Article

COIIoT – An Interface between CoAP and OSGP Protocols for the Integration of Internet of Things Devices with Smart Grids

Felipe Viel [1,*], Luis Augusto Silva [1], Valderi Reis Quietinho Leithardt [1,3,4*], Gabriel Villarubia González [2], Raimundo Celeste Ghizoni Teive [5] and Cesar Albenes Zeferino [1,*]

[1] Laboratory of Embedded and Distribution Systems, University of Vale do Itajaí, Rua Uruguai 458, C.P. 360, 88302-901 Itajaí, Brazil; {viel, valderi, zeferino}@univali.br, luis.silva@edu.univali.br
[2] Expert Systems and Applications Lab, Faculty of Science, University of Salamanca, Plaza de los Caídos s/n, 37008 Salamanca, Spain; gvg@usal.es
[3] Departamento de Informática, Universidade da Beira Interior, 6201-001 Covilhã, Portugal
[4] COPELABS, Universidade Lusófona de Humanidades e Tecnologias, 1749-024 Lisboa, Portugal
[5] Laboratory of Applied Intelligence, University of Vale do Itajaí, Rua Uruguai 458, C.P. 360, 88302-901 Itajaí, Brazil; rteive@univali.br
* Correspondence: viel@univali.br (F.V.), valderi@univali.br (V.L.), zeferino@univali.br (C.Z.)

Abstract: The evolution and miniaturization of the technologies for processing, storage, and communication have enabled computer systems to process a high volume of information and make decisions without human intervention. Within this context, several systems architectures and models have gained prominences, such as the Internet of Things (IoT) and Smart Grids (SGs). SGs use communication protocols to exchange information, among which the Open Smart Grid Protocol (OSGP) stands out. In contrast, this protocol does not have integration support with IoT systems that use some already consolidated communication protocols, such as the Constrained Application Protocol (CoAP). Thus, this work develops the integration of the protocols OSGP and CoAP to allow the communication between conventional IoT systems and systems dedicated to SGs. Results demonstrate the effectiveness of this integration, with the minimum impact on the flow of commands and data, making possible the use of the developed CoAP-OSGP Interface for Internet of Things (COIIoT).

Keywords: internet of things; smart grids; protocol communication; interoperability; CoAP; OSGP

---

1. Introduction

The term Internet of Things (IoT) was first proposed by Kevin Ashton in 1999 to draw the attention of P&G (Procter & Gamble) executives to the use of Radio-Frequency Identification (RFID) in supply chains and establish a relationship with the Internet [1]. The International Telecommunication Union (ITU) defines IoT as a global infrastructure for the information society, enabling advanced services by interconnecting things based on interoperable information and communication technologies [2]. This infrastructure also involves technologies for acquisition, storage, and processing of embeddable data [3–6]. The term IoT fits into this context and encompasses any object, device, machine, and utensil with which we interact and depend on [7].

The main features of infrastructure in IoT systems are [3]: (*i*) *device heterogeneity*, given the amount and diversity of applications and data types that are required; (*ii*) *resource constraint*, such as limited power supply and processing capacity; (*iii*) *spontaneous interaction*, for example, from users and objects; (*iv*) *ultra-wide-scale communication network*, to cater for all nodes and the large number of events that can happen; (*v*) *dynamic networks*, to allow a constant connection; (*vi*) *data context*, to process data and information; (*vii*) *intelligence*, for decision making; and (*viii*) *application environment*, to learn how to process the data.

 doi:10.20944/preprints202001.0170.v1

The primary features of IoT applications include [3]: (*i*) *diversity*, as different applications have different requirements and probably require varying architectures; (*ii*) *realtime*, to perform actions according to the time requirements of the target application; (*iii*) *security and privacy*, to protect applications, networks and users; and (*iv*) *service model*, with the Everything-as-a-Service (XaaS) model being very efficient, scalable and user-friendly [8,9]. IoT applications are diverse and some typical applications include healthcare, Industry 4.0, logistics, and smart cities [3]. The latter is further subdivided into traffic, sanitation, agriculture, security surveillance, and Smart Grids (SGs), which are the focus of this article.

SGs are different from traditional power grids, which are structures with the only function of transmitting and distributing electricity from remote power plants to end consumers. On the other hand, SGs present the idea of intensive use of information and communication technology in the electrical network through the possibility of communication of the status of various components on the network. This feature enables the implementation of strategies for control, operation, and improvement of the system in a much more efficient manner than those currently existing.

Several standards can be used for IoT communication, such as Constrained Application Protocol (CoAP) and Message Queue Telemetry Transport (MQTT). These protocols can be applied to a general IoT context, but are not recommended or standardized for use in SGs, as the Open Smart Grid Protocol (OSGP) is. A solution is necessary to integrate those protocols and enable data exchanges between general application protocols and specific application protocols such as OSGP.

Given the heterogeneity of general and specific applications and protocols, we identified a gap regarding the mapping of data packets between CoAP and OSGP. This mapping would enable IoT systems applied to residential and industrial plants to obtain and provide information to SGs. Within this context, this paper presents the integration between the CoAP and OSGP communication protocols in a solution that we call CoAP and OSGP Integration for the Internet of Things (COIIoT). To the best of our knowledge, this is the first adaptation between CoAP and OSGP.

The remainder of this article is organized as follows. Section 2 provides a brief background on communication protocols with an emphasis on CoAP and OSGP. Next, Section 3 discusses some related work. Section 4, in turn, presents the architecture of the proposed integration architecture and the materials and methods employed in its development and evaluation. Following, Section 5 presents and discusses the results obtained in the experiments. Finally, in Section 6, we give the final remarks and discuss future work.

2. Background

2.1. Smart Grids

SGs are introducing a new paradigm for electrical power systems. This new concept intrinsically carries some characteristics indicated in [10], which are: intelligence, efficiency, accommodation, quality, resilience, reliability, and green. According to these authors, SGs will effectively be a complex Cyber-Physical System (CPS) that will take advantage of multiple embedded intelligent technologies to increase the safety and reliability of the power system.

There are four major changes that mark the implementation of SGs [11]:

- *Advanced metering infrastructure*: i.e., the intelligent meters.
- *Intermittent and renewable energy sources*: consumers will be able to supply energy to the network, leading to an additional bidirectional flow of energy, control, and communication in the electrical grid. This bi-directional flow of energy and information is an important characteristic of SGs that we can consider as an overlapping of the Internet on the electrical network.
- *Electric vehicles*: the use of this kind of vehicle will influence the function and reliability of distribution systems.

 doi:10.20944/preprints202001.0170.v1

- *Cyber-physical systems*: even though in any level of the electrical power system contains some aspect of automation, communication, and software, transforming the current electrical grid into a true CPS requires a massive infusion of technology.

Furthermore, SGs have some peculiar characteristics[11]:

- *Self-recovery*: the ability to automatically detect, analyze, respond, and restore failures in the network.
- *Empowerment of the consumer*: the ability to include equipment and behavior of consumers in the processes of planning and operation of a network.
- *Tolerance to external attacks*: the ability to mitigate and resist physical and cyber-attacks.
- *Quality of energy*: providing electricity with the quality demanded by the digital society.
- *Accommodate a large variety of sources and demands*: the ability to integrate transparently (plug-and-play) a variety of energy sources of various sizes and technologies.
- *Reduce the environmental impact of the electrical production system*: reducing losses and using sources of low environmental impact.
- *Make feasible and benefit from competitive energy markets*: favor the retail market and microgeneration.

The term empowerment reflects the importance that the consumer will have in the SGs as a continuous source of information and data. At the same time, SGs will be able to receive information and commands from the electrical utility, based, for example, on a demand response program, which is one of the several services available in a SG.

As highlighted in [12], SGs offer communication control and services that are capable of executing dynamic energy management, Advanced Metering Infrastructure (AMI) with Automatic Meter Reading (AMR) devices, and demand response services. Concerning automatic meter reading, in this study, the authors present results involving the use of the data from the AMR installed in a SG in Korea to generate clusters of typical load curves, which will serve to classify consumers who do not have AMR.

Demand response is considered an attractive way to reduce peak demand and obtain ancillary services in power systems. With SGs, the demand response services will be facilitated by the automation of infrastructure from the distribution network, by the possibility for bidirectional communication, and by the energy management programs or demand-side management, implemented in the SGs. In this case, the use of electronic meters with embedded intelligence is essential for the success of this type of program.

The innovations present in SGs, such as electronic meters, communication networks, Phasor Measurement Unit (PMU) sensors, and control devices (intelligent relays), in addition to integration of renewable energy sources and electrical vehicles, will increase the vulnerability of the electrical grid to cyber-attacks, thus increasing the urgency and importance of cyber-security studies, as stressed by [13]. In this sense, the authors of [10] emphasizes that cyber-security will be a necessary service in SGs and highlight three main aspects: reliability, integrity, and availability. These authors also pointed up the characteristics that the intelligent networks must have against cyber-attacks, involving the ability to identify, react to attacks, and prevent difficulties (attacks, intrusions, and failures) in real-time.

2.2. IoT Communication Protocols

In IoT, communication devices and networks are not isolated but connected and integrated to form a computer network. This feature brings the need for regular communication within the computer network, but it is constrained by the devices that make up IoT systems, which has limited power supply and storage and processing capacities.

Traditional protocols, such as Transmission Control Protocol (TCP), do not deal well with the limitations imposed by IoT devices, given the overheads generated by the process layers in these protocols. Furthermore, these protocols face addressing issues with each device. One of the alternatives to solve this problem was the adoption of Internet Protocol v6 (IPv6) and its new concepts, such as IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), conceived of the idea of assigning an address to computationally restricted devices [14]. To this end, the Internet Engineering Task Force (IETF) and other standards bodies have defined and developed application protocols for resource-constrained devices. Examples of protocols developed by IETF are CoAP and Routing Protocol for Low Power and Lossy Networks (RPL) [15].

In addition to CoAP, other protocols are used to develop solutions for IoT, especially when focusing on Machine-to-Machine (M2M) communication. These alternatives may include: (*i*) MQTT; (*ii*) Extensible Messaging and Presence Protocol (XMPP); (*iii*) RESTful Services; (*iv*) Advanced Message Queuing Protocol (AMQP); (*v*) Data Distribution Service (DDS); and (*vi*) OSGP. The two most common types of communication in IoT are *request/response*, as in CoAP, and *publish/subscribe*, used in MQTT.

The MQTT protocol was introduced by IBM in 1999 and standardized by OASIS in 2013. This protocol provides built-in connectivity between applications, middleware, networks, and communication technology. MQTT is asynchronous, based on *publish/subscribe* communication, and is made up of a broker (who contains topics) and several clients (who publish or subscribe to the topics). A client can send data to a topic or receive data from a topic it subscribes as a publisher updates this topic. This protocol has reliability on three Quality of Service (QoS) levels: (*i*) *fire and forget*; (*ii*) *delivered at least once*; (*iii*) *delivered exactly once*. MQTT executes over TCP, and thus security is performed using the services of this protocol [16–19].

2.3. Constrained Application Protocol

CoAP is a synchronous *request/response* protocol developed for resource constrained devices. The protocol uses the methods GET, POST, PUT, and DELETE from Hypertext Transfer Protocol (HTTP), allowing these two protocols to work together [20]. The commands used by CoAP allow interactions in a client/server architecture. CoAP operates on the User Datagram Protocol (UDP) to keep its implementation lightweight. Because it works over UDP, CoAP implements its reliability mechanisms by using two bits in the packet header to define the message type and the QoS level. The messages can be: (*i*) *commitable*; (*ii*) *unverifiable*; (*iii*) *recognition*; and (*iv*) *reset* [15,17,18]. Fig. 1 simply illustrates the protocol stack of CoAP by identifying its two layers.

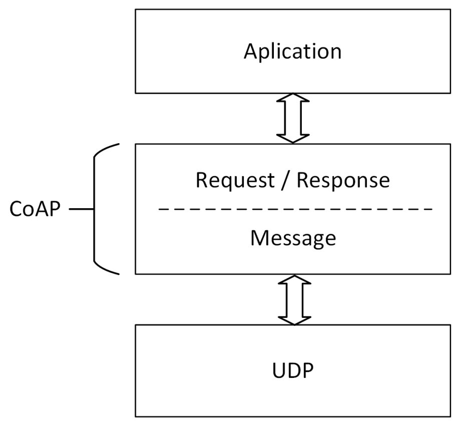

Figura 1. Simplified layered stack from CoAP (adapted from [20]).

2.4. Open Smart Grid Protocol

OSGP is a reference architecture for communication between devices operating over SGs [21]. The main purpose of this architecture is to provide greater control of electricity consumption and supply between customers and service providers in order to provide useful information to consumers of these services. OSGP is standardized by European Telecommunications Standards Institute (ETSI) and its layers are defined by the ETSI GS OSG 001, ISO/IEC 14908 and ETSI TS 103 908 specifications.OSGP

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 January 2020 doi:10.20944/preprints202001.0170.v1

is designed to work on a variety of SG devices. In order to avoid collisions, the protocol uses a master-slave architecture, and the nodes are not able to hear each other. Therefore, OSGP does not support overlapping transactions, and procedures must be executed strictly one at a time, waiting for a first result of the procedure before triggering the next.

OSGP is divided into three main layers: (*i*) *Application* (ETSI GS OSG 001) [21]; (*ii*) *Network* (ISO/IEC 14908); and (*iii*) *Physical* (ETSI TS 103 908). Fig. 2 illustrates the protocol stack and the corresponding seven layers of the Open System Interconnection (OSI) model. The protocol is of *request/response* type [22,23].

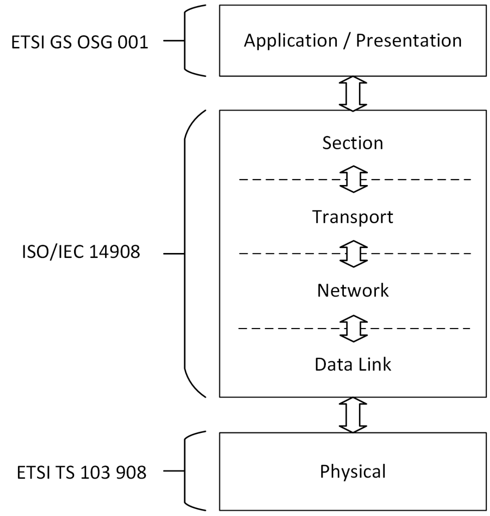

Figura 2. Simplified layered stack of OSGP (adapted from [24]).

3. Related Work

We can find in the literature a set of works that present the adaptation of CoAP to communication protocols used in SGs, such as the ETSI M2M, Distributed Network Protocol 3.0 (DNP3.0) and International Electrotechnical Commission (IEC) 61850 protocols. As we can see in the works summarized in Table 1, some techniques used for protocol adaptation include native Application Interface Programming (API) and Uniform Resource Identifier (URI) mapping. These solutions are analyzed below.

Tabela 1. Related work.

Work	IoT	SG	Adaptation
[25]	CoAP	ETSI M2M	Native API
[26]	CoAP	DNP3.0	URI mapping
[27]	CoAP	IEC 61850	URI mapping
[28]	CoAP	IEC 61850	URI mapping

In [25], the authors present the ETSI M2M system that addresses some issues in SGs identified and discussed in the literature. The work adopts CoAP as its native application layer protocol and uses it to carry all messages. Such services are made available using an open and native API. CoAP is used for communication with the transmission system, the distribution system, and consumers. The service-oriented demonstration prototype enables integration of a variety of M2M devices and utilizes a variety of communication technologies such as IEEE 802.15.4, 3G and GSM/GPRS.

The authors of [26] present an integration between CoAP and DNP3.0 protocols in order to implement a gateway to support CoAP in SGs; the integration allows communication M2M communication. To perform the integration between the protocols, the authors implemented a mapping layer as an interface. This interface is required because both protocols use layers that offer services

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 January 2020 doi:10.20944/preprints202001.0170.v1

with different protocols. DNP3.0 uses High-level Data Link Control (HDLC) and CoAP uses UDP. The authors used the GET and PUT methods of CoAP mapped, respectively, to the READ and WRITE methods of DNP3.0 to test and evaluate their implementation. A comparison of CoAP with Simple Object Access Protocol (SOAP) showed that the former performs better than the latter because SOAP adopts TCP, even using DNP3.0 services through adaptations.

An integration of CoAP with the IEC 61850 protocol for SGs is presented in [27]. In the paper, the authors also applied a mapping layer approach to tailor message exchange between CoAP and IEC 61850, which uses TCP as the transport layer. The methods offered by both protocols are mapped to perform the conversion and enable message exchange. A gateway is responsible for protocol integration through a mapping layer. On conversion, the IEC 61850 protocol information model is converted to a URI of the CoAP protocol. In traffic tests, the authors observed that packet delay in the network was negatively affected, having a higher latency than in the individual evaluation of the protocols.

In [28], the authors review protocols possible to be integrated with IEC 61850. Through this review, the authors decided to use CoAP because it is a standard already used in IoT and for being targeted at resource-constrained devices. The mapping of CoAP (GET, PUT, POST, and DELETE) methods to IEC 61850 methods is performed using CoAP URI. For adaptation between the protocols, the authors developed modules that have specific functions and reduce the complexity of integration. The authors state that their implementation is better than that of [27] because it covers all Abstract Communication Service Interface (ACSI) services.

The works described above demonstrate the relevance of adapting CoAP for use in conjunction with SGs systems. According to the reported literature, the integration of a widely used IoT protocol developed for resource-constrained devices allows increasing the insertion of devices into the system. This integration enables a broader range of functionalities to be made available to SGs, as well as adding higher levels of management and automation to consumers and service providers.

In this article, we introduce the integration between CoAP and OSGP, which also allows the IEC 61850 and OSGP protocols to exchange messages through CoAP. As we mentioned before, the proposed solution is named COIIoT, the acronym for 'CoAP and OSGP Integration for Internet of Things.' The next section describes its architecture and how we implemented and evaluated it.

4. Materials and Methods

4.1. Development

We employed the default frameworks of CoAP and OSGP without any modification. For the integration between the protocols, we applied a function mapping layer, individually assigning each type of request/response. We defined the mapping between packets so that the OSGP packet travels internally in CoAP and vice-versa. The proposed model acts as a gateway, serving for the integration between the protocols, and its structure is shown in Fig. 3. CoAP uses UDP on the transport layer to travel to the gateway, while OSGP travels through its specific protocols to SGs, i.e., ISO/IEC 14908, and ETSI TS 103 908.

We also addressed the mapping of requests and responses and used two packet types in OSGP: *request* (which is categorized as Full or Partial) and *response*. For CoAP, we implemented the PUT, GET, and POST methods and use the MicroCoAP [29] library for implementation. This library was chosen because it has a small size and comprises all the methods necessary for this work. The entire development was done using C language to enable future implementations on resource-constrained devices such as microcontrollers.

As specified by the application layer [21], the OSGP transmission packet has all the message fields in Most Significant Byte (MSB) format, with the *data* field being the only exception. For security reasons, a sequence number and a summary are added by the security mechanism of OSGP in all messages. For pending tables, all *count* and *length* fields include the Pending Event Descriptor (PED),

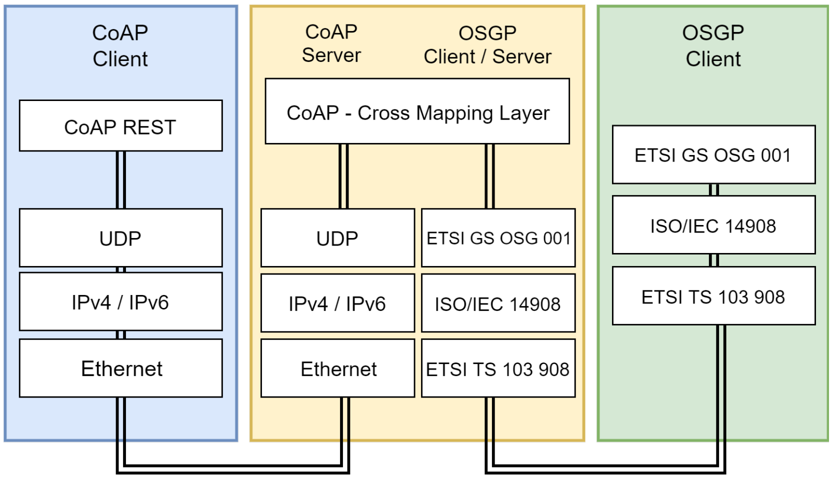

Figura 3. COIIoT architecture.

when it is present. The current version of COIIoT does not yet fully implement the support for pending tables, as it is not a requirement for communication. However, this version already provides the treatment of this structure in code if it is requested.

The checksum is calculated based on the PED and *data* fields. For partial reads and writes, offset is not affected by the presence of the PED. For example, to read 4 bytes of offset 3 from a pending table, *count* would equal 10, and offset would equal 3 [30]. The device using OSGP knows by the table ID whether to expect PED or not.

Fig. 4 shows the packet mapping between CoAP and OSGP. In this figure, the client CoAP sends a message directed to the SG network via COIIoT) interface. CoAP executes a GET method using the MicroCoAP library, which is mapped to an OSGP Partial Read request. At the mapping, the CoAP-based packet extracts the request type, the message identifier, and the packet size, and then analyzes the received message. Afterward, the mapping layer matches the request received from CoAP to a request to be sent to OSGP. The payload of the CoAP packet is mapped into two fields of the OSGP packet. The first field (*count*) contains the message size, and the second field (*offset*) encloses the contents of the message.

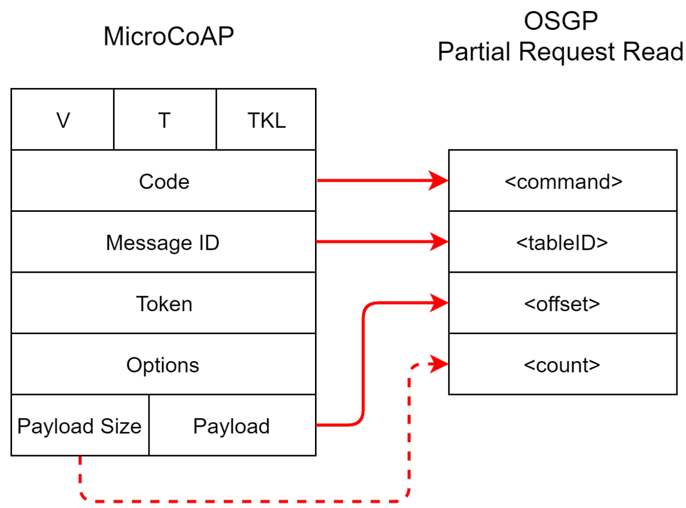

Figura 4. Mapping a request packet from CoAP to OSGP.

In Fig. 5, the OSGP packet inserted into PED for pending queue management is shown; it will be included in the *options* field of CoAP. The *count* field reports the message size and is mapped directly to the *payload size* field of the CoAP packet. Among the services mapped between CoAP and OSGP,

doi:10.20944/preprints202001.0170.v1

some of them depend on information from other protocols and implementations, such as pending events from OSGP.

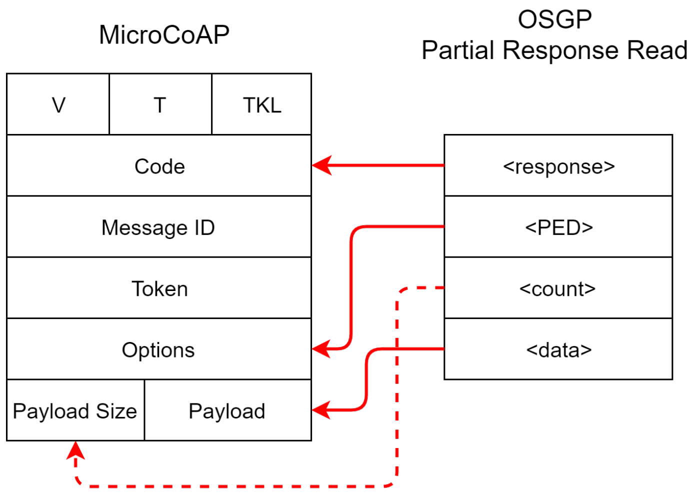

Figura 5. Mapping a response packet from OSGP to CoAP.

Algorithms 1–4 describe the methods used to map requests from the CoAP domain to the OSGP domain. For instance, in Algorithm 1, the *tableID* parameter, which is as an identifier in OSGP, takes the protocol header identifier from CoAP. The command that will be executed is defined by code and remains unchanged. The mapping still checks against PED because if the OSGP packet implements this attribute, it is mapped to a CoAP parameter. Following, Algorithms 5–8 (shown in Appendix A) describe the methods developed to map requests from the OSGP domain to the CoAP domain. Finally, Algorithms 9–12 (also shown in Appendix A) describe the methods designed to map the responses to the requests received by each domain. It is worth noting that the same method is used to map the responses for partial and full write requests, as well as for partial and full read requests.

Algorithm 1: PUT_to_Partial_Write_request

Input : CoAP PUT packet
Output: OSGP APDU partial write request

if *CoAP packet* $\leq$ *OSGP APDU write request* **then**
    APDU.command $\leftarrow$ write_partial;
    APDU.tableID $\leftarrow$ packet.id;
    APDU.offset $\leftarrow$ packet.payload_offset;
    APDU.count_bytes $\leftarrow$ packet.payload_size-3;
    APDU.pedding_event $\leftarrow$ NULL;
    if *PED* **and** *packet.options* $=$ *Size*1 **then**
        APDU.pedding_event $\leftarrow$ packet.options;

foreach *payload_data i* $\in$ *packet* **do**
    APDU.payload $\leftarrow$ packet.payload;

return OSGP APDU partial write request

Algorithm 2: POST_to_Full_Write_request

Input : CoAP POST packet
Output: OSGP APDU full write request

```
if CoAP packet ≤ OSGP APDU write request then
    APDU.command ← write_full;
    APDU.tableID ← packet.id;
    APDU.count ← packet.payload_lenght;
    APDU.pedding_event ← NULL;
    if PED and packet.options = Size1 then
        APDU.pedding_event ← packet.options;
foreach payload data i ∈ packet do
    APDU.payload ← packet.payload;
return OSGP APDU full write request
```

Algorithm 3: GET_to_Partial_Read_request

Input : CoAP GET packet
Output: OSGP APDU partial read request

```
if CoAP packet payload lenght ≤ OSGP APDU read request then
    APDU.command ← partial_table_read;
    APDU.tableID ← packet.id;
    APDU.count ← packet.payload_lenght;
    APDU.offset ← packet.payload.data;
return OSGP APDU partial read request
```

Algorithm 4: GET_to_Full_Read_request

Input : CoAP GET packet
Output: OSGP APDU full read request

```
APDU.tableID ← packet.id;
APDU.command ← full_table_read;
return OSGP APDU full read request
```

We can see from the algorithms that packets from both protocols have their fields redistributed to the other protocol during mapping. This approach differs from the one used in the works mentioned above, which apply URI mapping.

It is worth mentioning that the algorithms presented above (and in Appendix A) do not represent all the constraints imposed by the programming language and the libraries used, which mainly add complexity in the implementation.

4.2. Evaluation

As a test scenario, we used a CoAP client named Copper for the Mozilla Firefox browser and a client implemented using the libCoAP library. The test environment that simulated a gateway featured the Debian GNU/Linux 8 (Jessie) 64-bit operating system, Intel® Dual-Core™2.5 GHz 64-bit processor and 4 GB of main memory.

 doi:10.20944/preprints202001.0170.v1

In testing, we measured the time to perform mapping on each communication, and verified and characterized function by function. For testing, we used varying OSGP packet sizes up to 114 bytes (the limit specified by the protocol). For CoAP, we adopted the same packet size in all communications, i.e., 512 bytes.

It is noteworthy to mention that our evaluations were performed on mapping methods only and did not take the function calls into account. This approach was used because this work describes a mapping layer that integrates a larger IoT system. Another aspect that we did not address in this work is the security of the transmitted data because it does not impact the interface operation directly. We consider that security is a requirement that can be addressed directly at the sender and receiver ends, which is beyond the scope of this work.

5. Results

Table 2 summarizes the results obtained from the experiments, presenting the memory requirements for every packet and the latency of each mapping function. We can observe that the OSGP packets require less memory than the CoAP packets, which has the same size for all the functions. It is worth mentioning that all the tests were performed using packets with a 6-byte payload. Regarding performance, we can observe that the most expensive communication is the one that comprises the mapping of a CoAP PUT request to an OSGP Partial Write request (790 ns), and its corresponding response (470 ns). This worst case mapping consumes 1.260 ms for request and response. The experimental results also show that the response algorithms, although relatively simple in their implementation, have a relatively high impact when compared to request methods. This cost is due in part to the library used to implement the CoAP protocol.

Tabela 2. Packets size and latency of mapping methods.

Type	Method	Algorithm	CoAP (Bytes)	OSGP (Bytes)	Latency (ns)
Request	PUT → Partial Write	1	512	112	790
Request	POST → Full Write	2	512	88	520
Request	GET → Partial Read	3	512	64	375
Request	GET → Full Read	4	512	24	275
Request	Partial Write → PUT	5	512	112	646
Request	Full Write → POST	6	512	88	390
Request	Partial Read → GET	7	512	64	196
Request	Full Read → GET	8	512	24	265
Response	Write → PUT/POST	9	512	8	470
Response	Read → GET	10	512	72	440
Response	PUT/POST → Write	11	512	8	286
Response	GET → Read	12	512	72	310

The low latency observed in the results above is due to the type of gateway used in our experiments, a desktop computer that plays the role of a server present in the environment automated by IoT technologies. This approach allows for a low interface impact in conjunction with other features that may be present in home automation systems, for example. However, it is worth noting that a desktop-based gateway has high power consumption, and we could employ more efficient solutions based on low-power embedded processors. Nevertheless, this approach would increase mapping latency, and this trade-off should be evaluated considering the requirements of the target application.

 doi:10.20944/preprints202001.0170.v1

6. Conclusions

This work presented COIIoT, a mapping interface for the integration between CoAP and OSGP. The goal of this interface is to enable the data exchange between IoT devices used in home and industry applications and a SG infrastructure. The developed interface comprises a set of mapping functions that translates the methods used in each protocol and has a low cost that enables its use with low impact in communication. Furthermore, the mapping interface reduces the time necessary for application development as it abstracts the complexity involved in the communication between the protocols.

It is worth highlighting that latency is a crucial aspect to be taken into account in SGs, especially concerning the communication between automation devices of a power distribution network. Thus, despite being preliminary, the results obtained in this work indicate the kind of communication protocol influences the latency.

As future work, we intend to apply COIIoT in a physical testbed to evaluate it in a real application. Our goal is to obtain metrics regarding performance and energy consumption when running it COIIoT on microcontroller-based IoT devices communicating with a SG infrastructure.

Author Contributions: Conceptualization, F.V., L.S., V.L., and C.Z.; methodology, F.V. and L.S.; validation, F.V. and L.S.; writing, original draft preparation, F.V., L.S., V.L., and C.Z; writing, review and editing, C.Z., R.T. and G.G.

Funding: This research was funded in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES) – Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) – Processes 315287/2018-7 and 436982/2018-8, and Fundanção de Amparo à Pesquisa do Estado de Santa Catarina (FAPESC) – Grant 2019TR169. Supported by project 'Plataforma de Vehículos de Transporte de Materiales y Seguimiento Autónomo' – Target. 463AC03.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ACSI	Abstract Communication Service Interface
AMI	Advanced Metering Infrastructure
AMR	Automatic Meter Reading
AMQP	Advanced Message Queuing Protocol
API	Application Interface Programming
CoAP	Constrained Application Protocol
COIIoT	CoAP and OSGP Integration for Internet of Things
CPS	Cyber-Physical System
DDS	Data Distrisbuition Service
DNP3.0	Distributed Network Protocol 3.0
ETSI	European Telecommunications Standards Institute
HDLC	High-level Data Link Control
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPv6	Internet Protocol v6
ITU	International Telecommunication Union

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 January 2020 doi:10.20944/preprints202001.0170.v1

M2M	Machine-to-Machine
MQTT	Message Queue Telemetry Transport
MSB	Most Significant Byte
OSGP	Open Smart Grid Protocol
OSI	Open System Interconnection
PED	Pending Event Descriptor
PMU	Phasor Measurement Unit
QoS	Quality of Service
RFID	Radio-Frequency Identification
RPL	Routing Protocol for Low Power and Lossy Networks
SG	Smart Grid
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
XaaS	Everything-as-a-Service
XMPP	Extensible Messaging and Presence Protocol

Apêndice A

This appendix present the algorithms designed for the methods implemented to map the requests from the OSGP domain to the CoAP domain (Algorithms 5–8), as well as the responses between the two domains (Algorithms 9–12).

Algorithm 5: Partial_Write_to_PUT_request

Input : OSGP APDU partial write request
Output: CoAP PUT packet

packet.version $\leftarrow$ CoAP Version;
packet.request_type $\leftarrow$ PUT;
packet.hdr.code $\leftarrow$ WRITE;
packet.id $\leftarrow$ APDU.tableID;
packet.command $\leftarrow$ APDU.command;
packet.options $\leftarrow$ 0;
packet.payload.data $\leftarrow$ APDU.offset $\vee$ APDU.payload;
return CoAP packet PUT request

Algorithm 6: Full_Write_to_POST_request

Input : OSGP APDU full write request
Output: CoAP POST packet

packet.version $\leftarrow$ CoAP Version;
packet.request_type $\leftarrow$ POST;
packet.hdr.code $\leftarrow$ WRITE;
packet.id $\leftarrow$ APDU.tableID;
packet.command $\leftarrow$ APDU.command;
packet.options $\leftarrow$ 0;
packet.payload_length $\leftarrow$ APDU.count_bytes;
packet.payload.data $\leftarrow$ APDU.payload;
return CoAP packet POST request

 doi:10.20944/preprints202001.0170.v1

Algorithm 7: Partial_Read_to_GET_request

Input : OSGP APDU partial read request
Output: CoAP GET packet

```
packet.version ← CoAP Version;
packet.request_type ← GET;
packet.hdr.code ← READ;
packet.id ← APDU.tableID;
packet.command ← APDU.command;
packet.options ← 0;
packet.payload.data ← APDU.offset;
return CoAP GET packet request
```

Algorithm 8: Full_Read_to_GET_request

Input : OSGP APDU full read request
Output: CoAP GET packet

```
packet.version ← CoAP Version;
packet.request_type ← GET;
packet.hdr.code ← READ; packet.id ← APDU.tableID;
packet.command ← APDU.command;
packet.options ← 0;
packet.payload.data ← NULL;
return CoAP GET request
```

Algorithm 9: Write_to_PUT_POST_response

Input : OSGP APDU write response
Output: CoAP packet

```
if APDU response = OK then
    packet.hdr.code ← Valid Code;
else
    packet.hdr.code ← Invalid Code;
return CoAP packet
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 January 2020 doi:10.20944/preprints202001.0170.v1

Algorithm 10: Read_to_GET_response

```
Input  : OSGP APDU partial read response
Output: CoAP packet
if APDU response = OK then
    packet.hdr.code ← Valid Code;
else
    packet.hdr.code ← Invalid Code;
packet.payload_lenght ← APDU.count_bytes;
if APDU PED ≠ NULL then
    packet.options.num ← Size1;
    packet.options.buffer_data ← APDU.PED;
    packet.options.buffer_size ← APDU.PED_size;
    packet.hdr.num_opts ← 1;
if APDU bytes > MAX_SIZE and table read then
    return ERROR;
else
    packet.payload.data ← APDU.data;
return CoAP packet
```

Algorithm 11: PUT_POST_to_Write_response

```
Input  : CoAP packet
Output: OSGP APDU write response
if CoAP code = write complete then
    return OSGP APDU write complete
else
    return OSGP APDU write error
```

Algorithm 12: GET_to_Read_response

```
Input  : CoAP packet
Output: OSGP APDU read response
if CoAP code = write complete then
    APDU.count_bytes ← packet.payload_length;
    APDU.payload ← packet.payload.data;
    return OSGP write complete
else
    return OSGP write error
```

References

1. Ashton, K. That 'internet of things' thing. *RFiD Journal* **2011**, *22*.
2. Wortmann, F.; Flüchter, K. Internet of things. *Business & Information Systems Engineering* **2015**, *57*, 221–224. doi: 10.1007/s12599-015-0383-3.
3. Razzaque, M.A.; Milojevic-Jevric, M.; Palade, A.; Clarke, S. Middleware for internet of things: a survey. *IEEE Internet of Things Journal* **2016**, *3*, 70–95. doi: 10.1109/JIOT.2015.2498900.
4. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of things for smart cities. *IEEE Internet of Things journal* **2014**, *1*, 22–32. doi: 10.1109/JIOT.2014.2306328.
5. Da Xu, L.; He, W.; Li, S. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics* **2014**, *10*, 2233–2243. doi: 10.1109/TII.2014.2300753.
6. Vargas, D.C.Y.; Salvador, C.E.P. Smart IoT gateway for heterogeneous devices interoperability. *IEEE Latin America Transactions* **2016**, *14*, 3900–3906.
7. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials* **2014**, *16*, 414–454. doi: 10.1109/SURV.2013.042313.00197.
8. Banerjee, P.; Friedrich, R.; Bash, C.; Goldsack, P.; Huberman, B.; Manley, J.; Patel, C.; Ranganathan, P.; Veitch, A. Everything as a service: Powering the new information economy. *Computer* **2011**, *44*, 36–43. doi: 10.1109/MC.2011.67.
9. Viel, F.; Silva, L.A.; Leithardt, R.V.; Zeferino, C.A. Internet of Things: Concepts, Architectures and Technologies. 2018 13th IEEE International Conference on Industry Applications (INDUSCON). IEEE, 2018, pp. 909–916.
10. Green, R.C.; Wang, L.; Alam, M. Applications and trends of high performance computing for electric power systems: Focusing on smart grid. *IEEE Transactions on Smart Grid* **2013**, *4*, 922–931.
11. Falcão, D.M. Smart Grids e Microgrids: the future is already present. 2009 Simpósio de Automação de Sistemas Elétricos (SIMPASE), 2009, pp. 1–11. (in Portuguese).
12. Kim, Y.I.; Ko, J.M.; Choi, S.H. Methods for generating TLPs (typical load profiles) for smart grid-based energy programs. 2011 IEEE Symposium on Computational Intelligence Applications In Smart Grid (CIASG). IEEE, 2011, pp. 1–6.
13. Giani, A.; Bitar, E.; Garcia, M.; McQueen, M.; Khargonekar, P.; Poolla, K. Smart grid data integrity attacks. *IEEE Transactions on Smart Grid* **2013**, *4*, 1244–1253.
14. Silva, R.; Leithardt, V.R.; Silva, J.S.; Geyer, C.; Rodrigues, J.; Boavida, F. A Comparison of Approaches to Node and Service Discovery in 6lowPAN Wireless Sensor Networks. Proceedings of the 5th ACM Symposium on QoS and Security for Wireless and Mobile Networks; ACM: New York, NY, USA, 2009; Q2SWinet '09, pp. 44–49. doi: 10.1145/1641944.1641954.
15. Sheng, Z.; Yang, S.; Yu, Y.; Vasilakos, A.; Mccann, J.; Leung, K. A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities. *IEEE Wireless Communications* **2013**, *20*, 91–98. doi: 10.1109/MWC.2013.6704479.
16. Stanford-Clark, A.; Truong, H.L. Mqtt for sensor networks (mqtt-sn) protocol specification. *International business machines (IBM) Corporation version* **2013**, *1*.
17. Jaffey, T. MQTT and CoAP IoT Protocols. *Eclipse Newsletter* **2014**.
18. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials* **2015**, *17*, 2347–2376. doi: 10.1109/COMST.2015.2444095.
19. Salman, T. Internet of Things Protocols and Standards, 2015.
20. Shelby, Z.; Hartke, K.; Bormann, C. The constrained application protocol (CoAP), 2014.
21. ETSI. GS OSG v1.1.1 (2012-01) - Open Smart Grid Protocol (OSGP), 2012.
22. Albano, M.; Ferreira, L.L.; Pinho, L.M. Convergence of Smart Grid ICT architectures for the last mile. *IEEE Transactions on Industrial Informatics* **2015**, *11*, 187–197. doi: 10.1109/TII.2014.2379436.
23. Kursawe, K.; Peters, C. Structural weaknesses in the open smart grid protocol. Availability, Reliability and Security (ARES), 2015 10th International Conference on. IEEE, 2015, pp. 1–10. doi: 10.1109/ARES.2015.67.
24. ISO/IEC. ISO/IEC 14908-1: information technology – control network protocol – part 1: protocol stack. Standard, 2012.

25. Lu, G.; Seed, D.; Starsinic, M.; Wang, C.; Russell, P. c. Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE. IEEE, 2012, pp. 148–153. doi: 10.1109/WCNCW.2012.6215479.
26. Shin, I.J.; Eom, D.S.; Song, B.K. The CoAP-based M2M gateway for distribution automation system using DNP3 in smart grid environment. Smart Grid Communications (SmartGridComm), 2015 IEEE International Conference on. IEEE, 2015, pp. 713–718. doi: 10.1109/SmartGridComm.2015.7436385.
27. Shin, I.J.; Song, B.K.; Eom, D.S. International Electronical Committee (IEC) 61850 Mapping with Constrained Application Protocol (CoAP) in Smart Grids Based European Telecommunications Standard Institute Machine-to-Machine (M2M) Environment. *Energies* **2017**, *10*, 393. doi:10.3390/en10030393.
28. Iglesias-Urkia, M.; Urbieta, A.; Parra, J.; Casado-Mansilla, D. IEC 61850 meets CoAP: Towards the integration of Smart Grids and IoT standards. *ACM* **2017**. doi: 10.1145/3131542.3131545.
29. Iglesias-Urkia, M.; Orive, A.; Urbieta, A. Analysis of CoAP implementations for industrial Internet of Things: A survey. *Procedia Computer Science* **2017**, *109*, 188–195.
30. ETSI. Open Smart Grid Protocol (OSGP): Smart Metering/Smart Grid – ETSI TS 104 001 V2.2.1 (2019-01), 2019.