Colosseum as a Digital Twin: Bridging Real-World Experimentation and Wireless Network Emulation

Davide Villa, Miead Tehrani-Moayyed, Clifton Paul Robinson,
Leonardo Bonati, Pedram Johari, Michele Polese, Tommaso Melodia
Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, U.S.A.
E-mail: {villa.d, tehranimoayyed.m, robinson.c, l.bonati, p.johari, m.polese, melodia}
@northeastern.edu

Abstract—Wireless network emulators are being increasingly used for developing and evaluating new solutions for Next Generation (NextG) wireless networks. However, the reliability of the solutions tested on emulation platforms heavily depends on the precision of the emulation process, model design, and parameter settings. To address, obviate, or minimize the impact of errors of emulation models, in this work, we apply the concept of Digital Twin (DT) to large-scale wireless systems. Specifically, we demonstrate the use of Colosseum, the world's largest wireless network emulator with hardware-in-the-loop, as a DT for NextG experimental wireless research at scale. As proof of concept, we leverage the Channel emulation scenario generator and Sounder Toolchain (CaST) to create the DT of a publicly available over-the-air indoor testbed for sub-6 GHz research, namely, Arena. Then, we validate the Colosseum DT through experimental campaigns on emulated wireless environments, including scenarios concerning cellular networks and jamming of Wi-Fi nodes, on both the real and digital systems. Our experiments show that the DT is able to provide a faithful representation of the real-world setup, obtaining an average similarity of up to 0.987 in throughput and 0.982 in Signal to Interference plus Noise Ratio (SINR).

Index Terms—Digital Twin, Wireless Channel Emulation, Experimental Wireless Research, Ray-tracing, Channel Sounding, Mobile Networking.

1 Introduction

Current wireless technologies are the key enablers of the digital world. They meet the ever-growing demand of connecting larger and larger groups of people, vehicles, wearables, robots—virtually anything; they enable hosts of applications previously unthinkable: from autonomous vehicles roaming space and the oceans to life-saving telemedicine. A lead actor in this wireless revolution is the 5th generation mobile network (5G), which has taken decisive steps toward redefining the cellular architecture. 5G provides unprecedented connectivity rates, capacity, and latency by opening and streamlining access to the Radio Access Network (RAN), using new frequency bands and advanced spectrum management techniques [2]. The development of 6G is already undergoing and is expected to achieve even higher capacity, lower latency, and increased bandwidth compared to currently deployed 5G systems [3, 4], enabling the true realization of the Internet of Things (IoT) and connecting over 30 billion devices by 2030 [5].

In this context, powerful experimental wireless platforms have been recently developed to provide an ecosystem for advanced wireless research through repeatable and reproducible experimentation and the creation of large datasets. These platforms are becoming the nexus of Artificial Intelligence (AI)-enabled wireless research, where researchers can design, develop, train, and test new solutions for Next Generation (NextG) wireless systems. Examples include the US NSF Platforms for Advanced Wireless Research (PAWR) program with its four at-scale, outdoor programmable platforms [6], and indoor testbeds including the Drexel Grid [7], ORBIT [8], and Arena [9].

While these testbeds provide good examples of indoor and outdoor wireless propagation environments, their scale can hardly capture the dynamics of real-world deployments. Also, their scope is limited to the physical environment where they are deployed. Alternatively, for site-independent wireless experimentation, researchers can use large-scale wireless emulation platforms. By emulating a virtually unlimited variety of scenarios, these instruments are becoming a key resource to design, develop, and validate networking solutions in quasi-realistic environments, at scale, and with a diverse set of fully-customizable Radio Frequency (RF) channel conditions, traffic scenarios, and network topologies [7, 10, 11]. An exemplary large-scale emulation-based wireless platform is Colosseum—the world's largest wire-

This is a revised and substantially extended version of [1], which appeared in the Proceedings of ACM WiNTECH 2022.
This work was partially supported by the U.S. National Science Foundation under grant CNS-1925601, by the U.S. Department of Transportation, Federal Highway Administration, and by OUSD(R&E) through Army Research Laboratory Cooperative Agreement Number W911NF-19-2-0221. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

less network emulator with hardware-in-the-loop [10].

Solution development and testing for NextG networks are in fact evolving toward integrating actual networked systems with a digital model that provides a replica of the physical network to be used for continuous prototyping, testing, and self-optimization of the living network. These *Digital Twins (DTs)* [12] are trending at the forefront of wireless research testing and prototyping [13]. Similar to the DTs used for some time in the industrial sector, a digital replica of a telecommunication network enables researchers to design optimized network architectures and to develop new AI-based features to further expand and enhance the capabilities of NextG systems. In contrast to the currently used simulation-based network planning tools, DT-based systems will be connected to real-world deployed physical subsystems with real-time feedback loops, providing high-fidelity design and planning platforms.

While recent research focuses on NextG telecommunication systems as a technology for reliable two-way communication between specific physical objects and their digital models, the realization of high-fidelity emulation-based DTs for wireless systems as a whole, namely, a Digital Twins for Mobile Networks (DTMN) [14, 15], is still a challenge largely untackled. In this work, we provide the first demonstration of the capabilities of Colosseum as a DTMN. Particularly, we develop and test a comprehensive set of tools to: (i) create an emulated DT of virtually any real-world wireless scenario in Colosseum; (ii) validate the emulated environment through channel sounding; and (iii) twin a standardized protocol stack through a Continuous Integration/Continuous Delivery (CI/CD) framework. The first two elements are carried out by using *CaST*, a *channel emulation generator and sounder toolchain*, preliminarily presented in [1], which we extend to include capabilities to realize the third element. As a use-case, we leverage the extended version of *CaST* to create and deploy a realistic DTMN of Arena [9], an over-the-air wireless testbed, on Colosseum.

The main contributions of this work are:

- We extend CaST to develop the first DTMN on Colosseum, using Arena as a use case. This use case demonstrates the scope and capabilities of Colosseum as a DT, providing the research community with a set of tools to twin real-world environments.
- We develop a CI/CD pipeline for real-time twinning of selected protocol stacks, e.g., cellular and Wi-Fi. This shows the flexibility of our tool by enabling researchers to test the latest version of open-source protocols as they are released, efficiently and automatically.
- We compare key network performance metrics, namely, throughput and Signal to Interference plus Noise Ratio (SINR), of the Arena/Colosseum DTMN to validate the fidelity of our twinning process. This is performed through a cellular network scenario with the open-source srsRAN software suite implementation [16], and via an adversarial jamming scenario using a GNU Radio-based Wi-Fi protocol stack [17].

Results show that the twinning process is able to faithfully create a digital version of a real-world RF environment, achieving an average normalized cross-correlation similarity of 0.987 in throughput and 0.982 in SINR using the cellular stack, and the Wi-Fi stack in the jamming scenario.

The remainder of this paper is organized as follows. In Section 2 we provide a brief primer on the concept of DTs for NextG systems in the context of wireless network emulators. Section 3 presents the platforms that we use to develop and implement our DTMN. Section 4 defines the steps required for digitizing a real-world environment. Section 5 describes our experimental setup and results. Section 6 surveys related previous works. Finally, Section 7 concludes the paper.

2 Digital Twins

The DT concept is finding increasing momentum as a means of enhancing the performance of physical systems by using their virtual counterparts [18]. The origin of this name is universally credited to Grieves and Vickers [19], who define a DT as a system consisting of three primary elements (Figure 1): (i) a physical product in the real world; (ii) a virtual representation of the product in the virtual world; and (iii) a connection of data and information tying the two.

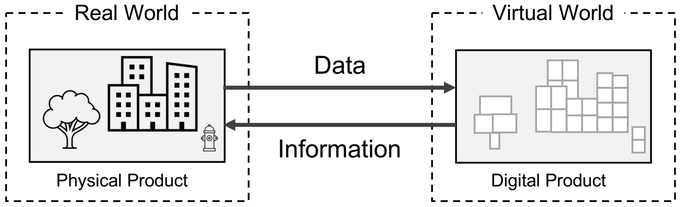


Fig. 1: High-level representation of digital twin components.

Over the years, the DT idea has extended into new domains, starting from its description and adding different flavors to this concept. For example, some works consider DT as an enabler for Industry 4.0 applications, as detailed in [20], while others suggest its use in areas such as product design, assembly, or production planning [21]. Moreover, the continuous evolution of DTs and their applications ushered the concept of Digital Twin Networks (DTNs), as systems interconnect multiple DTs [22]. Finally, DTs have been adopted in the context of the wireless communications ecosystem and cellular networks.

In this work, we apply the concept of DT to experimental wireless research, and, to the best of our knowledge, in what is the *first example of DTMN for real-world applications.* Figure 2 shows a high-level diagram of all main components of our DT representation. Specifically, we develop a set of tools to create and validate a comprehensive digital representation of a particular real-world system inside a virtual environment. This would enable researchers to run wireless experiments inside a DT of virtually any type of physical environment; develop and test new algorithms; and derive results as accurately and as close as possible to the behavior that they would obtain in the real-world environment.

To this aim, we propose Colosseum, the world's largest wireless network emulator [10], as a DT for real-world wireless experimental testbeds and environments. Thanks to its large-scale emulation capabilities, Colosseum twins both the real and digital worlds by capturing conditions of real environments and reproducing them in a high-fidelity emulation. This is done through so-called RF scenarios that model the characteristics of the physical world (e.g., channel effects, propagation environment, mobility, etc.) and convert them

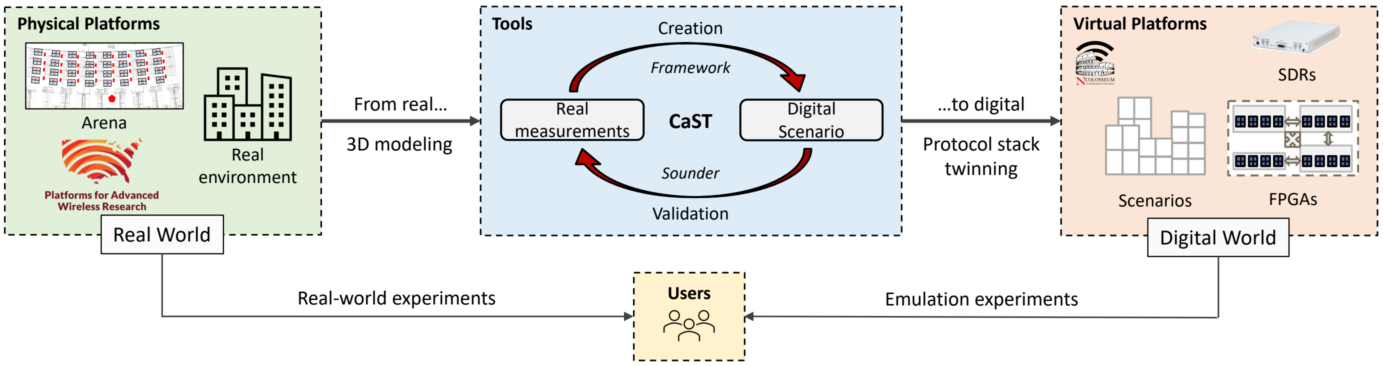


Fig. 2: Main components of our high-level representation of a DT.

into digital emulation terrains to be used for wireless experimentation. Colosseum can also twin the protocol stack itself, i.e., it allows the deployment of the same generic software-defined stacks that can replicate the functionalities of real-world wireless networks, e.g., O-RAN managed cellular protocol stacks [23, 24], orchestrators [25], and different Transmission Control Protocol (TCP) congestion control algorithms [26].

Colosseum is not limited to Software-defined Radio (SDR) devices, but it also supports the integration of Commercial Off-the-Shelf (COTS) devices—as long as they expose RF connectors compatible with those of the Massive Channel Emulator (MCHEM) USRPs, e.g., SMA connectors or equivalent ones—as demonstrated in [27], where System-on-Chip (SoC) boards running OpenWiFi are used. Additionally, custom equipment and waveforms can be integrated within Colosseum, as demonstrated in prior work where we integrated a proprietary jammer [28], and a radar waveform [29], within the system.

Through the utilization of these scenarios and the twinning of protocol stacks and generic waveforms, users can collect data and test solutions in many different environments representative of real-world deployments, and fine-tune their solutions before deploying them in production networks to ensure they perform as expected. An example of this is provided by [30], where data-driven solutions for cellular networking were prototyped and tested on Colosseum before moving them to other platforms. Overall, this allows users to retain full control over the digitized virtual world, to reproduce all—and solely—the desired channel effects, and to repeat and reproduce experiments at scale. This is particularly important for AI/Machine Learning (ML) applications [30], where: (i) access to a large amount of data is key to designing solutions as general as possible; and (ii) AI agents need to be thoroughly tested and validated in different conditions to be sure they do not cause harm to the commercial infrastructure.

To enable RF twinning between physical and digital worlds in Colosseum, we utilize our recently developed tool Channel emulation scenario generator and Sounder Toolchain (CaST), an end-to-end toolchain to create and characterize realistic wireless network scenarios with a high degree of fidelity and accuracy [1]. CaST is composed of two main parts: (i) a streamlined framework to create realistic mobile wireless scenarios from real-world environments (thus digitizing them); and (ii) a SDR-based channel sounder to characterize emulated RF channels. The protocol stack twinning is enabled by a Continuous Integration (CI) and Continuous Delivery (CD) platform that can deploy in the Colosseum system the latest, or a specifically desired, version of a wireless protocol stack. We support any software-defined stack that has been designed for real-world experiments and have implemented a specific version of a CI/CD framework for the OpenAirInterface 5G cellular implementation [31].

As proof of concept, we use CaST to create the DT of a publicly available over-the-air indoor testbed for sub-6 GHz research, namely Arena [9]. This allows us to showcase the capabilities of Colosseum as a DT platform, as well as the level of fidelity that can be achieved by the twinning process and operations.

3 Digital Twin Platforms

In this section, we describe the two platforms that are part of our DT ecosystem: (i) Colosseum, for large-scale emulation/digitization of physical environments, is described in Section 3.1; and (ii) Arena, for over-the-air real-world experimentation, in Section 3.2.

3.1 Large-scale Emulation: Colosseum

Colosseum is the world's largest publicly available wireless network emulator with hardware-in-the-loop. At a high level, Colosseum consists of five main components, depicted in Figure 3 [10]: (i) 128 Standard Radio Nodes (SRNs); (ii) the Massive Channel Emulator (MCHEM); (iii) the Traffic Generator (TGEN); (iv) the GPU nodes; and (v) the management infrastructure.

The Standard Radio Nodes (SRNs), which are divided into four quadrants, comprise 128 high-performance Dell PowerEdge R730 compute servers, each driving a dedicated USRP X310 SDR—able to operate in the $[10\,\text{MHz}, 6\,\text{GHz}]$ frequency range—through a 10 Gbps fiber cable. These servers are equipped with Intel Xeon E5-2650 CPUs with 48 cores, as well as NVIDIA Tesla K40m GPUs, to support heavy computational loads (e.g., AI/ML applications) and be able to properly drive their dedicated SDR. Users of the testbed can reserve SRNs for their experiments through a web-based Graphical User Interface (GUI), as well as specify the date/time, and amount of time they need these resources for. At the specified reservation time, Colosseum exclusively

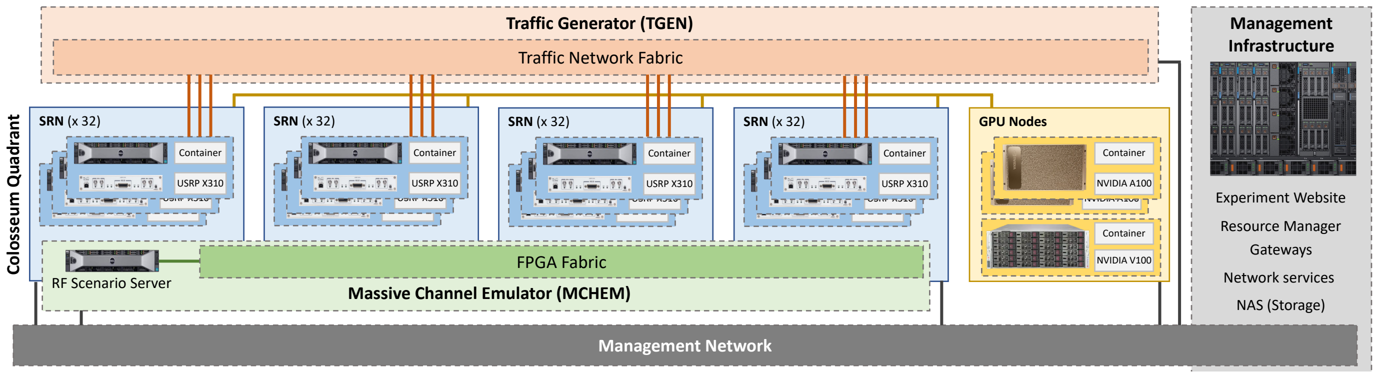


Fig. 3: Colosseum architecture, adapted from [10].

allocates the requested resources to the users and instantiates on them a softwarized protocol stack—also specified by the user when reserving resources—in the form of a Linux Container (LXC). After these operations have been carried out, users of the testbed can access via SSH to the allocated SRNs, and use the softwarized protocol stack instantiated on them (e.g., cellular, Wi-Fi, etc.) to drive the SDRs and test solutions for wireless networking in a set of diverse environments emulated by Colosseum.

These environments—called RF scenarios in the Colosseum jargon—are emulated by Colosseum MCHEM. MCHEM is formed of 16 NI ATCA 3671 Field Programmable Gate Array (FPGA) distributed across the four quadrants of Colosseum. Each ATCA module includes 4 Virtex-7 690T FPGAs that process through Finite Impulse Response (FIR) filters the signals from/to an array of USRPs X310 (32 USRPs per MCHEM quadrant, for a total of 128 USRPs across the four quadrants of Colosseum) connected one-to-one, through SMA cables, to the USRPs driven by the SRNs controlled by the users (see Figure 4).

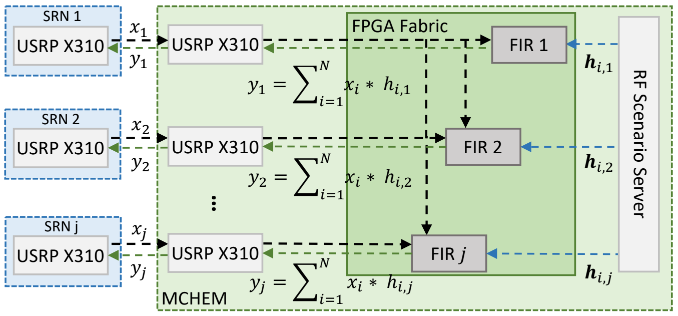


Fig. 4: FPGA-based RF scenario emulation in Colosseum, from [10].

Instead of being transmitted over the air, signals generated by the SRN USRPs are sent to the corresponding USRP on the MCHEM side. From there, they are converted in baseband and to the digital domain, and processed by the FIR filters of the MCHEM FPGAs that apply the Channel Impulse Response (CIR) corresponding to the RF scenario chosen by the user of the testbed (see Figure 4).

Specifically, these FIR filters comprise 512 complex-valued taps that are set to reproduce the conditions and characteristics of wireless channels in real-world environments, i.e., the CIR among each pair of SRN. As an example, and as depicted in Figure 4, signal x_i generated by one of the SRNs is received by the USRP of MCHEM and transmitted to its FPGAs. Here, the FIR filters load the vector $h_{i,j}$ corresponding to the 512-tap CIR between nodes i and j (with $i, j \in \{1, ..., N\}$ set of SRNs active in the user experiment) from the RF scenario server, which contains a catalog of the scenario available on Colosseum. Then, they apply these taps to x_i through a convolution operation. The signal $y_j = \sum_{i=1}^{N} x_i * h_{i,j}$ resulting from this operation, i.e., the originally transmitted x_i signal with the CIR of the emulated channel, is finally sent to SRN j. Analogous operations also allow Colosseum to perform superimposition of signals from different transmitters, and to consider interfering signals (besides the intended ones), as it would happen in a real-world wireless environment [32]. In this way, thus, Colosseum can emulate effects typical of real and diverse wireless environments, including fading, multi-path, and path loss, in terrains up to $1\ \text{km}^2$ of emulated area, and with up to 80 MHz bandwidth, and can support the simultaneous emulation of different scenarios from multiple users. Furthermore, Colosseum is capable of emulating node mobility discretely. Every millisecond, the RF Scenario Server loads different pre-defined channel taps into the Colosseum FPGAs, effectively mimicking changes in channel conditions resulting from node position changes.

Similarly to the emulation of RF environments, the Traffic Generator (TGEN) allows users of the testbed to emulate different IP traffic flows among the reserved nodes. This tool, which is based on the U.S. Naval Research Laboratory's Multi-Generator (MGEN) [33], enables the creation of flows with specific packet arrival distributions (e.g., Poisson, uniform, etc.), packet size, and rate. These traffic flows, namely *traffic scenarios*, are sent to the SRNs of the user experiment that, then, handles them through the specific protocol stack instantiated on the SRNs (e.g., Wi-Fi, cellular, etc.).

Recently, Colosseum added various GPU nodes to the pool of resources that can be reserved by users. These include two NVIDIA DGX servers, state-of-the-art computing solutions with 8 NVIDIA A100 GPUs each and interconnected through a Tbps internal NVlink switching interface, and one large memory node (Supermicro SuperServer 8049U-E1CR4T) with 6 NVIDIA V100 GPUs, 128-core Intel Xeon Gold 6242 CPUs, and 3 TB of RAM. These resources, which can be reserved from the same web-based GUI used for the SRNs, can stream data in real-time from/to the SRNs

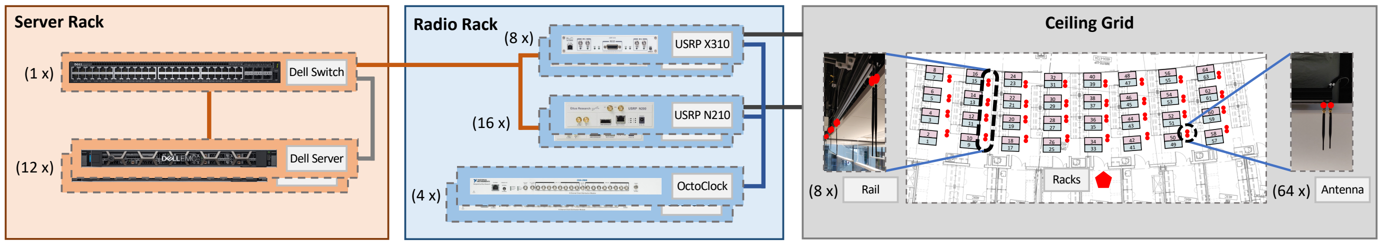


Fig. 5: Arena architecture.

through high-speed links and have the capability of powering computational-intensive workloads, such as those typical of AI/ML applications.

Finally, Colosseum includes a management infrastructure—not accessible by the users—that is used to maintain the rest of the system operational (see Figure 3). Some of the services offered by this include: (i) servers that run the website used to reserve resources on the testbed; (ii) resource managers to schedule and assign SRNs and GPU nodes to users; (iii) multiple Network Attached Storage (NAS) systems to store experiment data and container images; (iv) gateways and firewalls to enable user access and isolation throughout experiments; and (v) precise timing servers and components to synchronize the SRNs, the GPU nodes, and the SDRs.

3.2 Over-the-Air Experimentation: Arena

Arena is an over-the-air wireless testbed deployed on the ceiling of an indoor laboratory space [9]. The architecture of Arena is depicted at a high level in Figure 5. Its main building blocks are: (i) the ceiling grid; (ii) the radio rack; and (iii) the server rack.

The ceiling grid concerns 64 VERT2450 omnidirectional antennas hung off a $2450\ \text{ft}^2$ indoor office space. These are deployed on sliding rails and arranged in an 8×8 configuration to support Multiple Input, Multiple Output (MIMO) applications. The antennas of the ceiling grid are cabled through $100\ \text{ft}$ low-attenuation coaxial cables to the radio rack. This is composed of 24 USRP SDRs (16 USRP N210 and 8 USRP X310) synchronized in phase and frequency through four OctoClock clock distributors. Similarly to the USRPs on Colosseum, these SDRs can be controlled through softwarized protocol stacks (e.g., cellular, Wi-Fi, etc.) deployed on the compute nodes of the server rack, to which they are connected through a Dell S4048T-ON Software-defined Networking (SDN) programmable switch. The server rack includes 12 Dell PowerEdge R340 compute nodes that are powerful enough to drive the SDRs of the radio rack and use them for wireless networking experimentation in a real wireless propagation environment.

Because of the similarities offered by these two testbeds, software containers can be seamlessly transferred between the Colosseum and Arena testbeds with minimal modifications (e.g., specifying the network interface used to communicate with the SDRs), as discussed in Section 4.2.) As we will show in Section 5, this allows users to design and prototype solutions in the controlled environment provided by the Colosseum *digital twin*, to transfer them on Arena, and to validate these solutions in a real and dynamic wireless ecosystem.

4 Digitizing Real-world Environments

The process of digitizing real-world environments into their DT representation is composed of different steps: (i) RF scenario twinning, in which the physical environment is represented into a virtual scenario and validated thereafter; and (ii) protocol stack twinning, in which softwarized protocol stacks are swiftly transferred from the real world to the DT, thus allowing users to evaluate their performance in the designed virtual scenarios. We will describe these steps in the remainder of this section.

4.1 RF Scenario Twinning

The RF scenario twinning operations are performed by our Channel emulation scenario generator and Sounder Toolchain (CaST) [1], which we made publicly available to the research community.[1] This tool allows users to characterize a physical real-world RF environment and to convert it into its digital representation, to be used in a digital twin, such as the Colosseum wireless network emulator. CaST is based on an open SDR-based implementation that enables: (i) the creation of virtual scenarios from physical terrains; and (ii) their validation through channel sounding operations to ensure that the characteristics of the designed RF scenarios closely mirror the behavior of the real-world wireless environment.

4.1.1 *Scenario Creation*

The scenario creation framework consists of several steps that capture the characteristics of a real-world propagation environment and model it into an RF emulation scenario to install on Colosseum. These steps, which are shown in Figure 6, concern: (i) identifying the wireless environment to emulate; (ii) obtaining a 3D model of the environment; (iii) loading the 3D model in a ray-tracing software; (iv) modeling nodes and defining their trajectories; (v) sampling the channels between each pair of nodes; (vi) parsing the ray-tracing output of the channel samples; (vii) approximating the obtained channels in a format suitable for the emulation platform (e.g., Colosseum MCHEM FPGAs); and, finally, (viii) installing the scenario on Colosseum.

Identify the Wireless Environment. The first step consists of identifying the wireless environment, i.e., the physical location to twin in the channel emulator. The area to

1. https://github.com/wineslab/cast

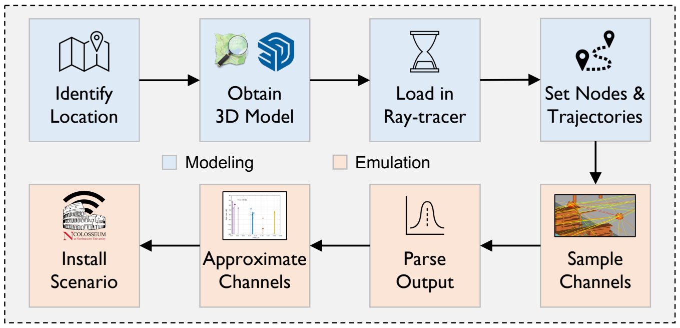


Fig. 6: CaST scenario creation workflow.

model can be of different sizes, and representative of different environments, e.g., indoor (see Section 5.3), outdoor (as shown in [1]), urban, or rural.

Obtain the 3D Model. The second step concerns obtaining the 3D model of the area to digitize. This can be obtained from various databases, e.g., Open Street Map (OSM), which is publicly available for outdoor environments or it needs to be designed using 3D modeling software, e.g., SketchUp.

Load the Model in the Ray-tracer and Assign Material Properties. The 3D model obtained in the previous step needs to be converted into a file format (e.g., STL) suitable to be loaded into a ray-tracing software, e.g., the MATLAB ray-tracer or Wireless InSite (WI), a commercial suite of ray-tracing models and high-fidelity Electro-Magnetic (EM) solvers developed by Remcom [34]. Each object in the 3D model imported by the ray-tracing software consists of surfaces, and the material properties of these surfaces should be set to have reasonable ray-tracing results. The level of granularity in this step may depend on the ray-tracer platform, e.g., in the WI, the material properties can be assigned to each surface. In the current version of MATLAB ray-tracer, this assignment is limited to the terrain and the buildings. The flexibility in assigning materials with a high level of detail leads to complex structures in the environment objects and accurate ray-tracing results.

Model Nodes and Define Trajectories. Once the 3D model of the environment has been loaded in the ray-tracing software and the material properties are assigned, the radio nodes need to be modeled, which includes setting the nodes' radio parameters, modeling the antenna pattern, and defining locations of the nodes in the physical environment. These nodes can be either static or mobile, in which case their trajectories and movement speeds need to also be defined. The radio parameters of the nodes, e.g., carrier frequency, bandwidth, transmit power, receiver noise figure, ambient noise density, and antenna characteristics, need to be set as well.

Sample the Channels. At this point, the channel is sampled through the ray-tracing software with a predefined sampling time interval T_s, which allows for capturing the mobility of the nodes in a discrete way. To this aim, the node trajectories are spatially sampled with a spacing $D_i = V_i \cdot T_s$, where V_i is the speed of node i. Since spatial consistency plays a key role in providing a consisting correlated scattering environment in the presence of mobile nodes, we follow the 3GPP recommendations and consider a coherence distance of 15 m to guarantee an apt spatial consistency [35].

Parse the Output. The next step consists of parsing the ray-tracer output to extract a synchronized channel between each pair of nodes in the scenario for each discrete time instant t spaced at least 1 ms. The temporal characteristic of the wireless channels is considered as a FIR filter, where the CIR is time-variant and expressed by:

$$h(t,\tau) = \sum_{i=1}^{N_t} \tilde{c_i}(t) \cdot \delta(t - \tau_i(t)), \tag{1}$$

where N_t is the number of paths at time t, and τ_i and c_i are the Time of Arrival (ToA) and the path gain coefficient of the i-th path, respectively. The latter is a complex number with magnitude a_i and phase φ_i

$$\tilde{c_i}(t) = a_i(t) \cdot e^{j\varphi_i(t)} \tag{2}$$

Approximate the Channels. The CIR characterized in the previous steps needs to be converted in a format suitable for MCHEM FPGAs, e.g., 512 channel taps, 4 of which assume non-zero values, spaced with steps of 10 ns and with a maximum excess delay of 5.12 μs. To do this, we leverage a ML-based clustering technique to reduce the taps found by the ray-tracing software, align the tap delays, and finalize their dynamic range, whilst ensuring the accuracy of the emulated scenario [36].

Install the Scenario. Finally, the channel taps resulting from the previous steps are fed to Colosseum scenario generation toolchain, which converts them in FPGA-friendly format and installs the resulting RF scenario on the DT, ready to be loaded on-demand by the RF Scenario Server.

4.1.2 Scenario Validation

Now that the scenario has been created and installed in the DT, we validate its correct functioning through the channel sounder embedded in CaST [1]. In doing this, we also ensure that the scenario installed in the DT closely follows the behavior experienced in the real-world environment.

The main steps of CaST channel sounder, shown in blue shades in Figure 7, are: (i) the transmission of a known code sequence used as a reference for the channel sounding operations; (ii) the reception of the transmitted code sequence, processed by MCHEM through the channel taps of the emulated RF scenario; (iii) the post-processing of the received data and its correlation with the originally transmitted code sequence; and (iv) the validation of the results with the modeled channel taps.

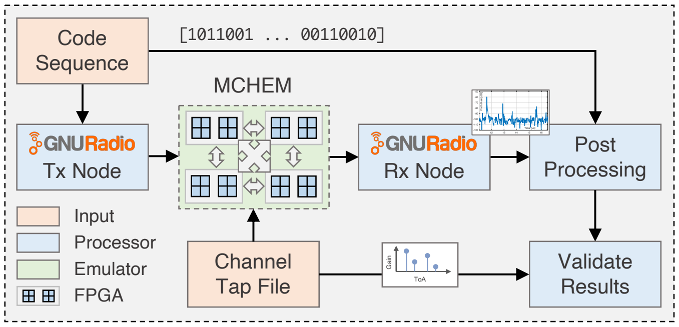


Fig. 7: CaST channel sounding workflow.

The CaST sounder uses a transmitter and a receiver node implemented through the GNU Radio open-source SDR development toolkit [37]. This software toolkit allows implementing and programming SDRs through provided signal processing blocks that can be interconnected to one another.

In our sounding application, the transmitter takes as input a known code sequence—how to derive the specific code sequence is described in Section 5.1—and transmits it to the receiver node through the wireless channel emulated by the Colosseum DT through the RF scenario to evaluate. The transmitted signal is composed of sequential repetitions of the code sequence encoded through a Binary Phase-shift keying (BPSK) modulation. While other modulation types are not restricted, we leverage BPSK because it offers sufficient channel information for the sounding in Colosseum. Additionally, it allows for simple data computations that are less susceptible to errors and approximations, resulting in a cleaner and less disrupted signal. Data is streamed to the USRP controlled by the SRN that transmits it to the receiving node through MCHEM. For increased flexibility of the channel sounder, CaST allows users to set various USRP parameters, such as clock source, sample rate, and frequency.

At the receiver side, the SRN USRP samples the signal sent by MCHEM, i.e., the transmitted signal processed with the channel taps of the emulated scenario. This signal is cross-correlated with the originally transmitted known code sequence to extract the CIR $h(t)$ of the emulated scenario, and the Path Loss (PL) $p(t)$. The CIR is then used to obtain the ToA of each multi-tap component of the transmitted signal, which allows measuring the distance between taps, while the PL allows measuring the intensity and attenuation of such components as a function of the time delay. To perform the above post-processing operations, let $c(t)$ be the N-bit known code sequence, and $s^{IQ}(t)$ and $r^{IQ}(t)$ the In-phase and Quadrature (IQ) components of the transmitted $(s(t))$ and received $(r(t))$ signals, respectively. The IQ components of the CIR are computed by separately correlating $r^I(t)$ and $r^Q(t)$ (i.e., the I and Q components of $r^{IQ}(t)$) with the I and Q components of $s(t)$ divided by the inner product of the transmitted known sequence with its transpose:

$$h^I(t) = \frac{r^I(t) \otimes s^I(t)}{s^{I^T}(t) \times s^I(t)}, \tag{3}$$

$$h^Q(t) = \frac{r^Q(t) \otimes s^Q(t)}{s^{Q^T}(t) \times s^Q(t)}, \tag{4}$$

where $\otimes$ is the cross-correlation operation [38] between two discrete-time sequences x and y, which measures the similarity between x and shifted (i.e., lagged) repeated copies of y as a function of the lag k following:

$$\chi(k) = \sum_{n=1}^{N} x(n) \cdot y(n+k), \tag{5}$$

with χ denoting the cross-correlation and N the length of the x sequence. It is worth noticing that if the considered modulation is a BPSK, the denominator is equal to the length N of $c(t)$. The amplitude of the CIR can be computed as:

$$|h(t)| = \sqrt{(h^I(t))^2 + (h^Q(t))^2} \tag{6}$$

and the path gains as:

$$G_p(t)[dB] = 20log_{10}(|h(t)|) - P_t - G_t - G_r, \tag{7}$$

where P_t is the power of the transmitted signal, and G_t and G_r are the transmitter and receiver antenna gains expressed in dB.

4.2 Protocol Stack Twinning

The twinning of protocol stacks from real to virtual environments (and back) is key in the DT ecosystem, as it allows users to swiftly transfer and evaluate real-world solutions in a controlled setup through automated tools. Twinning at the protocol stack level, combined with the RF scenario twinning discussed in Section 4.1, makes it possible to seamlessly prototype, test, and transition end-to-end, full-stack solutions for wireless networks to and from digital and physical worlds. After validation in the controlled environment of the DT—to make sure whatever is tested works as expected—the protocol stack solutions can be transitioned back to real-world deployments where they are ultimately used on a production network. As an example, in our prior works [24, 30], we have shown how AI solutions for 5G cellular networks trained and tested on the digital twin environment—Colosseum—can be effective and also work on real-world environments—Arena and the PAWR platforms [6].

At a high level, the twinning of the protocol stack involves: (i) tracking one or multiple remote, centralized version control systems that host the code of the protocol stack; and (ii) providing pipelines that can automatically replicate the same software build in the digital and physical domains. In addition, it is possible to embed automated steps for the performance validation (i.e., profiling of relevant performance metrics), similar to the scenario validation step of CaST.

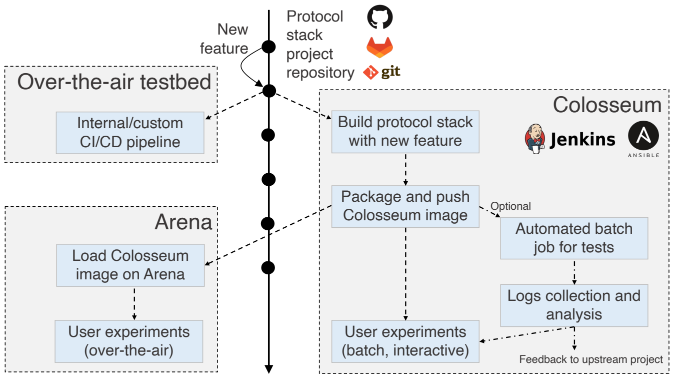

New feature
Protocol stack project repository
git
Over-the-air testbed
Internal/custom CI/CD pipeline
Colosseum
Build protocol stack with new feature
Jenkins
ANSIBLE
Package and push Colosseum image
Optional
Automated batch job for tests
Logs collection and analysis
Arena
Load Colosseum image on Arena
User experiments (over-the-air)
User experiments (batch, interactive)
Feedback to upstream project

Fig. 8: Protocol stack twinning workflow across a digital environment, i.e., Colosseum, and two physical environments, i.e., Arena and a generic over-the-air testbed.

Figure 8 illustrates how the protocol stack twinning is implemented in Colosseum (right), with extensions to a generic over-the-air testbed (top left) and the specific integration with Arena (bottom left). The figure refers to a project repository for a sample protocol stack, hosted on a versioning platform that supports the `git` version control system (e.g., GitHub, GitLab, among others). In Colosseum, we implemented this pipeline for the OpenAirInterface reference stack for 5G base stations and User Equipments (UEs), and are working toward the integration of additional components for O-RAN testing [30].

Whenever a new feature (i.e., a commit on selected branches, or a pull request) is pushed to the target project repository, a CI/CD framework implemented with Jenkins and Ansible triggers the automated process in the digital Colosseum domain. Specifically, Jenkins monitors the remote repository and orchestrates the kickoff of the build job, and Ansible applies the relevant configuration parameters to the machine that actually executes the build job (e.g., a Colosseum SRN or a dedicated virtual machine on AWS). Once the build is successful, the Jenkins job packages the output of the process into an LXC image which is stored on the Colosseum NAS.

Once this is done, Colosseum can be further used to perform automated testing, e.g., to automatically test solutions and algorithms on the DT and collect relevant metrics from such experiments. These can be shared with the relevant stakeholders, e.g., the developers of the protocol stack framework being deployed and tested. Moreover, since no over-the-air transmissions happen in Colosseum, as the channels are emulated through MCHEM (see Section 3.1), this DT environment enables users to test networking solutions over frequencies and bandwidths that would normally require compliance with the Federal Communications Commission (FCC) regulations.

Finally, the image with the relevant components can be used by experimenters in Colosseum or moved to the physical domain, e.g., Arena, for validation on a real-world infrastructure. In addition, the protocol stack can be twinned in other over-the-air testbeds following their internal and custom CI/CD pipelines, as long as the centralized repository that the different testbeds track provides shared specifications for the build environment (e.g., operating system, compiler versions, packages, etc.). As an example, the Colosseum protocol stack twinning process already replicates internal CI/CD pipelines used in the Eurecom/OpenAirInterface facilities [39].

5 Experimental Evaluation

This section discusses the capabilities of our DT system through experimental evaluations, and it is organized as follows: (i) we showcase CaST tuning process (Section 5.1); (ii) we leverage CaST to validate Colosseum scenarios, both with single and multiple taps (Section 5.2); (iii) we describe the Arena scenario designed as part of this paper (Section 5.3); and (iv) we compare some experimental use cases (e.g., for cellular networking and Wi-Fi applications) both in the Arena testbed and in its DT representation (Section 5.4).

5.1 CaST Tuning

As a first step, we tune CaST parameters and configurations (see Section 4.1) outside the Colosseum channel emulator to identify a code sequence with high auto-correlation and low cross-correlation between transmitted code sequence and received signal which functions well within our combination of software and hardware. It is worth noting that the tuning of CaST is a one-time operation, and it would need to be repeated only upon changes in the Colosseum hardware (e.g., radio devices and channel emulation system). This step, which is key for CaST to be able to derive taps from arbitrary CIRs, is performed in the controlled environment shown in Figure 9.

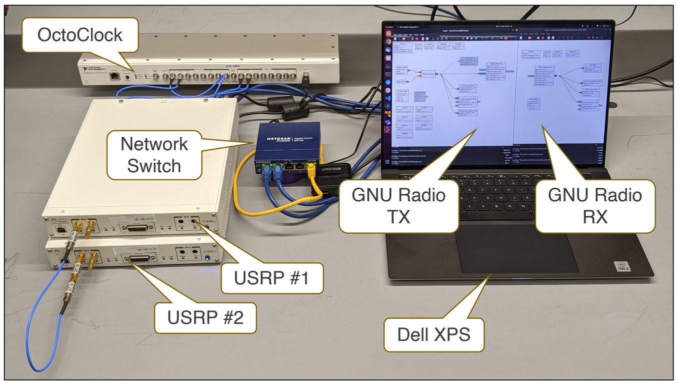


Fig. 9: Controlled laboratory environment used for the CaST tuning process.

This consists of two USRP X310 SDRs equipped with a UBX-160 daughterboard and synchronized in phase and frequency through an OctoClock clock distributor to mirror the same deployment used in Colosseum. Differently from the Colosseum deployment, however, the two USRPs are connected through a 12 inches SMA cable, and 30 dB attenuators (to shield the circuitry of the daughterboard from direct power inputs, as indicated in their datasheet). This is done to derive the above-mentioned code sequences in a baseline and controlled setup without additional effects introduced by over-the-air wireless channels, or channel emulators. The USRPs are connected through a network switch to a Dell XPS laptop, used to drive them. The sounding parameters used in this setup are summarized in Table 1. We consider different values for the gains of the USRPs (i.e., in $[0, 15]$ dB) to evaluate their effect on the sounding results. The receiving period time and data acquisition are set to 3 s.

TABLE 1: Configuration parameters used in the controlled laboratory setup.

Parameter	Value
Center frequency	1 GHz
Sample rate	$[1, 50]$ MS/s
USRP transmit gain	$[0, 15]$ dB
USRP receive gain	$[0, 15]$ dB

Finding the Code Sequence. Code sequences have been widely investigated in the literature because of their role in many different fields [40, 41]. Good code sequences achieve a high auto-correlation (i.e., the correlation between two copies of the same sequence), and a low cross-correlation (i.e., the correlation between two different sequences). For our channel-sounding characterization, we consider and test four different code sequences by leveraging the laboratory environment shown in Figure 9:

- *Gold sequence*. These sequences are created by leveraging the XOR operator in various creation phases applied to a pair of codes, u, and v, which are called a preferred pair. This pair of sequences must satisfy specific requirements to qualify as suitable for a gold sequence, as detailed in [42]. Gold sequences have small cross-correlation within a set, making them useful when more nodes are transmitting in the same frequency range. They are mainly used in telecommunication (e.g., in Code-Division Multiple Access (CDMA)) and in satellite navigation systems (e.g., in GPS). In this work, we use a Gold sequence of $255\mathrm{bits}$ generated with the MATLAB Gold sequence generator system object with its default first and second polynomials, namely $z^6 + z + 1$ and $z^6+z^5+z^2+z+1$, for the generation of the preferred pair sequences.

- *Golay complementary sequence.* Being complementary, these sequences have the property that the sum of their out-of-phase aperiodic auto-correlation coefficients is equal to 0 [43]. Their applications range from multi-slit spectrometry and acoustic measurements to Wi-Fi networking and Orthogonal Frequency Division Multiplexing (OFDM) systems. In our tests, we use a 128-bit type A Golay Sequence (Ga_{128}) as defined in the IEEE 802.11ad-2012 Standard [44].
- *Loosely Synchronised (LS) sequence.* These sequences exhibit the property of reaching very low auto-correlation and cross-correlation values in a certain portion of time, based on the maximum delay dispersion of the channel, called Interference Free Window (IFW). This allows the mitigation of the interference if the maximum transmission delay is smaller than the IFW length. In our experiments, we use an LS sequence generated following the directions in [45], and only leveraging the first codeset of $\{-1, 1\}$ without including the IFW.
- *Galois Linear Feedback Shift Register (GLFSR) sequence.* These sequences add time offsets to Linear Feedback Shift Register (LFSR) codes by leveraging extra XOR gates at the output of the LFSR. This allows to achieve a higher degree of randomness if compared to the classic LFSR, making them more efficient and fast in detecting potential faults with increased auto-correlation results [46]. In this paper, we leverage GNU Radio to generate a 255-bits sequence with the following parameters: shift register degree 8, bit mask 0, and seed 1.

Each of these sequences has been separately used by the transmitter node to construct the sending signal and to send it to the receiver node with a sample rate of 1 MHz. After that, the receiver node performs the post-processing operations. Results of 800μs CIR for each code sequence are shown in Figure 10. We can notice that all code sequences are able to correctly identify the starting position of the transmitted signal, as shown by the peak values. The distance D_{peak} of each peak can be written as a function of the code length N and the sampling rate SR.

$$D_{peak} = \frac{N}{SR} \tag{8}$$

Therefore, D_{peak} is equal to 255 μs for the Gold, LS, and GLFSR codes, each showing 3 transmitted sequences in Figure 10, and to 128 μs for the Ga_{128} code, which displays 6 sequences instead. We notice that GLFSR shows the highest auto-correlation and lowest cross-correlation among the four considered code sequences. This results in an overall cleaner CIR. For these reasons, we adopt the GLFSR code sequence in our experimental evaluation through CaST.

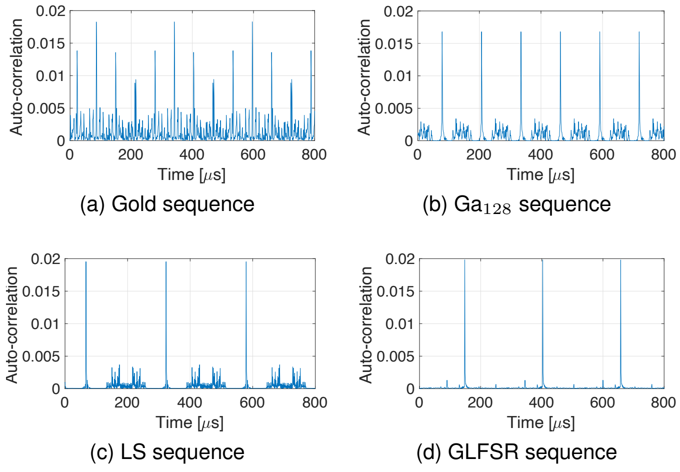


Fig. 10: Correlation of different code sequences in the controlled laboratory environment.

CaST Validation in a Laboratory Environment. After identifying the code sequence for our application, we evaluate CaST in the laboratory setup shown in Figure 9. To this aim, we test our sounder with a GLFSR code sequence and various configuration parameters, e.g., sample rate, center frequency, and antenna gains, to study its behavior and gather reference information to be leveraged in the Colosseum experiments. Figure 11 shows a time frame of the received path gains for the case with 0 dB (blue line in the figure), and 30 dB total transmit and receive gains (15 dB at both transmitter and receiver sides, orange line).

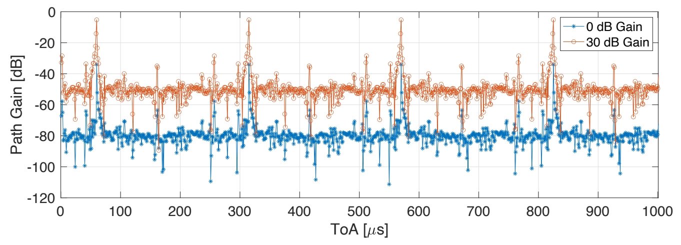


Fig. 11: Received path gains in the controlled laboratory environment with 0 and 30 dB total transmit and receive gains use cases (15 dB at both transmitter and receiver sides).

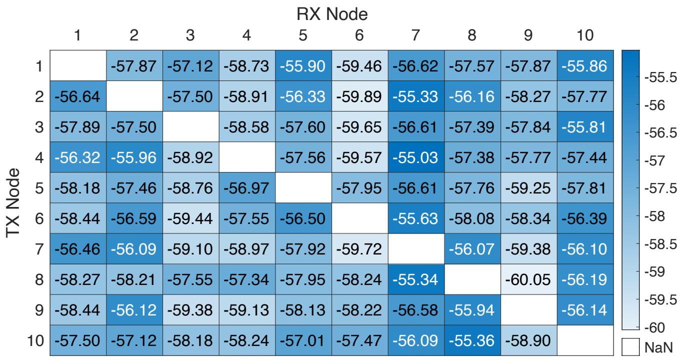
The figure shows signals that repeat based on the length of the transmitted code sequence, i.e., every 255 sample points (or equivalently every 255μs, since one point equals to $1/\text{sample_rate} = 1\ \mu$s). The peaks represent the path loss of the single tap of this experiment, which are equal to 34.06 dB for the 0 dB case, and 5.24 dB for the 30 dB case. Since we have 30 dB attenuation in this validation setup, these results are in line with our expectations (with some extra loss due to the physical components of the setup, e.g., cable attenuation and noise). We also notice that in the 30 dB case, the measured loss is slightly more severe due to imperfections in the power amplifiers of the USRPs. We use these results as a reference for our channel-sounding operations.

5.2 Validation of Colosseum Scenarios through CaST

After the tuning and validation in the controller laboratory environment, we can leverage CaST to validate the behavior of Colosseum MCHEM. We deploy the CaST sounder on the Colosseum wireless network emulator by creating an LXC container from the open-source CaST source code. This container, which has been made publicly available on Colosseum, contains all the required libraries and software to perform channel-sounding operations, as well as for the post-processing of the obtained results. This enables the reusability of the sounder with different SRNs and scenarios, as well as portability to different testbeds (e.g., to the Arena testbed described in Section 3.2). It also allows the automation of the channel sounding operations through automatic runs supported by Colosseum, namely *batch jobs*.

To achieve our goal of characterizing MCHEM, we test a set of synthetic RF scenarios (i.e., single- and multi-tap RF scenarios) on Colosseum, i.e., scenarios created specifically for the purpose of channel sounding. These scenarios have been manually generated with specific channel characteristics to validate the behavior of MCHEM, and have been made publicly available for all Colosseum users. The parameters used in this evaluation are the same as the ones in Table 1 with the only exception of the sample rate that is set at 50 MS/s to have a 20 ns resolution (thus being able to properly retrieve tap delays and gains), and the GLFSR code sequence found above.

Single-tap Scenario. The first synthetic RF scenario that we consider is a single-tap scenario with nominal 0 dB path loss (i.e., 0 dB of path loss added to the inherent loss of the hardware components of the testbed). To find the base loss of MCHEM, i.e., the loss due to Colosseum hardware-in-the-loop, we instantiate CaST on 10 SRNs, and sound the channels among them, measuring the path loss of each link, shown in Figure 12.



	1	2	3	4	5	6	7	8	9	10
1		-57.87	-57.12	-58.73	-55.90	-59.46	-56.62	-57.57	-57.87	-55.86
2	-56.64		-57.50	-58.91	-56.33	-59.89	-55.33	-56.16	-58.27	-57.77
3	-57.89	-57.50		-58.58	-57.60	-59.65	-56.61	-57.39	-57.84	-55.81
4	-56.32	-55.96	-58.92		-57.56	-59.57	-55.03	-57.38	-57.77	-57.44
5	-58.18	-57.46	-58.76	-56.97		-57.95	-56.61	-57.76	-59.25	-57.81
6	-58.44	-56.59	-59.44	-57.55	-56.50		-55.63	-58.08	-58.34	-56.39
7	-56.46	-56.09	-59.10	-58.97	-57.92	-59.72		-56.07	-59.38	-56.10
8	-58.27	-58.21	-57.55	-57.34	-57.95	-58.24	-55.34		-60.05	-56.19
9	-58.44	-56.12	-59.38	-59.13	-58.13	-58.22	-56.58	-55.94		-56.14
10	-57.50	-57.12	-58.18	-58.24	-57.01	-57.47	-56.09	-55.36	-58.90	

Fig. 12: Path loss heatmap as measured by CaST in a 0 dB Colosseum RF scenario with 10 SRNs.

Each cell in the figure represents the average path loss for 2 s of reception time between transmitter (row) and receiver (column) nodes. Results show an average Colosseum base loss of 57.55 dB with a Standard Deviation (SD) of 1.23 dB. We also observe that the current dynamic range of Colosseum is approximately 43 dB, i.e., between the 57.55 dB base loss at 1 GHz and the noise floor of -100 dB.

Multi-tap Scenario. The second synthetic RF scenario that we consider is a four-tap scenario in which taps have different delays and path gains. We characterize such a scenario on Colosseum through CaST channel sounding operations. Results for the emulated and modeled path gains for a single time frame are shown in Figure 13 in blue and orange, respectively.

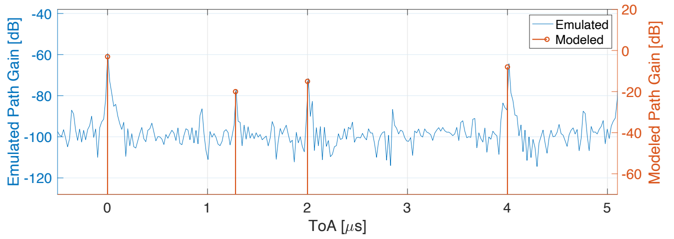


Fig. 13: Comparison between emulated and modeled path gains in Colosseum for a single time frame.

We notice that the ToAs match between the modeled CIR and the taps emulated by the Colosseum RF scenario, namely they occur at 0, 1.28, 2, and 4 μs. We also notice that the received powers are in line with our expectations. Indeed, by adding the Colosseum base loss computed in the previous step to the power measured by CaST (in blue in the figure), we obtain the modeled taps (corresponding to -3, -20, -15, and -8 dB, shown in orange in the figure).

We now analyze the accuracy of the measurements performed with CaST by computing the relative difference between the emulated taps over time. We do so by considering $1,500$ time frames. Results show that the average difference between the strongest tap of each time frame is in the order of 10^{-6} dB, with a SD of 0.03 dB. Analogous results occur for the second tap—which is the weakest tap in our modeled CIR—with a SD of 0.17 dB, and for the third and fourth taps. Finally, differences between the first and second taps of each time frame (i.e., between strongest and weakest taps in our modeled CIR) amount to 0.52 dB with a SD of 0.18 dB. These results are a direct consequence of the channel noise, which impacts weaker taps more severely.

Overall, results demonstrate MCHEM accuracy in emulating wireless RF scenarios in terms of received signal, tap delays, and gains. This also shows CaST effectiveness in achieving a 20 ns resolution, thus sustaining a 50 MS/s sample rate, and a tap gain accuracy of 0.5 dB, which allows CaST to capture even small differences between the modeled and emulated CIR.

5.3 Arena Digital Twin Scenario

We use Sketchup [47] software to create a 3D representation of the Arena testbed. This software allows users to model a broad range of environments starting from an architectural layout (e.g., of the Arena testbed, a picture of which is shown in Figure 14a), and with different surface renderings, e.g., glass walls and windows, wooden walls, carpeted floors [47]. The resulting 3D model (shown in Figure 14b) is then fed to the ray-tracing software, Wireless inSite [34] in this case, to create a DT scenario on Colosseum following the steps described in Section 4.1.

For the developed Arena scenario, we model the antenna points of the Arena testbed in 32 locations (one for each antenna pair), as well as 8 static nodes distributed in their

(a) Real-world location

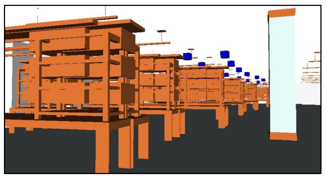

(b) Digital-twin scenario

Fig. 14: The conversion from a real-world location, into a digital medium scenario used to create the digital twin representation.

surroundings, and 2 mobile nodes traversing the laboratory space at a constant speed of 1.2 m/s. The height of the nodes (both static and mobile) is set to 1 m, e.g., to emulate handheld devices, or devices lying on table surfaces. The modeled locations and nodes are shown in Figure 15, where the red circles represent the antenna pairs of Arena, while the blue squares and green pentagons identify the static and mobile nodes, respectively. The dashed green arrows denote the movement direction of the mobile nodes. By using a Dell T630 machine with 2 Xeon E52660 14 cores CPU, 128 GB RAM, and Tesla K40 GPU, the ray-tracing operations of this scenario took 14 h and 21 m, while the channel approximation process 2 h and 33 m. Additionally, the installation process in Colosseum required around 19 h and 30 m by leveraging a virtual machine hosted on a Dell PowerEdge M630 Server with 24 CPU cores and 96 GB of RAM. It is worth noting that these are one-time operations and the scenario is played on-demand right away afterward.

Figure 16 shows the heat map of the path loss among the transmit-receive node pairs (the mobile nodes are considered in the starting position on the left). As expected, closer nodes experience a lower path loss, which increases with the distance between the nodes. A similar trend is also visible for the static nodes, even though this is less noticeable due to their scattered locations. On the other hand, due to their remote starting locations on the side of the room, the mobile nodes exhibit a very high path loss against all nodes, as depicted in Figure 16. These path losses decrease as they get closer to each node on their path, and increase again while reaching their end locations on the other side of the room.

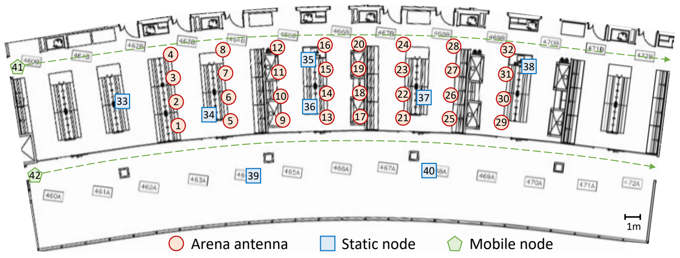


Fig. 15: Location of the nodes in an Arena DT scenario.

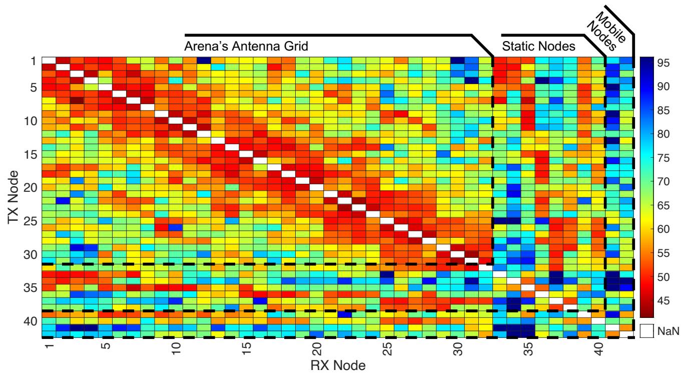


Fig. 16: Heat map of the path loss among the nodes of Figure 15, with a line separator between antenna, static, and mobile. The mobile nodes are considered in the starting position on the left.

5.4 Experimental Use Cases

In this section, we show outcomes of relevant experimental use cases run on both the Arena testbed, as well as on its DT representation. The first use case involves the deployment of a cellular networking system using the srsRAN software suite, while the second one encompasses a Wi-Fi adversarial jamming use case built on GNU Radio.

5.4.1 Cellular Networking

Single BS. In the first cellular networking use case, we leverage SCOPE [48]—an open-source framework based on srsRAN [16] for experimentation of cellular networking technologies—to deploy a twinned RAN protocol stack with one Base Station (BS) and three UEs in the Arena over-the-air testbed and in the Colosseum emulation system. The same node positions, shown in Figure 17, are used in the two platforms: the BS, which transmits over a 10 MHz spectrum, is located on node 12, two static UEs on nodes 34 and 37, and one mobile UE on node 41. In Arena, UEs are implemented through commercial smartphones (Xiaomi Redmi Go), while on Colosseum, they are deployed on the SDRs of the testbed.

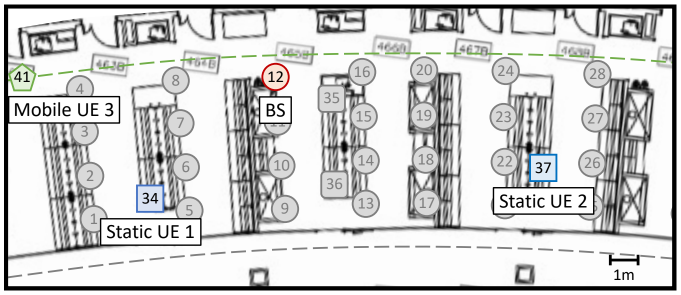


Fig. 17: Location of the nodes in the cellular experiment.

We conduct two experiments on each system: the first one involves a downlink User Datagram Protocol (UDP) traffic sent at a 5 Mbps rate; the second one TCP downlink traffic. The traffic generation for each experiment is achieved using iPerf, a benchmarking tool designed for assessing the performance of IP networks [49]. The following results show the average of at least 5 separate experiment realizations.

Figure 18 shows the UDP downlink throughput for static (blue and orange lines), and mobile (yellow line) nodes on the Arena (Figure 18a) and Colosseum (Figure 18b) testbeds. We can notice similar trends and patterns exhibited on both testbeds. Specifically, the throughput of the static nodes remains stable around 5 Mbps in both Colosseum and Arena, where we notice a less stable behavior due to the use of over-the-air communications, and potential external interference.

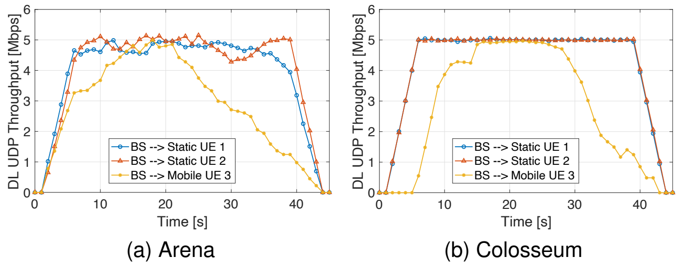


Fig. 18: UDP downlink throughput of the cellular use case on the Arena and Colosseum testbeds.

As expected, the throughput of the mobile node—that starts from the top-left location shown in Figure 17 and travels to the right along the trajectory depicted with the green line in the figure—increases as the node gets closer to the BS (where it reaches a 5 Mbps peak), and then decreases as the node gets farther away.

Figure 19 and Figure 20 plot the TCP downlink throughput and SINR results of the second experiment for static (blue and orange) and mobile (yellow) nodes on Arena (Figure 19a and Figure 20a) and Colosseum (Figure 19b and Figure 20b). Also in this use case, a similar pattern is clearly

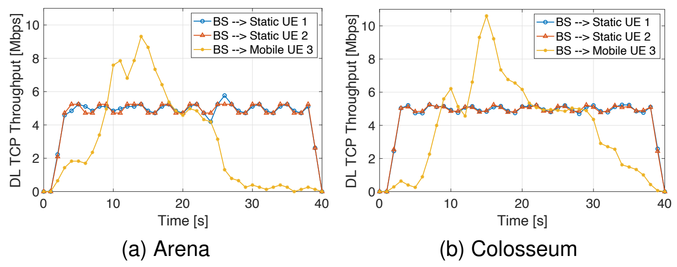


Fig. 19: TCP downlink throughput of the cellular use case on the Arena and Colosseum testbeds.

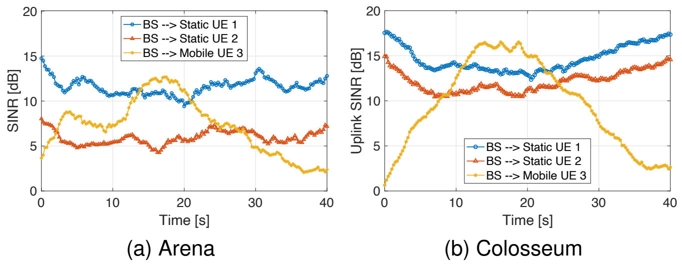


Fig. 20: TCP SINR of the cellular use case on the Arena and Colosseum testbeds.

noticeable. In particular, the two static nodes maintain a relatively stable throughput of the nominal 5 Mbps TCP traffic on both testbeds with no apparent impact during the passage of the mobile UE close to the static UEs, which in the Arena case is moved manually. This behavior is visible in the SINR results, which show a decrease due to the created interference when the mobile node approaches each of the static UEs. The mobile node exhibits a similar trend with two high peaks in throughput on both systems. These peaks can be attributed to the TCP protocol, which retransmits data to ensure delivery in case of packet failures as soon as the signal improves, resulting in higher application throughput values compared to the nominal 5 Mbps. Specifically, each peak corresponds to the time right after the mobile device transitions close to static node 1 (around time 10 s), and then to static node 2 (around time 15 s). Due to the increased interference, the mobile node loses more packets, which will eventually be retransmitted.

Multiple BSs. In the second cellular networking use case, we use a similar setup as for the first experiment by leveraging SCOPE and Xiaomi Redmi Go phones to deploy a twinned srsRAN protocol stack with two BSs located in positions 10 (BS 1) and 25 (BS 2), as shown in Figure 21. We assign three UEs to each BS, for a total of 6 UEs (i.e., the number of smartphones currently at our disposal). Specifically, UE 1, UE 2, and UE 3 are assigned to BS 1, while UE 4, UE 5, and UE 6 are assigned to BS 2.

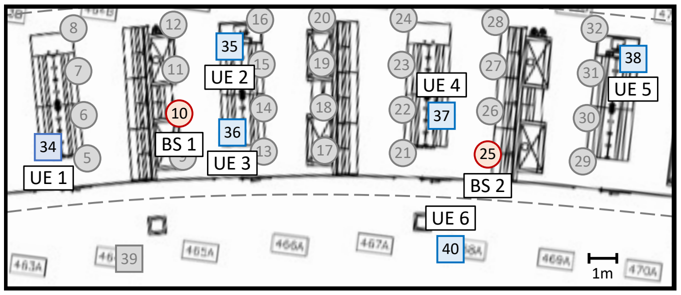


Fig. 21: Location of the nodes in the multiple BSs cellular use-case, consisting of 2 BSs (BS 1, BS 2), and 6 static UEs, equally assigned to each BS: UE 1, UE 2, and UE 3 for BS 1; UE 4, UE 5, and UE 6 for BS 2.

Once each UE has completed the attachment procedures, we initiate a continuous 5 Mbps UDP downlink traffic stream from each smartphone using iPerf towards its corresponding BS. Figure 22 shows the average downlink throughput results over a period of more than 30 minutes of data collection, along with 95% confidence intervals. These metrics are collected at the data-link layer at the BS level through the SCOPE framework. We observe that all UEs are able to consistently ensure the 5 Mbps throughput with a very low 95% confidence interval showing an average margin of error of 0.02 Mbps. This demonstrates the accuracy of the digital twin representation, even in the presence of multiple BSs and UEs, showcasing the scalability of our DTMN.

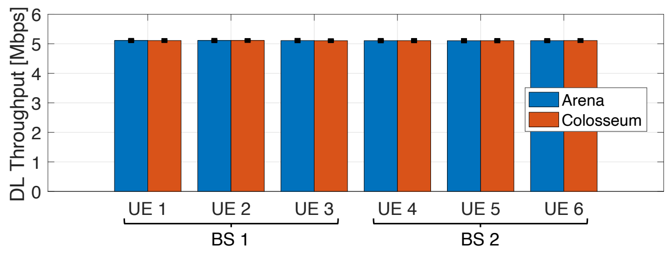


Fig. 22: Downlink throughput average results and 95% confidence interval of the second cellular networking use-case with 2 BSs and 6 UEs.

Comparison. By analyzing the results after the UEs have completed the attachment procedures, we can assess the similarity between the two testbeds by following

$$\rho(k) = \frac{\sum_{n=1}^{N}(x(n)-\bar{x})(y(n+k)-\bar{y})}{\sqrt{\sum_{n=1}^{N}(x(n)-\bar{x})^2 \sum_{n=1}^{N}(y(n)-\bar{y})^2}}, \quad (9)$$

which measures the normalized cross-correlation $\rho(k)$ between the Arena data sequence, denoted as $x(n)$, and shifted lagged copies of the Colosseum data sequence, denoted as $y(n)$, as a function of the lag k. In this context, $\bar{x}$ and $\bar{y}$ represent the means of the input sequences and N is their length. If $x(n)$ and $y(n)$ have different lengths, zeros are appended to the end of the shorter vector to ensure both sequences have the same length N.

Table 2 presents the normalized cross-correlation results and their averages for throughput and SINR of each UE, considering the maximum values between 10 lags, of the UDP and TCP single BS use-case experiments. The multiple BSs experiments would yield similar comparison results.

TABLE 2: Normalized cross-correlation results, and their averages, considering the maximum values between 10 lags for the single BS cellular experiment.

Metric	Static UE 1	Static UE 2	Mobile UE 3	Average
UDP Throughput	0.986	0.998	0.999	0.994
TCP Throughput	0.937	0.998	0.998	0.978
TCP SINR	0.997	0.994	0.982	0.991

We can observe a very high similarity between the two testbeds in all use-case experiments, with individual UE values consistently above 0.93 and an average exceeding 0.97 for each use-case metric. It is worth noting, as observed in Figure 20, that the SINR results in Arena are quite similar from UE 1 and UE 3, and they present a fixed difference of about 5 dB compared to Colosseum for UE 2. This difference can mainly be attributed to the additional and uncontrolled interference and impairments of a real-world RF environment, as well as the different power levels between a simulated UE and a real smartphone. However, these variances can be compensated for in the DT by adjusting factors such as the node gains at the transmitter and receiver, as well as by adding stochastically representative interference models to the channel that represents the real-world behavior more closely. These findings confirm the capabilities of the DT to perform emulated cellular experiments that closely replicate the behavior of real-world setups and environments, even in the presence of mobile nodes.

5.4.2 Wi-Fi Jamming

Adversarial jamming has continuously plagued the wireless spectrum over the years with the ability to disrupt, or fully halt, communications between parties. While there are potential solutions to specific types of jamming, due to the open nature of wireless communication, this kind of attack continues to find ways to be effective. However, the development of new techniques to counter this attack is not always straightforward, as even experimenting with possible solutions requires complying with strict FCC regulations [50]. Even though some environments allow for jamming research, e.g., anechoic chambers or Faraday cages, these setups can hardly capture the characteristics and scale of real-world network deployments. To bridge this gap, a DT environment—such as the Colosseum wireless network emulator—could be fundamental in further developing techniques for jamming mitigation research as shown in our previous work in [28] where we implement jamming software within Colosseum to test the impact that jamming signals have within a cellular scenario as well as compare real-world and DT throughput results.

Here, we leverage the GNU Radio-based IEEE 802.11 implementation [17] to deploy two Wi-Fi nodes (Transmitter (TX) and Receiver (RX)) communicating over a 20 MHz spectrum on the Arena testbed [9]. Additionally, we leverage GNU Radio to deploy a jammer (both stationary and mobile) that transmits Gaussian noise signals to hamper the correct functioning of our Wi-Fi network. Our setup can be seen in Figure 23. For the sake of fairness in the transmitted signals, in the stationary case, we deployed our nodes so that the Wi-Fi transmitter and jammer are at the same distance from the Wi-Fi receiver. We consider two common forms of static jamming: (i) jamming through narrowband signals (shown in Figure 24a); and (ii) jamming through wideband signals (Figure 24b).

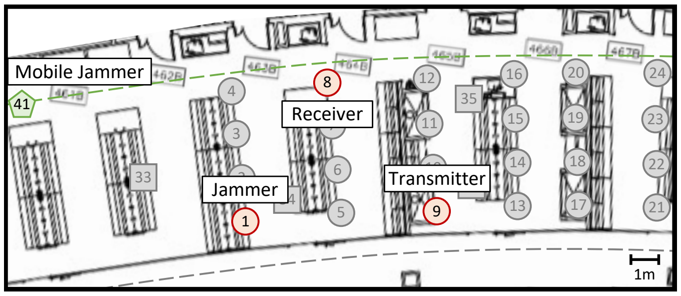


Fig. 23: Location of the nodes in the jamming experiment, consisting of three static (1, 8, 9) and one mobile (41).

The first type of jamming only occupies a small portion of the Wi-Fi bandwidth (i.e., ~156 kHz), resulting in a minimal displacement of the Wi-Fi signals. On the contrary, the latter covers half of the spectrum used by the Wi-Fi nodes (i.e., 10 MHz), causing larger disruptions in the network.

Static Jamming. Figure 24 evaluates how narrowband and wideband stationary jammers impact the throughput and SINR of a Wi-Fi network in the real and DT-based scenarios. In this experiment, the Wi-Fi nodes communicate for 60 seconds, and the jammer starts transmitting at second 20 for a duration of 20 seconds. Specifically, Figure 24a shows Wi-Fi throughput and SINR for the narrowband jamming experiment in both the real-world and DT, while the wideband jamming experiment throughput and SINR results as perceived by the Wi-Fi nodes are shown in Figure 24b.

By looking at the narrowband jamming case, we notice that in the real-world experiment, the Wi-Fi throughput achieves between 5 and 6 Mbit/s when there is no jammer (Figure 24a). Once the jammer starts (at second 20), we notice a rapid decrease in the throughput (i.e., between 37% and 43% decrease). The wideband jammer (Figure 24b), instead, has a more severe impact on the Wi-Fi throughput, causing a performance drop between 94% and 96% (with the through-

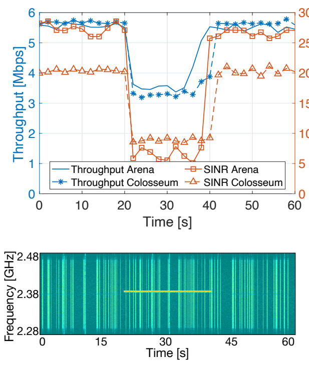
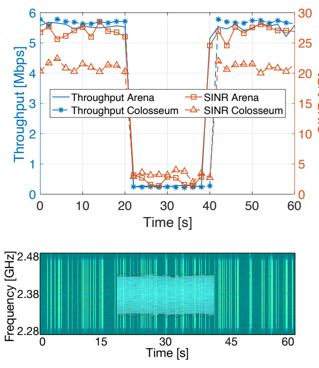


(a) Narrowband (b) Wideband

Fig. 24: Throughput and SINR results on the Arena and Colosseum testbeds of the jamming experiments for the narrowband and wideband use cases. The spectrogram is shown for both forms of jamming, showing the wideband and narrowband signals over a channel.

put achieving values between 220 and 290 kbit/s). In both narrowband and wideband cases, we notice that the behavior obtained in the DT is consistent with that of the real-world scenario. Analogous trends can be seen for the SINR of both signal types, where the narrowband jammer causes an SINR decrease of approximately 20 dB (i.e., ∼77% decrease), while the wideband jammer of approximately 25 dB (i.e., ∼92% decrease) in the real-world scenario. Similarly to the previous case, results are consistent with those of the DT.

Mobile Jamming. Now, we evaluate the impact that a narrowband and wideband mobile jammer (node 41 in Figure 23) moving at pedestrian speed has on the Wi-Fi throughput. Wi-Fi nodes are located as in the previous case, i.e., nodes 8 and 9 in the figure. Results are shown in Figure 25. As expected, the impact of the jamming signal on the Wi-Fi throughput varies as the jammer moves closer or farther from the Wi-Fi receiver, and it also depends on the type of jamming, i.e., narrowband vs. wideband. Specifically, as the jammer gets closer to the Wi-Fi nodes (i.e., seconds 5 to 30) in the narrowband case, we observe a ∼90% decrease in the Wi-Fi throughput in both real-world and DT scenarios (see Figure 25a). A comparable decrease can be observed in the SINR as well, where we notice a clear drop in both the over-the-air Arena case and the DT of about ∼65% in a lookalike trend. Following a similar pattern, in the wideband case, we observe a more pronounced drop as the jammer approaches (i.e., seconds 5 to 30), reaching peaks near 100%

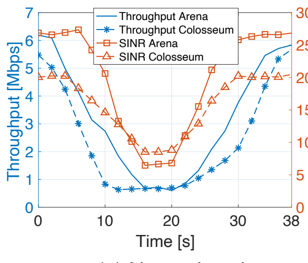
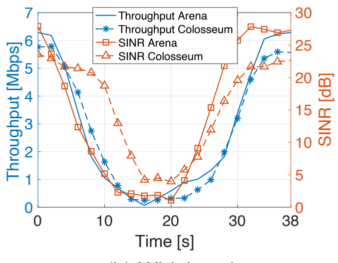


(a) Narrowband (b) Wideband

Fig. 25: Impact of a moving jammer for narrowband (a) and wideband (b) use cases on the throughput and SINR of Wi-Fi nodes on Arena and Colosseum testbed.

drops in throughput and SINR in the closest locations with the nodes at around second 15.

Comparison. As for the cellular experiment, Table 3 shows the normalized cross-correlation results and their averages, considering the maximum values between 10 lags, for each jamming experiment. We observe a strong similarity between the two testbeds in both static and mobile experiments, with individual values consistently above 0.93.

TABLE 3: Normalized cross-correlation results, and their averages, considering the maximum values between 10 lags for each jamming experiment.

Metric	Narrowband	Wideband	Average
Static Throughput	0.996	0.982	0.989
Static SINR	0.986	0.984	0.985
Mobile Throughput	0.982	0.993	0.988
Mobile SINR	0.993	0.935	0.964

Overall, considering all sample experiments in this work, our DTMN is able to achieve an average similarity of 0.987 in throughput and 0.982 in SINR. These results prove the ability of our system to properly emulate various use case experiments with different protocol stacks and scenarios.

6 Related Work

The concept of DT is rapidly gaining momentum in both industry and academia. Initial approaches showcase the use of DTs for industry 4.0 [20], and to assist design, assembly, and production operations in the manufacturing process [21]. A comprehensive literature review on DT-related applications in manufacturing is provided by Kritzinger et al. in [51].

Recently, researchers and practitioners have started to apply the concept of DT to the wireless ecosystem due to the potential of digitalization processes, and easier integration and monitoring of interconnected intelligent components, as Zeb et al. discuss in [52]. Nguyen et al. theoretically discuss how DTs can enable swift testing and validation on real-time digital replicas of real-world 5G cellular networks [53], while Khan et al. provide the architectural requirements for 5G-oriented DTs, mentioning them as key components for the development of 6G networks [54]. He et al. leverage the DTs and mobile edge computing in cellular networks to enhance the creation of digital models affected by the straggler effect of user devices in a Federated Learning (FL) process [55]. Lu et al. incorporate DTs into wireless networks to mitigate long and unreliable communications among users and BS and define a permissioned blockchain-based FL framework for edge computing [56]. Zhao et al. combine DTs with software-defined vehicular networks to learn, update, and verify physical environments to foresee future states of the system while improving the network performance [57].

Overall, the above works agree on the potential of DTs in: (i) assessing the network performance; (ii) creating realistic and accurate system models; (iii) predicting the impact of changes in the deployment environment; and (iv) reacting and optimizing the performance of the network.

The works most similar to our CaST toolchain in modeling and simulating channel characteristics are those of Patnaik et al. [58], Ju and Rappaport [59], Bilibashi et al. [60], and Oliveira et al. [61]. Specifically, Patnaik et al. compare the response of FIR filters with their simulated counterpart [58], while Ju and Rappaport devise a technique to improve the representation of channel impairments and variations for adaptive antenna algorithms in a mmWave channel simulator [59]. Bilibashi et al., and Oliveira et al., instead, leverage ray-tracing approaches to include mobility in the emulated channels in [60] and [61], respectively. However, these works only target specific use cases, and they cannot model generic scenarios and deployments, as instead our CaST toolchain does.

Finally, to the best of our knowledge, there are no practical works that encompass all the various building blocks of a DT system, from channel characterization and modeling to large-scale experimentation on a DT, to real-world validation on an over-the-air testbed, as instead, we carry out in this work.

7 Conclusions

In this paper, we applied the concept of DT to the wireless communication field, and we have presented Colosseum, the World's largest wireless network emulator, as an ideal candidate for a DTMN. We demonstrated its capabilities by digitizing an over-the-air testbed, namely Arena, and by, first tuning, and then running various use case experiments on both testbeds. The results showed that the DT is able to accurately represent the real-world environment. Thanks also to its public release, the Colosseum DT enables the whole research community to properly run wireless experiments and to generate results as accurate as possible to the ones from real-world experimentation.

References

[1] D. Villa, M. Tehrani-Moayyed, P. Johari, S. Basagni, and T. Melodia, "CaST: A Toolchain for Creating and Characterizing Realistic Wireless Network Emulation Scenarios," in *Proceedings of the ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization (WiNTECH)*, Sydney, Australia, October 2022.

[2] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead," *Computer Networks*, vol. 182, pp. 1–28, 29 August 2020.

[3] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28–41, 2019.

[4] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G Networks: Use Cases and Technologies," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, March 2020.

[5] L. S. Vailshery. (2023, March) Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/. Accessed March 2023.

[6] Platforms for Advanced Wireless Research (PAWR). https://www.advancedwireless.org. Accessed March 2023.

[7] K. R. Dandekar, S. Begashaw, M. Jacovic, A. Lackpour, I. Rasheed, X. R. Rey, C. Sahin, S. Shaher, and G. Mainland, "Grid Software Defined Radio Network Testbed for Hybrid Measurement and Emulation," in *Proceedings of IEEE SECON*, Boston, MA, USA, June 2019.

[8] M. Kohli, T. Chen, M. B. Dastjerdi, J. Welles, I. Seskar, H. Krishnaswamy, and G. Zussman, "Open-Access Full-Duplex Wireless in the ORBIT and COSMOS Testbeds," *Computer Networks*, 2021.

[9] L. Bertizzolo, L. Bonati, E. Demirors, A. Al-shawabka, S. D'Oro, F. Restuccia, and T. Melodia, "Arena: A 64-antenna SDR-based Ceiling Grid Testing Platform for Sub-6 GHz 5G-and-Beyond Radio Spectrum Research," *Computer Networks*, vol. 181, p. 107436, 2020.

[10] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, A. Bagga, P. Patel, V. Petkov, M. Seltzer, F. Restuccia, M. Gosain, K. R. Chowdhury, S. Basagni, and T. Melodia, "Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation," in *Proceedings of IEEE DySPAN 2021*, Virtual Conference, December 13–15 2021, pp. 1–9.

[11] M. L. Sichitiu, I. Güvenç, R. Dutta, V. Marojevic, and B. Floyd, "AERPAW Emulation Overview," in *Proceedings of ACM WiNTECH 2020*, London, UK, September 25 2020, pp. 1–8.

[12] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications," *IEEE Access*, vol. 7, pp. 167 653–167 671, 2 December 2019.

[13] R. Saracco. (2022, June 12) 5G network digital twin(s). IEEE Future Directions: https://www.advancedwireless.org. Accessed March 2023.

[14] P. Öhlén. (2021, July) The future of digital twins: what will they mean for mobile networks? https://www.ericsson.com/en/blog/2021/7/future-digital-twins-in-mobile-networks. Accessed March 2023.

[15] Spirent, "Simplifying 5G with the Network Digital Twin," *White paper*, 2019.

[16] I. Gomez-Miguelez, A. Garcia-Saavedra, P. Sutton, P. Serrano, C. Cano, and D. Leith, "srsLTE: An Open-source Platform for LTE Evolution and Experimentation," in *Proceedings of ACM WiNTECH*, New York City, NY, USA, October 2016.

[17] B. Bloessl, "IEEE 802.11 a/g/p Transceiver," November 2022, original-date: 2013-03-24T16:26:18Z. [Online]. Available: https://github.com/bastibl/gr-ieee802-11

[18] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, May 2020.

[19] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication," *White paper*, March 2015.

[20] R. Rolle, V. Martucci, and E. Godoy, "Architecture for Digital Twin implementation focusing on Industry 4.0," *IEEE Latin America Transactions*, vol. 18, no. 05, pp. 889–898, April 2020.

[21] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, September 2019.

[22] Y. Wu, K. Zhang, and Y. Zhang, "Digital Twin Networks: A Survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, May 2021.

[23] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, October 2021.

[24] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing Machine Learning-based xApps for Open RAN Closed-loop Control on Programmable Experimental Platforms," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022.

[25] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "OrchestRAN: Network Automation through Orchestrated Intelligence in the Open RAN," in *Proc. of IEEE INFOCOM*, Virtual Conference, May 2022.

[26] A. Pinto, A. Ashdown, T. Hassan, H. Cheng, F. Esposito, L. Bonati, S. D'Oro, T. Melodia, and F. Restuccia, "An Emulation-Based Framework for Transport Layer Measurements over 5G Wireless Networks," in *Proc. of ACM WiNTECH*, Madrid, Spain, October 2023.

[27] L. Baldesi, F. Restuccia, and T. Melodia, "ChARM: NextG Spectrum Sharing Through Data-Driven Real-Time O-RAN Dynamic Control," in *Proceedings of IEEE INFOCOM*, May 2022.

[28] C. P. Robinson, L. Bonati, T. Van Nieuwstadt, P. Johari, M. Polese, H. Nguyen, C. Watson, and T. Melodia, "eSWORD: Implementation of Wireless Jamming Attacks in a Real-World Emulated Network," in *Proc. of IEEE Wireless Communications and Networking Conf. (WCNC)*, Glasgow, UK, March 2023.

[29] D. Villa, D. Uvaydov, L. Bonati, P. Johari, J. M. Jornet, and T. Melodia, "Twinning Commercial Radio Waveforms in the Colosseum Wireless Network Emulator," in *Proc. of ACM WiNTECH*, Madrid, Spain, October 2023.

[30] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN Gym: AI/ML Development, Data Collection, and Testing for O-RAN on PAWR Platforms," *arXiv:2207.12362 [cs.NI]*, pp. 1–12, July 2022.

[31] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G Era," *Computer Networks*, vol. 176, p. 107284, 2020.

[32] A. Chaudhari and M. Braun, "A Scalable FPGA Architecture for Flexible, Large-Scale, Real-Time RF Channel Emulation," in *Proceedings of IEEE ReCoSoC*, Lille, France, July 2018.

[33] U.S. Naval Research Laboratory. MGEN Traffic Emulator. https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/MGEN. Accessed March 2023.

[34] Remcom, *Wireless InSite Reference Manual, version 3.3.3*, June 2019.

[35] ETSI, "5G Study on channel model for frequencies from 0.5 to 100 GHz," 3GPP TR 38.901 version 14.3.0 Release 14, 2018.

[36] M. Tehrani Moayyed, L. Bonati, P. Johari, T. Melodia, and S. Basagni, "Creating RF Scenarios for Large-scale, Real-time Wireless Channel Emulators," in *Proceedings of IEEE MedComNet 2021*, Virtual Conference, June 2021, pp. 1–8.

[37] Eric Blossom and GNU Radio Project. GNU Radio. https://www.gnuradio.org. Accessed November 2022.

[38] J. R. Buck, M. M. Daniel, and A. C. Singer, *Computer Explorations in Signals and Systems Using MATLAB®*, 2nd ed. Prentice Hall, 2002.

[39] R. Defosseux, B. Djalal, R. Hardy, M. Ismail, and R. Schmidt, "OAI continuous integration home," 2023. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/ci/continuous-integration-home

[40] J. M. Velazquez-Gutierrez and C. Vargas-Rosales, "Sequence Sets in Wireless Communication Systems: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 19, pp. 1225–1248, December 2016.

[41] S. Stanczak, H. Boche, and M. Haardt, "Are LAS-codes a Miracle?" in *Proceedings of IEEE GLOBECOM 2001*, San Antonio, TX, November 25–29 2001, pp. 589–593.

[42] Z. Xinyu, "Analysis of M-sequence and Gold-sequence in CDMA System," in *Proceedings of IEEE ICCSN 2011*, Xián, China, May 27–29 2011, pp. 466–468.

[43] M. Golay, "Complementary series," *IRE Transactions on Information Theory*, vol. 7, no. 2, pp. 82–87, April 1961.

[44] IEEE 802.11 Wireless LAN Working Group, "IEEE Standard for Information Technology – Part 11 - Amendment 3," *IEEE Std 802.11ad-2012 (Amendment to IEEE Std 802.11-2012)*, pp. 1–598, December 2012.

[45] E. García Núnez, J. García, J. Ureña, C. Pérez-Rubio, and A. Hernández, "Generation Algorithm for Multilevel LS Codes," *Electronics Letters*, vol. 46, pp. 1465–1467, November 2010.

[46] D. K. Pradhan and M. Chatterjee, "GLFSR–A New Test Pattern Generator for Built-in-self-test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 238–247, February 1999.

[47] @Last Software and Google. SketchUp. https://www.sketchup.com. Accessed November 2022.

[48] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An Open and Softwarized Prototyping Platform for NextG Systems," in *Proc. of ACM Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, Virtual Conference, June 2021.

[49] R. McMahon, B. Kaushik, and T. Auckland. iPerf. https://iperf.fr/. Accessed November 2022.

[50] "Jammer Enforcement," Mar. 2011. [Online]. Available: https://www.fcc.gov/general/jammer-enforcement

[51] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, June 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[52] S. Zeb, A. Mahmood, S. A. Hassan, M. J. Piran, M. Gidlund, and M. Guizani, "Industrial digital twins at the nexus of NextG wireless networks and computational intelligence: A survey," *Journal of Network and Computer Applications*, vol. 200, p. 103309, April 2022.

[53] H. X. Nguyen, R. Trestian, D. To, and M. Tatipamula, "Digital Twin for 5G and Beyond," *IEEE Communications Magazine*, vol. 59, no. 2, pp. 10–15, February 2021.

[54] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions," *IEEE Communications Magazine*, vol. 60, no. 1, pp. 74–80, January 2022.

[55] Y. He, M. Yang, Z. He, and M. Guizani, "Resource Allocation Based on Digital Twin-enabled Federated Learning Framework in Heterogeneous Cellular Network," *IEEE Transactions on Vehicular Technology*, pp. 1–10, September 2022.

[56] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-Latency Federated Learning and Blockchain for Edge Association in Digital Twin Empowered 6G Networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5098–5107, August 2021.

[57] L. Zhao, G. Han, Z. Li, and L. Shu, "Intelligent Digital Twin-Based Software-Defined Vehicular Networks," *IEEE Network*, vol. 34, no. 5, pp. 178–184, June 2020.

[58] A. Patnaik, A. Talebzadeh, M. Tsiklauri, D. Pommerenke, C. Ding, D. White, S. Scearce, and Y. Yang, "Implementation of a 18 GHz Bandwidth Channel Emulator Using an FIR Filter," in *Proceeding of IEEE EMC 2014*, Raleigh, NC, August 4–8 2014, pp. 950–955.

[59] S. Ju and T. S. Rappaport, "Simulating Motion-incorporating Spatial Consistency into NYUSIM Channel Model," in *Proceedings of IEEE VTC 2018–Fall)*, Chicago, IL, August 27–30 2018, pp. 1–6.

[60] D. Bilibashi, E. M. Vitucci, and V. Degli-Esposti, "Dynamic Ray Tracing: Introduction and Concept," in *Proceedeings of EuCAP 2020*, Copenhagen, Denmark, March 15–20 2020, pp. 1–5.

[61] A. Oliveira, I. Trindade, M. Dias, and A. Klautau, "Ray-tracing 5G Channels from Scenarios with Mobility Control of Vehicles and Pedestrians," in *Proceedings of SBrT 2019*, Petrópolis, RJ, Brasil, September 2019.

Davide Villa received his B.S. in Computer Engineering from University of Pisa, Italy, in 2015, and his M.S. in Embedded Computing Systems from Sant'Anna School of Advanced Studies and University of Pisa, Italy, in 2018. He worked as a Research Scientist in the Embedded Systems and Network Group at United Technologies Research Center in Cork, Ireland, from 2018 to 2020. He is currently pursuing a Ph.D. in Computer Engineering at Northeastern University in Boston, USA. His research interests are on 5G and beyond cellular networks, channel characterization for wireless systems, O-RAN, and software-defined networking for wireless networks.

Miead Tehrani-Moayyed is a Ph.D. candidate in Computer Engineering at Northeastern University in Boston. He is working under Prof. Stefano Basagni's supervision on RF channel models for static and mobile scenarios, from simulations to models for large-scale emulations. His research interests include applying AI/ML algorithms to wireless communication, propagation models for next-generation cellular systems, and computer networks. He received his M.S. in Computer Systems Architecture Engineering from Azad University, Iran in 2013 and his B.S. in Computer Engineering from Shomal University, Iran in 2007.

Clifton Paul Robinson is a Cybersecurity Ph.D. candidate at Northeastern University advised by Professor Tommaso Melodia in the Wireless Networks and Embedded Systems (WiNES) Laboratory. He received his B.S. in Computer Science & Mathematics at Bridgewater State University in 2018, and his M.S. in Cybersecurity from Northeastern University in 2020. His main area of research is in wireless network security, focusing on adversarial signals, and signal processing and detection with deep learning applications.

Leonardo Bonati is an Associate Research Scientist at the Institute for the Wireless Internet of Things, Northeastern University, Boston, MA. He received a Ph.D. degree in Computer Engineering from Northeastern University in 2022. His main research focuses on softwarized approaches for the Open Radio Access Network (RAN) of the next generation of cellular networks, on O-RAN-managed networks, and on network automation and orchestration. He served multiple times on the technical program committee of the ACM Workshop WiNTECH and as guest editor of the special issue of Elsevier's Computer Networks journal on Advances in Experimental Wireless Platforms and Systems.

Pedram Johari is a Principal Research Scientist at the Institute for Wireless Internet of Things at Northeastern University, Boston, MA. Pedram received his Ph.D. in Electrical Engineering from the University at Buffalo in 2018 where he also was a Research Assistant Professor and Adjunct Lecturer in 2018 and 2019. His research interests are in the fusion of AI and future generation of cellular networks (5G and beyond), in particular, focused on spectrum sharing, vehicular communications, full protocol wireless network emulators enabling wireless digital twins, and Internet of wireless medical things. Pedram has collaborated with several research institutions, including University at Buffalo, Georgia Tech, Qualcomm, MathWorks, InterDigital, MITRE, and VIAVI.

Michele Polese is a Principal Research Scientist at the Institute for the Wireless Internet of Things, Northeastern University, Boston, since March 2020. He received his Ph.D. at the Department of Information Engineering of the University of Padova in 2020. He also was an adjunct professor and postdoctoral researcher in 2019/2020 at the University of Padova, and a part-time lecturer in Fall 2020 and 2021 at Northeastern University. His research interests are in the analysis and development of protocols and architectures for future generations of cellular networks (5G and beyond). He has contributed to O-RAN technical specifications and submitted responses to multiple FCC and NTIA notice of inquiry and requests for comments. He received several best paper awards, is serving as TPC co-chair for WNS3 2021-2022, as an Associate Technical Editor for the IEEE Communications Magazine, and has organized the Open 5G Forum in Fall 2021.

Tommaso Melodia is the William Lincoln Smith Chair Professor with the Department of Electrical and Computer Engineering at Northeastern University in Boston. He is also the Founding Director of the Institute for the Wireless Internet of Things and the Director of Research for the PAWR Project Office. He received his Ph.D. in Electrical and Computer Engineering from the Georgia Institute of Technology in 2007. He is a recipient of the National Science Foundation CAREER award. Prof. Melodia has served as Associate Editor of IEEE Transactions on Wireless Communications, IEEE Transactions on Mobile Computing, Elsevier Computer Networks, among others. He has served as Technical Program Committee Chair for IEEE INFOCOM 2018, General Chair for IEEE SECON 2019, ACM Nanocom 2019, and ACM WUWnet 2014. Prof. Melodia is the Director of Research for the Platforms for Advanced Wireless Research (PAWR) Project Office, a $100M public-private partnership to establish 4 city-scale platforms for wireless research to advance the US wireless ecosystem in years to come. Prof. Melodia's research on modeling, optimization, and experimental evaluation of Internet-of-Things and wireless networked systems has been funded by the National Science Foundation, the Air Force Research Laboratory the Office of Naval Research, DARPA, and the Army Research Laboratory. Prof. Melodia is a Fellow of the IEEE and a Senior Member of the ACM.