A step towards runnable papers using R2lab

Thierry Parmentelat, Mohamed Naoufal Mahfoudi, Thierry Turletti, Walid Dabbous

A step towards runnable papers using R2lab

Thierry Parmentelat†, Mohamed Naoufal Mahfoudi†, Thierry Turletti†, Walid Dabbous†

***Abstract*—In this paper, we present R2lab, an open, electromagnetically insulated research testbed dedicated to wireless networking. We describe the hardware capabilities currently available in terms of Software Defined Radio, and the software suite made available to deploy experiments. Taking as a pretext a dummy experiment, we show how it all fits into a notebook-based approach to getting closer to runnable papers.**

***Index Terms*—R2lab, Reproducibility, `nepi-ng`, Jupyter Notebooks, Faradized and Anechoic Chamber, Wireless Testbed**

I. Introduction

Validation and performance evaluation of new network protocols is done using different complementary approaches such as analytical modeling, simulation, experimentation or any combination of them [1]. In the wireless networking domain, modeling and simulation results may not be realistic enough, because the interaction between MAC and physical layers is complex to model due to the random behavior of the wireless environment. It is therefore essential in all cases to run experiments with real hardware and software components to get meaningful evaluation. In addition, in order to obtain sound scientific results, researchers should be able to reproduce the experiments in the same conditions.

However, it is well known that wireless networking experiments are difficult to reproduce due to interferences and fluctuating characteristics of the physical layer [2]. Therefore, in order to achieve reproducibility of experiments, we need an overall evaluation methodology providing sufficient details on the scenarios, hardware and software configuration and making available all the code and scripts used to replicate the experiments [3]. In addition, stochastic physical layer variables, acquisition hardware and software must all be controlled.

The R2lab testbed addresses precisely this controllability dimension. R2lab is an open wireless tested, located in an anechoic chamber for reproducible research in wireless Wi-Fi and 4G/5G networks. It provides controlled wireless hardware resources in this insulated environment, as well as rich software tools to allow reproducible research. R2lab is part of the FIT[1] federation, an open large-scale testing infrastructure for systems and applications on wireless and sensor communications. Located at Inria Sophia-Antipolis, R2lab proposes thirty seven customizable commercial off-the-shelf wireless devices, together with USRP nodes and commercial LTE phones, fit to create rich experimental setups.

R2lab runs on a global reservation mechanism - only one experiment is allowed to run at any given time. The nodes are exposed as bare metal, experimenters chose the software image they want to run on each node. Prebuilt images leverage some high end software suites - see sectionIII-C. Additionally, R2lab offers various software tools, to support easy and efficient experimentation. These tools allow to book the whole testbed, manage node images, remotely control the wireless devices, deploy various scenarios and collect results in an easy way.

R2lab is operational since December 2015 and several research activities have already used it (Mininet Wi-Fi [4], Orion [5], 5G network deployment [6], Wi-Fi conflict graphs [7], Software Defined Wireless Mesh Networks [8]).

The goal of this paper is twofold. It first provides a presentation of the R2lab components, both hardware and software, that are specific to R2lab. Then, on a more methodological note, we present the orchestration tool that is recommended for experimenters to script a complete experiment, and describe one sample Jupyter notebook that demonstrates how it could be possible to publish a runnable paper that explains how to reproduce a pre-written experiment.

The plan of the rest of the paper is as follows. Section II presents the R2lab hardware, and section III describes the R2lab-specific software tools. For the methodological aspects, section IV describes the `nepi-ng` orchestration tool, and section V shows a simple use case with a Jupyter[2] notebook providing the base for runnable papers. Section VI concludes the paper.

II. R2lab Hardware

A. State of the Art

Several testbeds are available in the wireless community. For instance, Iris[3], CortexLab[4] and CORNET[5] focus on cognitive radio research. Some others like ORBIT[6], NiTos[7] and w-iLab.t[8] not only target physical layer experts but also networking researchers, by proposing hybrid nodes (SDR devices and also PCs with COTS Wi-Fi and/or low-power technologies). R2lab belongs to this second category of testbeds and focuses on reproducible research, made possible by a faradized anechoic chamber and a collection of software used to simplify running experiments and writing runnable papers.

†Inria, Université Côte d'Azur, Sophia Antipolis, France

[1]Future Internet Testing Facility, see URL https://fit-equipex.fr/testbeds

[2]Open source Jupyter Notebook, see http://jupyter.org/.

[3]Iris, see https://iris-testbed.connectcentre.ie/.

[4]CortexLab, see http://www.cortexlab.fr/.

[5]CORNET, see https://cornet.wireless.vt.edu/.

[6]ORBIT, see http://www.orbit-lab.org.

[7]NiTos, see https://nitlab.inf.uth.gr/NITlab/nitos.

[8]w-iLab.t:
see http://doc.ilabt.imec.be/ilabt-documentation/wilabfacility.html.

B. Room characteristics

The R2lab platform sits in an insulated anechoic chamber located in a basement of a building at Inria, Sophia Antipolis, France. Figure 1 shows a snapshot from inside the room. It hosts thirty-seven PC nodes on the ceiling scattered on a fixed grid; more than half of the nodes feature an SDR board, of various kinds. In addition, two remotely controllable commercial phones are available.

Figure 1: R2lab room

The room size is about 90m^2, roughly 11m x 8m, although its shape is not a plain rectangle, as shown on Figure 2. This picture shows the ground plan layout of the nodes that are arranged in a grid with about 1.0m (vertical) and 1.15m (horizontal) of distance between them, except for the two pillars in the room. Such an environment allows running various scenarios with wireless nodes that can be with line of sight or not between each other.

Figure 2: R2lab topology

It is insulated from the outside electromagnetic conditions by a Faraday cage, that is is made of 0.10 mm thick copper foils, which has high attenuation properties for the electromagnetic field (up to 120 dB), see Table I.

RF absorbers are needed to prevent high level of reflections on the copper foils. We use IMOCELL HPP 20 ultra broadband pyramidal absorbers, made of carbon loaded polyurethane foam, and specially designed for microwave applications. Their performance are shown in Table II.

Faraday cage	**Frequency**		
	20MHz to 1GHz	1 - 3 GHz	3 - 5GHz
Attenuation	>100dB	> 90dB	> 80dB

Table I: Faraday cage shielding performance

HPP 20 absorber	**Frequency in GHz**			
	1	3	6	>10
Reflectivity (dB)	-30	-38	-45	-50

Table II: Typical reflectivity (dB) at nominal incidence

C. Icarus nodes

The 37 wireless nodes are Icarus[9] computers provided by NITlab[10] with the following features:

- CPU Intel® Core™ i7-2600, 8M Cache at 3.40 GHz
- 8GB DDR3
- 240 GB Solid State Drive
- 3 Gigabit Ethernet interfaces: one for remote node power and reset management, one for control used by the testbed management framework for providing access, and one for data, dedicated to experimentation, e.g, to create wired link or to connect to an SDR device such as USRP2 or N210.
- 2 Wi-Fi MIMO NICs dedicated to experimentation: one Atheros 802.11 93xx a/b/g/n and one Intel 5300. Each card is connected to 3 dual-band 5dBi antennas, operating on both 2.4GHz and 5GHz. Antennas are spaced of 2.8cm, which corresponds to half the wavelength at 5GHz, see photo at Figure 3.

Figure 3: Wi-Fi dual-band antennas for a node

[9]Icarus node: https://nitlab.inf.uth.gr/NITlab/.
[10]NITlab:
https://nitlab.inf.uth.gr/NITlab/hardware/wireless-nodes/icarus-nodes.

To control and monitor each Icarus node, we use the NITlab's Chassis Manager Card (called CM card); this device embarks a tiny web server that can serve http requests to power on/off and reset the motherboard, or one attached USB device. This latter feature allows to physically reset a USRP or LTE dongle without the assistance of the Icarus node itself.

D. SDR devices

About half of Icarus nodes are attached to an SDR device. R2lab currently supports USRP1, USRP2, N210, B210 and X310 ETTUS[11] devices and also LimeSDR from Lime Microsystems[12], see Table III.

Device type	USRP1	USRP2 & N210	B210	X310	LimeSDR
# of	4	8	6	1	2

Table III: Available SDR devices

Each N210 device includes a SBX-40 USRP daughterboard (400 MHz - 4.4 GHz, 40 MHz BW), whereas the X310 board includes two SBX-120 USRP daughterboards (400 MHz - 4.4 GHz, 120 MHz BW).

Note that a subset of USRP B210 boards are connected to LTE Band 7 duplexers. This is useful when running OpenAir-Interface eNodeB or UE software as the distance between the RX and TX antenna SMA connectors on the B210 board is such that the TX antenna generates too much interference to the RX channel.

Finally, two out of the 37 nodes host a Huawei 3372 USB dongle with the following characteristics: FDD LTE with DL:150 Mbps and UL:50 Mbps, BW: 20 MHz. Bands: 800 / 900 / 1800 / 2100 / 2600 MHz.

E. Commercial Phones

Two commercial phones (Nexus 5 and iPhone 6s) are currently available right inside the chamber. Each one is connected through USB to a computer (that also sits in the room) including convenience helpers to manage the phone remotely. These computers can be accessed through ssh, or via remote access software based on VNC.

III. R2lab Software

This section gives a brief description of the software deployed in order to operate the R2lab testbed and to expose it to experimenters. Many different aspects need to be taken into account here, but a fair split of the desired functionalities would be as follows.

A. Membership and Reservations

In our case, the reservation model is wholesale; a user has to reserve the whole testbed, i.e., one cannot reserve only a portion. This is on purpose, and a way to ensure that two concurrent experiments will not be able to contaminate each other's radio spectrum.

[11]Ettus Research: https://www.ettus.com/.

[12]Lime Microsystems: http://www.limemicro.com/.

The first and most obvious features are the ones that deal with keeping track of the various computing resources and of users. In other words, we need a data model, and related API, for first-citizen objects like nodes, users, reservations.

For fulfilling this need we have decided to use the PlanetLab API component, which we happen to be very familiar with, and that we found to be able to cope with the job with zero modification. The only twist was to define a single dummy meta-node to materialize reservations. This choice also came with the additional benefit of providing a federation-ready software layer, which came in handy, as R2lab is a member of the FIT testbed federation.

B. Access network

The Icarus nodes obviously do not have direct Internet connectivity, they can be reached through a single point of entry at a gateway box named `faraday.inria.fr`. Registering as a member of R2lab grants access to that gateway, from where `ssh` access to the nodes is possible during a reserved timeslot.

C. Low-level nodes management

From that gateway, we also need some low-level tools to physically manage nodes; in this category we can mention features like: turning nodes on, or off, or reset their motherboard; monitoring their status; loading or storing images on or from the hard disk drive; this is the purpose of the `rhubarbe` software tool[9], extensively based on python's 'asyncio' library, and that takes advantage of Emulab's image management tool named `frisbee`[10].

We provide prebuilt images that contain the Intel CSI tool [11] and Atheros CSI tool [12], that allow to collect the channel state information from both Wi-Fi NICs present on the nodes. We also provide images that are ready for GNU Radio [13] and OpenAirterface [14].

D. Web site

https://r2lab.inria.fr is a django application that offers the usual set of features for such a testbed, in terms of detailed hardware description and several tutorials. Authenticated users can manage their reservations, and see a live status of all nodes in terms of power, reachability, loaded image, and similar. Figure 4 shows an example captured on a session that involves only two nodes running different Linux distributions.

IV. Experimenter's tool

With all the above, experimenters can register for an account, reserve timeslots either interactively or programmatically, monitor and setup the testbed, so in essence they already have enough to use R2lab.

However, managing potentially tens of nodes through a multi-hop ssh access gets tricky when performance matters. In addition, an average reservation on R2lab only lasts one or two hours, and so it is desirable to allow for some powerful and yet effective means to script for experiments. This is obviously even more stringently desirable in the perspective of reproducible research.

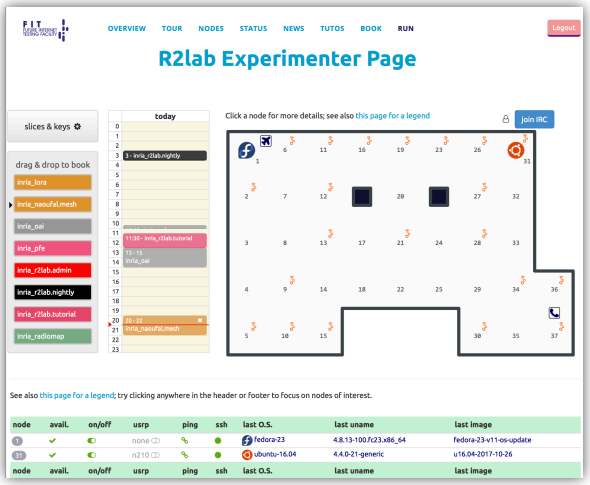

Figure 4: Live status in the 'r2lab.inria.fr' website

For these reasons we also offer a tool named `nepi-ng`, that is particularly well suited for controlling an R2lab experiment right from the experimenter's laptop. Actually `nepi-ng` is not specific to R2lab; its capabilities currently only rely on ssh, and so it could perfectly cope with other testbeds like PlanetLab, or even be used for daily sysadmin-oriented tasks.

In the `nepi-ng`'s declarative paradigm, an experiment is described as a set of nodes and jobs. Nodes are python objects that materialize ssh connections, and can be nested to create multi-hops. Each job is attached to one node, and materializes a sequence of remote actions, like running a remote command, or running a script whose source is local, or copying files back and forth. The ability to remotely copy a local script is a convenient way to defer low-level details to, say, shell scripts, while the overall experimental logic remains controlled by `nepi-ng`.

Jobs are meshed inside a dependency graph, which expresses chronological synchronization. As an illustration of these basic mechanisms, figure 5 was automatically derived from our tutorial *My first nepi-ng script*[13].

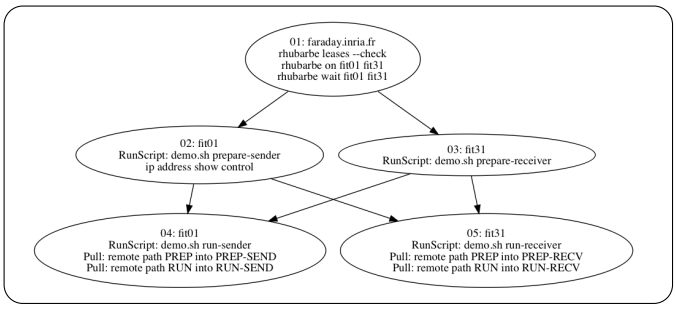

Figure 5: Sketch of a simple `nepi-ng` script

Figure 6 shows an extract of the corresponding code, that in particular illustrates how the chronological dependencies is expressed through the `required` relationship.

```
job_warmup = SshJob(
    node = gateway,
    command = [
        Run("rhubarbe leases --check"),
        Run("rhubarbe on", nodename1, nodename2),
        Run("rhubarbe wait", nodename1, nodename2),
    ]
)

job_prep_send = SshJob(
    node = node1,
    command = [
        RunScript("demo.sh", "prepare-sender"),
        Run("ip address show control"),
    ],
    required = job_warmup,
)
```

Figure 6: Extract of the same `nepi-ng` script

`nepi-ng` design was greatly inspired by `d3.js`, a well-known visualization library. Many such visualization tools, like `gnuplot` or `matplotlib` to name just a couple, propose an encapsulating approach, in the sense that from low-level graphical objects like lines and text, they build abstractions like donut or bar charts. The approach taken by `d3.js` is totally disruptive in this respect, as it directly exposes the underlying low-level objects, and focuses on a framework that promotes one unique workflow for updating visualization as the data evolves.

Although addressing a completely different area, `nepi-ng` adopts a similar point of view; rather than offering abstractions for controlling the whole variety of underlying hardware, it offers a convenient framework for orchestrating the overall logic for an experiment, but the gory details, typically of the actual devices configuration, is left to the experimenter who can use any suitable software for reaching her objectives.

As a result, `nepi-ng`, which is implemented as two very small python libraries, amounts to a total of a mere 3 KLOC. It makes here again extensive usage of python's coroutines-based asynchronous programming paradigm. Thanks to this, it is possible to control all 37 R2lab nodes in a single-threaded application, that is optimal in terms of ssh connections, in the sense that at most exactly one connection is created to every node actually involved.

Finally, it is worth noting that no prior exposition to asynchronous programming is required to use this paradigm, since a script is essentially purely declarative, so the details of using coroutines and an event loop are hidden to a beginner user.

V. A SAMPLE NOTEBOOK

For the sake of illustrating a methodological approach to publish experiment results, we have defined a dummy experiment, that consists of measuring, for all couples of nodes (a, b) in the testbed, the power received by b when a is sending. It is a dummy experiment in the sense that we do not intend to interpret the result in any significant way, although it does help

[13] https://github/parmentelat/r2lab-demos/my-first-nepi-ng-script

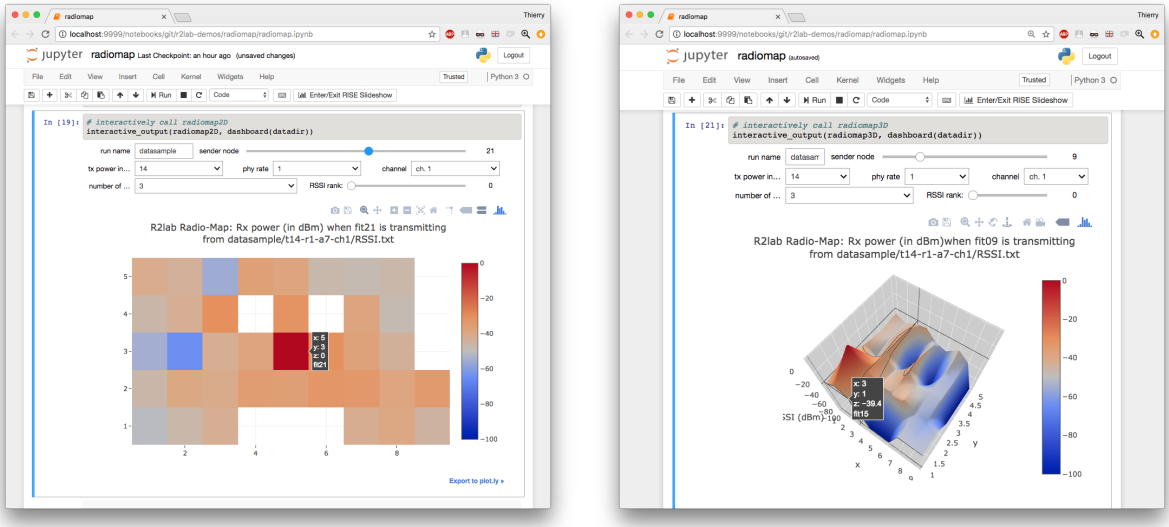

Figure 7: A glimpse at the Jupyter notebook for R2lab radiomap
click the images to run on `mybinder.org`

to understand the disposition of the room and its peculiarities, particularly the impact of the pillars and of blind corners.

Jupyter notebooks are hybrid documents that mix formatted text and executable code. In this section, we are exploring the capabilities offered by using Jupyter notebooks, in conjunction with publicly available infrastructures like `github.com` and `mybinder.org`, as a step towards runnable papers.

Again the experiment in itself is designed to be straightforward, as our focus here is purely methodological; we want to exhibit one possible way of publishing this dummy experiment, so as to maximize reproducibility.

Scenario Description

As usual, the experiment can be thought of as two successive phases.

Data collection consists of generating traffic between a Tx and Rx node and measuring the RSSI at the level of the receivers. The procedure used for collecting the measurements is as follows:

1) Create an ad-hoc network at a given frequency.
2) Generate traffic using `ping` with different parameters (see Table. IV) from a Tx node while all the other nodes are listening. The nodes take turn in sending the traffic.
3) Capture the incoming packets with `tcpdump` on the monitor interface of each receiving node.
4) Process the dump files in the nodes and collect the generated RSSI to the experimenter's local machine.

Data processing can then occur in order to provide higher level quantitative feedback as a result of the experiment, as well as visualization of some representative outcome of the experiment. In the case of this simplified experiment we will simply provide a small visualization tool, that allows a user to select a specific sender node, and that displays the perceived power from all other nodes.

Of course, data collection requires actual physical access to the experimental resources, while data processing can be carried out in the void. Also, several environmental parameters can impact the measurements (see below). Collecting data for a large number of combinations of these environmental parameters can require a vast amount of time, and so it is perfectly reasonable to perform data collection incrementally in several complementary measurement campaigns.

The notebook

We provide a Jupyter notebook that can be used to perform either of the stages described above.

Data collection: the environmental parameters of the experiment protocol that can be controlled in the notebook are listed in tableIV.

NIC Driver	**ath9k**	**iwlwifi**
Transmit Power (in dBm)	5 to 14	0 to 15
Number of antennas	1, 2, 3	3
Physical Tx Rate	1 to 54 Mbps	
Wi-Fi Channel	1-11, 36, 40, 44, 48, 52, 56	

Table IV: Controllable parameters from the notebook

The data collection code leverages both `nepi-ng` and `rhubarbe`. Data gathered along the various runs accumulates in a simple directory structure, so that data collection is trivially incremental.

Data processing: once performed in plain python code, data can be plotted interactively right inside the notebook using various visual tools, in the sense that the reader can select either the environmental parameters that he is interested in - when available of course, or select a sender node, and in the case of 3D diagrams, can navigate inside the figure space. These possibilities naturally reach way beyond what is usually feasible from a static printable paper.

The notebook can be downloaded as part of a `git` repository[14], and run from anyone's computer. Clickable Figure 7 will lead you directly to a pre-deployed instance of that notebook, hosted in the `mybinder.org` public infrastructure, as a best-effort attempt to make it usable as seamlessly as possible.

The experimenter has the ability to study the impact of parameters on the transmission range as well as on the channel response. The radio maps can be used as a prospection tool for researchers to select the most convenient nodes to be used for their experimentation scenarios by taking advantage of the R2lab's layout, for example when various types of communication scenarios are required, e.g., with line-of-sight, non-line-of-sight, near-line-of-sight. Additionally, radio maps can be used as a diagnosis tool to identify possible issues with NICs and antennas, as well.

The notebook is part of a larger git repository that contains other required artifacts. This repository comes with some measurements; in a real experience, this would correspond to the data acquired by the author. This means that in a first step, a reader could easily reproduce the visualizations of the paper from that data, before she can go further and perform her own data acquisition.

Discussion

As an outcome of this exercise, we have learned the following about the advantages and disadvantages of this approach for publishing the details of an experiment.

On the pro side, notebooks are a very effective way to convey, at the same time, ideas and their implementation. The format allows to provide for executable documents, but authors have a great level of flexibility to show the important pieces of the code and to hide less crucial details in code stored separately.

Still on the bright side, the current public offering allows to combine `github.com` and `mybinder.org` and to offer single-click access to a runnable notebook, that completely removes the burden of software installation, at the price of more limited features though.

On the downside however, running the notebook from that public environment is limited to postprocessing, as gaining physical access to R2lab requires ssh credentials. And in any case, whether the notebook is run from that public spot or from your laptop, a reservation needs to be obtained before data collection can be carried out.

Sustainability is the major challenge here. Of course, we cannot assume that `mybinder.org` will be up forever, this is why we rely on this platform only as best-effort mode, essentially to smoothen the adoption process of Jupyter, which besides is gaining increasing popularity in several scientific communities. The notebook in our case, together with its depending code and notes, can be considered as a regular artifact, insofar as its real purpose is to be run locally.

[14]The notebook source from its github repository https://github/parmentelat/r2lab-demos/radiomap/radiomap.ipynb

VI. Conclusion

The combination of R2lab's controllable environment and experiment orchestration tools, together with the Jupyter notebook paradigm, paves the way for a new era in the reproducible research domain. We argue that, even though the technical background is not quite available yet, it should be possible eventually for researchers to submit papers written in a completely redesigned format, as compared to camera-ready paperware that is, fundamentally, similar to what Gutemberg could have printed in the late XV^{th} century. Such a medium would allow to bundle, in a more relevant manner, (a) text that describe ideas, and related mathematical material, (b) environment described either as data or as parameters, (c) code to collect, process and render output data, and (d) the raw data as collected by the authors. Our feeling is that the separation currently made between, on the one hand, material belonging in the (a) family - *"the paper"* - perceived as the first-class citizen, and on the other hand the other (b..d) classes - *"the artifacts"*, is actually wrong in the first place.

We are planning to study, as part of future work, the missing pieces that could in the future allow for papers to be submitted in a form closer to a runnable paper.

Acknowledgments

This work is funded by the French ANR through the "Investments for the Future" Program under grant ANR-11-LABX-0031-01. The R2lab wireless testbed at Inria has been funded by the ANR Equipex FIT 6165.

References

[1] Kim *et al.*, "Enabling iterative development and reproducible evaluation of network protocols," *Computer Networks*, vol. 63, pp. 238–250, 2014.
[2] Tala *et al.*, "Guidelines for the accurate design of empirical studies in wireless networks," in *International Conference on Testbeds and Research Infrastructures*. Springer, 2011, pp. 208–222.
[3] Mahfoudi *et al.*, "Lessons Learned while Trying to Reproduce the OpenRF Experiment," in *Reproducibility'17 - ACM SIGCOMM Reproducibility Workshop*, vol. 41, no. 1, LA, USA, Aug. 2017, pp. 21 – 23.
[4] Reis Fontes *et al.*, "How far can we go? Towards Realistic Software-Defined Wireless Networking Experiments," *The Computer Journal*, Jun. 2017.
[5] Mahfoudi *et al.*, "ORION: Orientation Estimation Using Commodity Wi-Fi," in *ICC Workshop on Advances in Network Localization and Navigation*, Paris, France, May 2017.
[6] ——, "Deploy a 5g network in less than 5 minutes," in *Proceedings of the SIGCOMM Posters and Demos*. ACM, 2017, pp. 113–115.
[7] Busson *et al.*, "A simple method to infer wi-fi conflict graph," in *CoRes ALGOTEL Workshop*, Quiberon, France, May 2017.
[8] Pakzad, "Towards software defined wireless mesh networks," Ph.D. dissertation, The University of Queensland, 2017.
[9] Parmentelat, "Testbed management framework for r2lab," https://github.com/parmentelat/rhubarbe.
[10] Hibler *et al.*, "Fast, scalable disk imaging with frisbee," in *Proceedings of the USENIX Annual Technical Conference*, 2003.
[11] Halperin *et al.*, "Tool release: Gathering 802.11n traces with channel state information," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 53–53, Jan. 2011.
[12] Xie *et al.*, "Precise power delay profiling with commodity wifi," in *Proceedings of ACM MobiCom*, NY, USA, 2015, pp. 53–64.
[13] Blossom, "GNU radio: tools for exploring the radio frequency spectrum," *Linux journal*, vol. 2004, no. 122, p. 4, 2004.
[14] Nikaein *et al.*, "OpenAirInterface: A Flexible Platform for 5G Research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 33–38, Oct. 2014.