(DEMO) EDICT: A simulation tool for performance metrics datasets in IoT environments

Houssam Hajj Hassan, Georgios Bouloukakis, Denis Conan, Ajay Kattepur, Mahdi Trabolsi, Nikolaos Papadakis, Djamel Belaïd, Kostas Magoutis

(DEMO) EDICT: A Simulation Tool for Performance Metrics Datasets in IoT Environments

Houssam Hajj Hassan[1], Georgios Bouloukakis[1], Denis Conan[1], Ajay Kattepur[2], Mahdi Trabolsi[1, 3], Nikolaos Papadakis[1, 4], Djamel Belaïd[1], Kostas Magoutis[4, 5]
[1]*SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, France*
{houssam.hajj_hassan, georgios.bouloukakis, denis.conan, djamel.belaid}@telecom-sudparis.eu
[2]*Ericsson AI Research, India*, ajay.kattepur@ericsson.com
[3]*Université Jean Monnet, Sainte-Étienne, France*, mahdi.trabolsi@etu.univ-st-etienne.fr
[4]*Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH)*,
{papadakni, magoutis}@ics.forth.gr
[5]*Computer Science Department, University of Crete, Greece*

***Abstract*—This paper demonstrates EDICT, a simulation tool for performance metrics datasets in IoT environments. Such environments are represented using the NGSI-LD data model. EDICT uses NGSI-LD instances and Open Queueing Networks for the composition of a concrete QoS model that is then simulated to generate a performance metrics dataset. This dataset captures performance metrics of Edge interactions such as response time and throughput. We demonstrate the utility of EDICT by showing how a perfomance metrics dataset can be generated for a specific IoT environment.**

***Index Terms*—Simulation, IoT, Edge Environments, Datasets.**

I. EDICT Architecture

With the recent advances in Artificial Intelligence (AI) and Machine Learning (ML) techniques to provide IoT-based solutions, there is a growing need for datasets that capture the performance of data flows in smart environments. EDICT [1] is a simulation tool for generating performance metrics datasets of IoT enviroments that aims to facilitate IoT system tuning tasks. This paper demonstrates how EDICT can be used for such purposes. Fig. 1 shows the high-level architecture of EDICT. We briefly describe next the main components of this architecture.

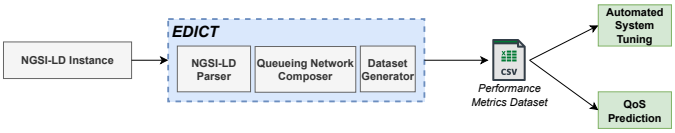

Fig. 1: EDICT high-level architecture

A. IoT System Representation

We leverage the NGSI-LD [2] specification to represent characteristics of smart environments. We propose an *NGSI-Model* to represent IoT interactions in smart environments by extending existing models and enriching them with new *entities* and *relationships*. The model captures information related to the devices and applications deployed in the smart environment, as well as the QoS requirements that applications define. A description of our proposed NGSI-LD representation of smart environments is available in [1].

B. Queueing Network Composition

We rely on Open Queueing Networks to create a generic *QoS Model* that represents application-layer interactions in IoT environments. EDICT takes as input the NGSI-LD representation of a smart environment and parses it to extract the information needed to instantiate the generic QoS Model and compose the corresponding *Queueing Network* that evaluates Edge interactions in the provided environment. This is possible through API calls to the open-source Java Modelling Tools (JMT)[1] queueing simulator. EDICT composes and simulates the JMT queueing network to generate a dataset containing performance metrics of data flows in the IoT-enhanced environment. A more detailed description about the queueing network composition process is provided in [3].

C. Performance Metrics Dataset Generation

After simulating the composed JMT queueing network, EDICT extracts the simulation results from the JMT `jsimg` files and creates a performance metrics dataset as a `csv` file. The generated dataset includes metrics related to response time, throughput, and data drop rate for each data flow in the simulated smart environment. In addition, the file contains the configuration parameters of the data exchange system. These parameters include the priorities and data drop rate set for each application category, as well as the available network resources and the network allocation policy used. These *features* facilitate the use of the dataset for automated system tuning approaches [4]. We demonstrate how the dataset can be readily used for QoS prediction in [1].

II. Using EDICT

In this section, we demonstrate how EDICT can be used by IoT systems designers to generate a performance metrics dataset of Edge interactions in smart environments. For the sake of example, we consider an IoT system consisting of 30 devices that capture 30 different observations, with 16 applications to receive the observations generated by the IoT

[1]https://jmt.sourceforge.net/

devices. The deployed applications belong to four application categories with different QoS requirements: analytics (AN), real-time (RT), transactional (TS), and video streaming (VS). The EDICT code can be downloaded at https://github.com/SAMSGBLab/edict.

A. *Defining the IoT System Components*

As a first step, IoT designers need to define the components of their IoT system infrastructure. For this purpose, EDICT provides a Graphical User Interface (GUI) to add, edit, and delete such components. The drag and drop interface (Fig. 2) allows designers to add new devices, and edit and delete existing ones. For each device, designers have the option to specify the size of messages generated by the device, the frequency at which messages are generated, a probability distribution based on which messages are generated, and the observations that the device captures.

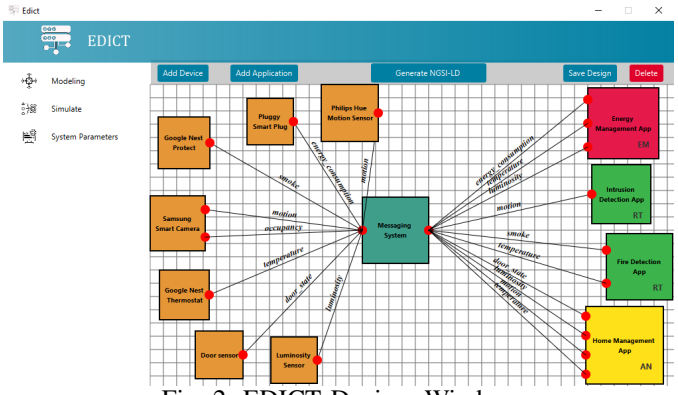

Fig. 2: EDICT Devices Window

Next,designers can add new applications and editing their properties (the application category they belong to, the observations they receive, and the rate at which they process messages). This is where designers also have the option to define specific priorities for applications. In a similar fashion, designers can define properties of application categories, QoS requirements, and observations in dedicated interfaces.

Once all IoT system components are defined, EDICT generates the JSON-LD files corresponding to these components according to the NGSI-LD model presented in [1]. The NGSI-LD context used in all JSON-Ld files is https://raw.githubusercontent.com/SAMSGBLab/edict--datamodels/main/context.jsonld. Listing 1 shows how an IoT device is defined in JSON-LD format. Each device has a `name` and is identified by a unique URN (`id`). A device sends messages with a `messageSize` at a specific `publishFrequency`. In addition, we consider that devices generate messages based on a probability distribution (`dataDistribution`). A device captures one or more observation; this is represented in the `capturesObservation` *relationship*.

```
1  "id": "urn:ngsi-ld:edict:Device:dcbd9ee7-...",
2     "type": "Device",
3     "name": "device 1",
4     "publishFrequency": 150,
5     "messageSize": 2000,
6     "dataDistribution": "exponential",
7     "capturesObservation": ["urn:ngsi-ld:edict:
       Observation:85c38830-...", ... ]},
8     ...
```

Listing 1: IoT device definition

Similarly, the definition of applications deployed in the smart space is shown in Listing 2. Each application belongs to an `applicationCategory` and is assigned a `priority`. Note that a lower integer represents a higher priority. Applications receive one or more observation, which are defined as a list in the `receivesObservation` field. In addition, applications process the received messages at a specific rate (`processingRate`), and following an exponential probability distribution (`processingDistribution`).

```
1  "id": "urn:ngsi-ld:edict:Application:a3b6dc85-...",
2     "type": "Application",
3     "name": "app 1",
4     "applicationCategory": "AN",
5     "priority": 0,
6     "processingRate": 1000,
7     "processingDistribution": "exponential",
8     "receivesObservation": [
9             "urn:ngsi-ld:edict:Observation:01437e79-...",
10            "urn:ngsi-ld:edict:Observation:58f0581d-...",
        ... ],
11    ...
```

Listing 2: Application definition

Each application deployed belongs to a specific category (Listing 3).

```
1  "id": "urn:ngsi-ld:edict:ApplicationCategory:b340208c
       -...",
2     "type": "ApplicationCategory",
3     "name": "analytics",
4      ...
```

Listing 3: Application category definition

Finally, QoS requirements are defined in terms of maximum response time (in seconds), minimum throughput (in Kbps), and maximum drop rate (Listing 4).

```
1     {"id": "urn:ngsi-ld:edict:QosRequirement:9ffd2f4b
       -...",
2     "type": "QosRequirement",
3     "name": "realtime requirements",
4     "maxResponseTime": 0.4,
5     "minThroughput": 28.2,
6     "maxDropRate": 0.02,
7     ...
```

Listing 4: QoS requirements definition

Note that IoT designers that are familiar with the NGSI-LD data model and that already have a representation of their IoT system in the JSON-LD notation may readily upload their files to EDICT. We provide the complete JSON-LD files for the aforementioned setup in https://github.com/SAMSGBLab/edict--datamodels.

B. *Setting the Edge Infrastructure Configuration Parameters*

After defining the components of their IoT system infrastructure, designers can now specify the configuration parameters of their systems, as shown in Fig. 3. Designers can specify the available network resources (i.e., the available bandwidth between the data exchange system and the applications), and

Subscription Flow	Category	Priority AN	Priority RT	Priority TS	Priority VS	Dropping AN	Dropping RT	Dropping TS	Dropping VS	Network policy	Network resources	Response time	Throughput	...
amazonecho/app14	TS	0	0	0	0	0	0	0	0	default	650	1.694455	...	
amazonecho/app14	TS	0	0	1	0	0	0	0	0	maxmin	650	0.121488	...	
intrusion/app12	RT	0	0	0	0	5	0	0	2	shared	650	1.232349	...	
energymanagement/app1	AN	0	3	1	2	0	0	0	0	default	650	3.466959	...	
...	...	...	...	...	....	...	...							

TABLE I: Output Dataset Format

the network resource allocation policy to be used. Currently, EDICT supports three network allocation policies: (i) the *default* policy, where all network resources are used to forward all data flows, (ii) the *shared* policy, where network resources are equally shared between the application categories defined, and (iii) the *max-min* policy, which shares network resources among categories based on the max-min resource allocation policy. Moreover, the configuration parameters include setting drop rates for application categories, and defining the capacity (in number of messages) of the data exchange system. Designers can then define some simulation settings: the simulation duration, an alias to be used for saving the simulations results, and a global message size to be used when running the simulation.

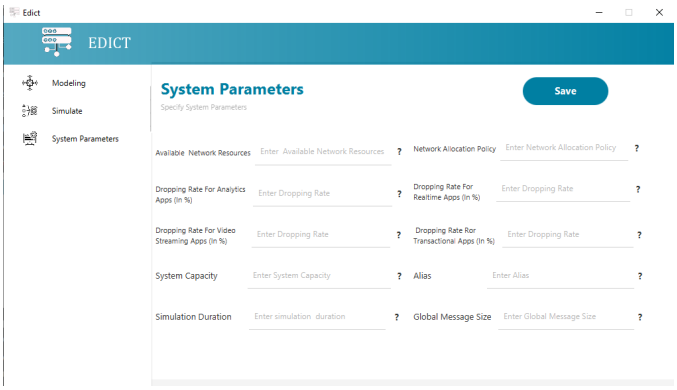

Fig. 3: EDICT Parameters Window

C. Generating a Performance Metrics Dataset

Once the configuration parameters are set, EDICT is ready to generate the performance metrics dataset. Through calls to the JMT library, EDICT composes and simulates the JMT queueing network (Fig. 4). Then, EDICT generates a dataset as a `csv` file consisting of performance metrics of the simulated system. IoT designers can also test different configuration parameters for the same system to evaluate the performance of their systems under different situations (e.g., using different network resource allocation policies, different drop rates). This enables creating a richer dataset, where each row contains the metrics for a flow matching a subscription under different configurations parameters. As depicted in Table I, metrics are generated for each flow matching a subscription, and when applicable, for the whole system. For example, EDICT stores end-to-end latency per flow and the utilization of the data exchange system under all configurations specified. In Table I, each row contains the metrics for a flow matching a subscription under different configurations parameters. For example, if the developer chooses to simulate an IoT system using the default network allocation policy as well as the max-min policy, a subscription flow would have two entries in the dataset that represent its metrics under these two policies. Note that the columns in this `csv` file are *features* that impact the performance of the IoT system (i.e., response time of data flows). Hence, as shown in [1], this dataset format enables applying ML models directly for system tuning purposes (e.g., runtime QoS prediction).

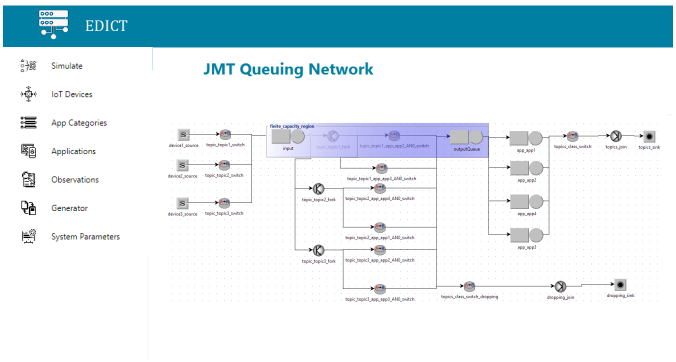

Fig. 4: JMT Composed Queueing Network

III. Conclusion

This paper shows how to use EDICT, a tool for generating performance metrics datasets in IoT environments. Through standard NGSI-LD data modeling and queueing network composition, EDICT is able to generate a metrics dataset that captures the performance of IoT data flows in smart environments. The dataset generated by EDICT can be integrated in various IoT system tuning approaches such as adaptive data flow management and runtime QoS prediction.

References

[1] H. Hajj Hassan, G. Bouloukakis, A. Kattepur, D. Conan, and D. Belaïd, "EDICT: Simulation of Edge Interactions across IoT-enhanced Environments," in *IEEE/ACM International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, Paphos, Cyprus, 2023.

[2] "Context Information Management (CIM) NGSI-LD API V1.4.2," https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.04.02_60/gs_cim009v010402p.pdf, 04 2021.

[3] H. Hajj Hassan, G. Bouloukakis, A. Kattepur, D. Conan, and D. Belaïd, "EDICT: Simulation of Edge Interactions across IoT-enhanced Environments," Télécom SudParis, Tech. Rep., 2023. [Online]. Available: https://hal.science/hal-04078497

[4] ——, "PlanIoT: A Framework for Adaptive Data Flow Management in IoT-enhanced Spaces," in *IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, Melbourne, Australia, 2023.