Lightweight Trace-Driven Burst Traffic Generation for 5G Network Digital Twins

Ghinwa Ismail, Samir Si-Mohammed, Fabrice Theoleyre

Lightweight Trace-Driven Burst Traffic Generation for 5G Network Digital Twins

Ghinwa Ismail
ICube lab
CNRS / Univ. Strasbourg, France
Email: gismail@unistra.fr

Samir Si-Mohammed
CRAN lab
CNRS / Univ. of Lorraine, France
Email: samir.si-mohammed@univ-lorraine.fr

Fabrice Theoleyre
ICube lab
CNRS / Univ. Strasbourg, France
Email: fabrice.theoleyre@cnrs.fr

***Abstract*—The Digital Twin (DT) architectures for 5G often lack support for coexisting traffic classes with distinct Quality of Service (QoS) constraints, conversational audio (low latency/jitter), adaptive video (high throughput with buffering), and interactive web transactions (latency sensitive but bursty) constraints. In this setting, application and traffic modeling becomes a core component of the DT, since it controls how realistic the offered load is. We develop a lightweight traffic model that mimics flow characteristics in a cellular network, and specifically how periods of activity and inactivity alternate over time. The traffic generator consists of a discrete burst-level traffic model that combines a Markov-based model with a simple temporal-dynamics model to ensure statistical robustness. The model is trained offline from packet traces and is intended to be used in a DT of the radio access and core network, so that different loading scenarios can be tested without running full application stacks on large populations of UEs. We demonstrate how such models can be constructed for multiple application types and combined into a single framework.**

***Index Terms*—Digital Twins; 5G; Traffic models; Multimodal**

I. Introduction

5G networks are characterized by a high degree of traffic heterogeneity, supporting ultra-reliable low-latency communications (URLLC), and ultra-massive machine-type communications (mMTC) for highly energy-constrained devices, as well as enhanced Mobile Broadband (eMBB) services with high bandwidth demands [1]. To address this complexity, 5G networks extensively rely on virtualization technologies to isolate heterogeneous services, along with artificial intelligence techniques to enable continuous network optimization.

As defined by Grieves and Vickers [2], a Digital Twin (DT) is a virtual replica of a physical system that continuously mirrors its state, enabling monitoring, prediction, and control. DTs have been increasingly used in 5G research to emulate, monitor, and optimize network behavior under realistic yet controllable conditions [3].

However, current DTs frameworks often lack support for coexisting traffic classes with distinct Quality of Serrvice (QoS) constraints, conversational audio (low latency/jitter), adaptive video (high throughput with buffering), and interactive web transactions (latency-sensitive but bursty) [4]. A multimodal traffic model is required since each application exhibits specific characteristics: for instance, while video streaming is bursty by nature [5], industrial applications may generate close to periodic traffic.

This work was partly funded by the French government under the France 2030 ANR program PEPR Networks of the Future (ref. ANR-22-PEFT-00010)

Emulation seems attractive to simulate the traffic generated by a collection of users/terminals [6]. Unfortunately, emulation is often computationally expensive since the DT has to execute a virtualized stack of protocols. Even when the stack is partially emulated, the challenge is to select what parts to emulate, to maintain both scalability and accuracy.

To address the wireless network complexity, DT architectures typically separate *when* traffic is generated from *how* it is realized at the packet level. The importance of traffic statistics for QoS has long been illustrated by simulation studies of bursty Variable-Bit-Rate (VBR) traffic: when multiple ON/OFF flows with long-range dependent burst patterns share a router, the resulting queueing delay, packet loss, and goodput are strongly affected by the burstiness parameters of the offered load [7].

These gaps motivate scalable and flexible traffic modeling that can plug into DT workflows for experimentation and testing. Our contribution is to design a model which is:

1) Self-configurable: a traffic capture is sufficient to train our model.
2) Multimodal: several applications can be simulated, and the resulting traffic can be used in a 5G DT.
3) Lightweight: we maintain a low computational complexity to improve the scalability.
4) Tunable: we can increase the number of flows or the traffic per flow to enable *what-if* scenarios.
5) Robust across statistics: it captures both marginal burst-feature distributions and the temporal persistence/transition structure within flows.

II. Related Work

To design realistic Digital Twins, the underlying traffic models must accurately capture the statistical properties observed in real cellular networks. [8]. We therefore first review traffic characterization studies before discussing the families of models proposed to emulate them.

A. Traffic characteristics

Measurements at the operator level show that traffic volumes are highly skewed, with a few dominant applications and devices generating the majority of data [9]. At finer granularity,

Radio Access Network (RAN) studies reveal self-similarity and burstiness at millisecond timescales, largely driven by user behavior [10]. Such findings provide the statistical priors for model design: heavy-tailed distributions guide flow-level envelopes, while temporal correlations and burstiness inform packet-level synthesis.

At the application level, studies of Voice over IP (VoIP) over mobile networks show that jointly modeling traffic and QoS/Quality of Experience (QoE) descriptors (*e.g.,* bandwidth, delay, jitter, Mean Opinion Score (MOS)) is essential to understand and control perceived quality [11].

B. Synthetic traffic models

Early synthesis focused on *synthetic-only* generators, independent of the live network. Hidden Markov Models (HMMs) use a few latent states to model packet lengths and inter-arrival times. They offer interpretable structure and a light footprint, but may struggle to reproduce long-range effects (slowly decaying correlations and heavy-tailed activity patterns) unless the state space is made large [12].

Flow-record Generative Adversarial Network (GAN) have been proposed to learn categorical fields (IPs, ports, protocol) that pass domain checks [13]. These tools show realism for narrow domains (*e.g.*, DNS), but length-sensitive semantics and checksums remain fragile without post-rules [14].

To account for temporal distributions, sequence models tokenize packet size and timing, enabling Transformer decoders to capture packet sequences and transfer learned representations to classification tasks, while remaining agnostic to protocol phases [15]. More recently, protocol-constrained generators synthesize packet capture headers and repair fields post-generation, but they do not learn timestamps [16]. Synthetic-only generators are lightweight and capture packet structures, but they miss realistic timing and long-range dynamics.

C. Data-oriented models

A second class of methods constructs models directly from real traffic traces or testbed data to preserve both temporal dynamics and structural patterns. For example, Swing [17] recreates users, sessions, and connections, replaying them over inferred network paths while estimating delay, capacity, and loss from passive measurements. Its wavelet-based analysis shows that traffic burstiness across timescales is accurately reproduced only when both traffic structure and network path constraints are jointly preserved.

Recent work has proposed low-complexity generative models that operate at coarser time scales and provide load *envelopes* for the emulated network. For instance, knowledge-enhanced GAN synthesize hourly base-station loads by combining daily and weekly cycles with residual components, and by conditioning on urban knowledge graphs that encode, for example, land use or points of interest [18]. These models significantly improve over Recurrent Neural Network (RNN) and Temporal Convolutional Network (TCN) baselines for city-scale traffic forecasting at the hourly cadence, and are well suited to dimension and drive large-scale DT scenarios.

D. Emulation and DT platforms

In many DT architectures for 5G, the radio access and core network are emulated or simulated on a testbed, while traffic is injected by configurable software generators that emulate large populations of UEs. On platforms such as Colosseum [3], the Multi-Generator (MGEN) tool is commonly used as a flexible packet generator. Traffic traces or high-level specifications can be mapped into MGEN scripts and replayed with controlled start times, rates, and flow patterns, providing a repeatable traffic source for DT experiments [19].

Since the DT aims to represent many base stations, network slices, and heterogeneous applications, the overall architecture becomes complex: it is not feasible to explicitly simulate every user and application stack in detail [20]. In this setting, application and traffic modeling becomes a core component of the DT. It determines how realistic the offered load is, how different "what-if" scenarios are exercised, and how easily the DT can be scaled while keeping the traffic generation models computationally lightweight [21].

Within each base station, packet- and flow-level traffic characteristics can then be refined by more detailed models, which take the coarse-grained load per application or service as an input and treat them as realistic flows and packet sequences that interact with the emulated RAN and core [22].

As highlighted in Table I, existing methods rarely combine burst-level realism with learned temporal dynamics in a form that is directly composable for digital-twin experiments, which motivates our proposed pipeline.

III. Objectives and Background

To the best of our knowledge, this is the first trace-to-model pipeline that jointly captures **when** packets are transmitted, **how** they form bursts, and **how** activity/inactivity alternates over time, using a fine-grained, statistically robust burst representation and a learned temporal dynamics model

The resulting traffic model is intended to be used in a DT of the radio access and core network. Different loading scenarios can be tested without having to run full application stacks on large populations of UEs, improving the scalability.

In this context, our design focuses on three primary objectives: i) **realism**, reproducing the main statistical characteristics of the original flow (*e.g.*, ON/OFF periods, packet length), ii) **reproducibility**, and iii) **scalability** to model infrastructures supporting a large collection of flows.

A. Assumptions and scope

We do not rely on application-specific messages or protocol parsing. Instead, we only use features that are commonly extracted from packet traces (*i.e.,* timestamps, sizes, directions, and flow identifiers). Thus, models for different types of traffic (such as video, web, and voice) can be learned from their traces and then composed within the same generator.

Our approach is trace-based and uses only packet timestamps, sizes, and L3/L4 headers; payloads are not used. Flows are reconstructed from 5-tuples when ports are available. We

TABLE I
COMPARISON OF TRACE-DRIVEN TRAFFIC MODELING APPROACHES (CAPABILITIES RELEVANT TO SCALABLE TRAFFIC INJECTION).

Work	Input	Temporal explicit		Temporal model	Gen vs Rep	Scaling / control knob
		Level	ON/OFF			
Bonati [19]	Traces / KPI	Testbed workload	-	Trace twinning	Replay	Workload shaping + replay configuration
Koktas [21]	Packet traces	Packet	Implicit	HMM sequencing	Generative	Control via state priors/transitions and emission params
NetDiffusion [16]	PCAP (pkt headers)	Packet	-	Diffusion model	Generative	Control via conditioning, sampling cost higher
Flow-GAN [13]	Flow records	Flow	-	i.i.d. flows	Generative	Control via feature conditioning, no within-flow sequencing
Swing [17]	Traces (sessions)	Pkt/session	Implicit	Closed-loop replay	Replay	Replay inferred structure, limited parametric scaling
Our work	PCAP (per-app)	Burst	Explicit	Burst Markov	Generative	Scale number of flows, controlled resampling

train models per application and per direction (uplink/downlink) independently. We generate burst-level traffic; packet-level reconstruction within bursts is left for future work.

B. Flow-level modeling

We focus only on data-plane packets, i.e., packets that primarily carry application payload. Control-plane packets are typically modeled separately by the DT to be technology- and protocol-agnostic. Fig. 1 illustrates the flow-to-burst decomposition and the corresponding ON/OFF periods used throughout the paper. Our traffic model is decomposed into:

Flows: A flow is a unidirectional sequence of packets observed at the capture point that share the same five-tuple (Sou/Des IP addresses / port numbers, transport) [23].

Directions: Packets (and flows) are associated with a direction (uplink or downlink) according to whether the local endpoint appears as source or destination in the IP header.

Bursts: Within a flow, packets belong to same burst if the time gap between them stays below a set threshold ($theta$).

ON and OFF periods: For burst k, the ON duration is the time between its first and last packet. The OFF duration is the idle time immediately preceding the burst.

IV. EXTRACTION OF KEY FEATURES FROM THE TRACE

We first convert a packet capture into modeling-ready traffic objects that can be emulated by a cellular DT. We reconstruct unidirectional flows, separate data-plane packets from signaling, segment each flow into bursts (ON periods) and the preceding idle gaps (OFF periods), and extract per-burst size and timing features used by the model.

A. Signaling vs. data separation

Signaling packets correspond to connection establishment, teardown, maintenance and name resolution, whereas data packets correspond to the bulk of the payload transfer. In practice, we attach to each packet a categorical label (*signaling* / *data*) using lightweight rules based on header fields and simple meta-information (*e.g.,* transport flags, well-known ports, short acknowledgement or keep-alive messages, DNS or TLS handshake exchanges). We use a heuristic for this classification, with the possibility of a small number of misclassifications. Importantly, we train the burst-level model on data-labeled packets only.

B. Burst extraction (ON/OFF segmentation)

We group packets into bursts to expose each flows active (ON) and idle (OFF) phases, interpreting bursts as transmission periods and the intervening gaps as silent intervals, in line with classic ON/OFF traffic models [7].

Consider a flow $f \in \mathcal{F}$ with N_f packets. Once the packets of f are sorted by timestamp, they are indexed sequentially ($i = 1, \ldots, N_f$). Let $t_{f,i}$ denote the timestamp of packet i in flow f (in Sec) and $l_{f,i}$ its length in bytes. We compute inter-packet gaps as

$$\Delta t_{f,i} = t_{f,i+1} - t_{f,i}, \qquad i = 1, \ldots, N_f - 1. \tag{1}$$

Given a threshold $\theta > 0$, packets i and $i+1$ belong to the same burst if

$$\Delta t_{f,i} \leq \theta. \tag{2}$$

A *burst* is a maximal group of contiguous packets in a flow for which all consecutive inter-packet gaps are at most θ.

Rather than fixing θ manually, we set it automatically from the empirical inter-packet gap distribution. We aggregate all strictly positive gaps $\{\Delta t_{f,i}\}$ and apply robust clipping:

$$\Delta t_{f,i} \leftarrow \mathrm{clip}(\Delta t_{f,i},\, 0.02,\, 1.5) \ \text{ s.}$$

We then evaluate candidate thresholds at multiple percentiles (e.g., the 90th, 95th, and 97th) and select the one exhibiting the highest local curvature change in the log-transformed distribution, falling back to the 97th percentile if no clear elbow is detected.

Given a threshold $\theta > 0$, we define the *burst index function* $b : \{1, \ldots, N\} \to \{1, \ldots, K_f\}$ by

$$b(1) = 1, \qquad b(i{+}1) = b(i) + \mathbb{1}(\Delta t_i > \theta), \quad i = 1, \ldots, N{-}1, \tag{3}$$

where K_f is the number of bursts in flow f, and $\mathbb{1}(c)$ is the indicator function, equal to 1 if condition c holds, and otherwise equal to 0. For readability, when considering a single flow f we omit the subscript f.

Burst k is then the set of packet indices

$$\mathcal{I}_k = \{\, i \in \{1, \ldots, N\} : b(i) = k \,\}, \tag{4}$$

with first and last packet indices $n_k = \min(\mathcal{I}_k)$ and $m_k = \max(\mathcal{I}_k)$.

We also define the burst start and end times as $s_k = t_{n_k}$ and $e_k = t_{m_k}$.

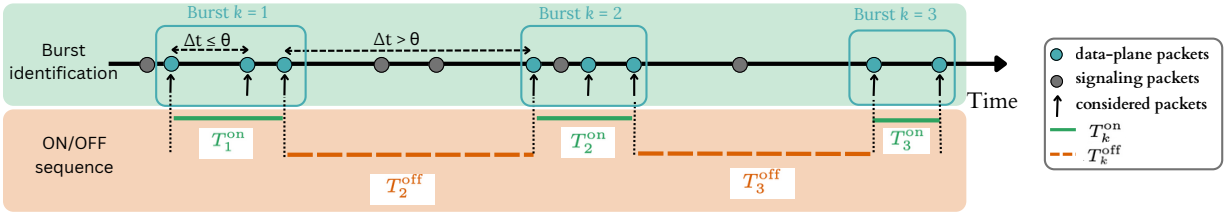


Fig. 1. Example of a flow at the capture point. Packets are grouped into bursts based on inter-packet gaps, yielding an alternating sequence of ON and OFF periods. We focus on the data packets (blue) only.

The ON duration of burst k is

$$T_k^{\text{on}} = e_k - s_k = t_{m_k} - t_{n_k}, \tag{5}$$

and the OFF duration associated with burst k is the idle time immediately preceding it. For $k \geq 2$,

$$T_k^{\text{off}} = s_k - e_{k-1} = t_{n_k} - t_{m_{k-1}}. \tag{6}$$

The first burst of each flow has no preceding OFF period; we therefore leave T_1^{off} undefined (stored as `NaN` in our data).

Each burst is then mapped to a small set of interpretable features that summarize its local behavior. These raw quantities are typically heavy-tailed which means they are often strongly skewed with occasional very large values. Consequently, we work in log space to stabilize the variance and reduce skew.

Specifically, I_k^{first} flags the first burst of a flow ($k = 1$), for which T_k^{off} is undefined, and $I_k^{\text{zero-on}}$ flags bursts with effectively zero ON duration (typically single-packet bursts).

Finally, each burst is represented by the feature vector:

$$\begin{aligned}\mathbf{x}_k = (&\log T_k^{\text{on}},\ \log T_k^{\text{off}},\ \log N_k,\ \log B_k,\ \log r_k^{(N)},\\ &\log r_k^{(B)},\ \log s_k,\ \log \rho_k,\ I_k^{\text{first}},\ I_k^{\text{zero-on}}).\end{aligned} \tag{7}$$

Table II summarizes the main notations used in the burst-level model and generator.

Mixing single-packet bursts with multi-packet bursts introduces a large mass near $T_k^{\text{on}} = 0$, which would artificially shift the fitted distribution of $\log T_k^{\text{on}}$. Thus, we also scale-normalized descriptors of burst intensity and pacing:

$$\begin{aligned} r_k^{(N)} &= \frac{N_k}{\max(T_k^{\text{on}}, \epsilon_{\text{on}})}, \quad r_k^{(B)} = \frac{B_k}{\max(T_k^{\text{on}}, \epsilon_{\text{on}})},\\ s_k &= \frac{B_k}{\max(N_k, 1)}, \qquad \rho_k = \frac{T_k^{\text{off}}}{\max(T_k^{\text{on}}, \epsilon_{\text{on}})}. \end{aligned} \tag{8}$$

where ϵ_{on} the minimum ON-duration clamp before log transform.

To avoid $\log(0)$, we compute ϵ_{on} from the observed ON-duration samples after preprocessing (signaling removal and burst extraction). Let $x_{\min}$ be the minimum strictly positive value of T_k^{on} in these samples. We set $\epsilon_{\text{on}} = \max(\epsilon_{\min}, \min(x_{\min}/10, \epsilon_0))$ with $\epsilon_0 = 10^{-4}$ and $\epsilon_{\min} = 10^{-6}$.

TABLE II
MAIN NOTATIONS USED IN THE BURST-LEVEL MODEL AND GENERATOR.

Symbol	Meaning
θ	Inter-packet gap threshold (s) used for burst segmentation.
T_k^{on}	ON duration of burst k.
T_k^{off}	OFF duration before burst k ($k \geq 2$; undefined for $k = 1$).
$N_k,\ B_k$	Packets per burst and bytes per burst for burst k.
I_k^{first}	First-burst indicator ($k = 1$), where T_k^{off} is undefined.
$I_k^{\text{zero-on}}$	Zero-ON indicator (typically single-packet bursts).
m_{cs}	HDBSCAN `min_cluster_size`.
m_{s}	HDBSCAN `min_samples`.
$\boldsymbol{\pi}$	Initial distribution over types.
A	Markov transition matrix: $A_{ij} = \Pr(z_{k+1} = j \mid z_k = i)$.
L_f	Number of bursts in flow f (sequence length).
σ_{jit}	Log-space jitter standard deviation in type-conditional bootstrap sampling.
ϵ_{on}	Minimum ON-duration clamp before log transform.
$r_k^{(N)}$	Packet rate.
$r_k^{(B)}$	Byte rate.
s_k	Bytes-per-packet ratio.
ρ_k	OFF/ON ratio.

V. BURST-LEVEL TRAFFIC MODELING

After extracting bursts and converting them into feature vectors, we create a compact, state-based statistical model for traffic generation. Instead of working in the raw continuous feature space, we first cluster similar bursts into a limited set of discrete burst types. Then we model how these burst types transition over time within each flow. The result is a low-dimensional representation of burst behavior together with a simple temporal-dynamics model that becomes the core of the traffic generator. For instance, a burst with continuous features as defined in eq. 7 is assigned a discrete type $z_k \in \{1, \ldots, K\}$ (*i.e.,* a 'short–small' vs. a 'long–large' burst), and we model the sequence $\{z_k\}$ within a flow via a Markov chain.

A. Burst characterization via clustering

Burst characterization is an unsupervised clustering problem in the burst-feature space. For a given application and direction, let n denote the number of bursts, and each burst k is represented with a feature vector $\mathbf{x}_k$ (eq. 7). After standardization, we obtain $\tilde{\mathbf{x}}_k$ and we stack them row-wise into $\tilde{\mathbf{X}} \in \mathbb{R}^{n \times 10}$. Clustering $\tilde{\mathbf{X}}$ yields a finite set of discrete *burst*

characterizations (cluster labels), where each includes bursts with similar timing and size characteristics. These labels define the discrete state space of the temporal model. Each flow is then represented as a sequence of states $z_k \in \{1, \ldots, K\}$.

For each characterization label, we construct a simple label-conditional generator in log space by resampling training bursts assigned to that label and adding a Gaussian jitter. Typical characterizations include short-idle small bursts, long idle intervals followed by large bursts, and single-packet events with near-zero ON duration.

The burst-feature distribution is heavy-tailed, multi-modal, and contains clusters with very different densities. Because methods that need a preset number of roughly spherical clusters (e.g., k-means) or only Gaussian shapes (e.g., GMMs) are fragile and demand extensive tuning, we use Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). This density-based hierarchical clustering algorithm (i) determines the number of clusters automatically, (ii) handles clusters of varying density and shape, and (iii) marks low-density points as noise instead of forcing them into unsuitable clusters.

In practice, for each application/direction pair, we first standardize the burst features using a `StandardScaler` fitted on training bursts only. We then fit an HDBSCAN model on the standardized training bursts, with `prediction_data=True` to enable out-of-sample labeling. HDBSCAN has two main hyperparameters (minimum cluster size m_{cs} and minimum samples m_{s}). In our implementation, these are selected automatically on the *training* data via a grid search that maximizes the Calinski–Harabasz index computed on non-noise points, while enforcing simple sanity constraints (e.g., a bounded number of clusters and a limited noise fraction) to keep clusters sufficiently populated.

After fitting HDBSCAN on the training bursts (with `prediction_data=True`), training bursts retain their labels from the fitted clustering. We then assign labels to the test bursts using HDBSCAN's out-of-sample routine `approximate_predict`, which predicts membership in one of the learned clusters without refitting and returns ($z_k = -1$) for bursts that are deemed noise (with a membership strength score).

Bursts labeled as noise are excluded from subsequent modeling. The remaining K non-noise labels define the burst types used by the Markov model and as the conditioning variable for state-specific burst generators (optionally reindexed to $\{1, \ldots, K\}$ for convenience). HDBSCAN does not provide an exact parametric out-of-sample predictor, `approximate_predict` assigns new points based on the fitted condensed tree and mutual reachability structure.

Fig. 2 illustrates a t-SNE projection of the clustered burst features for Filimo application. The visualization confirms the multi-modal nature of the traffic, where HDBSCAN successfully isolates distinct densities and non-spherical shapessuch as the prominent "curved arm" cluster which represent specific application-layer behaviors that would be conflated by traditional centroid-based methods.

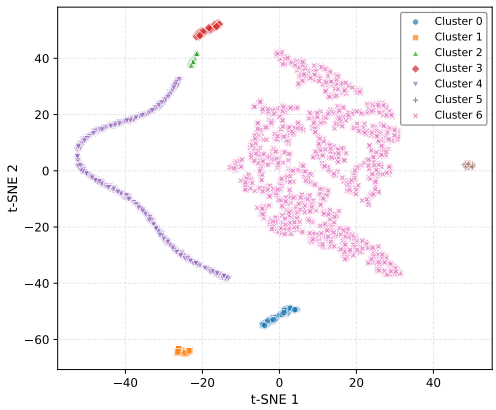


Fig. 2. t-SNE projection of the burst feature space. Colors represent the K discrete burst types (0–6) identified by HDBSCAN, illustrating the clear separation of traffic regimes used for the temporal Markov model.

Clustering quality metrics confirm good separation: DBCV scores range from 0.3–0.7, mean silhouette scores exceed 0.5, and noise fractions remain below 5% across all applications.

B. Temporal dynamics of burst sequencing

After labeling the bursts, we need to capture how burst types evolve over time within a flow. We model the ordering of burst types within each flow using a simple discrete-time stochastic model. A standard tool for this purpose is a first-order Markov chain, where the probability of the next burst type depends only on the current one.

We adopt this first-order assumption as a deliberate trade-off between temporal realism and model compactness, since our goal is to preserve dominant local burst transitions while keeping the generator lightweight and scalable.

In our setting, the role of the latent process is played by the sequence of burst-type labels, and the observations are the burst feature vectors. Importantly, we do not infer the discrete labels jointly with the Markov Model, instead, burst-type labels are obtained beforehand by clustering in the feature space, and we subsequently fit only the temporal dynamics.

Concretely, after clustering, each burst is assigned a discrete burst-type label $z_k \in \{1, \ldots, K\}$, which serves as a compact representation of its local behavior. For a given application and direction, ordering the bursts by start time yields, for each flow f, a burst-type sequence $z_{f,1}, \ldots, z_{f,L_f}$, where L_f is the number of bursts in flow f. We model this sequence as a homogeneous first-order Markov chain, *i.e.,* the next burst type depends only on the current one.

In Markov Model terminology, $\{z_{f,k}\}$ corresponds to the discrete-time chain, and the burst feature vectors correspond to emissions drawn from type-conditional distributions (estimated empirically in the next section). Since the labels $z_{f,k}$ are observed from clustering, estimating the Markov parameters reduces to counting type occurrences and transitions.

Let $\mathcal{F}$ denote the set of training flows. For each state $i \in \{1, \ldots, K\}$ we define

$$N_i^{\text{start}} = |\{\, f \in \mathcal{F} \mid z_{f,1} = i \,\}|, \tag{9}$$

the number of training flows whose first burst type is i. The total number of training flows is therefore

$$|\mathcal{F}| = \sum_{i=1}^{K} N_i^{\text{start}}. \tag{10}$$

Similarly, for each pair of states $i, j \in \{1, \ldots, K\}$ we define

$$N_{ij}^{\text{trans}} = |\{\, (f,k) \mid f \in \mathcal{F},\ 1 \le k < L_f, \\ z_{f,k} = i,\ z_{f,k+1} = j \,\}|, \tag{11}$$

i.e., the total number of observed transitions from state i to state j across all training flows.

The maximum-likelihood estimates of the initial distribution $\boldsymbol{\pi}$ and the transition matrix A are then

$$\hat{\pi}_i = \frac{N_i^{\text{start}}}{|\mathcal{F}|}, \qquad i = 1, \ldots, K. \tag{12}$$

$$\hat{A}_{ij} = \frac{N_{ij}^{\text{trans}}}{\sum_{j'=1}^{K} N_{ij'}^{\text{trans}}}, \qquad i, j = 1, \ldots, K. \tag{13}$$

In implementation, we apply additive smoothing with $\alpha = 10^{-3}$ to the start and transition counts before normalization to avoid zero-probability transitions; rows with no outgoing transitions are replaced by $\hat{\pi}$.

By construction, each row of $\hat{A}$ sums to one. For states with no observed outgoing transitions (*i.e.,* $\sum_j N_{ij}^{\text{trans}} = 0$), we replace the corresponding row by a default distribution, chosen in practice as the empirical initial distribution $\hat{\boldsymbol{\pi}}$, to avoid degenerate behavior.

VI. Traffic emulation

At this stage of the pipeline, the burst-level model has been learned: each burst is assigned a discrete *type* label and the temporal evolution of types within a flow is captured by a Markov chain. We now turn these learned components into a lightweight generator that can emulate traffic.

A. Type-conditional burst sampling

Let $z_k \in \{1, \ldots, K\}$ denote the burst-type label assigned by clustering. The first component of the emulator is a type-conditional burst generator: for each type $c \in \{1, \ldots, K\}$, we treat the training bursts assigned to c as samples from the conditional distribution of burst features given $z_k = c$. Concretely, we collect the log-transformed continuous burst features into a matrix:

$$\mathbf{X}_{\log}^{(c)} \in \mathbb{R}^{n_c \times 4},$$

whose rows are $(\log T_k^{\text{on}}, \log T_k^{\text{off}}, \log B_k, \log N_k)$ for the n_c bursts in type c. We also store binary masks indicating special cases (*i.e.,* undefined T_k^{off} for first-burst events, and true zero-ON bursts). Working in log space also makes the Gaussian jitter step equivalent to a small multiplicative perturbation in the original domain, avoiding unrealistic additive offsets for large vs. small bursts.

To generate a synthetic burst of type c, we sample one training row $\mathbf{x} \in \mathbb{R}^4$ from $\mathbf{X}_{\log}^{(c)}$ (uniformly at random) and add a small Gaussian perturbation $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\, \sigma_{\text{jit}}^2 \boldsymbol{\Sigma}_c)$, $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$, where $\boldsymbol{\Sigma}_c \in \mathbb{R}^{4\times 4}$ is the empirical covariance of $\mathbf{X}_{\log}^{(c)}$ (to preserve correlations between the core log-features). If $\boldsymbol{\Sigma}_c$ is ill-conditioned, we fall back to independent Gaussian jitter. This acts as a simple kernel smoothing of the empirical distribution while avoiding exact replay of training points.

We map $\tilde{\mathbf{x}}$ back to the original scale by exponentiating its components to obtain candidate values for T_k^{on}, T_k^{off}, B_k, and N_k. In implementation, we enforce feasibility constraints: true zero-ON bursts are assigned $T_k^{\text{on}} = 0$; when a valid OFF duration is required but the selected training burst has undefined T_k^{off} (first-burst event), we resample T_k^{off} from the empirical distribution of valid OFF values within the same type c (or fall back to a small positive value if needed). Finally, non-zero ON durations are clamped to be at least $\epsilon_{\text{on}} > 0$, byte counts are clamped to be at least one, and packet counts are rounded to an integer with $N_k \ge 1$.

This procedure outputs a synthetic burst $(T_k^{\text{on}}, T_k^{\text{off}}, N_k, B_k)$ conditioned on $z_k = c$.

B. Chaining burst types into synthetic flows

The type-conditional generator allows to sample individual bursts given a burst type. To emulate complete flows, we must capture how types follow one another and how many bursts a typical flow contains. We combine the Markov model of Section V-B with the empirical distribution of flow lengths.

For each application and direction, let $\mathcal{F}_{\text{train}}$ denote the set of training flows and $L_f \in \mathbb{N}$ the number of bursts in flow f. The Markov model $(\hat{\boldsymbol{\pi}}, \hat{A})$ is estimated on the corresponding type sequences $\{z_{f,1}, \ldots, z_{f,L_f}\}$, and the empirical distribution of flow lengths is given by the multiset $\{L_f : f \in \mathcal{F}_{\text{train}}\}$.

Synthetic traffic is generated flow by flow. For each synthetic flow, we: (i) sample a flow length L from $\{L_f\}$; (ii) generate a type sequence $Z_1, \ldots, Z_L$ from the Markov chain by drawing $Z_1 \sim \hat{\boldsymbol{\pi}}$ and $Z_{k+1} \mid Z_k = i \sim \hat{A}_{i,\cdot}$; (iii) for each position $k = 1, \ldots, L$, call the type-conditional burst generator with label Z_k, with `need_valid_off` set to `False` for $k = 1$ and `True` for $k \ge 2$.

The result is a collection of synthetic flows, each represented as an ordered sequence of bursts. Absolute burst start times along a flow can be reconstructed by cumulatively summing ON and OFF durations from an arbitrary origin, so that the synthetic bursts can be mapped to packet-level timestamps when driving the DT.

C. Discussion

This traffic generator is only a first step in designing a scalable DT for 5G networks. It provides the traffic models to inject *fake* packets into the DT. However, the *what-if scenarios*, the models to estimate end-to-end Key Performance Indicators (KPIs), etc. remain to be proposed to complete such DT. We let this integration to the future work.

TABLE III
APPLICATION-SPECIFIC PARAMETERS USED IN THE EVALUATION (DOWNLINK).

App	θ [s]	ϵ_{on}	σ_{jit}	HDBSCAN $(m_{\mathrm{cs}}, m_{\mathrm{s}})$
Aparat	0.020	2.4×10^{-6}	0.06	(30, 20)
Filimo	0.020	2.8×10^{-6}	0.04	(40, 25)
iGap	0.041	3.1×10^{-6}	0.06	(15, 15)
Telegram	0.090	2.1×10^{-5}	0.05	(30, 20)
YouTube	0.057	1.7×10^{-5}	0.05	(50, 25)

VII. VALIDATION AND EXPERIMENTAL RESULTS

We implement the complete framework: application segmentation, identification of bursts in the packet capture, training of the Markov Chain.

Then, the generator exploits the model to generate synthetic traces. All stochastic components of the generator (bootstrap resampling, jitter, and type sampling) are controlled by explicit random seeds, so that synthetic traces are reproducible given the learned parameters and the chosen configuration. Table III reports the application-specific parameters used in our evaluation. To ensure reproducibility, our implementation is made publicly available[1], including the scripts and the configuration used in this paper.

A. Dataset & methodology

We evaluate the proposed generator using the ITC-NetBlend-60 dataset (Scenario A) [24], which provides PCAP captures exported to CSV of Android application traffic collected at the end host. We consider five applications: Aparat, iGap, Filimo, Telegram and YouTube. For each application, we use the four available PCAP files (no exclusions). For brevity, we present results for the downlink direction only, even though both directions can be modeled in the same way.

After preprocessing and signaling removal, each application (4 PCAP files) comprises roughly 150–365 reconstructed flows and 1,832–6,087 bursts ($\approx$ 8–21 bursts/flow), totaling 47k–200k downlink packets. This highlights a sizeable and highly bursty dataset with substantial variability across applications.

We split flows into disjoint partitions: 70/30% for train/test, the split is performed at the flow level to avoid leakage. We use a fixed random seed (`seed=123`) to ensure reproducibility.

We evaluate accuracy on held-out test flows, for each application. We learn the burst-type clustering, type-conditional generators, and Markov transition model on the training flows only, and then generate synthetic traffic using the learned model. We generate a synthetic burst set with the same number of bursts as the held-out test set.

B. Distributional fidelity

We compare real held-out bursts to synthetic bursts using the two-sample Kolmogorov–Smirnov (K-S) statistic, which measures the maximum gap between the empirical CDFs: $D = \sup_x |F_{\mathrm{real}}(x) - F_{\mathrm{syn}}(x)|$. The statistic satisfies $D \in [0, 1]$ and smaller values indicate closer agreement. We use KS as the main effect-size indicator because it is non-parametric and sensitive to differences in both location and shape of the distributions [25]. Table IV summarizes KS values and quantile errors on held-out test flows.

[1]https://github.com/GhinwaISMAIL/multimodal-traffic-digital-twins

TABLE IV
K-S AND QUANTILE ERRORS ON HELD-OUT FLOWS (DOWNLINK).

App	Feat.	KS	$\epsilon_{0.50}$	$\epsilon_{0.90}$	$\epsilon_{0.99}$
Aparat	T^{on}	0.048	-0.159	-0.189	-0.426
	T^{off}	0.072	0.107	0.617	0.228
	B	0.099	-0.282	-0.243	-0.245
	N	0.058	-0.222	-0.240	-0.428
Filimo	T^{on}	0.089	0.599	0.195	-0.037
	T^{off}	0.075	-0.067	0.032	7.318
	B	0.186	0.317	0.090	-0.239
	N	0.094	0.333	0.009	-0.239
iGap	T^{on}	0.123	-0.208	-0.557	-0.248
	T^{off}	0.228	0.946	9.176	0.693
	B	0.121	0.245	-0.432	-0.172
	N	0.063	0.000	-0.430	-0.169
Telegram	T^{on}	0.075	0.833	-0.222	-0.170
	T^{off}	0.040	0.024	-0.144	0.302
	B	0.081	0.038	-0.617	-0.011
	N	0.075	0.000	-0.613	-0.010
YouTube	T^{on}	0.031	0.685	-0.089	0.096
	T^{off}	0.112	0.301	0.943	0.172
	B	0.196	-0.462	-0.199	0.544
	N	0.034	0.000	-0.184	0.574

NB: $\epsilon_q = (Q_q^{\mathrm{syn}} - Q_q^{\mathrm{real}})/Q_q^{\mathrm{real}}$.

We observe that Aparat yields the lowest KS values overall (best agreement), while Filimo and YouTube exhibit larger KS values for bytes (KS $= 0.186$ and 0.196, respectively), reflecting heavier and more heterogeneous tails. Telegram shows good agreement across all features, with KS values below 0.10. Overall, the generator reproduces well the central mass of the distributions. The remaining errors concentrate in the upper tail, in particular, the OFF duration for Filimo and iGap, which is consistent with rare long-idle events being harder to reproduce than the central mass.

We also compare quantiles at $q \in \{0.50, 0.90, 0.99\}$ to summarize differences at typical and tail-relevant operating points. We report the relative quantile error

$$\epsilon_q = \frac{Q_q^{\mathrm{syn}} - Q_q^{\mathrm{real}}}{Q_q^{\mathrm{real}}}, \tag{14}$$

where Q_q^{real} and Q_q^{syn} are the q-quantiles of the considered feature. For heavy-tailed features, extreme quantiles can exhibit high variance under finite samples, especially for OFF durations, where the longest idle periods are rare. As a result, relative errors at $q = 0.90$ and $q = 0.99$ may become large even when the bulk of the distribution is well reproduced. We therefore interpret $\epsilon_{0.90}$ and $\epsilon_{0.99}$ primarily as tail diagnostics rather than as hard optimization targets, and we analyze them

jointly with the KS statistics and the empirical CDFs. This choice is aligned with our design goal: a lightweight model that preserves burst/idle structure and temporal dynamics.

We assess if the generator preserves the marginal distributions of the four core burst features: ON duration T_k^{on}, OFF duration T_k^{off}, bytes per burst B_k, and packets per burst N_k.

Fig. 3 provides a visual assessment of distributional fidelity by comparing empirical CDFs of real and synthetic burst features. We select Aparat and Filimo as representative examples because they illustrate two regimes observed in our dataset.

Aparat corresponds to a well-modeled case with consistently low distributional errors, while Filimo represents a more challenging case with heavier tails and larger deviations. This pair therefore highlights both the strengths and the limitations of the proposed lightweight generator.

We observe that the generator closely tracks the CDFs of Aparat across all features (Fig. 3). This agreement is consistent with the low KS values reported in Table IV: $\text{KS} \leq 0.072$ for T^{on}, T^{off} and N, and $\text{KS} = 0.098$ for B. The model captures both timing (ON/OFF) and typical burst sizes for Aparat.

Filimo is more challenging; the synthetic CDF follows the global trend but exhibits visible gaps, especially for bytes per burst, this is reflected by larger KS values (up to $\text{KS} = 0.186$ for bytes) and by large tail errors for OFF duration (*i.e.,* $\epsilon_{0.99}$ for T^{off}). A plausible explanation is that Filimo combines long idle periods with occasional large bursts. Such extremes are rare, and their empirical frequency varies across captures. With a compact bootstrap-and-jitter emission model and finite training data, the tail is harder to match than the central mass.

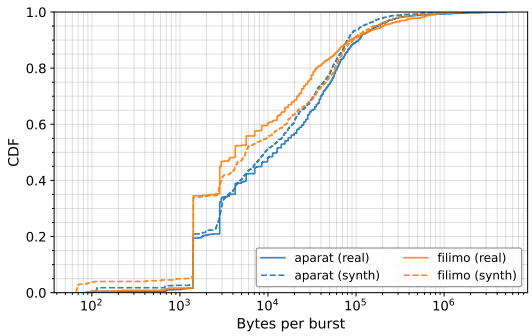
C. Temporal fidelity

We also evaluate the temporal component of the model. For each application, we estimate an empirical transition matrix $\hat{A}$ from burst-type sequences within each held-out test flow. We compute $\hat{A}^{\text{real}}$ from real held-out flows and $\hat{A}^{\text{syn}}$ from synthetic flows generated by the model.

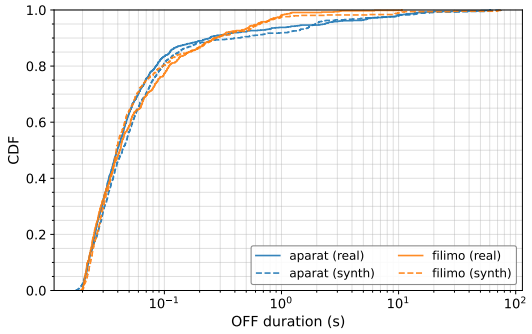
Fig. 4 reports the transition mismatch for YouTube (downlink). We choose YouTube as an illustrative example because it exhibits a larger number of burst types, making it a comprehensive test for temporal fidelity. We plot the signed difference $\Delta\hat{A} = \hat{A}^{\text{syn}} - \hat{A}^{\text{real}}$, where positive values indicate over-produced transitions and negative values indicate under-produced transitions. The largest discrepancies concentrate on low-probability off-diagonal entries. In contrast, the diagonal mass remains close to zero, which indicates that the generator preserves state persistence. This suggests that the model captures the main burst-type sequencing dynamics, while mismatches mainly affect rare transitions, which are sensitive to finite training data and to the exclusion of low-density bursts as noise during clustering.

D. Computational footprint

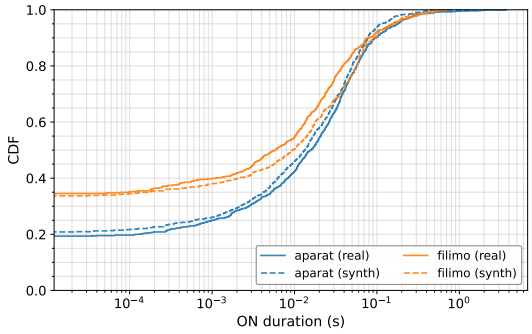
We finally quantify the storage and runtime overhead of the proposed burst-level generator to assess whether it is lightweight. All timings are obtained on a Mac laptop equipped with an Apple M4 Pro CPU (14 logical cores), 24 GB RAM, running macOS 15.2, using Python 3.12.0.



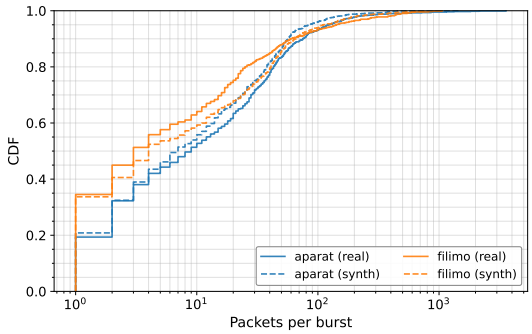
(a) Cumulative Distribution Function of the number of bytes per burst



(b) Cumulative Distribution Function of the OFF duration (*i.e.*, inter-burst time)



(c) Cumulative Distribution Function of the ON duration(*i.e.*, burst duration)



(d) Cumulative Distribution Function of the number of packets per burst

Fig. 3. Empirical CDFs of downlink burst features for Aparat and Filimo. Solid lines show real traces and dashed lines show synthetic bursts generated by the proposed model.

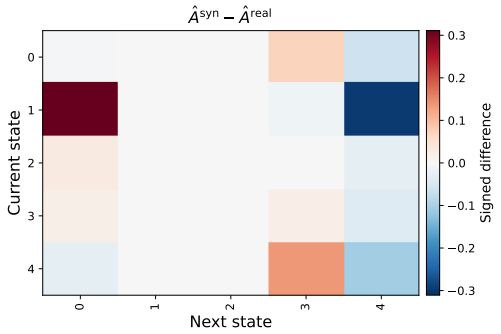


Fig. 4. Signed difference between synthetic and real transition matrices ($\hat{A}^{\text{syn}} - \hat{A}^{\text{real}}$) for YouTube (downlink) on held-out flows. Positive values (red) indicate over-produced transitions; negative values (blue) indicate under-produced transitions.

1) Model size: The learned artifact is compact. The first-order Markov component (`markov_model`) requires ≈1.9 KB when serialized, while the collection of type conditional emission models (`cluster models`) requires ≈144.6 KB, for a total footprint of ≈146.5 KB.

2) Generation throughput: We decompose generation into (i) sampling the discrete state sequence from the Markov model and (ii) sampling burst features conditioned on the sampled states. For $T = 200{,}000$ bursts, Markov sequence sampling takes 0.92 s on average ($\approx 2.16 \times 10^5$ transitions/s), and conditional burst sampling takes 8.45 s ($\approx 2.37 \times 10^4$ bursts/s). The end-to-end generator that outputs a tabular trace of $T = 200{,}000$ bursts takes 13.75 s on average ($\approx 1.45 \times 10^4$ bursts/s). The remaining ≈4.38 s (≈32% of the runtime) is dominated by Python-level data assembly (*i.e.,* per-burst object creation, type conversions, and DataFrame construction) rather than by the probabilistic model itself. Overall, the results confirm that the approach is lightweight in storage and in core sampling complexity, and that the main runtime cost stems from the emission sampling step.

VIII. Conclusion & Perspectives

This paper presents a lightweight, trace-driven traffic generator for 5G and beyond networks, designed for cellular Digital Twins. Instead of relying on application semantics, it learns traffic directly from packet captures by modeling flows as sequences of ON/OFF bursts described with compact features. Burst types and their transitions are learned via clustering and Markov modeling, combined with bootstrap sampling to ensure statistical robustness. Our design prioritizes compactness and scalability. As a result, rare tail events remain more difficult to reproduce than the central mass of the distributions. Still, the evaluations show that the generator preserves key burst-level statistics and temporal patterns across applications.

As a future work, the same pipeline can be replicated across additional scenarios, devices, and traffic directions. Specifically, we should study how uplink and downlink traffic could be coupled to model a fully bidirectional traffic session. More generally, we have also to investigate inter-flow interactions, *i.e.*, different flows may be correlated. Finally, packet-level semantics and application-specific structures can be layered on top of our burst-level generator using fine-grained synthesis methods, e.g., diffusion-based packet-header generation [16].

References

[1] J. Navarro-Ortiz et al. A survey on 5g usage scenarios and traffic models. *IEEE Communications Surveys & Tutorials*, 22(2):905–929, 2020.

[2] Michael Grieves et al. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary Perspectives on Complex Systems*, pages 85–113. Springer, 2017.

[3] Davide Villa et al. Colosseum as a digital twin: Bridging real-world experimentation and wireless network emulation. *IEEE Trans. Mobile Comput.*, 23(10):9150–9166, 2024.

[4] Xinyu Huang et al. Digital twin-driven network architecture for video streaming. *IEEE Network*, 38(6):334–341, 2024.

[5] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. Beauty and the burst: Remote identification of encrypted video streams. In *USENIX Security*, pages 1357–1374, August 2017.

[6] Zhi Tao et al. Deep learning-based modeling of 5g core control plane for 5g network digital twin. *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[7] Kannikar Siriwong et al. Study of bursty internet traffic. In *IEEE NCA*, pages 53–60, 2007.

[8] Ying Zhang et al. Understanding the characteristics of cellular data traffic. In *ACM SIGCOMM Cellnet Workshop*, pages 13–18, 2012.

[9] M. Zubair Shafiq et al. Characterizing and modeling internet traffic dynamics of cellular devices. *SIGMETRICS Perform. Eval. Rev.*, 39(1):265–276, 2011.

[10] Pablo Fernández Pérez et al. Characterizing and modeling mobile networks user traffic at millisecond level. In *Proc. ACM WiNTECH*, pages 64–71, Madrid, Spain, 2023.

[11] Mario Di Mauro et al. Multivariate time series characterization and forecasting of VoIP traffic in real mobile networks. *IEEE Trans. Netw. Serv. Manag.*, 20(4):4646–4662, 2023.

[12] Alberto Dainotti et al. Internet traffic modeling by means of hidden markov models. *Comput. Netw.*, 52(14):2645–2662, 2008.

[13] Markus Ring et al. Flow-based network traffic generation using generative adversarial networks. *Computers & Security*, 2019.

[14] Adriel Cheng. PAC-GAN: Packet generation of network traffic using generative adversarial networks. In *Proc. IEEE IEMCON*, pages 728–734, Vancouver, BC, Canada, 2019.

[15] Xuying Meng et al. Netgpt: Generative pretrained transformer for network traffic. *arXiv preprint arXiv:2304.09513*, 2023.

[16] Xi Jiang et al. NetDiffusion: Network data augmentation through protocol-constrained traffic generation. *Proc. ACM Meas. Anal. Comput. Syst.*, 8(1):1–32, 2024.

[17] K. V. Vishwanath et al. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Trans. Netw.*, 17(3):712–725, 2009.

[18] Shuodi Hui et al. Large-scale urban cellular traffic generation via knowledge-enhanced GANs with multi-periodic patterns. In *ACM KDD*, pages 4195–4206, Long Beach, CA, USA, 2023.

[19] Leonardo Bonati et al. Twinning commercial network traces on experimental open RAN platforms. In *Proc. ACM WiNTECH*, 2024.

[20] Michele Polese et al. Colosseum: The open ran digital twin. *arXiv preprint arXiv:2404.17317*, 2024.

[21] Enes Koktas et al. State aware traffic generation for real-time network digital twins. In *IEEE PIMRC*, Istanbul, Türkiye, 2025. IEEE.

[22] Xizheng Wang et al. Resolving packets from counters: Enabling multi-scale network traffic super resolution via composable large traffic model. In *NSDI*, pages 1541–1561, Philadelphia, PA, USA, 2025. USENIX.

[23] Benoît Claise and Brian Trammell. Information Model for IP Flow Information Export (IPFIX). RFC 7012, September 2013.

[24] Marziyeh Bayat et al. Itc-net-blend-60: A comprehensive dataset for robust network traffic classification in diverse environments – scenario a. Mendeley Data, V3, 2024.

[25] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *JASA*, 46(253):68–78, 1951.