CaST: A Toolchain for Creating and Characterizing Realistic Wireless Network Emulation Scenarios

Davide Villa, Miead Tehrani-Moayyed, Pedram Johari, Stefano Basagni, Tommaso Melodia
Institute for the Wireless Internet of Things, Northeastern University, Boston MA, 02115, USA
{villa.d,tehranimoayyed.m,p.johari,s.basagni,melodia}@northeastern.edu

ABSTRACT

Large-scale wireless testbeds are being increasingly used in developing and evaluating new solutions for next generation wireless networks. Among others, high-fidelity FPGA-based emulation platforms have unique capabilities for faithfully modeling real-world wireless environments in real-time and at scale, while guaranteeing repeatability. However, the reliability of the solutions tested on emulation platforms heavily depends on the precision of the emulation process, which is often overlooked. To address this unmet need in wireless network emulator-based experiments, in this paper we present *CaST*, a *Channel emulation generator and Sounder Toolchain* for creating and characterizing realistic wireless network scenarios with high accuracy. *CaST* consists of (i) a framework for creating mobile wireless scenarios from ray-tracing models for FPGA-based emulation platforms, and (ii) a containerized Software Defined Radio-based channel sounder to precisely characterize the emulated channels. We demonstrate the use of *CaST* by designing, deploying and validating multi-path mobile scenarios on Colosseum, the world's largest wireless network emulator. Results show that *CaST* achieves ≤ 20 ns accuracy in sounding Channel Impulse Response tap delays, and 0.5 dB accuracy in measuring tap gains.

CCS CONCEPTS

• **Networks → Network performance evaluation**; *Mobile networks.*

KEYWORDS

Ray-tracing, Channel Sounding, Wireless Mobile Networks

ACM Reference Format:
Davide Villa, Miead Tehrani-Moayyed, Pedram Johari, Stefano Basagni, Tommaso Melodia. 2022. CaST: A Toolchain for Creating and Characterizing Realistic Wireless Network Emulation Scenarios. In *Proceedings of 16th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterizatiion (WiNTECH '22)*. ACM, Sydney, NSW, Australia, 8 pages. https://doi.org/10.1145/3556564.3558243

This work was partially supported by the U.S. National Science Foundation under grant CNS-1925601, and by the U.S. Department of Transportation, Federal Highway Administration.

1 INTRODUCTION

The wireless networking industry is experiencing a tremendous growth, as shown by the standardization of 5th generation (5G) technologies and by the vigorous rise of 6G [9]. The need for faster, more reliable, and low-latency wireless technologies is providing a major motivation for researchers to define and develop hosts of new solutions for next generation wireless networks. In parallel, there has been significant interests and promising advancements in the use of Artificial Intelligence (AI) and data-driven methods to address complex problems in the wireless telecommunications domain that are envisioned to largely replace the traditional model-driven techniques in the years to come.

Needless to say, developing new AI-driven telecommunication solutions requires extensive testing in a variety of environments to demonstrate desired performance. However, it is costly and often unfeasible to develop and debug new solutions on large and diverse real-world experimental setups. In this context, large-scale wireless emulation platforms have been widely demonstrated to be a valuable resource to design, develop, and validate new applications in quasi-realistic environments, at scale, and with a variety of different topologies, traffic scenarios, and channel conditions [2, 5, 18]. These network emulators can represent virtually any real-world scenario, also enabling repeatability of experiments.

The reliability of the solutions developed in emulated platforms depends greatly on the precision of the emulation process and of the models of the environment. Most channel emulators are based on Finite Impulse Response (FIR) filters with pre-defined complex-valued taps that represent the characteristics of the channel, as the Channel Impulse Response (CIR) in the baseband. Additional complexity is added by multi-path scenarios with mobile nodes, such as Vehicle-to-everything (V2X) and Unmanned Aerial Vehicle (UAV) communications, which are relevant to next generation networking. Modeling these scenarios is no easy feat. Trade-offs and limitations imposed by the design of the channel emulator, and impairments from hardware-in-the-loop features may compromise the accuracy of the channel modeling process and consequently of the emulated RF environment. We observe, however, that the validation of the emulated channel characterization is often neglected and considered to be true as defined by the model parameters. It is therefore necessary to appropriately evaluate the implementation of the channel models, measure potential emulation errors, and to use the finding to further develop corrective measures to compensate for deviations from desired and expected behaviors.

Validating emulation and simulation models of wireless scenarios has been done before. Patnaik et al., for instance, investigate the difference between an FIR filter response and its simulated twin [14]. Ju and Rappaport consider spatial consistency to better simulate channel impairments in a mmWave channel simulator [11].

Researchers have also exploited ray-tracing to include mobility in emulated channels [1, 13, 14]. To the best of our knowledge, however, these works concern very specific scenarios and experiments.

In this paper, we develop a general, open-source and fully customizable Software-defined Radio (SDR)-based toolchain to streamline the generation and validation of virtually any type of wireless environment that can be implemented into wireless network emulators at scale. Our *Channel emulation generator and Sounder Toolchain (CaST)* brings to the wireless network emulator landscape a fully open, virtualized and software-based channel generator and sounder toolchain. Specifically, our contributions are as follows:

i. We design and develop a streamlined framework to create realistic wireless scenarios with mobility support and based on precise ray-tracing methods for FIR-based emulation platforms such as Colosseum [2].
ii. We develop an SDR-based channel sounder to precisely characterize emulated RF channels. The sounder framework is fully containerized, scalable, and automated to capture gains and delays of the channel CIR taps.
iii. We test and validate multi-path mobile scenarios on Colosseum, showing that *CaST* achieves up to 20 ns accuracy in sounding CIR tap delays, and 0.5 dB accuracy in measuring tap gains.

In addition, *CaST* is openly available to the whole research community.[1] *CaST* enables wireless research community to design, develop, and test their own new realistic channel models resembling accurate wireless environment of their choice by utilizing a variety of input sources, e.g., ray-tracing software, statistical channel modeling tools, and real-world measurements, among others.

The rest of the paper is organized as follows. Sections 2 and 3 describe the two main components of *CaST*, sounding and channel mobility generation, respectively, with an overview of the emulation process of Colosseum. Section 4 provides the validation and experimental results of *CaST*. Section 5 concludes our work.

2 CHANNEL SOUNDING

Wireless network emulators such as Colosseum [2] do not involve communication over-the-air. The RF medium is emulated by means of FIR-based filter taps in the baseband. In this context, the main purpose of channel sounding is validating channel emulation traces rather than acquiring physical environment characteristics. Consequently, the results of the sounding are compared with the original desired channel model that is given to the network emulator as input. These input channel models can be created by different tools, including statistical channel modeling software, ray-tracers, or real-world measurements.

2.1 The Colosseum emulation system

Channel emulators are efficient tools for repeatable experiments of solutions for wireless networked systems. Remotely and publicly accessible emulators include AERPAW, for UAV-enabled applications [18], Drexel Grid, an indoor testbed coupling physical SDR and virtual nodes [5], and Colosseum, the world largest wireless network emulator [2], which we use to develop and test *CaST*. Colosseum is a testbed environment with hardware-in-the-loop used to deterministically model real-world RF scenarios. It provides an open and programmable experimental platform for wireless telecommunication research at scale. The Colosseum Massive Channel Emulator (MCHEM) is built around a network of high-performance Field Programmable Gate Array (FPGA) on the ATCA-3671 boards and Ettus Research USRP X310 radios. It supports 256 fully connected bidirectional RF channels with 80 MHz of instantaneous bandwidth with an RF range between 10 MHz to 6 GHz. The array of Ettus Research USRP X310 High Performance SDR, each equipped with two UBX-160-LP RF daughter cards, is used as the RF front-end, while ATCA-3671 devices are used to perform the channel computations. The USRP X310s are programmed using GNU Radio, UHD and Verilog code, while the ACTA-3671 is programmed using BEEcube Platform Studio along with custom Verilog code. With 128 nodes (256 radio front-ends), Colosseum can emulate up to 65K fully independent channels between each transmitter-receiver pair. Colosseum provides a unique and ideal environment for researchers to create, emulate, run, and test a large amounts of different experiments and real use-case scenarios, allowing a fertile ground where to develop and validate *CaST*.

2.2 *CaST* channel sounding architecture

The *CaST* channel sounder consists of two types of nodes: a transmitter, which sends a known sequence, and one or more receivers. The nodes are developed using the GNU Radio open source SDR development toolkit, which enables the design and implementation of software radios through signal processing blocks via hardware or software in-the-loop. The transmitted sequence and the received data are stored and then post-processed using MATLAB and Python to obtain channel characterization parameters. Finally, the results are analyzed and compared with the original CIR. The corresponding workflow block diagram is shown in Figure 1.

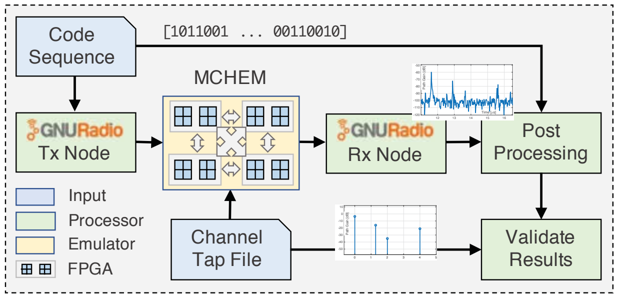


Figure 1: *CaST* channel sounding workflow block diagram.

Transmitter node. The transmitter node is designed as a GNU Radio flowgraph with the main duty to build the sequence code and to send it to the receiver via MCHEM. The sending signal is built upon a *Vector Source* block that streams a vector based on an input sequence code and repeats it indefinitely. We consider a Binary Phase-shift keying (BPSK) modulation where the real part is given by the vector source and transformed in a series of $\{-1, 1\}$, while the imaginary part is given by a null source. This stream of data can be displayed through a series of QT Graphical User Interface (QT-GUI) blocks to show time, frequency, and constellation plots of the signal, and is input to a *UHD: USRP Sink* block. The latter

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

block connects the GNU Radio software to a USRP hardware device and streams the data with different parameters, e.g., clock source, sample rate, and center frequency. The signal is then transmitted by the USRP to MCHEM.

Receiver node. The receiver node receives and stores the data from MCHEM. The signal is received via a *UHD: USRP Source* block that streams samples from a USRP hardware device to the GNU Radio software. The USRP acts as a receiver with appropriate parameters, e.g., clock source, sample rate, and center frequency. The received data can be displayed through a series of QT-GUI blocks, and are saved using a *File Sink* block for further post-processing. It is worth noting that since we operate in a controlled and close environment without any over-the-air transmissions, there is no need to perform any additional control or filtering of the output bandwidth of the transmitted signal to comply with Federal Communications Commission (FCC) regulations. Moreover, MCHEM filters the unwanted signals that are received, leaving only the frequencies of a particular scenario that is running. Thanks to these unique aspects of channel emulators, the transmitter and receiver nodes can be composed of simple yet efficient structures that optimize the channel sounding process and eliminate undesired artifacts.

2.3 Data processing

The received data are processed to obtain two main performance metrics: CIR or $h(t)$, and Path Loss (PL) or $p(t)$. The CIR is useful to understand how well the channel reacts in correlation with a given input and reflects the Time of Arrival (ToA) of the multipath components of a transmitted signal. The PL gives information on the intensity of the received signal power on each multipath as a function of time delay, as well as its attenuation. These are fundamental components that need to be considered in the design of a telecommunication system and useful metrics to validate the accuracy of the channel emulation.

Let $c(t)$ be the known code sequence of N bits used by the transmitter node, and $s^{IQ}(t)$ the modulated transmitted sequence with its In-phase and Quadrature (IQ) components. Similarly, let $r^{IQ}(t)$ be the raw IQ components stored by the receiver node. The CIR IQ components can be computed by separately correlating $r^I(t)$ and $r^Q(t)$ of the received data with the I or Q components of $s(t)$ divided by the inner product of the modulated transmitted sequence with its transpose, as shown in Equations 1 and 2, respectively [7]:

$$h^I(t) = (r^I(t) \otimes s^I(t))/(s^{I^T}(t) * s^I(t)) \tag{1}$$

$$h^Q(t) = (r^Q(t) \otimes s^Q(t))/(s^{Q^T}(t) * s^Q(t)) \tag{2}$$

where $\otimes$ be the cross-correlation between two discrete-time sequences x and y [3] which measures the similarity between x and shifted (lagged) repeated copies of y as a function of the lag. Note that if the considered modulation is BPSK, the denominator will be equal to the length of $c(t)$. The amplitude of the CIR can be obtained by Equation 3:

$$|h(t)| = \sqrt{(h^I(t))^2 + (h^Q(t))^2} \tag{3}$$

and the magnitude of the path gains can be calculated as Equation 4:

$$G_p(t)[dB] = 20log_{10}(|h(t)|) - P_t - G_t - G_r \tag{4}$$

where P_t is the power of the transmitted sequence, and G_t and G_r are the gain of transmitter and receiver amplifiers all in dB, respectively. The PL of each multipath can be retrieved by looking at the maximums of $G_p(t)$ in a certain window where the signal of that multipath is received.

3 MOBILITY IN RF SCENARIOS

The Colosseum MCHEM supports time-variant channels with minimum coherence time of 1 ms. These channels are read from a tap file that consists of FIR complex coefficient values for each pair of nodes in a scenario captured at every 1 ms for the entire duration of any particular scenario. To introduce mobility in Colosseum, we generate a tap file that contains the time-variant CIR. To facilitate supporting various platform inputs to Colosseum, we divide the mobile scenario generation process into two tasks: (i) scenario generator toolchain; and (ii) mobile channel simulator. The toolchain installs the scenario on Colosseum, incorporates the RF channels data and the traffic metadata into the scenario, and assigns a unique scenario ID. On the other hand, the mobile channel simulator estimates the channels between the mobile nodes using Electro-Magnetic (EM) ray-tracing simulator, and implements the movement of the radio nodes in the ray-tracing RF environment. The mobility simulator is implemented on top of a commercial ray-tracing simulator, namely, Wireless InSite (WI) [17], consisting of two steps: (i) sampling the mobile channel using the ray-tracer, and (ii) parsing the ray-tracing outputs to extract the channels for each time instant of emulation. These steps are followed by a channel approximation process that is required to adapt the output channels for emulation [20].

3.1 V2X scenario in Tampa, FL: A use-case

We consider a scenario around the Tampa Hillsborough Expressway in Tampa, FL. In order to simulate the scenario in WI, we need a 3D model of the wireless environment. We obtained such a model from Open Street Map (OSM) in XML format and converted it to an STL file, which is supported by WI. The resulting scenario is depicted in Figure 2.

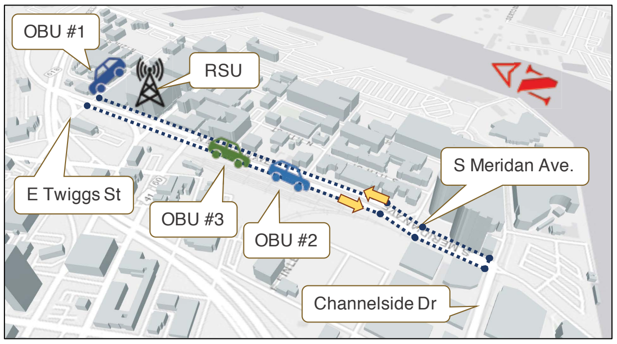


Figure 2: Tampa, FL, V2X scenario simulation environment in WI.

In this scenario, we consider a V2X model with four nodes. One is a Road-Side Unit (RSU) mounted on the traffic light at the intersection of E. Twiggs St. and N. Meridan Ave. The other three nodes are Onboard Units (OBUs) installed on three vehicles: one is stationary and parked in the parking lot of the Tampa Expressway Authority; the other two vehicles are following each other at a constant speed of 25 Mph on Meridan Avenue, from E. Twiggs St. to Channelside Dr., and then back to E. Twiggs St. (Figure 2). The radio parameters of the nodes are listed in Table 1.

Table 1: Wireless parameters for the Tampa simulation scenario.

Parameters	Values for V2X
Carrier frequency	5.915 [GHz]
Signal bandwidth	20 [MHz]
Transmit power	20 [dBm]
Antenna pattern	Omnidirectional
Antenna gain	5 [dBi]
Antenna Height	RSU: 16 [ft], OBUs: 5 [ft]
Ambient noise density	-172.8 [dBm/Hz]

3.2 Sampling mobile channels

Sampling interval and spatial spacing. The effect of mobility is captured by sampling the channels with a predefined sampling time interval of T_s while the transmitter or receivers are moving. This channel sampling concept can be implied in the ray-tracing simulation by spatially sampling the trajectory of the mobile nodes in the scenario with a spacing of $D_i = V_i.T_s$, where V_i is the velocity of the i^{th} mobile node.

For mobility simulation, it is important to consider spatial consistency, which means that the mobile nodes will experience a similar scattering environment with smooth channel transitions due to the motion. We note that spatial consistency does not deal with the small-scale correlation of received power levels, but rather focuses on providing a consistent and correlated scattering environment that a mobile node experiences [11]. We consider the coherence distance of 15 m recommended in 3rd Generation Partnership Project (3GPP) Release 14 [6], which inherently assures the spatial consistency since we use ray-tracing of the physical environment to simulate the spatial and time evolution, so that the generated parameters of two close locations are highly correlated.

Defining trajectory. The spatial sampling concept can be implemented in WI using the "route" or "trajectory" type for transmitters and receivers, where the user can define the trajectory of the mobile nodes in the WI GUI. This trajectory is defined by determining some control points to specify the start, the end, and where the movement direction is changed. Then, WI fills the trajectory with multiple transceiver points and the user can set the spacing between these points. To mimic the channel sampling in real world, we set the spacing with the value calculated as $D = V.T_s$, where Ts is constant for all the mobile nodes, hence the spatial sampling of each node is determined by its velocity. In our use-case scenario, we consider $V = 25$ Mph for mobile vehicles and we set the sampling interval time to 447 ms, hence $D = 5$ m which guarantees the spatial consistency. Given the total length of the path ~ 2 Km, WI generates $N_s = 391$ samples along the trajectory shown with dark blue dots/line in Figure 2.

3.3 Parsing ray-tracing outputs

In the next step, we parse the spatial samples and the valid channels per time from the ray-tracing outputs to extract synchronized channels between the samples of mobile and other stationary nodes. WI runs the ray-tracing process for a total number of channels between each pair of transceivers $N_{ch} = (N_{nodes})^2$, and will store the results at each timestamp, i.e., a total of $N_s.N_{ch}$ channel realizations.

WI stores the ray-tracing output of each individual transmitter in a separate file which represents the channels between that individual transmitter and all the receivers in the scenario. Algorithm 1 shows the parsing of the channel outputs between each pair of nodes at all timestamps for the entire duration of a scenario T_{total}.

Algorithm 1 Mobility simulator algorithm.

```
1:  Number of samples, $N_s = \frac{T_{total}-1}{T_s} + 1$
2:  $channel$= struct (gain, ToA, AoAs, AoDs)
3:  Channel matrix **CH**: 3D matrix ($N_{node}$, $N_{node}$, $N_s$) of channel
4:  $time$= time evolution of simulation
5:  for sample $s = 1$ to $N_s$ do
6:      for each TX, $i$ do
7:          for each RX, $j$ do
8:              if $V_i = 0$ & $s > 1$ then
9:                  CH(i,j,s)=CH(i,j,1)
10:                 continue
11:             find TX($i$) sample, $x$ = min($s$, max TX($i$))
12:             ▷ Read ray-tracing output file TX($i$) for $x$
13:             find RX($j$) sample, $y$ = min($s$, max RX($j$))
14:             ▷ channel = Extract channel RX($j$) for $y$
15:             CH (i,j,s) = channel
16:             $time(s) = (s-1) \times T_s$
    Output: $time$, CH
```

The output of the *CaST* mobility simulator is a 3D matrix structure that consists of N_s 2D matrix pages. Each page stores the channels between the transmitters in rows and receivers in columns.

3.4 Time-variant multi-path parameters

We consider the temporal characteristic of the wireless channel, as an FIR filter, where the CIR varies in time and can be expressed as Equation 5. N_t is the number of paths at time t, c_i is the i^{th} path gain coefficient, and τ_i is the ToA of the i^{th} path, which both vary in time. Further, the path gain coefficient is a complex number which carries the magnitude, a_i and phase shift φ_i of the i^{th} path (Equation 6).

$$h(t,\tau) = \sum_{i=1}^{N_t} \tilde{c_i}(t).\delta(t - \tau_i(t)) \tag{5}$$

$$\tilde{c_i}(t) = a_i(t).e^{j\varphi_i(t)} \tag{6}$$

We obtain the time-variant CIR from the estimated parameters of channel paths for each of the valid channel sampled at time instants t from the ray-tracer output. The ray-tracer reports the paths between transmitters and receivers and calculates ToA, received power, and phase shift of the received signal, as well as the angular characteristics for each path. This process takes into account the path trajectory distance and the reflection coefficient of the materials at each reflection point. In WI, the simulation output finds the paths with power > -250 dBm, which include the ones below the noise floor. Then, we prune the paths with received power lower than the noise floor. This is computed using Equation 7, where N_o, B and F are the ambient noise density $[dBm/Hz]$, the receiver bandwidth $[Hz]$, and the receiver noise figure $[dB]$, respectively.

$$Noise[dBm] = N_o + 10 * \log B + F \tag{7}$$

However, WI does not directly report the gain parameter of the paths for the valid channels at time instant t, so we calculate the gain of the paths as complex numbers using Equation 8, where P_{Tx} is transmit power in dB, and P_{Rx_i} and φ_i are the received power and signal phase per each path $i \in \{1..N_t\}$, respectively.

$$\tilde{c_i}(t) = 10^{(P_{Rx_i}(t) - P_{Tx})/20} * e^{j\varphi_i(t)} \qquad i = 1..N_t \tag{8}$$

We simulate the use-case scenario discussed in Section 3.1 in WI by considering 4 reflections to find the paths between the transmitter and receivers for a reasonable simulation time. We compute the time-variant CIR of the RSU to OBU#3 channel, which is depicted later in Figure 8a (see Section 4). As a final metric, we can evaluate the impact of mobility on path loss, which is one of the important channel parameter for large-scale channel characterization to recognize the fading impact of mobility and multi-paths [12]. To this end, we calculate L_p, link path loss, using Equation 9 which is the magnitude of coherently summation of the coefficients in dB. The results of path loss are analyzed in details in Section 4.4.

$$L_p(t) = -20.\log\left|\sum_{i=1}^{N_t} \tilde{c_i}(t)\right| \tag{9}$$

3.5 Emulation channel taps approximation

Due to computational restrictions and trade-offs, FIR-based channel emulators can only account for limited number of non-zero filter taps (4 taps in the case of Colosseum MCHEM [4]). However, the ray-tracer derived models typically include numerous paths between transmitters and receivers. Furthermore, delays of paths may not be necessarily aligned with the pre-defined step-wise delays of the emulator FIR filter taps. To this end, we use the *tap approximation* framework proposed in our recent work [20] that employs a Machine Learning (ML)-based clustering method to convert the ray-tracer channels into the taps format compatible with the requirements/limitations of the channel emulator. This involves approximating the channel to an acceptable number of taps, aligning the tap delays to the pre-defined indices, and adjusting the dynamic range of the taps while preserving a desirable accuracy from the original channel.

4 EXPERIMENTAL RESULTS

We first run *CaST* in a lab testbed (Figure 3) to tune the parameters in a controlled environment in the absence of channel emulator impairments. One of the main goals of this step is to find a code sequence that can result in a high auto-correlation and a low cross-correlation between transmitted code sequence and received signal, which consequently can reveal channel taps. Secondly, *CaST* is used to understand the behavior of Colosseum emulation by testing a set of synthetic scenarios, i.e., created specifically for the sounding purpose. Finally, quasi-real-world scenarios with static and mobile nodes developed in Section 3 are deployed and characterized.

A customized Linux Container (LXC) image containing *CaST* is created and uploaded to Colosseum. This container has all the required libraries and software for the channel sounding system and its post-processing operations. This enables the re-usability of the sounding with different Colosseum Standard Radio Node (SRN) and scenarios, and allows for the automation of all the processes till the generation of the final results. This image is open-source and will be made available for all Colosseum users in the publicly accessible Network Attached Storage (NAS).

4.1 Lab testbed validation

The lab testbed is shown in Figure 3.

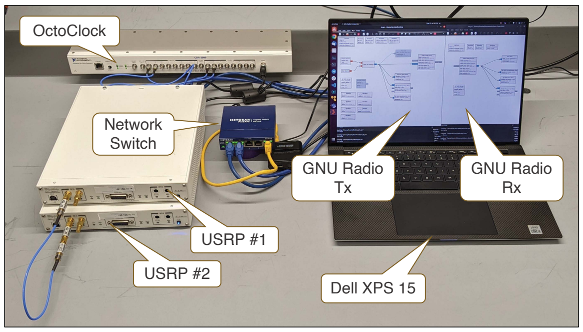


Figure 3: Lab testbed environment.

It consists of two NI/Ettus Research USRP X310 radios, each one equipped with one UBX-160 daughterboard. The radios are synchronized in time and frequency through an NI/Ettus OctoClock clock distributor CDA-2990 which generates the clock internally. A Dell XPS 15 computer is used to program the radios, and to perform the post-processing operations. The default connection between the two USRP consists of a 12 inches SMA cable and three attenuators for a total of 30 dB of attenuation. The sounding parameters for the lab experiments are summarized in Table 2. The gains of the USRP vary between 0 and 15 dB to understand their effect. The code sequence is a Galois Linear Feedback Shift Register (GLFSR) (the effectiveness of different code sequences are analysed in Section 4.2). The receiving period time and data acquisition are set to 3 s.

Figure 4 shows a time frame of the received path gains for the case with 0 dB gains (blue line) and 30 dB gains, consisting of 15 dB on the transmitter and 15 dB on the receiver (orange line).

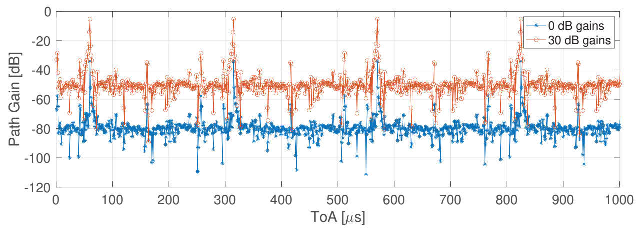


Figure 4: Received path gains lab testbed in 0 and 30 dB gains use cases.

The signal cycles based on the transmitted sequence length, i.e., every 255 sample points, or equivalently every 255 μs since 1 point is equal to $1/sample_rate = 1\,\mu$s. The peaks represent the path loss of the single tap of this lab experiment and is equal to 34.06 dB for the 0 dB case and 5.24 dB for the 15 dB. These numbers are in line with the expectation since we have 30 dB loss due to the attenuators and a few more due to the radios, cable, attenuators, computational imprecision, and some background noise. Moreover, we can notice that in the 30 dB case, the loss is slightly more since there USRP might not add 30 dB total but slightly less. These numbers are used as a reference point for the next experiments and validations.

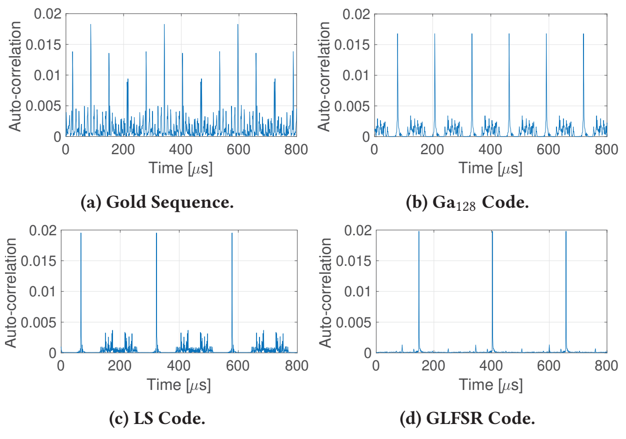
Table 2: Configuration parameters for the lab testbed.

Parameter	Value
Center frequency	1 GHz
Sample rate	Various 1 – 50 MS/s
USRP tx gain	Various 0 – 15 dB
USRP rx gain	Various 0 – 15 dB
Code sequence	GLFSR 255 bits

4.2 Code sequences

Code sequences have been widely investigated in literature given their high utility in many different fields [21] [19]. Good code sequences target a high auto-correlation, i.e., correlation between two copy of the same sequence, and low cross-correlation, i.e., correlation between two different sequences. By exploiting the lab testbed environment described above, four code sequences are tested to find out the one with the CIR that best fits our experiments:

- Gold Sequence [22] of 255 bits created with the Matlab Gold sequence generator System object with its default parameters.
- Golay Sequence type A (Ga_{128}) [10] with a size of 128 bits and defined in the IEEE Standard.
- Loosely Synchronised (LS) Code [16] generated via [8] exploiting just the first codeset of $\{1, -1\}$ without including the Interference Free Window (IFW).
- GLFSR [15] of 255 bits created via the GNU Radio *GLFSR Source* pseudo-random generation block with the following parameters: degree of shift register 8, bit mask 0, and seed 1.



(a) Gold Sequence. **(b) Ga_{128} Code.** **(c) LS Code.** **(d) GLFSR Code.**

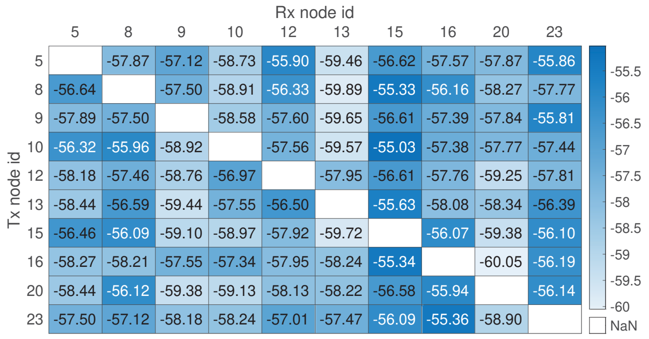
Figure 5: CIR for different code sequences in the lab testbed.

The results of a time frame CIR for each code sequence is show in Figure 5. We can see that the GLFSR sequence outperforms the other three having the highest auto-correlation, and lower cross-correlations compared to the other ones resulting in a cleaner CIR. For this reason, in our experiments the GLFSR is used as the main code sequence source.

4.3 Synthetic scenario validation

The first set of scenarios are synthetic, i.e., manually generated with specific characteristics to validate MCHEM behavior. The set of used parameters is the same as Table 2 but with a sample rate of 50 MS/s to have enough resolution (20 ns per sample) to properly retrieve tap delays and gains.

The first tested scenario is the simplest one single-tap with nominal 0 dB path loss. The results show that the signal properly cycles every 5.1 μs with a recurrent average loss of around 58 dB. This loss can be traced back to Colosseum equipment in-the-loop, consisting of four USRP X310 radios and several cables, and emulation computation approximations. However, the origin of this loss will require further investigation. We can refer to this number as Colosseum base loss. To confirm its value, Figure 6 shows the results for 10 nodes in the same 0 dB scenario.



	5	8	9	10	12	13	15	16	20	23
5		-57.87	-57.12	-58.73	-55.90	-59.46	-56.62	-57.57	-57.87	-55.86
8	-56.64		-57.50	-58.91	-56.33	-59.89	-55.33	-56.16	-58.27	-57.77
9	-57.89	-57.50		-58.58	-57.60	-59.65	-56.61	-57.39	-57.84	-55.81
10	-56.32	-55.96	-58.92		-57.56	-59.57	-55.03	-57.38	-57.77	-57.44
12	-58.18	-57.46	-58.76	-56.97		-57.95	-56.61	-57.76	-59.25	-57.81
13	-58.44	-56.59	-59.44	-57.55	-56.50		-55.63	-58.08	-58.34	-56.39
15	-56.46	-56.09	-59.10	-58.97	-57.92	-59.72		-56.07	-59.38	-56.10
16	-58.27	-58.21	-57.55	-57.34	-57.95	-58.24	-55.34		-60.05	-56.19
20	-58.44	-56.12	-59.38	-59.13	-58.13	-58.22	-56.58	-55.94		-56.14
23	-57.50	-57.12	-58.18	-58.24	-57.01	-57.47	-56.09	-55.36	-58.90	

Figure 6: Path loss heatmap in a 0 dB scenario with 10 nodes.

Each cell represents the average path loss for 2 s reception time between a transmitter node (row id) and a receiver node (column id) using both the first daugherboard. Results confirm 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 operations is around 43 dB, i.e., between the maximum value at 0 dB scenario of 57.55 dB and the minimum one, given by the noise floor at 100 dB. These are fundamental findings that need to be considered when designing scenarios and analysing results.

The next synthetic scenario analyzed consists of four taps with different delay times and path gains (Figure 7 orange stems). The resulting received gains (Figure 7 blue lines) show that the ToA of the different taps are exactly the same, namely $[0 - 1.28 - 2 - 4]$ μs.

The received powers are in line with the expectations. If we add the Colosseum base loss, the gain results (right y-axis) match the original ones (left x-axis), particularly $[3 - 20 - 15 - 8]$ dB losses. By considering a large number of time frames, e.g., fifteen hundreds, we can calculate the relative differences of the received taps over time to obtain information on the accuracy of these measurements.

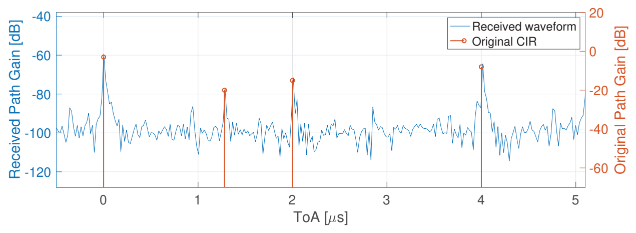
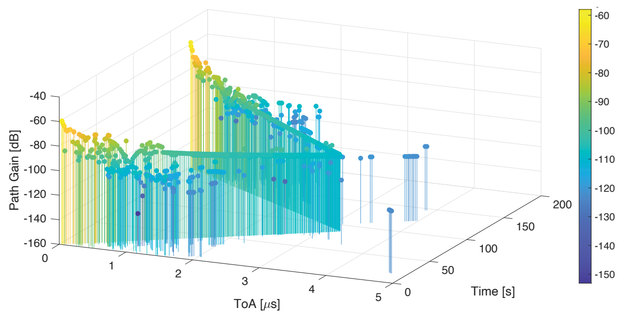


Figure 7: Received path gains and original CIR comparison.

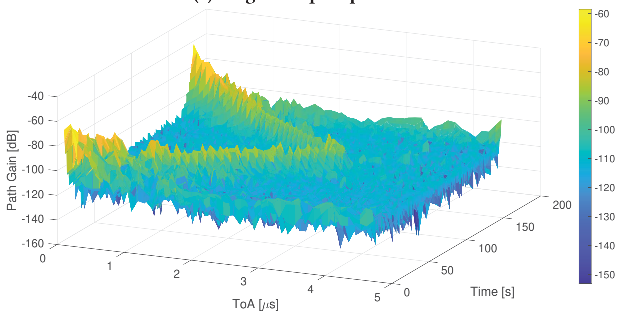
The average difference between the highest first taps of each time frame is in the order of 10^{-6} with a SD of 0.03 dB. The same average happens for the lowest second taps with a slightly higher SD of 0.17 dB. On the other hand, considering the differences between the highest (1st tap) and lowest (2nd tap) for each time frame, we have an average difference of 0.52 dB with a SD of 0.18 dB. These results are a consequence of the contribution of the noise, which is largest in the lowest taps compared to the highest ones. These results prove that MCHEM is emulating the channel correctly for both delay and gain taps, and that the received expected signal is consistent with the original one. Moreover, this shows that *CaST* is able to achieve a resolution of 20 ns, sustaining the sample rate of 50 MS/s and an accuracy on the tap gains measurements of 0.5 dB.

4.4 Mobility scenario validation

In the last set of experiments we test the mobility scenario described in Section 3.1. This scenario has been installed in Colosseum with an increase of 60 dB in all taps, to fall inside the Colosseum dynamic range. The parameters for the sounding process are the same as those listed in Table 2 with 15 dB gains and 10 MS/s sample rate. Since the total scenario time is 175 s and processing all the data together would require extreme memory, the rx time is divided into three chunks of around 60 s each. In this way, each chunk is around 5 GB in size and it takes about 30 minutes to be processed. The results of each chunk are cleaned and merged together to create the ultimate outcome. The received path gains have been adjusted by removing the Colosseum base loss and adding the original 60 dB increase. Figure 8b shows how the received path gains (xy-axis) vary over the scenario time (z-axis). We can notice that the strongest tap resembles the same 'V-shape' behavior seen in the original taps representation (Figure 8a) as a direct consequence of the movement to and from the static RSU node. Moreover, Figure 9 shows the link path loss of node 1 (RSU) and mobile node 3 (OBU#2) against the mobile node 4 (OBU#3).



(a) Original taps representation.



(b) Received path gains.

Figure 8: Results comparison for the mobile use case scenario.

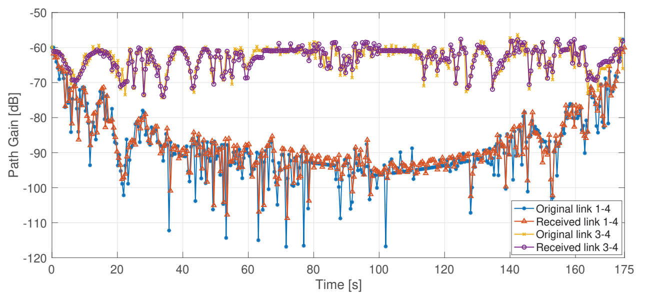


Figure 9: Comparison between original and received path gains for OBU#3 (node 4) against RSU (node 1) and OBU#2 (node 3).

The overall original and received results are fluctuating due to the multi-path fading, but they almost perfectly align between each other. In this case, for link 1-4 a very similar received power trend with a convex 'U-shape' is noticeable. The mobile node is first getting away from the RSU moving south decreasing the gains and increasing the ToA, then the pattern is reserved as the vehicle turns around and travels back to the intersection and the RSU. On the other hand, link 3-4 has a more stable trend since the two vehicle nodes perform the journey together. These results confirm that Colosseum is emulating the channel correctly even in a mobility scenario, and validates the capabilities of *CaST* even when the metrics are changing over time.

5 CONCLUSIONS

This paper presents *CaST*, a fully open, publicly available, software-based and virtualized realistic channel generator and sounder toolchain. It enables researchers to design, create, and validate realistic channel models in scenarios with mobile nodes. The system is validated through experimental tests on a lab setup and on Colosseum by designing, deploying and validating multi-path mobile scenarios. Our results show that *CaST* achieves up to 20 ns accuracy in sounding the CIR tap delays, and 0.5 dB accuracy in measuring the tap gains. Future work on *CaST* include enhancing current results by adding signal propagation delay and phase sounding; accelerating data processing operations by using new FPGA and GPU-based solutions; and improving accuracy by considering techniques that reduce noise and fluctuations.

REFERENCES

[1] D. Bilibashi, E. M. Vitucci, and V. Degli-Esposti. 2020. Dynamic Ray Tracing: Introduction and Concept. In *Proceedeings of EuCAP 2020*. Copenhagen, Denmark, 1–5.

[2] 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. 2021. Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation. In *Proceedings of IEEE DySPAN 2021*. Virtual Conference, 1–9.

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

[4] A. Chaudhari, D. Squires, and P. Tilghman. 2018. Colosseum: A Battleground for AI Let Loose on the RF Spectrum. *Microwave Journal* 61, 9 (September 2018), 22–36.

[5] K. R. Dandekar, S. Begashaw, M. Jacovic, A. Lackpour, I. Rasheed, X. R. Rey, C. Sahin, S. Shaher, and G. Mainland. 2019. Grid Software Defined Radio Network Testbed for Hybrid Measurement and Emulation. In *Proceedings of IEEE SECON 2019*. Boston, MA, 1–9.

[6] ETSI. 2018. 138 901-2018 5G Study on channel model for frequencies from 0.5 to 100 GHz. 3GPP TR 38.901 version 14.3. 0 Release 14.

[7] C. Garcia-Pardo, J.-M. Molina-Garcia-Pardo, J.-V. Rodriguez, and L. Juan-Llacer. 2010. Comparison between Time and Frequency Domain MIMO Channel Sounders. In *Proceedings of IEEE VTC 2010–Fall*. Ottawa, ON, Canada, 1–5.

[8] E. García Núnez, J. García, J. Ureña, C. Pérez-Rubio, and Á. Hernández. 2010. Generation Algorithm for Multilevel LS Codes. *Electronics Letters* 46 (November 2010), 1465–1467.

[9] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi. 2020. Toward 6G Networks: Use Cases and Technologies. *IEEE Communications Magazine* 58, 3 (March 2020), 55–61.

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

[11] S. Ju and T. S. Rappaport. 2018. Simulating Motion-incorporating Spatial Consistency into NYUSIM Channel Model. In *Proceedings of IEEE VTC 2018–Fall)*. Chicago, IL, 1–6.

[12] A. F. Molisch. 2005. Ultrawideband Propagation Channels-theory, Measurement, and Modeling. *IEEE Transactions on Vehicular Technology* 54, 5 (September 2005), 1528–1545.

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

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

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

[16] C. Pérez-Rubio, J. Ureña, Á. Hernández, A. Jiménez Martín, W. P. Marnane, and F. Álvarez. 2007. Efficient Real-Time Correlator for LS Sequences. In *Proceedings of IEEE ISIE 2007*. Vigo, Spain, 1663–1668.

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

[18] M. L. Sichitiu, I. Güvenç, R. Dutta, V. Marojevic, and B. Floyd. 2020. AERPAW Emulation Overview. In *Proceedings of ACM WiNTECH 2020*. London, UK, 1–8.

[19] S. Stanczak, H. Boche, and M. Haardt. 2001. Are LAS-codes a Miracle?. In *Proceedings of IEEE GLOBECOM 2001*. San Antonio, TX, 589–593.

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

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

[22] Z. Xinyu. 2011. Analysis of M-sequence and Gold-sequence in CDMA System. *Proceedings of IEEE ICCSN 2011*, 466–468.