UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
Más que un estado sólido

https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRÓNICA

A RESEARCH VISION

Emulator for data analytics and IoT development projects

Emulador para desarrollo de proyectos IoT y analíticas de datos

Andrés Armando Sánchez-Martin [1], ***Luis Eduardo Barreto-Santamaría*** [2], ***Juan José Ochoa-Ortiz*** [3], ***Sebastián Enrique Villanueva-Navarro*** [4]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 11/11/2019
Recibido: 28/12/2019
Aceptado: 09/02/2020

Keywords:
Architecture
Data analytics
Emulator
IoT
Networking.

CC BY NC

Palabras clave:
Arquitectura
Análisis de datos
Emulador
IoT
Redes

ABSTRACT

One of the difficulties for the development and testing of data analysis applications used by IoT devices is the economic and temporary cost of building the IoT network, to mitigate these costs and expedite the development of IoT and analytical applications, it is proposed NIOTE, an IoT network emulator that generates sensor and actuator data from different devices that are easy to configure and deploy over TCP/IP and MQTT protocols, this tool serves as support in academic environments and conceptual validation in the design of IoT networks. The emulator facilitates the development of this type of application, optimizing the development time and improving the final quality of the product. Object-oriented programming concepts, architecture, and software design patterns are used to develop this emulator, which allows us to emulate the behavior of IoT devices that are inside a specific network, where you can add the number of necessary devices, model and design any network. Each network sends data that is stored locally to emulate the process of transporting the data to a platform, through a specific format and will be sent to perform Data Analysis.

RESUMEN

Una de las dificultades para la realización de pruebas de aplicaciones de análisis de datos obtenidos por dispositivos IoT, es el costo económico y temporal de la construcción de la red IoT, para mitigar estos costos y agilizar el desarrollo de aplicaciones IoT y analíticas, se propone NIOTE, un emulador de redes IoT que genera datos de sensores y actuadores diferentes tipos de dispositivos, fáciles de configurar y desplegar sobre protocolos TCP/IP y MQTT, esta herramienta sirve como apoyo en ambientes académicos y validación conceptual en el diseño de redes IoT. El uso del emulador facilitara el desarrollo de este tipo de aplicaciones, optimizando el tiempo de desarrollo y mejorando la calidad final del producto. Para desarrollar este emulador se utilizaron conceptos de programación orientada a objetos, arquitectura y patrones de diseño de software, que permitieron emular el comportamiento de los dispositivos IoT que se encuentran dentro de una red específica, donde se puede agregar la cantidad de dispositivos que sean necesarios, modelar y diseñar cualquier red deseada. Cada red creada envía datos que son almacenados de forma local para simular el proceso de transportación de los datos a una plataforma, si se desease aplicar, a través de un formato especifico en el que la información será enviada para hacer Análisis de Datos.

[1]BSc. in Systems engineering, Universidad Católica de Colombia, Colombia. MSc. in Systems and Computing Engineering, Pontificia Universidad Javeriana, Colombia. Current position: Universidad de San Buenaventura, Colombia. E-mail: aasanchez@usbbog.edu.co
[2]BSc. in Systems engineering, Universidad Católica de Colombia, Colombia. MSc. in Systems and Computing Engineering, Pontificia Universidad Javeriana, Colombia. Current position: Universidad de San Buenaventura, Colombia. E-mail: lbarreto@usbbog.edu.co
[3]BSc. in Systems engineering, Universidad de San Buenaventura, Colombia. E-mail: oojuan@academia.usbbog.edu.co
[4]BSc. in Systems engineering, Universidad de San Buenaventura, Colombia. E-mail: vsebastian@academia.usbbog.edu.co

Cite this article as: A. A. Sánchez-Martin, L. E. Barreto-Santamaría, J. J. Ochoa-Ortiz and S. E. Villanueva-Navarro, "Emulator for data analytics and IoT development projects", Visión electrónica, vol. 14, no. 1, pp. 57-63, january 2020.
https://doi.org/10.14483/22484728.16043

1. Introduction

At present, the definition of the Internet of things (IoT) is not so unknown, if you look for a specific definition, you can find that it is a set of physical devices connected by a network: cell phones, cars, houses and other components formed with electronics, software, and sensors allowing these devices to collect and exchange data [1].

IoT devices are responsible for obtaining data and sending it to the cloud, allowing the connection between these objects. According to the SAP research center, these devices are perfectly integrated into the information network, which makes it possible to interact with them through the internet, being able to check or edit their status in real-time [2].

One of the great fields of use of the data provided by IoT devices is focused on data science that in turn contains Big data and Data Analytics [3]. The Data Analytics process consists of analyzing the data that is collected and stored within the software that performs this process of analysis of previously collected and stored [4].

In order to observe the behavior of an analytical platform for data from IoT devices, you can choose to recreate a real network of physical devices, for this a large amount of economic investment is required for the acquisition of the devices [5], so it is not usually a viable option and network simulation is used.

In the market we find some simulators, among them we have [6, 7]:

- ns-3 (Network Simulator version 3): Sensor network simulator that works on different scales, field applications, and field conditions.
- OMNeT ++: It is a c ++ simulation library, oriented to the simulation of communication networks and distributed systems but also used to simulate a network of IoT sensors [8].
- TOSSIM: Simulator for TinyOS wireless sensor networks [9].
- Avrora: Network simulator of instruction-level sensors and with cycle precision [10].

According to the information of each one of the emulators, it is evident that all simulate only sensor networks, some even only one type of sensor as is the case of TOOSSIM [9]. Unlike these NIOTE allows recreating a more complete network of IoT devices with different types of sensors and actuators. The article contains the architectures of IoT and SDN devices along with a description of the emulation process. There is a chapter dedicated to the description of the emulator's functionalities and another for the architectures and design patterns used for development.

2. Topic development

The development of this emulator was based on facilitating the design and construction processes of IoT networks, with a variety of devices, which have different communication protocols, such as Wi-Fi, Bluetooth, and Zigbee, which means that it will generate varied information that can be analyzed. The emulator architecture is based on a 3-layer design, application, service, and data, which gives us in broad strokes the way this emulator will behave.

Using IoT implies also identifying an architecture, but this is developed according to the creation of IoT networks, within the emulator, to define its architecture, as well as knowing how the data manipulation and interactions between different devices.

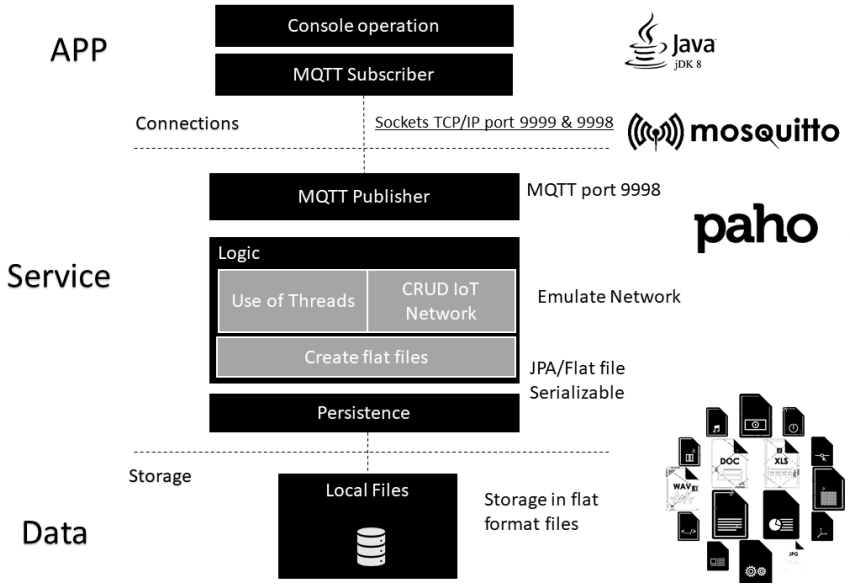
2.1. IoT and SDN Architecture

The IoT architecture consists of a set of devices capable of transferring information between them and at different points, relevant information about a case study. There are many architecture models, but not concrete or general, they can be raised as best suited to the problem or case study of the project.

On the other hand, there is the SDN architecture that is based on network management through software. The use of IoT devices has had a great evolution in recent years that when combined or being built in a network, another type of architecture is created that has also been used for projects that involve IoT, this architecture is called Software Defined Networking, which is based on the simplification of the management and security of the networks, by separating the control scheme from the data scheme, being the centralized control scheme so that not only a single device is programmed but the entire network can be programmed net.

A centralized control scheme simplifies the administration of large-scale IoT networks, which is what is needed for the emulator, as it is also flexible in managing the storage of the data to be analyzed and to continuously maximize performance, making it as optimal as possible in the flow of data within the IoT network [11].

Figure 1: Architecture 3 layers of the NIOTE emulator.



Source: own

The emulation process consists of observing an event in real life and generalizing it so that it adapts to any model of the event being carried out [11]. For the creation of this emulator, we rely on the idea of help to develop projects that are focused on the use of IoT networks, how to facilitate the design of the network structure that is being created.

Having the design and a previous operation of a network, there is a lot of work done since the process of network construction is optimized, physically, as well as a lower expense in the acquisition of IoT devices, that are required for the proposed network. Then the emulator enters as a solution to this problem, to be able to create network models faster, based on the case study or network that you want to mount. Another important point is to identify what are the transfer methods or protocols that these devices will use, where it focuses on finding what is the format in which the data is being sent when you want to store the data in an external source, which I managed to identify the format in which it is sent and it is correct.

2.2. *Functionality Definition*

Within the functionalities of the emulator, you have the creation of the different objects [12] The first object that contains all the others in the network, we can create a new network, where we will have different IoT devices (sensors and actuators), which are the ones that will generate the data, depending on the type of sensor or actuator. Then, to create a network there is another possibility and it is to import an already created network, that is, we can load a previously developed network, with the configurations made and devices added.

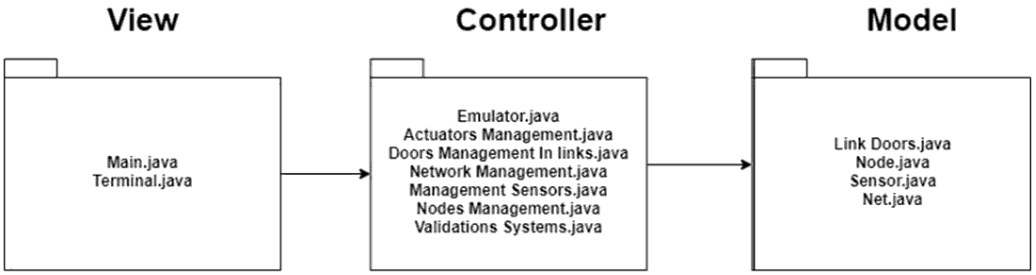
The following objects can be created, deleted, modified and consulted, these objects are the gateways that will allow the connection between the server and client, as well as a publisher and subscriber to receive the data that is sent by this medium. Then there are the nodes, which are the ones that store the data of the sensors and actuators, devices responsible for generating the data. Each gateway can have many nodes, which receive different data from many sensors or actuators, as there can also be many gateways within the network.

The data that is generated from the sensors and actuators are passed through sockets, in a subscriber publisher model [13], to present the data that has been generated and analyze what format they have, to define the type of data they are these.

2.3. *Architecture and Pattern*

To define the architecture, a client-server architecture was used to establish communication between applications [14] and a publisher-subscriber architecture for the reception of the data that will be sent [13].

Figure 2: Package diagram of the design pattern.



Source: own

The MVC design model (Controller View Model) [15], was used as the software design pattern, in which we divide the application into packages where the programming in each of these is different, in the model package we have everything related with the connection to the database, where you can have the database tables as classes, which allows you to export the attributes of this table and represent it in the attributes of a class, but for the emulator, you will not use the database data, but we will store all the information in separate files.

Inside the controller, the package is all the logic processes that will be inside the emulator, as well as the processes of creation, modification, elimination, and consultation of the different objects existing within the model package, which are the different IoT devices.

The package of the view is the one that allows us to obtain the logical processes and to show them through a console. To better visualize this pattern, you can see Figure **2**.

2.4. Prototype

The development of the emulator was based on the model of the proposed architecture, where the objects that make up an IoT network were defined, these are the sensors, actuators, nodes, gateways (Gateway) and network. Then the logic was structured within each object and finally the graphic interface was designed through the console.

The Figure **3** shows how the emulator works from the moment it starts.

First, you start by creating a new network.

Now the attributes of the new network are placed. These attributes are an identifier, the name of the network and a brief description, as seen in the Figure **4**.

Figure 3: Start of the application, the main screen of the emulator.

```
run:
***Bienvenido a Net IoT Emulator***
------------------------------------------------
------------------Menu Inicial------------------
------------------------------------------------
0. para salir
1. para crear red
2. cargar una configuracion previa de red e iniciar
3. Acerca de...
```

Source: own

Figure 4: Network creation.

```
run:
***Bienvenido a Net IoT Emulator***
------------------------------------------------
------------------Menu Inicial------------------
------------------------------------------------
0. para salir
1. para crear red
2. cargar una configuracion previa de red e iniciar
3. Acerca de...
1
ingrese ID unico de la red
1020
ingrese nombre de la red
RedIoT
ingrese la descripcion de la red
Esta es la red para datos ambientales
```

Source: own

Having already created a network you can do several things, such as deleting and restarting the network, so that you return to the default or new values, you can configure the network and, finally, control the entire network. In the Figure **5**, the network will be configured, since it has no objects, then the different devices of the network will be created.

Figure 5: Main network menu, choosing the option to configure the network.

```
Net IoT Emulator
Red: 1020 - RedIoT - red
------------------------------------------------
-----------------Menu Principal-----------------
------------------------------------------------
0. para salir
1. para borrar y reiniciar la red
2. para configurar la red
3. para controlar la red
4. Acerca de...
```

Source: own

When entering the network configuration menu, you can configure each of the objects that a network can have, you can also save this configuration in a separate file to be able to recover this configuration, database need. Here you will choose the option Configure Gateway, to create this object within the network, this process is identical for each of the objects that make up the IoT network, as shown in the Figure 6.

Figure 6: Network configuration menu, we choose the option to configure the gateway.

```
Net IoT Emulator
Red: 1020 - RedIoT - red
------------------------------------------------
--------------Configuracion de Red--------------
------------------------------------------------
0. para volver al menu anterior
1. para configurar Puertas de Enlace
2. para configurar Nodos
3. para configurar Sensores
4. para configurar Actuadores
5. para construir la red
6. para guardar la red
```

Source: own

Within this menu, you can manage the gateway object, in which you can see all the existing doors in a network, visualize a specific gateway, create a gateway, modify it and remove it from the network. The Figure 7 shows the composition of the menu.

After placing the gateway attributes, it is created and added to the network.

This creation process is identical for the creation of the other objects of the IoT network and at the same time that objects are created, which ones are created and added to the network can be visualized.

Figure 7: Gateway configuration, specification of menu options.

```
Net IoT Emulator
Red: 1020 - RedIoT - red
------------------------------------------------
---------Configuracion Puerta de Enlace---------
------------------------------------------------
0. para volver al menu anterior
1. para ver todas las Puertas de Enlace
2. para ver Puerta de Enlace por ID
3. para crear Puerta de Enlace
4. para modificar Puerta de Enlace por ID
5. para eliminar Puerta de Enlace por ID
3
```

Source: own

Figure 8: Creation of the gateway and defining its attributes.

```
Digite ID de la Puerta de enlace
3020
Digite descripcion de la Puerta de enlace
Es la prueba
Digite direccion logica de la Puerta de enlace
Digite puerto de servicio de la Puerta de enlace
Digite protocolo de comunicacion externo de la Puerta de enlace
Wifi
la puerta de enlace con ID: 3020 se agrego
```

Source: own

2.5. Prerequisites for emulator use

To develop the emulator, certain validations of the system or equipment that the emulator is using were taken into account. These validations are for the client-server and subscriber publisher connection protocols. Therefore, if any equipment does not meet the requirements that the emulator needs, it is responsible for notifying the user, through the console and instantly running the emulator, it verifies that the machine meets the requirements to operate. If you do not have the prerequisites, you must install and configure the parameters that the emulator sets when notifying the user, such as the installation of MQTT on the ports you specify and the ports for communication of TCP / IP sockets on the machine correspondent.

Within the parameters to be configured, there is the theme of the ports that the application will use to perform the communication protocols, already mentioned, for our emulator the following ports are required for the client-server connection: it requires ports 9998 and 9999 for communication, which is of the TCP / IP type. Also, how an MQTT protocol will

be used (which is a protocol for machine-to-machine communication, or as it is said in English Machine-to-Machine (M2M). For this protocol a program called Mosquitto is required that allows this communication, which runs this service on port 1883. Previously, it is mentioned that the emulator informs the user that it is what is missing for the application to work correctly, in the Figure 9, which shows us the specifications and requirements that are needed for the correct execution of the emulator.

With the prerequisites that the emulator asks for, they are not extensive, but they are very specific, that without them it is not possible to run the emulator, as we also see that we only require non-hardware software, unlike other emulators that need too many resources external to operate, according to the architecture that these emulators raised.

Figure 9: Prerequisites of the emulator to function properly.

```
run:
***Net IoT Emulator necesita el Broker de Mosquitto-MQTT y los puerto 9998 y 9999 para funcionar
Verifique que Mosquitto-MQTT y TCP/IP esten instalados y configurados
Verifique que Mosquitto-MQTT esta corriendo en el puerto 1883 y agregado al firewall
Verifique que los puertos 9998 y 9999 estan disponible y agregados al firewall
BUILD SUCCESSFUL (total time: 0 seconds)
```

Source: own

3. Tests and validations

The previous section shows the main base of the emulator that allows creating a network and the different devices it contains. The emulator is focused on devices that have many communication protocols when the time comes to send the information or store it, it will be sent through the MQTT transfer protocol, which is one of the necessary configurations to use the emulator. Besides, the emulator can generate, store and send large volumes of data allowing to see how the behavior of this data is through the console, as well as the behavior if removing part of the gateways, as per example nodes, you can see many things within the networks created and the behavior of each of the devices.

The way in which the emulator data is generated is through the use of threads, which is a type of logic that allows us to create records in time periods, in order to emulate the generation of varied data, this allows us to create or identify a type of format in which we can send and store the data.

4. Conclusions

The emulator allows the creation of different objects such as Gateway doors, nodes, sensors and actuators; Depending on the interest of the user, they can join the network or remain outside. The sensors and nodes that are in the network generate data and send them through sockets so that the platform to which the tests are carried out captures them.

The MVC design pattern used for the development of the emulator allows updating the structure of the objects, without altering the software processes thus facilitating the implementation of different types of devices.

The emulator is thinking entirely for the emulation of IoT networks, it has the advantage of containing different types of sensors and actuators, which allows emulating a complete network of devices in which data of all types are sent with a frequency given by the user. All these components will make the tests to the applications cover all the aspects and it is possible to develop better quality software.

The development of the emulator allows us to show how the construction mechanics of an IoT network are, that this is made up of multiple devices or objects, which are necessary to be able to generate the data, as the generality of the emulator can also be seen since allows to create IoT networks from any model that is proposed, so as not to limit it. An important factor of this prototype is that it does not require any external resources, either at the software or hardware level, but rather a light emulator, which can be used in a local environment, without harming any development or production that is being carried out during the course of the IoT project that arises.

5. Acknowledgements

We have to recognize the support of the DaTAn seedbed that is the space where the emulator was made and thanks to the Systems Engineering program for the

academic contributions for its development, as well as the University of San Buenaventura for providing the necessary tools for Emulator development.

References

[1] A. Pal and B. Purushothaman, "IoT Technical Challenges and Solutions", Artech House, 2017, pp. 15-40.

[2] S. E. Abasolo, M. A. Carrera, R. X. Gordillo and C. G. Romero, "Evaluación del modelo de referencia de Ïnternet of Things"(IoT), mediante la implantación de arquitecturas basadas en plataformas comerciales, open hardware y conectividad IPv6", thesis, Universidad de las Fuerzas Armadas ESPE, Ecuador, 2013.

[3] J. García, J. M. Molina, A. Berlanga, M. A. Patricio, Á. L. Bustamante and W. R. Padilla, "Ciencia de datos Técnicas analíticas y aprendizaje estadístico en un enfoque práctico", Bogotá: Alfaomega, 2018.

[4] B. Prieto-Valero, "La nueva función de control de gestión: data analytics para ser más competitivos", 2017. [Online]. Available at: https://www.tendencias.kpmg.es/2017/06/la-nueva-funcion-de-control-de-gestion-data\-analytics-para-ser-mas-competitivos/

[5] J. E. Luzuriaga, M. Zennaro, C. Tavares, J. C. Cano and P. Manzoni, "Evaluando un escenario de pruebas para el IoT entre la emulación y el uso de dispositivos reales", España: Ediciones Universidad Salamanca, 2016.

[6] M. Martin-Iglesia, "Análisis de la simulación de dispositivos, circuitos y sistemas electrónicos para Internet de las cosas (IoT)", thesis, E.T.S.I. Industriales (UPM), 2019.

[7] G. Riley and T. Henderson, "The ns-3 Network Simulator", in Modeling and Tools for Network Simulation, K. Wehrle, M. Güneş and J. Gross, Berlín: Springer, 2010, pp. 15- 34. https://doi.org/10.1007/978-3-642-12331-3

[8] A. Varga, .ºMNeT++", in Modeling and Tools for Network Simulation, K. Wehrle, M. Güneş and J. Gross, Berlín: Springer, 2010, pp. 35-59. https://doi.org/10.1007/978-3-642-12331-3_3

[9]] P. Levis, N. Lee, M. Welsh and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications", SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems, 2003. https://doi.org/10.1145/958491.958506

[10] B. L. Titzer, D. K. Lee and J. Palsberg, "Avrora: Scalable sensor network simulation with precise timing", Fourth International Symposium on Information Processing in Sensor Networks, 2005. https://doi.org/10.1109/IPSN.2005.1440978

[11] C. González, O. Flauzac and F. Nolot, "Evolución y Contribución para el Internet de las Cosas por las emergentes Redes Definidas por Software", Congreso Internacional en Inteligencia Ambiental, Ingeniería de Software y Salud Electrónica y Móvil – AmITIC, 2018.

[12] D. J. Barnes and M. Kölling, "Programación orientada a objetos con Java", Pearson Educación, 2007.

[13] F. Moreno-Cerdá, "Demostrador arquitectura publish/subscribe con MQTT", thesis, Universitat Politècnica de Catalunya, España, 2018.

[14] E. Marini, "El Modelo Cliente/Servidor", 2012. [Online]. Available at: http://index-of.co.uk/REDES/linuxito%20-%20El%20Modelo%20Cliente-Servidor.pdf

[15] Y. D. González and Y. F. Romero, "Patrón Modelo-Vista-Controlador", *Revista Telemática*, vol. 11, no. 1, pp. 47-57, 2012.