OneLNK: One Link to Rule Them All: Web-based Wireless Experimentation for Multi-vendor Remotely Accessible Indoor/Outdoor Testbeds

Mohammad M. R. Lunar*, Jianxin Sun*, John Wensowitch†, Michael Fay*, Halit Bugra Tulay‡, Venkat Sai Suman Lamba Karanam*, Brian Qiu§, Deepak Nadig*, Garhan Attebury*, Hongfeng Yu*, Joseph Camp†, Can Emre Koksal‡, Dario Pompili§, Byrav Ramamurthy*, Morteza Hashemi¶, Eylem Ekici‡, Mehmet C. Vuran*

*Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA; †Electrical and Computer Engineering, Southern Methodist University, Dallas, TX,; ‡ Electrical and Computer Engineering, The Ohio State University, Columbus, OH; § Electrical and Computer Engineering, Rutgers University–New Brunswick, NJ; ¶ Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS.

Corresponding:mlunar@cse.unl.edu

ABSTRACT

As evolving wireless network architectures become more diverse, complex, and interdependent, and equipment costs prohibit broad access to such networks, remotely accessible experimental testbeds are gaining interest in recent years in wireless communication and networking research. This interest has exacerbated in 2020 and became a vital need during the current global pandemic. However, providing end-users of various educational backgrounds access to radio devices from a heterogeneous set of vendors is challenging. This paper introduces OneLNK, a remotely accessible testbed consisting of radio devices from three different vendors and developed using open source cloud-native technologies. End-users can access the functionalities of OneLNK from a single webpage without any local installations. Using the web URL, users can operate radio devices, set experiment parameters, observe results in real-time, and save generated experiment data for all radio devices. The interactive web UI and its working mechanism for supporting radio equipment are covered with specific experiment capabilities. A diverse set of radio equipment (mmWave, sub-GHz SDR, and sub-6GHz SDR) are facilitated to explain these capabilities. Moreover, measurements of path loss, Received Signal Strength (RSS), and Signal-to-Noise Ratio (SNR) using devices from three different vendors operating on a vast spectrum (568 MHz, 5.8 GHz, and 60 GHz) are reported. The majority of the remotely accessible OneLNK platform was developed remotely during the pandemic by a team of experts from five U.S. states.

CCS CONCEPTS

• **Hardware → Wireless integrated network sensors**; • **Networks → Network experimentation**.

KEYWORDS

USRP, IRIS, Remote Wireless, Testbed, VM, Web Technologies

1 INTRODUCTION

Evolving capabilities of 5G/6G networks and their software-defined architectures lead to diversification in operational frequencies, network equipment, and high-end radios from different vendors to be integrated into a single network. Thus, access to fully functional wireless experimentation platforms for research and evaluation purposes has become cost-prohibitive, motivating the wireless networking community to develop large-scale experimentation platforms that can be remotely accessible by the community. Coincidentally, the global pandemic due to the novel coronavirus has led to extensive university operations restrictions in campus lockdowns and remote learning activities. For example, in Spring 2020, along with many universities, the University of Nebraska-Lincoln (UNL) switched to remote teaching. This transition places a significant burden on instructors in restructuring online higher education classes to maintain the same quality as face-to-face classes. An essential aspect of this burden is transforming lab and project components of specific courses requiring equipment access. More specifically, part of this work focuses on the challenges in teaching wireless communication, wireless networking, and general networking classes that require remote access to radios, computing devices, and networking equipment to conduct lab modules and semester projects. Consequently, instructors need access to remotely accessible platforms that can be shared across students, classes, departments, and universities to maintain high quality in higher education.

We develop a cloud-native set of tools underlying a web-based User Interface (UI) to address these issues in this work. A simple web browser can operate this interactive UI, hence a diverse set of radios, without additional local software tools or network configurations. Besides, end users can interface with radio devices from different vendors. The end users can access the testbed and run

WiNTECH'21, January 31-February 4, 2022, New Orleans, LA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8703-3/22/01…$15.00
https://doi.org/10.1145/3477086.3480835

experiments even from a low-end mobile device. To facilitate real-world wireless networking experiments, we present OneLNK, an *indoor* testbed on the UNL campus with radio devices from *three* different merchants operating on sub-1GHz, sub-6GHz, and mmWave spectrum. Traditionally, to operate these radios remotely, users need to configure a specific set of software tools, device drivers, and network settings. It becomes much complicated when radio devices from multiple vendors need to work together. A remote setup generates an extra overhead to the learning curve for the user group not vastly familiar with these radio devices. These challenges are addressed through the OneLNK software architecture.

In the rest of the paper, we discuss related work in remotely accessible networking testbeds. Then, we present OneLNK in detail to demonstrate the experimentation capabilities of the developed testbed:

- Both software and hardware architectures of OneLNK are presented.
- The robustness of OneLNK is demonstrated with exemplary experiment scenarios. A landing webpage for OneLNK is also made available for interested users.[1]
- We provide results from multi-frequency link measurements[2] and containerized open-source RAN deployments on OneLNK.

2 RELATED WORK

Several initiatives for remotely accessible testbeds are observed in the literature. Each of them has specific functionality scopes and user access tools. On the one hand, we observe testbeds for empirical cloud experimentation (e.g., Chameleon Cloud [6], CloudLab [1]). Users can reserve compute and storage resources for performing cloud operations in these testbeds. On the other hand, Emulab [3] and PlanetLab [13] focus on network service experiments in their platform. Users need to install a specific set of software for utilizing the full functionalities of these testbeds. DeterLab [7] provides cybersecurity testing facilities to the user. None of these testbeds offer substantial wireless communication research resources.

Global Environment of Network Innovations (GENI) [4] provides an experiment facility with 4G LTE base stations and customizable user devices. Although account management and resource reservations can be scheduled through a web portal, users need to use python's geni-lib for designing custom tools. An initiative of Rutgers University named ORBIT [10] arranges 400 indoor nodes with programmable wireless network equipment. Users need to install custom tools in the testbed Virtual Machines (VMs) for accessing these wireless nodes. In addition, prior knowledge of the orbit management framework (OMF) is also required. Similarly, Fed4FIRE+ (https://www.fed4fire.eu) testbed also uses a Java-based desktop tool (jfed) for ensuring full functionalities to the user.

Moreover, federated testbeds are also available in the literature. OneLab [2] is a federated testbed that provides features of FIT [9], CloudLab [1], and CorteXlab (https://www.cortexlab.fr). A portion of the FIT testbed supports web portal-based access. For other cases, custom software is used. Another federated testbed is Fabric (https://fabric-testbed.net), which is currently under development.

[1]https://onelnk.unl.edu
[2]Source code available at https://github.com/UNL-CPN-Lab/OneLNK.git/

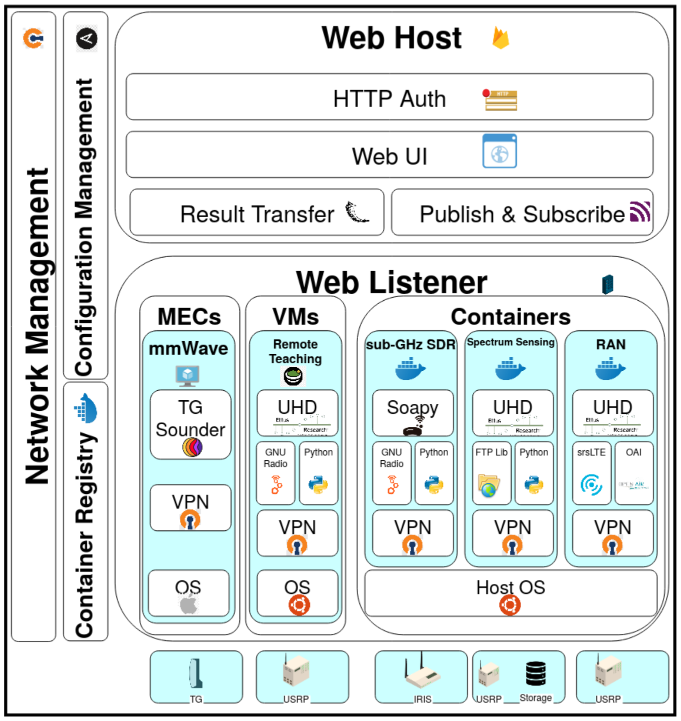


Figure 1: OneLNK Software Architecture.

The National Science Foundation (NSF), through the Platforms for Advanced Wireless Research (PAWR) program, supports robust remote wireless testbeds. Currently, three testbeds are either operational or under development: the Platform for Open Wireless Data-driven Experimental Research (POWDER) [5], Cloud Enhanced Open Software Defined Mobile Wireless Testbed for City-Scale Deployment (COSMOS) [11], and Aerial Experimentation and Research Platform for Advanced Wireless (AERPAW) (https://aerpaw.org).

Both POWDER and COSMOS provide web-based facilities for registration, resource allocation, and resource status monitoring. Most functionalities that do not require any graphical user interface (GUI) can be accessed through ssh for these testbeds. However, both testbeds need additional software tools to be installed locally to utilize their full capability. A few examples of additional required tools are X11 (https://docs.powderwireless.net/users.html; 5.1.4) and GENI Resource Specification (RSpec) for POWDER, and OMF, a control, and management framework for networking testbeds, for COSMOS. As of this writing, AERPAW is under development. All these testbeds are summarized in Table 1.

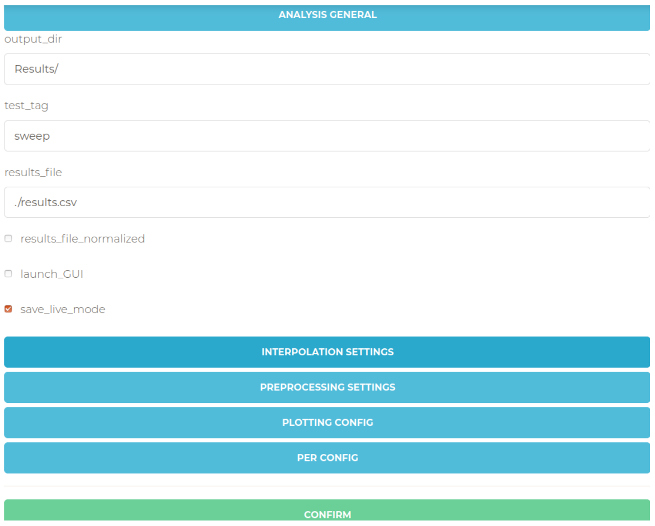
Although several remotely accessible testbeds are available, none of these solutions provide complete web UI-based experimentation functionalities with cloud-native software tools to the best of our knowledge. In this paper, we aim to bridge this gap. Moreover, we present a set of experiments that a single web URL can entirely execute.

3 MOTIVATION

This section discusses the expected functionalities of a remote testbed for performing wireless communication and networking

Table 1: Functionalities and Requirements of Existing Testbeds

Testbed	URL	Functionalities			Required Software Other than Web Boowser
		Cloud	Networking	Wireless Comm.	
Chameleon Cloud	https://whttps://ww.chameleoncloud.org	✓	✓	✗	None
CloudLab	https://www.cloudlab.us	✓	✓	✗	Vagrant and Virtualbox
Emulab	https://www.emulab.net	✓	✓	✗	X11 client
PlanetLab	https://planetlab.cs.princeton.edu	✗	✓	✗	PlMan, Link emulator
DeterLab	https://www.isi.deterlab.net	✓	✓	✗	Magi core
GENI	https://www.geni.net	✓	✓	✓	geni-lib, jFed
ORBIT	https://www.orbit-lab.org	✓	✓	✓	OMF, X11
Fed4FIRE+	https://www.fed4fire.eu	✓	✓	✓	jFed
OneLab	https://onelab.eu	✓	✓	✓	OneLab experiment control tool
CorteXlab	https://wiki.cortexlab.fr	✓	✓	✓	X11
POWDER	https://powderwireless.net	✓	✓	✓	X11, Geni RSpec
COSMOS	https://www.cosmos-lab.org	✓	✓	✓	OMF
OneLNK	https://onelnk.unl.edu	✓	✓	✓	None



output_dir: Results/

test_tag: sweep

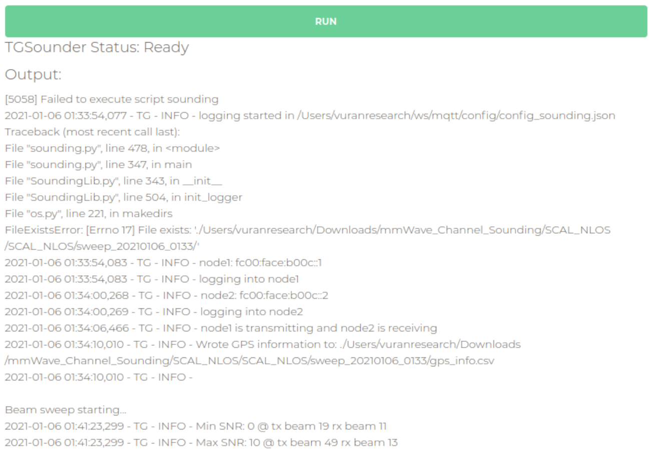
results_file: ./results.csv

☐ results_file_normalized

☐ launch_GUI

☑ save_live_mode

Figure 2: Configuration screen for the TG Sounder.



TGSounder Status: Ready

Output:

```
[5058] Failed to execute script sounding
2021-01-06 01:33:54,077 - TG - INFO - logging started in /Users/vuranresearch/ws/mqtt/config/config_sounding.json
Traceback (most recent call last):
File "sounding.py", line 478, in <module>
File "sounding.py", line 347, in main
File "SoundingLib.py", line 343, in __init__
File "SoundingLib.py", line 504, in init_logger
File "os.py", line 221, in makedirs
FileExistsError: [Errno 17] File exists: './Users/vuranresearch/Downloads/mmWave_Channel_Sounding/SCAL_NLOS
/SCAL_NLOS/sweep_20210106_0133/'
2021-01-06 01:33:54,083 - TG - INFO - node1: fc00:face:b00c::1
2021-01-06 01:33:54,083 - TG - INFO - logging into node1
2021-01-06 01:34:00,268 - TG - INFO - node2: fc00:face:b00c::2
2021-01-06 01:34:00,269 - TG - INFO - logging into node2
2021-01-06 01:34:06,466 - TG - INFO - node1 is transmitting and node2 is receiving
2021-01-06 01:34:10,010 - TG - INFO - Wrote GPS information to: ./Users/vuranresearch/Downloads
/mmWave_Channel_Sounding/SCAL_NLOS/SCAL_NLOS/sweep_20210106_0133/gps_info.csv
2021-01-06 01:34:10,010 - TG - INFO -

Beam sweep starting...
2021-01-06 01:41:23,299 - TG - INFO - Min SNR: 0 @ tx beam 19 rx beam 11
2021-01-06 01:41:23,299 - TG - INFO - Max SNR: 10 @ tx beam 49 rx beam 13
```

Figure 3: Runtime status operation for the TG Sounder.

experiments. Due to the diversity in user groups and experiment requirements, a robust functionality set is necessary for the smooth operation of these testbeds. We classify these functionalities into four: remote operability, user access, experimental diversity, and open design approach. An overview of each set is presented here.

Remote Operability: We discuss remote operability based on three aspects: remote access, remote monitoring, and remote maintenance. Remote access provides features and flexibility to a testbed user. A user is not expected to physically configure and set up a device in a remote testbed in most cases. Therefore, all components of equipment need to be fully configurable through a remote setup. In addition, the user should have access to the output data and log files through a remote network.

Similarly, testbed system administrators also need functionalities that limit physical presence in the testbed site as much as possible. Dependency of physical site maintenance and troubleshooting can introduce a bottleneck to the overall site reliability and service availability. As a result, remote health monitoring and site maintenance for hardware and software units are necessary for remote wireless testbeds.

User Access: As mentioned in Section 1, the demand for remote wireless testbed continuously increases among diverse users. As a result, it is crucial to provide a standard user access workflow, easily utilized by users from all backgrounds. It might not be appropriate if a testbed requires a collection of tools, drivers, and packages on the client machine for gaining access to the testbed. Therefore, the web-based user interface can be a more appropriate access mechanism for involving different user communities to use the testbed. Moreover, many experiment scenarios require hardware programming (e.g., field-programmable gate array (FPGA) of SDR). The necessary development environment for these programming needs to be provided by the testbed so that users can directly concentrate on the programming instead of worrying about tools and drivers for preparing the development environment.

Experimental Diversity: The types of devices and experiment scenarios might also have a wide variety for a specific testbed. It is

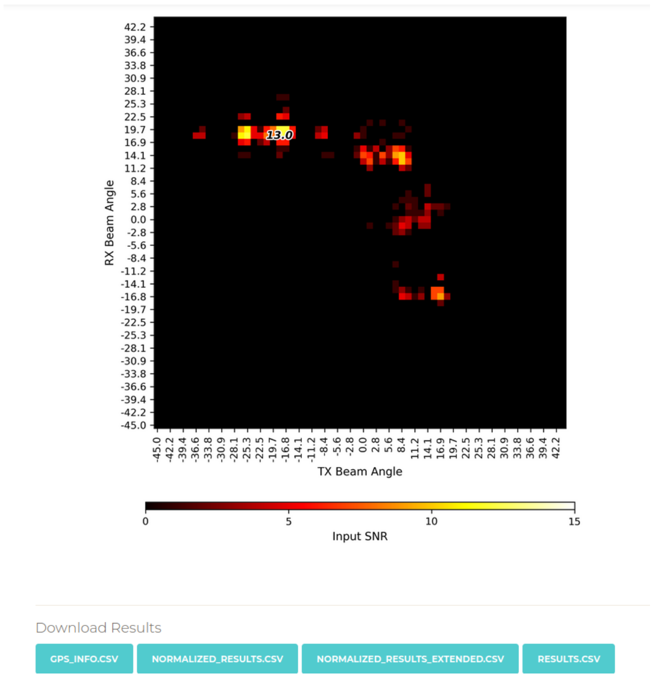


Figure 4: TG Sounder experiment results with the downloadable file link.

also essential to provide a generic user interface for the significant capabilities of these various devices and experiment scenarios. Otherwise, the learning curve might be steep for specific users. Besides, necessary drivers and software packages are needed for each radio device. Availability and ease of access to such packages will ensure users' smooth operation of each radio device's experiments.

Open Design Approach: Wireless communication and networking is a rapidly evolving research field. Hence, the radio devices, antennas, operating frequencies, and network topology must be easily upgraded. Therefore an open design approach where new devices and network scenarios can be comfortably implemented is a must for the successful testbed operation in the long run. A clearly explained bring your own device (BYOD) policy is also very helpful to achieve this goal.

4 ONELNK ARCHITECTURE AND FUNCTIONALITIES

In this section, we discuss the architectural overview and functionalities of OneLNK. First, software (Section 4.1) and hardware (Section 4.2) architectures are presented, followed by how this architecture provides the expected functionalities in Section 3.

4.1 Software Architecture

The OneLNK software architecture contains four main subsystems: web host, web listener, management modules, and testbed resources (Fig. 1).

4.1.1 Web Host. Web host encapsulates user authentication, ssh, and web portals. Web portals provide a common interface for the interaction and monitoring of all other subsystems. The web interface provides users with the following tools:

Cross-platform Application: A web application is implemented to run on various platforms with responsive web design. *Intuitive and Customized UI:* A user-friendly interface is designed to help access OneLNK efficiently, even for novice users. The web user interfaces are designed for specific characteristics of various radio devices and capabilities. *Configuration:* The user can update the configurations of the testbed components (Fig. 2). *Operation:* The user can launch the experiment and monitor its runtime status in real-time (Fig. 3). *Result Visualization:* The WebUI will render preliminary visualization of the experimental result for quick evaluation (Fig. 4). *Raw Data Retrieving:* The user can download all relevant raw resulting data to the local machine for future analysis.

Web applications provide a user portal as the primary interface between the user and the testbed. The web-based user portal reveals the status and activities of the testbed and provides user documentation and knowledge articles about directly utilizing the platform and recent scientific outcomes. This portal also redirects users to the desired devices that they want to access for the experiment.

The architecture of the existing web application includes three main parts, WebUI [8], servers (MQTT broker (https://mqtt.org) and Flask (https://flask.palletsprojects.com/en/1.1.x)) and the daemon process running on the testbed node. The WebUI provides a user-friendly interface to collect user input and demonstrate a visual representation of the experimental result. The interface is specifically designed by considering the characteristics of different radio devices and their experimental environments to provide intuitive and effective user interaction with the hardware. The daemon process of the testbed node listens and parses the message received to execute diverse operations of the software and hardware of the testbed. Two types of servers are used here to realize the communication between the WebUI and testbed node. The MQTT broker server is implemented for real-time message exchange through the MQTT protocol, while the Flask server transfers the resulting data file through HTTP protocol. The web application, composed of multiple components, provides various functions to users for practicing experiments on the testbed efficiently and effectively. The WebUI interface allows the user to interact with various hardware and software resources on the testbed.

The back-end server application is implemented using the Angular framework and deployed on the Firebase platform from Google (https://firebase.google.com). The server application handles several primary functionalities: authentication, user management, resource monitor and experiment allocation. User authentication is provided through HTTP basic auth (Username/Password) [12]. Once a user is authorized, authentication will expire automatically after a preset time of inactivity or manually logging out. More than one user can log in to the system through the WebUI, and the user management functionality manages their activities. Multiple users can perform experiments on different radio devices concurrently. The system monitors the status of various radio devices and allocates available radio devices as requested to the user.

4.1.2 Web Listener. Web listener receives user instructions from the web hosts and executes them to run the experiment. In OneLNK, the web listener utilizes three major components: containers, VMs, and multi-access edge computing (MEC).

Containers: Containers provide an encapsulated platform where all required software and driver packages and configurations are combined and executed quickly and easily on different computing platforms. We utilize Docker containers for this purpose.

VMs: Encapsulated VMs containing required packages and configurations are also available for OneLNK users. The Holland Computing Center (HCC) hosts both remote desktop and Guacamole VM from its clusters. Browser-specific third-party add-on supports (e.g., Google Chrome Remote Desktop) are also available for users when required.

MEC: OneLNK also allows users to utilize MECs for applications where virtualization constrains either implementation or performance of the experiment. Similar to containers and VMs, the required software packages and network settings are pre-configured for these MECs.

To support devices from multiple vendors, primitive driver applications for all devices on the platform, including UHD, SoapySDR, and TG Sounder, are available for each web listener component. Furthermore, multiple versions or tags for common driver versions are also included. The details of these APIs are summarized below.

UHD: It is the open-source radio API for operating USRPs developed and maintained by Ettus research (https://kb.ettus.com/UHD). This API provides the programmability of USRP devices from host machines. UHD provides operation for software executed by both external host computing devices and internal FPGAs.

TG Sounder: It is the API for operating TG mmWave channel sounders. This API ensures access to the TG sounder from a host computing device with a few simple command-line instructions. The API takes experiment parameters from simple JSON configuration files, which are easily configurable by the end-user. The current version of the API only supports control PCs operated by Mac OSX.

SoapySDR: It is an open-source API and library that works with various software-defined radio vendors. It is a versatile framework that can directly interface with other workflows, such as GNURadio (https://gnuradio.org). SoapySDR's (https://github.com/skylarkwireless/sklk-soapyiris) API simplifies capturing and generating samples while still providing the flexibility to implement complex signal processing and interface with various software packages. SoapySDR's primary role within the testbed is to provide software support for the Skylark IRIS SDR platform.

OneLNK provides the facility to access the testbed for the users without mastering all details about the device and network configurations in the testbed. In addition to these libraries, users can exploit their own software packages and libraries based on their requirements. OneLNK does not restrict any user to utilize a software package, custom-developed tools, custom libraries into its platform.

4.1.3 *Management Modules.*
Management modules are in the leftmost portion of Fig. 1. These modules provide common technologies that experiments might leverage for user convenience or platform flexibility. The network management module provides authentication between edge compute clusters of radio devices and UNL's VPN server. It is ensured by a standard public key infrastructure (PKI) system of OpenVPN 2.0 (https://openvpn.net). The compute clusters are directly connected with radio devices and work as the OpenVPN client. Necessary certificates and private keys are installed on the client machine by the site admin before starting web access. Required network configurations are also available and configured with automated scripts by this module. The configuration management module handles common configurations. It leverages open-source tools like Ansible (https://www.ansible.com). These configurations provide the capability to execute complex experiment scenarios to the experimenters easily. Container registry houses containers of preconfigured software and network packages for authenticated OneLNK users. This registry is hosted by UNL ITS (https://its.unl.edu) and can be accessed similarly to Docker Hub (https://hub.docker.com).

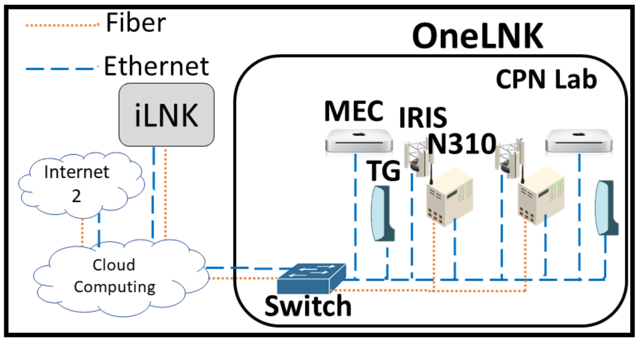


Figure 5: OneLNK Hardware Architecture.

4.1.4 *Testbed Resources.*
OneLNK contains RF-based devices with supporting resources to run experiments on the platform: data storage and administration/versioning. This enables the testbed resources to be scaled or rescheduled independent of each of the above layers. For TG sounders, the output can be directly downloaded from the web page after completing the experiment. The user can access either a physical or a web-based VM for the experiment for the other devices. Raw data and log files can be uploaded to a cloud from that VM. Necessary API and tools to connect an FTP cloud server from the VM are also available for the users. The bottom part of Fig. 1 presents these resources.

4.2 Hardware Architecture
Fig. 5 depicts a high-level hardware architecture of the current OneLNK deployment. OneLNK resides in the Cyber-physical Networking Laboratory at UNL, where three different pairs of radio devices are deployed on the east and west walls, at the height of 9′. Fig. 6a). They are USRP N310 from Ettus (https://www.ettus.com), Terragraph (TG), mmWave channel sounder (https://terragraph.com), and IRIS SDR from Skylark (https://skylarkwireless.com). USRPs are installed with 4x1 linear antenna arrays to enable MIMO applications. The operating range of the USRPs is 10 MHz – 6 GHz and the antenna array is constructed by dual-band (2.4 and 5 GHz) omnidirectional antennas. The operating frequency of the TG sounder is 60 GHz. The IRIS-030 SDR platform is paired with the IRIS-FE-02-UHF frontend radio module to enable communication in the TV spectrum (470 – 700 MHz). IRIS radios can function standalone as standard user equipment or paired with a Faros Hub to enable massive MIMO applications. The overall hardware architecture is shown in Fig. 5, where each radio device and its corresponding

computing nodes are depicted according to physical site deployment. Moreover, the radios are connected with the network switch to ensure remote access.

Network devices are installed in each site to ensure both inter-site device connectivity and remote access. These devices serve to route traffic between edge sites and the core and provide for firewall, Virtual Private Network (VPN), and management network segmentation needs.

4.3 OneLNK Functionalities

Based on the architectural overview discussed in this section, we summarize the functionalities of OneLNK as follows:

Remote Operability: All OneLNK devices are remotely accessible through the Internet. The authentication and security of these devices are ensured through web logon and VPN. In addition, an admin portal is developed to monitor each device's status. The auto error notification is generated and uploaded to a monitoring email server by radio devices. Remote accessible PDUs are installed for remote power cycling of the radio devices. Radio devices are connected with a remote accessible compute node via a universal asynchronous receiver transmitter (UART) for device health monitoring. Required sites are equipped with LTE dongles. LTE provides the Out-of-band (OOB) network connectivity for these compute nodes and ensures the access of the devices during a possible network failure.

User Access: As discussed in Section , 4.1.1 the user can access OneLNK using a web browser with a single URL.

Experimental Diversity: Section 4.1.2 provides required packages and environments to run various experiments in the OneLNK platform. A OneLNK user can execute experiments with minimum configuration and installation overhead using these modules using those pre-configured containers and VM.

Open Design Approach: We have already integrated our previously deployed testbed iLNK [15] with the OneLNK platform. All radios of iLNK can be accessed and utilized with OneLNK functionalities. Full functionalities of each iLNK radio are available for any OneLNK users, and any software packages or network configuration of OneLNK can be entirely utilized for those iLNK radios. A detailed BYOD policy is also prepared further to include new radio devices in the OneLNK platform. The inclusion of new radio devices in OneLNK is made straightforward. The system admin can integrate any networked radio device into this platform with a static IP address assignment and OpenVPN configuration.

5 EXPERIMENT CAPABILITIES

In this section, the experiment capabilities that OneLNK enables are described. Fig. 6 shows the deployment of the radio devices in OneLNK. The functionalities of radio devices from each vendor and their possible use cases are described here.

5.1 mmWave Channel Sounding

TG channel sounder is designed to measure and model 60 GHz mmWave channels (https://terragraph.com). The sounder can measure the channel characteristics such as the directional path loss, delay spread, channel impulse response, and Signal-to-Noise Ratio (SNR). The users can determine the operation of the sounder by configuring the parameters via the WebUI, as shown in Fig. 2. The sounder parameters include the direction of transmission, modulation, and coding scheme (MCS) and the antenna masks that provide a mechanism to control the beamwidth. After each experiment, the WebUI provides the raw measurement data to users to allow maximum flexibility in analyzing data. The WebUI also provides the plots of path loss, received power, and SNR besides the raw data.

5.2 IRIS SDR Communication

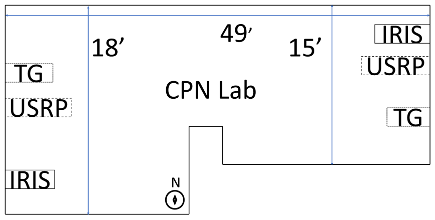
The IRIS SDR is a multipurpose field programmable radio platform covering a wide range of operating frequencies with its assortment of interchangeable analog front-ends. In addition, the IRIS SDR offers the unique capability of synchronizing across multiple IRIS devices, which enables scalable next-generation MIMO applications. When paired with open-source software, such as SoapySDR (https://github.com/pothosware/SoapySDR/wiki), various applications can be developed and tested within the SDR framework. The general-purpose nature of SoapySDR enables various interaction methods, including C++, C, and Python library bindings, as well as plugins into existing toolsets such as GNURadio's popular signal processing flowgraphs. SoapySDR provides the libraries and APIs to generate and capture RF samples while maintaining the flexibility to implement complex signal processing complemented by a diverse open-source community of existing software solutions. These third-party libraries cater to complex RF signal processing as well as simplifying user access and interaction.

Webhooks are implemented to manage the underlying system for remote access by leveraging the versatile nature of the IRIS hardware and SoapySDR software framework. Utilizing existing web technologies such as HTTP, JavaScript, and MQTT, web applications are deployed to provide users an intuitive method of interacting with the IRIS SDR hardware and underlying software libraries. MQTT provides an asynchronous publish-subscribe pipeline of radio configuration data to the back-end server while simultaneously streaming resulting data to the web front-end. Generic front-ends are designed using HTTP and JavaScript to allow users to interact with various experiments without rewriting code. The back-end applications are developed using existing SoapySDR APIs to provide the MQTT calls for publishing and subscribing to data.

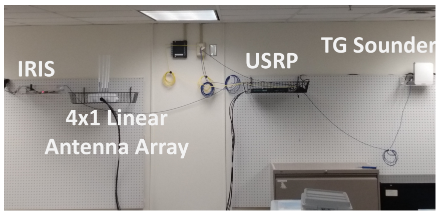
Furthermore, by leveraging gr-soapy, the GNURadio wrapper for SoapySDR, an end-user can augment an existing design process by VMs. Users familiar with the GNURadio flowgraph design process can quickly and easily integrate existing designs into the framework. While those who require the level of control or performance that a full programming language provides can just as easily adapt and deploy their code using one of the SoapySDR APIs.

5.3 Spectrum Sensing

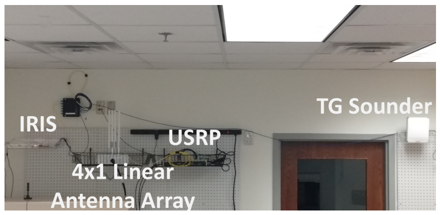
Spectrum sensing uses a fully automated spectrum capturing solution to periodically and persistently collect spectrum samples and store the sensing data in the cloud. Once scheduled with the container, the user does not need to interact with the testbed (**schedule-and-forget**), and the runtime manager periodically executes spectrum sampling tasks. Each site of the OneLNK testbed is capable of



(a) Testbed layout



(b) Testbed deployment (West wall)



(c) Testbed deployment (East wall)

Figure 6: OneLNK indoor testbed.

executing spectrum sensing experiments with USRP N310. Spectrum sensing data can immediately be uploaded to the cloud for post-processing. The spectrum sensing page of the web URL provides real-time visualization of data captured in the cloud.

5.4 Remote Teaching

OneLNK provides remote experimentation supports for wireless researchers and facilitates learning opportunities for students in basic and advanced wireless networking courses. During Spring 2021 semester, OneLNK, in collaboration with HCC, provides remote access to all students of a wireless communication networks course through Guacamole VM (https://guacamole.apache.org). The VMs are hosted inside a cluster of HCC named Anvil. Each VM is preconfigured with all necessary drivers, software, and networking package by Ansible (https://www.ansible.com). Therefore, each VM has GNU Radio, UHD, and OpenVPN clients with necessary certificates and keys inside an Ubuntu 20.04 OS to access OneLNK devices without worrying about these configurations directly. In addition, each VM also has preinstalled TigerVNC (https://tigervnc.org) server and Apache Tomcat (http://tomcat.apache.org) server. As a result, students can access this facility with a single click to the VM and run their course experiments.

Previously, course students had to spend a significant amount of time on their course project to properly install all software packages with correct driver versions and configure the VPN network correctly. After introducing this new remote learning mechanism, this initial configuration overhead is significantly reduced, and students can fully concentrate on their course project from the beginning of the course.

5.5 RAN

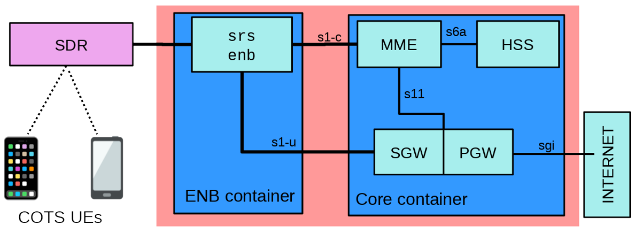


Figure 7: RAN in containers structure

The software architecture described in Fig. 1 allows us to flexibly and reliably set up a RAN network and perform end-to-end (e2e) experiments with consumer smartphones. A common issue with many open-source RAN platforms such as srsLTE (https://youtu.be/82IElHovrVw), and Open Air Interface (OAI) (https://youtu.be/p8vhtMI-UzA) requires a sophisticated hardware setup with complex networking. Containerization in this platform permits reproducible and configurable containers to replace the laborious hardware setup and network configurations that vary between experiments and individual setups. Through the power of virtualization and networking, containerization removes this tight coupling and brings benefits to stateless cloud management, thus guaranteeing reproducible RAN software configuration to research. Containerized RAN allows for new types of provisioning of resources at the PHY level of communication (raw IQ, modulation, channel coding, and equalization) to the high-level MAC level (access management) class of novel RAN applications in fields like robotics, IoT, and drones. An energy-efficient resource allocation RAN scheme is depicted in [14].

6 RESULTS

This section summarizes results from multi-frequency link measurements and open-source deployments using the OneLNK platform. The link measurements using three different RF devices and an open-source RAN deployment are presented next. We choose a generic experiment that can be executed in all three devices of OneLNK. Here the goal is to observe channel behavior at three different frequency bands by exploiting a similar experiment with these radio devices.

6.1 Link Measurements

We measure path loss, received signal strength (RSS), and signal-to-noise ratio (SNR) for three experiments. Each experiment utilizes specific hardware set from a vendor and represents a particular wireless networking channel. The experiments are executed with mmWave channel sounder at 60 GHz, USRP N310 at 5.8 GHz, and IRIS SDR at 568 MHz. All three tests are conducted on the OneLNK site.

Although these radios measure the same quantity, the experiment settings had a few differences due to the available modules in these devices. For example, mmWave utilizes IEEE 802.11*ad* scheme for its measurement. On the other hand, IRIS and USRP use a basic OFDM Tx-Rx module for the measurements. The Effective Isotropic Radiated Power (EIRP) was also set differently for ensuring connectivity to all these channels. Since lower EIRP values could not establish communication between TG sounders, we use 16dB higher EIRP for mmWave than USRP and IRIS radios. Tx and Rx linear distances between mmWave, IRIS, and USRP radios were 15.24m, 15.18m, and 15.05m, respectively.

These three link measurements are depicted in Fig. 8 for a one-minute interval. Since the distance between Tx and Rx nodes is not the same for all these tests, the path loss is normalized with the Tx-Rx distance. USRP observes 1.6dB more mean path loss per meter than IRIS. On the other hand, TG sounder observes around 1dB extra average path loss per meter than IRIS.

RSS measurements for these radio devices follow a similar trend. IRIS has the strongest mean RSS in these experiments. It is $23dB$ more than USRP. The impact of attenuation at a comparatively higher frequency (5.8GHz) is observed here. Due to a higher EIRP RSS of TG sounder average also very similar to USRP.

The nature of SNR is slightly different in this experimentation set due to the diversity in modulation schemes and communication channels. mmWave radio achieves the lowest SNR due to the higher propagation loss and multi-path in the indoor environment. On the other hand, we observe 4dB higher SNR for USRPs than IRIS radios, despite the higher operating frequency. USRP N310's additional low noise amplifier (LNA) can be the reason behind it.

In this one simple experiment, we can acquire comprehensive hands-on insights into real-world wireless communication. This can create a notable impact on the researchers and trainees in this focus area.

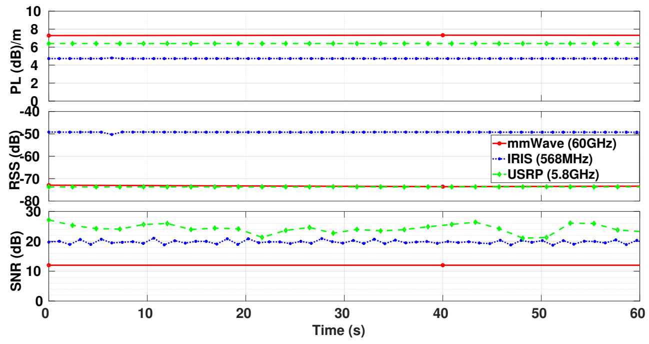


Figure 8: Path loss, RSS, and SNR from mmWave channel sounder, IRIS, and USRP.

6.2 RAN

We can configure two experimental profiles for end-to-end testing of bandwidth and latency on two different open-source LTE stacks (OAI and srsLTE) through our RAN testbed. These tests are done at two geographical locations: Rutgers and UNL. We were able to take the containers built from one site and transplant them to the other, resulting in quick reproducible experiments. A basic containerized OAI RAN architecture is given in Fig. 7, which requires a single host to run a traditionally multihost setup (https://youtu.be/p8vhtMI-UzA). In the srsLTE demonstration, we show the operation of a different COTS (https://youtu.be/82IElHovrVw). Within the srsLTE configuration, we were also able to swap out individual components, which have been containerized, between the two open stack frameworks to show inter-operability of the srsLTE base station and the OAI core network and vice versa. A quick comparison between the various configurations shows srsLTE to be a more versatile platform, accepting multiple COTS UE devices without issue and handling both an Apple smartphone and a Motorola phone. This is a feature of containers that allows for the quick deployment of these tests.

7 CONCLUSION

This paper introduces the OneLNK wireless testbed that governs all functionalities with a single web URL. During the current COVID-19 situation, the importance of a system-independent user access platform can significantly be appreciated. Achieving hands-on experiences with a real-world testbed can be a lot easier if the user can execute experiments without worrying much about drivers and software tools. The demonstration in this paper proves the feasibility of this approach for a fully remote access operation. Through cloud-native web technologies that are tightly integrated with state-of-the-art wireless experimentation platforms, we can expand the wireless experimentation end-user community to the likes of Alexa, Siri, and Google.

ACKNOWLEDGMENT

This work was supported in part by NSF CNS grants #1619285, #1731833, #1816938, #1817105, #1948511, and #1955561.

REFERENCES

[1] A. Akella. 2015. Experimenting with Next-Generation Cloud Architectures Using CloudLab. *IEEE Internet Computing* 19, 5 (2015), 77–81.
[2] L. Baron and et.al. 2015. OneLab: Major computer networking testbeds open to the IEEE INFOCOM community. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 3–4.
[3] C. Siaterlis et al. 2013. On the Use of Emulab Testbeds for Scientifically Rigorous Experiments. *IEEE Comm. Surveys Tutorials* 15, 2 (2013).
[4] A. Gosain and I. Seskar. 2017. GENI wireless testbed: An open edge ecosystem for ubiquitous computing applications. In *2017 IEEE PerCom Workshops*. 54–56.
[5] J. Breen et.al. 2020. POWDER: Platform for Open Wireless Data-Driven Experimental Research. In *Proc. ACM WiNTECH '20*. London, UK.
[6] J. Mambretti, J. Chen, and F. Yeh. 2015. Next Generation Clouds, the Chameleon Cloud Testbed, and Software Defined Networking (SDN). In *Proc. ICCCRI '15*. 73–79.
[7] J. Mirkovic and T. Benzel. 2012. Teaching Cybersecurity with DeterLab. *IEEE Security Privacy* 10, 1 (2012), 73–76.
[8] Jakob Nielsen. 1999. User Interface Directions for the Web. *Commun. ACM* 42, 1 (Jan. 1999), 65–72.
[9] O. Oubejja and et.al. 2019. Framework for PHY-MAC layers Prototyping in Dense IoT Networks using FIT/CorteXlab Testbed. In *IEEE INFOCOM 2019 Workshops*. 923–924.
[10] R. Beuran et al. 2011. Challenges of Using Wireless Network Testbeds: A Case Study on ORBIT. In *Proc. ACM WiNTECH*. Las Vegas, NV.
[11] D. Raychaudhuri and et.al. 2020. Challenge: COSMOS: A City-Scale Programmable Testbed for Experimentation with Advanced Wireless. In *Proc. ACM MobiCom'20* (London, UK).
[12] J. Reschke. 2015. *The 'Basic' HTTP Authentication Scheme*. RFC 7617. RFC Editor. 1–15 pages. https://tools.ietf.org/rfc/rfc7617.txt
[13] L. Tang, Y. Chen, F. Li, H. Zhang, and J. Li. 2007. Empirical Study on the Evolution of PlanetLab. In *Sixth Intl. Conf. on Networking (ICN'07)*.
[14] A. Younis, T. Tran, and D. Pompili. 2019. Energy-efficient resource allocation in C-RANs with capacity-limited fronthaul. *IEEE Transactions on Mobile Computing* (2019).
[15] Zhongyuan Zhao, Mehmet C. Vuran, Baofeng Zhou, Mohammad M.R. Lunar, Zahra Aref, David P. Young, Warren Humphrey, Steve Goddard, Garhan Attebury, and Blake France. 2021. A city-wide experimental testbed for the next generation wireless networks. *Ad Hoc Networks* 111 (2021), 102305.