06

Fecha de presentación: Junio, 2018
Fecha de aceptación: Julio, 2018
Fecha de publicación: Octubre, 2018

TEACHING

OF DISTRIBUTED ARCHITECTURES AT THE UNIVERSITY FOR SATISFYING INTERNET OF THINGS DEMAND

LA ENSEÑANZA DE ARQUITECTURAS DISTRIBUIDAS EN LA UNIVERSIDAD PARA SATISFACER LA DEMANDA DEL INTERNET DE LAS COSAS

Dra. C. Guadalupe Ortiz[1]
E-mail: guadalupe.ortiz@uca.es
Dr. C. Alfonso García-de-Prado[1]
E-mail: alfonso.garciadeprado@uca.es
Dr. C. Juan Boubeta-Puig[1]
E-mail: juan.boubeta@uca.es
[1] Universidad de Cádiz. Puerto Real. España.

Suggested citation (APA, sixth edition)

Ortiz, G., García-de-Prado, A., & Boubeta-Puig, J. (2018). Teaching distributed architectures at the university for satisfying internet of things demand. *Universidad y Sociedad*, 10(5), 51-59. Recuperado de http://rus.ucf.edu.cu/index.php/rus

ABSTRACT

Teaching of distributed architectures in the computer engineering degrees has traditionally been based on contents related to well-known established software paradigms and architectures. However, over the last few years, new solutions in the field of distributed architectures have emerged, especially within the field of the Internet of Things (IoT). In this article we tackle a methodology for teaching distributed architectures for the IoT from this new perspective. In this context, the description, implementation and testing of distributed architectures for IoT is therefore proposed. The methodology includes the development and testing of a case study relevant for the scope of IoT and smart cities using emerging technologies. In particular, in this paper we show the procedure followed to implement and test a case study related to traffic regulation and emergency vehicles movement, enabling such vehicles to reach their destination in the minimum amount of time.

Keywords: Computer sciences, distributed architectures, Internet of things.

RESUMEN

La enseñanza de las arquitecturas distribuidas en los estudios universitarios de ciencias de la computación o ingeniería informática se ha basado tradicionalmente en contenidos relacionados con paradigmas y arquitecturas de software bien conocidos y establecidos. Sin embargo, en los últimos años han surgido nuevas soluciones en el campo de las arquitecturas distribuidas, especialmente en el campo del Internet de las Cosas (IoT). En este artículo abordamos una metodología para enseñar arquitecturas distribuidas para el IoT desde esta nueva perspectiva. En este contexto, se propone, por tanto, la descripción, aplicación y prueba de arquitecturas distribuidas para el IoT. La metodología incluye el desarrollo y ensayo de un caso de estudio relevante en el ámbito del IoT y las ciudades inteligentes mediante el uso de tecnologías emergentes. En particular, en este artículo mostramos el procedimiento seguido para implementar y probar un caso de estudio relacionado con la regulación del tráfico y el desplazamiento de vehículos de emergencia, permitiendo que dichos vehículos lleguen a su destino en el menor tiempo posible.

Palabras clave: Ciencias de la computación, arquitecturas distribuidas, Internet de las cosas.

INTRODUCTION

Teaching distributed architectures in computer engineering is mainly based on contents related to traditional software architectures. However, over the last few years, new architectures have emerged providing novel solutions in the field of distributed architectures, such as Hadoop-based approaches (Agneeswaran, 2014) or Service-Oriented Architecture (SOA) ones (Papazoglou, 2012), respectively. On the other hand, the relevance of the Internet of Things (IoT) in recent years (European Research Group in the Internet of Things, 2012), also requires special attention as to how architectures are adapted in this context. All these issues have made us considering how to improve teaching in this field and have derived us to propose new technologies and methodologies accordingly.

In the past we started introducing distributed programming research concepts in the computer science degree (García de Prado & Ortiz, 2017), but in this case we go one step forward. In this context, the description, implementation and testing of distributed architectures for the IoT is therefore considered of high relevance for computer science curricula (Dempsey, 2017). In particular, we propose the study of event-driven service-oriented architectures (Taylor, 2009) for the IoT following the methodology described in the following section. The greatest handicap is found when testing the implemented architectures, due to the fact of using new and constantly evolving technologies for the implementation. Additionally, there is a lack of appropriate tools available for testing. For this reason, we also propose the use of a tool for the generation of synthetic data we created with the aim of testing IoT architectures, as part of the teaching methodology.

The rest of the paper is organized as follows. First of all, we provide a brief technological background to facilitate the paper comprehension to readers coming from other disciplines. Secondly, we explain the methodology proposed for teaching distributed architectures for satisfying IoT demand. Then, we explain a real case study proposed and implemented by a student following the methodology during 2017-2018 academic year. Moreover, we also explain how the student could test the case study thanks to the use of a tool developed by the authors. Then, discussion and educational results are provided and, finally, conclusions are presented.

DEVELOPMENT

In this section, we explain some high level information about the technologies used, as previously said, so that to facilitate the paper comprehension to readers coming from other disciplines.

Internet of Things

IoT is defined in a wide way as a network formed by interconnected physical objects uniquely identified (Atzori, Iera & Morabito, 2010) and implies obtaining, transferring, processing and analysis of data coming from such objects, as well as integration with the software architectures making use of such data. Currently, several algorithms, tools, and technologies enable IoT applications and emerging architectures for a variety of application domains (Buyya & Vahid Dastjerdi, 2016). Such architectures should provide a set of offered services, communication networks and event processing and should fulfil key requirements such as interoperability, reliability and scalability (Buyya & Vahid Dastjerdi, 2016).

The relevance of implementing suitable architectures for the IoT is not only a question of research, but also an economic issue: the economic impact expected from IoT applications is 11% of the worldwide economy (Buyya & Vahid Dastjerdi, 2016). This is why we consider of high relevance to include such topics in education.

Enterprise Service Bus

Before explaining Enterprise Service Buses (ESBs), we have to introduce SOAs. A SOA consists of a paradigm for the design and implementation of loosely coupled distributed systems which make use of services for their implementation. For these architectures the focus remains on the business process rather than on the technologies, allowing an easier integration of third-party services (Papazoglou, 2012).

The basic unit of software to implement a SOA is a web service. With the growth of service components and processes in SOAs, a new service infrastructure is required for maintaining applications in a flexible way and such an infrastructure should provide support for a message middleware. These requirements are fulfilled by an ESB, which provides services to more elaborated architectures through a messaging system (Papazoglou, 2012), supplying interoperability among diverse formatting and communication standards.

nITROGEN

nITROGEN (http://ucase.uca.es/nITROGEN) is a tool that generates synthetic data and connects these data with IoT oriented architectures (García De Prado, Ortiz, & Boubeta-Puig, 2017; García-de-Prado, Ortiz, & Boubeta-Puig, 2017). The tool can create 4 different types of data:

arrays, numbers, strings and dates following several predefined or customized data distributions. It can also connect and submit the generated data to different IoT platforms, applications and messaging brokers, such as ThingSpeak, Mosquitto, RabbitMQ, databases as MySQL or noSQL ones, or simply save the data on local files. Of course, the tool is capable of creating multiple threads for simultaneous data generation, as well as controlling the speed of creation of such data, by different configurable rates.

Complex Event Processing

For some scenarios, traditional SOA have evolved towards Event-Driven SOAs (ED-SOA or SOA 2.0), where communication between the different agents of the system are carried out by events, rather than using remote procedure calls (Luckham, 2012).

In order to analyse and correlate big amounts of events in such architectures, it is necessary to integrate Complex Event Processing (CEP) (Luckham, 2012), which is a technology that allows capturing, analysing and correlating a large amount of heterogeneous data with the aim of detecting relevant situations in a particular domain (Inzinger, Hummer, Satzger, Leitner & Dustdar, 2014). With CEP languages we can define the so called event patterns where we specify the conditions to be met in order to detect such situations, namely complex events. In this type of architectures, the use of a message broker can be key to succeed. Message brokers implement asynchronous communications which allow source and target messages to be completely decoupled, as well as permitting storing the messages in the broker until processed.

The methodology proposed for teaching distributed architectures in the scope of IoT requires a great practical load and a constant work throughout the semester. In the following paragraphs we explain the steps proposed for the methodology in this paper, also represented in Figure 1:

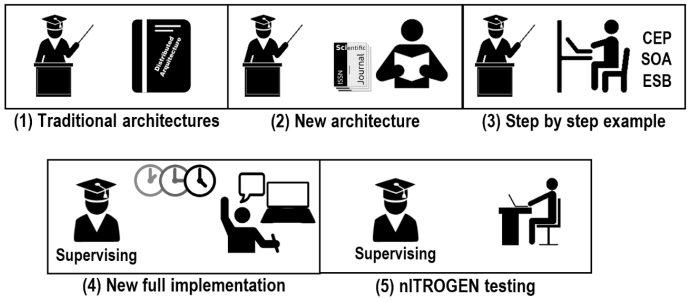

Figure 1. Proposed methodology.

1. First of all, the study of basic theoretical knowledge about distributed architectures is required. In this case, the explanation is not different from that provided in the usual teaching methodologies: a generalized knowledge of the evolution of this type of architecture is explained.
2. Secondly, new technologies and newly created architectures for the IoT are explained. In this first contact, this new knowledge is explained theoretically and a couple of relevant research papers are proposed for reading and discussion. Likewise, various areas related to the IoT in general and to smart cities in particular are discussed to show the applicability of the technologies studied.
3. The technologies to be used are then studied by developing a simple, step-by-step example. In this sense, CEP, SOA and ESBs are studied as a basis for the implementation of SOA 2.0 for the IoT.
4. The bulk of the work is in the development of the software architecture needed to tackle a specific case study in the field of IoT and smart cities at the student's choice. For the development the student will have to use the technologies previously studied. The teacher will assist him during the full process of implementation of the architecture, supporting any doubts that may arise or suggesting improvements when necessary.
5. Finally, the time has come to test the architecture. Distributed architectures must be scalable as well as efficient. To this end, the student is offered the opportunity to test his architecture using nITROGEN, our synthetic data generator for the IoT. The teacher will again give support to the doubts that arise to the student and will guide him in the correct way to test the system.

In this section, we motivate the case study challenge and we introduce the software architecture, both proposed by one of our students during 2017-2018 academic year.

Motivation

Today, due to the high density traffic, it is becoming increasingly difficult for emergency vehicles to trace and make a fast route to their destination. Even though the drivers are trained to handle troubles, emergency vehicle's drivers cannot deal with some problems on the road, like vehicles jams, opposite one-way or forbidden roads or red traffic lights. Every second counts for success, and success means saving a life.

Therefore, we need a way of stopping or reactivating traffic to give priority to emergency vehicles, to let such vehicles reach the top speed needed on every moment. With the evolution of intelligent systems and the ability to detect events and locations, technologies are ready to respond to this need. In order to be able to develop an

implementation that meets this necessity, we need to have the real-time location of the emergency vehicle in question from the start of the journey to the destination point, as well as additional information on traffic status and traffic light control.

In this case study we will focus on two scenarios:

1. The first one focuses on the emergency vehicle circulating through an avenue: in such case, the emergency vehicle should maintain a speed between 40 km/h and 60 km/h if the traffic does not cause any troubles. In this case, we want to test the option of stopping the traffic not only in the avenue, but also in any access to it from adjacent streets, therefore letting the emergency vehicle to evade the stopped cars, without needing to be aware of whether they can move and let free space.
2. The second one is based on a narrow one-way street which traffic-light is usually in red because it leads to an avenue. In this case the emergency vehicle has to pass through such street. The situation typically can lead to a jam and the vehicle is forced to stop; this could lead into several seconds or minutes of delay, even increased if the vehicles in front of it are too slow.

Software Architecture

The software architecture implemented in this case study is composed of the following elements, as shown in Figure 2:

- Sensors: Due to the fact that we cannot use real emergency vehicles information nor up-to-date traffic information, we will generate such sensor data through the use of nITROGEN. So, we will simulate the emergency vehicle position and the traffic light status. In particular, we will provide the distance between the current and target locations in a straight street, the state of the traffic light (R or G), and the direction —from 1 to 4—, representing the directions back and forward in the avenue and accessing it from the right or the left.

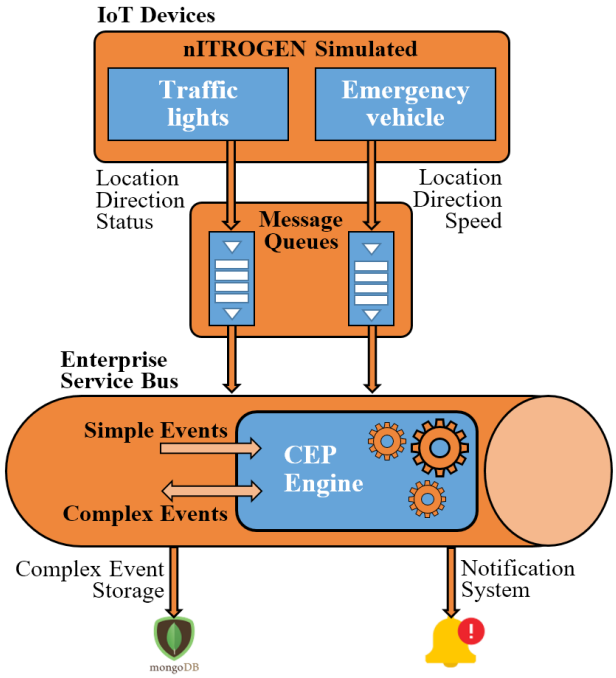

Figure 2. Architecture proposed for the case study.

- Message queues. The message queues can receive data from un unlimited number of sources concerning emergency vehicles, traffic information, et cetera. In this case, they receive the data generated in the simulator.
- Enterprise Service Bus: The ESB will act as a middleware for managing and exchanging the information between all the different components involved in the process. It is subscribed to the messages received in the message queues and it redirects such information to the CEP engine integrated with it, as well as submits notifications according to the complex events detected.
- Complex Event Processing Engine: The events created through the use of nITROGEN for simulating a traffic light and the emergency vehicle directions are processed in the CEP engine where we have defined the event patterns to be detected (see the following section). Depending of the complex events detected new signals can be sent to the traffic light to change their states.
- Notification system: this prototype only sends electronic mail notifications, but the real implementation should be able to send the appropriate order to the corresponding traffic light, as previously explained.
- NoSQL Database: We have such a database to store the number of state changes, in order to be able to know for future decisions if we are blocking a traffic

light too much time, or the opposite, if the emergency vehicle needs more time.

Event Patterns

First, we have defined the CEP domain by using the MEdit4CEP tool (Boubeta-Puig, Ortiz & Medina-Bulo, 2015), a model-driven solution for bringing CEP technology closer to any user, hiding all implementation details from them. This tool provides us with the syntactical validation and automatic transformation of the graphical event pattern models into Esper Processing Language (EPL) code. In particular, this CEP domain is composed of the ***TrafficLight*** event type (see Figure 3). This event type has the following event properties: ***trafficLightPosition*** (from 0 to 60, a traffic light's position related to the point of origin), ***ambulancePosition*** (from 0 to 60, an ambulance's position related to the point of origin), ***direction*** (from 1 to 4, representing the directions back and forward in the avenue and accessing it from the right or the left), ***state*** (R or V, the state of the traffic light), and the ***timestamp*** in which the event has happened.

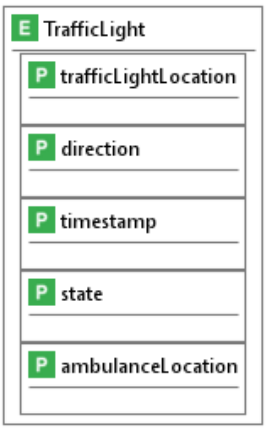

Figure 3. Ambulance CEP domain modeled with MEdit4CEP.

Once the CEP domain was defined, we have defined the following three event patterns:

- ***RedTrafficLightState***: this pattern detects when an ambulance driving on a two-way road is reaching traffic lights placed in a crossroads or pedestrian crossing. As a consequence, all these traffic lights' states will be set to red. Figure 4 depicts this pattern modeled with MEdit4CEP. Then, the pattern model has been automatically transformed into Esper EPL implementation code (see Listing 1).
- ***GreenTrafficLightState***: this pattern detects when an ambulance driving on a one-way road is reaching traffic lights. Then, all these traffic lights' states will be set to green in order not to block the ambulance.
- ***SideTrafficLightState***: this pattern will allow to block the horizontal traffic to the ambulance direction, i.e. traffic lights whose states are 3 and 4. That way, traffic is cut to let the ambulance go through the road.

Listing 1. Esper EPL implementation code for RedTrafficLightState pattern automatically generated by MEdit4CEP.

```
@Name("RedTrafficLightState")
@Tag(name="domainName", value="Ambulance")
insert into RedTrafficLightState
select a1.trafficLightLocation as trafficLightLocation1,
 a1.direction as direction1,
 'R' as state1,
 'R' as state2,
 a2.trafficLightLocation as trafficLightLocation2,
 a2.direction as direction2,
 a2.timestamp as timestamp
 from pattern [((every a1 = TrafficLight(a1.direction = 1)) -> a2 = TrafficLight((a2.direction = 2 and a2.ambulanceLocation < a2.trafficLightLocation and (a2.trafficLightLocation - a1.trafficLightLocation) < 25.0)) where timer:within(10 seconds))]
```

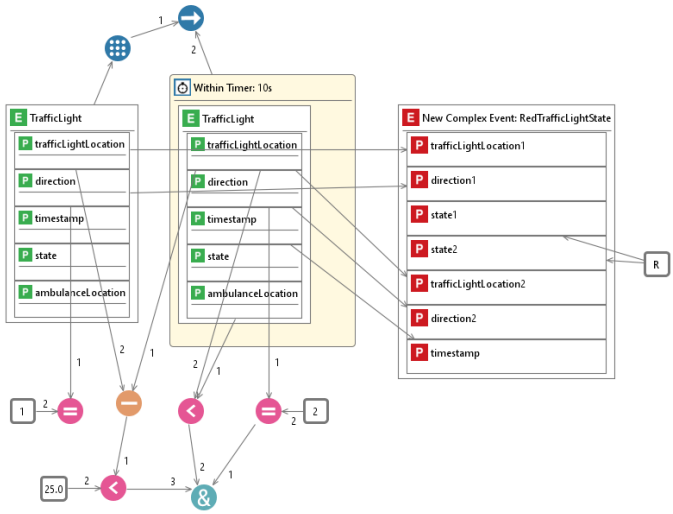

Figure 4. RedTrafficLightState pattern modeled with MEdit4CEP.

As previously explained, nITROGEN is a tool focused on generating data of different types, simulating the different sensors connected among the network.

The main window in nITROGEN let us set several features. First of all, we are going to create groups of channels of

communication. This can be done by following the next steps:

1. We create a group by clicking ADD button in the top left-hand side of Figure 5.
2. After creating a group, at least one channel should be created in it by clicking ADD button below the group's config label.

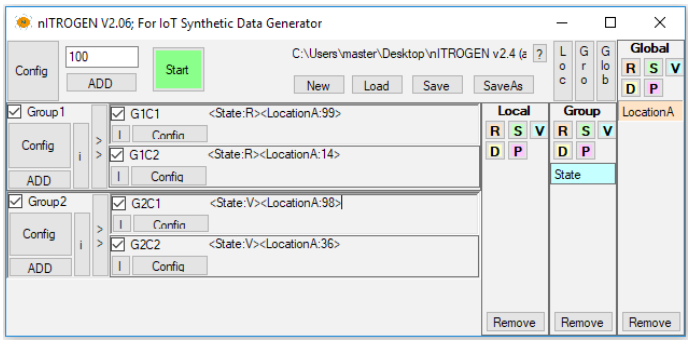

Figure 5. Several channels configured in nITROGEN.

As Figure 5 shows, we have 3 levels of variables — local, group and global— (see the right-hand side of the figure). In each level we can create one or more variables of random data for any of the 4 available types: R—Random Numeric—, S—String—, V—List of values— and D—dates and timestamps.

1. The local level is specific to a particular channel. A channel is a single flow of data submitted to a target element. Every channel can be composed of several variables as well as static information. It can then be configured pressing its config button, as later explained and can be enable or disable at any time marking or unmarking the tick box, respectively.
2. In the group level, any created variable maintains the same value for all the channels of the group. A group can be used to create together different channels which are submitting data to different target applications, but all channel data are submitted at the same time with the same frequency.
3. Finally, global level includes variables which can be used by any channel and that are taking the same value in any channel and group during every execution.

Configuring nITROGEN

In the configuration panel (see Figure 6), we can change the speed of events production by configuring the frequency variables. As shown in the figure, we can choose between number of ticks per second or per minute that will be used as the temporal unit. By default, the data for every group of channels is generated for every tick, but such frequency can be modified at any time.

We can also set when the simulation should stop: it can be done after some seconds or ticks, or we can follow on simulating events until it is stopped manually.

Finally, the refresh information is only for visual control, allowing us to define how often we want to refresh the data in the screen, regardless of the amount of generated data.

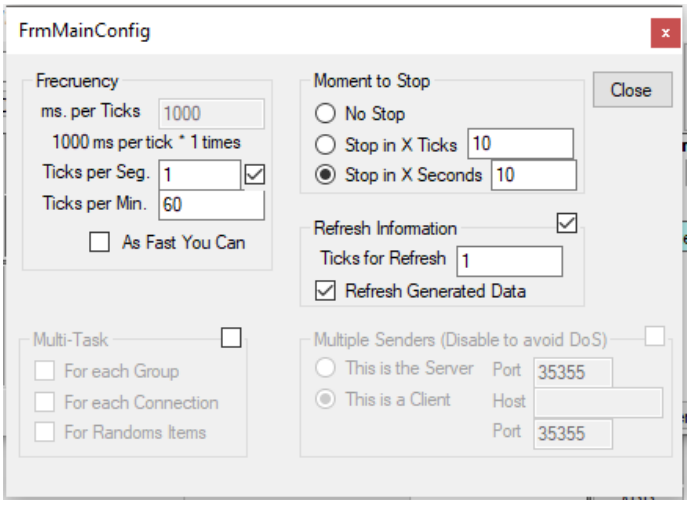

Figure 6. Configuration panel.

Example of Generated Data

Figure 7 shows the configuration of one of the four channels we created for the example of the narrow street.

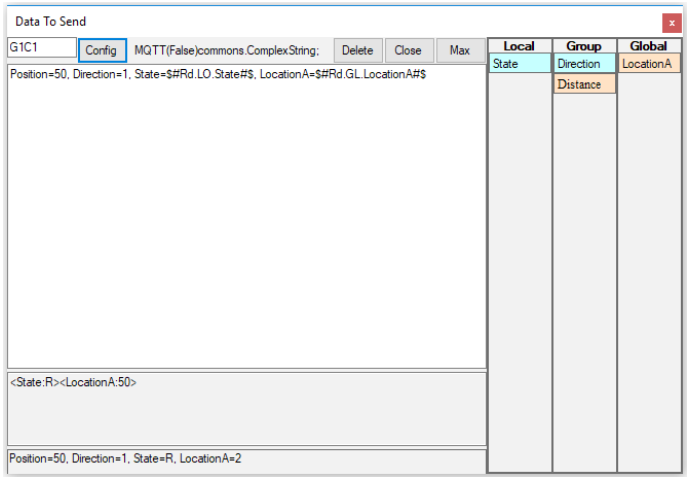

Figure 7. Configuration panel for the generated data of a panel.

For our case study, in both examples (avenue and narrow street) we need to generate the same type of data but with a different number of channels. This is because we need to stablish the location and direction of the traffic lights, which are set statically. The traffic light state (if it is red or green) is defined at group level, but the location of the emergency vehicle was defined as a global variable,

since the example provided is for one unique vehicle. The former is an array of characters, composed by R or G, representing both possible states, and the latter will be a random number generator configured to follow a particular distribution (see Figure 8).

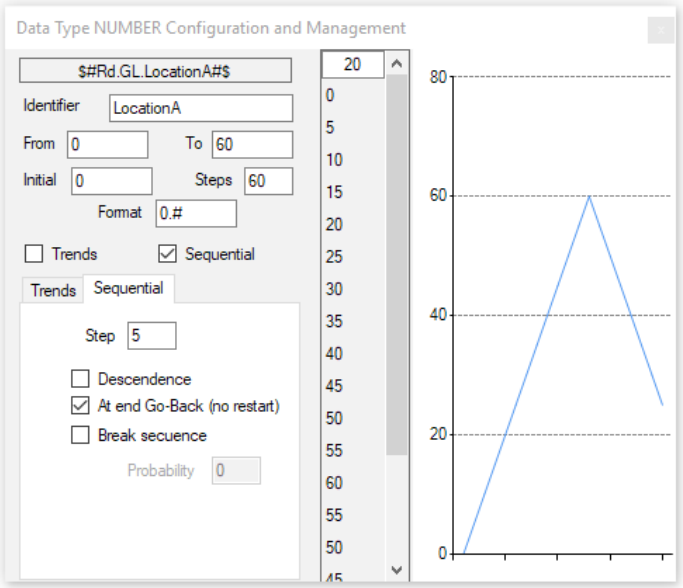

Figure 8. Configuration panel for the random number generator.

nITROGEN allow us to configure the target destination of the simulated data. In this particular case, the data will be sent to the message queues in the architecture (see Figure 9).

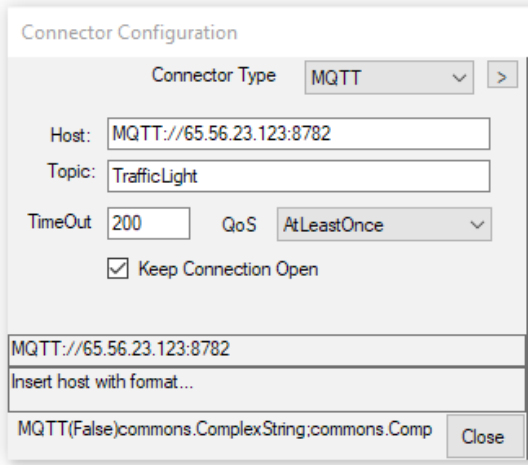

Figure 9. Configuration of data destination.

Please note that the student, who implemented the case study, decided to generate a fixed value for traffic-lights location, a sequential one for the relative position of the emergency vehicle, and a random one for the states of the traffic lights. Even though the student was aware the example did not reflect reality, he thought it was a good approach to test the event patterns detection. Additionally, the system can scale by setting several traffic lights per districts, where every district data is generated by one channel, so avoiding a centralized structure.

Results and future trends

By establishing the configuration mentioned before, we have obtained events with the following format, for instance, "*Location=50, Direction=1, State=R, LocationA=5*" . All the events created according to the configured rate have been immediately submitted automatically to the message queue and the system has immediately processed all the events related to emergency vehicles path and destination, as well as traffic light status in the vehicle way route. The patterns previously described have successfully detected in real time when a traffic light status had to be changed to facilitate the emergency vehicle advance and have submitted the corresponding alerts to the person in charge.

In the future, thanks to 5G technologies, we could achieve an acceptable response time for all the events and data that must be processed on real time in the proposed case study. Moreover, further complex event patterns could be defined by traffic experts for big cities, and could be deeply tested using nITROGEN before putting the system into production. This could lead not only to emergency vehicles saving time, but also to better traffic circulation, and therefore saving the citizens money and decreasing the pollution.

Additionally, other emerging technologies could be used in conjunction with those proposed here. The use of VANET (Vehicle Ad-Hoc Network) could improve the latency and traffic lights could react based on the signals sent from the cars in the road in real time.

Upon course finalization, the student feelings concerning the technologies used when learning about distributed architectures were twofold. On the one hand, they found of great interest the technologies studied, since there are not many available tutorials for their learning in the Internet, and the support of the teacher was key to acquire the knowledge. Besides, the fact of integrating several emerging technologies rather than using them isolated was recognized as an added-value in the learning process. Even more, the students are conscious of that they are learning emerging technologies and they are therefore obtaining skills demanded in the industry.

On the other hand, the experience with nITROGEN was found useful not only for the scope of the subject, but for a variety of case-studies, technologies and systems implementation. They found this tool especially useful for

connecting (in a simulated way) an increasing number of electronic devices to the network, just for sending the gathered data, or for processing such data having the chance of observing the results obtained in the system for several situations without the need of having real data. Thus, they highlighted the possibility of studying the viability of theoretical systems, opening the chance to create them in the future, once their viability was previously checked. The fact of being able to generate static and random data, according to a particular distribution, was considered as a great advantage together with the fact of not having limitation in the number of channels and amount of data to be simulated.

CONCLUSIONS

We have followed the proposed methodology during this academic year and the results were fully successful. The students were motivated, acquired a deep knowledge on emerging technologies and architectures for the IoT and were able to develop a full case study along the semester. Additionally, testing was also performed through the use of the nITROGEN tool, which let them simulate the sensors data required in the IoT case study. Thanks to the channel feature, which enables us to configure the information type, format and where the generated data is going to be sent, they could simulate as many devices as they want.

As an additional future activity in our methodology, we are planning to require students to present their project to a wider public, so that the attendees can vote the most interesting proposal for the IoT and smart cities. This way we encourage the students not only to implement a good project and prepare an adequate presentation, but also to continue with the project after the course is finalized if they get good feedback. Besides we plan to combine this proposal with additional innovation activities for the learning process (Ortiz, García de Prado, & Boubeta-Puig, 2017).

BIBLIOGRAPHIC REFERENCES

Agneeswaran, V. S. (2014). ***Big data analytics beyond hadoop: real-time applications with storm, spark, and more hadoop alternatives***. Upper Saddle River: Pearson Education.

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. ***Computer Networks***, ***54***(15), 2787–2805. Retrieved from https://www.cs.mun.ca/courses/.../IoT-Survey-Atzori-2010.pdf

Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2015). MEdit4CEP: A model-driven solution for real-time decision making in SOA 2.0. ***Knowledge-Based Systems***, ***89***, 97–112. Retrieved from https://www.semanticscholar.org/paper/MEdit4CEP%3A-A-model-driven-solution-for-real-time-in-Boubeta-Puig-Ortiz/4c49d26e3a676ac1ab75e958062e58177b782606

Buyya, R., & Vahid Dastjerdi, A. (2016). ***Internet of things: principles and paradigms***. Massachusetts: Morgan Kaufmann.

Dempsey, M. (2017). Teaching the Internet of Things. Retrieved from http://connectedtech.org/blog/teaching-the-internet-of-things/

European Research Group in the Internet of Things. (2012). The Internet of Things 2012 New Horizons. Retrieved from http://www.internet-of-things-research.eu/pdf/IERC_Cluster_Book_2012_WEB.pdf

García de Prado, A., & Ortiz, G. (2017). Experience on Introducing Parallel and Distributed Architecture Research Concepts in Computer Engineering Grade Students. In ***INTED2017 Proceedings.***

García De Prado, A., Ortiz, G., & Boubeta-Puig, J. (2017). CARED-SOA: A Context-Aware Event-Driven Service-Oriented Architecture. ***IEEE Access***, ***5***, 4646–4663. Retrieved from https://ieeexplore.ieee.org/iel7/6287639/7859429/07874075.pdf

García-de-Prado, A., Ortiz, G., & Boubeta-Puig, J. (2017). COLLECT: COLLaborativE ConText-aware service oriented architecture for intelligent decision-making in the Internet of Things. ***Expert Systems with Applications***, ***85***, 231–248.

Inzinger, C., Hummer, W., Satzger, B., Leitner, P., & Dustdar, S. (2014). Generic event-based monitoring and adaptation methodology for heterogeneous distributed systems: event-based monitoring and adptation for distributed systems. ***Software: Practice and Experience***, ***44***(7), 805–822. Retrieved from http://dsg.tuwien.ac.at/staff/inzinger/dl/SPE_2014_monina.pdf

Luckham, D. C. (2012). ***Event processing for business: organizing the real-time enterprise***. Hoboken: John Wiley & Sons.

Ortiz, G., García de Prado, A., & Boubeta-Puig, J. (2017). Fostering Learning through Media Games in Computer Science. ***EDULEARN 17 Proceedings*** (pp. 1576–1579).

Papazoglou, M. (2012). ***Web services and SOA: principles and technology*** (2nd ed). New York: Pearson Education.

Taylor, H. (Ed.). (2009). ***Event-driven architecture: how SOA enables the real-time enterprise***. Upper Saddle River: Addison-Wesley.