Original scientific paper

Simulation Environment for Scalability and Performance Analysis in Hierarchically Organized IoT Systems

Haris Turkmanović, Ivan Popović, Zoran Čiča, and Dejan Drajić, *Senior Member, IEEE*

***Abstract* — The accelerated development of technologies, especially in the field of telecommunications, ease the integration of embedded devices within various IoT applications. Modern IoT applications assume heterogenous embedded platforms capable of collecting, processing, and exchanging data between the tiers of the IoT system architecture. Designing a multi-tier IoT system, even in the case of architecture that involves a small number of intelligent embedded devices, can be a very demanding process, especially when dealing with the strict requirements of IoT application concerning application performance, scalability, and energy consumption. In this paper, an open-source simulation framework for the performance analysis of an arbitrary multi-tiered IoT system is presented. Framework supports insight into the data availability within the tiers of IoT system enabling designers to evaluate the performance of IoT application and to engineer the system operation and deployment. Besides the performance analysis, proposed framework enables the analysis of energy consumption, architecture scalability utilizing different communication patterns and technologies. The case study of a large-scale IoT application for demonstrating the framework potential regarding the scalability and data availability analysis is also given.**

***Keywords* — IoT, Simulation framework, Large Scale, Scalability.**

I. Introduction

In recent years we have witnessed the increasing prevalence and use of various IoT systems within different application areas (automotive, health, smart homes, smart cities, smart industry, etc.). An IoT system includes devices that produce data, mostly by sensing their environment, and exchange generated data with each other or with other parts of the IoT system through an internet network. Modern IoT applications are characterized by the constant trend of increasing the number of devices, and it is assumed that this trend will be kept in the future [1]. Due to the increase in the number of devices, there is a need to design an optimal architecture of modern IoT applications that is, among other things, able to meet the requirements in the field of storage and processing of a huge amount of data.

Designing the architecture of today's modern IoT applications is a very serious challenge. It is not only important to create an architecture that will enable the storage and processing of a huge amount of data, since it is also important to select appropriate communication technologies and available protocols which optimally satisfy application requirements such as scalability, real-time performance, efficient use of available energy, and many others.

For a long time, the main architectural model in IoT application was given in the form of centralized computing and storage power. Because of limited storage and processing power of the end IoT sensing devices, data generated by these devices were sent to the central server for further processing. For IoT applications that are not demanding in terms of data latency, this architectural model is still applied today [2]. However, the current infrastructure of the Cloud is not designed to achieve the real-time performance that is required by the most modern IoT applications. Also, because this is a centralized computing model, there is an issue related to scalability. Therefore, over time, a novel computing model was developed with the idea to bring storage and processing power closer to the IoT end devices to reduce the data flow to the central server [3][4]. In the Fog architectural model, the topology can be hierarchical, which means the data from the end IoT devices can be processed at several tiers; each tier carrying out specific data filtering and analytics and deciding whether to pass to the next tier for further Fog/Cloud processing [5], [6]. In this case, it is possible to reduce the amount of data exchanged between tiers closer to the centralized services and overall system scalability is increased [7]. Until now, there have been several multi-tiered Fog model proposals which address different requirements of IoT applications [8], [9].

To create architecture that will satisfy strict IoT application requirements, it is necessary to have an insight

Paper received May 25, 2022; revised October; accepted October 29, 2022. Date of publication December 26, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Miroslav Lutovac.

This paper is revised and expanded version of the paper presented at the 29th Telecommunications Forum TELFOR 2021 [28].

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

Corresponding author is Haris Turkmanović from the School of Electrical Engineering, University of Belgrade, Serbia (e-mail: haris@etf.bg.ac.rs).

Ivan Popović is with the School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: popovici@etf.bg.ac.rs).

Zoran Čiča is with the School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: cicasyl@etf.bg.ac.rs).

Dejan Drajić is with the School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia and with Innovation Center, School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: ddrajic@etf.bg.ac.rs).

into the behaviour of the system, especially in the data flow through the IoT system, from the moment when data are generated on the IoT sensing devices until the data are processed on each layer, in case of a multi-tiered architectural model, or at the core services, in case of Cloud processing model. This approach is very important in the design phase before an application is deployed in the real environment, to better understand system behaviour but also to choose the best communication technologies and communication models which will be used within the selected architecture model [10].

Insight into the system behaviour can be achieved in two ways: by using real hardware and measurement equipment – which is in most cases very impractical [11] or by creating, or using already developed, simulation frameworks – this is a very efficient approach that speeds up the development of IoT architecture especially in case of large scale IoT applications [12], [13].

Simulators allow researchers to provide a proof-of-concept for new solutions in a virtual environment by avoiding time-consuming, heavyweight, or expensive real-world experimentation [14]. Until now, different simulation frameworks have been developed to ease the process of designing IoT applications [15]. They can be classified into three categories: full-stack simulators, big data processing simulators, and network simulators [12]. Full-stack simulators, such as DPWSim [16] and iFogSim [17], provide a wide range of models for different IoT components but they are not capable of simulating scenarios that include the flow of a huge amount of data within IoT system. Big data processing simulators, such as IOTSim [18], focus on the big data processing within IoT applications but they do not enable precise implementation of IoT end devices models engaged in IoT architecture. Network simulators, such as NS3 [19] and CubCarbon [20], provide an insight into IoT behaviour from a functional aspect but they do not provide support for a detailed energy and scalability analysis of arbitrary IoT architectures.

This paper presents a simulation framework that enables the simulation of data flow through any IoT application. It enables the easy development of large-scale distributed IoT applications. Also, it is possible to easily create an architecture for any IoT application, even for the application that deals with a huge number of IoT end devices. The developed framework enables the analysis and quantification of different IoT application parameters such as real-time performance parameters, IoT device consumption parameters, scalability of the architecture, etc. It is also possible to have an insight into data availability at any moment at any part of the IoT system architecture. This framework was originally presented in [26] while this paper presents a more detailed description of simulation framework core's functionalities and also demonstrates framework overall capabilities in the field of IoT system scalability, energy and real-time analysis.

The rest of the paper is organized as follows: Section II presents the developed simulator and describes the used model. In section III, on the example of multi-tiered IoT application, it is demonstrated how to quantify scalability of the architecture by using a developed simulation framework. Additionally, data availability time distribution is presented and analysed. In section IV, a conclusion is given with future work directions.

II. Simulation Framework Description

The developed simulation framework enables the creation of arbitrary hierarchically organized architectures for a wide range of IoT applications. Within the framework, general models of components have been developed that makeup one IoT architecture in the general case. Using these models, it is possible to create an architecture for any IoT application. Within the first part of this section, the available general models, as well as their parameters, are described. In the second part of this section, the general technical characteristics of the developed simulator are presented.

III. Models used within Simulation Framework

To enable the simple creation of the IoT application architecture, and to examine the scalability properties of the designed architecture, a logical decomposition of the IoT architecture into basic building components is performed in this research. From a high-level perspective, most IoT architectures consist of devices that generate data and/or perform data processing, links that connect these devices, and protocols used to encapsulate the generated data and send it to the destination [21], [22]. According to this classification the following models are introduced in the simulator: node model, link model, and protocol model. Within all these models, it is possible to configure certain parameters to make the best representation of desired IoT application architecture.

The Node model is introduced to represent devices within the IoT application that can produce and/or process or consume data. For example, in IoT applications, this model represents an IoT sensing node or smart network gateway. Two types of node models are supported within the simulation framework: producer node model and consumer node model. Consumer node model represents the device that is only capable of receiving, processing and optionally transmitting data to other nodes (if it is not the final node on the data path). For example, in real IoT applications, the consumer node model represents gateways that have capabilities to implement some data processing algorithm (for example, algorithms for data aggregation [23]). The producer node model represents a device, which is also capable of implementing simple data processing, but the main role of this model is to represent IoT devices that generate data with some predefined frequency. The list of parameters for consumer node and for producer node that can be configured within is presented in Table 1.

The Link model enables the simulation of communication between architecture components, and it presents a high perspective representation of real physical communication links. Parameters which can be configured within this model are listed in Table 2. These parameters define communication performance based on knowledge of the communication model and communication

technology. They refer to physical layers, data links, network layer, application layer, etc.

The protocol model defines a communication protocol between a producer and consumer device. The configurable parameters within this model are presented in Table 3.

TABLE 1 - LIST OF THE CONFIGURABLE PARAMETERS FOR NODE MODEL

Param. Name	Description	Supported by node model	
		Producer	*Consumer*
PS - *Processing speed [B/R]*	Defines data processing speed in bytes per time resolution	Yes	Yes
CL - *Compression Level*	Defines ratio for data reduction before they are sent from node	Yes	Yes
AL - *Aggregation Level*	Define how many data packets are stored on the node before they are sent	Yes	Yes
DPC - *Data processing consumption [mA]*	Current consumption when device is active	Yes	Yes
LPC - *Low-power mode consumption [mA]*	Current consumption when device is in low-power mode	Yes	Yes
SR - *Sampling rate [R]*	Data generation frequency	Yes	No
DS - *Data size [B]*	Size of generated data	Yes	No

TABLE 2 - LIST OF CONFIGURABLE PARAMETERS FOR LINK MODEL

Parameter name	Description
S - *Speed [B/R]*	Link speed. Unit is bytes per time resolution
MTU - *MTU Size[B]*	Size of the link MTU
TCT - *Transceiver Consumption during Transmit [mA]*	Consumption when transmitting data from node
TCR - *Transceiver Consumption during Receive [mA]*	Consumption when receiving data on the node

TABLE 3 - LIST OF CONFIGURABLE PARAMETERS FOR PROTOCOL MODEL

Parameter name	Description
O - *Overhead size [B]*	Size of the overhead introduced in data processing
ARQ - *ARQ handshaking*	Defines whether response data packet will be generated when request packet is delivered on final node.

IV. TECHNICAL DESCRIPTION OF THE SIMULATOR

The developed simulation environment consists of two parts: GUI part [24] and the core application part [25] and results parser part. GUI part is written in the C# programming language, and it gives a user the ability to easily create the architecture of large-scale IoT applications. An overview of one GUI part, responsible for creation of the nodes, is presented in Fig. 1.

Thanks to the developed GUI, users can easily create all nodes, links, and protocols. It is also possible to make the connection between the nodes by using the previously created link and it is possible to assign protocols to certain data which are produced on IoT end nodes and it is very easy to configure all model parameters listed in Tables 1-3. When an IoT application architecture is created, configuration files are produced by the GUI. These files are used by the core of the simulation framework.

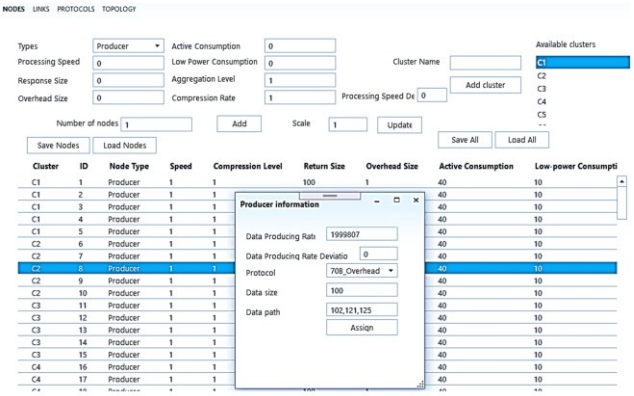

Fig. 1. Simulator GUI.

The core application of the simulation environment is written in the C programming language because it is time efficient. Especially, this is important in the case of large-scale IoT applications where the number of nodes that are involved in simulation, can be very huge and simulation may execute for a very long time. This core application as simulation input parameters takes configuration files produced by the GUI application. When a simulation is done, results are written in two types of log files: ***Node log file*** and ***System log file***.

For each node in the system, a separate *Node log file* is created. This file contains information like when data is created (in case of producer node), when data is received, when data processing is started, when data processing is ended and others. An example of one *Node log file* is presented in Fig. 2.

```
1000.000000,LP,EXIT,10.000000,10000.000000
1000.000000,A,100,C_Req,0,100,0,0,0,0,30.000000,10000.000000
1000.000000,A,100,P_D_S_Req,0,100,0,0,0,0,30.000000,10000.000000
1000.100000,A,100,P_D_E_Req,0,100,0,100,0,1,30.000000,10003.000000
1000.100000,LP,ENTER,10.000000,10003.000000
2000.000000,LP,EXIT,10.000000,20002.000000
2000.000000,A,100,C_Req,1,100,0,100,0,1,30.000000,20002.000000
2000.000000,A,100,P_D_S_Req,1,100,0,100,0,1,30.000000,20002.000000
2000.100000,A,100,P_D_E_Req,1,100,0,100,0,1,30.000000,20005.000000
2000.100000,LP,ENTER,10.000000,20005.000000
3000.000000,LP,EXIT,10.000000,30004.000000
```

Fig. 2. Part of the node log file.

Each action on the node is sampled to one line in *Node log file*. There are two types of log lines in *Node log file*: **LP**- lines and **A**-lines. **LP**-line gives information about node status while node is in a *Low-Power* mode. **A**-line gives information about current data processing status and node status when node is in an active state.

The first column for both log line types is the same and presents a timestamp of the action. Node log line type is written in the second column. In case of LP-line type, the third column marks whether a node exits from Low Power mode (EXIT), or a node enters a LP state (ENTER). The fourth column presents node consumption in a LP mode while the fifth column presents how much energy node consumes during Low-Power. In case of A-line the third column presents size of data that currently is processed on the node. The fourth column represents data state (request/response data created, processing data, processing data overhead, send data, receive data). Data ID is presented in the fifth column. Based on data ID it is possible to track specific data over network. From column 6 to column 10 is listed additional node information related to data processing. Column 11 presents node

consumption during an active state while column 12 presents energy consumption between a current and previous action on the node.

One general simulation log file, *System log file*, is created and it shows when data are created and when data have arrived at the destination node. An overall system performance related to the data availability across the IoT system can be examined by analysing only this file.

By examining these two types of created log files, it is possible to get an insight into data flow across the network and it is also possible to analyse different IoT application performances of interest such as nodes consumption, scalability, average data delivery time, real-time performance, etc.

V. USE CASE

As already mentioned, the developed simulation environment enables the analysis and quantification of the influence of different architecture parameters on the performance of IoT applications. In [26], we have presented the potential of developed simulation framework in the case of energy consumption analyses on end IoT nodes.

The architecture of the multi-tiered IoT system, used in our example, is shown in Fig. 3. Within this section, we show the potential that the developed simulation environment has in the quantification of architecture scalability (section A) and distribution of data availability within the IoT network (section B).

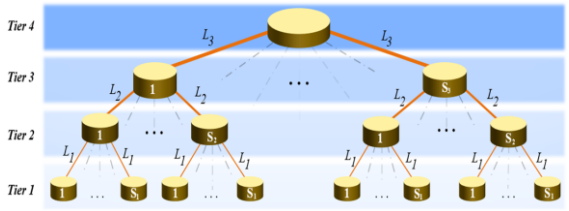

Fig. 3. Multi-tier IoT system architecture used in analysis.

Presented architecture consists of 4 tiers. Each tier is composed of data processing devices that are introduced into the simulation by using a node model. Devices that belong to the same tier have the same characteristics, and accordingly, the parameters of the node model used in the simulation are also the same. There are ***S***x20 devices located at the first tier which produce data. For example, these devices present IoT sensing devices in real world environment.

We have used the same architecture topology in section A to demonstrate the simulator capability in the scalability analysis of arbitrary architecture, and in section B to present the simulator capability to be used in data availability analysis.

A. Scalability analysis

IoT devices are introduced in simulation by using the model of producer node and they are in Tier 1. The second tier includes devices that process the data received from the lower-tier devices. For example, devices within this tier represent a smart gateway that is capable of implementing data size reduction algorithms to reduce the overall data flow through the system. These devices are introduced in simulation by using the model of consumer node. Each device located in tier 2 receives data from ***S*** devices in layer 1. The third layer consists of devices that are also modelled as consumer nodes. The devices of this tier additionally process the data received from the 5 devices of the lower tier. The central device is the only device in the fourth tier, and it has the highest data processing speeds compared to all other devices that make up the IoT system. Each Tier's parameters values are listed in Table 4.

Link L_1 is established between the first and second-tier devices. This link represents slow wireless communication links usually used within IoT sensing devices (Lora, NB-IoT, GPRS, etc.). Between the second- and third-tier devices a link L_2 is established which represents a wire link with a higher speed (for example, 10Mbps Ethernet link). L_3 link is established between the third- and fourth-tier devices and it is the fastest speed link in architecture (for example, 1Gbps Ethernet link). Each Link's parameters values are listed in Table 5.

TABLE 4 - VALUE OF NODE MODEL PARAMETERS IN ALL HIERARCHICAL TIERS

Tier	Model	PS [MB/s]	CL/AL	DPC [mA]	LPC [mA]	SR [mA]	DS [B]
1	Node producer	1	1/1	40	10	2s	100
2	Node consumer	1000	1/1	1000	200	x	x
3	Node consumer	10000	1/1	5000	500	x	x
4	Node consumer	100000	1/1	8000	900	x	x

TABLE 5 - VALUE OF LINK MODEL PARAMETERS

Link type	S [MB/s]	MTU [B]	TCT [mA]	TCR [mA]
L_1	0.05	1500	400	400
L_2	1.25	1500	400	400
L_3	125	1500	400	400

TABLE 6 - VALUE OF PROTOCOL MODEL PARAMETERS

Protocol	Overhead [B]	ARQ
P	70	Disabled

The same protocol model is used over the entire IoT system architecture and its values are listed in Table 6.

During analysis the number of S_1 nodes (nodes located at the first tier that are connected to the same node at the second tier) is modified from 5 to 10000 while the number of $S_{2\text{-}3}$ nodes remains constant: S_2=5, S_3=4. Analysis is done to find the end limitation of the architecture for a specific sampling rate. Parameters of the protocol model used within the simulation are configured to achieve a push communication model [22]. In this communication model, data is sent from IoT sensing end device (node) to the server after they are produced. Results of the simulations for different *S* values are presented in Fig. 4.

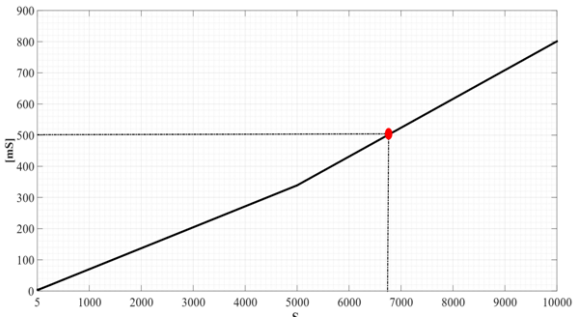

Fig. 4. Average data availability time.

It is expected that for a required sampling rate and selected architecture, there is a limitation in terms of the maximum number of nodes in architecture. Fig. 4 present results obtained after simulation over IoT architecture from Fig. 3. It is noticeable that our architecture for a required sampling rate and amount of generated data has limits in terms of scalability. The maximum number of producer nodes supported by this IoT architecture is near S_I x 20 = 6800 x 20 =136000 nodes.

B. Data availability distribution analysis

This analysis was done on the same IoT architecture as the system architecture in analysis and model's parameters values of the IoT system architecture are the same as the values presented in Tables 4 and 5.

The IoT system topology considered in this analysis has a fixed number of devices in the lowest hierarchical layer and is 100. Data within this IoT system are produced exclusively from sides of the device from the lowest hierarchical layer with a period of 2s. All devices generate 100B packets. Within this analysis, the distribution of data availability on the central device within the fourth hierarchical layer was observed.

In this analysis we present the distribution of data availability for two classes of applications.

Class 1: with this class of applications, it is necessary to observe the time required for the data to arrive at the destination node and the time required for the response from the destination node to reach the node that generated the data.

Class 2: this class of applications includes applications whose work is based on the availability of data at the destination node. This means that in these applications, the time required for the data to reach the destination node should be observed until the path time to the source is considered.

To achieve described applications behaviour, different protocol model parameters are used within a certain application class. These model parameters values are listed in Table 7.

TABLE 7 – APPLICATION CLASSES PROTOCOL'S MODELS PARAMETERS VALUES

Application class	Protocol	Overhead [B]	ARQ
1	**P_1**	70	Enabled
2	**P2**	70	Disabled

Data availability time is considered as the main parameter for profiling the performance of dedicated applications that run on a distributed dedicated system. Data availability times are calculated differently for different application classes. Data availability time for Class 1 applications is the time it takes for the data to reach the destination node, to be processed at the destination node, to generate a response, and to return the response from the destination node to the node from which the data was generated. For Class 2 applications, the data availability time is the time it takes for the generated data to reach the destination node and be processed at the destination node.

Since during the data path, and during the data generation, there are certain deviations in terms of link speeds and data generation periods, during the simulation it is not possible for data on nodes to appear at precisely defined time intervals and their availability on a node cannot be accurately predicted. Deviations in the system occur since data on a node is generated with a period that deviates from the defined period by a defined percentage. This deviation was introduced into the simulation to simulate timing asynchrony on geographically dislocated processing devices, which is often the case in distributed dedicated systems. In addition to non-synchronization in terms of data generation periods, link models support a parameter that defines the limit of deviation of link speed from the defined speed. All the above deviations have the task of simulating the phenomena in real purpose systems as closely as possible.

As already mentioned, all these deviations lead to the fact that it is not possible to predict exactly when data is available and when the application can use it, but it is necessary to observe the function of distribution of data availability and extract information of interest. The distribution functions for one system architecture and for the two observed application classes are given in Fig. 5.

During the analysis of results, it was assumed that the data are available on the node at the time when 99 percent of the generated data will be available. This time will be marked as T_{99} parameter in the continuation of the analysis. The red dot on the graphic indicates that time for some arbitrary architecture.

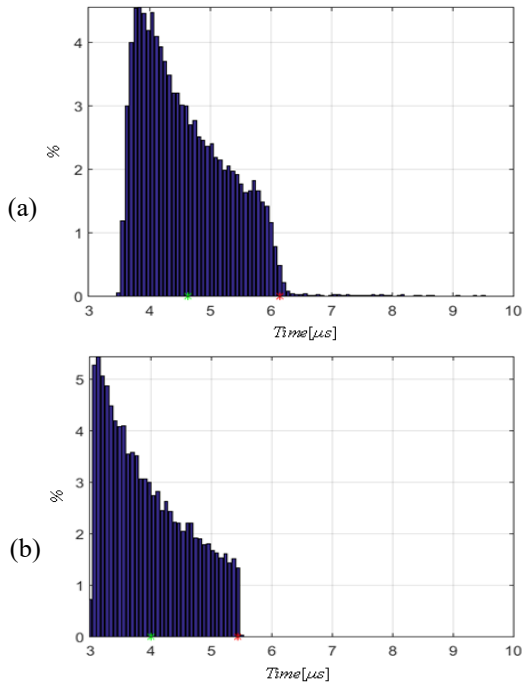

Fig. 5. (a) Distribution of data availability for class 1 applications, (b) Distribution of data availability for class 2 applications.

In addition to the data availability distribution on a node, in determining the performance of a system, it is sometimes easier to observe the data availability integral of the two application classes shown in Fig. 6. If the distribution integrals from Fig. 6 are observed, it is easier to determine the value of the parameter T_{99}.

VI. Conclusion

The proposed simulation framework supports the investigation of operational performance of multi-tier IoT system, introducing a set of features for modelling data processing and communication across the system architecture. Framework capabilities include the analysis of data availability, system scalability, and energy consumption under different communication and deployment scenarios. The case study of large-scale IoT deployment was given to demonstrate the framework capabilities and to describe the simulation process. Distribution of data availability in a multi-tier IoT system and the supported scalability analysis under the performance constraints can be used to reveal the limitations of the IoT system deployment.

As part of further research, we plan to additionally improve the simulation framework in the domain of user experience regarding system architecting, component modelling, parametric analysis, and in improving the processing performance of simulator core.

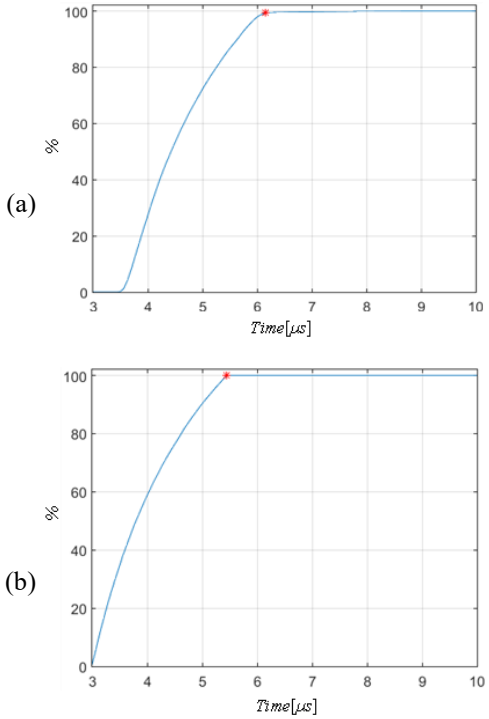

Fig. 6. Integral data distribution allocation (a) for class 1 application, (b) for class 2 application.

References

[1] *Cisco Annual Internet Report 2018-2023*, White Paper

[2] M. Ejaz, T. Kumar, M. Ylianttila and E. Harjula, "Performance and Efficiency Optimization of Multi-layer IoT Edge Architecture," *2nd 6G Wireless Summit (6G SUMMIT)*, pp. 1-5, 2020.

[3] M. Veeramanikandan., S. Sankaranarayanan, "Publish/subscribe based multi-tier edge computational model in Internet of Things for latency reduction," *Journal of Parallel and Distributed Computing*, vol. 127, 2 pp. 18-27, 2019.

[4] N. Dao, Y. Lee, S. Cho, E. Kim, K. Chung and C. Keum, "Multi-tier multi-access edge computing: The role for the fourth industrial revolution," in proc. of *ICTC 2017*, pp. 1280-1282, 2017.

[5] C. Tseng and F. J. Lin, "Extending scalability of IoT/M2M platforms with Fog computing," *IEEE 4th World Forum on Internet of Things (WF-IoT)*, pp. 825-830, 2018.

[6] Y. Yang, "Multi-tier computing networks for intelligent IoT," *Nature Electronics* 2, pp. 4-5, 2019.

[7] P. Manna, R. K. Das, "Scalability in Internet of Things: Techniques, Challenges and Solutions," *International Journal for Research in Engineering Application & Management (IJREAM)*, pp. 259-261, 2021.

[8] D. Sinha Roy, R. K. Behera, K. H. K. Reddy and R. Buyya, "A Context-Aware Fog Enabled Scheme for Real-Time Cross-Vertical IoT Applications," in *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2400-2412, April 2019.

[9] B. V. Philip, T. Alpcan, J. Jin and M. Palaniswami, "Distributed Real-Time IoT for Autonomous Vehicles," in *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1131-1140, Feb. 2019.

[10] P. Patel, D. Cassou, "Enabling high-level application development for the Internet of Things," *Journal of Systems and Software*, vol. 103, pp. 62-84, May 2015.

[11] X. Zenga, S. K. Gargb, P. Strazdinsa, P. P. Jayaramanc, D. Georgakopoulosc, R. Ranjand, "IOTSim: A simulator for analysing IoT applications," *Journal of Systems Architecture*, vol. 72, pp. 93-107, 2017.

[12] M. Chernyshev, Z. Baig, O. Bello and S. Zeadally, "Internet of Things (IoT): Research, Simulators, and Testbeds," in *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1637-1647, June 2018.

[13] G. Z. Papadopoulos, J. Beaudaux, A. Gallais, T. Noël and G. Schreiner, "Adding value to WSN simulation using the IoT-LAB experimental platform," *IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 485-490, 2013.

[14] G. Z. Papadopoulos, A. Gallas, G. Schreiner, E. Jou, T. Noel, "Thorough IoT testbed characterization: From proof-of-concept to repeatable experimentations," *Computer Networks*, vol. 119, 4 pp. 86-101, June 2017.

[15] Udoh, Itorobong S. and G. Kotonya, "Developing IoT applications: challenges and frameworks," *IET Cyper-Phys. Syst, Theory & Appl*. 3, pp. 65-72, 2018.

[16] H. Son, L. G. Myoung, C. Noel, L. Nguyen, K. Heo, B. Mihaela, G. Patrick, "DPWSim: A Simulation Toolkit for IoT Applications Using Devices Profile for Web Services," *IEEE World Forum on Internet of Things, WF-IoT*, March 2014.

[17] H. Gupta, A. M. Dastjerdi, S. K. Ghosh, R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," Wiley - Special Issue: *Cloud and Fog Computing*, vol. 47, Issue 9, pp. 1275-1296, September 2017.

[18] Z. Xuezhi, G. Saurabh, S. Peter, J. P. Prakash, G. Dimitrios, R. Ranjan, "IOTSim: a Cloud based Simulator for Analysing IoT Applications," *Journal of Systems Architecture*. Vol. 72, pp. 93-107, January 2017.

[19] T.R. Henderson, M. Lacage, G.F. Riley, "Network Simulator with NS-3 Simulator," *SIGCOMM*, 2008.

[20] K. Mehdi, M. Lounis, A. Bouncer, T. Kechadi, "CupCarbon: A Multi-Agent and Discrete Event Wireless Sensor Network Design and Simulation Tool," *Seventh International Conference on Simulation Tools and Techniques*, August, 2014.

[21] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama and N. Kato, "A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1457-1477, 2017.

[22] H. Shen, G. Bai, "Routing in wireless multimedia sensor networks: A survey and challenges ahead," *Journal of Network and Computer Applications*, pp. 30-49, 2016.

[23] L. Feng, P. Kortoçi, Y. Liu, "A multi-tier data reduction mechanism for IoT sensors," *Seventh International Conference on the Internet of Things (IoT '17)*. Association for Computing Machinery, New York, NY, USA, Article 6, pp. 1–8, 2017.

[24] LSNSimulator core source code, https://github.com/turkmanovic/LSNSimulator

[25] LSNSimulator GUI, https://github.com/turkmanovic/LSNSimulatorApp.git

[26] H. Turkmanović, I. Popović, D. Drajić and Z. Čiča, "Lunching real-time IoT applications on energy-aware embedded platforms," *15th International Conference on Advanced Technologies, Systems and Services in Telecommunications*, Niš, pp. 279-282, October 2021.

[27] R. C. Sofia, P. M. Mendes, "An Overview on Push-Based Communication Models for Information-Centric Networking," *Future Internet*, 11(3), 74, 2019.

[28] H. Turkmanović, I. Popović, Z. Čiča and D. Drajić, "Simulation framework for performance analysis in multi-tier IoT Systems," *2021 29th Telecommunications Forum (TELFOR)*, 2021, pp. 1-4, doi: 10.1109/TELFOR52709.2021.9653170.