Design and Implementation of a Gateway for Pervasive Smart Environments

Diana C. Yacchirema, Manuel Esteve, Carlos E. Palau
Communications Department
Universitat Politècnica de València (UPV)
Valencia, Spain
diayac1@doctor.upv.es, mesteve@dcom.upv.es, cpalau@dcom.upv.es

Abstract— Wireless sensor and actuator networks play an important role in the Internet of Things (IoT) as they are key for the creation of Pervasive Smart Environments. These networks are developed through the interconnection of various heterogeneous devices. Due to the heterogeneity of the devices in terms of the communication protocols and the technologies used, the interconnection between them is still difficult. Particularly, interconnection and interoperability in IoT are two of the great challenges that needs to be faced in order to implement pervasive smart environments. We present the design and implementation of a Smart IoT Gateway which allows technical and syntactic interoperability in IoT using standard open protocols and standard data representation formats. We evaluated and demonstrated the effectiveness of the Smart IoT Gateway in terms of interoperability through its application to a use case in the industry sector.

Keywords— Internet of Things; Gateway; Interoperability; Interconnection; MQTT; CoAP.

I. INTRODUCTION

The Internet of Things (IoT) is considered the new technological paradigm and is composed of a large amount of independent devices, such as sensors, smartphones, surveillance video cameras and others. The number of connected devices grows at a rate of 30% per year and is expected to reach 50 billion in 2020 [1]. These predictions show a new way of interaction where any device at any time in any place can be connected to the Internet. The vision of IoT is to allow all these devices to communicate and interact with each other, [2][3] thus being able to obtain information from the environment, to process such information and to share it. Wireless Sensor and Actuator Networks (WSANs) are an alternative for detecting, obtaining information and acting on the physical environment. A WSAN is a network composed of tiny devices in which a variety of sensors and actuators are embedded [4]. These devices are able to obtain information on a specific environment, process and act on this information and communicate via wireless links.

However, due to the high degree of heterogeneity of protocols, communication technologies and data of the devices, the interconnection is not easy. More specifically, the interconnection and interoperability requirements are two of the great challenges that IoT must overcome in order to fully develop pervasive smart environments.

On the other hand, WSAN devices have limited resources regarding battery, communication ranges and bandwidth. These limitations hamper transmission and the sharing of the data obtained from long distance environments.

Without interoperability, 40% of the potential benefits of IoT would not be possible [5]. Among some current proposals to solve the interoperability of IoT (IoT Gateway [6], Smart IoT Gateway [7], Middleware for Smart gateways [8], Framework IoT [9], Smart Home Gateway[10] and UT-GATE [11]) only UT-GATE, allows interoperability on the level of communication protocols and data. As this Gateway is oriented towards a specific field, the data format is not uniform. On the other hand, Gateway does not contemplate the use of IoT protocols.

We propose the design and implementation of a Smart IoT Gateway named Gateway for Technical and Syntactic Interoperability (GW-TSI), which includes functions like transformation, processing, storing and data delivery. Additionally, GW-TSI also has functions for data forwarding and conversion of protocols, currently present in many existing proposals. These functions together with the use of open protocols, allow for the interoperability on a technical and syntactic level in an IoT architecture. Therefore, GW-TSI allows the development of application domain agnostic. We assess its effectiveness in an industrial use case, specifically in the transport and logistics field.

This paper is organized as follows: Section II discusses the related work in terms of their contribution to the technical and syntactic interoperability of IoT. Section III shows the design and implementation of GW-TSI based on open protocols for IoT. Section IV demonstrates the application of GW-TSI in a use case related to the transport and logistics field and Section V concludes the paper and points out the future working lines.

II. RELATED WORK

In this paper, we focus on technical and syntactic interoperability. We described some of the existing interoperability proposals (gateways, middlewares and frameworks) and the interoperability levels that they offer. Interoperability is the capacity of "*two or more systems to exchange data and use information*" [12]. In IoT, interoperability can be applied on different levels:

- Technical interoperability focuses on the communication protocols and the necessary infrastructure (hardware and software) to provide communication between devices.

- Syntactic interoperability is associated to the format of the data from the messages of the devices. The data transported by the protocols require specific syntax and encoding parameters.

- Semantic interoperability is related to the technologies required to allow that the information is understandable and shared by the different elements.

Authors in [6] [7] proposed gateways for the interconnection of WSNs with communication networks, such as the Internet. These gateways allow the interoperability of the different protocols, such as ZigBee, GPRS, Ethernet and RS485, through protocol conversion. Bimschas et al. [8] present, a middleware, which enables the execution of application code inside the gateway aiming towards the interconnection of WSNs and the Internet. Such middleware provides the interoperability between HTTP and CoAP by mapping the semantics between both protocols. Both of these proposals, however, do not implement functions like transformation, processing and data storage, which are necessary to provide intelligence to the Gateway and to enable interoperability.

Anon et al. [9] proposed an IoT framework based on Cloud Computing profiting from its calculation power and scalable storage capacity. This framework allows the interconnection and the interoperability between networks 802.15.4 and Cloud Computing. It also assigns a uniform format to messages between these two networks. The processing and storage of data take place in the Cloud, which causes an increase in latency and in bandwidth consumption.

Kim et al. [10] proposed a gateway for a specific field of application. This gateway allows the interoperability between different protocols of home networks, such as x10, Insteon, ZigBee and UPnp. On the other hand, as a disadvantage, it cannot be customized for different IoT applications.

Rahmani et al. [11] present, a Gateway which acts as a middle layer between the WSNs and Internet, thus delivering high level services like data central processing and local storage. The proposal aims to establish the interoperability on the level of communication protocols (WiFi, Bluetooth and 6LoWPAN) and data. The gateway, however, is oriented towards a specific IoT application, which implies that the format of data is also specific. On the other hand, the gateway does not contemplate the use of IoT protocols.

III. DESIGN AND IMPLEMENTATION OF GW-TSI

In order to establish the interoperability between devices, we use MQTT and CoAP protocols due to its many advantages when applied to IoT environments. Both protocols are open, lightweight and standard and have been designed for highly constrained environments, such as WSANs. These protocols allow interconnection and interoperability of devices working with different communication technologies and

protocols. They have different implementations that facilitate the development of pervasive smart environments.

A. Architecture

The challenges involved in the IoT and the works discussed above have shaped the design and implementation of GW-TSI presented in this proposal. (Fig. 1) The GW-TSI spans from the immediate needs of interconnection and interoperability between heterogeneous devices considering the limitations of their resources up to data exchange with the endpoints (e.g. services and applications) through the Internet. Standards for IoT interoperability are currently non-existent. We propose the use of standard open lightweight IoT protocols which provide interoperability between devices and the use of formats for data exchange between protocols.
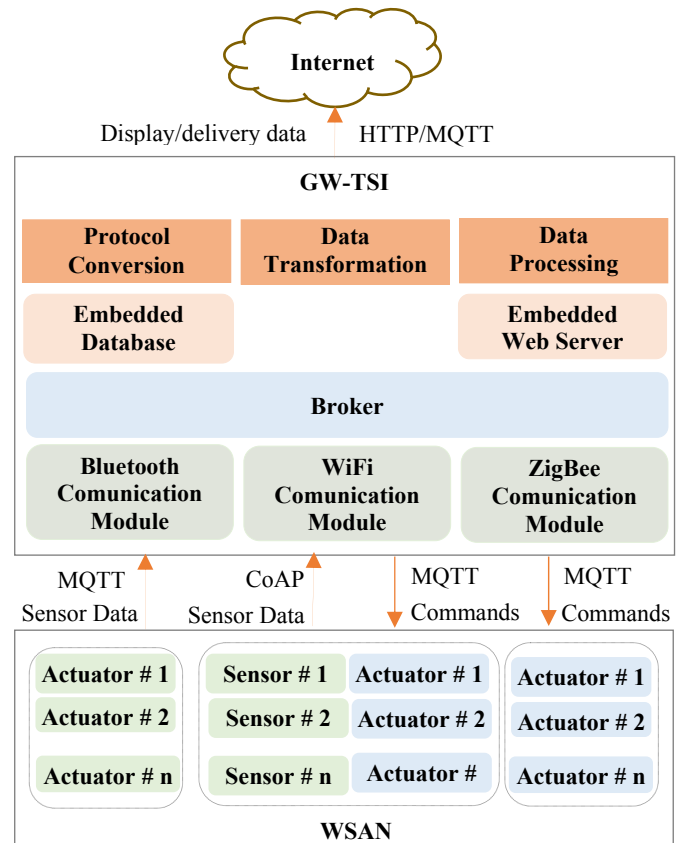

Fig. 1. The architecture of GW-TSI.

1) Sensor Nodes

WSANs are composed of large numbers of sensor devices and various type of actuators, which are distributed in a particular area. Sensors devices detect and gather data about the physical world while actuators perform actions to change the behaviour of the environment. These data are sent to the GW-TSI using optimal lightweight IoT protocols (CoAP and MQTT) to be used by devices and networks with limited resources. In IP-based WSANs (e.g. WiFi), CoAP protocol is used, whereas non-IP based WSANs (e.g. Bluetooth and ZigBee) use the MQTT protocol.

CoAP is a lightweight protocol that allows communication with high interoperability between devices and the easy integration of these devices into the existing network infrastructures. Its operation is similar to the HTTP protocol in that they both rely on the REST model and the basic operations (GET, PUT, POST and DELETE), with the difference that CoAP is used in devices and networks with low power and limited resources, where the use of protocol HTTP would be inefficient. CoAP supports reliability by using confirmable messages (CON) in response to a request.

MQTT is a communication protocol that uses the publish/subscribe model designed to connect devices, networks, applications, services and middleware. To establish the connection it uses a routing mechanism: one to one, one to many and many to many. It is therefore, ideal for interconnecting devices based on different technologies and communication protocols. This model is based on the definition of specific topics. In our architecture, such topics are associated to different types of sensors (temperature, humidity and gas). This model consists of three components: broker, subscriber and publisher. Sensor devices will perform as publishers by sending data to GW-TSI and the actuators devices and endpoints will perform as subscribers, which should have been previously subscribed to one or more topics of interest (Fig. 2 (1)). MQTT supports three levels of Quality of Service (QoS): The message will be delivered at most once, with no confirmation (QoS 0), the message will be delivered at least once, and with confirmation (QoS 1) and the message will be delivered exactly once by using a four-step handshake (QoS 2).

2) Gateway

The GW-TSI provides different communication modules that support the connection of heterogeneous devices, which work with protocols and wireless technologies such as WiFi, ZigBee and Bluetooth. GW-TSI receives the data from the devices of the non-IP based WSANs through MQTT protocol. In this case, GW-TSI must be initially subscribed to the topics that will be published by the sensors. On the other hand, the CoAP protocol is used to obtain the data from the devices of the IP-based WSANs. In this case, the sensors use the PUT method with CON message (Fig. 2 (2)) for sending the data to the GW-TSI. If the resource specified (/SensorType) by the request URI exists, then the CoAP server should respond with a 2.04 (Changed) Code, indicating that the resource has been changed according to indications included in the request. If the resource doesn't exist, the CoAP server creates a new resource and returns a 2.01 (Created) Code to the sensors (Fig. 2 (3)). The resources represent the different types of sensors.

The data transferred to the GW-TSI (Fig. 2 (4)) are heterogeneous in terms of format, data type and other parameters. To solve this problem, the GW-TSI converts these data into format JSON, through the Data Transformation function. JSON is the format chosen for the data format due to its advantages when compared to other more widespread formats for representation and exchange of data such as XML and, most important, it is a language that can be used in devices with limited resources and environments in which the volume of data can be huge.

The GW-TSI analyses and processes the data from the environment through the Data Processing function on the basis of predefined policies stored inside a database. If the data obtained meet the conditions established in the policies, an event is generated (Fig. 2 (5)). GW-TSI then detects the event and sends commands to the actuators (Fig. 2 (6)) so they perform actions on the environment and delivers notifications to the endpoints (Fig. 2 (7)) through the broker. Before then, GW-TSI, using the Protocol Conversion function, packages and maps the data to a specific format, according to the destination communication protocol.

The GW-TSI stores the data obtained by the sensor devices and the status value of the actuators in a database. These data are displayed through a web application using the HTTP protocol.

As a result, this architecture is capable of providing interoperability at a technical and syntactic level.
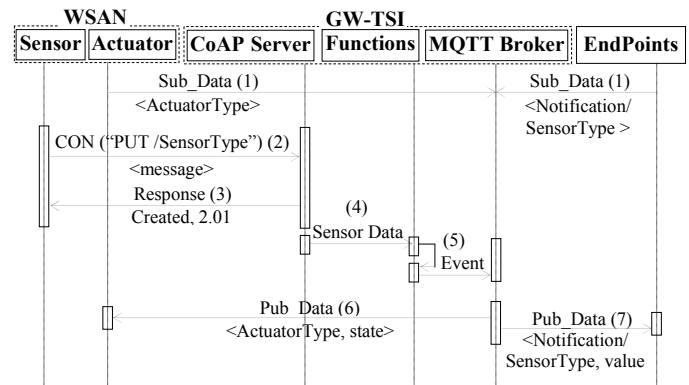

Fig. 2. Basic sequence diagram for sending data from devices of the IP-based WSANs to actuators and endpoints.

B. Hardware

Fig. 3, (a) shows the hardware structure of the GW-TSI, which consists of two main components: a processor and the ports and communication interface. The processor is the main component of GW-TSI, where the protocol conversion function is performed, as well as the transformation, processing, storage and data delivery. A Raspberry Pi 2 model B (Fig. 3 (a)) is used for the building of the gateway, which comprises an ARM Quad-Core processor of 900 MHz, 1GB RAM memory and 32 GB of storage memory on a SD card. Microcontroller Raspberry Pi 2 model B has been chosen because it is an open source hardware platform and because it offers the possibility of adding components and modules thus allowing for the gateway's scalability.

Ports and communication interfaces permit the connection of devices from the WSANs to GW-TSI. We opted to use wireless communication technologies: WiFi and ZigBee. USB WiFi 802.11 b/g/n with a speed of 150 Mbps is used to establish the connection to WiFi devices. Module 802.15.4 XBee low power mesh is made up of XBee Explorer Dongle and an XBee 2mW Wire Antenna series. It is this module which is used to connect the devices to GW-TSI through ZigBee technology.
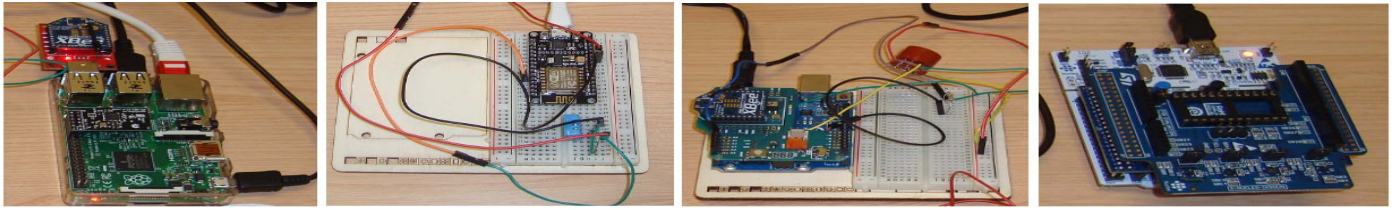
Fig. 3. (a) Hardware of GW-TSI. (b) Node Sensor WiFi. (c) Node Sensor ZigBee. (d) Node STM32 Nucleo L1

In this use case, we use the following wireless nodes: (i) NodeMCU V2 (Fig. 3 (b)) based on ESP8266, which uses ARM Cortex M4 de 32 bits as the CPU and 1-wire 802.11 b/g/n as the Wireless communication module. (ii) Arduino Uno (Fig. 3 (c)) which uses a XBee 2mW Wire Antenna series 2 connected to a Xbee Shield and (iii) STM32 Nucleo L1(Fig. 3 (d)), which uses a NUCLEO-L152RE board with an X-NUCLEO-IDS01A5 expansion board and X-NUCLEO-IKS01A1 sensor expansion board, integrated with temperature, humidity and motion MEMS sensors and actuators (leds).

C. Software

With the aim of implementing a flexible solution able to adapt to the requirements of any scenario of IoT application, all software used is open source. Our Gateway comprises different parts of software, with Raspbian as operating system, which is able to implement the functionality of GW-TSI in Python.

aiocap[1] was used to implement the CoAP server, embedded within the GW-TSI, thus allowing to obtain the data sent from the WiFi WSANs. aiocap is a CoAP-native implementation on Python based on RFC 5272.

Mosquitto[2] is a broadly adopted open source message broker that uses the protocol MQTT v3.1. Mosquitto is used for the implementation of the broker, which allows sending of commands to the actuators and delivering of notifications to the endpoints, as well as obtaining data sent from ZigBee WSANs.

MySQL was used to store data from the sensors and the state of the actuators. Apache is used for the configuration and implementation of HTTP Web server and PHP programming language to develop the web application, which allows the visualization of the data from the devices.

The program is a Python-written code, which executes the functions: transformation, processing and storage of data.

IV. EVALUATION

A. Use Case – Transport and Logistics

GW-TSI has been applied and evaluated in industry, more specifically in the transport and logistics field. Around 96% of companies in the transport sector consider that the IoT is a top key strategic initiative.

These companies will adopt IoT in this decade for the enhancement of their operational speed, with the aim of improving productivity and profit. GW-TSI will serve as the basis for the development an IoT architecture that allows monitoring of environmental conditions of freight transport with the aim of detecting and reporting in real time the changes that could pose a risk to the freight and even the driver's life.

A use case is implemented through the use of several GW-TSIs connected to each other in order to provide an IoT architecture which is hierarchic and scalable as shown in Fig. 4.
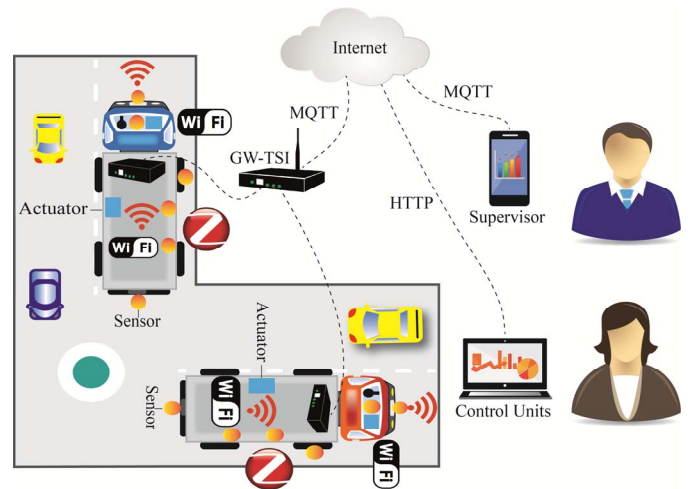

Fig. 4. Use Case of GW-TSI

Considering a scenario where several wireless devices use different protocols and communication technologies have been strategically distributed in a freight transport forming a WSAN. The GW-TSI can obtain a variety of data from the sensors regarding the environmental conditions of freight transport, such as humidity, temperature or gas emissions, using MQTT and CoAP protocols.

Data are transformed, processed and stored through the different functions implemented in the gateway. The GW-TSI detects risk situations, for example, temperature changes in the freight containers. In this case, the GW-TSI using the MQTT protocol sends commands to the actuators to they perform actions on the environment that help in prevention of risk situations. On the other hand, the GW-TSI will transmit the details concerning this situation: type of sensor, measurement, freight identification number or driver identification number to the control units, the supervisor and rest of the staff in charge, by sending of notifications. The GW-TSIs can be connected to

---

[1] https://aiocoap.readthedocs.org/en/latest/
[2] http://mosquitto.org/

each other through another GW-TSI, thereby obtaining a flexible and hierarchical architecture. On the endpoint side, control units or supervisors can visualize graphically the data coming from the different sensors from any mobile device - tablets or smartphone, or on a desktop computer, through a Web application hosted inside the GW-TSI using HTTP protocol.

B. Testing Environments

In order to verify the effectiveness of the GW-TSI functionality in a real environment, we evaluate both of the lightweight protocols (MQTT and CoAP) used for the interconnection and interoperability of the different devices in terms of the latency and bandwidth used. On several occasions, we have captured the traffic generated by sending 100 measurements from the sensor devices to the GW-TSI. We compared the two protocols in a reliable scenario, in which acknowledgments are used as a response to the messages sent by the sensor devices. CoAP uses confirmable messages (CON) and MQTT works with QoS 1.

The developed prototype includes sensor devices, actuators, GW-TSI, a wireless router and a laptop for initiating tests. Table I. presents the main characteristics of the devices used to deploy the prototype. Sensor devices are connected to the GW-TSI wirelessly via IEEE 802.15.4 and IEEE 802.11 and the actuators via IEEE 802.15.4. The laptop is connected to the GW-TSI through the WiFi channel. It represents an endpoint. The types of sensors used are: gas, temperature and humidity. The actuators are equipped with several LEDs that turn on in response to commands sent by the Gateway. The diagram of the test environment is shown in Fig. 5

The sensor devices (WiFi), actuators and laptop are connected to GW-TSI via Wireless router, while the sensor devices (ZigBee) are connected directly to the GW-TSI, which acts as ZigBee Coordinator. Both CoAP server and MQTT broker run on GW-TSI.
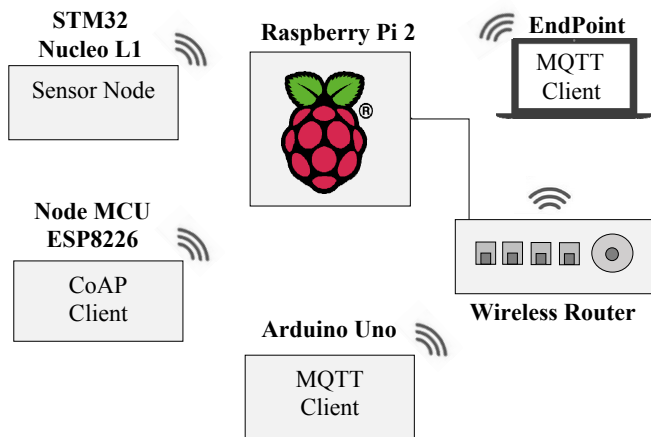

Fig. 5. Diagram of Test Environment

TABLE I.   DEVICES USED IN THE TEST ENVIRONMENT

Device	Specifications			
	CPU	*RAM*	*Interfaces*	*OS / Development platform*
Raspberry Pi (2 model B)	ARM Quad-Core 900 MHz	1 GB memory and 32GB SD storage memory	802.11 b/g/n, Xbee serie2 Serial (802.15.4)	Raspbian Jessie 4.1/ Contiki / Python
Node MCU ESP8226	ARM Cortex M4 32 bits	256k Flash memory, 64k RAM and 2k de EEPROM	802.11 b/g/n	Lua
Arduino Uno	ARM CortexM3 32 bits	32k Flash memory, 2k SRAM and EEPROM 1KB	Xbee serie2 (802.15.4)	Processing
STM32 Nucleo L1	STM32Cube firmware	-	802.15.4	Contiki

C. Results

1) Latency

Fig. 6, shows the values obtained from the Round Trip Time (RTT) of MQTT and CoAP protocols. CoAP is more lightweight than MQTT; the average RTT is 41% lower than MQTT; this is due to packet size and the use of the UDP protocol for the transport thereof. In the case of CoAP, an average 0.17 ms is needed to obtain confirmation of the delivery of a package compared with 0.29 ms that MQTT needs.

2) Bandwidth

Fig. 7, depicts the percentage of bytes transferred per second on each transaction between the MQTT Subscriber and MQTT Broker and between the CoAP client and CoAP server. The average number of bytes transferred by the MQTT protocol is 13% higher than the CoAP protocol due to the overhead introduced by the TCP connection establishment and termination.
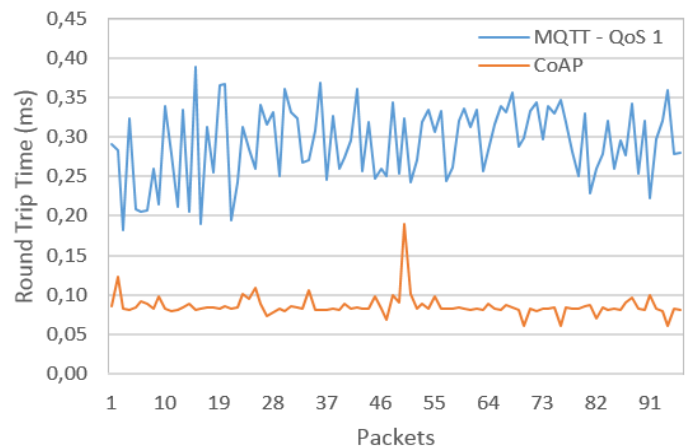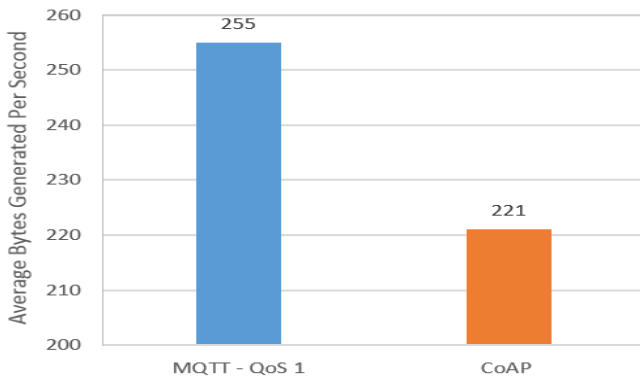

Fig. 6. Round Trip Time for MQTT and CoAP

Fig. 7. Average Bytes generated per second for MQTT and CoAP

Compared with the implementations of the IoT solutions designed to enable interconnection and interoperability of devices, our proposal is the only one which allows connectivity, interoperability on a technical and syntactic level, and which is able to support different protocols and wireless communication technologies. Table II, shows this comparison.

TABLE II. IOT SOLUTIONS AND SUPPORTED PROTOCOLS AND INTEROPERABILITY

IoT Proposals	Conectivity			IoT Protocols		Interoperability	
	WiFi	ZigBee	BLE	CoAP	MQTT	T[b]	S[c]
[6]	x					x	
[7]						x	
[8]	x	x		x		x	
[9]		x	x			x	
[10]		x				x	
[11]	x	x	x			x	x[a]
GW_TSI	x	x	x	x	x	x	x

a. Means for Interoperability not using a standard format

b. T: Technical Interoperability

c. S: Syntactic Interoperability

V. CONCLUSIONS

Two of the main challenges in IoT are to ensure the interconnection and interoperability between heterogeneous devices working with different hardware and software and various technologies and communication protocols. Solving the problem of interoperability will allow us to create pervasive smart environments. In this paper, a new Smart IoT Gateway named GW-TSI has been implemented, acting as a key element to allow the sending of information from heterogeneous environments such as WSANs to end systems or applications over the Internet. We demonstrated that GW-TSI allows establishing interoperability on a technical and syntactic level in an IoT architecture that responds to multiple real-world scenarios. Technical interoperability is achieved through open, lightweight and standard protocols in the field of IoT such as MQTT and CoAP and syntactic interoperability by using standard formats for the exchange of messages between different communication protocols. Gateway functionality is evaluated through the application to a use case of logistics and transport. For future work, we aim to extend the functionalities of the Smart IoT gateway, so it may allow the administration of devices and incorporate security mechanisms for the transport of information.

REFERENCES

[1] CERP-IoT Cluster of European Research Projects on the Internet of Things, "Vision and Challenges for Realising the Internet of Things," European Union, 2010.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] A. Guerrieri, L. Geretti, G. Fortino, and A. Abramo, "A service-oriented gateway for remote monitoring of building sensor networks," *2013 IEEE 18th Int. Work. Comput. Aided Model. Des. Commun. Links Networks, CAMAD 2013*, pp. 139–143, 2013.

[5] Mckinsey Global Institute, "the Internet of Things : Mapping the Value Beyond the Hype," 2015.

[6] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IOT Gateway: BridgingWireless Sensor Networks into Internet of Things," in *EEE/IFIP International Conference on Embedded and Ubiquitous Computing*, 2010, pp. 347–352.

[7] S. Guoqiang, C. Yanming, Z. Chao, and Z. Yanxu, "Design and implementation of a smart IoT gateway," in *Green Computing and Communications (GreenCom), IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, 2013, pp. 720–723.

[8] D. Bimschas, H. Hellbrück, and R. Mietz, "Middleware for smart gateways connecting sensornets to the internet," in *Proceedings of the 5th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks. ACM.*, 2010, pp. 8–14.

[9] F. Anon, V. Navarathinarasah, M. Hoang, and C.-H. Lung, "Building a Framework for Internet of Things and Cloud Computing," *2014 IEEE Int. Conf. Internet Things(iThings), IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput.*, no. iThings, pp. 132–139, 2014.

[10] J. E. Kim, G. Boulos, J. Yackovich, T. Barth, C. Beckel, and D. Mosse, "Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes," in *Eighth International Conference on Intelligent Environments*, 2012, pp. 206–213.

[11] A. Rahmani, N. K. Thanigaivelan, T. N. Gia, J. Granados, B. Negash, P. Liljeberg, and H. Tenhunen, "Smart e-Health Gateway : Bringing Intelligence to Internet-of-Things Based Ubiquitous Healthcare Systems," in *Consumer Communications and Networking Conference (CCNC), 12th Annual IEEE*, 2015, pp. 826–834.

[12] IERC, "IoT Semantic Interoperability:Research Challenges, Best Practices, Recommendations and Next Steps," 2011.