Analysis of an edge-computing-based solution for local data processing at secondary substations

Author: Néstor Rodríguez Pérez
Master in Smart Grids
Universidad Pontificia de Comillas
Madrid, Spain
nestor.rp@alu.comillas.edu

Academic Supervisor: Miguel Ángel Sanz Bobi
Telematics and Computer Science Dept.
Universidad Pontificia de Comillas
Madrid, Spain

Industrial Supervisor: Aurelio Sánchez Paniagua
AMI, *i-DE (Iberdrola Group)*
Madrid, Spain

***Abstract*—This project analyses the technologies proposed in the open Edge Workload Consolidation architecture (defined by Intel and Minsait) and their alternatives, considering the specific requirements and perspective of a power distribution utility (i-DE) to implement it at secondary substations. The analysis is focused on the edge node (hardware requirements and software), communications (inner between applications, between the node and the management system, and between the node and other devices) and the management system (requirements, trends, and deployment process). Furthermore, the advantages, challenges and possible functionalities in the LV distribution grid are discussed, as well as the economic impact that it would have on the system and on the electric utility.**

To support the analysis and extract practical conclusions, a local test environment is used for technologies comparisons and for the development and testing of a real-time balance computing application programmed in Python, for what a data generator is developed in Node-RED. To check the deployment process, a remote test environment, provided by Minsait, is used, deploying a modified version of the balance application to process S02-like reports, commonly provided by data concentrators at secondary substations.

The analyzed architecture is found to be appropriate and the solution, thanks to its modularity and its open source basis, has the potential to be the new paradigm of how the LV grid is monitored and controlled, providing great benefits to the utility, to the system, and to the electric industry in general.

Keywords— edge computing, smart grid, power distribution, Smart Secondary Substation, microservices

I. INTRODUCTION

In the recent years, the massive deployment of smart metering and the use of big data in the central systems of electric distribution utilities have constituted the main steps towards a smarter distribution grid, considerably increasing the amount of information gathered as well as the centralized computing and storage requirements. The implementation of edge computing, following an Internet of Things (IoT) and microservices approach, in the distribution grid, could alleviate the load of the central system and the communications infrastructure and, at the same time, provide semi-real time control and monitoring capabilities, reasons why it is being considered as the next step towards a smarter grid.

In Europe, the OpenNode project [1] could be considered the first attempt of implementing a modular edge computing solution to achieve what is known as a "Smart" Secondary Substation. Later, Siemens introduced Gridlink [2], a communications infrastructure to deploy Java applications, testing it with a voltage control application using OLTC transformers. In its "Technical Guide about Smart Secondary Substations" [3], ABB only considers specialized smart devices, but not different modular applications ("microservices") in the same device. Without entering into architectural details, [4] discusses the applications, advantages, and challenges of edge computing in distribution networks. [5] defines an edge computing architecture that proves to reduce the transmission bandwidth and delay in communications for the same number of devices when compared to a cloud architecture (centralized) and [6] directly proposes a fog-computing-based architecture (which is based on edge computing) for the distribution grid that uses the MQTT protocol (very used in IoT) and Node-RED as programming tool.

On February 2020, Intel and Minsait (Indra) published the open Edge Workload Consolidation Architecture (eWLCA) [7] that specifies the use of different open source technologies (e.g. MQTT protocol, InfluxDB database, Docker, Kubernetes, etc.) for the deployment of microservices in the edge. Minsait and the Spanish electric distribution utility i-DE (Iberdrola Group) are developing a proof of concept of edge computing at secondary substations (SSs) following this architecture.

This project analyses the technologies proposed in the eWLCA and their possible alternatives, considering the requirements of the power distribution utility (e.g. operation, security, etc.); its advantages, challenges, and possible functionalities in the LV distribution grid, and the economic impact that the implementation of this solution would have on the entire system and on the utility, in order to determine the convenience (or not) of applying edge computing at SSs with this architecture. To carry out this analysis and to extract practical conclusions, two tests environments are used: one local and one remote. The local environment (Virtual machine with Ubuntu 18.04.4, 3.018 GB RAM and 30,75 GB SSD) is used to compare technologies and to develop and test a real-time balance application developed in python, for what a simple data generator in Node-RED is also developed. The remote environment, provided by Minsait, is used to test the remote deployment process and to test a modified version of the balance application to process S02-like reports that are commonly provided by data concentrators at SSs.

For the analysis, three main elements (Fig. 1) are distinguished in the solution based on the eWLCA:

- The **Edge Node** (edge compute device, intelligent node). It includes the hardware installed in the SS and the corresponding software architecture (e.g. OS, virtualization technology, databases, etc.).
- The **Management System** (on cloud or on premises). To monitor and manage the deployment of microservices (i.e. applications, functionalities) on the Edge nodes.

- **Communications**. Three environments must be distinguished: inner communications between applications in the node; communications between the node and the management system; and communications with other devices (in or out of the SS).

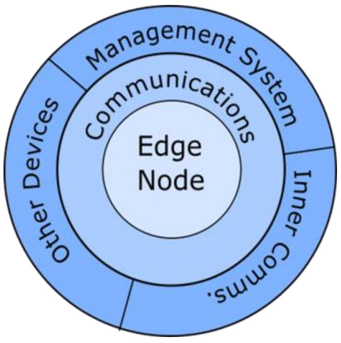


Fig. 1. Overall representation of the architecture.

II. Analysis

A. *Advantages, challenges and functionalities of the solution*

The **advantages** that are expected from this solution are numerous. The processing of data at SSs reduces the delay in communications and decisions and provides autonomy to the SS in case of communication failures. The use of virtualization technologies (e.g. Docker) allows the isolation between applications (and therefore, to ease their debugging) and the possibility of customizing the functionalities deployed per type of SS. These functionalities (i.e. microservices) deployment would be done from the central system, avoiding the need of a technician's displacement to the SS, and hence reducing the time between development and production (*Agile methodology*).

The eWLCA architecture is designed to decouple software from hardware, which will significantly increase the competitiveness for the development of microservices, reducing costs for the utility, and will provide flexibility when implementing the solution, since different hardware with different functionalities could be deployed based on the type of SS *("Customization" of functionalities*). The use of a unique device for multiple functionalities (i.e. microservices) will inevitably lead to the future substitution of some of the devices currently present in SSs (e.g. data concentrators, low voltage supervisors…). In terms of operation, many functionalities could be deployed for a better control and monitor of the LV grid, resulting in an ultimate improvement of the quality of service. Besides, this solution can be the driver for the development of innovative and disruptive functionalities, and current communications between SSs (same "LAN" through optical fiber or PLC) could be used to make the Edge Nodes work together for high computing processes (e.g. image processing, complex machine learning models, etc.).

Not only the analyzed solution will present advantages but also some **challenges** to be faced. Initially, the main challenge will be hardware related. This solution will imply a new device in the SS, so problems of space might appear. It will also be complex to determine the adequate relation between computing capacity and cost, as well as to find a device in the market that also complies with the requirements to be installed in SSs (e.g. electric isolation tests, temperature, etc.). In terms of operation, some disruptive functionalities where this solution could mean a significant impulse are not fully regulated yet (e.g. demand response, energy storage at distribution level, EV integration, etc.). Furthermore, some functionalities require data provided by smart meters. Currently, this data might be incomplete, inaccurate, or even unavailable due to temporary disconnections of smart meters. These communications with other devices should be improved in reliability and speed to take full advantage of the solution.

Regarding the possible **functionalities** of the edge-computing-based solution, they cover different areas of an electric distribution utility. **Grid knowledge and analysis:** phase connectivity algorithms, calculus of the impedance of LV lines, fraud detection algorithms and power losses accountability. **Grid monitoring**: detection, location and classification of LV faults, and state estimation of the LV grid. **Asset management and maintenance**: predictive maintenance applications, security functionalities (e.g. thermal cameras, movement detectors, etc.) and demand prediction for the SS. **Flexibility and innovative functionalities**: monitoring and control of distributed generation and energy storage, EV integration and demand response schemes. These types of functionalities still need strong regulation. In all likelihood, this overall list of functionalities will grow significantly when the solution gets tested on field successfully.

B. *Analysis of the Edge Node*

1) Virtualization Tecnhology

Three virtualization technologies are analyzed: Virtual Machines (VMs), Containerization (i.e. Docker) and Unikernels (Fig. 2)

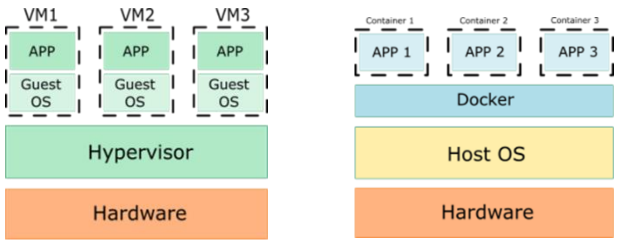


Fig. 2 General Virtual Machine (left) and containerization (right) architectures

The deployment of functionalities using exclusively **standard VMs** (a complete OS per VM, Fig. 2) would be extremely challenging despite its maturity and the high level of security and isolation provided, since each VM would have RAM and storage allocated by the hypervisor not only to run the specific application but to run an entire OS that contains some libraries that will never be used by the application [8]. The resulting images are very large (GBs) and have long development times. Besides, as multiple applications are expected to be deployed, the resources consumption in the device would be highly inefficient [8] and the maintenance of the OS would require more than one update per node.

Containerization (Docker) presents better characteristics where VMs lack. It allows the deployment of isolated applications under the same host OS (Fig. 2) and images are lighter (MBs). It also shows good security level from a local point of view (when there are no external intermediaries, such as public repositories) [9]. In terms of intellectual property security, docker images should be built retrieving access to their sh and bash console, with a non-root user, and with their source code obfuscated. There are also commercial tools to manage user's permissions (e.g. Docker Enterprise). Docker is very popular among developers because it is relatively simple, open source (free community version) and reduces time between development and production. Tools such as Docker Compose and Kubernetes have helped in this popularity growth.

Finally, in the last decade **Unikernels** have emerged as a virtualization alternative. Unikernels are similar to VMs but with an optimized kernel instead of an entire OS. Images are even lighter than containers [10] but more difficult to debug. Tests carried out by [11] showed that Unikernels could achieve better response times but similar performance to containers with heavy workloads. The biggest drawback is that this technology is at an early stage of development for production environments [9] and is still too complex for the average developer [12] (i.e. an OS expert is needed).

Therefore, based on this discussion, the recommended technology to be used in the Edge Node is **containerization using Docker**, which is also the technology defined in the eWLCA and the one used by Minsait for its PoC with i-DE.

2) *Operating System (OS)*

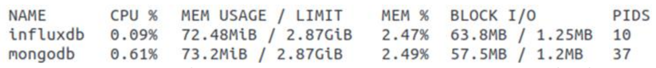
Docker needs a Linux kernel in order to be executed. Therefore, Windows Server is discarded since it would be necessary a Linux VM or to use Hyper-V isolation. The use of a Linux OS has the advantage of its fast vulnerabilities' detection and correction by the community (open source). Among the Linux OS available, it is recommended the use of one already homologated by the electric utility (RedHat or RedHat Oracle Linux in the case of i-DE) so that the maintenance can be carried out by the utility easily.

3) *Database*

JSON is one of the most used formats in IoT architectures to represent resources and for information exchange. Current STG-DC (i.e. central system - data concentrator) reports are in XML format, which is a bit complex to read and requires more time to be processed. Besides, other devices might use other formats, so JSON can be the common format to be used in the Edge Node. In fact, the Spanish utilities are working together (FutuRed working groups) to define a common JSON schema based on the Web of Things Architecture (WOT-A), sponsored by W3C [13]. Therefore, the database finally used must be appropriate to work with JSON documents.

The use of a relational (SQL) database is discarded since there are No-SQL databases optimized to store documents like JSON and a relational database usually requires more time to query data for semi-real time applications [14].

Among the non-relational (No-SQL) database types, the document-oriented (MongoDB) and the time series (InfluxDB) types are the most appropriate. Both MongoDB and InfluxDB were installed as containers on the local test environment to compare their functioning. In this environment, MongoDB consumes 6 times more CPU and have 4 times more open processes than InfluxDB, although both show similar memory consumption (Fig. 3). In addition to this, MongoDB needs at least 3 replicas to have availability (CAP model).



```
NAME       CPU %   MEM USAGE / LIMIT    MEM %   BLOCK I/O          PIDS
influxdb   0.09%   72.48MiB / 2.87GiB   2.47%   63.8MB / 1.25MB    10
mongodb    0.61%   73.2MiB / 2.87GiB    2.49%   57.5MB / 1.2MB     37
```

Fig. 3. . Screenshot of Docker stats showing the consumption of InfluxDB and MongoDB in Docker containers.

As the computing capacity of the Edge Node is very limited, **InfluxDB** is recommended to be used as database. It shows good performance when working with sensors and for data analytics (R and Python language support) [15], it has a SQL-like query language, it is compatible with JSON and it is specifically designed for metrics and events, which are main type of data that the Edge Node will process. InfluxDB is also the database defined in the eWLCA and the one used by Minsait for its PoC with i-DE.

4) *Hardware*

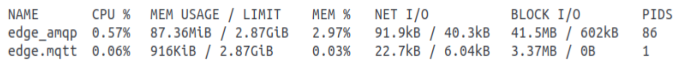
Some Edge platforms require their own specific device whereas other are more flexible and can use a third-party device. SSs can be considered as high-risk installations due to the voltage, the devices they contain and the temperatures that can be achieved, so, regardless of the platform finally used, the device must pass the requirements and tests to be installed in a SS. Specifically, up to 29 tests have to be passed, classified in 6 groups: *insulation*, *radioelectric disturbances*, *immunity*, *electrical*, *mechanical*, and *climatic*. These tests must be certificated by a laboratory. Besides, the dimensions of the device should not exceed 220x140x130 mm (width x height x depth) in order to be included in the electric cabinets currently deployed by i-DE.

C. *Communications*

The two IoT protocols considered for this analysis are MQTT(v3.1, v5.0 is too recent) [16] and AMQP (v1.0) [17] since they are the most popular in IoT. They are based on publish/subscribe and both of them are open protocols, although they differ in functioning and other characteristics. As mentioned in Introduction, three communications environments with different main requirements are distinguished in the solution: inner communications, communications between the edge node and the management system, and communications between the edge node and other devices.

1) *Inner communications*

In order to provide input data to the microservices and to allow the possible communications between them, an inner communications bus is needed. As this inner bus does not have to be accessible from the outside world, the security of the protocol is not very relevant. The main requirements are **lightness** and **reliability**. The more computing capacity is left to microservices in the Edge Node, the better. In order to compare MQTT and AMQP in these aspects, a test was carried out using "dockerized" MQTT and AMQP brokers by publishing four dummy JSON messages simultaneously on four different topics/queues every five seconds. All the messages were correctly received by the subscribers. Performance results are shown in Fig. 4.



```
NAME        CPU %   MEM USAGE / LIMIT    MEM %   NET I/O             BLOCK I/O         PIDS
edge_amqp   0.57%   87.36MiB / 2.87GiB   2.97%   91.9kB / 40.3kB     41.5MB / 602kB    86
edge.mqtt   0.06%   916KiB / 2.87GiB     0.03%   22.7kB / 6.04kB     3.37MB / 0B       1
```

Fig. 4. Screenshot of the MQTT-AMQP comparison test results.

Observing Fig. 4, MQTT is remarkably lighter than AMQP in CPU and memory consumption. Besides, MQTT provides three levels of quality of service (i.e. reliability) versus the two levels provided by AMQP. Therefore, the use of **MQTT** is recommended for the inner communications. MQTT is also the one used by Minsait for its PoC with i-DE.

2) Communications with the Management System

The broker for these communications would be in the central system for simplicity and security. In this case, the main aspect to consider is the **security** of the protocol. Both MQTT and AMQP provide TLS/SSL security but AMQP additionally has compatibility with SASL (authentication). The increase in the messages load should not be a problem since the aim of the solution is to reduce the amount of data that has to be sent to the central system for processing, so the current communications infrastructure should be enough. Furthermore, AMPQ provides more control over queues and has a sort of request/response feature (Remote Procedure Call) to execute functions on demand. In terms of message compatibility, neither MQTT nor AMQP put restrictions on the format of the message. Therefore, the use of **AMQP** is recommended for these communications. The PoC of Minsait with i-DE uses MQTT+TLS.

3) Communications with other devices

At the initial implementation of this solution, it will be necessary the development of **protocol adapters** to communicate with the different devices currently present in a SS (e.g. data concentrator, transformer, sensors, etc.) (Fig. 5). For new devices (e.g. HEMS, DG inverters, etc.), the communication protocol chosen to be used will definitely depend on the environment of these new devices (e.g. coverage, internet availability…). **MQTT** is the most used by researchers for this type of devices and is so light that an additional MQTT broker for these communications could be deployed on the Edge Node without problems. Furthermore, as it is an open protocol, manufacturers can easily produce compatible devices.

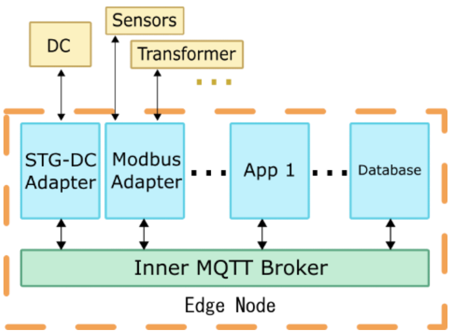
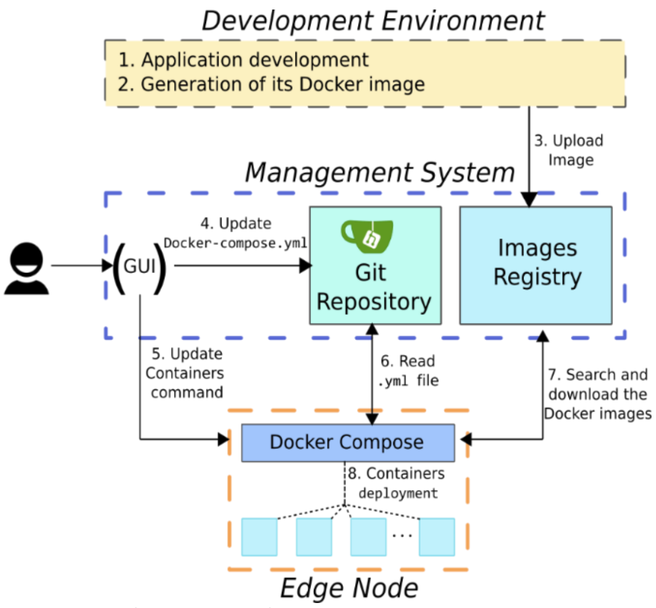


Fig. 5. Diagram of the integration of the Edge Node with already in-use communication protocols.

D. Management system

There are tens of Edge platforms in the market that have different purposes and characteristics. According to MachNation [18], the main aspects to consider are: communications **protocols compatibility** (e.g. protocol adapters), level of **autonomy** achievable**,** the **hosting** (on cloud or on premises), **hardware dependence** and **visualization capabilities.** Additionally, the electric utility should consider the experience of the vendor, the friendliness of the GUI of the platform, the programming languages that can be used for the applications, the integration of the platform with the central system (edge computing is not a substitute of the computing carried out at the central system, but a tool) and the security measures that the platform adopts.

In terms of workload orchestration, Kubernetes and Docker Compose are the common tools used with Docker. As the PoC of Minsait in i-DE, initially, only considers edge computing (and not fog computing), the tool to be used would be Docker Compose. Fig. 6 shows the development and deployment process that would be used and that is tested in test #2. Steps 6, 7 and 8 are automatically done by the Management System.



Development Environment
1. Application development
2. Generation of its Docker image
3. Upload Image
Management System
GUI
4. Update Docker-compose.yml
Git Repository
Images Registry
5. Update Containers command
6. Read .yml file
7. Search and download the Docker images
Docker Compose
8. Containers deployment
Edge Node

Fig. 6. Simplified diagram of the application deployment process. Icon source: https://icons8.com

III. Economic Impact

The analyzed solution would be categorized as a Type 2 investment by the Spanish regulatory body (investment in digitalization/automation, retributed according to the audited value). The device is assumed to cost ~450€ per unit and to have a regulatory life of 12 years (Smart Grids equipment). The immediate savings would be related with the reduction in **Time To Market** (TTM) of the functionalities (Table I) in comparison to the traditional approach for new functionalities deployment.

TABLE I. Summary of the differences between the two approaches for the development of a new functionaility

	Traditional Approach (Hardware+software)	New Approach (Edge Node) (Only software)
Average development time	3 years	4 months
No. Providers needed	3	1
Competitiveness for developments	Low	Very high
Cost of material	✓	✗
Cost of service	✓	✗
Cost of application	✓	✓

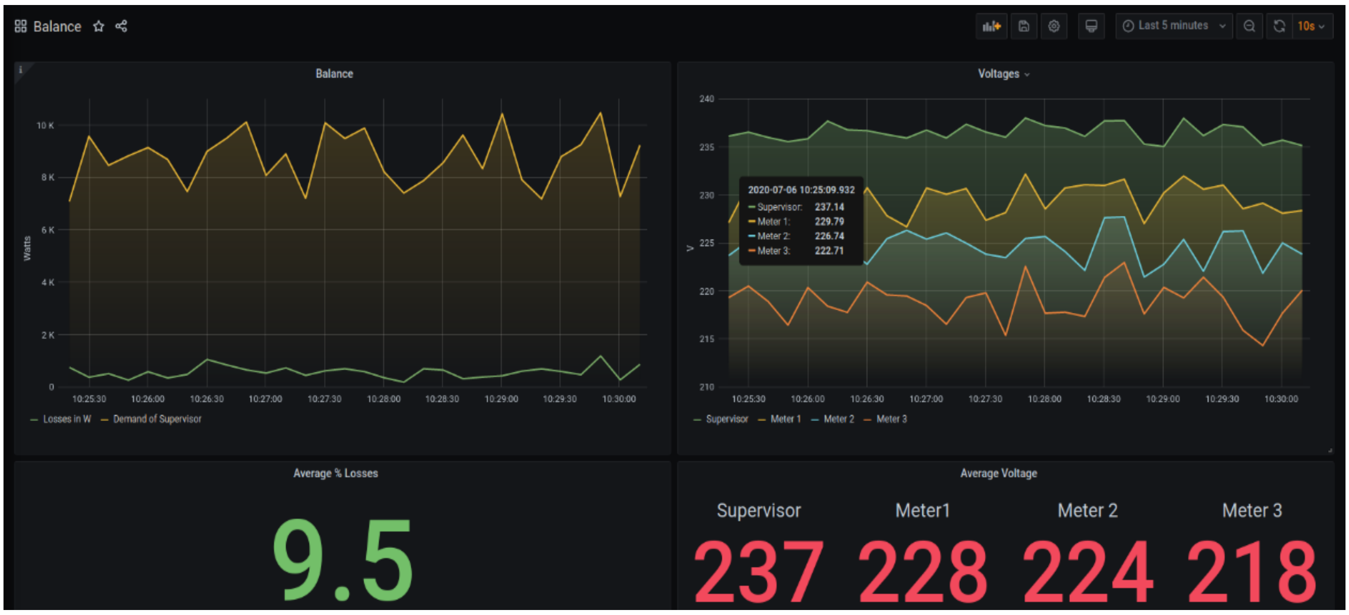


Fig. 7. Grafana dashboard to visualize, in real time, the data from the devices and the results of the balance application (9.5% of average losses and the absolute losses)

The equivalent cost of just deploying one new functionality would be reduced in an 87.5% with respect to the current approach. The future substitution of current devices by their "containerized" version will reduce the corresponding maintenance costs. Furthermore, **predictive maintenance** functionalities in the node could reduce maintenance cost in 12% and increase the lifetime of assets by 20% [19], which would be additionally retributed in ≥30% of the retribution, per asset, for operation and maintenance.

Considering that, nowadays, a connectivity algorithm developed by Ariadna Grid is executed at the central system, if it was deployed in the Edge Node the estimated savings in **central computing and storage** would account for ~15,000€ per year, and the corresponding reduction in **communications** has been valued by i-DE to be up to ~280,000€ per year.

Nowadays, the estimated annual value of **technical and non-technical losses** for i-DE are 364.62 M€. Some of the functionalities that could be deployed in the Edge Node would contribute to the reduction of losses (e.g. fraud detection, DR schemes, connectivity algorithm, etc.). TABLE II shows a summary of the calculated savings in this aspect.

TABLE II. SUMMARY OF ENERGY LOSSES SAVINGS

Energy losses	
Savings per device substituted/avoided and year (considering 25% of i-DE's SSs)	100,000 €
Fraud detection (qualitative)	Faster detection
	Less "false positives" => less inspections
	New frauds detected
Savings per year for phase balancing (25% scenario)	≥ 5,500,000 €

IV. Solution Testing

A. *Test #1: Real-time balance in local test environment*

A random data generator is developed in Node-RED to provide the real-time characteristic. The balance application, written in python, connects to the topic "measurements" of the inner MQTT broker to receive the data, in JSON, every 10 seconds (active power consumption registered by a supervisor and three meters), then it calculates the active power balance (knowing only the supervisor's ID and independently of the number of devices) and publishes the result on the "feeder/balance" MQTT topic. Node-RED is then used to check the correct functioning of the application and to store the results in the InfluxDB database (with tag equals to "balance").

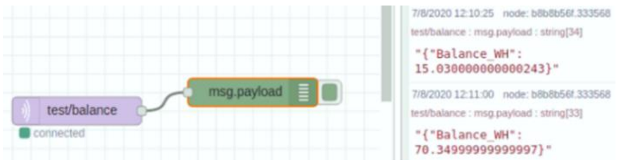
This test shows the possibilities that the solution could provide. The application is quite light considering that it contains a light but complete python environment (0.02% of CPU and 12.42 MiB of memory usage in this local test); its code can be easily adapted to a real environment, it works in real-time with MQTT without the need of querying the database or any other intermediary and it works in an isolated way (i.e. an error in the application would not affect the other containers). Furthermore, the input data and the result of the balance can be visualized in real time by using a Grafana (Fig. 7) or Chronograph, without the need of storing the data in the central system.

B. *Test #2: Real-time balance in remote test environment using S02-like reports*

This test checked the deployment process shown in Fig. 6 with a modified version of the balance application from test #1 to read JSON messages that follow an S02-like structure (S02 are "Daily incremental" reports defined by the STG-DC specification). The remote test environment is provided by Minsait for i-DE, so all the deployment process is done similarly to how it would be done for a real functionality deployment in a SS.

The balance application is successfully deployed in this remote environment. Its correct functioning is shown in Fig. 8. In this environment the ports of the database were not accessible, so the insertion of data results from the "test/balance" topic into the database could not be done in a simple way through node-RED. It would be done by changing the configuration of the database in the deployment file (Docker-compose.yml) or by using Telegraf, whose configuration file is also complex. Due to the nature of this test environment (own by Minsait), this configuration was not modified.

At this initial phase of the PoC, steps 2, 3 and 4 of the process (Fig. 6) can become a source of errors during deployment, mainly because of the complexity of the configuration files involved (e.g. docker-compose.yml and Dockerfile.yml) and because of the use of the command line tool, which is not very user-friendly.



```
7/8/2020 12:10:25   node: b8b8b56f.333568
test/balance : msg.payload : string[34]
"{"Balance_WH":
15.030000000000243}"
7/8/2020 12:11:00   node: b8b8b56f.333568
test/balance : msg.payload : string[33]
"{"Balance_WH":
70.34999999999997}"
```

Fig. 8. Node-RED Flow screenshot to check that the balance application deployed in the remote test environment works correctly

V. Conclusions and future works

This project has analyzed the technologies (and their alternatives) involved in an edge-computing-based solution that follows the eWLCA [7]; the advantages, disadvantages, and functionalities that this solution could have in the LV distribution and its economic impact on the system and on the utility (i-DE), fulfilling the objectives initially set. Furthermore, the solution has been partially tested using two different test environments (local and remote) by developing an active energy balance application that works in real time.

The conclusion that can be extracted from this project is that the application of edge computing at the secondary substation level has the potential to be the new paradigm of how the LV grid is monitored and controlled, providing great benefits to the utility, to the system, and to the electric industry in general. The technologies used in the analyzed architecture (eWLCA) are found to be the appropriate. Its modularity and the fact of being open-source-based will provide great flexibility to the utility and promote developments by academics and industry, although the deployment process tested should improve on user friendliness and on the capacity of making online configuration changes and new application deployments without interrupting operation. However, these aspects, together with the hardware challenge, are close to be solved in future phases of the PoC developed at i-DE.

For future works, the future use of an IoT protocol (e.g. MQTT, AMQP, CoAP...) for smart metering through PLC PRIME could be studied. The development of protocol adapter containers, as discussed, will also be necessary at the beginning of the implementation. SSs could be classified according to their characteristics, requirements, and necessary functionalities to standardize microservices deployments according to this classification. Finally, by taking advantage of this solution and the valuable information it will provide, the use of digital twins of SSs and the assets installed in them would also be an interesting future work.

References

[1] M. Alberto *et al.*, ‘OpenNode: A smart secondary substation node and its integration in a distribution grid of the future’, in *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sep. 2012, pp. 1277–1284.

[2] S. Cejka, A. Hanzlik, and A. Plank, ‘A framework for communication and provisioning in an intelligent secondary substation’, in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2016, pp. 1–5, doi: 10.1109/ETFA.2016.7733591.

[3] ABB, ‘Technical Guide: Smart Grids 2. The “smart” secondary substation’. Apr. 2016, Accessed: May 27, 2020. [Online]. Available: https://new.abb.com/docs/librariesprovider27/default-document-library/gu_smart-grid_cabina-secondaria(en)-_1vcp000620---digiprint.pdf?sfvrsn=2.

[4] C. Jinming, J. Wei, J. Hao, G. Yajuan, N. Guoji, and C. Wu, ‘Application Prospect of Edge Computing in Smart Distribution’, in *2018 China International Conference on Electricity Distribution (CICED)*, Tianjin, China, Sep. 2018, pp. 1370–1375, doi: 10.1109/CICED.2018.8592104.

[5] S. Chen *et al.*, ‘Internet of Things Based Smart Grids Supported by Intelligent Edge Computing’, *IEEE Access*, vol. 7, pp. 74089–74102, 2019, doi: 10.1109/ACCESS.2019.2920488.

[6] P. Wang, S. Liu, F. Ye, and X. Chen, ‘A Fog-based Architecture and Programming Model for IoT Applications in the Smart Grid’, *ArXiv180401239 Cs*, Apr. 2018, Accessed: Jun. 23, 2020. [Online]. Available: http://arxiv.org/abs/1804.01239.

[7] M. Ortega de Mues, D. Seseña, C. Martínez Spessot, M. Carranza, and J. Lang, ‘Creating an effective and scalable IoT infrastructure by introducing Edge Workload Consolidation (eWLC)’, *Intel*, Feb.03, 2020 https://www.intel.com/content/www/us/en/develop/articles/edge-workload-consolidation-ewlc.html (accessed Jun. 24, 2020).

[8] M. Caprolu, R. Di Pietro, F. Lombardi, and S. Raponi, ‘Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues’, in *2019 IEEE International Conference on Edge Computing (EDGE)*, Milan, Italy, Jul. 2019, pp. 116–123, doi: 10.1109/EDGE.2019.00035.

[9] A. Martin, S. Raponi, T. Combe, and R. Di Pietro, ‘Docker ecosystem – Vulnerability Analysis’, *Comput. Commun.*, vol. 122, pp. 30–43, Jun. 2018, doi: 10.1016/j.comcom.2018.03.011.

[10] M. J. D. Lucia, ‘A Survey on Security Isolation of Virtualization, Containers, and Unikernels’, *US Army Res. Lab.*, p. 18, May 2017.

[11] T. Goethals, M. Sebrechts, A. Atrey, B. Volckaert, and F. De Turck, ‘Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications’, in *2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*, Nov. 2018, pp. 1–8, doi: 10.1109/SC2.2018.00008.

[12] I. Eyberg, ‘Introduction To Unikernels’, *Nordic APIs*, Feb. 12, 2019. https://nordicapis.com/introduction-to-unikernels/ (accessed Jun. 03, 2020).

[13] M. Kovatsch, R. Matsukura, M. Lagally, T. Kawaguchi, K. Toumura, and K. Kajimoto, ‘Web of Things (WoT) Architecture’, Apr. 09, 2020. https://www.w3.org/TR/wot-architecture/ (accessed Aug. 03, 2020).

[14] S. Cejka, R. Mosshammer, and A. Einfalt, ‘Java embedded storage for time series and meta data in Smart Grids’, in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov. 2015, pp. 434–439, doi: 10.1109/SmartGridComm.2015.7436339.

[15] M. Nasar and M. Abu Kausar, ‘Suitability Of Influxdb Database For Iot Applications’, *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 10, pp. 1850–1857, Aug. 2019, doi: 10.35940/ijitee.J9225.0881019.

[16] OASIS Standard, ‘OASIS MQTT Version 5.0.’ OASIS Open, Mar. 07, 2019.

[17] OASIS Standard, ‘OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0’. Oasis Open, Oct. 29, 2012, [Online]. Available: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf.

[18] D. Tokar, ‘Whitepaper: Five requirements of a leading IoT edge platform’, *MachNation*, Sep. 18, 2017. https://www.machnation.com/2017/09/18/whitepaper-five-requirements-leading-iot-edge-platform/ (accessed Jul. 10, 2020).

[19] M. Haarman *et al.*, ‘Predictive Maintenance - Beyond the hype: PdM 4.0 delivers results’, PWC and Mainnovation, 2018. [Online]. Available: https://www.pwc.be/en/documents/20180926-pdm40-beyond-the-hype-report.pdf.