A Smart Meter Infrastructure for Smart Grid IoT Applications

A Smart Meter Infrastructure for Smart Grid IoT Applications

Matteo Orlando, Abouzar Estebsari, Enrico Pons, Marco Pau, Stefano Quer, Massimo Poncino, Lorenzo Bottaccioli and Edoardo Patti

***Abstract*—Electric infrastructures have been pushed forward to handle tasks they were not originally designed to perform. To improve reliability and efficiency, state-of-the-art power grids include improved security, reduced peak loads, increased integration of renewable sources, and lower operational costs. In this framework, "smart grids" are built around bidirectional communication technologies, where "smart meters" communicate with all other entities and collect data from the power grid, offering specific features to each actor playing in the energy marketplace. In this paper, to overcome some of the challenges raised by smart grids and smart meters, we propose a distributed metering infrastructure which provides bidirectional communication, self-configuration, and auto-update capabilities. Our 3-phase smart meters follow the basics Internet-of-Things principles and have the ability to run, either on-board or distributed on the network, multiple algorithms for smart grid management. These algorithms can be freely added, updated, or removed on-the-fly thanks to the auto-update feature of the system. Moreover, to reduce costs and improve scalability, we prove that it is possible to implement our smart meters using only off-the-shelf and inexpensive hardware devices. A digital real-time simulator (i.e., Opal-RT) has been used to assess the capabilities of both the infrastructure and the meter. Our experimental analysis shows that the latency introduced by the data transmission over the Internet is compliant with the limits imposed by the IEC 61850 standard. As a consequence, our architecture does not affect the operational status of the smart grid, making it a viable solution to support the deployment of novel services.**

***Index Terms*—Smart Grid; Smart Meter; Distribution Systems; Advanced Metering Infrastructure; Multi-Model Co-Simulation; Co-simulation Platform; Outage Management**

I. Introduction

POWER Networks have evolved in the last decade to reach a new level of efficiency and reliability, and to include new and pervasive Renewable Energy Sources (RES). To achieve these targets, smart grids are now implemented at all levels, from the power plants down to end-user networks. A modern grid does not only include wires, substations, transformers, and other electrical devices, but it also exploits the Internet to better manage the energy. In other words, a modern smart grid is an electrical grid that adopts strategies coming from the Information and Communication Technology (ICT) area to autonomously gather information and improve its efficiency, reliability, economics, and sustainability. Smart grids also enable new actors (such as aggregators, virtual power plants, and energy service companies) to participate in a fast-evolving and distributed marketplace, providing new services for smart energy management.

In this rapidly evolving scenario, ICT and the Internet-of-Things (IoT) play a crucial role. Indeed, IoT aims at connecting over the network everyday devices, to enable them to exchange information [1]. This paradigm allows developers to design new algorithms and autonomous systems that can collect data, process them, and take autonomous decisions [2] to improve the management of the grid.

Among the different devices that are possible to connect to a power grid, next-generation meters (also known as *smart meters*) implement IoT functionalities to collect data from the power grid and to communicate with other hardware and remote software entities over the Internet. The resulting Advanced Metering Infrastructure (AMI) [3] offers different features to each actor playing in the energy marketplace [4]. For example, Distribution System Operators (DSOs) can use smart meters to increase the efficiency of the distribution network and to improve its capability and reliability. Retailers can use smart meters to forecast the variations of the network load to create adequate policies to reply to demand. Prosumers can use smart meters to control their energy production (highly variable due to the intermittent behavior of renewable sources) and utilization and they can reduce the cost of their electricity bills [5].

Unfortunately, modern metering infrastructures have to face multiple challenges to be cost-effective, scalable, and reach the desired penetration on the market. Among these hurdles, we would like to recall the cost of the communication infrastructure [6], the price of the hardware devices [7], and the extremely high intricacy of modern networks. To propose cost-effective and scalable solutions, we thus need to deploy low-cost devices and to be able to configure, control, and update all units of the network remotely. For example, a self-healing grid should be able to automatically reconfigure itself in case of an outage, automatically performing fault location and service restoration [8]. Unfortunately, this process is still partially executed manually nowadays. This approach reduces the reliability of the network, increases the outage time, augments the network dependability from human intervention, and often results in customer dissatisfaction and extra costs for the system operators. To design a true self-healing grid, all previous process steps should be automated, minimizing human intervention. Distributed RES also need

M. Orlando, E. Pons, S. Quer, M. Poncino, L. Bottaccioli and E. Patti are with Politecnico di Torino, Turin, IT. E-mails: {name.surname}@polito.it.
A. Estebsari is with the London South Bank University, London, UK. E-mail: estebsaa@lsbu.ac.uk.
M. Pau is with RWTH Aachen University, Aachen, Germany. Email: mpau@eonerc.rwth-aachen.de

particular attention. RES breaks the traditional schema of the unidirectional energy flow as they can be installed by end-users (the so-called prosumers), which can both consume and produce energy, and are also quite unpredictable. For these reasons, State Estimation (SE) algorithms are required to help the operators to monitor the state of the grid. SE strategies evaluate the electrical parameters of each node and build a comprehensive picture of the grid to prevent possible contingencies,

In this paper, to overcome some of the above challenges, we propose two main contributions:

- We describe a distributed software infrastructure, including an advanced metering technology, providing several benefits. This infrastructure admits bidirectional communication for the smart units and it enables the devices deployed on the grid to self-configure and auto-update themselves. Moreover, it allows the information related to the deployed devices to be used by the remote services included in the system. In this way, we enable interoperability with third-party software, which could facilitate further general-purpose services and business cases.
- A design pattern for an IoT-enabled 3-phase smart meter. Our design defines a 3-phase meter with the capability to run multiple algorithms, either on-board, on-fog, or on-cloud. Thanks to the self-configuration and auto-update procedures, any algorithms can be added, updated, or removed on the fly without affecting the rest of the system. This makes the whole solution flexible and opens it to future improvements of the smart grid technology. In our view, this feature is essential to unlock new services and to foster new business opportunities for the actors playing in the energy marketplace.

During our design process, we pay particular attention to the software engineering phase and to the meter (hereinafter, referred to as 3SMA) design stage. We present an evaluation of our distributed infrastructure, and more specifically of our 3SMA, based on real-time simulations. Our simulations proved that our architecture imposes very low requirements for the device needed to build the 3SMA, thus reducing the cost and making the entire structure highly scalable.

A. Contributions

To summarize, our work includes the following novelties. We design and develop a scalable distributed software infrastructure for advanced metering presenting the following features:

- It provides bidirectional communication between smart devices and novel services.
- It enables cloud, fog, and edge computing to support new smart services to manage the grid.
- It unlocks new decentralized services and improves the quality and the performances of those already existing.
- It allows the self-configuration and the auto-update of the smart devices installed across the smart grid.
- It provides remote services with (near-) real-time information about both the smart devices and the smart grid status.
- It integrates third-party software and enables their interoperability with all the entities, either hardware or software, in our solution.

We also propose a novel design pattern to develop our 3SMA, i.e., a new 3-phase meter, which:

- Embeds IoT functionalities.
- Is compliant with the requirements of the proposed distributed software infrastructure.
- Provides bidirectional communication with the other actors in the infrastructure, either hardware or software.
- Allows remote grid data samplings and transmissions in (near-) real-time.
- Runs custom algorithms for advanced smart grid management, either on-board or distributed, thus, exploiting the offered cloud, fog and edge computing capabilities.

B. Roadmap

The paper is organized as follows. Section II focuses on related works and their main differences from the approach proposed in this paper. Section III first describes the proposed distributed software infrastructure for advanced metering, introducing both the identified actors, and the technologies adopted to enable the communication and the data exchange among them. Then, it presents the proposed IoT-based 3-phase smart meter (i.e., our 3SMA) with its main characteristics and interactions with the other entities in the distributed software infrastructure. Finally, Section III reports the on-cloud, on-fog, and on-edge applications used to test and assess both the 3SMA and the infrastructure. Section IV concentrates on the communication flows among the identified actors, either hardware or software, in our solution. Section V describes the case study and the experimental setup we used to evaluate our infrastructure. Section VI includes our experimental analysis of the metering infrastructure. To conclude, Section VII reports our final remarks and some hints on possible future works.

II. Related work

As introduced in Section I, advanced metering infrastructures for smart grids enable bidirectional communications to monitor the energy transmission and distribution process [9]. The last few years have seen growing attention on the area, focusing on different approaches and distinct technologies to reach advanced management architectures. Different standards and protocols are diffused and have been proposed such as DLMS/COSEM [10], SML [11] and IEC 61850 [12][13][14], these solutions aim to address different challenges of the Smart Grid scenario and are designed for different actors according to their needs. Thus interoperability plays a crucial role in the design of AMI and smart meters that need to offer multiple ways to communicate with the grid.

Meloni et al. [15] introduce a new infrastructure to guarantee device interoperability. They also introduce the

possibility to develop additional cloud services by providing proper API to obtain information from the meters. However, the suggested infrastructure manages only the distribution network and not the transmission one. Moreover, the authors design their meters as sensing devices able to collect and send data, but unable to perform any sort of data processing.

Chen et al. [16] illustrate a smart-metering architecture for IoT-based meters. They design it using edge computing to overcome the problems related to the latency and the bandwidth, typical of cloud computing. However, their architecture does not provide cloud or fog computing, which, in our opinion, are crucial to unlocking advanced and third-party services, for smart grid management.

Yan et al. [17] implement their infrastructure adopting fog computing. They perform data processing on clusters composed of a single board computers and located between the edge and the center of the metering infrastructure. In this way, they can reduce the amount of data exchanged. However, they do not support edge-computing features nor provide self-configuration and auto-update functionalities.

Ou et al. [18] describe a metering system for transmission networks using wireless technologies. They receive measures from the transmission line and communicate them to a central monitoring service. Unfortunately, their infrastructure does not support third-party applications and the meters are used only to transfer data without performing any data processing.

Lloret et al. [19] introduce a centralized architecture supporting different smart devices by using distinct communication protocols. Their infrastructure includes cloud services using big-data techniques to provide different types of services to users and DSO. However, their approach is not scalable enough for large smart grids. With thousands of devices, a centralized infrastructure implies the necessity to transfer and collect a huge amount of information. This process can cause latency, reduce the bandwidth, and increase network costs.

The previous analysis shows that the most promising approaches are the ones adopting fog and edge computing. Fog computing takes advantage of data collectors distributed across the grid to perform data pre-processing. Edge computing performs data processing directly on the smart meters. Unlike the previous works, i.e., [15], [18], [19], where the computation is performed in centralized form, with fog and edge computing the computational load is distributed among different devices. This strategy reduces, especially for critical applications, the latency and the bandwidth required to transfer information [20]. Forcan et al. [21] locate fog servers close to the metering devices. Servers perform data processing whereas the metering devices are just used for sensing. Liu et al. [22] use edge computing to support multiple network management algorithms, reduce the computation time, and decrease the bandwidth required for data transmission. Their infrastructure, as in all other works mentioned above, makes use of IoT to gather data from the Smart Grid. As [15], [19], and [22], they support the execution of different algorithms to improve and simplify smart grid management, and to increase its reliability.

Another key element of state-of-the-art metering infrastructures is the meter architecture. To have distributed, or even fully delocalized algorithms, and to reduce the necessity to communicate a huge amount of data [23], smart meters should be able to communicate through the Internet and to perform data processing. Unfortunately, the majority of the works designing smart meters mainly concentrate on the performance [24], [25], [26] of the physical device. On the contrary, there are relatively few works focusing on the development of smart meters using IoT technologies to enable new features for grid management.

Gallano et al. [27] design an IoT meter supporting a mobile application able to display the energy consumption. However, the meter is not included in a metering infrastructure, and it offer the possibility to run neither on-board nor distributed algorithms.

Using a single board computer, Sirojan et al. [28] develop a 3-phase meter able to collect the values of the current on each phase. The meter is also capable of performing simple data analysis, but it does not offer the possibility to run custom algorithms.

Chen et al. [29] test an IoT-enabled single-phase smart meter for demand-side management in smart homes only. Even if the meter is used as an edge computing device, only a few predefined algorithms can be executed on it without any possibility of customization.

Pegoraro et al. [30] illustrate a 3-phase smart meter prototype capable of running distributed state estimation. Their meter adopts IoT-based cloud systems for data exchange. Pignati et al. [31] suggest a similar solution in which the meter exploits the ICT system of a university campus. In both these works, the meters are used to collect information and to transfer it to concentrators located toward the edge of the communication network. These concentrators run the first step of a distributed algorithm performing data processing.

As mentioned before, both interoperability and flexibility need to be taken into account during the development of a smart device or an AMI. Protocols such DLMS [10] only offer the possibility to securely exchange metering data across the network and to update the meter. However, since it is designed for reading consumer data it could be unfeasible or hard to use it by the multiple concurrent applications for smart grid management.

One of the advantages of our approach, with respect to the ones described so far, is the possibility to use the meter to execute stand-alone or distributed applications. In addition, it is possible to completely customize the smart-meter, both in terms of hardware and software. Thus, the user is free to select the desired target accuracy and the applications to deploy on the meter. Moreover, our infrastructure supports self-configuration and auto-update strategies. In this way, our meters can automatically discover newly installed devices and we can change settings and software applications on the fly. As a consequence, we can deploy on the grid at run-time smart devices which differ from the designed meter and offer innovative and previously unforeseen services.

To summarize, Table I shows the main similarities and differences between our approach and the works previously described.

Infrastructure	Meter	Onboard intelligence		IoT	Computing			Self	Auto update	Third party
	prototype	Data processing	Custom algorithm		on Cloud	on Fog	on Edge	configuration		services
Proposed	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Gallano et al. [27]	✓	-	-	✓	-	-	-	-	-	-
Sirojan et al. [28]	✓	✓	-	✓	-	-	-	-	-	-
Chen et al. [29]	✓	✓	-	✓	-	-	✓	-	-	-
Pegoraro et al[30]	✓	-	-	✓	-	-	-	-	-	-
Pignati et al. [31]	✓	✓	-	✓	✓	-	-	-	-	-
A. Meloni et al. [15]	-	-	-	✓	-	-	-	-	-	✓
Q. Ou et al. [18]	-	-	-	✓	-	-	-	-	-	-
Chen et al. [16]	-	-	-	✓	-	✓	-	-	-	-
Yan et al. [17]	-	-	-	✓	-	✓	-	-	-	-
J. Lloretet al. [19]	-	-	-	✓	✓	-	-	-	-	✓
M. Forcan et al. [21]	-	-	-	✓	✓	✓	-	-	-	-
Y. Liu et al. [22]	-	-	-	✓	✓	-	✓	-	-	✓

TABLE I: A synoptic table comparing our approach with several state-of-the-art similar works. The table considers both the infrastructure architecture and the 3SMA design.

III. Metering infrastructure and smart meter

This section describes our metering framework. It first details our distributed software infrastructure, defining the actors involved and the technologies adopted to enable the communication among them. After that, it presents the design of our internet-connected 3-phase smart meter (the 3SMA) highlighting its main characteristics and its interactions with the other units. Finally, it reports the decentralized applications used to test both the 3SMA and the infrastructure. We focus on auto-reconfiguration for self-healing and state estimation capabilities. Notice that the system supports both on-board (deployed and executed directly on the 3SMA) and remote (deployed and executed on-fog or on-cloud) applications. Moreover, note that the algorithms presented in this work are just a mere example of the ones that could be used on our infrastructure. Indeed, thanks to the auto-update feature, both the infrastructure and the 3SMA meter have been designed to be flexible in terms of updating, adding, and replacing the application at run-time, without the necessity to reboot the whole system. Thus, our infrastructure will also trigger the possibility to use other innovative services in the future.

A. Distributed Metering Infrastructure

One of our targets is to design a highly scalable, distributed, and decentralized metering infrastructure to manage and monitor a high number of devices and services over the grid. As shown in Figure 1, the principal entities of this infrastructure are the *Message brokers*, the *Device Catalog*, the *3-phase Smart meter* (3SMA), and the *Remote service* applications. These units talk to each other by exploiting two communication paradigms, namely the request/response through the REST Web Services [32] and the publish/subscribe [33] through the MQTT protocol [34].

The Device Catalog (DeC), at the center of Figure 1, is an essential software component of our infrastructure as it acts as both service and device register. It is in charge of registering all active devices and software entities managing the smart grid. It also keeps track of the active units connected to the Internet, thus enabling service and device discovery. When, for any reason, a hardware device or a software entity cannot communicate with other units, the DeC marks it as unavailable and propagates this information to all other units connected to the infrastructure. This procedure makes the unavailability of a device known on the entire network. The DeC also stores the metadata for each 3SMA. Metadata includes the algorithms run by the meter and the topology of the portion of the grid that it has to monitor. The DeC can eventually provide this information to the remote service applications, such that these services are able to perform the right set-up to retrieve the required data. The DeC is also responsible for the self-configuration and auto-update of the 3SMA units. In other words, the DeC provides to each meter the endpoints of the message brokers, the metadata, and all the necessary information to keep the meter updated. Moreover, it provides to each remote service application a complete list of both 3SMA and other services' endpoints, and all the necessary information to manage the smart grid. As mentioned before, two communication protocols, REST and MQTT, are used in this framework. REST is a synchronous architectural style used to build web services exploiting the HTTP protocol [32]. It is widely supported and used for 1-to-1 communications. On the contrary, MQTT is an asynchronous protocol implementing the publish/subscribe paradigm [33]. It is quite common in the IoT world and it is optimal for 1-to-many and event-driven communications [34]. Due to their differences and their specific characteristics, these two protocols have been selected to perform different tasks. REST is mainly used for the self-configuration and auto-update of the IoT devices deployed over the grid. MQTT is adopted to exchange data with the meters and on remote service applications. More details on these protocols are reported in Section IV.

The two message brokers of Figure 1 manage the publish/subscribe communication among hardware and/or software components exploiting the MQTT protocol. We decided to use two different message brokers to separate the two main communication contexts. The first broker manages data exchange over the grid (i.e., topology changes, reconfigurations, etc.). The second broker monitors information. We adopt this configuration to minimize the delay of the MQTT messages related to device management. Indeed, depending on the amount of information we need to exchange and on the implementation of the message broker, a single broker might constitute a bottleneck for

our infrastructure, thus increasing the overall communication latency. As communication latency is a crucial factor in smart-grid management, we appropriately considered it and we minimized it by resorting to two message brokers in our configuration.

B. Our 3-phase Smart Meter Architecture

We designed our 3SMA to operate in our distributed infrastructure, described in Section III-A, and to fully exploit its features. Essentially, we optimized our 3SMA to perform three different tasks: self-configure, collect electric measures from the power grid, and run user-defined network algorithms. These tasks are logically performed by different units, as graphically illustrated in Figure 2. The tasks performed by each unit are the following.

The first unit, i.e., the communication interface, is represented on the left-hand side of Figure 2 and it is in charge of enabling the communication over the Internet among 3SMA and the other actors in the platform, either hardware or software.To be compliant with the infrastructure, we exploit both the communication protocols introduced in Section III-A, namely REST and MQTT. We use REST, i.e., the request/response approach, to gather the initial configuration of the 3SMA from the DeC unit. We also use the REST protocol to notify the DeC device that it is reachable and it is properly connected to the network. Finally, we use REST to update the 3SMA in terms of the endpoints, the MQTT broker that must be used, and the algorithms that must be run. On the other hand, we adopt the MQTT protocol, i.e., the publish/subscribe approach, to receive updates from the DeC unit concerning the status of the grid and the settings of the metering infrastructure (such as modifications on the topology of the grid, changes of the endpoint, etc.). Moreover, we also use the MQTT protocol to transfer the data collected by the 3SMA unit and the results computed by the on-board algorithms to the remote service applications.

The second logic unit of our 3SMA device, i.e., the self-configuration and algorithm unit at the center of Figure 2, is in charge of executing three main different tasks: Self-configuration, auto-updating, and data-processing. We stress the terms "self" and "auto" because one of the main features of our meter is that it can potentially run any algorithm, at least as far as it has the computational power to execute it. When we need to execute a new algorithm on the meter, we start by updating the algorithm inside the DeC device. Once done

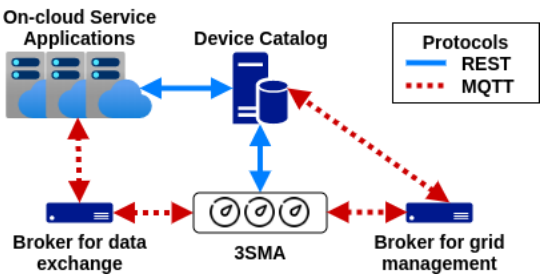

Fig. 1: Infrastructure schema.

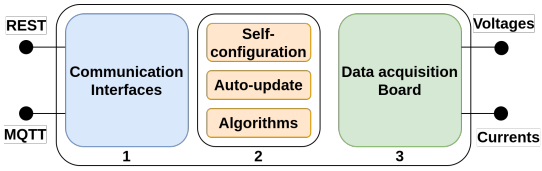

Fig. 2: Conceptual scheme of the 3SMA on-board software.

this, the 3SMA will receive all updates from the DeC, it will retrieve the new algorithm, and it will start executing it. This strategy can then be used to replace any existing algorithm on-the-fly, without affecting the rest of the system, and enabling our infrastructure to be ready for future extensions.

The third, and last, logic function performed by our 3SMA, i.e., the data acquisition part, is performed by the rightmost block of Figure 2. This unit collects all electrical quantity measures on the power grid and it makes them available for the algorithmic manipulation performed by the computational unit.

Obviously, the three logic units previously described can be realized, i.e., implemented, adopting different hardware devices. Our current hardware prototype combines a Raspberry Pi 3 Model B unit with a Data Acquisition board (DAQ) provided by Measurements Computing™ [35]. The Raspberry Pi unit is a small embedded computer and it is connected with the DAQ using a USB port. We selected these hardware devices because using open-source and relatively cheap units drastically reduces the costs of the entire infrastructure and improves its adoption in large networks improving our design scalability. As mentioned before, indeed scalability is one of the main challenges faced by modern infrastructures. With this hardware configuration, our DAQ is able to collect six channels, each one with a frequency that is limited to 6.4 kHz but it can be customized according to the necessity of our application. For example, for our initial tests, we limited the sampling rate to 3.2 kHz, i.e., to 50% of the maximum possible frequency. Anyway, this value was not only more than sufficient to perform our acquisition phase but it also allowed us to dedicate the remaining computational power to run our algorithms more efficiently.

Using this configuration, the measures are collected through the DAQ by the Raspberry Pi. The Raspberry Pi is also in charge of running the self-configure and auto-update routines, of executing the different algorithms, and of exchanging information with the remote service applications and the DeC. Thus, it is essentially the hardware unit making "smart" our 3SMA architecture.

C. Algorithm and remote service applications

As mentioned in Section III-B, our 3SMA supports on-board applications deployed and executed directly on the device. In general, these applications can be seen as a part of a more complex and complete software suite running in a distributed way over the network, thus offering on-edge, on-fog, and on-cloud features. To evaluate the computational power of our infrastructure, we tested it with two different and well-known applications: A Fault Location, Isolation and Service

Restoration (FLISR) algorithm and a State Estimation (SE) one.

Given a network fault, a FLISR procedure automatizes the localization and the isolation of the portion of the grid affected by the fault and, wherever possible, it reconfigures the power grid to minimize the number of disconnected customers [36]. This is a combination of fault location algorithms with Fault Detection, Isolation and, Restoration (FDIR) schemes to improve the performance of fault management. The aim is to restore most of the interrupted customers as quickly as possible.

To reach these targets, we deployed the algorithm proposed by Estebsari et al [36] on our 3SMA (i.e., on-edge). Their approach combines an impedance-based algorithm with a sparse voltage measurement-based strategy. Their procedure detects the bus pair affected by the fault and evaluates the distance between the fault and the first bus of the pair. The bus pair defines the secondary substations (i.e. bus) and the branch on which the fault occurred. When a fault occurs, the protection relays at the beginning of the feeders trip, and the circuit breakers disconnect the feeder from the supply. The required measurements to run the above-mentioned fault location algorithm are sampled by 3SMAs and sent to the one with the activated algorithm and installed at the beginning of the feeder. Once the measurements are received by this 3SMA, it runs the fault location algorithm. The results of this computation are then sent to the remote service application running on the cloud. This application reconfigures the network, providing a new grid topology to isolate the faulty portion of the network and to restore the power supply for the rest of the grid. To obtain the new network configuration, the procedure sends an actuation message to open or close every smart switch that needs to be switched. The new topology is also communicated to all entities (both hardware and software) that can be affected by the changes. Additional details on the communication phase of the FLISR process are given in Section IV-B.

The main target of a SE procedure is to provide an estimation, over the entire network, of some electrical values (such as voltages, powers, and currents), given the available local measures. This task is of paramount importance to promptly identify any critical problem in the grid, especially when distributed or renewable energy sources are connected to the system, as the energy production of those sources is largely unpredictable and volatile. Usually, system operators estimate the state of the network by adopting a centralized application collecting all the measures coming from the meters spread across the grid. Unfortunately, this process can be unfeasible on smart grids, or at least it can be very expensive, due to communication bottlenecks and unpredictable delays. Therefore, to distribute the computation on-edge and on-fog, we adopted the hierarchically and distributed SE algorithm introduced by Pau et al. in [37], [38]. This algorithm is based on the weighted least squares approach. Its target is to filter out the errors on the measures to provide the most probable operational state of the grid, exploiting the redundancy of the input measures (please, see [38] for more mathematical details on the algorithm).

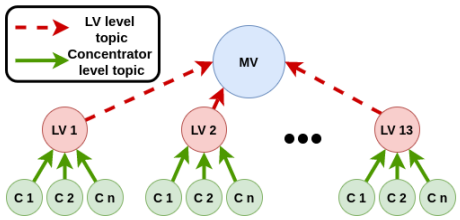

Fig. 3: Conceptual scheme of the distributed State Estimation algorithm.

The distributed implementation of the algorithm divides the estimation problem into multiple hierarchical SE levels, which are sequentially executed following a bottom-up process. In our implementation, the estimation process is divided into three different levels: The *Concentrator*, *Low Voltage*, and *Medium Voltage* level. This partition reflects the hierarchy existing on the grid and illustrated in Figure 3. In this representation, multiple concentrators C are connected to a single *LV substation*, and multiple LV substations are connected to a single *MV substation*. These connections can be easily represented with a tree structure which can also be exploited for the logical partition of the SE process. The first computation is made at the concentrator level. At this level, the state is estimated for all customer nodes subtended by the feeder bus to which the concentrator is associated. The LV level estimators use the results provided by the concentrators (voltage and total power at the feeder bus) to estimate the operating conditions of the low voltage grid feeders. Similarly, the MV level estimator uses the result of the MV and LV level (i.e., voltage and overall power) as input to evaluate the operating conditions of the subtended medium voltage grid. Differently from the conventional implementation of the SE algorithm, in our architecture both the LV level and the MV level state estimation algorithms are deployed on a 3SMA entity. Each 3SMA, in charge of the execution of the SE service, executes one of the two algorithms (namely, either the LV level or the MV level) according to the settings that it has received during the self-configuration and the auto-update procedure. These are two on-fog algorithms. More details on the communication flow and the coordination among the different actors running the distributed SE are provided in Section IV.

IV. Communications Flow

As analyzed in Section III, we adopt both the requests/response and the publish/subscribe communication paradigms [33]. We use REST during the initial communication phase. In this step, we define the settings of the DeC and 3SMA units, and we respect the client-server approach required by the protocol. On the contrary, we adopt the MQTT protocol every time an actor of the infrastructure needs to transfer information, even in (near-) real-time, to more than one unit. In this case, the publish/subscribe functionalities offered by MQTT completely satisfy multi-point communication requirements. Given this framework, this section describes the communication scheme used

between the units of our infrastructure. More specifically, we detail how REST has been used for the self-configuration and auto-update phases, and how the MQTT protocol has been exploited to manage faults (i.e., the FLISR service) and to coordinate the state estimation process.

A. Self-configuration and auto-update

Figure 4 illustrates the main steps of the self-configuration process, which is also the first phase running within the infrastructure every time a new 3SMA is installed or (re-) booted on the grid. The self-configuration capability avoids the necessity to manually configure each 3SMA and it also enables remote reconfigurations. These possibilities increase the scalability and maintainability of the whole infrastructure. As stated in Section III, the DeC stores all settings required by each 3SMA. As soon as a 3SMA is deployed and turned on, it sends a configuration request to the DeC, specifying its unique (device) identifier. Once this request is received, the DeC performs two tasks. First, it adds the specific device to its list of active units. Then, it sends back to the 3SMA the full list of settings. When the 3SMA receives its configuration setting back, it initializes its status following the procedure specified by the settings. Immediately after that, it starts collecting and sending data, running the specified algorithms and it publishes and/or subscribes to the MQTT topics indicated in the settings.

The other major feature of the 3SMA is its auto-update capability. Auto-update is illustrated in the bottom frame of Figure 4. The first step of this phase is similar to the self-configuration process. However, in the auto-update phase, the 3SMA periodically inquires the DeC to receive its settings and the DeC replies by updating its configuration. In addition, every time it receives a request from a 3SMA, the DeC stores the time-stamp of such a request, which is needed to keep the list of active and online devices up-to-date. Indeed, if the DeC does not receive an auto-update request from a specific 3SMA, after a certain amount of time (configurable by the user), it marks the 3SMA as "disconnected" until the next auto-update request. It is worth noticing that the self-configuration and the auto-update features increase the flexibility and the configuration and re-configurations speed of the whole system.

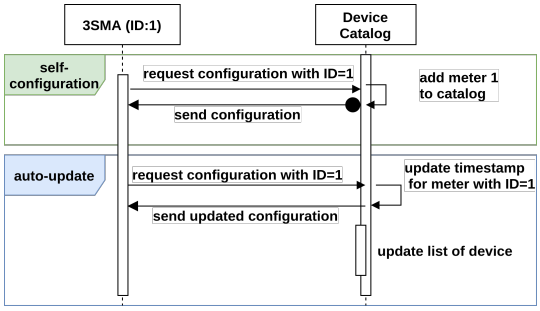

Fig. 4: Self-configuration and the auto-update: The main operational steps of our infrastructure.

B. Fault Location Isolation and Service Restoration

As illustrated in Figure 5, when a fault is detected, the 3SMA seeks the bus pair on which the fault occurred. Once the search is finished, the identifiers of these 2 buses, and the distance of the fault from the first bus of the pair, are published using the MQTT protocol. At this point, the remote service application in charge of the network reconfiguration, which is subscribed to the MQTT protocol, receives these identifiers and computes the new configuration of the network. While doing that, this application also takes into account the detected fault, so that the line between the two faulted buses is not included. Once this step is performed, the network configurator sends a command to the switches that need to open or close the connection to reconfigure the grid. In our tests, all open switches that could receive this kind of actuation commands were simulated on the real-time simulator. At the same time, the information related to the new topology of the grid is transferred to the DeC. When the DeC receives the updated topology, it uses MQTT to store and forward this information to all 3SMA units that are affected by the configuration change, such that these 3SMAs can update their settings accordingly. If, for any reason, a 3SMA device does not receive such update settings in (near-) real-time, it will receive them when it runs its auto-update routine (please, see Section IV-A). Notice that updating the setting is particularly important for the correct execution of the SE. When a network reconfiguration occurs, a 3SMA may need to run the algorithm for a different portion of the grid. In this case, the new settings are needed to reconfigure both the SE algorithm and the MQTT communication protocol.

C. State estimation

As mentioned in the section above, our distributed SE algorithm includes three layers: The concentrator-level, the LV grid-level, and the MV grid-level. Figure 6 describes how the MQTT protocol communicates results between different layers. Each concentrator-level SE is an MQTT publisher and uses the data coming from the downstream of an LV grid node as input data. Once the computation is finished, the results are published through the MQTT protocol with a topic that identifies the associated LV node. The LV grid-level SE algorithm is subscribed to the MQTT topics that identify each concentrator of the corresponding portion of

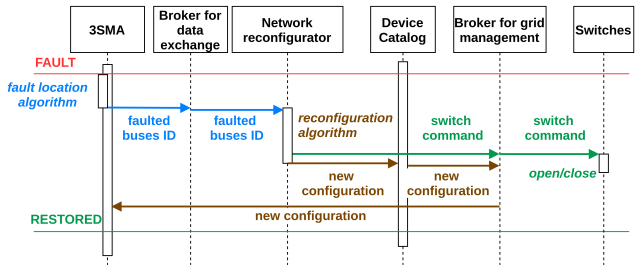

Fig. 5: Fault detection and restoration process: Communication flow among the different units of the grid.

the grid to receive their SE results. These results are used as input to evaluate the status of the low voltage grid. After the previous evaluation, the new results produced by each LV state estimation are published using the MQTT protocol that identifies the corresponding MV portion of the grid. Therefore, they are received by the MV grid-level SE. In turn, this level executes the last part of the distributed algorithm. The overall results are then published, using, again, the MQTT protocol. Each potentially interested remote service for grid management can obtain this information by subscribing to the related topic.

V. Case Study and Experimental Setup

In this section, we describe our testing methodology for the 3SMA unit and the distributed metering infrastructure defined in Section III. To perform the verification phase, we exploit the multi-model co-simulation platform introduced by Barbierato et al. [39]. This platform provides an environment to realistically simulate different smart grid scenarios. We focus on the simulation performed with Hardware- and Software-in-the-Loop.

To perform our simulations on different configuration scenarios, we model a small portion of an urban distribution grid with two feeders and one normally open switch. With this network, we simulate a small portion of the real distribution grid of Turin, a large city in the northwest district of Italy. Figure 7 represents the topology of the case study and the location of the fault. Overall, the grid is composed of two primary and twelve secondary substations, with a high voltage of 220 KV and a medium voltage of 22 KV. All twelve secondary nodes supply residential LV grids. In our tests, we also consider a constraint on the radial operation of the grid. It is important to notice that the switches are modeled as IoT devices simulated on the Opal-RT. Once the fault is isolated, we use this framework to send the necessary actuation messages (adopting the MQTT protocol) to all involved switches to restore the power supply.

To run the simulations of the power network, we use the Opal-RT© Digital Real Time Simulator. The model of the grid, implemented in MATLAB© Simulink using the RTLab© software provided by Opal, can be compiled and uploaded in the simulator. The simulator is able to execute the simulation with a fixed time step of 250 μs. During the simulation, the Opal-RT can provide up to 16 analog outputs. During our tests, we also simulate all sensing devices that are required on a real-world network. The output signals of the Opal-RT are scaled in the range $[0, +10]$ V to respect the input

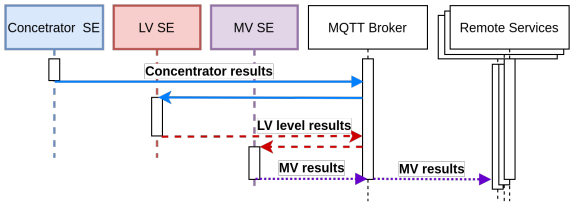

Fig. 6: Estimation process: Communication flow among the different layers and units of the grid.

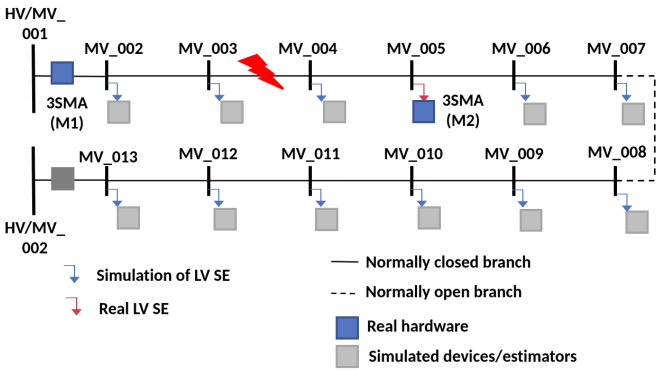

Fig. 7: Our simulated grid: Its topology with the location of all 3SMA prototypes and all other simulated devices.

range of the DAQ. Thus, the DAQ is directly connected to Opal-RT via the analog outputs. Figure 7 shows the topology of the grid implemented in the simulation, the location of the 3SMA units, and the location of the simulated devices (both smart meters and switches). To perform our test with Hardware-in-the-Loop, we substitute 2 out of the 14 smart meters by our real 3SMA prototypes. On the one hand, our 3SMAs are connected to the Opal-RT with their DAQs to sense and collect power measurements from a primary and a secondary substation (respectively, M1 and M2, in Figure 7). On the other hand, they are connected to the Internet to be automatically configured and to periodically send information (as described in Section IV). To perform more realistic simulations, the simulated devices (i.e., the remaining smart meters and switches) send and receive information via the Ethernet connection of the Opal-RT. Among the simulated devices, we also include those deployed at the low voltage level running the state estimation.

As described in Section II, one of the crucial aspects for any service implemented on a smart grid is the communication latency required to fulfill a request (measured as the time passed from the moment in which the event has been discovered and the final actuation). To take into account the Internet congestion during our evaluation, we use some of the features provided by our multi-model co-simulation platform [39]. Indeed, this platform integrates different network simulators to realistically simulate a Metropolitan Area Network (MAN). For our simulations, we choose Mininet [40]. Figure 8 reports the schema of our MAN backbone model. Nowadays, MAN backbones typically leverage upon fiber-optic links deployed across our cities that interconnect different backbone routers forming a ring configuration. Fiber optic links usually guarantee 100 Mbps connections and, in our model, they are full-duplex with a maximum length of about 10 Km and zero losses (Figure 8 represents these links with red edges). In Figure 8, we supposed that all smart devices, simulated in Opal-RT, and the two 3SMAs prototypes are connected to the same backbone router R1.

The remote service for Network Reconfiguration, the DeC, and the two MQTT message brokers, are connected to routers

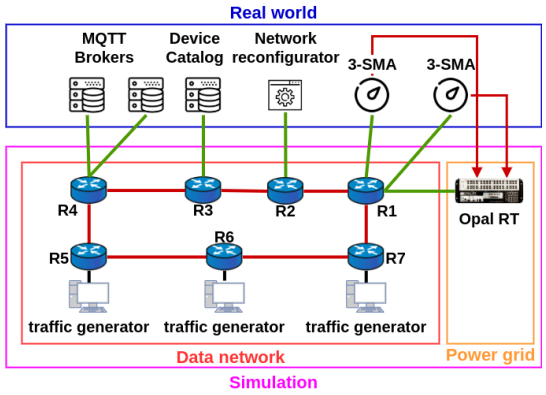

Fig. 8: Our MAN backbone model.

R2, R3, and R4, respectively. R1, R2, R3, and R4 are connected to their respective sub-networks through a 10 Mbps full-duplex link, with a maximum length of about 1 Km, and zero losses (Figure 8 represents these links with green edges). Finally, routers R5, R6, and R7 serve three other different sub-networks that generate background traffic with different rates to realistically congest the MAN by exploiting iPerf [41]. To avoid bottlenecks, the links between these last three routers and their traffic generators are 200 Mbps full-duplex, with a maximum length of about 1 Km, and zero losses (Figure 8 represents these links with black edges). Thus in our simulations, the 3SMAs, Opal-RT, DeC, network reconfiguration service, and message brokers are in different locations in the same metropolitan area and communicate over the internet.

Using this framework, we have been able to run a test for two scenarios: One including normal operations and one with some fault occurrences. In the first case, each 3SMA collects the data coming from the analog output of the Opal-RT, namely the 3 voltage and the 3 current values of the 3-phase signal. The data collected are continuously analyzed to check whether a fault has occurred and the state estimation algorithm is executed every minute. In the second scenario, whenever a fault (triggered using the graphical user interface of the real-time simulator) is identified by a 3SMA, the FLISR routine is automatically started to find the location of the fault, isolate it, and restore the service.

In our laboratory set-up, we employ, together with an Opal-RT and the two 3SMA prototypes, four different desktop computers to run the network reconfiguration services, the DeC, the two MQTT message brokers (we choose the Eclipse Mosquitto [42]), and the Mininet Network Simulator. It is worth noting that we implemented two different versions of our 3SMA. The first one follows the hardware specifications discussed in Section III-B, i.e., it uses a Raspberry Pi 3 Model B connected via USB with the DAQ provided by Measurements Computing™ [35]. Hereinafter, we refer to this 3SMA as 3SMA-RPi. The second version of 3SMA is used to assess the performance of the first one, and we implement it with a Laptop computer connected with a more precise (and expensive) data acquisition board, namely the BNC-2120[43] and the DAQCard-6062[44]. These boards support a sampling frequency up to 500 KS/s per channel, with smaller input noise, and a quite higher resolution. Hereinafter, we refer to this 3SMA as 3SMA-Laptop.

All these devices belong to the same dedicated local area network. They are physically connected through a 100 Gbps Ethernet switch that introduces a negligible communication latency of about 10 ns. Gigabit Ethernet equipment provides backward compatibility to older 100 Mbps and 10 Mbps legacy Ethernet devices [45]. All the traffic generated by this equipment is routed through the computer where the virtual MAN runs in Mininet. Table II reports all the parameters required to our model to configure the simulation scenario in terms of both power grid and MAN.

VI. Experimental Results

In experimental analysis, we consider two sets of experiments. The first one includes evidence on fault location isolation and service restoration (FLISR). The second one reports data on state estimation (SE). Each aspect is analyzed in one of the following subsections.

A. Fault Location Isolation and Service Restoration

In this section, we focus on the efficiency of our 3SMA-RPi unit to implement the FLISR procedure. We evaluate the computation efficiency and the accuracy of our 3SMA-RPi architecture and we also compare it with its laptop implementation, i.e., the 3SMA-Laptop, exploiting a more precise data acquisition board (see Section V). As illustrated in Figure 5, one of our targets is to minimize the time required to reconfigure the network when a fault occurs. In a real-world scenario, a crucial factor in measuring this time is the congestion of the MAN, which can strongly influence the communication delays. To take this aspect into account, we consider in our scenario the data network simulation previously described. We also tested different scenarios by changing the location of the fault in the grid to evaluate how this aspect influences our results.

During our tests, the 3SMA-RPi was always able to correctly detect the presence of a fault on the grid, with no false positive or false negative. Moreover, the fault location algorithm was always able to correctly identify the two buses among which the fault occurred and locate the fault with an error of ± 50 m. Notice that this relatively low accuracy

	Node type	# of Node	Voltage	Distance among nodes
Electric parameters	HV to MV substation	2	220kV/22kV	n.d.
	MV to LV substation	12	22kV/230V	220 m

	Node type	Connection type	Transfer rate
ICT parameters	Backbone routers connection	full-duplex,max 10 Km,zero losses	up to 100 Mbp
	R1,R2,R3,R4 towards users	full-duplex,max 1 Km, zero losses	up to 10 Mbp
	R5,R6,R7 towards traffic generators	full-duplex,max 1 Km, zero losses	up to 200 Mbp
	Physical connection	Gigabit Ethernet switch	up to 100 Mbp

TABLE II: Our simulated grid: The main parameters used by our model to set-up the simulation.

is strongly related to the quality of the USB DAQ used to collect the values of voltage and current. To double-check this accuracy, we repeated our test replacing the 3SMA-RPi with the 3SMA-Laptop. With the new unit, the error to locate the position of the fault was reduced to a value of ± 10 m. However, even if this accuracy analysis confirms our expectations, it is worth noticing that to correctly complete the FLISR process the meter just needs to correctly identify the two buses among which the fault occurred. For example, in Figure 9 the position of the fault does not have any meaningful effect on the execution time. This was equal to 8.17 ± 0.14 s for the 3SMA-RPi and to 0.531 ± 0.008 s on average using the 3SMA-Laptop.

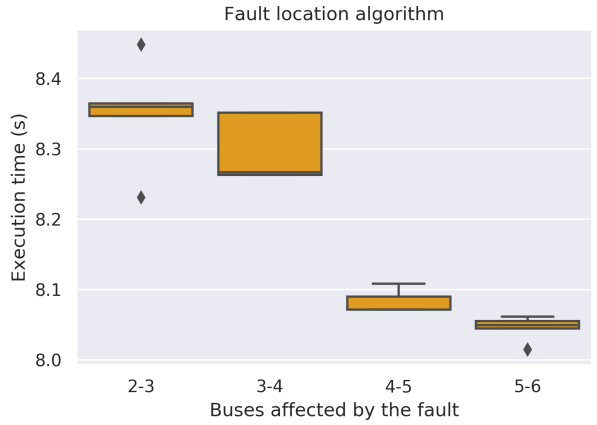

Fig. 9: The fault location phase: Execution times exploiting the 3SMA-RPi unit.

To properly evaluate our approach to fault restoration, the last factor we need to take into account is the communication delay. To do that, we measure the elapsed time between the moment in which our smart meter sends the MQTT message containing the information of the buses affected by the fault and the moment it receives the new topology from the network configurator through another MQTT message. The latency time is shown in Figure 10, with different levels of network congestion and a packet size of 1.4 KByte. As expected, the latency in the communication is mostly related to the congestion level. Even if this measure includes the execution time of the Network Reconfiguration algorithm, this time does not particularly affect the delay due to the transmission time as it is always lower than 0.028s. In the worst scenario of our simulation, with 100% of congestion, the medium latency time was equal to 0.49 s and the maximum latency time was 1 s. However, Internet service providers try to avoid such high network congestion as, in long periods, it could lead to the collapse of MAN itself. Thus, we can assume that the communication latency would be always smaller than 1 s.

To further detail our analysis, Figure 11 reports the time required by all main steps of our procedure in the worst-case scenario, i.e., their higher values obtained running the procedure several times. From the graphic, it is possible to notice that during our simulations the maximum time needed to perform the whole FLISR is about 10.5 s for the 3SMA-RPi unit. This maximum time decreases to approximately 2.5 s for the 3SMA-Laptop device. As shown in Figure 11, the majority of this time is taken by the execution of the fault location algorithm running on-board of our smart meter. On the contrary, the latency due to the data communication and execution of the Network Reconfigurator represents a minor contribution. Interestingly, this result also highlights that, with a relatively simple infrastructure and low-cost hardware devices, it is possible to drastically improve the time required by the operator to restore the energy distribution after a fault.

As a final consideration, Figures 11 and 12 show that the advantage of our approach in terms of service restoration. With the methodology traditionally used in real scenarios, the restoration phase can require from 40 minutes to more than 80 minutes [46] (Figure 12). With our infrastructure, the same process is completely automated and, as illustrated by Figure 11, just a few seconds are required for the entire process even in the worst-case scenario. Thus, the difference of about 8 s in the computation time between the 3SMA-RPi and the 3SMA-Laptop unit can be considered as negligible. As a consequence, our approach not only improves the quality of the service delivered to the customers, but it also helps the system operators to reduce the maintenance costs and it allows them to avoid penalties for long service failures.

B. State Estimation Results

As far as the state estimation algorithm is concerned, our target is to prove that our 3SMA-RPi is able to perform each step of the algorithm as efficient as the 3SMA-Laptop unit. We also want to prove that our 3SMA-RPi is able to complete each step in less than 1 minute, as this is the time occurring between two consecutive states evaluations. As in the previous section, to ensure a more realistic condition, we included in our setting the data network simulation to test the algorithm with different levels of network congestion.

Our first test focuses on verifying that the infrastructure and our two 3SMA prototypes are able to communicate, run the distributed algorithm, and update the MQTT topic in case of a network reconfiguration. The state estimation has been verified using both our prototypes running in two possible configurations, i.e., with 3SMA-RPi executing the LV grid-level SE and the 3SMA-Laptop running the MV grid-level

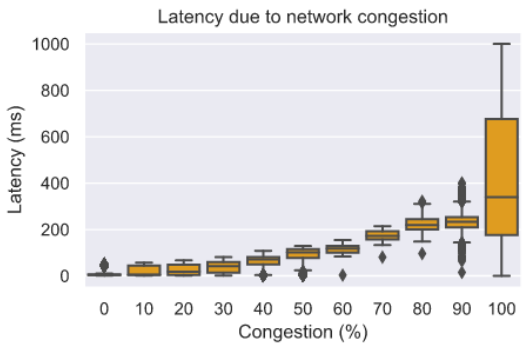

Fig. 10: Latency time as a function of the level of traffic over the network. The packet size is equal to 1.4 KByte,

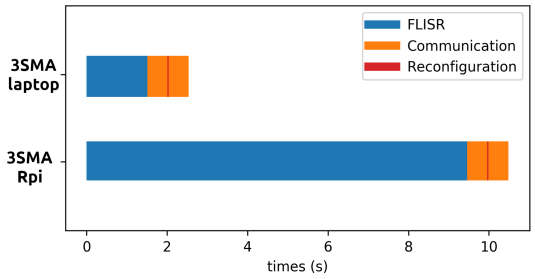

Fig. 11: The total FLISR time for the proposed infrastructure: A comparison between the low cost 3SMA-RPi unit and the more powerful but expensive 3SMA-Laptop.

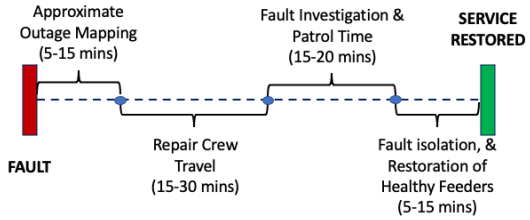

Fig. 12: Legacy process for fault management on real infrastructures [46].

SE, and vice-versa. This enables a fair and head-to-head comparison in terms of performance and accuracy.

As shown in Figure 13, when 3SMA-RPi (M2 in Figure 7) runs the LV grid-level SE, it takes about 11.9±0.4 s for each computation. When it executes the MV grid-level SE (M1 in Figure 7), it takes 4.13±0.03 s. Repeating the very same tests with the 3SMA-Laptop unit, the two steps are executed in 0.127±0.003 s and 0.057±0.003 s, respectively. As the same version of the algorithm is used on both units, the accuracy achieved is the same and we did not notice any difference in terms of outputs accuracy. Thus, we can claim that our 3SMA-RPi unit is perfectly suitable to address such computation, even if it takes a longer time for its execution. To better analyze our infrastructure, as in the previous cases, we need to consider the latency time added to the process by the communication network. Indeed, as introduced in the previous sections, the MQTT is used to send partial results from the smart meter performing the LV grid-level SE to the one running the MV grid-level SE algorithm. Since the size of the messages is quite similar to the one of FLISR (i.e., 1.4 KByte), the maximum possible latency in the worst case is less than 1 s, as shown in Figure 10. Since the proposed procedure runs a state estimation algorithm every minute, the 3SMA-RPi unit is fully compliant with this requirement as it is able to complete all phases, i.e., LV SE and MV grid-level SE, in less time. Moreover, in case of a fault, after the topology changes occurred during the restoration phase, the MQTT topic and the configuration of the SE algorithm are correctly updated according to the portion of the MV network monitored by the 3SMAs. As discussed in Section VI-A, this reconfiguration can last about 10.5 s. During the network reconfiguration, in case one of the 3SMAs deployed across the grid is disconnected from the Internet, it will receive the

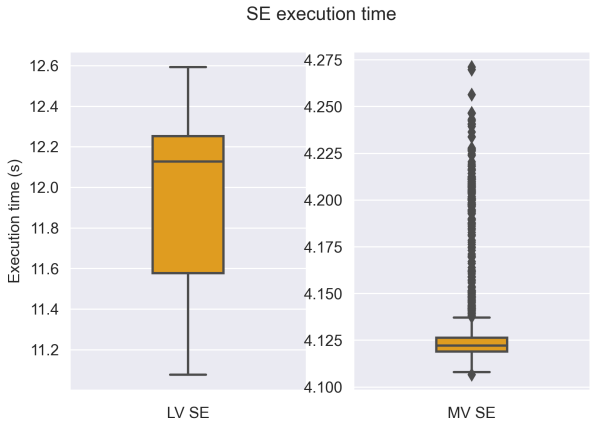

Fig. 13: Execution time of the state estimation for both LV and MV grid-level SE running on our 3SMA-RPi unit.

new setting when the next auto-update routine starts. The time period between two consecutive auto-update routines depends on the end-users settings configured in the DeC (see Section IV-A).

When our algorithm runs on the 3SMA-RPi, even if the memory usage is just at 3.9%, the CPU usage is always around 100%, with some outliers that even exceed that value. This result may look liks an issue, but there are two main factors to take into consideration:

- The algorithm is written in Matlab© and it runs on 3SMA-RPi using Octave, thus, the code is not optimized at all for embedded devices.
- The target of our test is to assess the possibility to run a distributed algorithm on the 3SMA prototype, not to optimize our application for real usage.

Once these two considerations are correctly taken into account, our results look promising. Even without any code optimization and with the adoption of low-cost hardware devices, the 3SMA is perfectly able to run the different levels of SE and to satisfy the time threshold required by the algorithm. In conclusion, using inexpensive hardware devices reduces costs and improves scalability but it also implies the necessity to carefully design and optimize the applications deployed on the devices.

C. Final Remarks on Communication Latency

Assessing the impact of data transmission in existing communication networks is crucial to address novel service requirements in terms of data transmission latency as well. Indeed, such latency can affect the operational status of the smart grid. The IEC 61850 standard [12] defines the communication requirements to be addressed in power distribution networks. Table III reports the performance classes defined by the standard. Our experimental results on time delay, performed by exploiting Mininet to simulate a realistic virtual MAN, satisfy the requirements of classes *TT0*, *TT1*, and *TT2*. This is confirmed when the MAN congestion rate is lower than 90%, as in this case, the maximum latency never exceeds 500 ms. When the congestion rate is 100%, the

Performance Requirements	Performance Classes	Values [ms]	Example of services
Transfer time	TT0	>1,000	Files, events log contents, SCADA
	TT1	1,000	Events, alarms
	TT2	500	Operator commands
	TT3	100	Slow automation interactions
	TT4	20	Fast automation interactions
	TT5	10	Releases, status changes
	TT6	3	Trips, blockings

TABLE III: The IEC 61850 [12] standard: Communication requirements and performance classes for power systems.

median and the maximum latency values are about 350 ms and 1 s, respectively. However, as previously discussed, this latter scenario is very critical and must be avoided because it could lead to the collapse of the MAN itself.

VII. Conclusion and future work

In this paper, we propose a low-cost smart meter architecture and a distributed software infrastructure for AMI, which can collect data from the network, communicate with other entities, and offer different features to each actor connected to the network. The usage of well-known IoT technologies ensures a high compatibility with other devices and third-party services. Our 3-phase meter architecture is able to run multiple software applications, either on-board or distributed over the network, and to auto-update its status when required. Moreover, new algorithms can be added, updated, or removed on the fly thanks to its auto-configuration capability. This ensures high compatibility with other device management tools and communication protocols such as DLMS/COSEM [10], SML [11], and IEC 61850 [12], which could be added and treated as any other data processing algorithm.

To verify the characteristics of the entire infrastructure, we run real-time simulations with different configurations and settings. The experimental results prove that the system can identify and locate grid problems at high speed, restoring the original functionalities faster than any other state-of-the-art solution. Moreover, the time delay on data transmission, estimated by including a MAN simulator in our realistic test-bed environment highlights that the resulting latency respects the limits imposed by the IEC 61850 standard [12].

Among the possible future work, we would like to mention that the proposed distributed software platform and the 3SMA will be integrated in a wider distributed multi-model co-simulation environment [47]. The target of this effort is to unlock other possible scenarios and test additional multi-energy services such as optimal management of renewable energy sources. In addition, we also plan to further optimize our algorithms and software running on our 3SMA to improve their performances when implemented on embedded systems with reduced computation power.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[2] Y. Saleem, N. Crespi, M. H. Rehmani, and R. Copeland, "Internet of Things-Aided Smart Grid: Technologies, Architectures, Applications, Prototypes, and Future Research Directions," *IEEE Access*, vol. 7, pp. 62 962–63 003, 2019.

[3] A. Ghosal and M. Conti, "Key management systems for smart grid advanced metering infrastructure: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2831–2848, thirdquarter 2019.

[4] A. Ghasempour and J. Lou, "Advanced metering infrastructure in smart grid: Requirements, challenges, architectures, technologies, and optimizations," in *Smart Grids: Emerging Technologies, Challenges and Future Directions*. Nova Science Publishers, 2017.

[5] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 3125–3148, 2019.

[6] A. Ghasempour, "Optimized scalable decentralized hybrid advanced metering infrastructure for smart grid," in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov 2015, pp. 223–228.

[7] M. Nardello, M. Rossi, and D. Brunelli, "An innovative cost-effective smart meter with embedded non intrusive load monitoring," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Sep. 2017, pp. 1–6.

[8] A. Zidan, M. Khairalla, A. M. Abdrabou, T. Khalifa, K. Shaban, A. Abdrabou, R. El Shatshat, and A. M. Gaouda, "Fault detection, isolation, and service restoration in distribution systems: State-of-the-art and future trends," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2170–2185, Sep. 2017.

[9] F. Al-Turjman and M. Abujubbeh, "Iot-enabled smart grid via sm: An overview," *Future Generation Computer Systems*, vol. 96, pp. 579–590, 2019.

[10] "Dlms: Device language message specification — dlms." [Online]. Available: https://www.dlms.com/home

[11] "Synchronous modular meter - vde fnn." [Online]. Available: https://www.vde.com/de/fnn/arbeitsgebiete/imesssystem/lastenhefte/lastgangzaehler

[12] P. CODE, "Communication networks and systems in substations–part 5: Communication requirements for functions and device models," 2003.

[13] Y. Yan, R. Q. Hu, S. K. Das, H. Sharif, and Y. Qian, "An efficient security protocol for advanced metering infrastructure in smart grid," *IEEE Network*, vol. 27, no. 4, pp. 64–71, 2013.

[14] F. Ye, Y. Qian, and R. Q. Hu, "A security protocol for advanced metering infrastructure in smart grid," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 649–654.

[15] A. Meloni and L. Atzori, "A cloud-based and restful internet of things platform to foster smart grid technologies integration and re-usability," in *2016 IEEE International Conference on Communications Workshops (ICC)*, 2016, pp. 387–392.

[16] S. Chen, H. Wen, J. Wu, W. Lei, W. Hou, W. Liu, A. Xu, and Y. Jiang, "Internet of things based smart grids supported by intelligent edge computing," *IEEE Access*, vol. 7, pp. 74 089–74 102, 2019.

[17] Y. Yan and W. Su, "A fog computing solution for advanced metering infrastructure," in *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D)*, 2016, pp. 1–4.

[18] Q. Ou, Y. Zhen, X. Li, Y. Zhang, and L. Zeng, "Application of internet of things in smart grid power transmission," in *2012 Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing*, 2012, pp. 96–100.

[19] J. Lloret, J. Tomas, A. Canovas, and L. Parra, "An integrated iot architecture for smart metering," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 50–57, 2016.

[20] M. Hussain, M. Beg *et al.*, "Fog computing for internet of things (iot)-aided smart grid architectures," *Big Data and cognitive computing*, vol. 3, no. 1, p. 8, 2019.

[21] M. Forcan and M. Maksimović, "Cloud-fog-based approach for smart grid monitoring," *Simulation Modelling Practice and Theory*, vol. 101, p. 101988, 2020.

[22] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, "Intelligent edge computing for iot-based energy management in smart cities," *IEEE Network*, vol. 33, no. 2, pp. 111–117, 2019.

[23] Y. Yan and W. Su, "A fog computing solution for advanced metering infrastructure," in *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*. IEEE, 2016, pp. 1–4.

[24] A. Angioni, G. Lipari, M. Pau, F. Ponci, and A. Monti, "A low cost pmu to monitor distribution grids," in *2017 IEEE International Workshop on Applied Measurements for Power Systems (AMPS)*. IEEE, 2017, pp. 1–6.

[25] P. Romano, M. Paolone, T. Chau, B. Jeppesen, and E. Ahmed, "A high-performance, low-cost pmu prototype for distribution networks based on fpga," in *2017 IEEE Manchester PowerTech*. IEEE, 2017, pp. 1–6.
[26] P. Romano, M. Paolone, J. Arnold, and R. Piacentini, "An interpolated-dft synchrophasor estimation algorithm and its implementation in an fpga-based pmu prototype," in *2013 IEEE Power & Energy Society General Meeting*. IEEE, 2013, pp. 1–6.
[27] J. Gallano, V. Malvas, J. Quirona, R. Soriano, M. Pacis, and F. Cruz, "Design and implementation of phasor measurement unit with iot technology," in *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 2020, pp. 1–6.
[28] T. Sirojan, S. Lu, B. T. Phung, and E. Ambikairajah, "Embedded edge computing for real-time smart meter data analytics," in *2019 International Conference on Smart Energy Systems and Technologies (SEST)*, 2019, pp. 1–5.
[29] Y.-Y. Chen, Y.-H. Lin, C.-C. Kung, M.-H. Chung, and I.-H. Yen, "Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes," *Sensors*, vol. 19, no. 9, 2019.
[30] P. A. Pegoraro, A. Meloni, L. Atzori, P. Castello, and S. Sulis, "Pmu-based distribution system state estimation with adaptive accuracy exploiting local decision metrics and iot paradigm," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 4, pp. 704–714, 2017.
[31] M. Pignati, M. Popovic, S. Barreto, R. Cherkaoui, G. D. Flores, J.-Y. Le Boudec, M. Mohiuddin, M. Paolone, P. Romano, S. Sarri *et al.*, "Real-time state estimation of the epfl-campus medium-voltage grid by using pmus," in *2015 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2015, pp. 1–5.
[32] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 2, pp. 115–150, 2002.
[33] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, p. 114–131, Jun. 2003.
[34] MQTT, "http://mqtt.org," Accessed on June 2021. [Online]. Available: http://mqtt.org
[35] "USB 12-Bit DAQ Devices with 8 Analog Inputs - Measurement Computing," Accessed on June 2021. [Online]. Available: https://www.mccdaq.com/usb-data-acquisition/USB-200-Series.aspx
[36] A. Estebsari, E. Pons, E. Bompard, A. Bahmanyar, and S. Jamali, "An improved fault location method for distribution networks exploiting emerging lv smart meters," in *2016 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*. IEEE, 2016, pp. 1–6.
[37] M. Pau, E. Patti, L. Barbierato, A. Estebsari, E. Pons, F. Ponci, and A. Monti, "A cloud-based smart metering infrastructure for distribution grid services and automation," *Sustainable Energy, Grids and Networks*, vol. 15, pp. 14–25, 2018.
[38] ——, "Design and accuracy analysis of multilevel state estimation based on smart metering infrastructure," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 11, pp. 4300–4312, 2019.
[39] L. Barbierato, A. Estebsari, L. Bottaccioli, E. Macii, and E. Patti, "A distributed multimodel cosimulation platform to assess general purpose services in smart grids," *IEEE Transactions on Industry Applications*, vol. 56, no. 5, pp. 5613–5624, 2020.
[40] "Mininet," Accessed on June 2021. [Online]. Available: http://mininet.org/
[41] V. GUEANT, "iperf," Accessed on June 2021. [Online]. Available: https://iperf.fr/
[42] "Eclipse Mosquitto," Accessed on June 2021. [Online]. Available: https://mosquitto.org/
[43] "BNC-2120." [Online]. Available: https://www.ni.com/en-us/support/model.bnc-2120.html
[44] "DAQCard-60602." [Online]. Available: https://www.ni.com/en-us/support/model.daqcard-6062.html
[45] "IEEE 802.3-2018 - IEEE Standard for Ethernet," Accessed on June 2021. [Online]. Available: https://standards.ieee.org/standard/8023-2018.html
[46] J. R. Agüero, "Applying self-healing schemes to modern power distribution systems," in *2012 IEEE Power and Energy Society General Meeting*. IEEE, 2012, pp. 1–4.
[47] D. S. Schiera, L. Barbierato, A. Lanzini, R. Borchiellini, E. Pons, E. Bompard, E. Patti, E. Macii, and L. Bottaccioli, "A distributed multimodel platform to cosimulate multienergy systems in smart buildings," *IEEE Transactions on Industry Applications*, vol. 57, no. 5, pp. 4428–4440, 2021.