An Open Dataset for Beyond-5G Data-driven Network Automation Experiments

Dung Chi Phung, Nour-El-Houda Yellas, Salah Bin Ruba, Stefano Secci

An Open Dataset for Beyond-5G Data-driven Network Automation Experiments

Chi-Dung Phung, Nour-El-houda Yellas, Salah Bin Ruba, Stefano Secci
Cnam, Paris, France. {first-name.last-name}@cnam.fr

***Abstract*—In this paper, we present the 5G3E (5G End-to-End Emulation) dataset created to support 5G network automation. The dataset contains thousands of time-series, built at different sampling rates, related to the observation of multiple resources involved in 5G network operation: radio, computing and network resources. The variety of collected features ranges from radio front-end metrics to physical server operating system and network function metrics. We describe the testbed we deployed to support the creation of traffic starting from real traffic traces of a commercial network operator.**

I. Introduction

In networking research in general, and mobile networking in particular, datasets availability plays a critical role in testing and validating new protocols, architectures, algorithms. A real-world dataset collected from a real running system is the ideal data source for validating studies. However, collecting this type of data is challenging due to legal and technical barriers, which is particularly important for recent 5G systems.

Due to difficulties in accessing real-world datasets, simulation/emulation systems are widely used to provide the data to validate the research. Although the data collected from the simulated operation of a networked system may not reflect all the characteristics of a real system, it allows flexible customization of system parameters to generate different datasets at any given time depending on the purpose of the study. As data-driven network architectures for beyond-5G and 6G system are the current front-line of many research projects, making available high-frequency component-level data monitoring information can favor the evaluation and comparison of novel algorithms, protocols and architectures.

In this paper, we document the process of creating a first version[1] of a 5G end-to-end emulated system dataset that we make available to the community working in network automation and control-loop systems [1]. We start by introducing a simulation/emulation hybrid system, using a combination of NS3 [2] network simulation tools with a realistic virtualized network system software stack. Using anonymous user traffic data from a national mobile network provider, we employ NS3 to simulate real-time user (client) data, which is then transferred from the simulated environment to the real network. After crossing the network fabric, this data traffic is ended by a simulation environment, where NS3 is used to simulate the application servers the clients connect to. During these data transfers, we collect information on the overall network system components at different sampling rates, thereby building a dataset composed of thousands of time-series, each related to a specific subsystem and a specific resource type.

Using real user data as input allows us to simulate a system which partially reflects the characteristics of a real system. Moreover, the combination with a simulation tool, allows us to quickly customize the system parameters according to the needs and to scale the experiments with a high number of user equipment (UE).

In Section II, we introduce the platform components. We start with a description of the anonymous mobile traffic dataset, and how we employ it to set up the NS3-based end-to-end connection emulation, simulating the user data closest to the real system data. Then we introduce the real network model, and how the traffic from the simulated client/server environment is fed back into the real network. Section III introduces the scope of the simulation and how the data-set features time-series are collected. The last section anticipates details on the technical demonstration, and draws further platform and dataset versions.

II. Emulation platform

In our platform, we combine both the NS3 simulation tool and a realistic virtualized 5G network stack. Figure 1 depicts the client-server view of the data-plane traffic flow over the platform, where we simulate the application connection handling on both the User Equipment and the server sides. Each UE has one NS3 instance that generates the application traffic for each TCP (Transmission Control Protocol) session from the mobile traffic dataset (described hereafter). The IP traffic from NS3 node is transferred into the UE's network to go through the mobile core network and then to the end-server. On the server site, network traffic is transferred back into NS3 to reach the application server emulated in NS3. Similar behavior happens when traffic is sent back from the server application. In this section, we first introduce the real-world mobile traffic dataset we used. Then we detail how NS3 is configured to simulate the application connections from mobile traffic dataset. Finally, we briefly describe the node-level view of the testbed along with the software components we employed.

A. Mobile traffic dataset

We use a real mobile traffic dataset from a national mobile operator; the data was collected at the 3G and 4G mobile cores (at Gi/Sgi and Gx/Gy interfaces) for each user session during a multi-week period in spring 2019. The data, collected at

[1]We present planned evolutions to appear in future dataset versions.

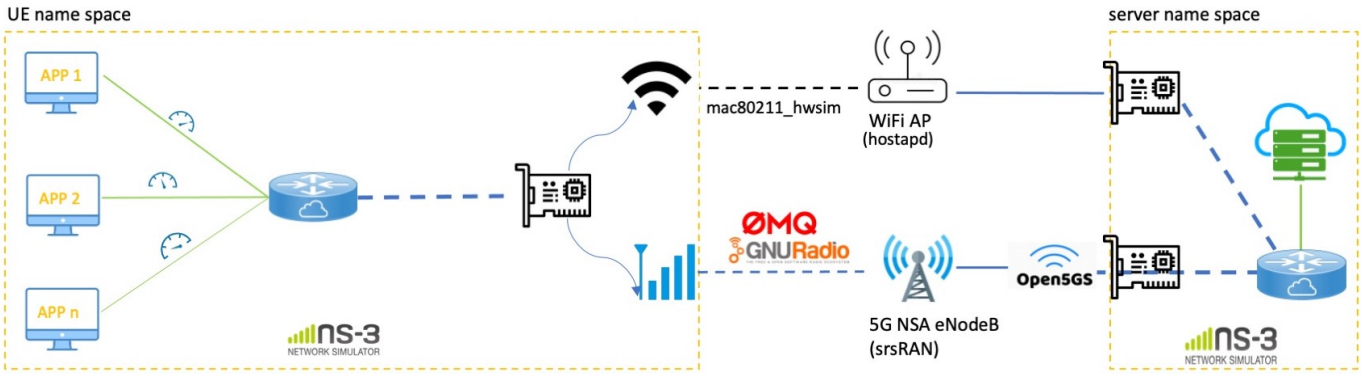

Fig. 1: Hybrid emulation platform: Per connection view.

the GGSN (Gateway GPRS Support Node), allows knowing precisely the category of a mobile service used on top of TCP during an Internet connection (e.g., mail, video streaming, Voice over IP, gaming) and the amount of data consumed for this service, for a single user. The data is organized in TCP sessions with different netflow-like information as below. To reduce the size of the input dataset, TCP sessions of less than 100KB are discarded. For our simulation setting, we exploit the following information on a per-TCP connection basis:

- SRCPORT/DSTPORT: source/destination TCP port;
- START/STOP: Start/Stop time of the TCP session;
- BYTEUP/BYTEDN: total upload/download bytes of the TCP session at IP level;
- NPACKETUP/NPACKETDN: number of packets containing Upstream/Downstream;
- LOCSTART: provides the user's basetation location, the cell is known at the beginning of the TCP session;
- USEFULUP/USEFULDN: useful number of bytes transferred in the TCP session (Up/Down);
- MAXVOLUP/MAXVOLDN: maximum number of bytes sent/received by the client during a period of 20s;
- DELAY_SYN_SYNACK: time in μs between the packet SYN and the response SYN-ACK;
- DELAY_SYNACK_ACK: time in μs between the packet SYN-ACK and the response ACK;
- REMAS: the remote Autonomous System number;
- MAXWINDOWUP/MAXWINDOWDN: maximum value announced for the TCP window, for Upstream /Downstream.

From these fields, we also derive additional parameters such as RTT within the core network, access link bandwidth, data transfer size, and window size for each TCP session.

B. NS3 simulation

Inside each simulated UE, we deploy an NS3 instance, which is responsible for simulating all UE traffic in real-time. A key step in NS3 simulation is to define the network topology. We propose a simulated topology that includes multiple APPLICATION_NODES connected to one gateway. Each node is responsible for simulating the traffic of one TCP session in real-time: the character of the link between each node to the gateway is set based on the TCP session value. On the server side, multiple NS3 instances are used, each instance being responsible for simulating a set of application servers. Also, a virtual tunnel is created to allow traffic from NS3 to be sent/received to/from the UE's network.

The input dataset composed of application-level netflow-like logs allows building statistical distribution for some features. In order to create sets of connections to emulate the real communications, NS3 is carefully configured. We detail next how we set up our NS3 simulation model.

1) Link parameters: We computed them as follows:

- RTT (Round-Trip Time): we use the DELAY_SYNACK_ACK/2 as RTT value.
- Client up/down access links bandwidth: set to k * (ByteUp/ByteDown) / Duration, where the duration is calculated as the time difference between the end and start of the session. k is set such that the number of packets created over the simulated connections is the closest to the real number of packets from the traces (NPACKETDN, NPACKETUP); overall, we found that the value allowing minimizing this gap is $k = 3$ for the given dataset.
- Router queuing: we use the default NS3 link queuing policy at the moment; in future versions, the queue size may be reconfigured as a function of the number of flows to mimic optimal queue sizing in 5G switching systems.

Slicing and related traffic differentiation policies are not integrated for the moment.

2) TCP session parameters: We convert each input TCP session to a TCP session between a client application node and a server application node over the platform. In NS3, we used TCPSOCKET to simulate the socket, and BULKSENDHELPER, PACKETSINKHELPER and THREEGPPHTTPHELPER are used as applications on the CLIENT/SERVER_NODE. The basic parameters are set as follows:

- Source/Destination IP address: we use the subnet 10.0.0.0/8 as IP address space of the client and 172.31.0.0/16 as the server space. The node IP address therefore takes the form of 10.x.y.z or 172.31.y.z with x,y, and z being random decimals from 1 to 255; in further

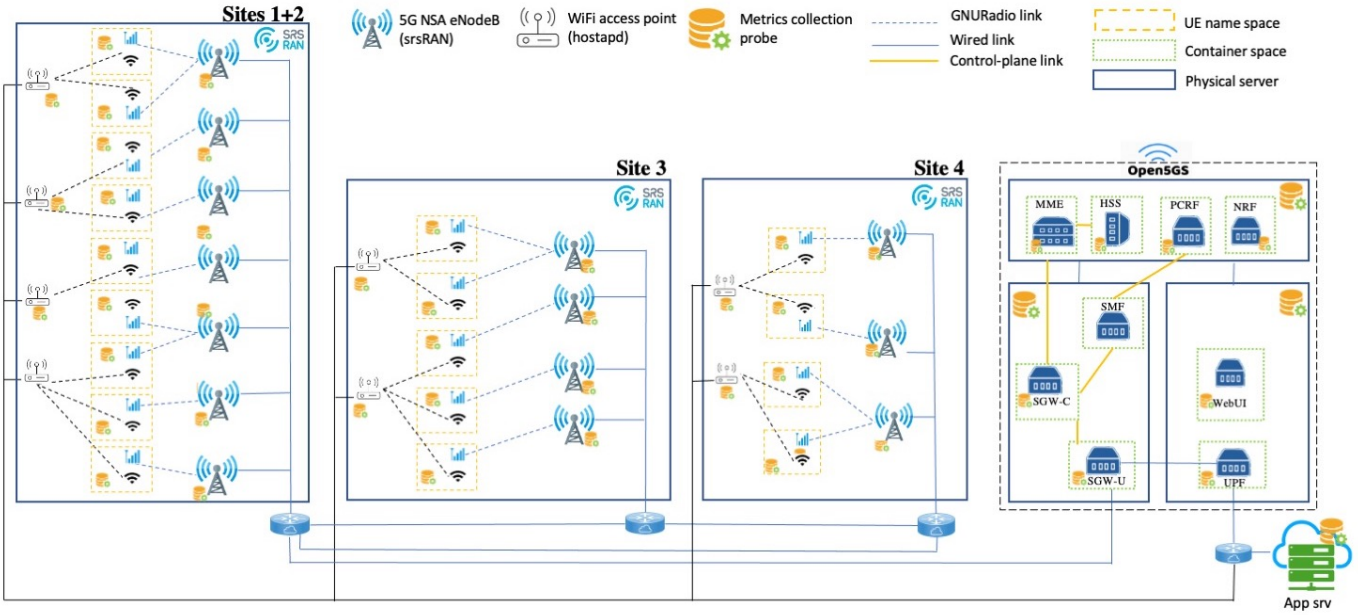

Fig. 2: Hybrid emulation platform: topology.

versions of the dataset, we plan to constrain these ranges to distinct subnets for distinct REMAS.

- Source/Destination port:
  - Source port: set to a random number;
  - Destination port: in the current version, we only use the port 443. We plan to set a port following empirical statistical distribution of the destination TCP ports usage from the dataset in future versions, using the following 99.9% percentile pool: [443, 80, 8080, 993, 8000, 993, 8000, 4070, 5223, 995, 110, 5228, 9339, 143, 587, 1935, 9001, 465, 1090, 25461, 440, 3101, 5222, 732, 3128,9933, 8081, 3724].
- DataSize: bytes sent from the client to the server application, set to max(BYTEUP, BYTEDOWN);
- WindowMax and window size: for the current version of the dataset, they are set as per the default NS3 behavior.

C. 5G RAN and core software stack

As depicted in Figure 1, data traffic quitting the NS3 environment is transmitted over a 5G platform network. In the RAN part, we use srsRAN version 21.10 [3], a 4G/5G software radio suite developed by SRS (Software Radio Systems), to deploy UE and gNodeB. The radio connection between UE and gNodeB is done via GNU Radio, an open-source tool that provides signal processing blocks to implement software-defined radios and signal-processing systems. The wifi interface will be used in further versions of the dataset, in order to test MPTCP/ATSSS (Multipath TCP/Access Traffic Steering, Switching and Splitting) functionalities to allow multipath forwarding across multiple radio access technologies. In the current dataset version, the channel conditions are perfect, but we plan to introduce mobility and channel propagation models in future releases.

The core of the 5G network is deployed through the Open5GS software suites [4] version 2.4.4, related to 5G Non-Stand-Alone (NSA) system; in future releases, we plan to move to a 5G SA system. After their containarization, we use Kubernetes [5], to deploy these basic 5G core functions: HSS, MME, UPF, PCRF, NRF, SGW-C, SGW-U, SMF on three physical servers.

D. System metric collection

To collect data about the overall network activity, metric collectors are installed at multiple levels in the platform. This allows us to collect information from:

- Physical level: using Node exporter [6] on all physical servers allows collecting all information about the server's activity (e.g. CPU, memory, network, storage).
- Container level: using Kubernetes, we collect the activity information of each container, thus getting information about the activity of each component of the 5G core.
- RAN level: information about UE and gNodeB activity can be fully collected using srsRAN.

We set the sampling rate to the smallest appropriate level, which is 150 ms for the physical/container level and 80 ms for the RAN level, hence collecting a maximum amount of system information and avoiding server computing saturation.

III. 5G3E (5G END-TO-END EMULATION) DATASET

From the mobile operator network, we select a small cellular region (TAC, Tracking Area Code). The whole area is covered by 4 towers-sites with 13 cells; the number of users with TCP sessions over 100 kbytes counted up to 52 UEs.

Figure 2 depicts the physical model of the system set-up to emulate the scenario. Three powerful servers connected in a triangle are used to simulate 4 sites. Sites 1 and 2 have total

Node type	CPU	Memory	Network	Disk	Radio	# per node	# nodes	# Total	Sampling rate
Physical	89 - 371	37 - 39	170 - 540	42 - 296	-	340 - 1080	6	3816	150 ms
Container	18	44 - 45	30 - 45	16 - 24	-	147 - 185	9	1433	150 ms
eNodeB	32	2	2	-	-	36	7	252	80ms
UE	32	2	2	-	11	47	35	1645	80ms

TABLE I. Numbers of features time-series of the current version of the dataset [1].

7 cells, while sites 3 and 4 have 3 cells, each serving 5 UEs. Based on the information about cell locations at the start of the session, we assign a corresponding set of TCP sessions to each site. Note that the 5G core is deployed on a clustered system with 4 physical servers, one represents the master node and the three others are used to deploy components of the core network. One physical server is used to simulate the server site; to avoid congestion, the server is not shared between UEs, and each UE has an independent set of servers for download and upload.

A. Training set creation

In order to create a first dataset that can be used for training purposes, i.e. to learn the network normal conditions, the system ran for 14 days of mobile traffic traces, from 16 March, 2019, to 29 March, 2019, which lead to the creation of 1.1 TB of data about system performance. As of our knowledge, no disaster nor major network collapse event happened over this period, hence qualifying the collected data as training set. These data can be considered as data representing the normal operation of the system.

Table I lists the number of distinct features we collect, for different node types, along with the number of nodes per node type. Each feature corresponds to a time-series over the dataset generation period. In the table, the features are grouped into five different groups: CPU, Memory, Network, Disk and Radio. We proceed with this type of grouping having in mind the forthcoming application of anomaly detection framework such as the one in [7], which uses such groups for different autoencoder neural networks.

B. Test set creation

In the second step, we created a test set with the objective to test anomaly detection algorithms such as the one in [7].

We inject some anomalies into the system, while gathering metrics. Currently, we focus on 4 injected anomaly behaviors:

- CPU overload: we use the stress-ng tool to stress CPU in a 5G core function. The load CPU with percent loading in set [10, 20, 30, 40, 50, 60, 70, 80].
- Access bandwidth: set to (k * (byteUp/Down) / duration) * M with k=3 and M in set [2, 3,4,5].
- Packet loss: different rates set on access link using iptables: [10, 20, 30, 40, 50, 60, 70, 80] %.
- Link failure: simulation of links failure between site3/site 4 and site1_2. We use BGP to detect the failures to automatically setup the backup link.

C. Data cleansing and postprocessing

As a final step, we proceed to clean and normalize the data. This data cleaning and normalization operations consist of:

1) Irrelevant data removal: during this step, we remove irrelevant features to infrastructure learning, for example, metrics related to Node exporter.
2) Feature size correction: consists of correcting samples with extra features inserted at specific columns. To do so, we replace each bad line by the previous one. In fact, the number of samples affected by this error is negligible compared to the total size of the data set, roughly 0.014% on average.
3) Features grouping: this step consists of grouping together metrics based on their types and origine, i.e., CPU, Memory, Network, Radio and Disk grouping.
4) Data scaling: the magnitude of numerical values of features is normalized. This is done by transforming the data to fit within the [0,1] scale, which can be useful for some artificial intelligence frameworks.
5) Data reshaping: this step consists of adding the look-back dimension to the data set. It is necessary if we need to perform time-series machine learning training.

D. Technical demonstration

During the demonstration, we show how the system works in a normal situation and some anomaly situations.

ACKNOWLEDGMENT

This work was funded by the ANR CANCAN (https://cancan.roc.cnam.fr; ANR-18-CE25-0011), H2020 AI@EDGE (https://aiatedge.eu; grant nb. 101015922), AMI-5G INFLUENCE and ANR MAESTRO-5G (https://maestro5G.roc.cnam.fr; ANR-18-CE25-0012) projects.

REFERENCES

[1] *5G3E-dataset*. URL: https://github.com/cedric-cnam/5G3E-dataset.

[2] *The NS3 network simulator*. URL: https://www.nsnam.org.

[3] *Open source SDR 4G/5G software suite from Software Radio Systems (SRS)*. URL: https://www.srslte.com.

[4] *Open source project of 5GC and EPC*. URL: https://open5gs.org.

[5] *Kubernetes*. URL: http://kubernetes.io.

[6] *Node exporter - Exporter for machine metrics*. URL: https://prometheus.io.

[7] Alessio Diamanti, José Manuel Sánchez Vílchez, and Stefano Secci. "An AI-empowered framework for cross-layer softwarized infrastructure state assessment". In: *IEEE Transactions on Network and Service Management* (2022), pp. 1–1. DOI: 10.1109/TNSM.2022.3161872.