How far can we go? Towards Realistic Software-Defined Wireless Networking Experiments

Ramon dos Reis Fontes[1], Mohamed Mahfoudi[2], Walid Dabbous[2], Thierry Turletti[2] and Christian Rothenberg[1]

[1] *University of Campinas (UNICAMP), Campinas, Brazil*
[2] *Université Côte d'Azur, Inria, France*
Email: ramonrf@dca.fee.unicamp.br

Software-Defined Wireless Networking (SDWN) is an emerging approach based on decoupling radio control functions from the radio data plane through programmatic interfaces. Despite diverse ongoing efforts to realize the vision of SDWN, many questions remain open from multiple perspectives such as means to rapid prototype and experiment candidate software solutions applicable to real world deployments. To this end, emulation of SDWN has the potential to boost research and development efforts by re-using existing protocol and application stacks while mimicking the behavior of real wireless networks. In this article, we provide an in-depth discussion on that matter focusing on the Mininet-WiFi emulator design to fill a gap in the experimental platform space. We showcase the applicability of our emulator in an SDN wireless context by illustrating the support of a number of use cases aiming to address the question on how far we can go in realistic SDWN experiments, including comparisons to the results obtained in a wireless testbed. Finally, we discuss the ability to replay packet-level and radio signal traces captured in the real testbed towards a virtual yet realistic emulation environment in support of SDWN research.

Keywords: Mininet-WiFi; Software-Defined Wireless Networks; OpenFlow; IEEE 802.11

Received 30 September 2016; revised 24 January 2017

1. INTRODUCTION

WiFi is ubiquitous and has become an indispensable part of our daily lives. IEEE 802.11x standards are present in almost any portable devices and allow rolling out wireless infrastructures based on access point devices at a relative low cost. While the IEEE 802.11x standards keep steadily evolving, there are still many structural barriers to openness that prevent the wireless infrastructure to adapt, be customized, allow for differentiation, and leverage all wireless capacity around us as consequence of being an infrastructure closed to innovation [1].

It is believed that the concept of Software-Defined Networking (SDN) [2] applied to wireless networks, commonly referred to as Software-Defined Wireless Networking (SDWN) [3, 4], can break down current structural barriers and contribute to a more innovative ecosystem. In spirit of (wired) SDN principles, SDWN decouples control and data planes, enabling the wireless network to become programmable by abstracting the underlying infrastructure from applications and network services that are offered with higher-level Application Programming Interfaces (APIs). As a consequence, wireless network infrastructures are presented with new deployment paths of research ideas around seamless communication over heterogeneous wireless networks [5], high transferring data over wireless medium [6], energy optimization [7], among others [3, 4].

The path towards actual roll-out is not without significant challenges, such as hardware availability, protocol implementations in software, resource constraints, realistic evaluation models, development and testing costs and efforts, and so on. Simulators, emulators and testbeds are common tools and approaches used in experimentally-driven research to evaluate the functionality and performance of networks. Aspects such as scalability, reproducibility and cost-benefit, among others, make the simulation and emulation the most preferred methods [8].

In the context of supporting SDWN research, a number of alternatives have been developed such as DCE/ns-3 [9] and OMNeT++ [10]. However, existing simulation tools face several limitations. For instance, OMNeT++ has its own switch/AP implementation and does not support third-party controllers. DCE/ns-3 uses API-specific glue code for its POSIX and kernel support, which does not cover system calls for all applications. Another common weakness of existing tools is the lack of 802.11 scan mechanisms which are critical for layer-2 handover mechanisms.

Addressing the gaps in SDWN experimentation tools from an emulation standpoint, we propose Mininet-WiFi [11] as an emulator for SDWN that: (*i*) allows the execution of real code with no modification in either kernel or applications, (*ii*) is able to support best of breed open source technologies (e.g., OpenvSwitch, any existing OpenFlow controller), (*iii*) supports the Linux mac80211 framework that allows testing most of the IEEE 802.11 functionality leveraging user space tools (e.g., hostapd and wpa_supplicant), among others. Mininet-WiFi is a fork of the container-based Mininet network emulator [12] extended to support WiFi by adding virtualized WiFi Stations (STAs) and Access Points (APs). Mininet-WiFi inherits all characteristics of Mininet, including the container-based emulation, a category of network emulation which allows the creation of small emulated networks on commodity hardware through the use of kernel level virtualization techniques [13]. Like real testbeds, Mininet-WiFi runs real code (e.g., OS kernel, external SDN controllers applications) with real traffic (pcap traces or actual physical, 802.11-enabled devices). Like simulators, it supports arbitrary topologies and well-controlled yet dynamic, real-time environments at low cost.

In this article, we pose the question on how far can we go in performing realistic SDWN experiments acknowledging that there is no tool that fits all scenarios and that the research and deployment path is dominated by fundamental trade-offs intrinsic to each approach (i.e., testbed, simulation, emulation). More specifically, we try to answer the question on delivering realistic SDWN experiments based on the design choices and current capabilities of Mininet-WiFi, which include the support of (user-defined) mobility models, wireless channel propagation models, the ability to integrate with physical devices (e.g., smartphones) [14], among other researcher-friendly features. To assess our claims of Mininet-WiFi delivering a sweet spot for realistic and reproducible SDWN experimentation, we reproduce related work experiments, we present a novel use case based on SSID-based forwarding, and we carry extensive evaluation work to validate the propagation models with traces and results from the R2lab testbed.[3] We also present the ability of replaying network conditions from the real world by taking into account observed variations in terms of bandwidth, latency, and packet loss.

The remainder of this article is organized as follows. The next section provides a primer on SDWN by introducing relevant concepts to understand the applicability and relevance of experimental tools. Section 3 discusses simulators, emulators and testbeds as candidate platforms to support experimentation in wireless networks. Section 4 presents the Mininet-WiFi emulator and the main features allowing realistic SDWN experiments. Section 5 presents the experimental outcomes of a real testbed compared to the results obtained with Mininet-WiFi and the DCE/ns-3. Section 6 showcases an approach to replay captured traces supported by Mininet-WiFi. Finally, Section 7 concludes with some final remarks and avenues for future work.

[3]http://r2lab.inria.fr

2. PRIMER ON SOFTWARE-DEFINED WIRELESS NETWORKING

Software-Defined Wireless Networking (SDWN) [3, 4] is based on providing programmatic centralized control of the network outside wireless-enabled devices (Access Points - APs) which enforce the data plane instructions (i.e.. policy decisions) received from the controllers. The principles of SDWN are similar to those of Software-Defined Networking (SDN) [2], i.e., a networking approach based on a programmatic separation of the control plane (aka. Network OS) from the data plane (aka. forwarding- or data-plane). The software-defined approach allows administrators to specify the behavior of the network in a logically centralized manner and at a higher-level through APIs provided by the controller platform that implements southbound interfaces to the forwarding devices –the OpenFlow protocol [15] being the most popular southbound interface (as illustrated

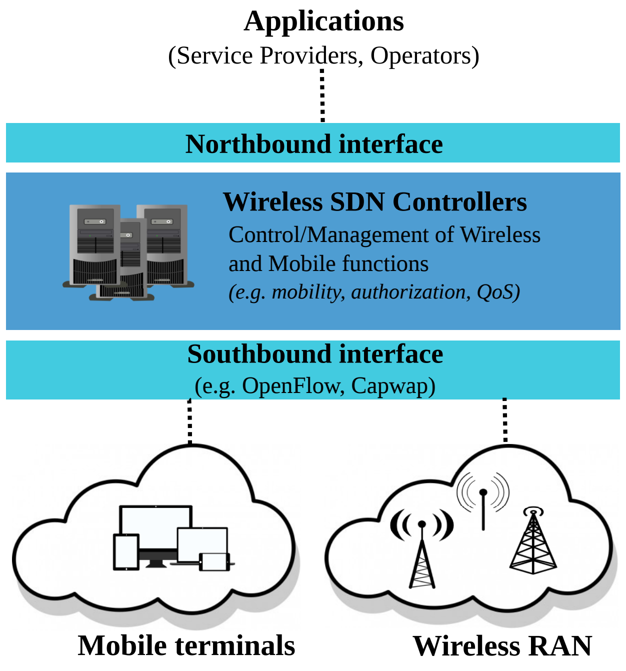


FIGURE 1. Generalized and Simplified Software-Defined Wireless Networking Architecture.

in Figure 1) but not the only one, CAPWAP [16], FORCES [17], or NETCONF [18] are also candidate protocols in scope.

SDWN has become an emerging and significant research branch of SDN, mainly driven by the increased interest of mobile network operators [19, 20] and the synergies with Network Function Virtualisation (NFV) [21]. The separation between control and data planes has existed in the wireless domain prior to SDN and OpenFlow. Indeed, IETF standardized both LWAPP (*Lightweight Access Point Protocol*) RFC5412 [22] and CAPWAP (*Control And Provisioning of Wireless Access Points*) RFC4564 [16]. several years ago. Many enterprise WLAN management systems use protocols such as LWAPP and CAPWAP to manage their wireless network systems. LWAPP defines the control messaging for setup and path authentication and run-time operations whereas CAPWAP is based on LWAPP and enables a controller to manage a collection of wireless access points. Within the Open Networking Foundation (ONF), the Wireless & Mobile Working Group (WMWG) is defining a common ground architectural framework along the necessary OpenFlow protocol extensions or enhancements to realize the identified use cases while leveraging related work in other Standards Developing Organizations (SDOs) such as 3GPP, IEEE, NGMN, ITU, ETSI, IETF, etc. As per [23], over 15 use cases have been identified, ranging from flexible and scalable packet core to unified access networks, encompassing different elements of OpenFlow-based or OpenFlow-oriented wireless and mobile network domains.

SDWN research in academia has bloomed over the last years (refer to [3] for a comprehensive survey), including proposals such as OpenRoads [1], Odin [24], OpenRF [25], Ethanol [26], among others. Architectures such as CloudMac [27] and Chandelle [28] use CAPWAP in their proposals. CloudMac describes current WLAN management protocols such as CAPWAP, as a protocol hard to extend with new functionalities since CAPWAP AP controllers are mostly proprietary systems. Chandelle, instead, proposes a smooth and fast Wi-Fi roaming with SDN/OpenFlow but suffers from integration challenges with traditional switches and CAPWAP. One issue with CAPWAP is that it tries to solve both control and provisioning/management at once, opening the door for conflicts due to the split of roles, e.g., consider the management layer hazards of an AP receiving a CAPWAP firmware update command.

Identified benefits of integrating WLAN and OpenFlow [23] are commonly related to centralized management and monitoring, unified policies, increased programmability and fine-grained control of WLAN functions. Taking into account these benefits and the limitations associated to CAPWAP –arguably the most advanced (closed) solution today for centralizing wireless networks management prior to SDN– some questions are inevitable: *“Is CAPWAP in scope of SDWN?”*, *“How to improve the OpenFlow specification to support centralized management of wireless networks?”*, *“Are radical new designs required?”* or *“How much can be leveraged from currently deployed infrastructures?”*. Although these questions are still majorly open, some noteworthy initials steps are undergoing. There is IETF work[29] on extending the CAPWAP protocol to support separate termination points for management, control and data plane tunnels, and the definition of the role of an AP and its controller(s) in RFC7494 [30]. In spirit of the longer-term mission and deliverables of the ONF WMWG, the OpenFlow protocol specification version 1.5 (section B.6.3 [31]) includes a revised behaviour when sending packets out to incoming ports, which was a longstanding issue when mapping wireless interfaces to switch ports.

In addition to CAPWAP-based products, there are multiple proprietary solutions (e.g., Aerohive, Aruba, Cisco HDX, Meraki, Ruckus) based on external controllers to manage a collection of APs. These commercial solutions introduce a number of extensions to standardized protocols or define their own APIs between the controller and APs, presenting differences in the refactoring of control and data plane functions in addition to a series of proprietary radio resource enhancements. While arguably all these solutions have proven to work well at scale, their cost is often prohibitive for many deployments and raise concerns due to their closely integrated nature, the consequent vendor lock-in, and the inability for in-house or third-party innovations.

As today, the most realistic way to experiment with WiFi and OpenFlow together is using open source firmware and OS solutions like OpenWRT that allows turning commodity wireless routers into OpenFlow-enabled switches. However, like any real testbed, such approach is subject to challenges related to the costs and scale of the experiments, in addition to reproducibility constraints as well as high setup times. Wireless SDN simulators and emulators, on the other hand, are interesting alternatives allowing to work with multiple devices (e.g. APs and STAs) at reasonable scale in experimenter-defined environments. Before detailing our proposed research path based on the Mininet-WiFi emulator, we next briefly survey the main available SDWN research platforms.

3. WIRELESS SIMULATORS, EMULATORS AND TESTBEDS

This section provides an overview of notable simulators, emulators and testbeds identifying their main characteristics towards a better understanding of the trade-offs when aiming at supporting realistic SDWN experimentation.

As is well-known but commonly underestimated or misjudged when choosing an experimental platform

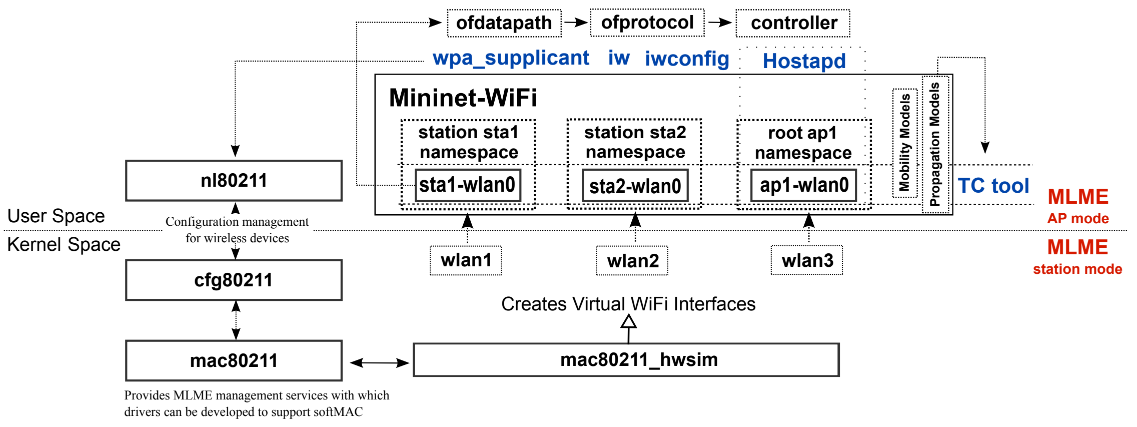


FIGURE 2. Main components of Mininet-WiFi.

TABLE 1. Ranking of Simulators, Emulators and Testbeds (adapted from [32])

Characteristic	Simulators	Emulators	Testbeds
Total Cost	●○○	●○○	●●●
Overall Fidelity	●○○	●●○	●●●
Replay Real Traces	●●○	●●○	●●●
Real Applications	●○○	●●●	●●●
Traffic Realism	●○○	●●●	●●●
Timing Realism	●●●	●●○	●●●
Scalability	●●●	●●○	●○○
Maintainability	●●●	●●●	●○○
Flexibility	●●●	●●●	●○○
Replication	●●●	●●●	●○○
Isolation	●●●	●●○	●●●

to support research efforts, each environment excels in some aspects but is subject to certain limitations and/or constraints, as depicted in Figure 3. While the exact quantification of each characteristic and the degree of realism ultimately depend on the accuracy of the model implemented in each specific tool among other platform aspects that may affect each feature, Table 1 (adapted from [32]) aims at illustrating the main strengths and shortcomings typically common to each type of experimentation approach as a first guide to choose the best type tool for a given set of research goals and constraints.

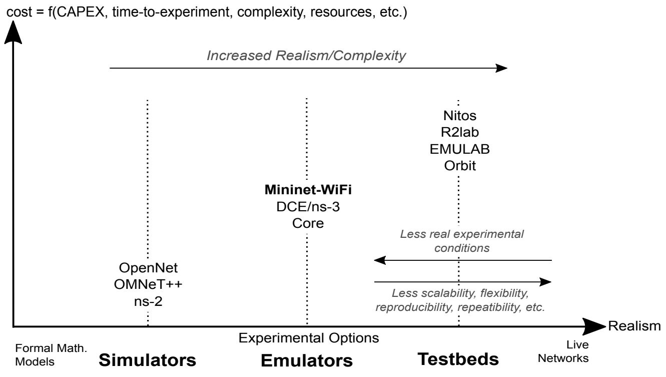

cost = f(CAPEX, time-to-experiment, complexity, resources, etc.)
Increased Realism/Complexity
Nitos
R2lab
EMULAB
Orbit
Mininet-WiFi
DCE/ns-3
Core
Less real experimental conditions
OpenNet
OMNeT++
ns-2
Less scalability, flexibility, reproducibility, repeatibility, etc.
Realism
Experimental Options
Formal Math. Models
Simulators
Emulators
Testbeds
Live Networks

FIGURE 3. Overview of related work and trade-offs of different wireless experimental platforms.

Turning now the attention to specific wireless simulators and emulators, Table 2 compares a number of features including *Type* (e.g., Simulator/Emulator, Open/Close source), *Programming Language* (which language the solution is written) and finally *Supported Protocols* (relevant protocols available by default), *Last Activity* (i.e., recent updates). Similarly, Table 3 attempts to categorize and compare relevant wireless testbeds considering different criteria.

As we can see, there are a couple of alternatives for OpenFlow-based SDWN experimentation such as DCE/ns-3, OpenNet, OMNeT++, and the Estinet and Nitos testbeds. Under the emulation/simulation space, OpenNet [34] highlights as recent work (arguably closest to Mininet-WiFi) on combining DCE/ns-3 and the Mininet SDN emulator [12] to provide rich SDWN experimentation features by allowing the execution of external controllers and real applications at the cost of a strongly coupled solution. In addition, OpenNet does not provide high-level abstraction APIs for wireless links nor emulation of wireless nodes (e.g., access points and stations are not equipped with wireless network interfaces), and neither mechanisms to select new access points before disconnection of current link to further shorten handover latency.

As today, there is few information regarding Estinet,

TABLE 2. Comparison between Mininet-WiFi and Wireless Simulators & Emulators

Software	**Type**	**Source Type**	**Programming Language**	**Supported Protocols**	**Last Activity**
Mininet-WiFi [11]	Emulator	Open	Python	Any (L3 - L7) IEEE 802.11, 802.3, OpenFlow	2016
DCE/ns-3 [9]	Emulator	Open	C++, Python	IEEE 802.11, LTE, OpenFlow	2016
Core[33]	Emulator	Open	several different languages	IEEE 802.2, 802.11	2015
OpenNet [34]	Simulator/ Emulator	Open	C++, Python	IEEE 802.11, LTE, OpenFlow	2016
OMNeT++ [10]	Simulator	Open	C++	IEEE 802.11, OpenFlow	2016
Estinet [35]	Simulator	Proprietary	?	IEEE 802.3, 802.11, OpenFlow	2015
ns-2 [36]	Simulator	Open	C++, TLC	IEEE 802.11, LTE	2012

TABLE 3. Comparative table between Mininet-WiFi and testbeds

Software	**Availability**	**Mobility Support**	**Supported Protocols**	**Last activity**
Mininet-WiFi [11]	Public	Yes	Any (L3 - L7) IEEE 802.11, 802.3, OpenFlow	2016
Nitos[37]	Public	No	IEEE 802.11, WiMAX, LTE, OpenFlow	2016
R2lab[38]	Public	No	IEEE 802.11, LTE	2016
EMULAB [39]	Public	No	IEEE 802.11	2016
Orbit[40]	Public	No	IEEE 802.11, WiMAX, LTE Bluetooth, ZigBee	2016

which according to the developers can be used for many different scenarios, including SDN. Being a proprietary solution and due to its testbed nature, the availability to the wider research community is limited. Nitos, in turn, supports four OpenFlow switches and allows users the possibility to conduct experiments in indoor and outdoor environments.

Table 4 compares Mininet-WiFi and DCE/ns-3.[4] In general, in contrast to Mininet-WiFi, DCE/ns-3 does not incorporate real-world network stacks yet and might not support execution of unmodified applications and/or without kernel modification. Another important weakness of DCE/ns-3 is about the development and/or improvements for IEEE 802.11. DCE/ns-3 depends on the development of new models while Mininet-WiFi relies on the mac80211 wireless stack of the Linux kernel. On the other hand, DCE/ns-3 supports greater variety of mobility and propagation models and also Long-Term Evolution (LTE) and because of this, DCE/ns-3 has been very useful during the development of Mininet-WiFi, serving as a basis for reference implementation.

[4]Information about DCE/ns-3 release 1.8 were obtained from the online manual: https://www.nsnam.org/docs/dce/manual/ns-3-dce-manual.pdf

	Mininet-WiFi (v1.9)	**DCE/ns-3** (v1.8)
sysctl, *ifconfig*, *route*	✓	✗
IPv6 address config.	✓	✗
full *POSIX*	✓	✗
poll implementation	✓	✗
Quagga routing stack	✓	✓
extensive test	✓	✗
real time scheduler	✗	✓
mobility models	●●○	●●●
propagation models	●○○	●●●
supported technologies	WiFi	LTE/WiFi

TABLE 4. Comparison between Mininet-WiFi and DCE/ns-3

4. MININET-WIFI

Mininet-WiFi is a fork of Mininet [12] extended with the required classes to add wireless channel emulation, node mobility, and support of 802.11 through *SoftMac*, a MAC layer that provides a flexible environment for experimenting with MAC protocols. The main components of the Mininet-WiFi architecture are illustrated in Figure 2. In the kernel-space, the module *mac80211_hwsim* is responsible for creating virtual Wi-Fi interfaces, enabling the creation of stations and access points. Still in the kernel-space, MLME

(*Media Access Control Sublayer Management Entity*)[5] is realized by stations, while *hostapd* is responsible for the counterpart task at the user-space side in APs.

Mininet-WiFi relies on a couple of standard Linux utilities such as *iw*, *iwconfig*, and *wpa_supplicant*. The first two tools are used for interface configuration and for getting information from wireless interfaces while the latter is used with *Hostapd* in order to support WPA (*Wi-Fi Protected Access*), among other tasks. Both *infrastructure* and *ad-hoc* networks are supported. Another fundamental utility to realize the emulation of the wireless channel is *TC* (*Traffic Control*),[6] a user-space utility program used to configure the Linux kernel packet scheduler to control packet rate, delay, latency, and loss. TC applies these attributes in virtual interfaces of stations and APs, allowing Mininet-WiFi to replicate with high fidelity the actual packet behavior as observed in the real world. The *mobility* and *propagation models* do not require any kernel modification or changes in the applications. More details about both mobility and propagation models are discussed below.

4.1. Mobility

Current mobility models supported by Mininet-WiFi include: *Random-Walk*, *Truncated-Levy Walk*, *Random-Direction*, *Random-Waypoint*, *Gauss-Markov*, *Reference-Point* and *Time-Variant Community*. In addition to these readily available model, the user is able to determine all positions that a node have to pass through and also its speed. This is important, because the user may have total control over the motion of the nodes. Basically, the mobility is defined by the Equation 1:

$$\begin{aligned} intervalT &= endTime - startTime \\ pos_x, pos_y, pos_z &= initPos_x, initPos_y, initPos_z \\ fac_x &= (finalPos_x - initPos_x)/intervalT \\ fac_y &= (finalPos_y - initPos_y)/intervalT \\ fac_z &= (finalPos_z - initPos_z)/intervalT \\ pos_x, pos_y, pos_z &= pos_x + fac_x, pos_y + fac_y, pos_z + fac_z \end{aligned} \tag{1}$$

First, we determine the interval time (`intervalT`) that the node will move to and them we calculate the motion (that includes the speed of the node) based on both initial (`initPos`) and final position (`finalPos`) previously defined by the user for getting what we call by motion factor (`fac`). Finally, we take both initial position and motion factor in order to calculate the next position of the node (defined and updated by `pos`) until the final position.

[5]Some of the functions performed by MLME are authentication, association, sending and receiving *beacons*, etc.

[6]http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html

4.2. Propagation Models

In the current version, Mininet-WiFi supports *Free-Space*, *Log-Distance*, *Two-Ray Ground*, and *International Telecommunication Union (ITU)* propagation models. Other propagation models can be easily implemented by extending a single class for propagation models. Propagation models calculate the power of the signal received by station (RSSI) and translate them into “equivalent” network attributes in practice, like the maximum supported rate or the expected packet loss. Equation 2 shows how the RSSI is calculated.

$$\begin{aligned} PathLoss &= PropagationModelFormula \\ signalStrength &= pT + gT + gR - PathLoss \end{aligned} \tag{2}$$

First, the user must set the propagation model (defined by `PropagationModelFormula`) to be used (free-space is defined by default if no propagation model is set) in order to calculate the path loss (`PathLoss`). Diverse parameters are considered in the propagation models' formula, such as distance between transmitter and receiver, frequency, system loss, etc. Then, the RSSI (`signalStrength`) is calculated taking into account the transmission power (`pT`), antenna gains of transmitter (`gT`) and receiver (`gR`) and the `Path Loss`.

The RSSI is used to calculate the maximum supported rate during a communication between two nodes. We first rely on the technical specification of the physical devices (e.g., commercial wireless NIC) to define the `custombw` thresholds. The link bandwidth depends on the actual signal strength (RSSI) and the maximum supported rate for a specific RSSI. Finally, taking into account the distance between the transmitter and receiver, the rate is calculated through Equation 3.

$$rate = custombw * (1.1^{-dist}) \tag{3}$$

A similar approach is used to calculate packet loss. However, as there is no specification for actual packet loss, we only take into account the distance between transmitter and receiver as depicted in Equation 4.

$$loss = (dist * 0.2)/100 \tag{4}$$

Equations 3 and 4 are based on generic approaches that by default try to mimic the behavior presented by R2lab. In the future, we plan to further improve the models by supporting *minstrel*,[7] a mac80211 rate control algorithm, and *wmediumd*,[8] related work that provides enhancements on the wireless medium for mac80211_hwsim.

4.3. Scenario Diversity

As previously discussed, Mininet-WiFi supports both infrastructure and ad-hoc networks modes, i.e., it supports APs where stations may connect to and

[7]https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel

[8]https://github.com/bcopeland/wmediumd

stations are also able to directly connect to each other. Both infrastructure and ad-hoc modes can coexist in the same topology/experiment, extending the variety of possible experimentation scenarios. As an example of ad-hoc networks, Mininet-WiFi supports VANET (Vehicular Ad Hoc Networks) scenarios through an integration with SUMO (Simulation of Urban MObility) providing the node mobility patterns. Further features in scope of diverse SDWN scenarios include support of Wi-Fi Direct (or Wi-Fi P2P), a technology developed by the Wi-Fi Alliance to replace the Wi-Fi ad-hoc mode, user-defined control on node mobility to allow limiting the region of motion (even when default mobility models are used), replaying network conditions (see Sec. 6.3); and allowing hybrid environments blending real physical nodes (e.g. smartphones) with the emulated environment as publicly demonstrated [14] and further discussed in the use case experiments (see Sec. 5.3). Further diverse scenarios enabled by the experimenter-friendly features are being driven by the user community.[9]

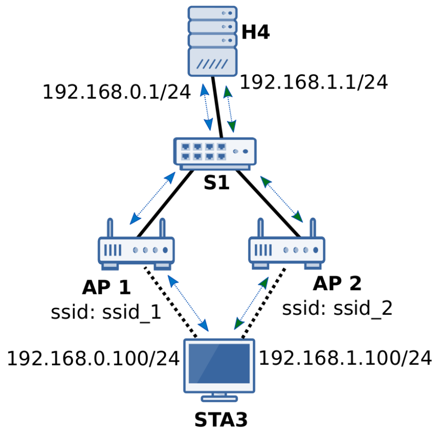
4.4. Limitations

Mininet-WiFi is under continuous development driven by SDWN experiments that define the feature roadmap. As of today, a number of limitations worth to note include: (*i*) lack of mechanisms for channel contention (e.g. CSMA-CA), as well as for MAC layer retransmission and interference; (*ii*) WDS - Wireless Distribution System support;[10] (*iii*) although beacons can be captured using any packet sniffer, changes on the RSSI by the propagation models in place are not included in the packet header yet; and (*iv*) any inherited limitations from Mininet, mainly concerning performance fidelity under high workloads and overall scalability in terms of total amount of nodes interfaces before performance fidelity degrades which ultimately depends on the system platform (e.g., number of cores, memory, etc.).

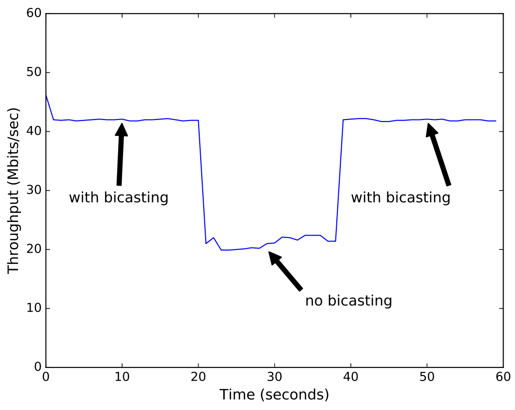
5. USE CASE EXPERIMENTS

This section showcases how Mininet-WiFi facilitates fast prototyping and experimentation of SDWN use cases along its ability to reproduce experiments from the literature. All code and instructions to reproduce the four selected use cases (and further ones) are publicly available in the source code repository[11] and documented in a comprehensive user manual.

[9]https://groups.google.com/forum/#!forum/mininet-wifi-discuss
[10]Required improvements of mac80211_hwsim are underway; https://www.youtube.com/watch?v=FgrESRbG61Y
[11]https://github.com/intrig-unicamp/mininet-wifi/tree/master/demos



(a) Sample Topology



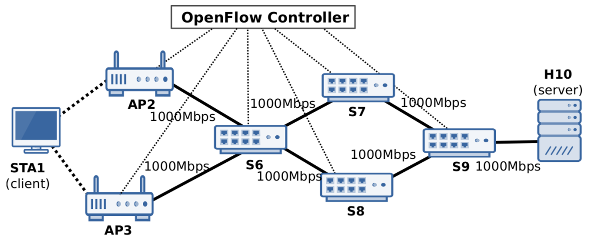
(b) Bandwidth Measurement between STA3 and H4

FIGURE 4. Bicasting over WiFi

5.1. Wireless n-Casting

This use case aims at replicating OpenFlow wireless work [41] on bicasting over multiple wireless links. The original experiment consisted of a video streaming application running in a mobile station, equipped with two WiFi and one WiMax interface, and receiving the packet flow over multiple radios simultaneously (n-casting). The mobile station was attached to the multiple APs using the same SSID and OpenFlow rules were used to duplicate packets in the wired network and re-write the L2/L3 headers at the radio access points. As a result, the quality of experience is improved by compensating packet loss over one wireless link by the duplicated stream received over the alternative links(s).

In the strawman scenario shown in Figure 4(a), we use Iperf to measure the bandwidth between STA3 and H4 during 60s. STA3 has two wireless interfaces and both interfaces are connected to different APs (AP1 and AP2), which are connected to an OpenFlow controller as well as the switch S1. OpenFlow rules ensure that packets are copied and sent through the different paths to the stations. During the first 20s the measured bandwidth is about 40Mbps, which seems coherent with



(a) Topology setup

```
              Client                                      Server
mininet-wifi> sta1 ifstat                        mininet-wifi> h10 ifstat
  sta1-wlan0          sta1-wlan1                   h10-eth0
 KB/s in  KB/s out   KB/s in  KB/s out            KB/s in  KB/s out
 246.04   5047.28    632.20  25951.69             30592.15   886.37
 245.37   5045.90    624.49  25949.27             30592.50   872.26
 246.96   5045.91    625.94  25950.82             30594.53   870.65
 245.09   5047.38    624.06  25949.16             30590.97   870.02
 245.19   5047.35    624.34  25950.53             30594.11   870.48
 247.08   5045.96    625.22  25949.61             30592.60   869.30
 245.21   5047.43    625.50  25950.95             30593.90   868.02
 246.84   5046.30    624.95  25948.97             30591.12   869.98
 244.79   5047.37    625.49  25950.64             30594.98   871.94
 246.59   5045.86    624.63  25950.56             30590.90   867.48
```

(b) Results

FIGURE 5. MultiPath TCP

the 54Mbps limit of each interface in mode g. When the time reaches 20s one wireless interface is disconnected (from AP2) causing a traffic decrease. The bandwidth at the station increases again when the wireless interface is connected back to AP2 at 40s (Figure 4(b)).

5.2. Multipath TCP

We now showcase the use of MPTCP (MultiPath TCP),[12] a TCP extension that allows end-hosts to use multiple paths to maximize network utilization and increase redundancy.

Previous work [42] presented the integration of Mininet and physical wireless devices in order to evaluate the gains of MPTCP in conjunction with SDN path control. We show the ability of Mininet-WiFi to reproduce similar experiments using only a virtual environment, including wireless devices, such as laptops acting as stations and APs. Allowing APs to be controlled by external OpenFlow controllers, different from previous work, both wired switches and the APs are unifiedly managed through the common flow abstraction.

The topology setup (Figure 5a) consists of 1 station (`sta1`) acting as a client and 2 access points (`ap2` and `ap3`), with the distance between *sta1* and *ap2* being 8.06 meters and 1.41 meters between *sta1* and *ap3*; 1 host (`h10`) acting as a server; and 1 OpenFlow controller responsible for the flow control based on a network-wide view. In order to obtain the results presented in Figure 5b, we measure the bandwidth between `sta1` and `h10` using *ifstat* to report network interfaces bandwith for each packet transmission. As expected, the results show that the data is transmitted by the two Wi-Fi

[12]http://www.multipath-tcp.org/

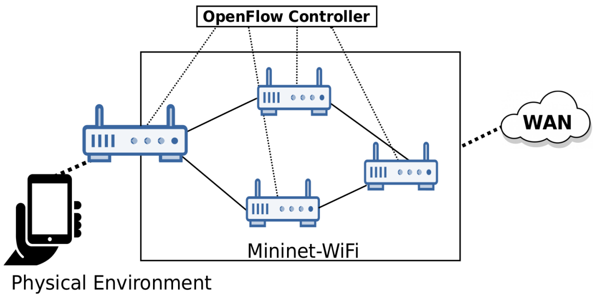


FIGURE 6. Realistic experiment in hybrid physical-virtual environments

interfaces of `sta1` (*sta1-wlan0 and sta1-wlan1*) and the amount of data received by the *sta1-wlan0* interface is less than *sta1-wlan1* (around 1.97Mbps and 5Mbps, respectively), due to the different modes of operation (IEEE 802.11g and IEEE 802.11n) used in each interface. Following the configuration provided by the AP TP-Link TLWR740N[13] chosen for this experiment, theoretically, up to 54Mbps over 802.11g and up to 130Mbps over 802.11n can be provided when the signal is up to -68 dBm. Considering the signal strength measured at the station ≃-34.3 dBm over IEEE 802.11g and ≃-19.1 dBm over IEEE 802.11n, the maximum data rate capacity is consumed.[14] The maximum data rate is calculated by taking into account the rate supported by TP-Link TLWR740N and Equation 3.

5.3. Hybrid Physical-Virtual Environment

This use case experiment was first presented in [14] and features a hybrid physical-virtual environment where real users connect their 802.11-enabled smartphones to interact with the virtualized infrastructure, including nodes forming a mesh subnetwork, and access the global Internet after having its traffic processed through a multi-hop OpenFlow network. Figure 6 illustrates the scenario, where virtual and physical mobile devices are able to interact with each other with the switch behaviour being defined by an OpenFlow Controller.

The OpenFlow controller initially discovers the topology and installs the required L2 flow entries to allow connectivity between the APs. Next, the user connects to AP1's SSID and tries to access any Internet Web page via HTTP. The OpenFlow controller installs one rule to re-write the IP destination address and to re-direct all user's HTTP traffic to a captive portal where the user is expected to authenticate in order to get Internet access and unlock the initial bandwidth limits enforced through OpenFlow 1.3 metering actions. The user can also communicate (e.g. PING) with mobile nodes forming a mesh network.

[13]Mininet-WiFi supports technical specifications of several devices

[14]http://www.tp-link.com.br/products/details/cat-9_TL-WR740N.html#specificationsf

5.4. SSID-based Flow Abstraction

This use case illustrates the ability of fostering SDWN research and experimentation by prototyping and trying out ideas around SSID-based packet forwarding as illustrated in Figure 7. This scenario is similar to the one described in [43] but is novel in the OpenFlow implementation choice. We mimic the case presented by the authors by assigning unique Service Set-Identifier (SSID) for each user (or group of users) and managing all flows through an OpenFlow controller that defines different bandwidth limit on an SSID basis. It is very common for an organization to have multiple SSIDs in their wireless network for various purposes, including: (*i*) to provide different security mechanisms such as WPA2-Enterprise for your employees and an "open" network with a captive portal for guests; (*ii*) to split bandwidth among different types of service; or (*iii*) to reduce costs by reducing the amount of physical access points.

Using multiple SSIDs requires the AP to map each station to a different network connection. Traditionally, this fixed mapping is accomplished through VLAN tagging. In our case, we use the OpenFlow protocol to apply rules based on input/output ports as instances of the SSIDs abstractions. Multiple SSIDs are created in APs and each SSID is linked to separated sub/virtual interfaces. OpenFlow rules defined how traffic is being handled and allowed through different SSIDs. By acting on ports, no changes were required to the OpenFlow protocol in order to support this use case. One drawback in our current implementation is the WiFi NIC limit of 8 sub/virtual interfaces per virtual interface constraining each AP to handle up to 8 different SSIDs whereas commercial WLAN solutions are known to be able to create up to 64 SSIDs per AP.

To the best of our knowledge, this is the first time that traffic based on a specific WLAN attribute (SSID) is defined through OpenFlow rules. The technique is an example that may lay the groundwork for the implementation and evaluation of more sophisticated scenarios using Mininet-WiFi, for instance: (a) Wi-Fi Hotspots for Public Wireless Access [44]; (b) Community Wi-Fi [44]; and (c) virtual Service Providers (vSPs) [45].

6. EXPERIMENTAL VALIDATION

This section delves into experiments to assess the realism of the wireless channel emulation and overall end-to-end system provided by Mininet-WiFi. To this end, we conduct a series of experiments in R2lab,[15] an anechoic wireless testbed that will allow us to compare the results obtained in the physical testbed to the results provided by Mininet-WiFi. In a nutshell, R2lab consists in a set of thirty-seven nodes on the ceiling of a room of approximately 90m^2 distributed in mesh layout to offer an advanced simulation Wi-Fi site (see

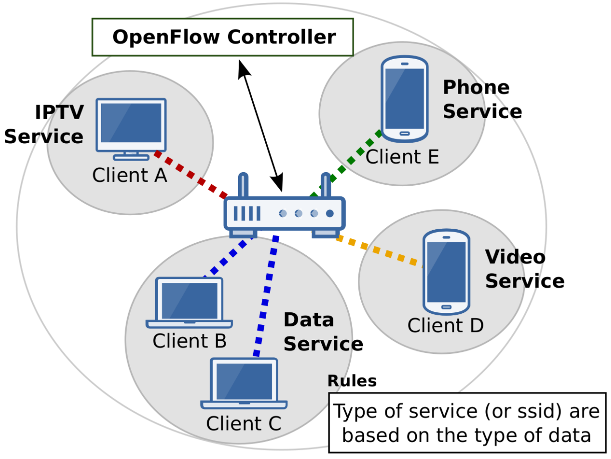
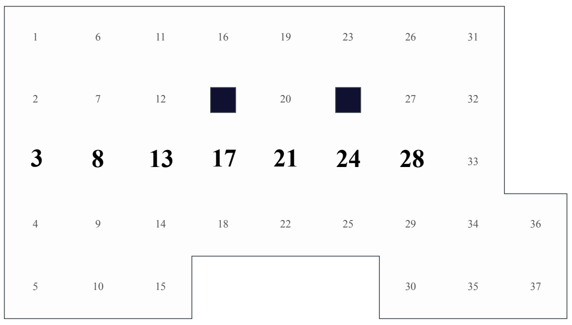
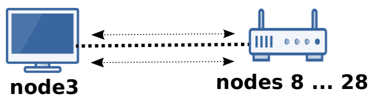


FIGURE 7. Forwarding by SSID



(a) Overview of R2lab's chamber



(b) Sample Topology

FIGURE 8. Experimental validation by using R2lab

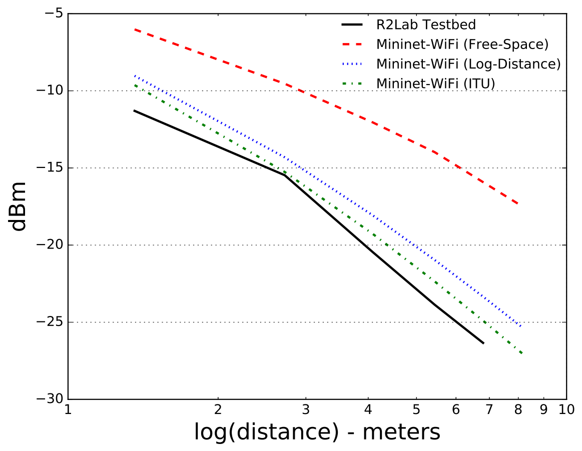


FIGURE 9. Signal Propagation in R2lab and in different indoor propagation models implemented in Mininet-WiFi

[15]http://r2lab.inria.fr/overview.md

Figure 8). Being an RF anechoic chamber built into a screened room, it provides a suitable environment for high-fidelity, reproducible Wi-Fi experimentation.

6.1. Propagation Model

In order to evaluate the propagation models implemented in Mininet-WiFi, this case experiment uses R2lab nodes to obtain the signal received by the stations with varying distances.

6.1.1. Approach

We first choose some nodes (3, 8, 13, 17, 21, 24 and 28) as reference nodes, where node 3 acts as the transmitting station and the other nodes as receivers (APs) located at different distances.[16] All the nodes were configured with specific rate mask limited to 11Mbps, running in mode b, channel 6 and transmission power equal to 15dBm. Then, we used readily available instrumentation (*wireshark*) to record the RSSI between the communicating nodes and compared them with the RSSI values provided by the three supported indoor propagation models in Mininet-WiFi. Parameters for each propagation model include:

- **Free-Space:** system loss = 2
- **Log-Distance:** exponent = 3; system loss = 2
- **ITU:** power loss coefficient = 32; floor penetration loss factor = 0; number of floors = 0

Note that variations in these parameters directly influence the outcome of each propagation model. `System loss` is a factor which is not related to propagation; `exponent` represents the path loss exponent whose value is normally in the range of 2 to 4 depending of the type of environment; `power loss coefficient` represents the quantity that expresses the loss of signal power with distance; `floor penetration loss factor` is an empirical constant dependent on the number of floors the waves need to penetrate; and `number of floors` represents the number of floors. Being user-defined parameters in the propagation class of Mininet-WiFi, the parameters can be tuned according to observed in particular scenarios with "snowflake" characteristics (e.g., room geometry, obstacles, etc.), a fact that can be exploited when aiming to reproduce physical experiments using Mininet-WiFi.

6.1.2. Results

Figure 9 illustrates the results obtained for the same experiment setup in R2lab and Mininet-WiFi. We can observe the relationship between RSSI and distance, i.e., the signal decrease as the distance between the transmitter and receiver increases. Based on these results, we conclude that the ITU propagation model is more appropriate. Furthermore, based on the measured RSSI values, the model can be calibrated to reflect with more accuracy the conditions of the target environment, R2lab in our case for the follow-up experiments. All logs from R2lab experiments can be found at https://github.com/ramonfontes/miscellaneous/tree/master/r2lab.

[16]Node 33 was not chosen due to observed misbehavior at the time of the experiments.

6.2. Simple File Transfer

We now aim at assessing the end-to-end user/application experience in a real setup, ns-3 and in Mininet-WiFi. The experiment consists of transferring a single file between two nodes in the R2lab testbed (Figure 10) and measure the *transferring time*, *throughput*, *latency* and *packet loss*. Then, we replicate the same scenario (e.g., node distance, WiFi modes, etc.) in Mininet-WiFi and ns-3 in order to compare the similarity of the obtained results and hence assess the realism that the different tools provide compared to a testbed experiment.

6.2.1. Approach

We select node 13 as the AP and nodes 3 and 24 as the server and client, respectively (see Figure 10). We then transfer a 62.6MB file between server and client by using `wget` over TCP and repeat this process for 10 times and measure the total transfer time, throughput, delay, and packet loss. Relevant setup details of this experiment towards reproducibility include: a) nodes working on channel 1, transmission power equals to 15dBm and IEEE 802.11b enabled; b) distance between nodes 3 and 13, and 13 and 24 equals to 2.72m and 4.08m, respectively.

6.2.2. Results

As we can observe from the results presented in Table 5, the experiment run in Mininet-WiFi yields results very close to those obtained in the R2lab testbed. The observed difference of the average transfer time of 10 runs (stdev) is only 5 seconds of a total of about 180 seconds. The measured bandwidth and end-to-end latency are also quite similar. When compared to the results obtained in ns-3 for the same configuration[17] we observe higher throughput over the wireless channel but less goodput. Inspecting the pcap capture we observe a high amount of TCP retransmissions and also

[17]Information about the file configuration can be found at the Mininet-WiFi's manual

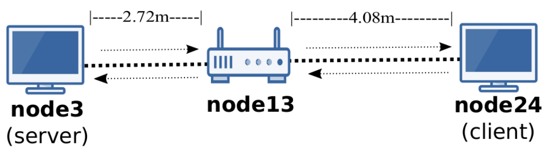


FIGURE 10. Sample Topology

	R2lab	**Mininet-WiFi** (v1.8)	**ns-3** (v3.25)
Throughput	≃337KB/s	≃352KB/s	≃622KB/s
Latency	≃1ms	≃1.2ms	≃1ms
Packet Loss	≃0%	≃0%	≃0%
Transf. Time	≃3min1sec	≃2min56sec	3min55sec

TABLE 5. Mean values of measures obtained from tests

out-of-order and duplicated packets corresponding to approximately 16% of the total packets transferred to complete the file transfer. The observed retransmissions explain why ns-3 provides higher throughput, but finalizing the file transferring in almost 4 minutes. We do not have a clear explanation for the deviations obtained in the ns-3 results, once we tried to our best to configure the simulation parameters to match the same experiment setup run in R2lab and ns-3.

6.3. Replaying Network Conditions

The wireless medium is known for the frequent variations in networking conditions change due to multiple reasons, such as cross traffic and many sources that contribute to fluctuations of the physical medium. We now focus on the ability of Mininet-WiFi to dynamically change the parameters of the network links (e.g. bandwidth, packet loss, latency and delay) based on the distance between the communicating nodes and eventually augmented with observations from an actual experiment in the real world.

Being able to replay real networking conditions based on traffic observations in real environments is useful to predict network performance under certain conditions, reason about the observed network behavior, and perform fair comparisons between alternative algorithms' implementations subject to the mirrors of the physical network. Previous works have explored this approach [46, 47], some of the including wireless scenarios, e.g., TraceR [48], OMNeT++ [10], SimGrid [49], and others. To the best of our knowledge, Mininet-WiFi pioneers the use of this technique in SDWN.

6.3.1. Approach

In this scenario, we transfer a 62.6MB file between two nodes in R2lab and collect real traffic using *wireshark*. By filtering by TCP protocol, we record information about bandwidth and latency. We then replay these information (traces) in Mininet-WiFi by dynamically redefining link bandwidth and latency using Linux TC and measure the results using both 802.11b and 802.11g.

6.3.2. Results

The results are presented in Figure 11. As expected, the total transfer time was less for 802.11g than 802.11b due to the higher transmission speed. Mininet-WiFi outputs a similar behaviour but with slightly higher throughput, and, consequently, finalizes the file transfer file process before R2lab. We believe that the difference is due to the amount of ACK frames generated during the communication by both receiver and transmitter, where in Mininet-WiFi those frames are sent out to a specific interface called *hwsim0* by default. This behaviour differs from the real world where wireless network interfaces are responsible for processing any packet including ACK frames. If more accurate results are desired, the parameters of the wireless channel emulation can be tuned, for instance to reduce the bandwidth (axis X) as needed. In the case of IEEE 802.11g, the average consumed bandwidth in Mininet-WiFi and R2lab during the file transferring was 1.66Mbps and 1.56Mbps, respectively. Thus, reducing the bandwidth provided by Mininet-WiFi in 0.1Mbps (i.e., about 1%) should be enough for getting further closer results when aiming at reproducing experiments for this specific use case scenario.

Regarding *latency* and *packet loss* both R2lab and Mininet-WiFi provide very similar results. Altogether, we believe that the results are accurate within an order of magnitude, which seems to be a "good enough" result for the wireless channel emulation sufficient to support SDWN research commonly focused on the higher-layer control and management features. Nevertheless, we plan to keep improving our approach to emulate the wireless medium with increased fidelity and reproducibility (aka "replay") features in order to provide more realistic experimentation options and deliver accurate results comparable to those from a real testbed.

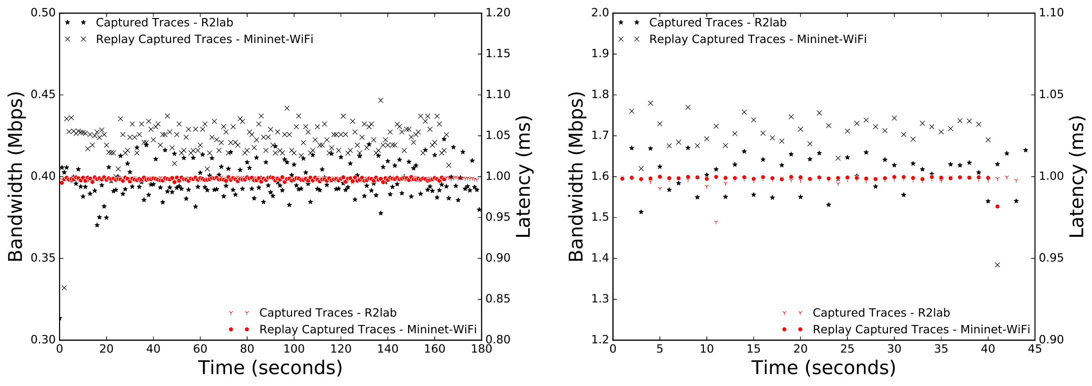
7. CONCLUSIONS AND FUTURE WORK

This article revolves around the question of carrying realistic SDWN experiments, and in particular, the advances reached by our Mininet-WiFi emulation platform.

Extending the success story of Mininet for wired SDN into wireless, Mininet-WiFi introduces itself as a powerful tool for SDWN research and WiFi experiments by supporting arbitrary Linux-based applications, OpenFlow programmability, external SDN controllers, mobility and propagation models among other experimenter-friendly features.

We believe that a lightweight virtualization together with good enough wireless channel emulation capabilities, mobility models and overall experimental scenario reproducibility are steps forwards towards cost-effective and realistic SDWN experimentation. Can we go further? Certainly.

With Mininet-WiFi we have achieved significant steps as validated through the prototype use cases presented and the proof-of-concept experiments to validate our claims of realistic SDWN emulation. To this end we conducted experiments in a real testbed and compared the results with those from the Mininet-



(a) IEEE 802.11b (b) IEEE 802.11g

FIGURE 11. Replaying Network Conditions

WiFi propagation models. Furthermore, we showed the ability to replay network conditions from the real testbed to reproduce in the emulated environment the expected behavior from a real world experiment.

Altogether, the obtained results suggest that Mininet-WiFi is able to mimic the real world with enough fidelity to support meaningful SDWN experiments. All necessary code and instructions to reproduce each of the experiments presented in this article and additional ones are available in the project website.[18]

In order to reach further, some of our ongoing efforts include (*i*) further validation (including scalability limits) via experiments of more complex scenarios, (*ii*) improve the replaying features of captured traces in order to provide more accurate results and facilitate reproducibility (e.g., configure statistics/results collection), and (*iii*) keep adding features and support of scenarios as requested by the increasing user community via the mailing list and the public code repository.[19]

ACKNOWLEDGEMENTS

This research was partially supported by FAPESP grant # 14/18482-4. The R2lab wireless testbed at Inria has been funded by the ANR Equipex FIT 6165.

REFERENCES

[1] Yap, K.-K., Sherwood, R., Kobayashi, M., Huang, T.-Y., Chan, M., Handigol, N., McKeown, N., and Parulkar, G. (2010) Blueprint for introducing innovation into wireless mobile networks. *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, New York, NY, USA, September VISA '10, pp. 25–32. ACM.

[2] Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015) Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, **103**, 14–76.

[3] Jagadeesan, N. A. and Krishnamachari, B. (2014) Software-defined networking paradigms in wireless networks: A survey. *ACM Comput. Surv.*, **47**, 27:1–27:11.

[4] Costanzo, S., Galluccio, L., Morabito, G., and Palazzo, S. (2012) Software Defined Wireless Networks: Unbridling SDNs. *Software Defined Networking (EWSDN), 2012 European Workshop on*, `http://dx.doi.org/10.1109/ewsdn.2012.12`, October, pp. 1–6. IEEE.

[5] Hossain, E., Chow, G., Leung, V. C. M., McLeod, R. D., Mišić, J., Wong, V. W. S., and Yang, O. (2010) Vehicular telematics over heterogeneous wireless networks: A survey. *Comput. Commun.*, **33**, 775–793.

[6] Tarokh, V., Seshadri, N., and Calderbank, A. (1998) Space-time codes for high data rate wireless communication: performance criterion and code construction. *IEEE Trans. Inform. Theory*, **44**, 744–765.

[7] Khajuria, R. and Gupta, S. (2015) Energy optimization and lifetime enhancement techniques in wireless sensor networks: A survey. *International Conference on Computing, Communication & Automation*, may, pp. 396–402. (IEEE).

[8] Imran, M., Said, A., and Hasbullah, H. (2010) A survey of simulators, emulators and testbeds for wireless sensor networks. *Inform. Technol. (ITSim) - Int. Symp.*, **vol.2**, 897–902.

[9] Mancini, E., Soni, H., Turletti, T., Dabbous, W., and Tazaki, H. (2014) Demo abstract: Realistic Evaluation of Kernel protocols and Software Defined Wireless Networks with DCE/ns-3, . pp. 335 – 337. Demo Abstract in Proceedings of ACM MSWiM, Montreal, Canada, September 21-26 2014.

[10] Varga, A. and Hornig, R. (2008) An overview of the omnet++ simulation environment. *Proceedings of the 1st Int. Conf. Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ICST, Brussels, Belgium, Belgium, March Simutools '08, pp. 60:1–60:10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[11] Fontes, R. R., Afzal, S., Brito, S. H. B., Santos, M.

[18]https://github.com/intrig-unicamp/mininet-wifi/wiki
[19]https://github.com/intrig-unicamp/mininet-wifi

A. S., and Rothenberg, C. E. (2015) Mininet-wifi: Emulating software-defined wireless networks. *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)*, Washington, DC, USA, November CNSM '15, pp. 384–389. IEEE Computer Society.

[12] Lantz, B., Heller, B., and McKeown, N. (2010) A network in a laptop: Rapid prototyping for software-defined networks. *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, New York, NY, USA, October Hotnets-IX, pp. 19:1–19:6. ACM.

[13] Peach, S., Irwin, B., and van Heerden, R. (2016) An overview of linux container based network emulation. *15th European Conference on Cyber Warfare and Security*, New York, NY, USA, July ECCWS 2016, pp. 253–259.

[14] Fontes, R. d. R. and Rothenberg, C. E. (2016) Mininet-wifi: A platform for hybrid physical-virtual software-defined wireless networking research. *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference*, New York, NY, USA, August SIGCOMM '16, pp. 607–608. ACM.

[15] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008) Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, **38**, 69–74.

[16] Yang, L., Govindan, S., and Cheng, H. (2015). Objectives for Control and Provisioning of Wireless Access Points (CAPWAP). RFC 4564.

[17] Doria, A., Dong, L., Wang, W., Khosravi, H. M., Salim, J. H., and Gopal, R. (2015). Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810.

[18] Enns, R., Bjorklund, M., Bierman, A., and Schnwlder, J. (2015). Network Configuration Protocol (NETCONF). RFC 6241.

[19] Bernardos, C., La Oliva, A., Serrano, P., Banchs, A., Contreras, L. M., Jin, H., and Zuniga, J. C. (2014) An architecture for software defined wireless networking. *Wirel. Commun., IEEE*, **21**, 52–61.

[20] Sama, M. R., Contreras, L. M., Kaippallimalil, J., Akiyoshi, I., Qian, H., and Ni, H. (2015) Software-defined control of the virtualized mobile packet core. *IEEE Commun. Magaz.*, **53**, 107–115.

[21] Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015) Network function virtualization: Challenges and opportunities for innovations. *Commun. Magaz., IEEE*, **53**, 90–97.

[22] Calhoun, P., Suri, R., Cam-Winget, N., Williams, M., Hares, S., Hara, B. O., and Kelly, S. (2010) Lightweight access point protocol - rfc 5412. Technical report.

[23] Foundation, O. N. Wireless & mobile. `https://www.opennetworking.org/images/stories/downloads/working-groups/charter-wireless-mobile.pdf`. (accessed 07 March 2017).

[24] Suresh, L., Schulz-Zander, J., Merz, R., Feldmann, A., and Vazao, T. (2012) Towards programmable enterprise wlans with odin. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, New York, NY, USA, August HotSDN '12, pp. 115–120. ACM.

[25] Kumar, S., Cifuentes, D., Gollakota, S., and Katabi, D. (2013) Bringing cross-layer mimo to today's wireless lans. *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, New York, NY, USA, October SIGCOMM '13, pp. 387–398. ACM.

[26] Moura, H., Bessa, G. V. C., Vieira, M. A. M., and Macedo, D. F. (2015) Ethanol: Software defined networking for 802.11 wireless networks. *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, `http://dx.doi.org/10.1109/INM.2015.7140315`, May, pp. 388–396. IEEE.

[27] Dely, P., Vestin, J., Kassler, A., Bayer, N., Einsiedler, H., and Peylo, C. (2012) CloudMAC - An OpenFlow based architecture for 802.11 MAC layer processing in the cloud. *Globecom Workshops (GC Wkshps), 2012 IEEE*, `http://dx.doi.org/10.1109/glocomw.2012.6477567`, December, pp. 186–191. IEEE.

[28] Monin, S., Shalimov, A., and Smeliansky, R. (2014). Chandelle: Smooth and Fast WiFi Roaming with SDNOpenFlow. `http://www.usenix.org/sites/default/files/ons2014-poster-monin.pdf`. (accessed 07 March 2017).

[29] Cao, Z., Zhang, R., Hui, D., Pazhyannur, R., Gundavelli, S., Xue, L., and You, J. (2016) Alternate Tunnel Encapsulation for Data Frames in CAPWAP. Internet-Draft draft-ietf-opsawg-capwap-alt-tunnel-08. Internet Engineering Task Force. Work in Progress.

[30] Shao, C., Hui, D., Bari, F., Zhang, R., Matsushima, S., and Pazhyannur, R. (2015). IEEE 802.11 Medium Access Control (MAC) Profile for Control and Provisioning of Wireless Access Points (CAPWAP). RFC 7494.

[31] Foundation, O. N. (2014). Openflow switch specification - version 1.5.0. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf`. (accessed 07 March 2017).

[32] Zimmermann, A., Günes, M., Wenig, M., Ritzerfeld, J., and Meis, U. (2006) Architecture of the hybrid mcg-mesh testbed. *Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, New York, NY, USA WiNTECH '06, pp. 88–89. ACM.

[33] Ahrenholz, J., Danilov, C., Henderson, T. R., and Kim, J. H. (2008) CORE: A real-time network emulator. *MILCOM 2008 - 2008 IEEE Military Communications Conference*, nov. (IEEE).

[34] Chan, M., Chen, C., Huang, J., Kuo, T., Yen, L., and Tseng, C. (2014) Opennet: A simulator for software-defined wireless local area network. *IEEE Wireless Communications and Networking Conference, WCNC 2014, Istanbul, Turkey, April 6-9, 2014*, `http://dx.doi.org/10.1109/WCNC.2014.6953088`, April, pp. 3332–3336. IEEE.

[35] Wang, S.-Y., Chou, C.-L., and Yang, C.-M. (2013) EstiNet openflow network simulator and emulator. *IEEE Communications Magazine*, **51**, 110–117.

[36] Kargl, F. and Schoch, E. (2007) Simulation of manets: A qualitative comparison between jist/swans and ns-2. *Proceedings of the 1st Int. Workshop on System*

Evaluation for Mobile Platforms, New York, NY, USA, June MobiEval '07, pp. 41–46. ACM.

[37] Pechlivanidou, K., Katsalis, K., Igoumenos, I., Katsaros, D., Korakis, T., and Tassiulas, L. (2014) NITOS testbed: A cloud based wireless experimentation facility. *2014 26th International Teletraffic Congress (ITC)*, Blekinge Institute of Technology, Karlskrona, Sweden, sep, pp. 1–6. (IEEE).

[38] Testbeds, O. F. I. (2016). R2lab testbed. `http://r2lab.inria.fr/`. (accessed 27 September 2016).

[39] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. (2002) An integrated experimental environment for distributed systems and networks. *SIGOPS Oper. Syst. Rev.*, **36**, 255–270.

[40] Raychaudhuri, D. (2003) Orbit: Open-access research testbed for next-generation wireless networks. *NSF award# ANI-0335244*, **7**.

[41] Yap, K.-K., Huang, T.-Y., Kobayashi, M., Chan, M., Sherwood, R., Parulkar, G., and McKeown, N. (2009) Lossless handover with n-casting between wifi-wimax on openroads. *ACM Mobicom (Demo)*, **12**, 40–52.

[42] Hyunwoo Nam, Doru Calin, and Henning Schulzrinne (2016) Towards Dynamic MPTCP Path Control Using SDN. *IEEE NetSoft*, February.

[43] Cook, C. and Schwengler, T. (2012). Simultaneous multi-mode wifi differentiated by ssid. `https://www.google.com/patents/US8131303`. US Patent 8,131,303.

[44] Bo Wang, D. A., Kenneth Wan and Thorne, D. (2015). Tr-321 - public wi-fi access in multi-service broadband networks. `https://www.broadband-forum.org/technical/download/TR-321.pdf`. (accessed 27 September 2016).

[45] Saucez, D., Farinacci, D., Iannone, L., and Haddad, W. (2015) A virtual Service Provider for SOHO networks. *European Workshop on Software Defined Networks*, Bilbao, Spain, September, pp. 121 – 122. IEEE.

[46] Eckhardt, D. A. and Steenkiste, P. (1999) A trace-based evaluation of adaptive error correction for a wireless local area network. *Mob. Netw. Appl.*, **4**, 273–287.

[47] Noble, B. D., Satyanarayanan, M., Nguyen, G. T., and Katz, R. H. (1997) Trace-based mobile network emulation. *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, NY, USA SIGCOMM '97, pp. 51–61. ACM.

[48] Acun, B., Jain, N., Bhatele, A., Mubarak, M., Carothers, C. D., and Kal, L. V. (2015) Preliminary evaluation of a parallel trace replay tool for hpc network simulations. *Euro-Par Workshops*, Lecture Notes in Computer Science, **9523**, pp. 417–429. Springer.

[49] Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014) Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *J. Parall. Distrib. Comput.*, **74**, 2899–2917.

[50] Rothenberg, C. E., Chua, R., Bailey, J., Winter, M., Corra, C. N. A., de Lucena, S. C., Salvador, M. R., and Nadeau, T. D. (2014) When open source meets network control planes. *Computer*, **47**, 46–54.