to the MQTT broker and allows a user to interact with the API by clicking on a web interface. The dashboard is *not* a "testbed server", and is not required for the OpenTestBed to run. The dashboard is developed as a Node-RED flow (which is part of the available source code).

The dashboard participates in the acceptability of the tesbed. Every 10 s, the dashboard will issue a command to send an image onto the screens of all OtBoxes. The result is the screens displaying different pictures in a round robin fashion.

IV. Examples Use Cases

A. Inria-Paris testbed

Fig. 1 is a picture of the Inria-Paris OpenTestBed right before deployment. It consist of 80 motes deployed in 20 OtBoxes. Fig. 5 shows the locations of the OtBoxes once deployed across two multi-story buildings. The OtBoxes are located throughout offices, meeting rooms and the main lobby. Since each OtBox just needs an electrical outlet, it can be installated anywhere. The full installation of the OpenTestBed takes less than an hour.

The total hardware cost of the Inria-Paris OpenTestBed is 9,480 €, an order of magnitude lower than some institutional testbeds of the same size. The OpenTestBed was developed and deployed by an engineering intern in 1 month.

To ensure acceptability of the testbed, the communications department of the institute manages the images that appear on the OtBoxes, turning the OtBoxes into information radiators

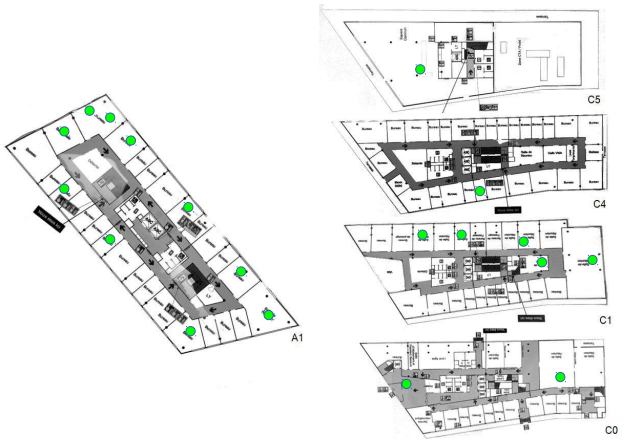

Fig. 5. The OtBoxes deployment across the Inria buildings A and C throughout different floors. Each green dot is an OtBox.

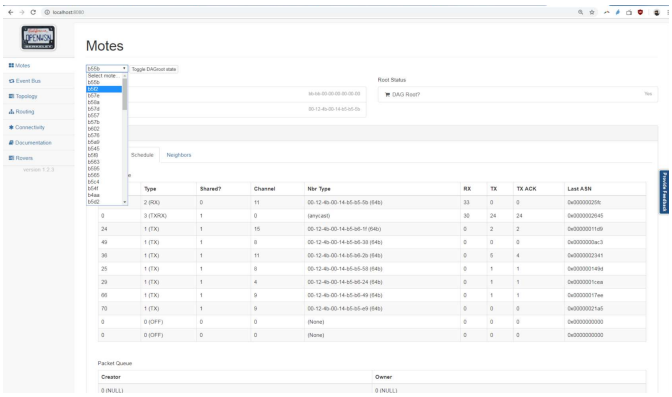

Fig. 6. The OpenVisualizer, the debugging/visualization tool of OpenWSN. The drop-down menu lists the motes in the Inria-Paris OpenTestBed.

(annoucements, events, etc.). The dashboard[4] runs as a service on IBM Cloud. The OtBoxes connect to a vanilla Mosquito MQTT broker running in the Inria datacenter.

The OpenTestBed has been very well received by the Inria community. People like its design, and the OpenTestBed often serves as a ice-breaker during meetings as many meeting rooms feature an OtBox.

B. Integration of the OpenTestBed into OpenWSN

OpenWSN [6][5] is the reference implementation of 6TiSCH, a protocol stack for the Industrial IoT standardized by the IETF. The OpenVisualizer is a tool to monitor and debug OpenWSN deployments. It shows the internal state (message queue, neighbour tables, scheduling table) of any node in the OpenWSN network on a web interface. It does so by parsing debug information each note periodically publishes on its serial port.

We added support for OpenTestBed into OpenWSN; this consists of two elements. First, we added

[4] For the Inria-Paris OpenTestBed, the dashboard runs at http://testbed.openwsn.org/.

[5] https://github.com/openwsn-berkeley/

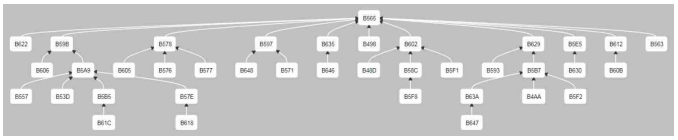

Fig. 7. Routing topology formed by an OpenWSN network running on the Inria-Paris OpenTestBed.

Fig. 8. Hardware employed in the testbed and the scenario where it is deployed.

`bootload=opentestbed` into OpenWSN's build environment. This allows a developer to automatically load the OpenWSN binary onto an OpenTestBed testbed as the last step of the build process. Second, we added the `--opentestbed` flag to the OpenVisualizer. This allows a developer to have the OpenVisualizer connect to all motes in an OpenTestBed instance, and visualize the state of all the nodes. Fig. 6 shows the resulting OpenVisualizer web interface. Fig. 7 shows the routing topology formed by those nodes.

C. w-iLab.t Testbed

The imec iLab.t testbed w-iLab.t is a diverse wireless testbed in Ghent, Belgium. It offers technologies such as 802.11a/b/g/n/ac, 802.15.1 (Bluetooth), 802.15.4, LTE and devices such as linux PCs, embedded IoT devices, software defined radios, mobile robots, environment emulators, and shielded boxes. Fig. 8 shows the environment of this testbed, and the different hardware used.

Zolertia re-motes (1 to 2 per node) are connected via USB to Linux boxes that can be reserved by experimenters. By default, the Linux box gets a fresh operating system automatically augmented with the ssh-keys of the team that has reserved the node. This testbed is compatible and federated with the Fed4FIRE and GENI API standards. The jFed tool is used to provision the Linux nodes.

The default Linux OS image loaded on the box has the necessary tools to configure the Zolertia re-motes, but there is no automation foreseen from the testbed side to flash all Zolertia re-motes at once. So each experimenter has his own toolset for doing this. By leveraging the OpenTestBed framework, the imec now offers this functionality to the testbed users.

jFed has the functionality to do advanced automated software deployment at the initial provisioning stage with the Experiment Specification functionality. The imec team used this to create an ESpec that deploys the OpenTestBed framework automatically. In the process, the tools to integrate natively with the w-iLab.t testbed were added to the OpenTestBed (including supporting different numbers of Zolertia devices per box, and automatic registration of unique addresses).

The simplicity of using the OpenTestBed framework with a simple command line interface is beneficial to our testbed users who are looking for a simple way to quickly manage an experiment with multiple motes.

V. Conclusion

This paper presents the OpenTestBed platform: a simple, cheap, versatile, scalable, easily deployable and replicable open-source testbed. Because of its simplicity, the OpenTestBed can easily be extended to support other low-power wireless devices. An 80-mote 20-OtBox OpenTestBed is deployed in an open-access fashion at Inria-Paris. The w-iLab.t testbed run by imec in Belgium now automatically starts an OpenTestBed instane for each low-power wireless experiment run. We hope the community can benefit from this platform and architecture, and that it can contribute to accelerating the development and evaluation of real-world IoT solutions.

References

[1] A. S. Tonneau, N. Mitton, and J. Vandaele, "How to Choose an Experimentation Platform for Wireless Sensor Networks? A Survey on Static and Mobile Wireless Sensor Network Experimentation Facilities," *Ad Hoc Netw.*, vol. 30, no. C, pp. 115–127, Jul 2015.

[2] L. Sanchez, J. A. Galache, G. V., J. M. Hernandez, J. Bernat, A. Gluhak, and T. Garcia, "SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities," in *2011 Future Network Mobile Summit*, June 2011, pp. 1–8.

[3] C. Adjih, E. Bacelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 459–464.

[4] C. A. Boano, M. Schuß, and K. U. Römer, "EWSN Dependability Competition: Experiences and Lessons Learned," *IEEE Internet of Things eNewsletter*, Mar 2017.

[5] K. Brun-Laguna, P. Henrique Gomes, P. Minet, and T. Watteyne, "Moving Beyond Testbeds? Lessons (We) Learned about Connectivity," *IEEE Pervasive Computing, Special Issue on Beyond Testbeds: Real-World IoT Deployments*, 2018, to appear in 2019.

[6] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: a standards-based low-power wireless development environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.2558