Twinning Commercial Network Traces on Experimental Open RAN Platforms

Leonardo Bonati[†], Ravis Shirkhani[†], Claudio Fiandrino[*], Stefano Maxenti[†], Salvatore D'Oro[†], Michele Polese[†], Tommaso Melodia[†]
[†]Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, U.S.A.
[*]IMDEA Networks Institute, Madrid, Spain

Abstract

While the availability of large datasets has been instrumental to advance fields like computer vision and natural language processing, this has not been the case in mobile networking. Indeed, mobile traffic data is often unavailable due to privacy or regulatory concerns. This problem becomes especially relevant in Open Radio Access Network (RAN), where artificial intelligence can potentially drive optimization and control of the RAN, but still lags behind due to the lack of training datasets. While substantial work has focused on developing testbeds that can accurately reflect production environments, the same level of effort has not been put into twinning the traffic that traverse such networks.

To fill this gap, in this paper, we design a methodology to twin real-world cellular traffic traces in experimental Open RAN testbeds. We demonstrate our approach on the Colosseum Open RAN digital twin, and publicly release a large dataset (more than 500 hours and 450 GB) with PHY-, MAC-, and App-layer Key Performance Measurements (KPMs), and protocol stack logs. Our analysis shows that our dataset can be used to develop and evaluate a number of Open RAN use cases, including those with strict latency requirements.

CCS Concepts

• **Networks** → **Network measurement**; **Network performance analysis**; **Mobile networks**.

Keywords

5G, Open RAN, Mobile Traffic Characterization, RAN Dataset.

ACM Reference Format:
Leonardo Bonati, Ravis Shirkhani, Claudio Fiandrino, Stefano Maxenti, Salvatore D'Oro, Michele Polese, Tommaso Melodia. 2024. Twinning Commercial Network Traces on Experimental Open RAN Platforms. In *The 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3636534.3697320

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/11
https://doi.org/10.1145/3636534.3697320

1 Introduction

The rise of big data, along with advancements in analytics and predictive modeling, has transformed research in computer vision, image processing, and Natural Language Processing (NLP) [8, 18]. The availability of datasets at large [19] led to significant progress in these fields, enabling also advancements of Artificial Intelligence (AI)/Machine Learning (ML) techniques applied to these areas, thanks to the creation of common benchmarks fostering research reproducibility.

In mobile networks, the availability of datasets is more scarce as operators lack interest in publicly releasing such data for reasons that span from privacy concerns for the sensitivity of the data, to legal and regulatory aspects, as well as for strategic advantage over their competitors. Nevertheless, initiatives that have made available mobile traffic data at metropolitan scale exist with a granularity of both multiple minutes (e.g., the Telecom Italia Big Data Challenge [3] and NetMob [23]) and milliseconds (e.g., the Madrid dataset [11]). These datasets have been extremely helpful in advancing cellular technologies. For example, minute-level data can be used to optimize network deployment planning [9, 14], routing [13], and to infer human and economic activities [37]. Similarly, millisecond-level data can be used to optimize resource allocation [7], channel sounding [12], congestion control over mobile networks [35] or to understand specific mechanisms like network ID assignment to users [2].

The recent interest in Open Radio Access Network (RAN)—and specifically O-RAN—deployments, where disaggregated cellular nodes can be reconfigured by AI/ML applications instantiated on RAN Intelligent Controllers (RICs), has resulted in the increased development of data-driven agents for RAN inference and control. However, the development, design, and training of these agents require massive amounts of data. Moreover, to ensure that agents can effectively adapt to a wide range of use cases and network conditions, such data needs to accurately reflect that of real-world deployments.

Related Work. Publicly available testbeds, such as Colosseum [34] and the testbeds of the PAWR program [28], in concert with open-source frameworks such as OpenRAN Gym [5], provide reliable research platforms to collect data at scale in heterogeneous wireless deployments. However, the quality of the collected data, and thus of the trained agents, is not solely related to the fidelity of the Radio Frequency (RF) setup, but also to that of the user traffic. While a substantial body of work has focused on developing testbeds representative of real-world deployments [6, 27, 31, 33, 34, 36], the generation of traffic that matches that of commercial networks is often overlooked. Indeed, most traffic models do not reflect commercial traffic found in real-world deployments, or they do so in a coarse way, which is not suitable for experimentation on testbeds [25].

Among the works that have tried to address this issue, a few leverage Generative Adversarial Networks (GANs) augmented with contextual information of the environment. For instance, [17] builds a knowledge graph to model spatial dependencies and content semantics, and uses it to generate cellular traffic. [38] proposes a deep transfer learning framework that uses historical information on existing cellular deployments to generate traffic for the planning of new sites by leveraging context information of source and target sites. [21] uses a Long Short Term Memory (LSTM) network to capture the temporal correlation of traffic traces clustered from a real-world dataset, and then mimics their structure through a GAN. However, these works do not focus on integrating the reproduced traces within simulators or platforms.

Similarly, [25] captures control-plane traffic of User Equipments (UEs) and models it via Semi-Markov models to evaluate and optimize core network deployments, such as SD-Core. [16], instead, leverages datasets and traffic models to develop a Machine-type Communications (MTC) traffic generator. However, these works only focus on a single type of traffic.

Some works twin real-world traces into the ns-3 simulator. For instance, [20] gathers data from virtual reality headsets and maps it into a burst traffic model. [1] builds a framework to reproduce the characteristics of traces collected by users, while [30] uses smartphones to generate and collect application traffic traces, such as file download and video streaming ones, and reproduce them in ns-3. However, these works only focus on network simulators, with some of them considering traffic from a single application at a time. Instead, our approach is generic and accounts for traffic from the various concurrent applications that users are potentially running.

Contribution. In this paper, we bridge this gap by making a twofold contribution. First, we propose a methodology to twin real-world mobile traffic workloads in Open RAN platforms. We do so by analyzing datasets gathered through cellular sniffers, and statistically reproducing the corresponding traffic through packet generation tools. Then, we collect, analyze, and publicly release[1] a large dataset (more than 500 hours and 450 GB of data) of cross-layer RAN Key Performance Measurements (KPMs) and protocol stack logs collected on an Open RAN deployment instantiated on Colosseum, where UEs are served the twinned network traces. Specifically, we collect Base Station (BS)- and UE-level KPMs from PHY, MAC, and App layers under different RAN configurations representative of AI/ML control policies, number of UEs, and traffic demand. Our dataset provides fine-grained and timestamped cross-layer metrics that make it possible to understand the connection between PHY and MAC KPMs measured at the BS and UEs, control policies, and end-to-end and App-layer KPMs that reflect user experience. Our analysis shows that PHY and MAC KPMs alone are not representative of the user experience, and datasets also need to contain end-to-end KPMs to properly capture the effect that control policies have on it.

2 Primer on Dataset Collection Tools

Passive LTE monitoring tools like FALCON [10] and LTESniffer [15] can be used to gather traffic traces from production BSs. Decoding the Physical Downlink Control Channel (PDCCH), which is sent without encryption, makes it possible to extract per-UE scheduling information. While commercial tools such as Keysight WaveJudge exist, the landscape of open-source monitoring tools for 5G is yet to be shaped because of the complexity of decoding the 5G control channel, due to configuration flexibility and encryption of PDCCH. To the best of our knowledge, 5GSniffer [22] is the only example of such monitoring tool, but it requires side-channel information about cell configuration in the event the production BS varies the Control Resource Set (CORESET) over time.

In the case of FALCON, this tool runs on a Linux host connected to a Software-defined Radio (SDR) and decodes the unencrypted LTE Downlink Control Information (DCI) at Transmission Time Interval (TTI)-level. The procedure keeps the UE identity anonymous and only shows its temporary ID (i.e., the Radio Network Temporary Identifier (RNTI)). For each RNTI, the monitoring tool also provides the ID of the frame containing the traffic allocation, the associated Transport Block Size (TBS), and transmission information, such as Modulation and Coding Scheme (MCS) and utilized Physical Resource Blocks (PRBs). As we will show, this information is sufficient to determine traffic characteristics at the BS- and UE-level, like the total traffic load or BS utilization (at the BS level) or the duration of per-user traffic bursts and idle times between subsequent traffic bursts (at the UE level).

[1]The collected dataset is available at https://github.com/wineslab/open-ran-commercial-traffic-twinning-dataset.

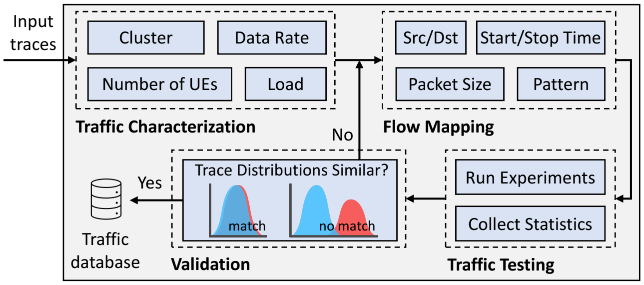

Figure 1: Pipeline to twin traffic traces from real-world datasets.

3 From Real Traces to Open RAN Platforms

Our pipeline to twin traces from real-world datasets is shown in Figure 1. At a high-level, it contains four main steps, described in the remaining of this section: traffic characterization, flow mapping, traffic testing, and validation. The goal of the pipeline is not to replicate transmissions on a packet-by-packet basis,[2] but rather to identify traffic profiles that match those recorded in the wild and to statistically replicate them in Open RAN platforms. Even though we focus on traces from an LTE commercial network, our procedure is generic and it can be applied to 5G traffic traces as well.

3.1 Traffic Characterization

We use a public dataset of LTE network obtained with FALCON from multiple BSs located in different areas of Madrid, Spain [11]. The dataset contains the decoded control channel information of 6 different BSs. In this paper, we focus on 3 BSs from a suburban area of Madrid (BS_1, with carrier frequency at 816 MHz, BS_2 at 1835 MHz, and BS_3 at 2650 MHz), which lets us cover three different spectrum bands.

At a high level, the traffic characterization follows four main steps: (i) aggregate the raw data at the granularity of 1 s; (ii) perform clustering on the pre-processed data (only needed if characterizing traffic profiles for network slices); (iii) aggregate the pre-processed or clustered data over temporal windows W; and (iv) compute per-UE statistics suitable for flow mapping. The input traces from the real-world dataset are loaded into memory by extracting a subset of data related to a temporal window of size W, which we treat as an aggregated data point to derive per-UE statistics on the traffic to twin within the window. We use the Toeplitz Inverse Covariance-Based Clustering (TICC) method to define traffic characteristics per slices, a key functionality of 5G networks [29] that, however, did not exist in LTE. This technique segments multivariate time series into distinct clusters. Rather than considering each point in isolation, TICC uses a sliding window to group observations within their temporal context. Since the original dataset comes with a too fine-grained resolution (i.e., at ms-level), we aggregate this information at 1 s granularity through a rolling-average strategy.[3] As an example, Figure 2 illustrates a trace for a BSs of the dataset that can be divided in $C = 3$ clusters with TICC: c_1 is associated to conditions with high load and BS resource utilization; c_2 is similar to c_1 but has a lower variability; c_3 identifies periods where at least one variate is low.

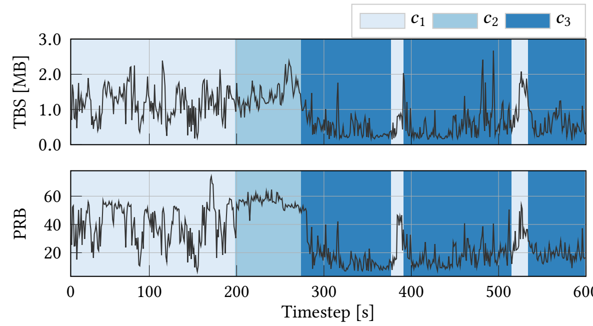

Figure 2: Example of clustering of traffic of a production BS operating with a 20 MHz channel bandwidth to identify slicing profiles.

We derive information that includes the clusters that are present in the trace, data rate, load, and average number of UEs for the considered window W. This procedure is necessary to identify the traffic profiles and their statistical properties, which are needed to twin the same statistical profile in the experimental platforms. An example is provided in Figure 3, where we process data from the real traces of Figure 2 using a window of size $W = 1$ minute. This makes it possible to convert the pre-processed data with granularity 1 s into aggregated per-UE traffic profiles with larger granularity, and to capture their statistical properties. These can be then used, for instance, to train AI/ML control policies.

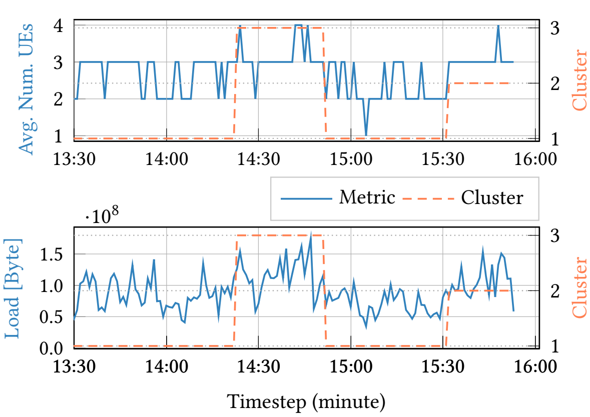

Figure 3: Snapshot of traffic of a production BS. We report traffic load, average number of UEs and cluster for windows of $W = 1$ minute.

[2]Note that this is generally not possible as traffic traces keep track of cell load and resource utilization only, but do not include transmitted packets.

[3]We experimented with much finer aggregation levels, e.g., at 10 ms, the duration of an LTE frame. In this case, TICC takes longer to define the clusters, but we do not experience major differences in the cluster definition.

```
1 70 ON 1 UDP DST 172.16.0.3/5000 PERIODIC [560.39 1250]
2 70 ON 2 UDP DST 172.16.0.4/5000 PERIODIC [560.39 1250]
3 70 ON 3 UDP DST 172.16.0.5/5000 PERIODIC [560.39 1250]
4 70 ON 5 UDP DST 172.16.0.7/5000 PERIODIC [10 125]
5 70 ON 6 UDP DST 172.16.0.8/5000 PERIODIC [10 125]
6 70 ON 7 UDP DST 172.16.0.9/5000 PERIODIC [10 125]
7
8 130 OFF 1
9 130 OFF 2
10 130 OFF 3
11 130 ON 1 UDP DST 172.16.0.3/5000 PERIODIC [512.05 1250]
12 130 ON 2 UDP DST 172.16.0.4/5000 PERIODIC [512.05 1250]
13 130 ON 3 UDP DST 172.16.0.5/5000 PERIODIC [512.05 1250]
14 130 ON 4 UDP DST 172.16.0.6/5000 PERIODIC [512.05 1250]
15 130 ON 8 UDP DST 172.16.0.10/5000 PERIODIC [10 125]
16
17 190 OFF 1
18 190 OFF 2
19 190 OFF 3
20 190 OFF 4
21 190 OFF 8
22 190 ON 1 UDP DST 172.16.0.3/5000 PERIODIC [555.73 1250]
23 190 ON 2 UDP DST 172.16.0.4/5000 PERIODIC [555.73 1250]
24 190 ON 3 UDP DST 172.16.0.5/5000 PERIODIC [555.73 1250]
```

Listing 1: Example of generated MGEN script to run at the BS. The script generates traffic flows for each UE connected to the BS.

Traces will be then used as input to the traffic generation tool on the experimental platforms, as discussed next.

3.2 Flow Mapping

Once the statistical behavior of each traffic flow is fully characterized, we map the statistical traffic flow information into traffic profiles compatible with Multi-Generator (MGEN) [32]—a traffic generator developed by the U.S. Naval Research Laboratory—to generate data to be exchanged between BS and UEs. Source and destination of the real-world traces are mapped to IP addresses of the BS and UEs available in the testbed, together with the size of the packets and the start and stop time of each traffic flow expressed as the amount of seconds from the script start. The traffic pattern also needs to be mapped to one supported by MGEN, which include Poisson-distributed, periodic, and burst traffic, among others. Additional parameters, such as the number of parallel flows among BS and UEs, are also specified at this time. Once this mapping is complete, MGEN scripts with instructions to generate the twinned traffic are built for every node used in the experiment performed on the testbed.

An example of an MGEN script that specifies per-UE downlink traffic from the BS is shown in Listing 1. For the sake of visualization, we only report instructions for the first 190 s of the experiment, while the complete script contains instructions for approximately 30 minutes. In this example, we consider a scenario with a BS and 8 UEs, which corresponds to the Open RAN deployment used to test and validate the twinned traffic (see Section 3.3). In accordance with the real-world traffic traces, the UEs are divided in two service classes: UEs 1-4 demand Enhanced Mobile Broadband (eMBB) traffic, UEs 5-8 Ultra Reliable and Low Latency Communications (URLLC) traffic. Even though here we only illustrate how to map downlink traffic, it is worth noticing that the same approach can be applied to uplink traffic as well, and can be extended to additional traffic classes, RF scenarios, and number of UEs. At second 70 of the experiment, i.e., after allowing some time for the UEs to connect to the BS, traffic flows for UEs 1-3 and 5-7 start. eMBB UEs are sent constant bitrate (called "periodic" by MGEN) traffic at 560.39 messages/s and messages of 1250 byte (`PERIODIC [560.39 1250]`), while URLLC UEs are sent constant bitrate traffic at 10 messages/s and messages of 125 byte (`PERIODIC [10 125]`), as shown in lines 1-6. UEs 4 and 8 are not sent any traffic initially, to be consistent with the real-world traces. At second 130, traffic flows for the eMBB UEs change. This is achieved by stopping the current flows (e.g., `130 OFF 1`) and starting new ones with different characteristics (`512.05 1250`). At the same time, two additional flows, one for an eMBB UE (UE 4) and one for an URLLC one (UE 8) are also started. This is shown in lines 8-15 of the listing. The flows for UEs 5-7, instead, remain active. Similarly, at second 190, the flows for UEs 1-3 are modified again, and those for UEs 4 and 8 are stopped. Overall, this lets us flexibly modify the traffic flows through scripted instructions that are then executed at run time. Further details on the MGEN syntax and capabilities can be found in the MGEN documentation [32].

3.3 Traffic Testing and Validation

After twinning the traces from the real-world dataset, we test the MGEN scripts of Section 3.2 by running experiments on the Open RAN platform. We deploy a cellular network with BS and UEs, and leverage the MGEN scripts to generate traffic to be exchanged among them. At the experiment run time, we collect traffic statistics and KPMs from the protocol stack of the RAN nodes, and from MGEN. These KPMs correspond to those an operator would provide for AI/ML agent design, and that traffic sniffers can only partially capture.

Statistics and KPMs are used to validate that the twinned traffic traces represent the profiles of the traces from the real-world dataset. This is done by normalizing the twinned and real-world traffic distributions (e.g., their Probability Density Functions (PDFs)) so that the x-axis takes values in $[0, 1]$, and comparing their similarity. This can be done through methods such as the Kolmogorov–Smirnov (K-S) test [24] and by defining a similarity criterion (e.g., K-S distance below a user-defined threshold). In case the similarity criterion is satisfied, the MGEN scripts are saved in a traffic database. Otherwise, the pipeline goes back to the flow mapping step of Section 3.2 to further tune the MGEN parameters, after which testing and validation are performed anew. This process repeats until the similarity criterion is satisfied.

Table 1: Resource allocation to the eMBB and URLLC slices.

Slice Resource Allocation	eMBB PRBs	URLLC PRBs
slicing_1	9	41
slicing_2	21	29
slicing_3	30	20
slicing_4	39	11
slicing_5	50	0

Table 2: UE allocation to the eMBB and URLLC classes of service.

UE ID	UE IMSI	eMBB Class	URLLC Class
1	1010123456002	x	-
2	1010123456003	x	-
3	1010123456004	x	-
4	1010123456005	x	-
5	1010123456006	-	x
6	1010123456007	-	x
7	1010123456008	-	x
8	1010123456009	-	x

4 Colosseum Open RAN Dataset

We leverage the network traces twinned in Section 3 to collect a large dataset of timestamped KPMs from an Open RAN deployment instantiated on the Colosseum testbed. This dataset can be used to design and train AI/ML models to be run as O-RAN applications, e.g., xApps, rApps, and dApps.

Colosseum is the largest Open RAN digital twin [34], with 128 pairs of compute nodes and SDRs (USRP X310) interconnected through a channel emulator. Users can leverage softwarized frameworks to instantiate BS and UE protocol stacks, and control them through O-RAN applications deployed on the RICs. Experimentation can be performed under a variety of RF environments reproduced by a channel emulator, which is capable of emulating wireless channel effects such as path loss, multi-path, and fading. Different traffic profiles can also be emulated through a traffic emulation system based on MGEN, as well as via tools such as iPerf.

We leverage OpenRAN Gym [5] to instantiate an Open RAN deployment on Colosseum and to perform an extensive data-collection campaign (more than 500 hours) to collect the dataset described in this section, which totals to more than 450 GB of data. For each experiment, we deploy a BS and up to 8 UEs belonging to the eMBB and URLLC classes (4 UEs each), which are allocated to separate slices of the BS. BS and UEs are based on the srsRAN software, and are instantiated in an emulated RF propagation environment corresponding to a cellular deployment in a neighborhood of Rome, Italy. To be in line with the twinned real-world LTE dataset, our BS leverages a frequency division duplexing configuration over 10 MHz of spectrum. We use the MGEN scripts of Section 3.2 to generate downlink traffic flows among the BS and UEs.

We collect more than 35 timestamped KPMs from the protocol stack of the BS (reported by the UEs or measured directly), from that of the UEs, and from MGEN, under different number of UEs and traffic demand. Protocol stack KPMs include PHY- and MAC-layer KPMs, while MGEN ones are from the App layer. Our experiments span different clusters identified in the Madrid dataset, as well as different schedulers and resources allocated to the eMBB and URLLC slices, which are representative of different AI/ML control policies.

Slice resources are computed in terms of PRBs that the BS is allowed to use for each one of them out of a budget of 50 PRBs (i.e., 10 MHz of spectrum). The slicing configurations that we consider are shown in Table 1, while we consider Round Robin (RR) (scheduling 0) and Proportional Fair (PF) (scheduling 2) as scheduling algorithms.[4] In the dataset, eMBB is marked as slice 0, URLLC as slice 1. UEs are allocated to the eMBB and URLLC classes, and, hence, slices, based on their International Mobile Subscriber Identity (IMSI), as reported in Table 2. The most relevant PHY- and MAC-layer KPMs collected from the protocol stacks of BS and UEs,[5] among which there are throughput, MCS, and buffer occupancy, are reported in Table 3. App-layer KPMs, instead, are computed by the MGEN receiver running at each UE that leverages information enclosed in the payload of the packets sent by the MGEN transmitter running at the BS. These include the timestamp at which packets were sent and received, as well as sequence number and payload size (in bytes) for each packet, and allow for the computation of statistics such as latency, throughput and packet loss.[6]

Table 3: Sample of per-UE protocol stack KPMs collected at the BS.

Metric	Description
dl_buffer [bytes]	Occupancy in bytes of the downlink buffer queue with the data to be transmitted to the UE
dl_mcs	Downlink MCS
tx_brate downlink [Mbps]	Downlink throughput in Mbps
tx_pkts downlink	Number of downlink transmitted packets
dl_cqi	Downlink CQI reported by the UE
sum_requested_prbs	Sum of the PRB needed to serve the UE*
sum_granted_prbs	Sum of the PRB granted to serve the UE*

*These are the total requested or granted PRBs over the 250 ms logging window.

Finally, we also collect logs from the protocol stacks of BS and UEs. These are stored in files named `enb.log` and `ue.log`, respectively, and can be used, for instance, for the analysis of the RNTI timer expiration, as demonstrated in [2].

[4]OpenRAN Gym also includes the Waterfilling (WF) algorithm (scheduling 1). Datasets including WF are described in [4, 29], but they neither twin traffic from commercial traces, nor include App-layer KPMs.

[5]BS KPMs are stored in files named `<ue_imsi>_metrics.csv` (one for each active UE) and in files named `enb_metrics.csv` (cell-wide KPMs); UE KPMs in files named `ue_metrics.csv`.

[6]These metrics are recorded at each active UE in a file named `mgen.csv`.

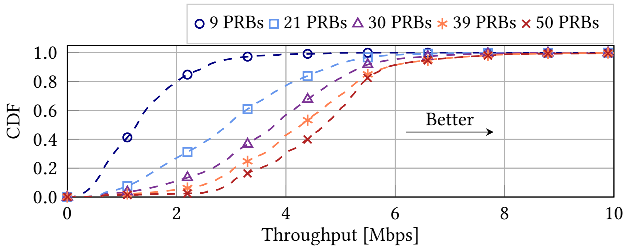

Figure 4: CDF of MAC-layer downlink throughput of eMBB UEs for different slicing configurations.

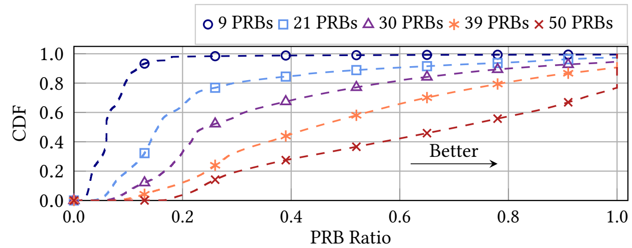

Figure 6: CDF of ratio between granted and requested PRBs for eMBB UEs for different slicing configurations.

5 Dataset Analysis

In this section, we overview relevant KPMs collected in our dataset for the slicing configurations shown in Table 1. These metrics include downlink throughput for both MAC and App layers, PRBs allocated to the slices, end-to-end latency reported by MGEN, and Channel Quality Information (CQI). While PHY- and MAC-layer KPMs are reported directly in the dataset files, App-layer ones have been computed from the MGEN logs. We derived the end-to-end latency by computing the difference between the receive and transmit timestamp of each packet. For the throughput, instead, we computed the amount of data transmitted over windows of 250 ms, which have been selected to align the MGEN logs with the metrics reported by the protocol stacks of BS and UEs.

Figure 4 shows the Cumulative Distribution Function (CDF) of the MAC-layer downlink throughput of the eMBB UEs for different slicing configurations. We notice that larger PRBs allocations yield higher throughput values, since UEs belonging to this class of service demand larger amounts of traffic. Throughput of the URLLC slice, instead, is omitted as the traffic demand the UEs is satisfied even with few PRBs.

Figure 5 depicts the bar plot of the downlink throughput of the eMBB UEs at MAC (plain bars) and App (hatched bars) layers for the different slicing configurations of Table 1. As

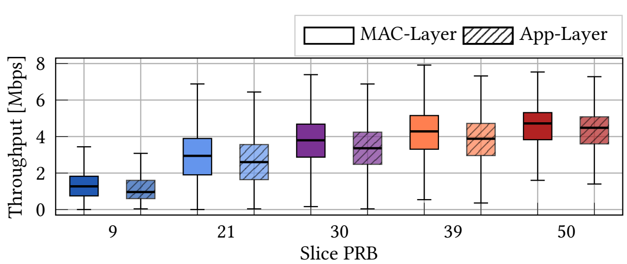

Figure 5: Bar plot of MAC- and App-layer downlink throughput of eMBB UEs for different slicing configurations.

expected, the throughput is higher at the MAC layer because of retransmissions to the UEs in the downlink direction.

We now investigate the ratio between the number of PRBs granted and requested by the UEs (the higher, the better) for different slicing configurations. CDF results for the eMBB UEs are shown in Figure 6, for the URLLC ones in Figure 7. Notice that the `slicing_5` configuration of Table 1 is not shown for URLLC, as this would correspond to 0 PRBs allocated to this slice. The PRB ratio gets higher as we allocate more PRBs to the eMBB UEs (Figure 6). This means that with more resources available, we are more likely to satisfy the traffic demand of the UEs. This is consistent with the results of Figure 4. In the case of the URLLC, instead, fewer PRBs are enough to satisfy the traffic demand of UEs (Figure 7), which are characterized by a less demanding traffic profile.

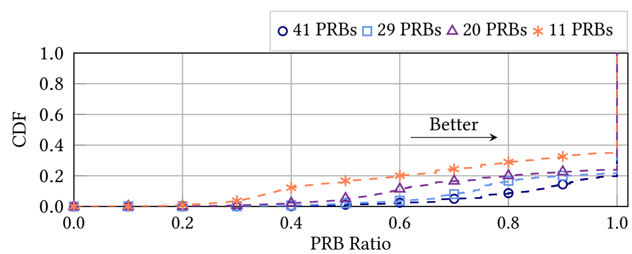

Figure 7: CDF of ratio between granted and requested PRBs for URLLC UEs for different slicing configurations.

The CDF of the end-to-end latency between the BS and UEs is shown in Figures 8 (for eMBB) and 9 (for URLLC). Similarly to what happens in Figure 7, the `slicing_5` case is not shown for the URLLC case. The latency decreases when more PRBs are allocated to the slices, with the worst-case latency being within 2 s for eMBB (9 PRBs case), and 10 ms for URLLC UEs (11 PRBs case) with probability 0.93.

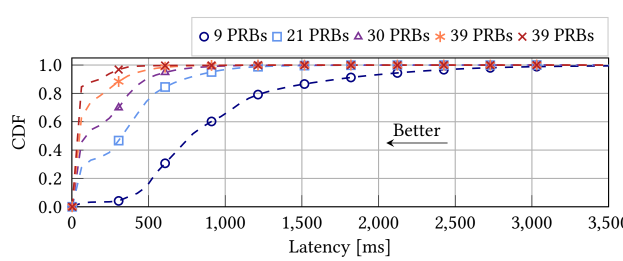

Figure 8: CDF of end-to-end latency of eMBB UEs for different slicing configurations.

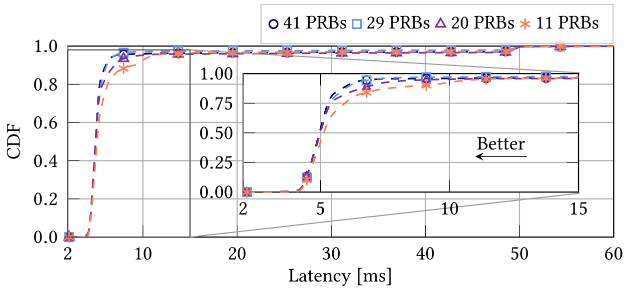

Figure 9: CDF of end-to-end latency of URLLC UEs for different slicing configurations.

Table 4: Probability of URLLC UEs satisfying end-to-end latency requirements (Req.) for different use cases in [26].

Use Case	Req.	41 PRBs	29 PRBs	20 PRBs	11 PRBs
AGV control	5 ms	0.625	0.613	0.474	0.598
Cloud gaming	7 ms	0.957	0.929	0.931	0.887
Robot tooling	10 ms	0.960	0.971	0.955	0.948
AR in smart factory	15 ms	0.964	0.974	0.959	0.961
Fault mgmt in distributed power generation	30 ms	0.968	0.977	0.963	0.970
UAV command and control	100 ms	0.999	0.999	0.999	0.999
Fault location identification	140 ms	0.999	0.999	0.999	0.999

Table 4 shows the probability that the URLLC UEs of our dataset meet the latency requirements of relevant 5G use cases defined in [26] for the different slicing configurations of Table 1. Use cases span from Automated Guided Vehicle (AGV) control with a 5 ms end-to-end latency requirement, to cloud gaming (7 ms), robot tooling (10 ms), Augmented Reality (AR) for smart factories (15 ms), automated fault management for power distribution grids (30 ms), Unmanned Aerial Vehicle (UAV) command and control (100 ms), and identification of fault location along electricity lines (140 ms). We notice that our dataset meets the requirements of most of the URLLC verticals shown in the table, with the 41 and 29 PRBs configurations (`slicing_1` and `slicing_2` in Table 1) outperforming the other ones. In general, more PRBs improve satisfaction of stringent latency requirements (e.g., AGV control, gaming), while less stringent latency requirements can be satisfied even with few PRBs allocated to URLLC UEs (e.g., 11 PRBs in the case of latency requirements $\geq$ 100 ms).

Finally, Figure 10 shows the heat map of the percentage of times the UEs report a certain CQI value to the BS, aggregated over all the slicing configurations of Table 1. The CQI is reported as an integer number from 0 to 15 (the higher, the better), each associated to a specific modulation that will be used for downlink transmissions. We notice that in most of the cases, UEs report a CQI between 8 and 12.

6 Discussion

Although we focused on an LTE dataset [11], our methodology is general. However, starting from an LTE dataset did not allow us to capture some effects typical of 5G networks, such

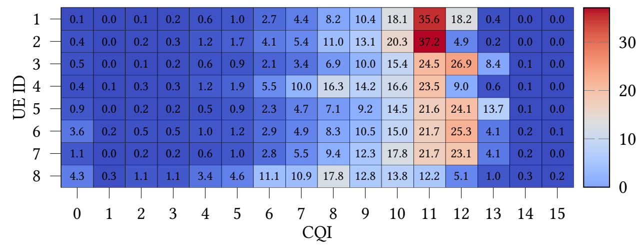

Figure 10: Percentage of times UEs report certain CQI values. CQI 0 indicates that the UE is out of range. UEs 1 and 2 are closer to the BS.

as multiple threads with distinct traffic patterns. In future works, we plan to apply our approach to datasets collected with 5G sniffers [22] and integrate them in Colosseum.

This work offers a tool to replicate data collected over the air in commercial deployments on Open RAN digital twins, and the possibility to convert static traces into dynamic traffic scenarios that can be used for a variety of tasks. For example, datasets generated with our methodology can be used as a data-augmentation mechanism for improving AI/ML-based forecasting and classification. Similarly, our methodology and dataset can also help in advancing EXplainable Artificial Intelligence (XAI), as the twinned traffic can be used to identify small- and macro-scale events, properties, and transitions that cause accuracy drop or unexpected behavior.

Finally, our dataset can be used to generate novel AI/ML algorithms for purposes that go beyond forecasting and traffic analytics, e.g., O-RAN-enabled control. Indeed, static traces are useful to train forecasters and classifiers, but are not well-suited to assess the effectiveness of control policies, and their impact on network performance due to their static nature. Our dataset, instead, has been generated by considering a large number of control policy configurations, which can truly offer insights on which control action delivers the best performance under varying network and traffic conditions.

7 Conclusions

In this paper, we proposed a methodology to twin commercial traffic traces from real-world datasets into Open RAN testbeds to improve the accuracy of cellular experimental research. We demonstrated our approach on the Colosseum Open RAN digital twin, where we collected a large dataset with cross-layer KPMs that we publicly released. Our dataset can be used for the design and evaluation of Open RAN use cases, including those with strict latency requirements.

Acknowledgments

This work was partially supported by the O-RAN ALLIANCE, by the U.S. National Science Foundation under grants CNS-1925601, CNS-2112471 and CNS-2120447, by the U.S. National Telecommunications and Information Administration (NTIA)'s Public Wireless Supply Chain Innovation Fund (PWSCIF) under Award No. 25-60-IF011, by the bRAIN project PID2021-128250NB-I00 funded by MCIN/AEI/10.13039/50

1100011033, and by the European Union ERDF "A way of making Europe." C. Fiandrino is a Ramón y Cajal awardee (RYC2022-036375-I), funded by MCIU/AEI/10.13039/501100 011033 and the ESF+.

References

[1] P. Agrawal and M. Vutukuru. 2016. Trace Based Application Layer Modeling in ns-3. In *Proceedings of IEEE NCC.*

[2] G. Attanasio, C. Fiandrino, M. Fiore, J. Widmer, and others. 2022. In-depth study of RNTI management in mobile networks: Allocation strategies and implications on data trace analysis. *Computer Networks* 219 (2022), 109428.

[3] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, and others. 2015. A multi-source dataset of urban life in the city of Milan and the Province of Trentino. *Scientific data* (2015).

[4] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia. 2021. Intelligence and Learning in O-RAN for Data-driven NextG Cellular Networks. *IEEE Communications Magazine* 59, 10 (October 2021).

[5] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia. 2022. Open-RAN Gym: AI/ML Development, Data Collection, and Testing for O-RAN on PAWR Platforms. *Computer Networks* (November 2022).

[6] J. Breen, A. Buffmire, J. Duerig, K. Dutt, and others. 2021. POWDER: Platform for Open Wireless Data-driven Experimental Research. *Computer Networks* 197 (October 2021), 1–18.

[7] N. Bui and J. Widmer. 2018. Data-Driven Evaluation of Anticipatory Networking in LTE Networks. *IEEE Transactions on Mobile Computing* 17, 10 (2018), 2252–2265.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] P. Di Francesco, F. Malandrino, and L. A. DaSilva. 2018. Assembling and Using a Cellular Dataset for Mobile Network Analysis and Planning. *IEEE Transactions on Big Data* 4, 4 (2018), 614–620.

[10] R. Falkenberg and C. Wietfeld. 2019. FALCON: An accurate real-time monitor for client-based mobile network data analytics. In *Proceedings of IEEE GLOBECOM.*

[11] P. F. Fernández Pérez, C. Fiandrino, and J. Widmer. 2023. Characterizing and Modeling Mobile Networks User Traffic at Millisecond Level. In *Proceedings of ACM WiNTECH.*

[12] C. Fiandrino, G. Attanasio, M. Fiore, and J. Widmer. 2021. Traffic-Driven Sounding Reference Signal ResourceAllocation in (Beyond) 5G Networks. In *Proceedings of IEEE SECON.*

[13] C. Fiandrino, C. Zhang, P. Patras, A. Banchs, and J. Widmer. 2020. A Machine Learning-based Framework for Optimizing the Operation of Future Networks. *IEEE Communications Magazine* 58, 6 (2020).

[14] G. Gemmi, M. Elkael, M. Polese, L. Maccari, and others. 2023. Joint Routing and Energy Optimization for Integrated Access and Backhaul with Open RAN. In *Proceedings of IEEE GLOBECOM.*

[15] T. D. Hoang, C. Park, M. Son, T. Oh, and others. 2023. LTESniffer: An Open-source LTE Downlink/Uplink Eavesdropper. In *Proceedings of ACM Conference on Security and Privacy in Wireless and Mobile Networks.*

[16] W.-H. Hsu, Q. Li, X.-H. Han, and C.-W. Huang. 2017. A Hybrid IoT Traffic Generator for Mobile Network Performance Assessment. In *Proceedings of IEEE IWCMC.*

[17] S. Hui, H. Wang, T. Li, X. Yang, and others. 2023. Large-scale Urban Cellular Traffic Generation via Knowledge-enhanced GANs with Multi-periodic Patterns. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining.*

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[19] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, and others. 2020. The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. *International Journal of Computer Vision* 128 (March 2020), 1956–1981.

[20] M. Lecci, A. Zanella, and M. Zorzi. 2021. An ns-3 Implementation of a Bursty Traffic Framework for Virtual Reality Sources. In *Proceedings of ACM Workshop on ns-3.*

[21] T. Li, S. Hui, S. Zhang, H. Wang, and others. 2024. Mobile User Traffic Generation via Multi-Scale Hierarchical GAN. *ACM Transactions on Knowledge Discovery from Data* (May 2024).

[22] N. Ludant, P. Robyns, and G. Noubir. 2023. From 5G Sniffing to Harvesting Leakages of Privacy-Preserving Messengers. In *Proceedings of IEEE Symposium on Security and Privacy.*

[23] O. E. Martínez-Durive, S. Mishra, C. Ziemlicki, S. Rubrichi, and others. 2023. The NetMob23 Dataset: A High-resolution Multi-region Service-level Mobile Data Traffic Cartography. arXiv:2305.06933 [cs.NI]

[24] F. J. Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.

[25] J. Meng, J. Huang, Y. C. Hu, Y. Koral, and others. 2023. Modeling and Generating Control-plane Traffic for Cellular Networks. In *Proceedings of ACM Internet Measurement Conference.*

[26] NGMN Alliance. 2019. Verticals URLLC Use Cases and Requirements. https://tinyurl.com/ynzphkdf

[27] A. Panicker, O. Ozdemir, M. L. Sichitiu, I. Guvenc, and others. 2021. AERPAW Emulation Overview and Preliminary Performance Evaluation. *Computer Networks* 194 (July 2021), 1–11.

[28] Platforms for Advanced Wireless Research (PAWR). Accessed 2024. https://www.advancedwireless.org.

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

[30] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan. 2020. Beyond Throughput, the next Generation: A 5G Dataset with Channel and Context Metrics. In *Proceedings of ACM MMSys.*

[31] D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, and others. 2020. Challenge: COSMOS: A City-Scale Programmable Testbed for Experimentation with Advanced Wireless. In *Proceedings of ACM MobiCom.*

[32] U.S. Naval Research Laboratory. Accessed 2024. Documentation for the MGEN Traffic Emulator. https://tinyurl.com/24w6m7pr

[33] D. Villa, I. Khan, F. Kaltenberger, N. Hedberg, and others. 2024. X5G: An Open, Programmable, Multi-vendor, End-to-end, Private 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface. *arXiv:2406.15935 [cs.NI]* (June 2024), 1–15.

[34] D. Villa, M. Tehrani-Moayyed, C. P. Robinson, L. Bonati, and others. 2024. Colosseum as a Digital Twin: Bridging Real-World Experimentation and Wireless Network Emulation. *IEEE Transactions on Mobile Computing* (January 2024), 1–17.

[35] Y. Xie, F. Yi, and K. Jamieson. 2020. PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements. In *Proc. of ACM SIGCOMM.* 451–464.

[36] H. Zhang, Y. Guan, A. Kamal, D. Qiao, and others. 2021. ARA: A Wireless Living Lab Vision for Smart and Connected Rural Communities. In *Proceedings of ACM WiNTECH.* New Orleans, LA, USA.

[37] M. Zhang, H. Fu, Y. Li, and S. Chen. 2019. Understanding Urban Dynamics From Massive Mobile Traffic Data. *IEEE Transactions on Big Data* 5, 2 (2019), 266–278.

[38] S. Zhang, T. Li, S. Hui, G. Li, and others. 2023. Deep Transfer Learning for City-scale Cellular Traffic Generation through Urban Knowledge Graph. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining.*