Creating RF Scenarios for Large-scale, Real-time Wireless Channel Emulators

Miead Tehrani-Moayyed, Leonardo Bonati, Pedram Johari, Tommaso Melodia, Stefano Basagni
Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA
Email: {tehranimoayyed.m, bonati.l, p.johari, t.melodia, s.basagni}@northeastern.edu

***Abstract*—Recent years have seen the introduction of large-scale platforms for experimental wireless research. These platforms, which include testbeds like those of the PAWR program and emulators like Colosseum, allow researchers to prototype and test their solutions in a sound yet realistic wireless environment before actual deployment. Emulators, in particular, enable wireless experiments that are not site-specific as those on real testbeds. Researchers can choose among different *radio frequency (RF) scenarios* for real-time emulation of a vast variety of different situations, with different numbers of users, RF bandwidth, antenna counts, hardware requirements, etc. Although very powerful, in that they can emulate virtually any real-world deployment, emulated scenarios are only as useful as how accurately they can capture the targeted wireless channel and environment. Achieving emulation accuracy is particularly challenging, especially for experiments at scale for which emulators require considerable amounts of computational resources. In this paper we propose a framework to create RF scenarios for emulators like Colosseum from rich forms of inputs, like those obtained by measurements through radio equipment or via software (e.g., ray-tracers and electromagnetic field solvers). Our framework optimally scales down the large set of RF data in input to the fewer parameters allowed by the emulator by using efficient clustering techniques and channel impulse response re-sampling. We showcase our method by generating wireless scenarios for Colosseum by using Remcom's Wireless InSite, a commercial-grade ray-tracer that produces key characteristics of the wireless channel. Examples are provided for line-of-sight and non-line-of-sight scenarios on portions of the Northeastern University main campus.**

I. Introduction

The last few years have seen the development of large-scale platforms for experimental wireless research. They allow users to operate on remotely-accessible nodes deployed in realistic configurations [1]. By virtualizing softwarized protocol stacks on general-purpose "white-box" hardware connected to programmable radios these platforms enable experimentation in a host of sectors of the wireless ecosystem, like 5G-and-beyond cellular, ad hoc networks and Internet of Things (IoT) applications, just to name a few. They also promote reproducible results by enabling users to package their solutions in shareable virtualized containers. This way, users can not only replicate original results but also, easily and faithfully, build upon them, pushing forward research and innovation.

Experimental platforms recently available to the wireless community include those of the U.S. National Science Foundation Platforms for Advanced Wireless Research (PAWR) program: POWDER [2], COSMOS [3] and AERPAW [4]. These testbeds allow researchers and practitioners to test their solutions in realistic setups. However, by design, they are constrained to the specific hardware, topology and location of the devices that are physically deployed. Another instrument for experimentation at scale is Colosseum, the world's largest wireless network emulator with hardware in the loop [5], [6]. Colosseum allows users to virtualize softwarized protocol stacks on remotely-accessible Software-defined Radios (SDRs). The emulator is not site specific, in that it is not constrained to a specific hardware topology. Channels among each pair of nodes are *emulated* through so-called *Radio Frequency (RF) scenarios*. This way, the users of Colosseum can test their solutions in a large variety of wireless configurations, modeling urban and rural setups, with variable number of users, traffic and mobility patterns [7]. Although very powerful, emulators like Colosseum are only as good as their ability of accurately capturing and modeling the effects of the wireless channel. Poorly designed scenarios would lead to a non-realistic emulation. Even worse, prototypes developed for such scenarios would not be applicable to the real world.

Achieving accuracy in creating large RF scenarios is challenging, especially for real-time wireless emulators. For instance, emulating the Power Delay Profile (PDP) that represents the complex multi-path characteristics of a wireless link may require a large number of non-zero valued taps of the Finite Impulse Response (FIR) filter that models the PDP [8]. The higher the number of taps $\neq 0$, the more accurate the multi-path model. Unfortunately, an emulator that in real-time must consider the combined effect of multiple signals at each receiver location, needs computational resources and time that are proportional to the number of filter taps $\neq 0$ to combine, which might render any useful experiment at scale impossible. This is why the design of large wireless network emulators requires to strike a judicious balance between accuracy and the resources that are needed to emulate complex RF scenarios. For instance, the designer of Colosseum had to settle for FIR filters with 512 taps, only 4 of which may have non-zero values [6]. This makes creating RF scenarios for large real-time emulator an interesting and non trivial problem.

In this paper, we propose a new framework for creating RF scenarios for time-domain FIR filter-based channel emulators like Colosseum. We start by considering wireless scenarios with rich RF information as produced by professional-grade ray-tracing software. The wireless channel is modeled taking into accounts typical effects such as path loss and multi-path

depending on the environment of a scenario of interest, e.g., considering the presence and make up of buildings and other fixtures and the location of the devices. We then describe a process to replicate the same scenario in the emulator, scaling down the ray-tracer output to determine the non-zero taps of the emulator filters, and what their non-zero value should be. The process is based on techniques such as clustering—to optimally select which taps should be different from zero—and on channel impulse response sampling—to align the tap delays to the input expected by the channel emulator. We showcase our method by generating wireless scenarios for Colosseum by using the commercial ray-tracer Wireless InSite by Remcom [9]. Examples are provided for LTE D2D-based line-of-sight and non-line-of-sight scenarios on portions of the Northeastern University main campus in Boston, MA. For these scenarios we show the effectiveness of our method by computing the emulated path loss and delay spread and comparing them to those of the ray-tracer. Our results show that the proposed method results in 0 dB error path loss, and in a significantly improved approximation of the delay spread with respect to baseline techniques.

We notice that although using Colosseum as an exemplary real-time wireless emulator, our framework is completely general, and can be used to create RF scenarios for any time-domain FIR filter-based channel emulator, starting from an RF-rich input from any ray-tracer or electromagnetic field solver, or from direct measurements in the field.

To the best of our knowledge, this is the first work on creating RF scenarios for large-scale, real-time wireless emulators. Previous works on wireless channel emulation, based on FIR filters or Field Programmable Gate Arrays (FPGAs), are mostly concerned with the definition and implementation of the instrument, rather than with the creation of RF scenarios. Examples include the work by Patnaik et al. that investigates the match between a FIR filter response and a simulated one [10], the work of Olmos et al. on the design and implementation of wide-band channel models for real-time emulators [11], and the papers by Eslami at al. [12], Matai et al. [13] and Buscemi and Sass [14], which propose solutions for modeling a real-time channel in the FPGA, as done for Colosseum [6].

The rest of the paper is organized as follows. A brief primer on Colosseum and its main components is provided in Section II. We describe how to create emulator-compatible scenarios from ray-tracing output in Section III. A demonstration of the proposed method is provided in Section IV. Finally, Section V concludes the paper.

II. Colosseum: A Primer

With a total of 21 server racks, more than 170 high-performance servers, 256 Universal Software Radio Peripherals (USRPs) X310 and the capability of emulating over 65K wireless channels, *Colosseum* is the world's largest network emulator [5]. Originally built by DARPA to support the Spectrum Collaboration Challenge [15], Colosseum is an SDR-based testbed for repeatable—yet realistic—experimental wireless research [6]. The Colosseum SDRs are equally divided among 128 remotely-accessible servers called Standard Radio Nodes (SRNs) that are assigned to the users of the testbed and a Massive Channel Emulator (MCHEM), which performs the real-time emulation of complex wireless environments. Colosseum enables users' experiments through the instantiation of virtualized Linux Containers (LXC) instances, and to operate the USRPs they are connected to, which act as an RF front-end. The USRPs on the SRN-side are connected in a one-to-one manner to the 128 USRPs of MCHEM. Instead of exchanging wireless signals *"en plein air,"* the Colosseum SRNs transmit their waveforms to MCHEM, which emulates the wireless channel specified in the RF scenario, and sends the resulting signal to the receiving SRNs.

In Colosseum, each wireless channel is emulated through 512 complex-valued FPGA-based FIR filter taps, which model the different paths, delays, and other conditions of the wireless channel between transmitter and receiver. The channel taps of Colosseum scenarios have a time-granularity of $1\,\mathrm{ms}$. For each instant of the emulated scenario, tap values for the channel between *any* two SRNs are loaded by MCHEM in real-time and applied—i.e., convoluted—to the signals generated by all the SRNs transmitting at the same time. Because of the elevated complexity and size of emulated scenarios, only four taps have non-zero values for each emulated channel [6], [16]. Even with this limitation, creating scenarios is computationally intensive. For instance, a 10-minute scenario with 50 communicating nodes requires more than 2 hours to build on a computer with 24 CPUs and 96 GB of RAM, and its output occupies more than 100 GB of storage. Once gone through MCHEM, i.e., after MCHEM has convoluted the signal in input from an SRN with the channel taps of the RF scenario, the signal is sent to the intended receiving SRNs. Consequently, choosing the right value for the channel taps of RF scenarios is paramount to guarantee a realistic channel emulation on emulators like Colosseum. The aim and main contribution of this work is to determine a general process for determining optimal values for the non-zero taps of a FIR filter-based wireless emulator.

III. Modeling Realistic RF Channel Emulation

The propagation model of the wireless signal in a channel is often represented by a set of parameters such as path loss, multi-path, and Doppler spread. Path loss describes how the signal power is attenuated in the channel; multi-path indicates the presence of multiple copies of the attenuated and delayed copies of the transmitted signal. Emulating the wireless channel requires finding a mathematical model that represents the characteristics of the channel. A model can be created in different ways, including experimental measurements or simulation software and theoretical analysis. The former can provide very accurate models. However, it is often costly and results to be site specific. Theoretical and simulation-based models are instead more flexible because of extensive capabilities in modeling a variety of complex scenarios. With today software and hardware, professional wireless simulators allow the creation of scenarios at scale in reasonable time.

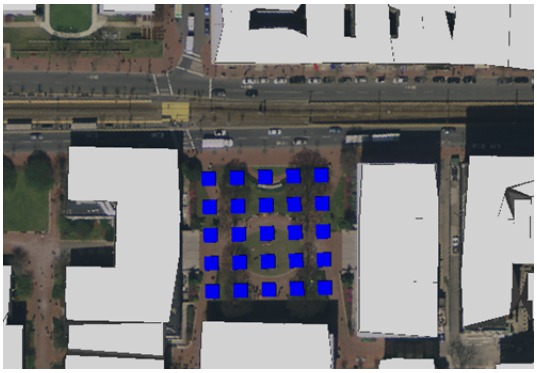

Fig. 1. Northeastern University campus simulation scenario.

In theoretical analysis, path loss is calculated based on physical phenomena such as absorption and spreading. Multi-path, however, can be derived by applying multi-path parameter extraction algorithms, such as SAGE [17], [18], CLEAN [19], and RIMAX [20], [21]. Alternatively, multi-path can be found by using ray-tracer channel simulation tools. In any case, the number of multiple paths is not fixed, and depends on the propagation environment. However, channel emulators can emulate a limited—and often fixed—number of paths. This limit is usually a result of specific signal processing techniques and/or hardware capabilities used in the channel emulation system. Therefore, there is a need for down-scaling the multi-path components with appropriate approximation methods that preserve the characteristics of the channel. In this section, we introduce a process to optimally approximate a wireless channel to a limited number of multi-paths imposed by the channel emulator's requirements. Our approach is a generic process that can be applied to virtually any channel model whether obtained by measurement or ray-tracer simulation.

Ray-tracer. The ray-tracer simulation is a promising method to predict the propagation channel between the transmitter and receiver in a high-frequency regime. It is based on ray optics theory and assumes that the radio waves propagate through the medium while experiencing reflection from surfaces, diffraction from edges of objects, and scattering from non-smooth surfaces. Ray-tracing software considers all combinations of these phenomena and forms the possible multiple paths between a pair of transceivers [22].

Scenario Definition. To illustrate our process of RF scenario generation we start from the output of the commercial ray-tracer Wireless InSite (WI) by Remcom [9]. As a case study, we consider the outdoor portion of the Northeastern University (NU) Boston campus depicted in Fig. 1. To obtain accurate RF results we use a high-resolution 3D shapefile of the area with $\sim$13k faces, and assign concrete and dry earth material properties to the buildings and terrain, respectively. In this setup, we model an LTE Device-to-Device (D2D) communications scenario [23], [24] whose parameters are shown in Table I. Since building and terrain properties are frequency-dependent, we derive the material properties from the recommended ITU model [25] at the carrier frequency of the simulation scenario.

TABLE I
SIMULATION SCENARIO PARAMETERS.

Parameters	LTE Scenario
Carrier frequency [GHz]	2.3
Signal bandwidth [MHz]	20
UE Transmit power [dBm]	24
UE antenna gain [dBi]	0
UE antenna pattern	Omni-directional
UE antenna height [m]	1.5
UE noise figure [dB]	5
Ambient noise density [dBm/Hz]	-167.1

Particularly, Fig. 1 models the NU Krentzman Quadrant, an outdoor space surrounded by three buildings and Huntington Avenue to the North. As potential positions of the LTE User Equipment (UE) we define a grid with 10 meters spacing that covers the area (blue squares in the figure) and provides 625 possible wireless channels. The grid spacing is approximately 100 times the wavelengths, which allows us to obtain a good compromise between capturing the large-scale characteristics of the channel and get a manageable complexity for both the ray-tracer and the emulation scenario generation process.

Assumptions. We configure the ray-tracer to find the paths with up to 3rd order reflections. Higher order reflections are not considered as they suffer from very high attenuation due to the longer paths. Furthermore, to reduce the computational burden, diffraction, transmission, and scattering effects are not considered. With this configuration, the ray-tracer is able to find the most dominant paths that contribute to the total received power. However, even with all these assumptions, the resulting model still includes tens of multi-path components, which cannot be represented in channel emulators with a limited number of non-zero valued FIR taps.

Our process of reducing the RF rich information, and especially many multi-path components of the channel obtained from the ray-tracer to a model for channel emulators with limited number of non-zero taps (such as Colosseum) is made up of two distinct steps. First, we utilize a Machine Learning (ML)-based clustering method to form groups of multi-path components and approximate each group with a single tap that represents the characteristics of all the components in that cluster. Then, we perform a re-sampling step where each of the approximated taps will be aligned to the specific FIR tap indices allowed by the channel emulator. We explain these steps in detail in the following subsections.

A. Channel multi-paths clustering and approximation

The wireless channel consists of Multi Path Components (MPC) between a transmitter and a receiver. The MPC has both spatial and temporal characteristics, which represent angles and Time of Arrival (TOA) respectively. The angles further include Angle of Arrival (AOA) at the receiver and Angle of Departure (AOD) at the transmitter, each of which consists of azimuth and elevation angles. Therefore, these MPC parameters form a multi-dimensional space. Needless to say, clustering multi-dimensional data by visual inspection is impractical especially for a large amount of measured data.

ML-based algorithms, such as *K-means*, are instead practical in such cases.

K-means. In machine learning, the *K-means* algorithm [26] is a well-known unsupervised clustering algorithm to optimally cluster high dimensional data. This algorithm and its variation, *K-power-means,* are widely used in wireless channel modeling applications to obtain accurate channel models [27]–[30]. Their effectiveness in estimating the number of clusters in a wireless channel for improved extraction of temporal and spatial channel characteristics has been studied in [31], [32].

We leverage the *K-means* algorithm as a clustering tool to find similar MPCs in a real world channel model and group them into a given number of clusters that is defined by the number of non-zero taps supported by a channel emulator. Each of these clusters is then identified by a single tap, i.e., the cluster center, that represents an approximation of all the MPCs in that cluster. The final output will be a K-tap channel model that can be used in the channel emulator and is representative of the initially simulated/measured channel model. Therefore, with this approach, we utilize all the channel characteristic information obtained from simulation/measurement and apply them in the approximation process with the objective of minimizing the approximation error.

The *K-means* algorithm takes the initial number of clusters K as *a priori* knowledge and finds cluster centroids by iteratively optimizing the position of the centroids and grouping the data points. This is done by minimizing the sum of distances between the cluster data points to the respective cluster centroid. Mathematically, the algorithm solves the multi-dimensional optimization problem of Equation 1. In this equation, K is the number of clusters, S_i is the set of data points in the ith cluster, x represents the multi-dimensional data points, c_i is the ith cluster centroid, and $d(\cdot)$ denotes the distance function between the data points in cluster S_i and the ith cluster centroid c_i.

$$\arg\min_{S} \sum_{i=1}^{K} \sum_{x \in S_i} d(x, c_i) \tag{1}$$

We note that the *K-means* algorithm is sensitive to the initialization of the centroids [33]. Many prior works have focused on finding an optimum initialization centroid for *K-means* with the goal of improving the quality of clustering and speeding up the convergence of the algorithm. These include random Forgy, MacQueen, and Kaufman, among others [34]–[36]. In this work, we use the random restart method, which relies on running the algorithm several times from different random starting points and choosing the best solution [37].

Generally, the *K-means* algorithm uses squared Euclidean distance as the distance function for the clustering metrics. However, in the context of wireless channel applications, this distance function cannot address the angular periodicity and the scale difference between angles and the delays. Although this function was extended to cope with the angular periodicity [38], it still cannot provide the same scale between the data point dimensions. As *K-means* is biased in favor of feature parameter with greater magnitude, data is normalized ahead of clustering to avoid this undesired effect [39]. Consequently, an alternative solution is Joint Squared Euclidean Distance (JSED), which is a straightforward extension of the Squared Euclidean Distance (SED) that normalizes the delays with the variance of the delays from all considered MPCs.

The JSED distance function makes the delay and angular distance values roughly in the same scale and provides joint 3-dimensional clustering [40]. However, the ultimate solution is Multi-path Component Distance (MCD) as it can effectively improve the clustering performance of channel data over the Euclidean distances [28]. MCD is an intermediate metric to quantify the multi-path separation of the radio channel by jointly considering the distance between angles of arrival/departure and the time of arrival of multiple paths. It represents the distance between two angle of arrivals/departures on the unit sphere, and scales the delay distance with the normalized delay spread. It also introduces a delay scaling factor, ζ to give the delay domain more "importance" when necessary for different technologies in wireless communication. This, for instance, has advantageous effects on automatic clustering for real world data [40]. The MCD distance between two paths i, j is written in Equation 2 and consists of the distances in TOA and angular dimensions (Equations 3 and 4, respectively), where $\Delta_{\tau_{max}} = max_{i,j}\{|\tau_i - \tau_j|\}$ and τ_{std} is the standard deviation of the MPC TOAs. Geometrically, the MCD distance represents the radius of a hyper-sphere in the normalized multi-path parameter distance space.

$$MCD_{ij} = \sqrt{\|MCD_{AOA,ij}\|^2 + \|MCD_{AOD,ij}\|^2 + MCD_{\tau,ij}^2} \tag{2}$$

$$MCD_{\tau,ij} = \zeta . \frac{\tau_i - \tau_j}{\Delta\tau_{max}} . \frac{\tau_{std}}{\Delta\tau_{max}} \tag{3}$$

$$MCD_{AOA|AOD,ij} = \frac{1}{2} \left| \begin{bmatrix} sin(\theta_i).cos(\phi_i) \\ sin(\theta_i).sin(\phi_i) \\ cos(\theta_i) \end{bmatrix} - \begin{bmatrix} sin(\theta_j).cos(\phi_j) \\ sin(\theta_j).sin(\phi_j) \\ cos(\theta_j) \end{bmatrix} \right| \tag{4}$$

Algorithm 1 shows the steps to cluster the MPCs into K groups and output the cluster centroids using the *K-means* algorithm. It takes the MPCs data points and the number of taps K as input, and prunes the paths with powers lower than the noise level. By doing this, it ensures the resulting channel is realistic. In line 5, the centroids are randomly initialized from the set of data points. Then, the MCD distances between all the paths and the K centroids are calculated. Next, each path is assigned to a cluster such that the distance from the cluster centroid is minimum. Finally, the position of the cluster centroids are updated to the *mean* value of the multi-dimensional parameters of the paths assigned to that particular cluster. The algorithm iterates until it reaches a maximum number of iterations or until it converges to a negligible update in the centroids displacements.

Algorithm 1 The K-means MCD clustering algorithm.

Input: MPC data of a specific Receiver Point
Input: K *= number of taps*
Input: P_{Th} *= Noise power level*
Select x_l *from MPC data with power* $\geq P_{Th}$
Initialize centroids c_k^0*, k=1..K randomly selected from* x_l
for $i = 1$ to MaxIteration **do**
  for each path x of x_l **do**
    ▷ Compute distances MCD(x,c_k^{i-1}), k = 1..K
    ▷ Assign path x to the nearest centroid c
    $L_x = \arg\min_k \{MCD(x, c_k^{i-1})\}$
  for each cluster k = 1 to K **do**
    ▷ Update centroids positions c_k^i
    ▷ $c_k^i \leftarrow$ mean of all paths assigned to cluster k
    $c_k^i = \mathbb{E}\{x_l | (L_x = k)\}$
  if $\mathbb{E}\{\Delta(c_k^i,\ c_k^{i-1})\} \leq$ Threshold **then**
    break
Output: c_k

For the channel approximation, our goal is to utilize the *K-means* algorithm to find similar paths in a multi-path model and form them into K clusters that represent a K-tap channel model (in the case of Colosseum $K = 4$). As the *K-means* algorithm satisfies this goal, we feed the number of supported non-zero FIR coefficients as the number of clusters to the algorithm. The output of the algorithm is the label of each MPC, which represents the approximated positions of the taps and the position of the cluster centroids. The next required information to reconstruct the approximated taps is the tap gain that can be calculated by coherent summation of the MPCs gain of each cluster. This can be written as shown in Equation 5, where $|h_x|$ and φ_x are the magnitude and the phase of the MPC gain in the cluster k, respectively.

$$h_{c_k} = \sum_{x \in k} |h_x| \, .e^{j\varphi_x}, \qquad k = 1..K \tag{5}$$

By extracting the TOAs from the cluster centroid and the approximated tap gains from Equation 5, we can reconstruct the CIR from the MPC data at each Receiver Point (RP) that includes only K non-zero taps. This step satisfies the limited number of taps constraint of time-domain FIR filter-based channel emulators.

B. Channel impulse response re-sampling

Time-domain FIR filter-based channel emulators have numerous FIR filter taps separated by a sampling interval to emulate channels with practical maximum excess delay. However, the number of non-zero taps is constrained by design. For example, Colosseum supports 512 taps with a 10 ns sampling interval, only 4 of which can assume non-zero values [41]. In this regard, our clustering approach (as explained in Section III-A) approximates the CIR to satisfy the number of non-zero taps; however, there is no guarantee that the approximated taps will have delays aligned to the resolution of the channel emulator filter taps. Therefore, we need to re-sample the approximated CIR to match the approximated taps' TOA to the channel emulator FIR filter indices.

Algorithm 2 The CIR re-sampling algorithm.

1: *Input: taps = Approximated taps*
2: *Input:* f_s *= FIR filters sampling frequency*
3: *Input:* N *= Number of FIR filters*
4: *Sampling interval:* $d_s = 1/f_s$
5: **for** $n = 1$ to N **do** ▷ Initialization
6: $\hat{d}[n] = n * d_s$
7: $\tilde{h}[n] = 0 + 0j$
8: **for** $k = 1$ to size(taps) **do**
9: $i = round(taps[k].delay/d_s)$
10: $\tilde{h}[i] = \tilde{h}[i] + taps[k].h$
Output: $\tilde{h}$**,** $\hat{d}$

Our re-sampling algorithm is shown in Algorithm 2. First, we initialize the FIR filter coefficients and delays (lines 5-7). This is done by constructing two series representing the filter taps' delays as well as the corresponding taps' gains initialized to zero. Then, we find the FIR tap delay closest to the approximated CIR taps given as input (i.e., the centroid of the associated cluster), and coherently sum up the gains of the MPCs in that cluster. In doing so, we ensure to assign all the approximated taps to their associated FIR filter coefficient indices. In case of multiple taps close to the same index, we sum up the gains of such taps (see equation 5), and assign a single tap index and gain that represents all the associated cluster centroids. As a result, all the approximated taps are matched with the FIR filter tap indices imposed by the channel emulator and represent the corresponding delays. Our approach also guarantees that the resulting approximated channel model has equal to or less than K taps, i.e., maximum number of taps allowed by the channel emulator.

IV. Case Studies

We provide a demonstration of our framework on Coloseum (Section II, [5]). We consider Line of Sight (LOS) and Non-line of Sight (NLOS) stationary scenarios in and around the Krentzman Quadrant of the NU Boston campus (Fig. 1).

We start by modeling the scenarios in Wireless InSite (WI) [9]. We choose the LTE D2D technology for the wireless channel with the parameters as shown in Table I. To keep the ray-tracer computational complexity at bay we configured it to simulate the channels using the reflection ray model for up to 3 orders of reflection. Accordingly, WI finds the paths between the transmitter and receiver locations and calculates the TOA, received power, and the phase of the received signal for each path. This is done by taking into account the path trajectory distance and the reflection coefficient of the materials at each reflection point. The simulation output include all paths with power greater than -250 dBm.

To obtain a realistic channel, we prune the paths with received power lower than the noise level. This is computed using Equation 6, where N_o, B and F are the ambient

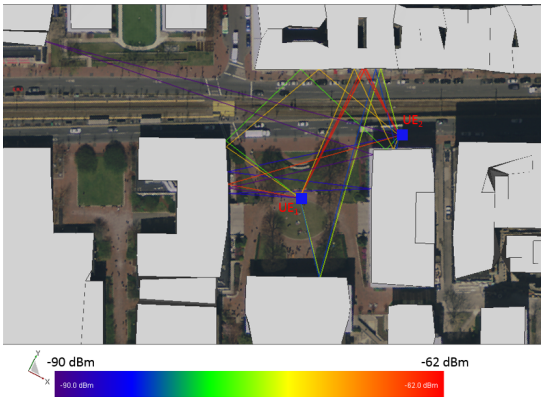

Fig. 2. NLOS scenario propagation paths, color-coded by received power.

noise density [dBm/Hz], the receiver bandwidth [Hz], and the receiver noise figure [dB], respectively. We set the noise level to -89.1 dBm according to a nation-wide measurement campaign for urban scenarios [42].

$$Noise[dBm] = N_o + 10 * \log B + F \tag{6}$$

In addition, as explained earlier in Section III, a time-domain FIR filter-based channel emulator takes the FIR filter coefficients, i.e., the gain of the channel taps, as its input. Since WI does not directly report this parameter, we calculate the gain of the paths as complex numbers using Equation 7. In this equation, P_{Tx} is the transmit power in dB, and P_{Rx_i} and φ_i are the received power and signal phase per each path $i \in \{1..N\}$, respectively.

$$\tilde{h_i} = 10^{(P_{Rx_i} - P_{Tx})/20} * e^{j\varphi_i}, \qquad i = 1..N \tag{7}$$

Non-line of Sight Scenario. The NLOS scenario is depicted in Fig. 2, in which the propagation paths between the UEs are color-coded by the received power strength. In this scenario, after the pre-processing steps, we have 13 paths with the received power above the noise level. The path gains representing the CIR of this scenario are plotted in Fig. 3 We apply our clustering method (see Section III) to cluster the paths, and approximate the gain of the cluster centroids using Equation 5. The MCD distance is configured with $\zeta = 3$ and $K = 4$, i.e., the number of taps supported by Colosseum. Then, we run the algorithm 10000 times with the random restart method and report the best solution, i.e., the one with the minimum Mean Squared Error (MSE).

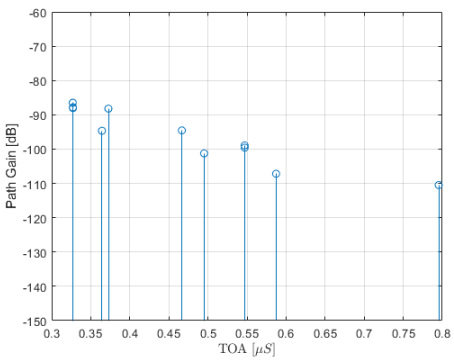

Fig. 3. NLOS scenario CIR.

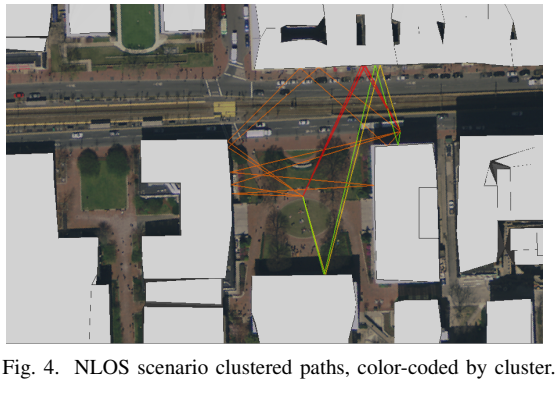

Fig. 4. NLOS scenario clustered paths, color-coded by cluster.

Fig. 4 shows the clustering algorithm results for the NLOS scenario. In this figure, paths in the same cluster are represented with the same color. (We omit the UE markers to improve the figure visibility.) The compactness of the clustering solution in terms of angles and path trajectory, which represents the path TOA, can be visually inspected in Fig. 4. The approximated CIR is shown in Fig. 5, where the thicker stems show the approximated taps for each color-coded cluster. These approximated taps are re-sampled using Algorithm 2, which we leverage to derive the matched index of the channel emulator FIR filter for each tap. The derived complex-value taps have I and Q components that can be readily used as the input to any FIR filter-based channel emulator. Finally, Fig. 6 depicts the re-sampled taps, FIR filter indices, and corresponding tap delays for the four non-zero taps prepared for the Colosseum channel emulator with 512 FIR filters and 10 ns sampling interval.

Line of Sight Scenario. Similarly, we repeat the process for the LOS scenario (shown in Fig. 7). In this scenario, the LOS path is the dominant path with significant gain and lower TOA. Therefore, it should be clustered in a separate group on its own, or with few paths, to have a desirable approximated channel. To do so, we increase the delay scaling factor of *K-means* MCD and set it to $\zeta = 6$ to augment the importance of the TOA over the angles. For the sake of conciseness, we only include the clustered CIR and the approximated CIR for the LOS scenario, which are shown in Fig. 8. We notice that the dominant approximated tap is close to the LOS path,

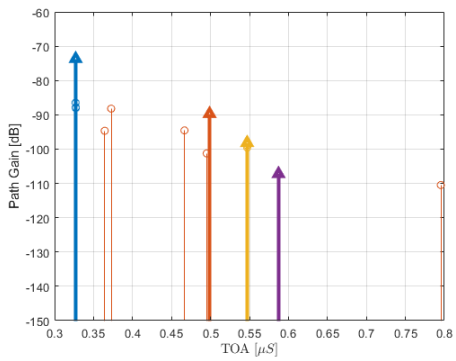

Fig. 5. NLOS scenario clustered CIR and approximated CIR.

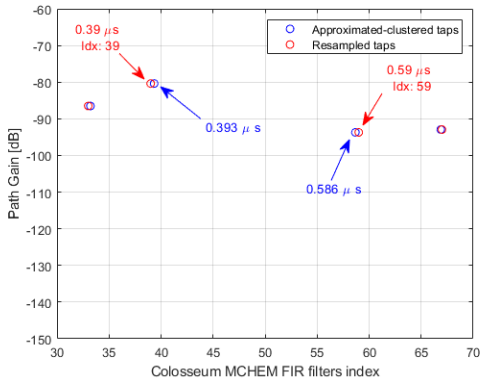

Fig. 6. NLOS scenario in Colosseum. Approximated vs. re-sampled CIR.

which preserves the LOS channel characteristic. An alternate approach would be using *K-power-means* algorithm that shifts the cluster centroids in favor of the higher gain paths. However, this is outside of the scope of this paper.

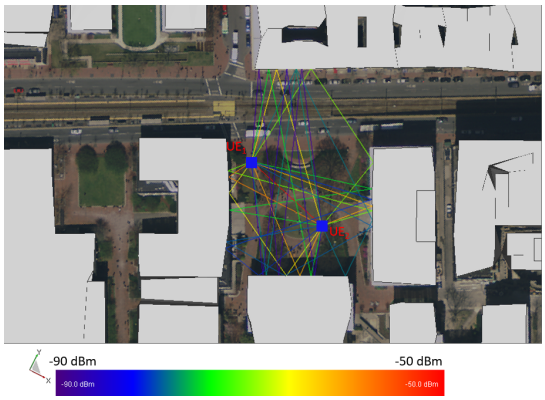

Fig. 7. LOS scenario propagation paths, color-coded by received power.

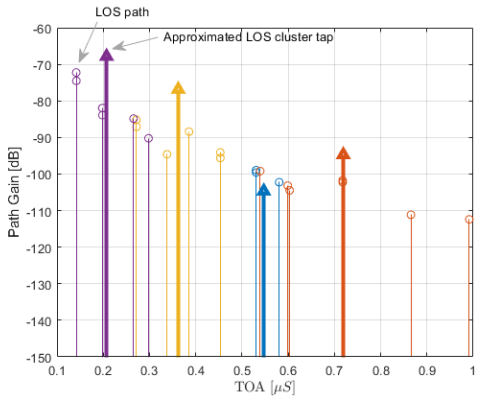

Fig. 8. LOS scenario clustered CIR and approximated CIR.

Lastly, we evaluate the performance of our ML-based approximation process using the path loss and RMS delay spread, which quantify the large-scale channel characteristics [43]. Following these criteria, we benchmark the approximated channels of our proposed method and the baseline method, namely, the *K-strongest path* approximation against the original ray-tracer channels in the LOS and NLOS scenarios previously shown for Colosseum. The *K-strongest path* method

TABLE II
CHANNEL APPROXIMATION: CLUSTERING VS. STRONGEST PATHS.

NLOS scenario	Path Loss [dB]	Delay Spread [nS]
Ray-tracer	72.80	44.80
Strongest paths	75.49	0.10
ML Clustering	72.80	20.28
LOS scenario	**Path Loss [dB]**	**Delay Spread [nS]**
Ray-tracer	69.88	60.21
Strongest paths	67.31	16.85
ML Clustering	69.88	53.28

approximates the channel by simply considering only the strongest paths for the K limited non-zero taps of the channel emulator. As shown in Table II, for both LOS and NLOS scenarios our process delivers the same path loss as estimated for the original ray-tracer channel. This is because our proposed method considers all the paths to approximate the tap gains. In contrast, the *K-strongest path* method imposes roughly 3 dB approximation error in path loss for both scenarios, which is a significant error. Furthermore, our process approximates the LOS channel with a delay spread close to the ray-tracer channel. In the NLOS scenario, the delay spread shows that the proposed method is able to capture the channel characteristic to some extent. However, the *K-strongest path* method fails to deliver reasonable delay spread since it just outputs the K first strongest paths, which are not representative of the delay spread of the channel. These results reveal the key role of the ML-based clustering algorithm in our channel approximation process to deliver a reliable channel model within the scope of the channel emulator.

V. CONCLUSIONS

In this paper, we propose a framework for creating RF scenarios for real-time time-domain FIR filter-based emulators like Colosseum. We start from an RF rich input as that provided by professional-grade ray-tracers to obtain realistic and present a method for optimally scaling down the input RF data to the fewer parameters of the emulator by using efficient clustering techniques and channel impulse response re-sampling. We showcase our method by generating wireless scenarios for Colosseum by using the commercial ray-tracer Wireless InSite. Examples are provided for LTE D2D-based LOS and NLOS scenarios on portions of the Northeastern University main campus. For these scenarios we show the effectiveness of our method by computing the channel path loss and delay spread and comparing them to those of the ray-tracer. Our results show that the proposed method results in 0 dB error path loss, and in a significantly improved approximation of the delay spread with respect to baseline techniques. Future work includes checking the integrity of the emulated scenarios via channel sounding and spectrum analyzers, and by modeling emulated scenarios with user mobility.

REFERENCES

[1] 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. 1–18, December 2020.

[2] J. Breen, A. Buffmire, J. Duerig, K. Dutt, E. Eide, M. Hibler, D. Johnson, S. Kumar Kasera, E. Lewis, D. Maas, A. Orange, N. Patwari, D. Reading, R. Ricci, D. Schurig, L. B. Stoller, K. Van der Merwe, K. Webb, and G. Wong, "POWDER: Platform for open wireless data-driven experimental research," in *Proceedings of ACM WiNTECH*, London, United Kingdom, September 2020.

[3] D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, D. Kilper, T. Chen, J. Kolodziejski, M. Sherman, Z. Kostic, X. Gu, H. Krishnaswamy, S. Maheshwari, P. Skrimponis, and C. Gutterman, "Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless," in *Proceedings of ACM MobiCom*, London, United Kingdom, September 2020.

[4] M. Sichitiu, I. Guvenc, R. Dutta, V. Marojevic, and B. Floyd, "AERPAW emulation overview," in *Proceedings of ACM WiNTECH*, London, United Kingdom, September 2020.

[5] Colosseum. https://www.colosseum.net. Accessed: March 2021.

[6] A. Chaudhari, D. Squires, and P. Tilghman, "Colosseum: A battleground for AI let loose on the RF spectrum," *Microwave Journal*, pp. 22–36, 13 September 2018.

[7] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in O-RAN for data-driven NextG cellular networks," *arXiv:2012.01263 [cs.NI]*, December 2020.

[8] A. Goldsmith, *Wireless Communications*, 2nd ed. Cambridge University Press, March 2020.

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

[10] 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 *Proceedings of IEEE EMC*, Raleigh, NC, USA, August 2014.

[11] J. J. Olmos, A. Gelonch, F. J. Casadevall, and G. Femenias, "Design and implementation of a wide-band real-time mobile channel emulator," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 3, pp. 746–764, May 1999.

[12] H. Eslami, S. V. Tran, and A. M. Eltawil, "Design and implementation of a scalable channel emulator for wideband MIMO systems," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 4698–4709, July 2009.

[13] J. Matai, P. Meng, L. Wu, B. Weals, and R. Kastner, "Designing a hardware in the loop wireless digital channel emulator for software defined radio," in *Proceedings of IEEE FPT*, Seoul, South Korea, January 2012.

[14] S. Buscemi and R. Sass, "Design of a scalable digital wireless channel emulator for networking radios," in *Proceedings of IEEE MILCOM*, Baltimore, MD, USA, November 2011.

[15] DARPA Spectrum Collaboration Challenge (SC2). https://archive.darpa.mil/sc2. Accessed: March 2021.

[16] A. Chaudhari and M. Braun, "A scalable FPGA architecture for flexible, large-scale, real-time RF channel emulation," in *Proceedings of ReCoSoC 2018*, Lille, France, July 9–11 2020, pp. 1–8.

[17] S. Mota, M. O. Garcia, A. Rocha, and F. Perez-Fontan, "Estimation of the radio channel parameters using the sage algorithm," *Radioengineering*, vol. 19, no. 4, pp. 695–702, December 2010.

[18] B. H. Fleury, M. Tschudin, R. Heddergott, D. Dahlhaus, and K. I. Pedersen, "Channel parameter estimation in mobile radio environments using the sage algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 3, pp. 434–450, March 1999.

[19] R. J.-M. Cramer, R. A. Scholtz, and M. Z. Win, "Evaluation of an ultra-wide-band propagation channel," *IEEE Transactions on Antennas and Propagation*, vol. 50, no. 5, pp. 561–570, August 2002.

[20] A. Richter, M. Landmann, and R. Thomä, "Rimax-a flexible algorithm for channel parameter estimation from channel sounding measurements," *Proceedings of European COST*, vol. 45, pp. 26–28, 2004.

[21] R. S. Thoma, M. Landmann, G. Sommerkorn, and A. Richter, "Multidimensional high-resolution channel sounding in mobile radio," in *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, Como, Italy, May 2004.

[22] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, July 2015.

[23] C. Xu, L. Song, and Z. Han, *Resource management for device-to-device underlay communication*. Springer, 2014.

[24] G. Fodor, D. Della Penda, M. Belleschi, M. Johansson, and A. Abrardo, "A comparative study of power control approaches for device-to-device communications," in *Proceedings of IEEE ICC*, Budapest, Hungary, June 2013.

[25] P Series, "Effects of building materials and structures on radiowave propagation above about 100 MHz," *Recommendation ITU-R, P. 2040–1*, July 2015.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[27] R. He, B. Ai, A. F. Molisch, G. L. Stuber, Q. Li, Z. Zhong, and J. Yu, "Clustering enabled wireless channel modeling using big data algorithms," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 177–183, May 2018.

[28] N. Czink, P. Cera, J. Salo, E. Bonek, J.-P. Nuutinen, and J. Ylitalo, "Improving clustering performance using multipath component distance," *Electronics Letters*, vol. 42, no. 1, pp. 33–35, February 2006.

[29] S. Cheng, M.-T. Martinez-Ingles, D. P. Gaillot, J.-M. Molina-Garcia-Pardo, M. Liénard, and P. Degauque, "Performance of a novel automatic identification algorithm for the clustering of radio channel parameters," *IEEE Access*, vol. 3, pp. 2252–2259, 2015.

[30] C. Gustafson, K. Haneda, S. Wyne, and F. Tufvesson, "On mm-wave multipath clustering and channel modeling," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 3, pp. 1445–1455, March 2014.

[31] M. T. Moayyed, B. Antonescu, and S. Basagni, "Clustering validation for mmWave multipath components in outdoor transmissions," in *Proceedings of IEEE WD*, Manchester, United Kingdom, April 2019.

[32] ——, "Clustering algorithms and validation indices for mmWave radio multipath propagation," in *Proceedings of IEEE WTS*, New York, NY, USA, April 2019.

[33] G. Hamerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," in *Proceedings of ACM CIKM*, McLean, VA, USA, November 2002.

[34] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern recognition letters*, vol. 20, no. 10, pp. 1027–1040, October 1999.

[35] M. Meilă and D. Heckerman, "An experimental comparison of model-based clustering methods," *Machine learning*, vol. 42, no. 1-2, pp. 9–29, 2001.

[36] P. S. Bradley and U. M. Fayyad, "Refining initial points for k-means clustering," in *Proceedings ACM ICML*, San Francisco, CA, USA, July 1998.

[37] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, February 2003.

[38] J. SALO, "Automatic clustering of nonstationary mimo channel parameter estimates," in *Proc. 12th Int. Conf. on Telecommun., Cape Town, South Africa, May 2005*, 2005.

[39] R. O. Duda, P. E. Hart *et al.*, *Pattern classification*. John Wiley & Sons, 2006.

[40] N. Czink, P. Cera, J. Salo, E. Bonek, J.-P. Nuutinen, and J. Ylitalo, *Automatic clustering of MIMO channel parameters using the multi-path component distance measure*. Citeseer, 2005.

[41] D. R. Barcklow, L. E. Bloch, S. W. Sweeney, B. E. Ahr, W. J. La Cholter, S. Berhanu, H. S. Bisyir, J. D. Cook, and D. M. Coleman, "Radio frequency emulation system for the defense advanced research projects agency spectrum collaboration challenge," *Johns Hopkins APL Technical Digest*, vol. 35, no. 1, 2019. [Online]. Available: https://www.jhuapl.edu/techdigest

[42] R. Leck, "Results of ambient RF environment and noise floor measurements taken in the US in 2004 and 2005," *Commission for Basic Systems Steering Group on Radio Frequency Coordination*, pp. 1–18, 2016.

[43] A. F. Molisch, "Ultrawideband propagation channels-theory, measurement, and modeling," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 5, pp. 1528–1545, September 2005.