# ns-3 Alignment with the Powder-RENEW Testbed

University of Washington Technical Report[1]

Thomas R. Henderson

University of Washington
Seattle, WA 98195

September 2020

[1]This research was funded in part by NSF award number 1836725: ICE-T: RC: Performance Evaluation of Advanced Wireless Network Edge Infrastructure - Network Simulation & Test Beds. Copyright 2020, University of Washington

# 1 Introduction

This technical report describes recent investigation into how the ns-3 network simulator [3] can be used within, and aligned with, the Powder-RENEW wireless testbed at the University of Utah [5]. This work was funded by the National Science Foundation grant to University of Washington entitled "ICE-T: RC: Performance Evaluation of Advanced Wireless Network Edge Infrastructure - Network Simulation & Test Beds."

The overall technical goals of the broader University of Washington ICE-T project are to:

1. extend the core wireless capabilities of ns-3, positioning the simulator as the research tool of choice for 5G wireless network simulations;
2. improve usability and create new educational/training materials; and
3. better align ns-3 with experiments on wireless network testbeds.

This technical report focuses on the third project topic listed above– the alignment of the ns-3 discrete-event network simulator with network testbeds in general, and specifically with recently announced National Science Foundation-funded Platforms for Advanced Wireless Research (PAWR)– city-scale advanced wireless infrastructure testbeds. In short, this technical report focuses on developing ns-3 to complement wireless experimentation anticipated for the PAWR testbeds. At the time of this work, only one such testbed, the Powder-RENEW testbed at the University of Utah, was operational, so this report focuses on ns-3 alignment with Powder, but similar efforts could be undertaken for other PAWR testbeds in the future.

## 1.1 Motivation

Wireless communication is one of the largest sectors of the global economy and has relevance to most aspects of modern society. The technology supporting wireless communications continues to rapidly evolve. To support research in next-generation wireless systems, NSF has partnered with US Ignite and industry consortium members to fund the deployment of multiple regional public/private wireless testbeds. These testbeds are starting to be deployed, and will feature many experimental radio and network software implementations oriented towards research.

Separately, NSF has funded the development of the ns-3 simulator, which has models for TCP/IP, Wi-Fi, 4G/5G cellular, and many other technologies. A simulator is a computer program with detailed abstractions for real-world networks, including models for wireless channels, wireless devices, protocol stacks, and applications. Simulators have certain advantages and disadvantages

with respect to testbeds, and the applicability of simulators vs. testbeds depends on the problem being studied. Simulators are a complementary tool to mathematical analysis and real-world testbeds.

Future researchers will want to use a mix of mathematical analysis, simulators, and testbeds or field experiments to conduct studies. Our hypothesis is that will benefit future researchers to have ns-3 and emerging wireless testbeds aligned where possible, so that users face fewer discontinuities when moving between the two environments, or when trying to use both within the same experiment.

## 1.2 Summary of Findings

In this work, we experimented with ns-3 operation on the Powder testbed, and we interacted with two university research groups conducting research using Powder. We also received some feedback from participants at an NSF Workshop on Wireless, Spectrum, and Innovation (August 2020), and have taken it into consideration in our findings. We summarize the key results below:

- The user experience with using the two tools is quite different. The skillsets necessary to use ns-3 and Powder are different, and the types of experiments that can be realized are also very different. Presently, there is a lack of documentation that allows users to easily migrate between the two tools. We have taken the following steps to improve this situation. First, we have created an extension module for the ns-3 App Store that provides Powder support scripts, programs, and documentation that will help users to get started using ns-3 with Powder. Second, we have created a Powder disk image with the latest ns-3 release (ns-3.32) pre-installed, as well as installation of specialized network interface drivers that boost the throughput and latency performance when using ns-3 in an emulation mode.
- ns-3 can be used in an emulation mode on compute resources on the Powder platform, allowing it to supplement Powder resources in an experiment. We describe the capabilities and limitations of ns-3 emulation mode, and provide performance results with using traditional ns-3 emulation as well as newer DPDK and Netmap emulation modes, along with scripts to aid with reproducing on the Powder testbed.
- Powder can be used to provide data to populate or calibrate trace-driven empirical performance models in ns-3, and we provide an example, from a summer REU project, for illustration; this example is also available as part of our Powder support module.
- Opportunities exist to harmonize toolchains and scripting that would enable users to create reproducible experiments in both ns-3 and Powder, because neither platform presently has a standardized template for users to do so. We suggest future directions for how ns-3 experiment control frameworks might be combined with Powder orchestration and measurements.

## 1.3 Outline

This report is organized as follows:

- Chapter 2 introduces the ns-3 network simulator and its existing capability to interact with testbeds;
- Chapter 3 describes the Powder-RENEW testbed;
- Chapter 4 describes how ns-3 emulation can allow ns-3 to be combined with other Powder resources in a single experiment;
- Chapter 5 describes how Powder measurement data can be used to build or calibrate similar ns-3 simulation models; and
- Chapter 6 provides a summary and suggests future directions for aligning experiment control frameworks.

# 2 Background on ns-3 and testbeds

ns-3 is the third major version of a discrete-event network simulator for computer network performance evaluation; ns-1 was started at Lawrence Berkeley National Laboratory in 1995. ns-3 is a (Layer-2 and above) packet-level simulator that provides models of packet transmissions over communication channels and does not attempt symbol-level realization typical of a MATLAB-level (Layer-1) physical simulations. ns-3 is now a mature tool featuring the following capabilities, some of which are unavailable even in commercial tools:

- a C++ core with an automated process for generating Python bindings, allowing the researcher to choose to author experiment scripts in either language;
- bit-level realism of packet data structures, which when combined with a special scheduler locking the progression of the simulator's virtual time with the real-world machine time, enables various *emulation* modes of operation in which ns-3 packets are sent to or read from actual network links;
- easy generation of packet traces that can be read by industry-standard tools such as tcpdump and Wireshark;
- a novel framework for compiling existing software implementations of many protocols, such as routing daemons and traffic generators, and even the Linux and FreeBSD networking stacks, and making them available as ns-3 models, with little to no software changes required; and
- core support for parallel, distributed simulations of point-to-point link topologies, allowing certain kinds of ns-3 scenarios to be spread across a cluster of workstations to leverage more computational power and memory.

These features benefit the researcher by permitting more software reuse and allowing more seamless transition between the simulation and experimental (testbed, field trial) phases of research.

A focus of this report is the potential for ns-3 to interact with testbeds. ns-3 has an emulation capability that, when enabled, changes the behavior as follows:

- the simulator clock is aligned with the host system clock,
- checksum computation, disabled by default to reduce computation when they are not being checked, is enabled, and
- the simulator is bound to one or more network devices such that it can send and receive packets over these interfaces.

The emulation capability is not limited to network interfaces; any file descriptor may be used. Emulation in ns-3 is only supported for Linux.

Emulation capabilities have been around since the early releases of ns-3. The initial and default capability has been built around the Linux packet socket capability. In short, ns-3 opens a packet socket to an existing network device, and reads and writes packets from the socket. The packet socket bypasses the TCP and IP networking layers in the Linux stack and interfaces directly with the device. There are different modes in which ns-3 can bind to this device; it can be configured to own the device (intercepting all traffic) or it can be bridged to it, such that it uses a Linux bridge device to allow it to coexist with the usage of the device by the host system.

This capability has been extended by a few previous efforts. We have previously documented (in 2009) how to run ns-3 in the Rutgers WINLAB Orbit radio grid [16], and documentation is still published on the ns-3 wiki regarding how to configure such operation. ns-3 emulation has been used to underpin virtual machine frameworks for networking research, including CORE [6], and Mininet [11]. Support for using ns-3 in the context of PlanetLab and the European OneLab was added several years ago while those projects were active [15]. More recently, National Instruments and Raytheon/BBN have both reported on adapting the capability to interface ns-3 emulation at the MAC layer with software-defined radios providing the PHY layer.

As part of this work, we performed testing and further integration of two improvements to the core ns-3 emulation capability, both of which can be used on Powder as described herein. Netmap [17] is a fast packet processing framework that bypasses traditional kernel processing and reduces memory copies, per-packet memory allocations, and system overheads. A project led by Pasquale Imputato adapted this work for ns-3 emulation [10]. Intel has also released a Data Plane Development Kit (DPDK) that improves packet processing performance for Intel network interface cards. A team at NITK Surathkal added ns-3 emulation support for DPDK [14]. Both emulation capabilities will be part of the next ns-3 software release.

# 3 Summary of Powder-RENEW testbed

The National Science Foundation's (NSF) Directorate of Computer and Information Science and Engineering (CISE) has recently funded three advanced wireless testbed projects, Powder-RENEW, AER-PAW, and COSMOS, as part of the Platforms for Advanced Wireless Research (PAWR) initiative. During the timeframe of this project, only Powder-RENEW was operational.

## 3.1 Powder-RENEW

Powder-RENEW is a joint project funded by the NSF AWR initiative. Powder *(Platform for Open Wireless Data-driven Experimental Research*) is developed by the University of Utah, based on the predecessor Emulab system [19], and is operated by the University of Utah, Salt Lake City, and the Utah Education and Telehealth Network. Powder includes technologies from RENEW *(Reconfigurable Ecosystem for Next-gen End-to-end Wireless)* [7], a project from Rice University billed as the "World's First Fully Programmable and Open-source Massive-MIMO Platform." RENEW includes components to enable research on massive MIMO (mMIMO) systems, including base stations and endpoints from an associated company, Skylark Wireless, and mMIMO specific software. Following the convention in the online manual, we will refer to the overall testbed system as "Powder."

In practical terms, Powder is a collection of wireless and computing hardware deployed in the Salt Lake City area, interconnected by infrastructure and wireless networks, and accessible to researchers via an online portal. Users with credentials may login to the portal and schedule and configure experiments to be run on the testbed, including selected experiments that are conducted over-the-air. The overall testbed is programmable, allowing researchers to implement new ideas and to gather data on performance over real wireless channels. The project advertises also that users can bring their own wireless devices to the testbed, in some capacity.

Powder uses the following terminology [18] to describe experimentation on the platform:

- **profile:** A *profile* encapsulates everything needed to run an experiment, including descriptions of the required physical resources, and the software to run on those resources.
- **Rspec:** An *Rspec* (resource specification) is a formalized description of all of the resources needed to support a profile.
- **experiment:** An *experiment* is an instance of a *profile*. Experiments on Powder are essentially software operations orchestrated across a distributed network system containing radios, programmable network devices, computing, and storage.

- **disk image:** The primary way that software is associated to profiles is through the use a disk image, which is a snapshot of the contents of a real or virtual disk. Disk images usually consist of a full operating system image, and nodes defined in the Rspec are booted from such images.

Powder is focused on the 700MHz to 6GHz band cellular networks, and currently contains the following physical resources (which will evolve over time).

- National Instruments and Ettus Research SDRs (base stations and fixed terminals) capable of 2x2 MIMO
- Skylark Wireless Massive MIMO equipment (base stations, fixed terminals, and near-edge computing resources)
- Cloud/edge computing clusters from CloudLab and EmuLab

Powder also will allow users to deploy their own radio systems. Many different software stacks are supported on the SDRs, including GNU Radio, OpenAirInterface, and srsLTE. Mobile terminals are expected to be added in 2020.

## 3.2 Powder and ns-3 terminology

Generally, a scientific experiment is defined as a procedure to gather data to make a discovery or test a hypothesis. Experiments that use simulations or testbeds require the definition and instantiation of an environment suitable to gather the data of interest. Powder uses the term *experiment* to refer to the instantiation of testbed resources, but not to refer to the methods or procedures that a researcher may use to gather data once instantiated. In that sense, what is called an experiment in Powder might instead be called an instantiation. A Powder *profile* is the recipe used by the system to construct the experiment, and it describes the resources required using a schema called an *Rspec*. Users then are provided with a fully instantiated platform, and the conduct of an actual experiment (further set up of run-time processes on computers, gathering of data, repetition of trials, processing and analysis of data, archiving of data) is left for the user to overlay on top of the Powder experiment.

In ns-3, users define simulation *scenarios* that fully define how to conduct a trial that produces data. There is generally no equivalent of a Powder *experiment* that constructs the simulation to a running state but that then awaits further inputs from the user to do something useful. Instead, one aspect of the scenario is to define simulation events that will trigger, during simulation runtime, the models to start acting in certain ways and producing data during the run. An ns-3 simulation is a running process that typically runs to completion based on its initial definition. In emulation mode, however, it is possible to send signals or data into the process, typically by sending packets into the network interface card that is bound to the emulation process. Therefore, one might equate an ns-3 *emulation instance* to what Powder calls an *experiment*. ns-3 does not have a specialized resource description language such as Rspec; instead, simulation scenario descriptions are written directly in C++ or Python code.

| **Powder term** | **ns-3 equivalent** |
|---|---|
| Rspec | C++ or Python code |
| profile | scenario |
| experiment | emulation instance |

Table 3.1: Comparison of Powder and ns-3 terminology

ns-3 also delegates the management of a full scientific experiment to the user. That is, a full experiment or study may involve parametric combinations or the management of multiple independent replications of a study. ns-3 does not standardize a way to manage this, but some frameworks for such management have emerged. One under active development is the Simulation Execution Manager *(sem)* developed specifically to manage reproducibility and to coordinate statistical analysis of simulations [12]. *sem* is written in Python and can orchestrate the running of multiple simulations and the writing of output data into a database. *sem* is not tightly coupled with ns-3 and could conceivably also be adapted to manage testbed experiments such as on Powder, and since it is a Python library, it has access to the Python packages available for a Python-based data science workflow.

```
"""Two physical nodes running Ubuntu 20.04 with additional ns-3 emulation
    support, connected by an Ethernet link.

Instructions:
Wait for the profile instance to start, and then log in to either host.
"""

import geni.portal as portal
import geni.rspec.pg as rspec

request = portal.context.makeRequestRSpec()

# Create two raw "PC" nodes
node1 = request.RawPC("node1", component_id="pc759")
node2 = request.RawPC("node2", component_id="pc758")

# Request that a specific image be installed on this node
node1.disk_image = "urn:publicid:IDN+emulab.net+image+ns-3-alignment:ubuntu
    -20.04-ns-3";
node2.disk_image = "urn:publicid:IDN+emulab.net+image+ns-3-alignment:ubuntu
    -20.04-ns-3";

# Create a link between them
link1 = request.Link(members = [node1, node2])

portal.context.printRequestRSpec()
```

Listing 3.1: Powder profile for two-node experiment

The program listing above is a Powder profile, written in Python, to instantiate two Linux computers connected by a 1 Gbps Ethernet link. The profile is written in Python and converted to the

lower-level Rspec format by a Python library called *geni-lib*. The lines 13-14 request the instantation of raw PCs called "node1" and "node2" (in the experiment) on physical hosts named "pc759" and "pc758" in the Emulab testbed. Lines 17-18 instruct Powder to load a specific operating system image; in this case, Ubuntu 20.04 Linux that has had ns-3 emulation capabilities added to it. Line 21 instantiates the link between the nodes. Note that this description does not ask the nodes to do anything once instantiated; it merely describes how to bring the system to an operational state.

```
1  {
2  <rspec xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jacks="
       http://www.protogeni.net/resources/rspec/ext/jacks/1" xmlns:client="http:
       //www.protogeni.net/resources/rspec/ext/client/1" xmlns:emulab="http://www
       .protogeni.net/resources/rspec/ext/emulab/1" xmlns="http://www.geni.net/
       resources/rspec/3" xsi:schemaLocation="http://www.geni.net/resources/rspec
       /3 http://www.geni.net/resources/rspec/3/request.xsd" type="request">
3    <rspec_tour xmlns="http://www.protogeni.net/resources/rspec/ext/apt-tour/1">
4      <description type="markdown">Two physical nodes running Ubuntu 20.04 with
       additional ns-3 emulation support, connected by an Ethernet link.</
       description>
5      <instructions type="markdown">Wait for the profile instance to start, and
       then log in to either host.
6  </instructions>
7    </rspec_tour>
8    <node client_id="node1" exclusive="true" component_id="pc759">
9      <sliver_type name="raw">
10       <disk_image name="urn:publicid:IDN+emulab.net+image+ns-3-
       alignment:ubuntu-20.04-ns-3"/>
11     </sliver_type>
12     <interface client_id="node1:if0"/>
13   </node>
14   <node client_id="node2" exclusive="true" component_id="pc758">
15     <sliver_type name="raw">
16       <disk_image name="urn:publicid:IDN+emulab.net+image+ns-3-
       alignment:ubuntu-20.04-ns-3"/>
17     </sliver_type>
18     <interface client_id="node2:if0"/>
19   </node>
20   <link client_id="link-1">
21     <interface_ref client_id="node1:if0"/>
22     <interface_ref client_id="node2:if0"/>
23   </link>
24 </rspec>
25 }
```

Listing 3.2: Corresponding Powder Rspec for two-node experiment

The above listing is the XML-based translation of the previous Python profile description. Users are able to write profile descriptions directly as Rspec XML, but would typically work at the Python layer and have Powder translate it to XML.

Figure 3.1 displays a screenshot of the topology view that Powder provides upon instantiating the above profile.

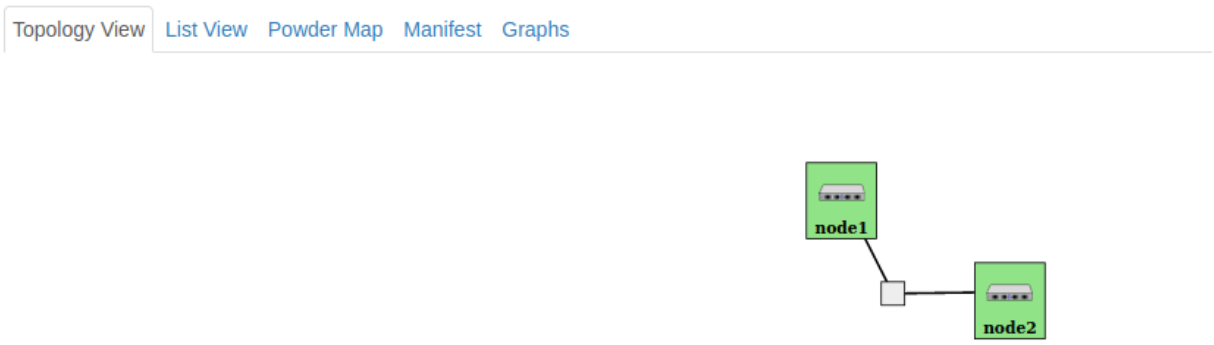


Figure 3.1: Depiction of two node profile on Powder

```
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"

using namespace ns3;

int
main (int argc, char *argv[])
{
  // Initialize some parameters and import command-line arguments
  bool useV6 = false;
  Address serverAddress;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
  cmd.Parse (argc, argv);

  // Create two simulation nodes
  NodeContainer n;
  n.Create (2);

  // Add an internet stack to each node
  InternetStackHelper internet;
  internet.Install (n);

  // Add a long-delay CSMA-based (Ethernet) link between the two nodes
  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
  csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
  csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));
  NetDeviceContainer d = csma.Install (n);

```

```
35   // Configure either IPv4 or IPv6 addresses
36   if (useV6 == false)
37     {
38       Ipv4AddressHelper ipv4;
39       ipv4.SetBase ("10.1.1.0", "255.255.255.0");
40       Ipv4InterfaceContainer i = ipv4.Assign (d);
41       serverAddress = Address (i.GetAddress (1));
42     }
43   else
44     {
45       Ipv6AddressHelper ipv6;
46       ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));
47       Ipv6InterfaceContainer i6 = ipv6.Assign (d);
48       serverAddress = Address(i6.GetAddress (1,1));
49     }
50
51   // Configure simple UDP-based applications
52   uint16_t port = 4000;
53   UdpServerHelper server (port);
54   ApplicationContainer apps = server.Install (n.Get (1));
55   apps.Start (Seconds (1.0));
56   apps.Stop (Seconds (10.0));
57
58   uint32_t MaxPacketSize = 1024;
59   Time interPacketInterval = Seconds (0.05);
60   uint32_t maxPacketCount = 320;
61   UdpClientHelper client (serverAddress, port);
62   client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
63   client.SetAttribute ("Interval", TimeValue (interPacketInterval));
64   client.SetAttribute ("PacketSize", UintegerValue (MaxPacketSize));
65   apps = client.Install (n.Get (0));
66   apps.Start (Seconds (2.0));
67   apps.Stop (Seconds (10.0));
68
69   // Run the simulation, and then deallocate the allocated memory
70   Simulator::Run ();
71   Simulator::Destroy ();
72 }
```

Listing 3.3: Similar ns-3 program

The above listing is a loosely equivalent ns-3 simulation program, edited from an existing example program in the ns-3 mainline (`examples/udp-client-server/udp-client-server.cc`). More specifically, the configuration lines up to line 49 are analogous to the two-link Powder profile shown above, in that they set up a similar two-node topology interconneced by an Ethernet link. Lines 51-67 configure the simulation to do something to produce data after the simulation starts to run (line 70). In Powder, lines 51-67 or other similar actions would instead be configured by the user at the command line or by installing further scripts on the Powder computers. In an ns-3 emulation, something analogous to lines 1-49 above would be defined, and (optionally) additional ns-3 triggered events could be configured to occur at a time offset from the start of the ns-3 process. Then, this compiled ns-3 program would need to be run on a Powder computer after the Powder experiment was started. ns-3 also provides the ability to write the above program in Python rather

than C++, but most users tend to write their scenarios in C++ because the underlying modeling language in ns-3 is C++.

## 3.3 Use cases for ns-3 and Powder

With the above differences in mind, we foresee the following use cases for ns-3 and Powder combinations.

- Using Powder compute nodes to execute **pure ns-3 simulations**.
- ns-3 **emulation** used in coordination with Powder nodes to construct hybrid topologies
- ns-3 **model construction or validation** based on Powder experimental data
- Experimental **workflow harmonization**

The first case is not a novel use case, but Emulab and Cloudlab have many powerful multi-core servers available that could be leveraged by users to execute pure ns-3 simulations, for cases in which the researcher may not have access to a compute cluster. We do not discuss this case further herein.

The third use case is to use Powder to validate ns-3 link and model performance on actual hardware and RF links (to the extent possible). This could mean, for instance, gathering empirical data that can be directly used in simulations, or used to validate the existing stochastic models in ns-3. Alternatively, a small-scale experiment could be conducted on Powder, and the data gathered from the small scale experiment used to guide the construction of a larger ns-3 simulation (larger than could be realized on the Powder platform). We elaborate on this case further in Section 5.

The second use case is to create an environment that mixes ns-3 emulation with Powder resources. In Section 2 above, we mentioned that it is possible to combine run ns-3 in emulation mode and connect it to network interface cards (NIC), or to connect it at a lower level (physical layer) to software-defined radios (SDR). We did not perform any work to try to instantiate or configure ns-3 running on Powder SDRs, but experimented with ns-3 emulation over Powder Ethernet NICs; this is described more in Section 4.

The final use case is to work towards aligning the workflows and scripting tools used in both environments. Both ns-3 and Powder have the capability to be used as components in a Python-centric workflow that additionally leverages additional Python libraries for data collection, analysis, and plotting. Once workflow management tools for higher-layer experiments are defined/standardized for Powder, ns-3 analogs can be implemented so that users can more easily map actions taken on the Powder testbed to actions in an ns-3 simulation. Output file formats can also be aligned to allow common data processing scripts to operate. The predecessor of Powder, Emulab, used ns-2 OTcl syntax to describe Emulab experiments. Now that Powder is using *geni-lib* for profile definition, ns-3 could conceivably support some of the *geni-lib* API so that users could write profile-like simulation or emulation scenarios. This report does not go into further detail on this aspect, but lists the topic as possible future work in Section 6.

# 4 ns-3 emulation on the Powder testbed

We start this section by describing a sample use case from an actual research project conducted on Powder during summer 2020. A group at the University of Missouri was interested in using Powder to study whether machine learning frameworks could detect anomaly events and take countermeasures for a swarm network of UAV drones (a flying ad-hoc network, or FANET) in which one or more nodes were compromised and interfering with airborne data collection or reduction activities. They wished to experiment with a custom secure protocol coordinating agents deployed on the UAV with agents in a backhaul network. The research team found that they could not instantiate the full topology on Powder because Powder only consists of fixed-link terrestrial over-the-air links at this time. They wanted to make use of Powder edge servers as computational resources.

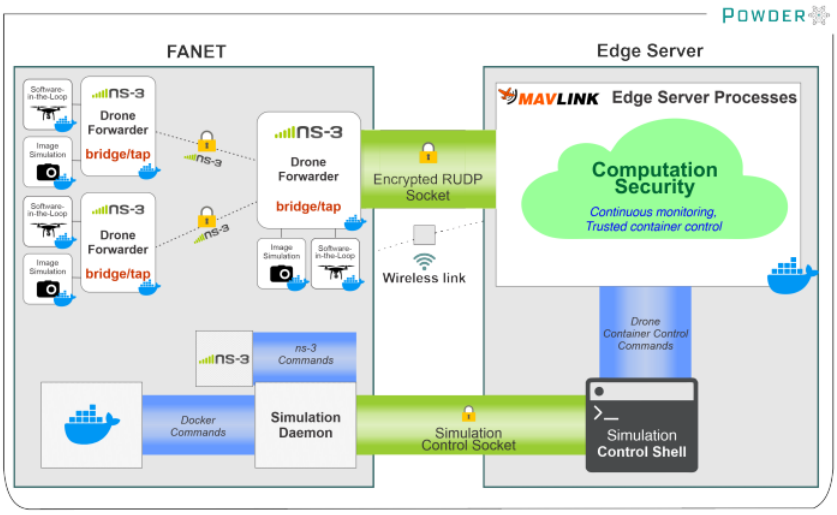


Figure 4.1: University of Missouri ns-3 FANET experiment (Figure 5 from [8]

In the late spring of 2020, we discussed this research problem with the University of Missouri team and answered questions pertaining to the use of ns-3 emulation. The team decided to implement a hybrid ns-3/physical experiment in Powder, using ns-3 emulation for the FANET components. They reported that the system could support up to 254 simulated drones, limited by the 8-bit size of the integer used to represent the number of systems with the Mavlink messaging protocol that they used. This work is reported in [8], and a summary figure is shown in Figure 4.1.

We experimented with ns-3 emulation configurations on Powder, and in the remainder of this section, we focus on some experiments and experiences in configuring Powder compute nodes for ns-3 emulation.

## 4.1 Deploying ns-3 on Powder

ns-3 can be deployed on existing hardware and disk images, using predefined experiment profiles. One such simple example is the "two-pc-link" example linking two Linux-based computers by a network link (an ns-3 enhanced version of this topology was introduced above in Section 3, but a generic instance of this profile is provided by the Powder team under the name *two-pc-link*, shown in Figure 4.2). This profile instantiates and connects two Linux-based computers by a network link. The default profile uses Ubuntu 18.04 Linux images, but other disk images are provided by Powder as alternatives.

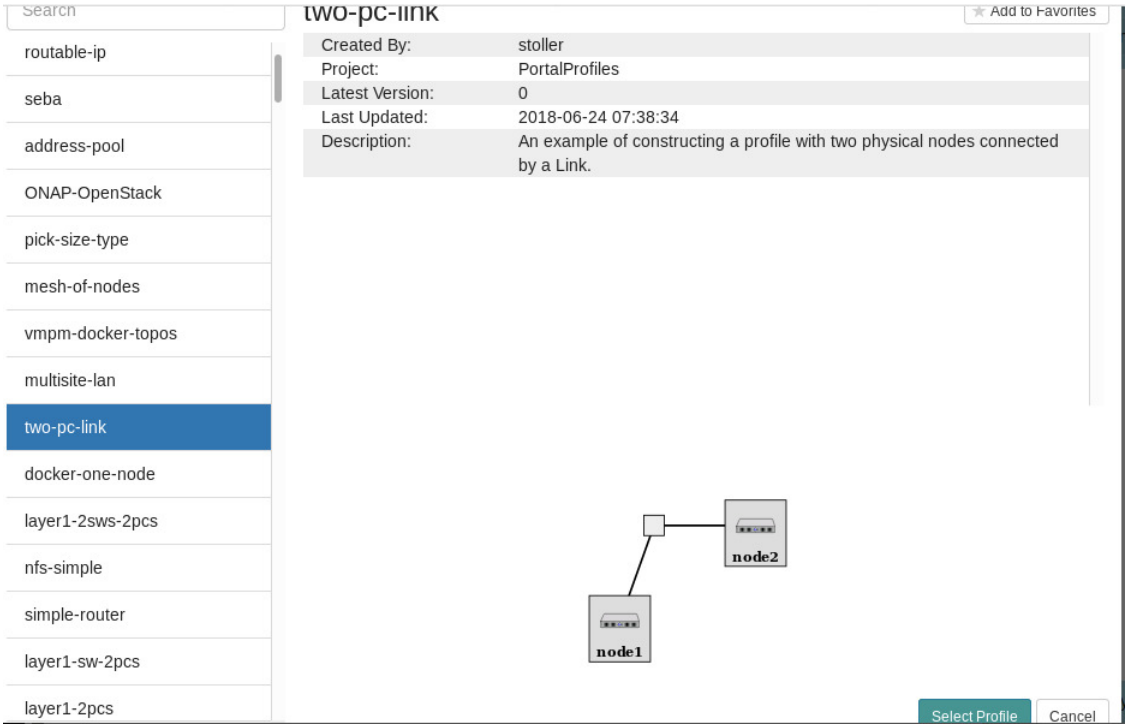


Figure 4.2: Depiction of two-pc-link profile on Powder

A Powder user can start from any of these profiles and images and add ns-3 to the running experiment. Powder provides shell access (via ssh or web console) to the virtual machines. The disk images have C++ development environments pre-installed, so installing ns-3 can be quickly done by downloading ns-3 from its public Git repositories or the main ns-3 web server. However, doing so from scratch every time can be tedious, so we have provided enhanced ns-3 disk images (under the URN `urn:publicid:IDN+emulab.net+image+ns-3-alignment:ubuntu-20.04-ns-3`) for reuse by others. These ns-3 disk images can be further updated or customized by users.

We have also used these images to test and benchmark the netmap and DPDK variants of ns-3 emulation devices, and aligned with the ns-3 code available in ns release 3.32 or later.

## 4.2 Emulation configuration and performance

We next describe some experiments conducted on Powder with ns-3 disk images (available at `urn:publicid:IDN+emulab.net+image+ns-3-alignment:ubuntu-20.04-ns-3`) that were based on the stock Ubuntu 20.04 Linux images provided by Powder. These images were extended as follows:

1. The latest ns-3 software (at the time of writing,
2. Installation of Ubuntu packages to enable DPDK (packages `dpdk, dpdk-dev, libdpdk-dev`)
3. Configuration and installation of netmap-enhanced drivers and the netmap kernel module
4. Installation of the iperf3 traffic testing tool
5. Update to the latest Linux kernel (5.4.0-47) for this Ubuntu system
6. A script to help users identify the device drivers in use, called `list-drivers.sh`, has been added.
7. A script to help users configure the system for different netmap configurations, called `netmap-config` has been added.

As stated above in Section 2, the built-in raw socket capabilities of Linux are sufficient for enabling ns-3 emulation, but there exist newer fast packet processing drivers for modern NICs that allow emulation to be conducted with higher fidelity, both in terms of improved throughput and improved latency accuracy. Support for these modes (netmap and DPDK) has recently been added to the ns-3 mainline, starting with release ns-3.32. They require a bit of extra node configuration and (possibly) tuning to use on Powder nodes. The *Powder support* module in the ns-3 App Store provides more information, including how to use Powder disk images customized to support ns-3.

### 4.2.1 RawPC selection

When an experiment is realized, Powder finds raw PCs that are presently unused from the Emulab cluster. Different PCs in the cluster have different hardware NICs and CPUs; therefore, in general, users will not necessarily obtain access to the same underlying hardware for every experiment instantiation, so scripts that make assumptions on underlying hardware may fail if the hardware changes.

When using netmap emulation, this uncertainty affects the configuration as follows. First, the ethernet device name will generally vary (e.g. `enp9s4f1`). Second, depending on the hardware, the driver loaded will be different. netmap requires to unload the stock driver and load a modified

driver (modified drivers are available in the directory `/usr/local/src/netmap` of the ns-3 image). In the case of Emulab node pc256, for instance, the Intel "e1000" driver is used, but for node pc759, the Intel "igb" driver is used.

Similarly, for DPDK emulation, the user must specify the matching poll mode driver for the underlying device. In the case of e1000, the driver is named `librte_pmd_e1000.so`. The user must also determine the underlying PCI device name and use that in the program rather than the more typical interface name such as `eth0`. There are several driver configuration steps and configuration of hugepages support as well.

There is a way to specify, in the experiment profile, to use specific hardware. A request for a generic PC can be made as follows:

```
node1 = request.RawPC("node1")
```

Listing 4.1: Request for generic PC

A specific request for an individual PC can be made by naming it as follows (in this case, pc759):

```
node1 = request.RawPC("node1", component_id="pc759")
```

Listing 4.2: Request for specific PC

While the latter will ensure that the same hardware is instantiated each time, it may result in experiments not being instantiated if Powder has already allocated that specific node to another user.

If specific nodes are not available, users can fall back to requesting other PCs in the pool that have the same hardware. To request a specific "class" of nodes, rather than an individual node, one can configure the profile as follows:

```
node1 = request.RawPC("node1")
node1.hardware_type = "d430"
```

Listing 4.3: Request for class of PC

The `d430` machine type is a newer 32-core Intel Xeon E5-2630 server with Intel Gigabit Ethernet cards, and is the basis for the experiments shown below.

For reproducibility, users should identify the compute nodes in use when reporting results.

### 4.2.2 ns-3 emulation performance

Figure 4.3 provides performance results comparing the ns-3 emulation performance on the `pc839` Emulab PC; a `d430` type machine as described above. The figure compares the raw packet sending throughput obtained for different packet sizes for three emulation modes: 1) the baseline raw (packet socket) emulation performance, 2) generic netmap support (leveraging the generic netmap kernel module but without modified device drivers), and 3) native netmap support (use of netmap-enable device driver). For raw packet sockets, kernel offload engines were enabled, but for netmap

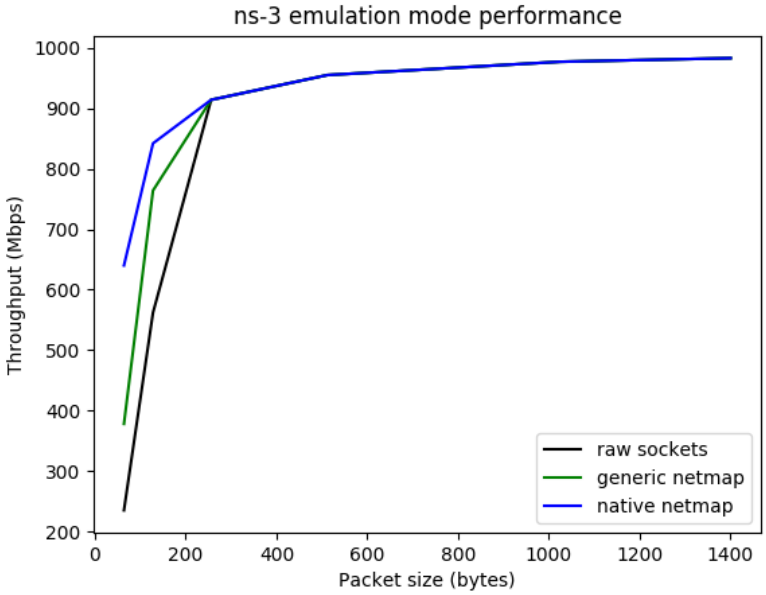


Figure 4.3: ns-3 emulation mode performance comparison

experiments, all offload was disabled. Packet sizes of 64, 128, 256, 512, 1024, and 1400 bytes were tested. The figure shows that for this class of machine, all three modes are able to achieve the capacity limit of the Gigabit Ethernet device for packet sizes of 512 bytes or greater, but for smaller packet sizes, the netmap modes obtain higher throughput.

# 5 Powder measurement data and ns-3

Powder experimental data can be used to build models in ns-3, or to validate existing models. This section describes how data generated by a recent Powder experiment can be used to build an ns-3 program with corresponding link performance.

## 5.1 ns-3 models based on Powder data

One observation about wireless experiments is that the link performance is typically much more variable than expressed in traditional stochastic path loss models, and may lead to more difficult real-world conditions than predicted by analysis or simulation. For instance, channel reciprocity theorems suggest that the path loss exhibited between two nodes should be the same in both directions. However, actual wireless links that are configured symmetrically may perform differently, due to implementation defects or other factors. As a result, more pathological conditions (such as node A hearing node B, but node B cannot hear node A) may arise in the field than in simulation. Channel conditions may exhibit time-of-day or seasonal variations due to interference or foliage effects. For this reason, it may be useful to take data from Powder experiments and import them into ns-3 models, either directly as imported empirical data points, or to use the data to build stochastic models.

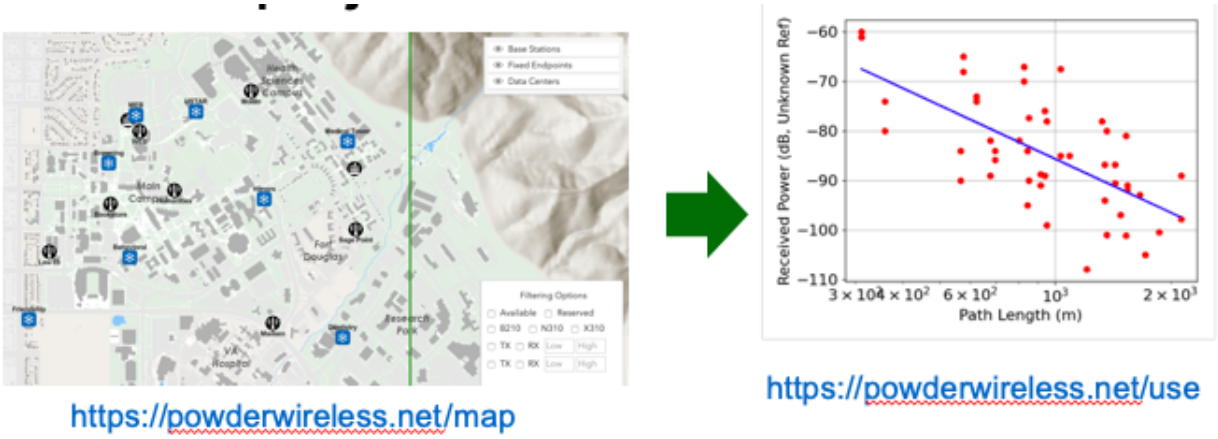


Figure 5.1: Path loss measurements with Powder

Prof. Neal Patwari of Washington University in St. Louis supervised a summer REU project on Powder, the results of which were highlighted on the Powder web site [1]. The project involved

channel measurements between the eight currently deployed CBRS rooftop nodes (a National Instruments X310), in which one rooftop node was used as a transmitter for a period of time, with the other seven as receivers, and repeating until 56 unidirectional links were measured. A figure published on the Powder web site ( 5.1) plots the 56 data points of received power as a function of distance (the coordinates of each node were known). Curve fitting resulted in a path-loss exponent of 3.6; the transmitted signal was a narrowband signal at 3555 MHz.

This type of data can be used in two ways to construct a similar ns-3 model. The traditional stochastic models (including the ns-3 LogDistance propagation loss model) can be used to provide a curve such as the blue fitted line in the plot (with proper exponent), but the variability and difference in observed receive power for the same path length but different directions (note that for each path length, corresponding to a unique pair of nodes, there are two measurements, and they do not often overlap) would not be captured by such a simple model. A more sophisticated loss model could be built based on the variance empirically observed.

Another possibility is to encode a table within an ns-3 empirical path loss model for a simulation scenario that used nodes only at those specific eight locations, and looked up one of the 56 average received powers based on the transmit and receive locations (which must match with locations in the table). This would be a very specialized model, perhaps called the `PowderCbrsRooftopLinkLossMo`
We have implemented this and posted it as example code in the Powder Support app in the ns-3 App Store (https://apps.nsnam.org/app/powder-support). The data for this app was provided by Neal Patwari; it allows the receive power to be set to the average value that was obseved by the tests conducted during this summer REU project. The example program is designed to reproduce the plot from Figure 5.1 by sending a packet through an ns-3 wireless channel from each of eight nodes to the other seven nodes, and to print out the received power, thereby reproducing the above plot.

This example is for demonstration purposes and is not likely to be used directly in any simulation programs (simulations are not likely to need to build a rooftop mesh network between these eight nodes at 3555 MHz) but the approach can be adapted to other measurements obtained in the future.

Figure 5.2 plots the received power for an ns-3 propagation loss model built from the 2020 REU data supplied by Neal Patwari. The input data was the geographic position (geographic coordinates) of all stations, and the average observed receive power at each receiver. The geographic position was converted to cartesian coordinates based on the ns-3 `GeographicPositions` class, and a constant position mobility model for each node was instantiated. Next, a custom propagation loss model was written that used that encoded a table for the received power (converted to watts) as a map based on location of each node. The conversions were necessary because ns-3 operates on cartesian coordinates and linear power units. The example program constructs a shared channel and attaches each node to the channel, and sends a packet from each node to all other nodes (similar to how the actual Powder experiment was conducted). The observed received power is then traced and an output file generated in the same format (dB) and layout as the original Powder data, and Matplotlib was used to produce the resulting plot. Source code to reproduce the data for this figure can be found in the `powder-support` extension module in the ns-3 App Store [2].

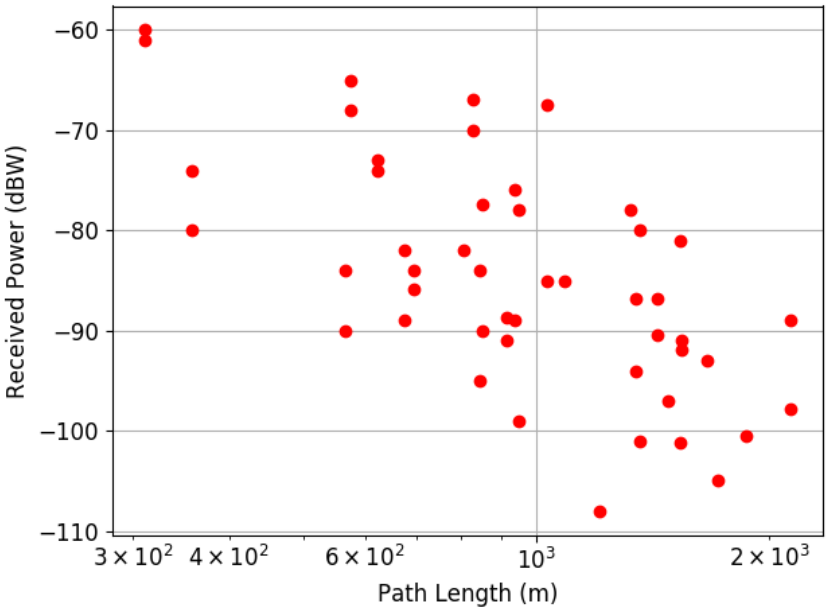


Figure 5.2: Path loss values from equivalent ns-3 program

## 5.2 Validation of ns-3 models with Powder models

ns-3 contains models for 4G LTE systems, documented in [4]. The models include detailed models of the LTE radio stack, including RRC, PDCP, RLC, and MAC layers, and a simplified model of the PHY, with block error rate curves imported from a study conducted on an LTE link simulator. ns-3 also includes a simplified Evolved Packet Core (EPC) model, with notional SGW, PGW, and MME nodes. The bulk of the modeling was conducted based on 3GPP standardization releases 8 through 12.

Powder currently uses two primary platforms (both leveraging software-defined radios) for supporting LTE on the testbed: Open Air Interface [13] and srsLTE [9]. At the time of this work (summer 2020), the only over-the-air capability deployed on Powder was srsLTE, on eight fixed rooftop eNB nodes with eight fixed UEs at ground level.

We experimented with the provided srsLTE Powder over-the-air experimental profiles, operating on the 2.4-2.6 GHz Broadband Radio Service and Education Broadband Service spectrum. The profile instantiates an srs-based eNB and an srs-based UE, and a srs-based EPC, and provides shell access to the experimenter. As with other Powder experiment definitions, the experiment sets up a working link but leaves it to the experimenter to use the srs software tools or GUI to send data and perform measurements. Because the nodes are fixed, and handover is not supported by srsLTE, it is difficult to perform over-the-air experiments at this time that match scenarios that users would construct with ns-3. We have hypothesized that the Powder srsLTE could be instrumented to check the link performance (MCS selection, BLER performance) as a function of received signal strength, by stepping down the transmit power, and compare with ns-3 corresponding results, but

such experiments are for further study, and may involve making modifications to srsLTE to expose additional trace data.

# 6 Summary and Future Work

ns-3 and emerging wireless testbeds such as Powder are complementary research tools. This report has illustrated how ns-3 can be used to create virtual nodes within the Powder testbed, to extend the Powder topology beyond what can be physically realized. ns-3 has the capability to emulate a large number of nodes, and with recently introduced emulation improvements, can saturate Gigabit Ethernet links with traffic generated from within the emulation. We have also described how data gathered in Powder can be used to build new ns-3 simulation models, and how experimental data may in the future be useful to validate or adjust the models being used in ns-3.

This project has resulted in the creation of a new ns-3 extension module for Powder support (scripts, documentation, and ns-3 programs) that is published on the ns-3 App Store. Future users should be able to more easily instantiate ns-3 emulations with this support, and code provided therein can serve as a template for other similar use cases. We have also created, at the Powder site, an ns-3-enhanced disk image that will reduce the need for future users to build up a generic node from scratch.

As Powder develops further, more validation work of the ns-3 LTE models can possibly be performed, and new models based on new Powder capabilities and empirical data can be generated. A similar approach can also be taken for other PAWR testbeds that are expected to come online over the next year.

The opening of further wireless testbeds will lead to further potential issues for users as they migrate from one research environment to another, as each framework likely uses different orchestration systems, web interfaces, and even terminology. There is an opportunity to try to develop an experiment control framework common to all of these environments. Standardizing on experimental control frameworks is challenging because users tend to like to construct their own based on the programming languages that they are comfortable with. We see an opportunity for a tool such as *sem*, introduced above in Section 3, to possibly be reused across ns-3 and testbeds, due to its architecture (being fairly decoupled from ns-3) and its implementation in Python. A key to establishing any such framework and driving usage will be to develop a set of clearly documented and robust examples that can be easily cloned and modified by users.

# Bibliography


[1] Powder use cases, 2020. URL https://powderwireless.net/use.

[2] Powder support module for ns-3, 2020. URL https://apps.nsnam.org/app/powder-support.

[3] ns-3 network simulator, 2020. URL https://www.nsnam.org.

[4] ns-3 model library, 2020. URL https://www.nsnam.org/docs/release/3.31/models/html/index.html.

[5] Powder-renew project, 2020. URL https://powderwireless.net.

[6] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim. Core: A real-time network emulator. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pages 1–7, 2008.

[7] R. Doost-Mohammady, O. Bejarano, L. Zhong, J. R. Cavallaro, E. Knightly, Z. M. Mao, W. W. Li, X. Chen, and A. Sabharwal. Renew: Programmable and observable massive mimo networks. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 1654–1658, Oct 2018. doi: 10.1109/ACSSC.2018.8645391.

[8] A. Esquivel, D. K. Ufuktepe, R. Ignatowicz, A. Riddle, C. Qu, P. Calyam, and K. Palaniappan. Enhancing network-edge connectivity and computation security in drone video analytics. In *IEEE Applied Imagery Pattern Recognition (AIPR) Workshop (To Appear)*, Oct. 2020.

[9] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith. Srslte: An open-source platform for lte evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, WiNTECH '16, page 25âĂŞ32, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342520.

[10] P. Imputato, S. Avallone, and T. Pecorella. Network emulation support in ns-3 through kernel bypass techniques. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS 2017, page 259âĂŞ260, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450363464.

[11] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304092.

[12] D. Magrin, D. Zhou, and M. Zorzi. A simulation execution manager for ns-3: Encouraging reproducibility and simplifying statistical analysis of ns-3 simulations. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '19, pages 121–125, New York, NY, USA, 2019. Association for Computing Machinery.

[13] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet. Openair-interface: A flexible platform for 5g research. *SIGCOMM Comput. Commun. Rev.*, 44(5): 33âĂŞ38, Oct. 2014. ISSN 0146-4833.

[14] H. Patel, H. Hiraskar, and M. P. Tahiliani. Extending network emulation support in ns-3 using dpdk. In *Proceedings of the 2019 Workshop on Ns-3*, WNS3 2019, page 17âĂŞ24, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450371407.

[15] A. Quereilhac, M. Lacage, C. Freire, T. Turletti, and W. Dabbous. Nepi: An integration framework for network experimentation. In *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, pages 1–5, 2011.

[16] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *IEEE Wireless Communications and Networking Conference, 2005*, volume 3, pages 1664–1669 Vol. 3, 2005.

[17] L. Rizzo. Revisiting network i/o apis: The netmap framework: It is possible to achieve huge performance improvements in the way packet processing is done on modern operating systems. *Queue*, 10(1):30âĂŞ39, Jan. 2012. ISSN 1542-7730.

[18] *The POWDER Manual*. University of Utah, 2020. URL `https://docs.powderwireless.net`.

[19] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.