High-Fidelity Cellular Network Control-Plane Traffic Generation without Domain Knowledge

Z. Jonny Kong
Purdue University
West Lafayette, IN, USA

Nathan Hu
Stanford University
Palo Alto, CA, USA

Y. Charlie Hu
Purdue University
West Lafayette, IN, USA

Jiayi Meng
University of Texas at Arlington
Arlington, TX, USA

Yaron Koral
AT&T Labs
Middletown, NJ, USA

ABSTRACT

With rapid evolution of mobile core network (MCN) architectures, large-scale control-plane traffic (CPT) traces are critical to studying MCN design and performance optimization by the R&D community. The prior-art control-plane traffic generator SMM heavily relies on domain knowledge which requires re-design as the domain evolves. In this work, we study the feasibility of developing a high-fidelity MCN control plane traffic generator by leveraging generative ML models. We identify key challenges in synthesizing high-fidelity CPT including generic (to data-plane) requirements such as multi-modality feature relationships and unique requirements such as stateful semantics and long-term (time-of-day) data variations. We show state-of-the-art, generative adversarial network (GAN)-based approaches shown to work well for data-plane traffic cannot meet these fidelity requirements of CPT, and develop a transformer-based model, CPT-GPT, that accurately captures complex dependencies among the samples in each traffic stream (control events by the same UE) without the need for GAN. Our evaluation of CPT-GPT on a large-scale control-plane traffic trace shows that (1) it does not rely on domain knowledge yet synthesizes control-plane traffic with comparable fidelity as SMM; (2) compared to the prior-art GAN-based approach, it reduces the fraction of streams that violate stateful semantics by two orders of magnitude, the max y-distance of sojourn time distributions of streams by 16.0%, and the transfer learning time in deriving new hourly models by 3.36×.

CCS CONCEPTS

• **Networks** → **Network simulations**; • **Computing methodologies** → **Modeling methodologies**.

KEYWORDS

4G/5G; mobile core network; control plane; traffic modeling and synthesis

IMC '24, November 4–6, 2024, Madrid, Spain.

ACM ISBN 979-8-4007-0592-2/24/11
https://doi.org/10.1145/3646547.3688422

ACM Reference Format:
Z. Jonny Kong, Nathan Hu, Y. Charlie Hu, Jiayi Meng, and Yaron Koral. 2024. High-Fidelity Cellular Network Control-Plane Traffic Generation without Domain Knowledge. In *Proceedings of the 2024 ACM Internet Measurement Conference (IMC '24), November 4–6, 2024, Madrid, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3646547.3688422

1 INTRODUCTION

The recently introduced Control-/User-Plane Separation (CUPS) in 3GPP Release 14 for 4G [1] and in 3GPP Release 15 for 5G [3], combined with a significant increase in control-plane traffic alongside explosive growth in data-plane traffic [6, 48], challenge mobile network operators and designers to innovate on mobile network architectural design not only for the data plane but also for the control plane in order to provide sustained mobile user experience.

Indeed, an increasing number of efforts have focused on open-source development of 3GPP-compliant MCN implementations [18] and novel MCN designs for improved performance [20, 24, 30]. Accurately assessing the performance of such open-source implementations or new MCN designs under real deployment scenarios, however, critically relies on using high-fidelity, large-scale control-plane traffic to drive the MCN operations. Despite the need, large-scale control-plane traffic traces in MCN are only accessible by mobile network operators, who are unlikely to share their traffic traces due to business and privacy concerns. As a result, the lack of public MCN control-plane traffic hinders the in-depth study of MCN design and performance optimization by the broad networking and systems communities.

A classic approach to mitigating the lack of real network traffic traces is to generate synthetic traces. However, the large body of traffic synthesis work for LTE and 5G has focused on the data plane of MCN, *i.e.,* on modeling data-plane traffic (*e.g.,* [11, 29, 37, 38, 40]). On the control-plane side, Dababneh *et al.* [16] modeled the total control-plane volume on LTE's MCN but did not model the fine-grained inter-arrival time of successive events or the intricate event dependence specified by 3GPP for each individual UE; they also ignored the traffic diversity in device types and time-of-day.

Compared to Internet data traffic generation, cellular network control-plane traffic generation faces some common fidelity requirements but also unique new requirements. A control-plane traffic trace consists of multiple streams of control events (samples), one stream per UE, typically throughout a day. **(1) Stateful semantics.** First and foremost, the traffic generator must accurately capture the rigid inter-dependence of sample features within a stream, adhering

to domain-specific rules, *i.e.,* 3GPP UE state machines, ensuring only semantically correct generated streams will be used to drive the MCN. **(2) Multi-modal features**. Like data traffic, control-plane traffic encompasses multiple fields. Specifically, each sample includes a control event type and a timestamp, and thus the traffic generator is tasked not only with producing realistic distributions for individual fields, but also generating realistic correlations between fields within each sample. **(3) Variable flow length**. As in data traffic, the traffic generator needs to capture diversity in UE flow length. For instance, the number of events within a fixed time window could vary among clients, which follows from the diversity in UE activity levels. **(4) Long-term data drifts**. As control-plane traffic lasts throughout the day, the traffic generator needs to capture data distribution drifts that occur over time, such as variations throughout a day.

To meet these requirements, the state-of-the-art cellular control-plane traffic generator SMM [45] employs a traditional modeling methodology. In particular, it derives a statistical model, *i.e.,* a Semi-Markov model, based on the 3GPP UE state machines and fits the model parameters (state transitions probabilities and sojourn times) on the real trace. Such a traditional modeling approach, however, critically relies on domain knowledge, *e.g.,* the 3GPP standard, and hence requires re-design as the domain knowledge evolves. First, a separate model needs to be designed for each generation of cellular technology (4G or 5G). Second, even for a particular generation of technology, standards evolve continuously over time as new 3GPP specifications are published. For example, the development of LTE was carried out through 3GPP Release 8 (2008) to Release 16 (2020) [4]. Therefore, continuous manual effort is required to keep such domain-knowledge-dependent generators up to date with the latest standards.

An additional limitation of the above traditional modeling approach is that the model parameters are rigid, and a single model cannot capture the diversity of control-plane traffic trace. As a result, the authors had to cluster control-plane-traffic and instantiate 20,216 models (one per cluster per hour) which are inconvenient to maintain and deploy.

In light of the limitation of the traditional modeling approach, in this work, we study the feasibility of developing a high-fidelity MCN control plane traffic generator by leveraging generative machine learning (ML) models. Our approach was inspired by the recent work on using Generative Adversarial Networks (GAN) combined with LSTM in synthesizing *data-plane traffic* traces [39, 73] which showed GAN-based approaches effectively learn the distributions and correlations of packet- or flow-level header fields in Internet traffic.

However, when we adapted prior-art GAN-based approaches to generate *control-plane traffic* of the cellular network, we found it suffers a number of limitations. (1) The prior-art approaches such as NetShare [39] cannot capture the stateful semantics, as 22.10% of the streams in the synthesized trace contain one or more events that violate state transition rules stipulated by the 3GPP protocols. (2) NetShare fails to capture fine-grained temporal properties, such as the distribution of sojourn times, *i.e.,* the duration that a UE stays in a particular state of the 3GPP state machine. For example, the maximum y-distance between the CDFs of the real and synthesized traces' sojourn times reaches 61.7%. (3) The GAN-based approach cannot efficiently adapt to data distribution drifts that occur through time via transfer learning. In particular, training a model for one hour and applying transfer learning for each of the next 5 hours takes almost 2X as long as direct training a 6-hour model from scratch. (4) These GAN-based LSTM approaches require complex and specialized enhancements to alleviate mode collapse [36]. (5) Finally, these approaches require complex enhancements to the model design to address LSTM's tendency to forget past states [7, 57] in order to capture long-term dependencies.

To overcome these drawbacks, we develop a transformer-based framework, CPT-GPT, that accurately captures complex dependencies among the events in each stream to meet the fidelity requirements (stateful semantics and distribution of multiple features). In doing so, it not only achieves comparable fidelity in traffic generation without domain knowledge as needed in SMM, but also achieves higher fidelity than prior-art GAN/LSTM-based approaches [39, 73] without the need to use GAN or model enhancements such as dealing with long-term dependencies and mode collapse. Instead, the design of CPT-GPT only has to focus on a few challenges in the input layer and output layer design: (1) How to design tokens for the input layer to capture multi-modal features per sample? (2) How to introduce generation stochasticity at the output layer for multi-modal features? (3) How to efficiently update the model over time?

We implement the complete cellular network control-plane generation model CPT-GPT in PyTorch, based on a decoder-based transformer model [62]. We extensively evaluate CPT-GPT by comparing it with the traditional modeling-based SMM [45] and the SOTA GAN-based NetShare [73], on a large-scale control-plane traffic trace (for 380K UEs) from a leading mobile operator. Our evaluation shows that (1) Compared to SMM, CPT-GPT requires no domain knowledge, yet it can achieve close-to-zero semantic violations, similarly accurate sojourn time distribution in terms of the max y-distance of the CDFs, and more accurate percentage breakdown of different event types. For example, averaged over the three types of UEs (phones, connected cars, and tablets), CPT-GPT's differences compared to the real dataset in the breakdown of the two dominant event types (`SRV_REQ` and `S1_CONN_REL`) are 1.01% and 1.21% lower than SMM, respectively. (2) Compared to NetShare, CPT-GPT achieves much higher fidelity in control-plane traffic generation. (i) It reduces the fraction of streams that violate stateful semantics by two orders of magnitude, from 22.1%, 11.5%, and 16.9% to 0.2%, 0.4%, and 1.5% for phones, connected cars, and tablets, respectively. (ii) It also improves the sojourn time distribution of the synthesized traces. For the three device types, the max y-distance of sojourn time distribution from the real trace is reduced by 10.7%, 9.1%, and 28.3% respectively, averaging over the two 3GPP states. (iii) CPT-GPT is more efficient in being applied transfer learning to adapt a pretrained model for a given hour-of-day to the subsequent hours. Specifically, it reduces the time needed to derive hourly models by 3.36×. (3) CPT-GPT generates diverse control-plane traffic without memorizing individual sequences from the training set. Specifically, when examining generated sub-sequences of length 20, none are repeated from the training set.

In summary, this work makes the following contributions:

- We develop the first control-plane traffic generator that does not require domain knowledge yet achieves comparable accuracy as

Table 1: Control-plane event types in 4G and 5G.

4G	5G	Description
ATCH	REGISTER	Register the UE with the MCN
DTCH	DEREGISTER	De-register the UE from the MCN
SRV_REQ	SRV_REQ	Create a signaling connection to allow UE to send/receive data and control-plane messages
S1_CONN_REL	AN_REL	Release the signaling connection and others resources in both control and data planes
HO	HO	Switch the UE from the current cell coverage serving it to another cell
TAU	–	Update the UE's tracking area

the prior-art Semi-Markov-Model-based traffic generator which heavily relies on domain knowledge.

- We show how transformer-based models can overcome the low fidelity limitations of prior-art GAN/LSTM-based models in synthesizing control-plane traffic, more efficiently employ transfer learning to deal with traffic shifts over time, and does not memorize individual sequences from the training set.

2 BACKGROUND & MOTIVATION

2.1 Cellular Network Control-Plane Traffic

The Mobile Core Network (MCN) serves as the central hub of the cellular network. It not only forwards data traffic between each user equipment (UE) and the Internet, but also orchestrates various functions of the cellular network to provide seamless and reliable communication services to the users. To effectively manage both data and control traffic within the cellular network, Control-/User-Plane Separation (CUPS) was first introduced in 3GPP Release 14 for 4G [1] and further refined in 3GPP Release 15 for 5G [3]. This separation divides the cellular network into a data plane and a control plane, streamlining the processing of data and control traffic, respectively.

Control events. To effectively manage and control the communications in the cellular network, 3GPP defines various types of control-plane events for both LTE and 5G.

Table 1 lists the primary control-plane events in 4G and 5G and describes their main functionalities. Note that (1) we ignore the events that happen only between the UE and the Radio Access Network (RAN), as they do not involve the MCN; (2) although each event corresponds to a series of control-plane messages among UE, RAN, and MCN, traffic synthesis is only concerned with generating control events originated from UEs, as mapping from a control-plane event to messages is fixed as dictated by the 3GPP protocol; and (3) what to do in case of event failure, *i.e.,* due to message exchange failure, is the responsibility of the downstream applications, which is beyond the scope of this paper; the traffic generator is concerned with generating semantically correct sequences of events.

Stateful semantics of control-plane traffic. In contrast with the stateless Internet traffic, the control-plane events in the cellular network are not independent. The 3GPP protocol not only specifies that every UE has to follow two state machines when interacting with the MCN, but also describes intricate dependence of the control events on the states in the UE state machines. Specifically, the two state machines for 4G/5G are EMM/RM (EPS[1] Mobility Mgmt. / Registration Mgmt.) and ECM/CM (EPS Conn. Mgmt. / Conn. Mgmt.) respectively [2, 3], each with two primary states (REGISTERED and DEREGISTERED for EMM/RM, and CONNECTED and IDLE for ECM/CM).

To capture the intricate dependence of the control events on the above four UE states, [45] developed two-level hierarchical state machines for 4G and 5G as shown in Figure 1. Figure 1a shows the two-level state machine of 4G. The top-level state machine is a merged state machine of the EMM and ECM state machines with three UE states, DEREGISTERED, CONNECTED, and IDLE, derived from insights into how a UE transitions in the two state machines. At the bottom level, there are two sub-state machines embedded in the top-level CONNECTED and IDLE states, which capture other dependence of the control events on the top-level states.

While 4G and 5G exhibit similarities in control events and UE states, it is necessary to customize and re-derive the two-level state machine of 5G to align with the 3GPP protocol as shown in Figure 1b. Specifically, as TAU is not supported in 5G, the corresponding states and transitions are removed from the two-level state machine of 4G. Moreover, several 4G event types, ATCH, DTCH, S1_CONN_REL, are replaced with REGISTER, DEREGISTER, and AN_REL, respectively.

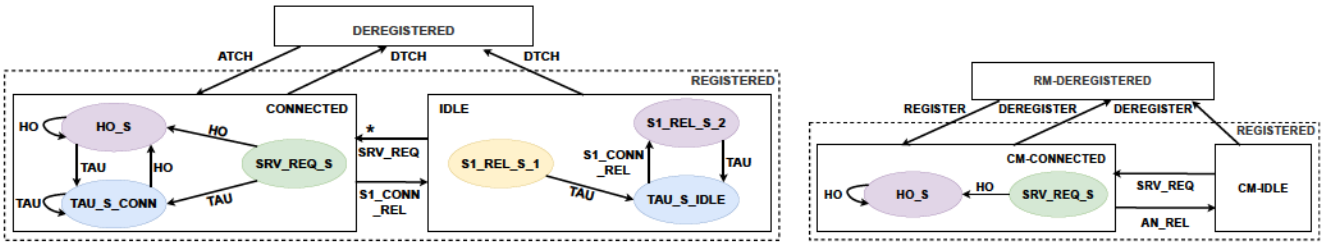
2.2 Motivation for Generating Control Traffic

To provide sustained mobile user experience, mobile network operators and designers continuously innovate on mobile network architectural design not only for the data plane but also for the control plane. Such innovations rely on realistic, high-fidelity control-plane traffic traces for evaluation, benchmarking, and optimization. We describe two motivating use cases in data-driven network design and management that demand realistic control-plane traffic traces.

Performance evaluation of MCN design. In-depth evaluation of various aspects of novel MCN design (*e.g.,* [20, 24, 30]), including throughput, end-to-end latency, scalability, and fault resilience, in handling control events originating from a large UE population rely on realistic control-plane workload. For example, right after the first control-plane traffic generator [45] became available, the Aether community started using it to study the scalability of Aether 5G core design.

Real-time network management. Network management is critical to various stakeholders ranging from network operators to end users, It has been intensively studied for data flows (*e.g.,* five-tuples) via telemetry [5, 10, 14, 15, 27, 41, 42, 50, 71, 75] which critically relies on real-time monitoring of network traffic. Accurate control-plane traffic models can help to develop effective monitoring schemes, *e.g.,* with better accuracy and smaller memory footprints. For example, such models can help to determine a good sampling rate for sampling-based monitoring when collecting telemetry metrics.

[1]EPS: Evolved Packet System, comprising the RAN and the MCN in 4G.



(a) 4G (b) 5G

Figure 1: The two-level hierarchical UE state machines of 4G and 5G [45].

3 OVERVIEW

We formally state the control-plane traffic generation problem and discuss design challenges and how the prior-art heavily relies on domain knowledge.

3.1 Problem Formulation

A cellular control traffic dataset $D = \{S_1, S_2, ..., S_n\}$ consists of multiple streams, where each stream S_i represents a sequence of events generated by a specific UE. Each stream $S_i = \{UE\_ID_i, device\_type_i, events_i\}$ consists of a UE identifier, a UE device type, and a sequence of events, $events_i = \{\{t_{i1}, e_{i1}\}, \{t_{i2}, e_{i2}\}, ..., \{t_{ik}, e_{ik}\}\}$, where each event e_{ik} is accompanied by a timestamp t_{ik} indicating when it occurred.

Given such a control plane traffic dataset D, our objective is to develop a framework capable of producing a synthesized dataset D' of arbitrary size that exhibits high fidelity in terms of a selection of fidelity metrics of importance to the downstream use cases.

3.2 Challenges

The primary challenges in synthesizing high-fidelity control-plane traffic datasets come from the generality of such a framework and from the multiple dimensions of fidelity required by the use cases of control-plane datasets:

- C1: *Generality – requiring minimal domain knowledge.* The framework should require as little domain knowledge as possible, since relying on domain knowledge leads to extra re-design effort upon domain knowledge changes, *e.g.,* revisions or new releases of 3GPP standards, or mixed technology deployment scenarios. Furthermore, even within the same generation of network technology, the standard evolves continuously with new 3GPP releases. For example, LTE has been developed through 3GPP Release 8 to 16 [4], whereas 5G began with 3GPP Release 15 and still actively evolving [22]. As the network standard continuously evolves, the control-plane traffic of UEs experiences a continuous shift which requires the models to be re-instantiated repeatedly.
- C2: *Stateful semantics.* Each stream of the control-plane traffic trace consists of samples, *i.e.,* control events, each of a particular event type (Table 1). The framework needs to capture the rigid inter-dependence of event types, dictated by domain rules (*e.g.,* 3GPP UE state machines). Semantic correctness is essential since only semantically correct datasets shall be used, *e.g.,* in comparing different designs or implementations of an MCN.
- C3: *Multimodal features.* In addition to event types, the samples in each control-plane stream also encompass several other fields, including an associated timestamp, that directly affect the load on the MCN. Thus, the framework needs to generate control-plane traffic with realistic distributions for each field, which are required to evaluate how an MCN operates under real-world workloads, such as the duration an MCN has to maintain per-UE states in stateful implementations (*e.g.,* [20]). These include distributions (breakdowns) of different control event types in the trace and the sojourn time of a UE staying in each state, as each type of events will invoke a different set of network functions in the MCN, and the sojourn time staying in each UE state affects the average and bursty load incurred by the UEs on the MCN.
- C4: *Variable flow length.* Like Internet data traffic, the framework needs to generate datasets consisting of streams of varying lengths, with the length distribution matching that of the real datasets. Realistic flow length distribution is important because it reflects the realistic workload experienced by the MCN in real deployment due to individual streams (UEs).
- C5: *Long-term data drifts.* The framework needs to capture control-plane traffic drifts over time, such as diurnal variations, due to variations of UE activity characteristics throughout different hours of the day. Accurately modeling control-plane traffic drifts enables evaluating autoscaling capabilities of MCN implementations [20, 24].

Fidelity metrics. To quantify how well a framework tackles the above challenges (C2-C5) in generating high-fidelity control-plane traffic, we use the list of fidelity metrics shown in Table 2. First, to calculate the percentage of *semantic violations* and *sojourn time distributions*, we replay the synthesized trace against the 3GPP-dictated state machine (Figure 1), and record the duration that the UE stays in each state and the number of events that violate state transitions, respectively. Second, *event type breakdown* and *flow length distribution* can be directly derived from the synthesized trace. Finally, to evaluate the *adaptability to data drifts*, we assume a transfer learning setup, where the model in the framework is trained on a trace collected in a particular hour, and subsequently fine-tuned to a trace collected from another hour that may have data distribution shift, and measure the re-training time and accuracy of the adapted model.

Table 2: Fidelity metrics used for evaluating the synthesized dataset.

Metric	Definition	Goal
Semantic violation	Percentage of events and streams that violates 3GPP-defined state transitions	Evaluate C2
Sojourn time distribution	The average time a client stays in a particular state	Evaluate C3
Event type breakdown	Percentage of each event type	Evaluate C3
Flow length distribution	Number of events in each stream	Evaluate C4
Adaptability to data drifts	Training time needed to adapt to another dataset of different characteristics	Evaluate C5

3.3 Prior-art Control-plane Traffic Generation

The state-of-the-art control-plane traffic generator SMM [45] relies heavily on domain-specific knowledge. (1) To satisfy stateful semantics (C2), the authors manually derived the above two-level hierarchical state machines for 4G and 5G (Figure 1) based on 3GPP standard to capture the intricate inter-dependence of the control events on the four UE states. (2) They then manually converted the two-level state machine into a Semi-Markov Model (SMM) to model the probability of the transition from one state to another and the duration of a UE staying in one UE state before switching to another (referred to as SMM sojourn time). (3) In fitting the parameters of the SMM (to satisfy C3 and C4), the authors applied further domain knowledge. They discovered high diversity of control-plane traffic in terms of device types, time-of-the-day, and across individual UEs, and then clustered the UEs in the real trace into hundreds of UE clusters, each with similar features such as flow length and variation of sojourn time which are domain-specific, and then instantiated an SMM for every UE cluster for each device type and hour-of-day. (4) To finally model sojourn time - a key SMM model parameter (to satisfy C3), they again applied domain knowledge; they derived one CDF model for each transition in the SMM, after discovering that traditional probability distributions used for modeling the inter-arrival time of Internet traffic (*e.g.,* Poisson, Pareto, Weibull, and TCPlib) cannot accurately model the control-plane traffic in cellular networks. As a result, a total of 20,216 Semi-Markov models were derived to cover all 24 hours of the day and three device types, *i.e.,* phones, connected cars, and tablets, which required deriving a total of 283,024 CDFs to model the sojourn time in all the models.

As SMM's design heavily relies on domain knowledge, *e.g.,* the 3GPP standard, it needs to be re-designed and re-instantiated as the domain knowledge evolves. In summary, *prior-art control-plane generator SMM heavily relies on domain knowledge which not only incurs significant manual effort but also requires repeating the effort in reacting to changes and complications in domain knowledge.*

4 SYNTHESIZING CONTROL-PLANE TRAFFIC WITHOUT DOMAIN KNOWLEDGE

To design a traffic generator that does not rely on domain-specific knowledge, we leverage generative ML models which can be directly trained end-to-end on raw traces, without needing domain knowledge such as 3GPP protocols. We begin by exploring prior-art ML-based methods for generating network time series, which have been shown to generate Internet data traffic with high fidelity, to understand their limitations in generating cellular network control-plane traffic traces. We then present the design of CPT-GPT, which addresses the limitations of existing ML-based approaches.

4.1 Dataset Overview

To enable our study of control-plane traffic generation, we collected a cellular control-plane traffic dataset in collaboration with a major carrier in the US. The dataset is collected on the operator's LTE network by randomly sampling from UEs across the entire U.S. over a span of 8 days in 2022. In total, the dataset contains 73,153,370 control events, from 430,939 UEs belonging to three device types: phones (278,389), connected cars (113,182), and tablets (39,368).

4.2 Prior-art ML-based Approach and Limitations

Encouraged by the effectiveness of Generative Adversarial Networks (GANs) applied to generating time sequences such as NLP [12], systems researchers have applied GANs to develop generators for networked traffic [13, 28, 39, 55, 73], webpage views [39], and cluster job traces [39] with encouraging results. Compared with classic simulation-based or statistical-based approaches, the GAN-based approach requires minimal domain knowledge and thus has the potential to support diverse types of network time series with little to no customization.

DoppelGANger [39] and NetShare [73] are state-of-the-art GAN-based frameworks designed for network traffic generation. Their key ingredient is to generate the metadata per stream (*e.g.,* 5-tuples of a TCP connection) and the stream of samples (*e.g.,* IP headers) conditioned on the metadata separately, using an MLP-based generator and an LSTM-based generator, respectively. Such a decoupled two-model architecture demonstrates superior fidelity over several previous GAN-based traffic generators (*e.g.,* [19, 74]) in generating Internet data traffic such as webpage access and cluster usage traces. NetShare builds upon DoppelGANger and specializes in generating IP header and flow traces. By incorporating domain knowledge such as domain-specific encoding schemes (*e.g.,* bitwise encoding [69] for IP addresses and IP2Vec [54] for ports), NetShare is shown to generate such traces with improved fidelity.

4.2.1 Limitations of GAN-based Approach. However, our evaluation reveals several limitations of prior-art GAN-based approaches in synthesizing cellular network control-plane traffic datasets.

Adapting NetShare to control traffic. We adapt NetShare to synthesize cellular control-plane traffic as follows. The original NetShare uses an MLP-based generator and LSTM-based generator to respectively generate a "metadata" and a "time series" that together form a stream, where metadata refers to fields that are universal to the entire stream (*e.g.,* the 5-tuple). The equivalent metadata in cellular network control-plane traffic is the UE ID. However, since a UE ID is a hashed string without semantic meaning, it is not meaningful to generate it with a model or evaluate its fidelity. Instead, we use a random string generator to generate UE IDs separately.

Table 3: Semantic violations in control-plane traffic synthesized by NetShare.

Perc. event violations	2.61%
Perc. streams w/ at least one violating event	22.10%
State and event of top 3 violation	
S1_REL_S, S1_CONN_REL	1.16%
S1_REL_S, HO	0.76%
CONNECTED, SRV_REQ	0.41%

Therefore, we discard the metadata generator, and only use NetShare's LSTM-based generator to generate time series where each sample in the time series contains three fields: event type, interarrival time, and a "stop flag" that indicates whether the current sample is the last in the stream. [2] Other model hyperparameters of NetShare are kept unchanged. For comparison, we present the dataset generated by both NetShare and CPT-GPT, with a detailed description of CPT-GPT's design deferred to §4. The model is trained following the methodologies that will be described in §5.1, and the inference setup is deferred to §5.1, §5.2 and §5.5.

Limitation 1 (L1): State-of-the-art ML-based traffic generators do not achieve stateful semantic correctness.

Many network datasets exhibit stateful semantics [21, 35, 45, 51], because they are usually produced by end hosts or middleboxes that operate according to some network protocols, upper-layer application rules, or middlebox policies. In the case of cellular control-plane traffic, the stream of control events produced by each UE is the outcome of a series of state transitions on the UE state machine derived from the 3GPP standard (Fig. 1).

We evaluate the semantic correctness of NetShare-generated trace by feeding each stream into the 3GPP state machine, and count how many events violate the state transitions. Table 3 shows that 2.614% of the events in the dataset synthesized by NetShare violate the 3GPP-defined state transitions. Such a degree of event violations translates to 22.10% of the streams generated containing at least one semantic-violating events, rendering these streams potentially useless for downstream applications. A detailed analysis shows the reason for such a high degree of stateful semantics violations is that NetShare fails to capture the fact that S1_CONN_REL and HO events should not occur when the UE is in S1_REL_S state (Fig. 1a).

Limitation 2 (L2): State-of-the-art ML-based traffic generators do not synthesize high-fidelity traces in terms of sojourn times.

Figure 2 compares the CDF of the average sojourn times per UE in the CONNECTED state for each synthesized dataset as well as the real dataset, for phone UEs. We see that NetShare does not perform as well in generating high fidelity sojourn times. Specifically, while the majority of streams in the real dataset have an averaged CONNECTED state sojourn time ranging from 5 to 50 seconds, NetShare frequently generates streams with sojourn times spanning

[2]NetShare could potentially be enhanced by replacing the LSTM model in its GAN generator with a transformer model. However, as we will demonstrate with CPT-GPT, a transformer model can achieve sufficient performance with supervised learning, eliminating the need for GAN-based training.

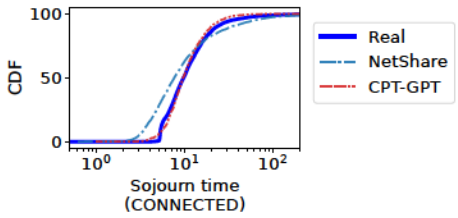


Figure 2: Distributions of the average sojourn time in the CONNECTED of each UE, comparing real and synthesized traces, for phone UEs.

Table 4: Time needed to train NetShare from scratch, or perform transfer learning on an existing model to adapt to different workloads. Training stops when fidelity metrics show diminishing returns. Measurement is done on an NVIDIA A100 GPU.

6-hour model from scratch	108.36 mins
1-hour model from scratch	43.08 mins
1-hour model from finetuning from another hour	30.41 mins
6 1-hour models total from transfer learning	195.12 mins

2 to 100 seconds with a non-negligible probability. This results in a significant difference in the maximum y-distance between the CDFs of the synthesized and real dataset, at 27.9% and 6.4% for NetShare and CPT-GPT, respectively. We refer readers to §5.2 for the results for the other two types of UEs.

Limitation 3 (L3): State-of-the-art ML-based traffic generators cannot efficiently adapt to different workloads via transfer learning.

Just like Internet data traffic, the control-plane traffic of cellular networks generated by UEs exhibits variations over time, which can be attributed to diurnal, weekly, or seasonal fluctuations in UE behavior, as well as network changes resulting from the continuous cellular network evolution or incremental deployment of a given generation of cellular networks. Therefore, the traffic generation framework may need to re-train a new ML model that captures the resulting variations in control-plane traffic, *e.g.*, once every month. To reduce the model training cost for such drifting workloads, instead of training a new ensemble of hourly models from scratch, a potentially efficient approach is to train a base model for a specific hour-of-day from scratch, and then adapt it using transfer learning to generate a model for each remaining hour of the day [8, 52, 68].

However, we found that state-of-the-art GAN-based frameworks are inefficient in performing such transfer learning. We measure NetShare's training time on trace collected over 6 hours for the phone device type, and compare it with training NetShare just on the first hour of the trace and then re-trained to adapt to the five subsequent hours recursively (we refer readers to §5.5 for the detailed methodology). Table 4 compares the training time for the two approaches. We observe that it takes 108.36 minutes to train a single NetShare model covering 6 hours of traces from scratch. In contrast, the time needed to train an ensemble of specialized

models via transfer learning is 195.12 minutes, almost twice as long. This indicates that the GAN/LSTM design cannot leverage transfer learning to adapt to shifting workloads effectively, which has been shown in prior works [49, 65, 66].

Limitation 4 (L4): State-of-the-art ML-based traffic generators use LSTMs which are insufficient in learning long-term dependencies in the traces.

DoppelGANger and NetShare use LSTMs to capture dependencies in sequential data. However, using off-the-shelf LSTMs to synthesize control plane traffic, *i.e.,* one step at a time, suffers from an inherent limitation to forget states over long sequences, as shown by authors of DoppelGANger and NetShare [39, 73]. To overcome this problem, DoppelGANger and NetShare employ batch generation, *i.e.,* generating multiple samples in each LSTM step, thereby reducing the number of passes to produce a stream of a certain length and hence the amount of forgotten states. However, since batch generation produces multiple samples each step, it sacrifices intra-batch dependencies, which could compromise fidelity, as each cellular control event depends on the event immediately preceding it. This potentially contributed to the low fidelity measures in L1 – L3.

Limitation 5 (L5): State-of-the-art ML-based traffic generators use GANs which suffer from mode collapse.

GAN training is known to suffer from mode collapse [36, 60, 61], where the generator learns to generate only one or few plausible outputs, rather than a wide variety of outputs as in the training dataset. Rather than using general solutions for mode collapse, DoppelGANger and NetShare alleviate mode collapse with a specialized normalization scheme. Specifically, each stream is normalized individually, using the min and max values within the stream, rather than a global min/max value. This shows off-the-shelf GAN-based traffic generators are susceptible to mode collapse, and require tailored, traffic type- and model-dependent enhancements to mitigate the issue.

4.3 Why Transformers?

A transformer [62] is a deep learning architecture built around the "attention" mechanism. Input text is represented as a sequence of tokens, and at each layer, an attention score is computed between every pair of tokens through a multi-head attention mechanism. The attention mechanism facilitates the learning of dependencies between distant tokens, in contrast to RNNs or LSTMs which process a stream of tokens sequentially. Moreover, such design allows for the parallel processing of the tokens which are highly efficient on today's hardware accelerators such as GPUs.

Transformers have the potential to address the aforementioned limitations (L1-L5) of existing ML-based traffic generators. (1) Transformers have been shown to perform well for a variety of generative tasks, such as generating code that conforms to a rigid grammar (L1) [34], as well as capturing complex dependencies among multiple fields in tokens in a stream (L2) for multi-modality tasks such as music generation [26]. (2) Furthermore, since their attention mechanism allows the input sequence to be processed in parallel, they have the potential to reduce the training time on today's hardware, compared to LSTMs which process the input sequentially (L3) [62]. (3) Transformer is known to have superior performance than LSTMs on learning long-term dependencies (L4) [33], as its attention mechanism models the dependencies without regard to their distance in the sequence. (4) Finally, they are typically trained in a supervised fashion and thus do not suffer from mode collapse like GANs (L5) [36].

4.4 CPT-GPT Design

Design 1: We employ a multi-modal tokenization scheme tailored for cellular network control traffic.

The powerful attention mechanism at the core of transformers was originally designed to capture pair-wise dependencies of *finite-sized, single-modality* tokens in NLP tasks that process streams of words from a finite set of vocabulary, *e.g.,* the state-of-the-art Llama-3 [47] employs a vocabulary size of 32,000. The finite-sized vocabulary of words allows them to be effectively encoded into one-hot tokens. In contrast, in network traffic, streams of samples, *e.g.,* UE control events in cellular networks, contain multiple modalities (*e.g.,* continuous interarrival time and categorical event types in control-plane traffic), and furthermore, the continuous field has a significantly larger input space than categorical fields. Together the mixed types of fields in network traffic result in a significantly larger input and output space than NLP tasks, rendering language-style one-hot tokenization inapplicable to network traffic.

Similarly, vision transformers designed for CV tasks [17] convert image patches to floating point matrices and then encode them into tokens, which also have a single modality although a large token space. In comparison, network traffic has multiple modalities of different types, either continuous or numerical, that need to be treated differently, and thus the tokenization scheme in vision transformers cannot be easily applied in network traffic either.

While one may concatenate multiple fields in each sample of network traffic stream and treat them as a single field, it is not clear whether language or vision transformers can learn the distribution of each individual field or capture the correlation across the fields.

Our solution. To integrate multi-modality data with the transformer architecture, we designed a specialized tokenization strategy where each token is the concatenation of multiple sub-tokens, each representing a specific modality, as illustrated in Figure 3. For categorical fields, such as event type, we convert them into sub-tokens using one-hot encoding. Numerical fields, such as interarrival time, are applied log scaling (*i.e.,* $x' = log(x + 1)$) and then linear scaled to a range of $0 - 1$, where 0 and 1 correspond to the minimum and maximum interarrival times across the dataset[3]. Additionally, generating streams of varying lengths requires an additional "stop flag" sub-token, marking whether the token is the last in a stream. The stop flag is a categorical field taking values of zero or one, similarly as in NetShare [73].

Correspondingly, as shown in Figure 3, the "embedding" layer in classic NLP transformers is replaced with a linear layer that maps a d_{token}-dimension space to a d_{model}-dimension space, where d_{token} is the dimension of the tokens (9 in our case, concatenating the

[3]The rationale behind log scaling is that interarrival time spans several orders of magnitude, where the majority of the data is concentrated in smaller ranges, as shown in Figure 7 in Appendix B. Log scaling expands the range of smaller values and compresses the range of larger ones, making the data more uniformly distributed.

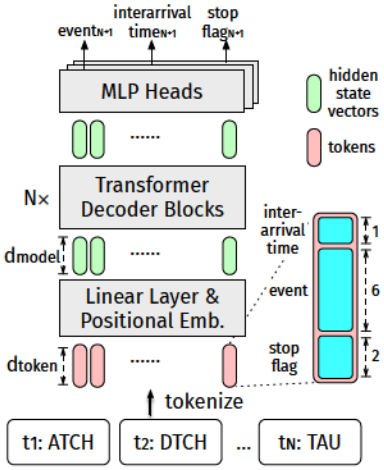


Figure 3: CPT-GPT architecture and tokenization scheme.

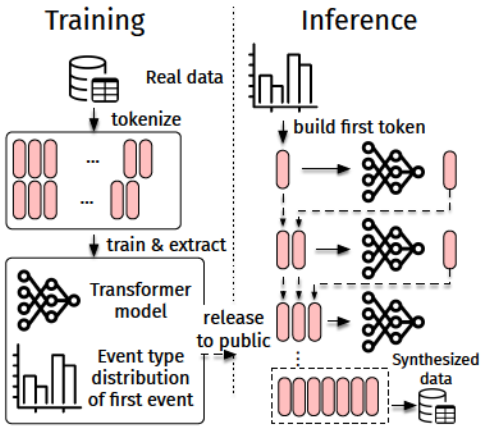


Figure 4: Operational architecture overview.

three sub-tokens 1+6+2), and d_{model} is a model hyperparameter representing the attention hidden size. Furthermore, to produce multi-modality outputs, CPT-GPT employs three multi-layer perception (MLP) heads following the final attention block, each making predictions for a particular field.

Design 2: We design the model to output distribution parameters for numerical fields to improve generation stochasticity.

Model architecture design for producing multi-modality output is fairly standard. For instance, in NetShare, each modality is assigned its own MLP head. For fields of a categorical modality, such as event type and the "stop flag", the MLP head uses a SoftMax in the last layer to output a vector representing the probability distribution over possible values. During inference, the output is selected either by sampling from the probability distribution, as commonly done in language models [53], or by simply choosing the element with the highest possibility, as in NetShare.

However, the interarrival time field in cellular control-plane traffic is of numerical modality with continuous values. Previous designs, such as NetShare, output a single numerical value from the corresponding MLP head. But as we will show in the ablation study (§5.3), such a design results in a lack of generation stochasticity, *i.e.,* the model tends to generate the mean value instead of producing a diverse range of values, failing to address C3 & C4. Overcoming this limitation requires designing alternative mechanisms to ensure generation stochasticity for such numerical fields.

Our solution. We enhance the MLP head for a numerical field (*e.g.,* the interarrival time) by configuring it to output the parameters of a probability distribution, rather than a single numerical value. We empirically chose normal distribution and make the model output both a mean and a standard deviation. During training, the Gaussian negative log-likelihood (NLL) loss function is used for numerical fields, while categorical fields continue to use cross-entropy loss, and the training minimizes the weighted sum of these losses across fields. During inference, values for numerical fields are randomly sampled from the predicted probability distributions, similarly as in the approach used for categorical fields.

Design 3: We employ transfer learning to efficiently generate models for network traffic shifts.

As mentioned in §3.2 (C5), the ML model needs to be updated to reflect time-varying characteristics in the dataset, stemming from short-term UE behavior fluctuations (*e.g.,* diurnal and weekly), as well as longer-term changes due to evolving cellular network deployments or technologies. Therefore, the traffic generation framework needs to periodically retrain the model on the latest collected network traces, which potentially incurs significant training costs.

Our solution. To reduce the training time needed to train new models as the dataset characteristics drift through time, we employ a transfer learning strategy where the transformer model is first trained on a specific hour, and then we perform transfer learning on it to generate models recursively for each subsequent hour. Such a transfer-learning scheme would provide significant time savings, as we will show that training based on a pretrained model from another hour is much faster than training a model from scratch.

4.5 Architecture Overview

Combining the design insights above, the overall architecture and workflow of CPT-GPT are shown in Figure 4.

Training. The traffic generation framework developer, *e.g.,* a cellular operator, collects a control-plane traffic dataset and applies CPT-GPT's tokenization scheme to tokenize it, and subsequently uses it to train the transformer model. Additionally, the developer extracts the event type distribution of the initial events across all streams in the dataset, which will be used to bootstrap the CPT-GPT inference as explained in the next paragraph. The trained model weights, along with the initial-event-type distribution, will be packaged together and released to the public.

Inference. To synthesize a control-plane traffic trace for a UE population of size N, a CPT-GPT user invokes the transformer-decoder-based CPT-GPT N times.

In each inference, CPT-GPT first randomly samples an event type from the released initial-event-type distribution. It then constructs a token with the event type and with the remaining two fields, *i.e.,* the interarrival time and the stop flag, set to zero. This is consistent

with model training where the first token always has an interarrival time and stop flag of zero (streams of length 1, *i.e.,* with the first token having a stop flag of 1, are excluded from the training). Finally, CPT-GPT uses the token as the prompt to bootstrap the inference; it runs CPT-GPT inference recursively, predicting the (K+1)-th token in the stream based on previous K tokens, until encountering a token containing a stop flag of 1.

5 EVALUATION

5.1 Evaluation Setup

Baselines. We compare CPT-GPT with both the traditional modeling-based and the state-of-the-art GAN-based approaches, *i.e.,* SMM [45] and NetShare [39]. Considering SMM requires more than 20K Semi-Markov models and 200K CDFs to cover all 24 hours of the day and the three device types, which is extremely cumbersome to derive and use, we also consider a variation of SMM (denoted as SMM-1), which generates a single Semi-Markov model per device type, and the original SMM model is referred to as SMM-20k. We already described in detail how NetShare was adapted to synthesize control-plane traffic in §4.2.1.

Training. We use a dataset collected over seven days in June 2022 as the training set, and use another dataset collected over one day in August 2022 for testing. The trace for the same UE across different days is treated as different UEs, and the 24-hour-long traces are divided into 24 traces of one hour in length each. Furthermore, both NetShare and CPT-GPT generate sequences up to a preconfigured maximum length. To enable a fair comparison between them, we configure and train their models to synthesize streams with a maximum length of 500, disregarding those exceeding this threshold (only 0.07% of the 1-hour flows exceed a length of 500). We tuned the hyperparameters for CPT-GPT to identify the largest model that does not overfit on the testing set with the "phone" device type. The tuning process led to models with 2 attention blocks, each with embedding dimension of 128 and MLP hidden size of 1024, for a total of 725K parameters and a weight size of 2.9 MB.

We initially train a NetShare and a CPT-GPT model from scratch on the subset of the dataset recorded by UEs with the "phone" device type. We then continue training these models using transfer learning (discussed in §4.4) to adapt them to the other two device types, connected cars and tablets.

Inference. The fidelity evaluation involves synthesizing 1000 UE streams for every traffic generator.

Time measurements. To ensure consistent runtime measurements across models, all experiments are run on servers equipped with an NVIDIA A100 GPU, an AMD EPYC 7543 CPU, and 256 GB of memory.

5.2 Overall Results

5.2.1 Stateful Semantic Correctness. We first quantify the semantic correctness of the synthesized dataset as follows. For each synthesized stream, we sequentially replay the events against the UE state machine (Figure 1a). On encountering a state-violating event, a counter is incremented and the state machine stays in the same state. To bootstrap the initial state of the state machine, we employ a heuristic that looks for the first ATCH, DTCH, SRV_REQ, or HO event, as these events lead to a deterministic destination UE state regardless of the source state and thus can be used to determine the subsequent states. Events preceding the state machine bootstrapping are excluded from the semantic correctness calculation.

Table 5 shows the percentage of state-violating events as well as the percentage of streams that contain at least one state-violating event. Since SMM-1 and SMM-20k have state machines built-in, they naturally do not generate any state-violating events and thus their results are not shown. In terms of the percentage of events that violate the stateful semantics, CPT-GPT consistently results in minimal violations, at 0.004%, 0.034%, and 0.079% for the three device types, respectively. This is two order-of-magnitude lower than that of NetShare, which results in 2.614%, 3.915%, and 3.572% events triggering violations.

The percentage of stream violations is naturally higher than event violations for both models, as a stream is deemed to violate the semantics if it contains at least one violating event. However, the stark contrast between NetShare and CPT-GPT remains. NetShare generates 22.1%, 11.5%, and 16.9% streams with violations for the three device types respectively, while CPT-GPT results in much lower violations of 0.2%, 0.4%, and 1.5%. This shows the transformer-based CPT-GPT is highly effective in comprehending and generating control-plane traffic with stateful semantics autonomously, without reliance on domain knowledge as in SMM-1 and SMM-20k, a strength that NetShare fails to achieve.

5.2.2 Distribution Metrics. Next, we compare the fidelity of the synthesized datasets of different approaches in terms of the three distribution-related metrics shown in Table 2, namely, sojourn time distribution, event breakdown, and flow length distribution.

Sojourn time. Table 6 shows the maximum y-distances of the sojourn time CDFs for each synthesized dataset and the real dataset, for both the CONNECTED and IDLE states. First, compared with SMM-20k, CPT-GPT achieves comparable fidelity out of the 6 scenarios (2 states × 3 device types). Specifically, CPT-GPT outperforms in 3 scenarios, for both states with tablet UEs (11.3% v.s. 17.6% for CONNECTED, 11.5% v.s. 15.4$ for IDLE) as well as in the CONNECTED state with phone UEs (6.4% v.s. 14.8%), while achieving lower accuracy in the remaining 3 scenarios. Second, compared to SMM-1 and NetShare, CPT-GPT achieves superior fidelity in the CONNECTED state sojourn time across all 3 device types, with maximum y-distances of 6.4% / 26.4% / 11.3% respectively, significantly lower than SMM-1's 40.1% / 45.1% / 44.0%, and NetShare's 27.9% / 61.7% / 53.6%. For IDLE state sojourn time, CPT-GPT's consistently outperforms SMM-1 across all device types and shows comparable fidelity to NetShare (*i.e.,* better for tablets, equal for phones, and worse for connected cars).

In Figure 5, we show the actual CDFs for different types of UEs. Examining the sojourn time distributions (left two columns), we observe that: (1) Generally, CPT-GPT and SMM-20k are the two most accurate generators, with distributions most closely mirroring that of the real dataset across all device types. (2) SMM-1 is the least accurate generator. For the CONNECTED state, it tends to generate longer-than-realistic sojourn times for phones and connected cars, but shorter-than-realistic times for tablets. For the IDLE state, it tends to generate too many sojourn times that are around 100 seconds for phones and tablets and too many sojourn times of around 200-300 seconds for connected cars. (3) While NetShare

Table 5: Percentage of events and streams that violate the stateful semantics defined by 3GPP standards. The results for SMM are omitted as it does not generate any state violations.

	Phone		Connected Car		Tablets	
	NetShare	CPT-GPT	NetShare	CPT-GPT	NetShare	CPT-GPT
Event violations (%)	2.614%	0.004%	3.915%	0.034%	3.572%	0.079%
Streams w/ at least one violating event (%)	22.1%	0.2%	11.5%	0.4%	16.9%	1.5%

Table 6: Maximum y-distance between the CDFs of the real and synthesized dataset.

		Phone				Connected Car				Tablet			
		SMM-1	SMM-20k	NetShare	Ours	SMM-1	SMM-20k	NetShare	Ours	SMM-1	SMM-20k	NetShare	Ours
Sojourn time	CONNECTED	40.1%	14.8%	27.9%	6.4%	45.1%	16.8%	61.7%	26.4%	44.0%	17.6%	53.6%	11.3%
	IDLE	37.6%	9.6%	12.0%	12.0%	46.8%	14.8%	16.2%	33.3%	35.5%	15.4%	25.7%	11.5%
Flow length	All events	44.2%	1.9%	1.6%	3.8%	54.7%	9.6%	1.4%	4.5%	60.2%	18.7%	3.8%	3.6%
	SRV_REQ	41.9%	3.7%	2.4%	4.3%	55.4%	9.7%	4.0%	5.9%	56.5%	13.1%	4.4%	5.0%
	S1_CONN_REL	43.5%	1.7%	1.5%	4.0%	56.0%	7.1%	3.5%	5.0%	60.0%	18.3%	3.4%	3.5%

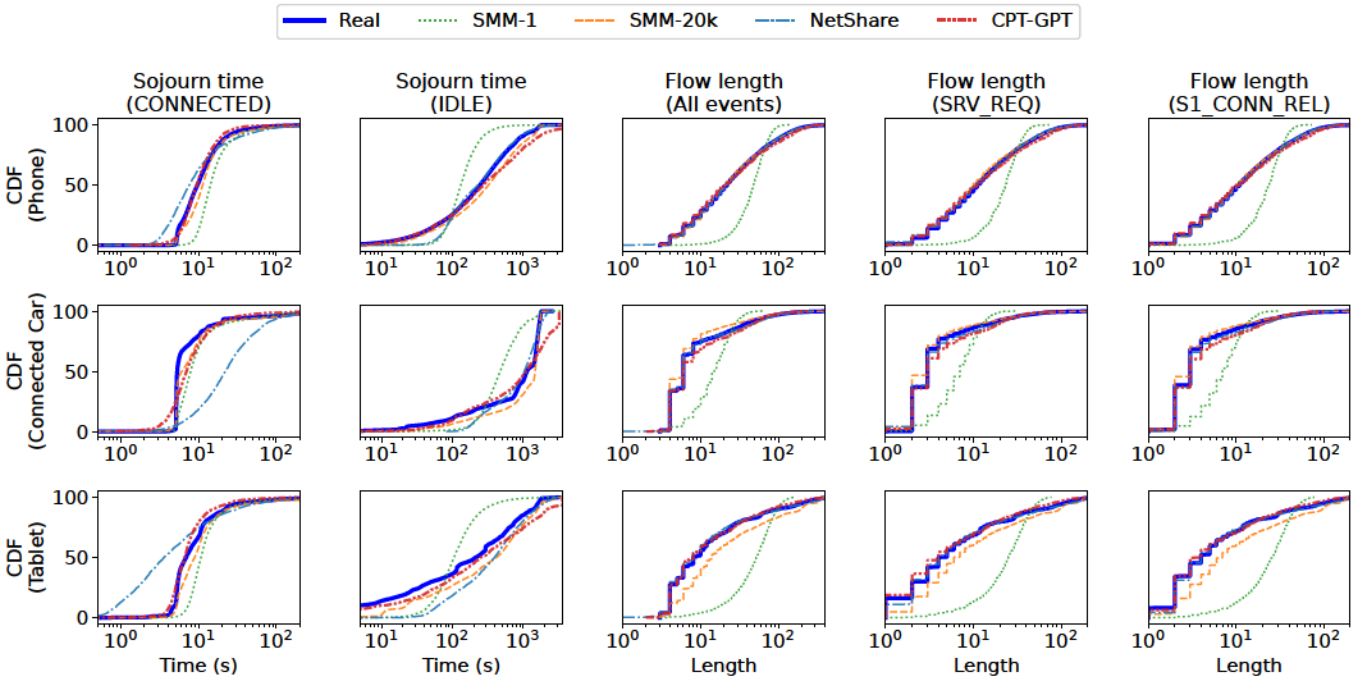


Figure 5: Distributions of fidelity metrics for different types of UEs.

generates IDLE sojourn times as realistic as CPT-GPT, it generates much less accurate CONNECTED state sojourn times, especially for connected cars and tablets, where it tends to generate too long and too short sojourn times, respectively.

In summary, CPT-GPT achieves superior sojourn time fidelity compared to NetShare and SMM-1 in most scenarios – on average, the average max y-distance over the two 3GPP states and three device types is reduced by 24.7% and 16.0%, respectively. More importantly, CPT-GPT achieves comparable accuracy (within 2.0% max y-distance) as SMM-20k.

Flow length. Figure 5 (middle column) shows the distribution of the number of events per stream. We refer readers to Table 6 which shows the maximum y-distances between the CDFs of real and synthesized dataset, with the values presented in percentages, where a smaller difference implies better accuracy in replicating the distribution of the real dataset. In addition to the flow length across all event types, we specifically highlight the length for SRV_REQ and S1_CONN_REL events (*i.e.*, the number of respective events in each stream), the two dominant event types, in the right two columns in Figure 5.

Table 7: Breakdown of event types of real dataset, and of each synthesized dataset shown as difference compared to the real dataset. A lower difference is more accurate.

	Phones					Connected Car					Tablets				
	Real	SMM-1	SMM-20k	NetShare	Ours	Real	SMM-1	SMM-20k	NetShare	Ours	Real	SMM-1	SMM-20k	NetShare	Ours
ATCH	0.12%	-0.01%	0.02%	-0.09%	-0.01%	1.00%	-0.04%	0.14%	-0.61%	-0.12%	1.13%	-0.50%	-0.37%	-0.70%	0.06%
DTCH	0.11%	0.01%	0.04%	-0.05%	-0.02%	0.97%	0.02%	0.21%	-0.53%	-0.18%	1.08%	-0.45%	-0.29%	-0.63%	-0.08%
SRV_REQ	47.06%	0.99%	0.75%	0.28%	0.66%	39.75%	6.11%	5.67%	1.38%	2.15%	44.51%	3.52%	3.03%	2.26%	-3.62%
S1_CONN_REL	48.25%	0.69%	0.64%	0.43%	0.34%	44.14%	3.63%	3.30%	0.08%	0.96%	47.70%	1.22%	1.03%	1.49%	0.03%
HO	2.88%	-1.13%	-1.12%	0.42%	-0.50%	8.59%	-6.17%	-5.92%	1.23%	-1.70%	2.61%	-1.50%	-1.36%	-1.41%	0.00%
TAU	1.59%	-0.56%	-0.35%	-1.00%	-0.47%	5.55%	-3.55%	-3.40%	-1.55%	-1.12%	2.97%	-2.29%	-2.05%	-1.01%	3.61%

For the flow length consisting of all events (see "All events" in Table 6), we observe the following: (1) CPT-GPT and NetShare are generally rank as the top two generators in terms of flow length distribution fidelity, with maximum y-distances of 3.6%–4.5% and 1.4%–3.8% respectively. Although NetShare shows superior fidelity than CPT-GPT on 2 out of 3 device types (phones and connected cars), the differences are marginal (2.2% and 3.1%) when compared with the discrepancies observed with the other generators. (2) SMM-1 produces highly inaccurate flow length distribution, resulting in a maximum y-distance of 44.2%, 54.7%, and 60.2% for the three device types, respectively, showing that a single Semi-Markov model is insufficient in accurately modeling the flow length distributions. The reason for such low fidelity, as shown in Figure 5, is that SMM-1 generates excessive flows of length between approximately 20 to 100 for phones and tablets and 10 to 20 connected cars. (3) SMM-20k, which utilizes over 20k Semi-Markov models, generates much more accurate flow length distributions than SMM-1, with maximum y-distances of 1.9%, 9.6%, and 18.7% respectively. However, while SMM-20k achieves a high fidelity of 1.9% that is comparable to NetShare and CPT-GPT for phones, its accuracy for connected cars and tablets is significantly lower, at 9.6% and 18.7%, respectively. As shown in Figure 5, SMM-20k performs poorly on connected cars and tablets as it tends to generate too short and too long flows for the two device types, respectively.

For the flow length of the two dominating events SRV_REQ and S1_CONN_REL, Table 6 shows that CPT-GPT and NetShare remain as the top two generators, both demonstrating significantly higher fidelity than SMM-1, and for connected cars and tablets, also outperforming SMM-20k. For example, in the case of phones, NetShare produces maximum CDF y-distances of 2.4% and 1.5% for the two event types respectively, and CPT-GPT results in 4.3% and 2.4%. While NetShare is slightly more accurate than CPT-GPT, their difference is marginal when compared SMM-1, which yields maximum y-distances of 41.9% and 43.5%; as Figure 5 shows, SMM-1 fails to learn the distribution of flow lengths and tends to generate too many flows of length 100-200 for phones and tablets, and 50-100 for connected cars. When compared to SMM-20k, CPT-GPT and NetShare again demonstrate comparable accuracy for phones (2.4%/4.3% v.s. 3.7%), and superior accuracy for connected cars (4.0%/5.9% v.s. 9.7%) and tablets (4.4%/5.0% v.s. 13.1%) as Figure 5 shows, SMM-20k tends to generate flows that are too short for connected cars and too long for tablets.

In summary, both CPT-GPT and NetShare demonstrate high fidelity in flow length modeling, significantly outperforming other generators. While NetShare outperforms CPT-GPT in more scenarios, the difference is marginal.

Event breakdown. Table 7 compares the differences in event breakdown between the real and the synthesized datasets of each generator. Compared with SMM-1 and SMM-20k, CPT-GPT achieves comparable or smaller discrepancies across the six event types, *i.e.*, within 0.66%, 2.15%, and 3.62% for the three device types respectively, while SMM-1/SMM-20k has discrepancies up to 1.13%/1.12%, 6.17%/5.92%, and 3.52%/3.03%. When compared to NetShare, CPT-GPT achieves much smaller discrepancies with the real dataset for ATCH and DTCH, within 0.02%, 0.18%, and 0.08%, while NetShare exhibits differences up to 0.09%, 0.53%, and 0.70%, for the three device types respectively. For the remaining event types, CPT-GPT and NetShare achieve similar levels of accuracy.

In summary, in terms of event breakdown, CPT-GPT is capable of synthesizing event distributions with equal or better accuracy than other generators despite not requiring any domain knowledge.

Table 8: Fidelity of CPT-GPT varying the loss weights of different fields, and predicting the interarrival time directly instead of its distribution.

		Ours (1:1:1)	Varying loss weights (event : arrival : stop_flag) 3:1:1	1:3:1	1:1:3	No dist. pred.
Violation	Events	0.04‰	0.04‰	0.20‰	0.48‰	0.10‰
	Streams	0.2%	0.2%	0.8%	0.4%	0.5%
Max y distance	Sojourn (CONN)	6.4%	8.4%	9.1%	6.7%	60.8%
	Sojourn (IDLE)	12.0%	11.8%	9.3%	10.3%	75.4%
	Flow length	3.8%	5.0%	2.4%	3.5%	69.9%
Avg. breakdown diff		0.7%	0.4%	0.2%	0.2%	0.3%

5.3 Sensitivity Analysis and Ablation Study

Varying Loss Weights. Up to now, CPT-GPT is trained to minimize the summation of the losses of the three fields (event type, interarrival time, and stop flag) with equal weights. Next, we study how CPT-GPT reacts to varying weights used in the summation of the total loss. We increase the weight of each field by 3x, while keeping the weights of the other two fields unchanged. For comparison, we show again the results for CPT-GPT trained with equal weights (1:1:1). Table 8 (middle column) shows that varying the weights results in little variations for all fidelity metrics. For example, across all weight combinations, the maximum y-distance of the sojourn time CDFs varies from 6.4% to 9.1% for the CONNECTED

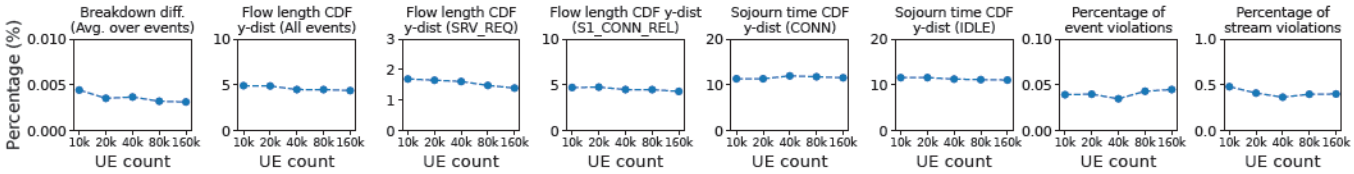


Figure 6: Fidelity of synthesized datasets compared to the real dataset, for varying UE population.

Table 9: Training time in minutes w/ and w/o transfer leraning on an NVIDIA A100 GPU.

		NetShare	CPT-GPT
No transfer learning		108.36	104.40
Transfer learning	First hour	43.08	21.81
	Finetune to each subsequent hour (avg.)	30.41	9.06
	Total	195.12	67.12

Table 10: Maximum CDF y-distances w/ and w/o transfer learning.

		w/o xfer learning		w/ xfer learning	
		NetShare	CPT-GPT	NetShare	CPT-GPT
Violation	Events	2.78%	0.07%	3.39%	0.05%
	Streams	34.58%	0.40%	37.57%	1.00%
Max y distance	Sojourn (CONN)	36.28%	9.39%	13.21%	12.48%
	Sojourn (IDLE)	21.16%	13.40%	28.43%	8.98%
	Flow length	3.30%	7.32%	2.24%	3.08%

state and 9.3% to 12.0% for the IDLE state. This shows the training of CPT-GPT is insensitive to the specific weights chosen.

Disabling the Prediction of Distribution Parameters. We next study CPT-GPT without design insight 2 (predict distribution parameters for numerical fields). To this end, we modify CPT-GPT to output a single scalar representing the interarrival time, instead of predicting its mean and variance. Consequently, during generation, there is no random sampling from the predicted distribution performed. Table 8 (right column) shows that doing so impacts fidelity metrics significantly. While the fidelity in terms of event breakdown remains unaffected, the maximum y-distances of flow length and sojourn time distributions increase sharply. For instance, the maximum y-distance of the flow length CDF increased by 15x, from 3.8% to 69.9%. This shows that predicting the interarrival time distribution parameters is critical in ensuring CPT-GPT's fidelity.

5.4 Scalability Study

To understand whether CPT-GPT is scalable in generating datasets with arbitrary sizes with high fidelity, we run CPT-GPT inference for varying numbers of times to generate synthesized datasets containing 10k, 20k, 40k, 80k, and 160k UEs, respectively. For each synthesized dataset, we compare it with a subset of the real dataset of the same size that is randomly sampled from the full testing dataset, which contains 380k UEs. As shown in Figure 6, the size of the synthesized dataset has minimal influence on all the fidelity metrics. This shows that CPT-GPT is capable of generating datasets of arbitrary sizes with high fidelity.

5.5 Adapting to Data Drifts

Next, we measure the training time needed to train each model to capture data drift across different hours of the day. To synthesize traffic across multiple hours, the operator can either train a single model on the entire dataset spanning over multiple hours, or train multiple specialized models, each tailored to a specific hour. In the latter setup, the operator may train one specialized model from scratch, and subsequently perform transfer learning on it to generate the remaining models. We measure the training time under both setups.

Since the traditional training loss for GAN does not necessarily correlate with the quality of the generated samples [43, 46] [4], we devise the following heuristics to compare the training time of NetShare and CPT-GPT in a fair manner: For each model, we save checkpoints every N epochs, producing K/N checkpoints in total, where K is the model's total number of epochs. For each checkpoint, we generate synthetic datasets and compute the fidelity metrics against the validation set. We then rank the checkpointed models for each fidelity metric, ranging from 1 (best) to N (worst). Finally, we sum the rankings for each checkpoint across fidelity metrics, select the top 20% checkpoints with the smallest ranking sums, and pick the earliest checkpoint among them.

Table 9 shows the training time for a single model on 6-hour traces, or six specialized models through transfer learning. Without transfer learning, NetShare and CPT-GPT require a similar time to train a model on 6-hour traces, taking 108.36 and 104.40 minutes, respectively. Employing transfer learning, CPT-GPT shows significantly reduced training time for adapting a 1-hour model to another hour, requiring only 9.06 minutes, much shorter than NetShare's 30.41 minutes. This difference may be attributed to CPT-GPT's use of supervised training, as opposed to NetShare's GAN training which is known to be difficult to converge [36, 60, 61]. Consequently, it takes merely 67.12 minutes to generate all six CPT-GPT models each for a specific hour. In comparison, NetShare cannot benefit from transfer learning, taking 195.12 minutes in total.

Table 10 shows the fidelity metrics for the trace generated for the 4-th hour, with and without transfer learning. We observe that the use of transfer learning does not have an obvious impact on accuracy for either NetShare or CPT-GPT. For instance, when trained with transfer learning, NetShare shows better CONNECTED state sojourn time and flow length distributions, but worse semantic violations and IDLE state sojourn times. For CPT-GPT, employing transfer learning results in better IDLE state sojourn times and flow

[4] Generator and discriminator losses can oscillate, and a decrease in loss is not a reliable indicator of improved generator performance. To assess whether training has converged, it is more effective to evaluate the models at various epochs throughout the training process.

Table 11: Percentage of n-grams in the generated dataset that repeats from the training dataset.

	ϵ=10%	ϵ=20%
n=5	57.879%	80.305%
n=10	0.003%	0.287%
n=20	0.000%	0.000%

length distributions, but worse CONNECTED state sojourn times and slightly more state violations.

In summary, CPT-GPT demonstrates much higher training efficiency on transfer learning, resulting in substantially reduced training time to synthesize multi-hour traces, with minimal impact on fidelity.

5.6 Data Memorization

Ideally, the generated dataset should closely mirror the real dataset in terms of aggregated statistical behavior, yet maintains diversity, *i.e.,* the generator should not memorize or replicate the training dataset. Achieving this is crucial for ensuring that downstream applications are exposed to a wide variety of traffic behavior, while also preventing leakage of private information.

Methodology. We adopt the methodology from the natural language domain [9, 44, 72] to quantify data memorization in CPT-GPT. Specifically, we extract all n-grams from both the generated and real dataset, where an n-gram is a continuous subsequence of length n. For each n-gram in the generated dataset, we scan through the n-grams in the training set and check for repeats. We report the percentage of n-grams from the generated dataset with at least one repeat found.

Unlike natural languages, cellular control-plane traffic is multi-modal, consisting of categorical (event type) and numerical (interarrival time) fields. We consider two n-grams to repeat if they share the same sequence of event types, and every corresponding pair of interarrival times fall within a relative tolerance ϵ, *i.e.,* $(1-\epsilon) < t_{generated,i}/t_{real,i} < (1+\epsilon)$, for $i = 1, 2, .., n$.

Generally, a larger n reduces the likelihood of repetitions as longer sequences are less likely to repeat, while a larger ϵ increases the likelihood of repetitions. We examine various ranges of n and ϵ.

Results. Table 11 shows the percentage of n-grams in the generated dataset that are repeated from the training dataset, for UEs of phone device type. We make the following observations: (1) Short sequences of length 5 have a high likelihood of repetition. However, such very short repetitions should not be considered memorization, as they are often constrained by control-plane protocols, rather than driven by end user behaviors. For example, HO is always followed by TAO in the CONNECTED state, and consecutively alternating SRV_REQ and S1_CONN_REL are common. (2) Sub-sequences longer than 10 are rarely repeated, even under loose tolerances of $\epsilon = 10\%$ and 20%. For example, with $n = 10$, only 0.003% and 0.287% of n-grams are found to be repetitions under 10% and 20% tolerances; with $n = 20$, no repeating n-grams are found under either tolerance.

In summary, CPT-GPT learns generalized information from the training set, instead of memorizing and repeating any samples from the training set.

6 RELATED WORKS

Traditional Modeling-based Traffic Generators. There have been a large body of work on using statistical models to model and generate traffic, *e.g.,* [56, 59, 63, 67]. However, these works rely on observed statistics from Internet traffic, which do not apply to control-plane traffic of cellular networks [45]. For the control-plane traffic of mobile networks, [16] models the total traffic volume instead of interarrival times and the dependence between events. The state-of-the-art control traffic generator, SMM [45], requires domain knowledge and suffers from its model complexity as discussed in §3.3.

ML-based Generators for Data Traffic. Many works study modeling Internet traffic using ML-based approaches. STAN [70] uses autoregressive neural models to synthesize flow-level traffic but fails to capture fine-grained features such as packet interarrival times. The GAN-based solutions (*e.g.,* [23, 39, 64, 69]) ignore temporal aspects of the traffic and are shown to have suboptimal fidelity compared with NetShare [73]. A few recent works [31, 58] model each pcap flow into a 2D image, and employ stable diffusion models to generate synthetic images and convert them to pacp flows. However, [31] is only capable of generate fixed-length flows, and requires domain knowledge to post-process the generate traffic to maintain high fidelity, whereas [58] can only generate single-modality traffic.

7 CONCLUSIONS AND FUTURE WORK

We proposed a transformer-based traffic generator framework for cellular network control-plane traffic critically needed in research and innovation on mobile core network design and implementation. CPT-GPT is based on a key observation that the generative model requires no domain knowledge and its attention mechanism has the potential to capture complex dependencies among the stream of control events by each UE. Our evaluation shows CPT-GPT synthesizes control-plane traffic with comparable fidelity as prior-art SMM but without domain knowledge, significantly higher fidelity than the state-of-the-art GAN-based scheme in terms of stateful semantics and interarrival time of control events, and does not memorize streams from training traffic.

Due to the lack of systematic support for 5G trace collection, in this paper, we could only use LTE's trace to showcase our transformer-based approach. The versatility of CPT-GPT from not relying on domain knowledge makes it generally applicable to synthesizing control-plane traffic in a wide variety of scenarios. In future work, we plan to evaluate CPT-GPT for next-generation networks (5G, 6G, et al.) and complex scenarios such as mixed LTE/5G networks involving frequent inter-RAT handovers. Additionally, given the coexistence of 4G and 5G in current deployments [32], we intend to evaluate CPT-GPT's performance on 4G/5G co-existing traffic, once such datasets become available. Finally, we plan to evaluate CPT-GPT's effectiveness on downstream applications as such applications become publicly available in the future.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd Chase Jiang for their helpful comments. This work is supported in part by NSF grant CNS-2312834.

REFERENCES

[1] 3GPP. 2016. *Architecture Enhancements for Control and User Plane Separation of EPC Nodes.* Technical Specification (TS) 23.214. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3077

[2] 3GPP. 2016. *General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access.* Technical Specification (TS) 23.401. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=849

[3] 3GPP. 2016. *System Architecture for the 5G System.* Technical Specification (TS) 23.501. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[4] 3GPP. 2024. About 3GPP. https://www.3gpp.org/about-us. (2024).

[5] Anup Agarwal, Zaoxing Liu, and Srinivasan Seshan. 2022. HeteroSketch: Coordinating Network-Wide Monitoring in Heterogeneous and Dynamic Networks. In *Proc. of USENIX NSDI.*

[6] Strategy Analytics. 2021. 5G Signaling and Control Plane Traffic Depends on Service Communications Proxy (SCP). https://carrier.huawei.com/~/media/cnbgv2/download/products/core/strategy-analytics-5g-signaling-en.pdf. (2021).

[7] Gaurav Arora, Afshin Rahimi, and Timothy Baldwin. 2019. Does an LSTM Forget More than a CNN? An Empirical Study of Catastrophic Forgetting in NLP. In *Proc. of the Annual Workshop of the Australasian Language Technology Association.*

[8] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. 2022. Ekya: Continuous Learning of Video Analytics Models on Edge Compute Servers. In *Proc. of USENIX NSDI.*

[9] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying Memorization across Neural Language Models. *arXiv preprint arXiv:2202.07646* (2022).

[10] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding Frequent Items in Data Streams. In *International Colloquium on Automata, Languages, and Programming.* Springer.

[11] Aleksandra Checko, Lars Ellegaard, and Michael Berger. 2012. Capacity Planning for Carrier Ethernet LTE Backhaul Networks. In *Proc. of IEEE WCNC.*

[12] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. 2018. Adversarial Text Generation via Feature-mover's Distance. *Advances in NeurIPS* 31 (2018).

[13] Adriel Cheng. 2019. PAC-GAN: Packet Generation of Network Traffic Using Generative Adversarial Networks. In *Proc. of IEEE IEMCON.*

[14] B. Claise. 2004. Cisco Systems NetFlow Services Export Version 9. (2004).

[15] Graham Cormode. 2008. Count-Min Sketch. *Encyclopedia of Algorithms* (2008).

[16] Dima Dababneh, Marc St-Hilaire, and Christian Makaya. 2015. Data and Control Plane Traffic Modelling for LTE Networks. *Mobile Networks and Applications* (2015).

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv 2020. *arXiv preprint arXiv:2010.11929* (2020).

[18] Ericsson. 2023. free5GC. https://free5gc.org/. (2023).

[19] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. 2017. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv preprint arXiv:1706.02633* (2017).

[20] Andrew E Ferguson, Jon Larrea, and Mahesh K Marina. 2023. CoreKube: An Efficient, Autoscaling and Resilient Mobile Core System. In *Proc. of ACM MobiCom.*

[21] Rodrigo Fonseca, George Porter, Randy H Katz, and Scott Shenker. 2007. X-trace: A Pervasive Network Tracing Framework. In *Proc. of USENIX NSDI.*

[22] Amitabha Ghosh, Andreas Maeder, Matthew Baker, and Devaki Chandramouli. 2019. 5G Evolution: A View on 5G Cellular Technology Beyond 3GPP Release 15. *IEEE access* 7 (2019), 127639–127651.

[23] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved Training of Wasserstein GANs. *Advances in NeurIPS* 30 (2017).

[24] Shaddi Hasan, Amar Padmanabhan, Bruce Davie, Jennifer Rexford, Ulas Kozat, Hunter Gatewood, Shruti Sanadhya, Nick Yurchenko, Tariq Al-Khasib, Oriol Batalla, et al. 2023. Building Flexible,{Low-Cost} Wireless Access Networks With Magma. In *Proc. of USENIX NSDI.*

[25] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2017. Logan: Membership Inference Attacks against Generative Models. *arXiv preprint arXiv:1705.07663* (2017).

[26] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music Transformer. *arXiv preprint arXiv:1809.04281* (2018).

[27] Qun Huang, Xin Jin, Patrick PC Lee, Runhui Li, Lu Tang, Yi-Chao Chen, and Gong Zhang. 2017. Sketchvisor: Robust Network Measurement for Software Packet Processing. In *Proc. of ACM SIGCOMM.*

[28] Shuodi Hui, Huandong Wang, Zhenhua Wang, Xinghao Yang, Zhongjin Liu, Depeng Jin, and Yong Li. 2022. Knowledge Enhanced GAN for IoT Traffic Generation. In *Proc. of ACM WWW.*

[29] Elias Jailani, Muhamad Ibrahim, and Ruhani Ab Rahman. 2012. LTE Speech Traffic Estimation for Network Dimensioning. In *Proc. of IEEE ISWTA.*

[30] Vivek Jain, Hao-Tse Chu, Shixiong Qi, Chia-An Lee, Hung-Cheng Chang, Cheng-Ying Hsieh, KK Ramakrishnan, and Jyh-Cheng Chen. 2022. L25GC: a Low Latency 5G Core Network Based on High-Performance NFV Platforms. In *Proc. of ACM SIGCOMM.*

[31] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. 2024. NetDiffusion: Network Data Augmentation through Protocol-constrained Traffic Generation. *Proc. of ACM SIGMETRICS* (2024).

[32] Rostand A K. Fezeu, Claudio Fiandrino, Eman Ramadan, Jason Carpenter, Lilian Coelho de Freitas, Faaiq Bilal, Wei Ye, Joerg Widmer, Feng Qian, and Zhi-Li Zhang. 2024. Unveiling the 5G Mid-Band Landscape: From Network Deployment to Performance and Application QoE. In *Proc. of ACM SIGCOMM.*

[33] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. 2019. A Comparative Study on Transformer vs RNN in Speech Applications. In *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU).*

[34] Anisia Katinskaia and Roman Yangarber. 2021. Assessing Grammatical Correctness in Language Learning. In *Proc. of the 16th Workshop on Innovative Use of NLP for Building Educational Applications.*

[35] Hyojoon Kim, Joshua Reich, Arpit Gupta, Muhammad Shahbaz, Nick Feamster, and Russ Clark. 2015. Kinetic: Verifiable Dynamic Network Control. In *Proc. of USENIX NSDI.*

[36] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. 2017. On Convergence and Stability of GANs. *arXiv preprint arXiv:1705.07215* (2017).

[37] Xi Li, Umar Toseef, Thushara Weerawardane, Wojciech Bigos, Dominik Dulas, Carmelita Goerg, Andreas Timm-Giel, and Andreas Klug. 2010. Dimensioning of the LTE Access Transport Network for Elastic Internet Traffic. In *Proc. of IEEE WiMob.*

[38] Xi Li, Umar Toseef, Thushara Weerawardane, Wojciech Bigos, Dominik Dulas, Carmelita Goerg, Andreas Timm-Giel, and Andreas Klug. 2010. Dimensioning of the LTE S1 Interface. In *Proc. of IEEE WMNC.*

[39] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *Proc. of ACM IMC.*

[40] Karl Lindberger. 1999. Balancing Quality of Service, Pricing and Utilisation in Multiservice Networks with Stream and Elastic Traffic. *Teletraffic Science and Engineering* (1999).

[41] Zaoxing Liu, Ran Ben-Basat, Gil Einziger, Yaron Kassner, Vladimir Braverman, Roy Friedman, and Vyas Sekar. 2019. Nitrosketch: Robust and General Sketch-Based Monitoring in Software Switches. In *Proc. of ACM SIGCOMM.*

[42] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. 2016. One Sketch to Rule Them All: Rethinking Network Flow Monitoring with Univmon. In *Proc. of ACM SIGCOMM.*

[43] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. 2018. Are GANs Created Equal? A Large-scale Study. *Advances in NeurIPS* 31 (2018).

[44] R Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. 2023. How Much do Language Models Copy from their Training Data? Evaluating Linguistic Novelty in Text Generation using RAVEN. *Transactions of the Association for Computational Linguistics* 11 (2023), 652–670.

[45] Jiayi Meng, Jingqi Huang, Y Charlie Hu, Yaron Koral, Xiaojun Lin, Muhammad Shahbaz, and Abhigyan Sharma. 2023. Modeling and Generating Control-Plane Traffic for Cellular Networks. In *Proc. of ACM IMC.*

[46] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which Training Methods for GANs do actually Converge?. In *International conference on machine learning.* PMLR, 3481–3490.

[47] Meta. 2024. Meta Llama 3. https://llama.meta.com/llama3. (2024).

[48] Nokia Siemens Networks. 2016. Signaling is Growing 50% Faster than Data Traffic. http://goo.gl/uwnRiO. (2016).

[49] Atsuhiro Noguchi and Tatsuya Harada. 2019. Image Generation from Small Datasets via Batch Statistics Adaptation. In *Proc. of IEEE/CVF ICCV.*

[50] Peter Phaal, Sonia Panchen, and Neil McKee. 2001. RFC3176: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. (2001).

[51] Salvatore Pontarelli, Roberto Bifulco, Marco Bonola, Carmelo Cascone, Marco Spaziani, Valerio Bruschi, Davide Sanvito, Giuseppe Siracusano, Antonio Capone, Michio Honda, et al. 2019. Flowblaze: Stateful Packet Processing in Hardware. In *Proc. of USENIX NSDI.*

[52] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving Language Understanding by Generative Pre-training. (2018).

[53] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*

1, 8 (2019), 9.
[54] Markus Ring, Alexander Dallmann, Dieter Landes, and Andreas Hotho. 2017. Ip2vec: Learning Similarities between IP Addresses. In *Proc. of IEEE International Conference on Data Mining Workshops (ICDMW)*.
[55] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. 2019. Flow-Based Network Traffic Generation Using Generative Adversarial Networks. *Computers & Security* 82 (2019), 156–172.
[56] Chloé Rolland, Julien Ridoux, and Bruno Baynat. 2007. LiTGen, a Lightweight Traffic Generator: Application to P2P and Mail Wireless Traffic. In *Proc. of PAM*.
[57] Monika Schak and Alexander Gepperth. 2019. A Study on Catastrophic Forgetting in Deep LSTM Networks. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Deep Learning: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part II 28*.
[58] Nirhoshan Sivaroopan, Dumindu Bandara, Chamara Madarasingha, Guillaume Jourjon, Anura P Jayasumana, and Kanchana Thilakarathna. 2024. NetDiffus: Network Traffic Generation by Diffusion Models through Time-series Imaging. *Computer Networks* 251 (2024), 110616.
[59] Joel Sommers, Hyungsuk Kim, and Paul Barford. 2004. Harpoon: a Flow-level Traffic Generator for Router and Network Tests. *ACM SIGMETRICS Performance Evaluation Review* 32, 1 (2004), 392–392.
[60] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. 2017. Veegan: Reducing Mode Collapse in GANs using Implicit Variational Learning. *Advances in NeurIPS* 30 (2017).
[61] Hoang Thanh-Tung and Truyen Tran. 2020. Catastrophic Forgetting and Mode Collapse in GANs. In *Proc. of IJCNN*.
[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in NeurIPS* 30 (2017).
[63] Kashi Venkatesh Vishwanath and Amin Vahdat. 2009. Swing: Realistic and Responsive Network Traffic Generation. *IEEE/ACM Transactions on Networking* 17, 3 (2009), 712–725.
[64] Pan Wang, Shuhang Li, Feng Ye, Zixuan Wang, and Moxuan Zhang. 2020. PacketCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification using CGAN. In *Proc. of IEEE International Conference on Communications (ICC)*.
[65] Yaxing Wang, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. 2020. Minegan: Wffective Knowledge Transfer from GANs to Target Domains with Few Images. In *Proc. of IEEE/CVF CVPR*. 9332–9341.
[66] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost Van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. 2018. Transferring GANs: Generating Images from Limited Data. In *Proc. of ECCV*.
[67] Michele C Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F Donelson Smith. 2006. Tmix: A Tool for Generating Realistic TCP Application Workloads in ns-2. *ACM SIGCOMM Computer Communication Review* 36, 3 (2006), 65–76.
[68] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A Survey of Transfer Learning. *Journal of Big data* 3, 1 (2016), 1–40.
[69] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular Data using Conditional Gan. *Advances in NeurIPS* 32 (2019).
[70] Shengzhe Xu, Manish Marwah, and Naren Ramakrishnan. 2020. STAN: Synthetic Network Traffic Generation using Autoregressive Neural Models. *arXiv preprint arXiv:2009.12740* (2020).
[71] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. 2018. Elastic Sketch: Adaptive and Fast Network-Wide Measurements. In *Proc. of ACM SIGCOMM*.
[72] Zhou Yang, Zhipeng Zhao, Chenyu Wang, Jieke Shi, Dongsun Kim, Donggyun Han, and David Lo. 2024. Unveiling Memorization in Code Models. In *Proc. of IEEE/ACM ICSE*.
[73] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. 2022. Practical GAN-Based Synthetic IP Header Trace Generation Using Netshare. In *Proc. of ACM SIGCOMM*.
[74] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. 2019. Time-series Generative Adversarial Networks. *Advances in NeurIPS* 32 (2019).
[75] Minlan Yu, Lavanya Jose, and Rui Miao. 2013. Software Defined Traffic Measurement with OpenSketch. In *Proc. of USENIX NSDI*.

APPENDIX

A ETHICS

Although generative ML modes can potentially memorize individual records in training [25], CPT-GPT is trained on LTE data with UE-specific information obfuscated, and therefore both the trace used for training and the trace it synthesizes do not reveal UE-specific information. Hence this work raises no ethical concerns.

B ADDITIONAL DATASET STATISTICS

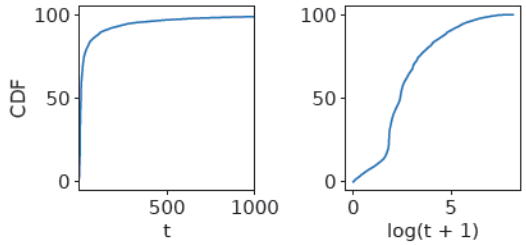


Figure 7: Interarrival time (t, unit in seconds) distribution for UEs of device type phone.

Figure 7 (left figure) shows the interarrival time distribution for UEs of the phone device type. The distribution exhibits a long-tailed pattern, characterized by a higher frequency of short interarrival times and a lower frequency of long ones. Therefore, CPT-GPT applies log transformation to the interarrival times, effectively reducing the impact of the long-tailed distribution, as shown in the right figure.