A Fog-based Architecture for Phasor Measurements in Smart Grids

Stefano Galantino, Fulvio Risso
Dept. of Computer and Control Engineering
Politecnico di Torino
Torino, Italy
{name.surname}@polito.it

Andrea Cazzaniga, Fabrizio Garrone, Roberta Terruggia,
Riccardo Lazzari
Tecnologie di Trasmissione e Distribuzione
Ricerca sul Sistema Energetico
Milano, Italy
{name.surname}@rse-web.it

***Abstract*—This paper investigates the application of Kubernetes and Edge computing technologies to operate IT services in the context of power systems and smart grids. Traditional services for grid monitoring such as Phasor Measurement Units (PMUs) and Phasor Data Concentrators (PDCs) require a centralized architecture and a rigid networking infrastructure in order to properly function, which today is only achieved at the High Voltage (HV) transmission level. Furthermore, manual intervention is often the only option for PMUs/PDCs maintenance. In this work, the traditional PMU/PDC services were deployed as docker-containers in a decentralized Kubernetes cluster, which can represent any kind of geographically dispersed TCP/IP network. By leveraging remote orchestration, several key benefits are achieved: (1) no manual reconfiguration of the PMU-PDC communications upon network reconfiguration, (2) automatic PMU traffic redirection in case of PDC service redeployment in a different location, and (3) reduced data-loss upon PDC failure and enhanced overall system resiliency due to minimized ICT services down-time.**

***Index Terms*—Cloud, Edge, Infrastructure Resiliency, Smart Grid**

I. Introduction

The proliferation of renewable energy sources (RES) into power systems poses new challenges to grid operators, who have to assure the proper and reliable operation of the electrical grid [1]. To this end, efficient and accurate state estimation (SE) is fundamental to achieve near-real time monitoring, especially at the distribution level where this function plays a central role in the implementation of smart grid features [2]. Research on distribution system state estimation (DSSE) began near 1990 [3] and recently, many projects have introduced the applications of phasor measurement units (PMUs) in the distribution networks [4].

PMUs are well-known devices that provide accurate and synchronized voltage/current phasors with a sampling rate up to 60 measurements-per-second and are commonly used in transmission systems [5]–[7], but they are experiencing an ever-growing interest also for distribution networks, in order to implement novel schemes for better control and fault location purposes [8]. Their adoption in the distribution layer may guarantee benefits in a number of use cases [9], [10].

Although PMUs can improve distribution network monitoring, a crucial aspect of future power systems is their resiliency also in the case of natural disasters (e.g., extreme weather events, earthquakes, etc.) or cyberattacks.

Thus, the resiliency of the power grid necessarily relies also on a robust ICT infrastructure. In fact, (*i*) the execution of critical services must be monitored and guaranteed also in case of application or node failures, either in case of unexpected disruptive events or possible cyber attacks; and (*ii*) both the power grid and the ICT infrastructure should survive to partition events and any possible network failure that might isolate one or more sites of the electrical power system. As of today, PMUs are mostly installed only on HV transmission lines and the process of gathering, validating, and analyzing PMUs data is rather complex and based on configurations that require detailed knowledge of both the power grid network as well as the IT communication network [7]. A change in such configurations often requires manual intervention, which is heavily time-consuming, hard to upscale when the complexity of the system grows and ultimately hinders the system resiliency.

The 2021 IEEE guidelines on PMUs installation [11] highlights the importance of "data quality" to support real-time mission critical application at the utility scale, where it is stressed that data availability is as crucial as data accuracy. Furthermore, it highlights how the planning of PMUs installation shall carefully evaluate the IT network capability to assess which services can be guaranteed.

In this paper we propose a novel IT architecture able to support the acquisition of phasor data coming from PMU. The architecture is based on Kubernetes, as the state-of-the-art Cloud/Edge orchestration technology, and it is designed with a few objectives in mind: (*i*) be robust concerning IT network reconfigurations, (*ii*) full automation of the PMUs/PDCs configuration parameters and i/o data streams, and (*iii*) real-time service monitoring and automatic restoration upon failure.

These achievements are essential for grid operators to upscale their maintenance and operation of PMUs systems to thousands of devices. Furthermore, new data access policies for future applications can be implemented easily from the central orchestrator, without requiring ad hoc solutions.

The rest of the paper is organized as follows. Section II details the requirements and the related work, as well as the proposed architecture in Section III. Section IV presents a pre-

TABLE I: Number of primary and secondary substations in Italy over years 2011-2019.

	2011	2012	2013	2014	2015	2016	2017	2018	2019
Primary substations	2.134	2.144	2.159	2.168	2.188	2.195	2.199	2.203	2.200
Secondary substations	432.074	436.204	438.359	439.558	441.056	442.418	443.774	445.159	446.410

liminary experimental validation of the proposed architecture, and finally Section V concludes the paper.

II. Functional Requirements

This paper focuses on the observability of the smart grid, which is based on two main devices: (*i*) **PMUs** (Phasor Measurement Units), devices that constantly collect relevant data of the power system and that act as simple data producers; (*ii*) **PDCs** (Phasor Data Concentrators), software services that collect input data coming from PMUs for data handling, processing and storage. A PDC can acts both as data producer and consumer, as data can be further forwarded to a hierarchically superior PDC. PMUs require physical measuring hardware, hence their location is bounded to the edge of the locations where the above devices are installed, while PDCs are just software services and hence do not have such strict requirements. Given the critical issues related to the electrical grid, whenever possible, additional resiliency features must be built on top of these services. Specifically, robustness properties must be guaranteed for both service execution (self-healing), and the generated data (no losses). Prior research proposed traditional virtualization technologies (e.g., VMs) to meet the above requirements [12], [13]. Although promising, traditional virtualization is not able to achieve the same recovery time, as well it demands more resources compared to lightweight virtualization.

A. Scalability

Table I shows the (growing) number of primary and secondary substations across the years reported in [14] by *e-distribuzione*, the main Italian company operating in the electrical distribution sector. Data shows that the power grid includes hundreds of thousands of peripheral sites that, in the smart-grid perspective, should be able to operate even if isolated. In this respect, not only IT centralized control is definitely not appropriate, but the overall architecture must be highly scalable, as each site should be able to cope with possible local failures as well as disconnections from the network backbone.

B. Data resiliency

Data resiliency is a key requirement for monitoring services, as historical data is fundamental for statistical or post-incident analysis [15]. Furthermore, smart grid networks should manage a huge amount of data, coming from many different types of devices, with different requirements [16]. Hence, data replication or, in some cases, regular backups, is crucial to withstand hardware and network failures.

A first level of data resiliency should be achieved at a site level, so that data are not tied to a single device but rather replicated across different storage devices. Another level of data resiliency should be achieved at a bigger scope, duplicating relevant data e.g., on cloud data centers, hence facilitating also the analysis of the state of the entire national power grid.

C. Communication latency and service redundancy

Although control services are not the objective of this work, the proposed architecture should also foster the minimization of network path lengths. This has positive impacts on the communication latency (with beneficial effects on critical control services for the power grid [15]), better resiliency, and reduced network perturbations (packet drops, jitter) due to bufferbloat (although the former seems to have limited importance in modern edge networks [17]). However, differently from the transmission system, in which the IT network topology usually follows the topology of the power grid (often, optical fibers lay together with electrical cables in the same location) and is privately self-managed by the energy provider, the distribution system usually relies on the general-purpose network connectivity provided by telecommunication companies, often through a 3G/4G/5G mobile connection. Consequently, the topology of the power grid does not match the physical topology of the IT network, hence communication between two different sites in the distribution portion of the network requires the transit in the network of the telecommunication provider. This has a huge impact on the overall IT architecture, as the actual topology of the physical IT network is of paramount importance to define where to locate redundant IT services, as well as to optimize communications and reduce latency.

III. Architecture

A distributed edge-based architecture needs to address additional problems compared to common cloud/based architectures. For instance, resources at the edge (e.g., CPU, RAM, storage, number of available devices/servers) are limited compared to the cloud; furthermore, network partitioning events that isolate one or more sites are a possibility that must be taken into account. Therefore each site needs to withstand network partitioning and isolation from the cloud; in addition, local service redundancy must be achieved in each site.

A. Service Orchestrator

Given the recent trend in the industry, our architecture is heavily based on Kubernetes,[1] which includes implementations targeting also low-resource devices, hence not requiring datacenter-grade servers. In addition to its native

[1] https://kubernetes.io.

features (e.g., automatic service restart/re-spawn in case of failure, multi-master capabilities, etc.), it includes a large software ecosystem that can provide well-tested solutions for many common problems, such as data redundancy. Specifically, *K3s*[2] is the Kubernetes distribution chosen for edge sites, which features a very limited resource consumption (CPU, memory, disk), performance close to vanilla Kubernetes [18] and a very simple setup. Orchestrator redundancy has been achieved with multiple masters (active plus standby). The K3s configuration has been further customized to reduce the interval between the detection of a node failure to the re-spawn of the services previously running on the failed node from 5 minutes (default value) to 40 seconds. This has been achieved by modifying the API server options `default-not-ready-toleration-seconds` and `default-unreachable-toleration-seconds` to 20 seconds each.

B. Geographical architecture

Our architecture foresees the creation of an autonomous Kubernetes cluster in each site (either primary or secondary substation), hence guaranteeing resiliency at the site level. However, to provide resiliency also in case of failure of local computing components, the system should be able to leverage services (e.g., another PDC) in a different location, possibly with low communication latency. Given that each power substation is directly connected to the telco network, the operator *micro-POP* (Point Of Presence), which is established in close vicinity of a small set of 5G antennas, might be the perfect place where to locate such redundancy or to deploy non-critical cross-substation services.

While the best location for a PMU is strictly dependent on the power grid [19] (i.e., determined through a proper cost/optimization process), a first level PDC can be installed at each site of the power grid, potentially even on the same device hosting the PMU, collecting the data of the local PMUs and guaranteeing local survivability in case of network outages. Their output stream is sent to the higher level PDC, located inside the telco micro-PoP, where the output can be sent as input to the local state estimator or other applications performing data processing or storage of historical data either locally within the PoP, or in the Cloud, leveraging high-performance computing (HPC) resources for the most demanding applications (e.g., fault analysis) [13]. Fig. 1 shows the resulting topology, where services can be executed either on physical devices (e.g., PMU), rugged embedded servers (e.g., PDC), or edge data centers (e.g., high-level PDC).

Leveraging shared resources provided by telcos, the security of the infrastructure has to be taken into high consideration; consequently, the traffic of the distribution system should flow on a secure logical infrastructure (e.g., MPLS, overlay network, etc.), in a way that clusters, services, and machines of the electricity provider cannot be reached through the Internet.

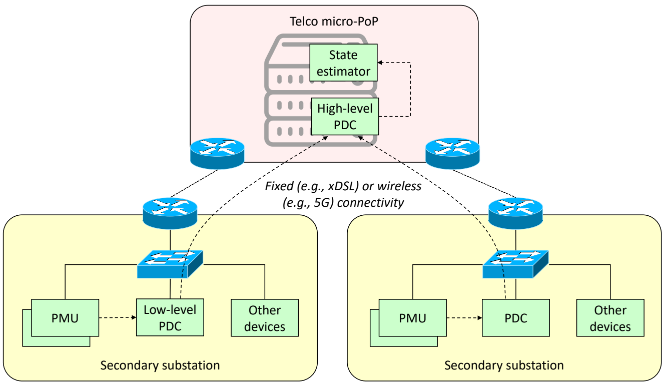

Fig. 1: Topology overview of the proposed architecture.

C. Multi-cluster

Given the huge number of sites (Tab. I), the proposed architecture must define a scalable mechanism to automate the service configuration on each site. To address the multi-cluster management issue, two main approaches have been considered: a federated multi-cluster architecture (based on either KubeEdge or KubeFed), and a centralized approach with a single source of truth (Eve-OS, Rancher Fleet).

The second approach has shown to be the most promising one thanks to the centralized control as well as the single point of management. Specifically Rancher Fleet appears to be a good compromise, providing the advantages of a single source of truth while being able to handle thousands of cluster configurations.[3] Specifically, a cluster in the cloud runs the Fleet *manager* and watches a `git` repository containing all the configurations, while Fleet *agents*, running in peripheral clusters, poll the central manager to initiate the site registration process and to obtain the initial cluster configuration. This process works also in case peripheral clusters are not directly exposed to the public internet and NAT traversal is required.

Each cluster is associated with the appropriate labels to identify the geographical region, area, and type of site (e.g., primary or secondary substation), allowing the Fleet manager to push exactly the configuration referring to the selected cluster, including active services, the required redundancy, number of replicas, and more. For instance, given that each secondary substation requires a PDC service, a single instance of the above service is deployed on all clusters whose labels include `site: secondary-substation`.

D. Data persistency

Kubernetes addresses the data persistence problem through built-in abstractions such as *PersistentVolumes* and *PersistentVolumeClaims*; however, the default driver provides basic data persistence but it does not support data replication. Our solution introduces an additional layer of abstraction for data persistence leveraging the enhanced storage features provided by Longhorn CSI,[4] which features advanced data management

[2] https://k3s.io/

[3] https://fleet.rancher.io

[4] https://longhorn.io/

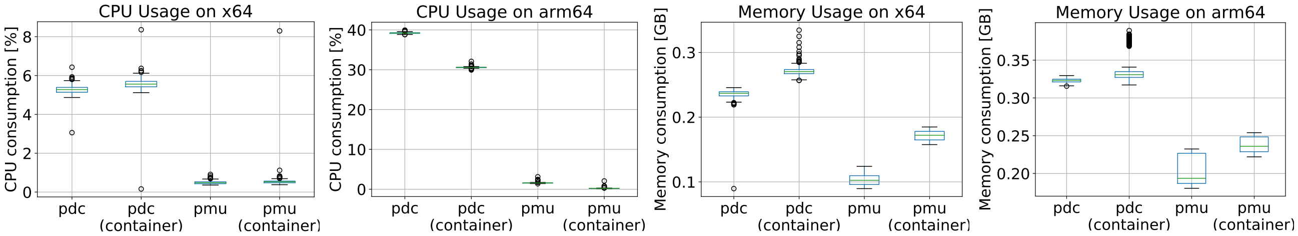

Fig. 2: CPU and memory usage comparison for service execution on bare metal and with containerization.

TABLE II: Relevant specifications of the machine used to carry out the tests.

Architecture	x86 (64-bit)	arm (64-bit)
Machine	VM	Raspberry Pi 4B
linux kernel	5.4.0-48-generic	5.4.0-1042-raspi
CPU model	Intel Xeon (Cascadelake)	Cortex-A72
CPU cores	4	4
CPU frequency	2.2 GHz	1.5 GHz
Memory size	8 GB	4 GB

capabilities coupled with minimal resource requirements e.g., compared to other software such as Rook/Ceph. The Longhorn service can be replicated on multiple nodes within each site, each instance attached to a distinct data volume created on the node itself. Longhorn instances are coordinated to guarantee that any data is replicated on different physical volumes (hence nodes), providing the selected level of redundancy to survive the departure of N data volumes.

A *StorageClass* applied to persistent data allows specifying the number of desired replicas and the time for scheduled backups. To guarantee better performance, we configured the `defaultDataLocality` of the StorageClass to `best-effort`, which forces Longhorn to keep a replica on the same node as the service that uses the volume. Moreover, additional configurations have been used to allow a quick instantiation of services using a persistent volume in case of node failure. Specifically, the option `nodeDownPodDeletionPolicy` has been configured to `delete-both-statefulset-and-deployment-pod` enabling the force deletion of containers on failed nodes, so that the volume can be immediately attached to new instances.

IV. Experimental Evaluation

Monitoring services (i.e., PMUsim and OpenPDC) have been containerized to be executed in a cloudified environment, hence only Linux-based operating systems have been considered for our evaluation. To keep the results consistent, Ubuntu 20.04 has been used as the base OS for all measurements. We considered both x86 and ARM architectures, 64-bits, which will be hereinafter referred to respectively as x64 and arm64. Further information are available in Tab. II.

A. Evaluation method

CPU and memory consumption metrics have been collected using `sysstat`, gathering information using Linux standard counters with a `cron` job executed every minute. CPU usage represents the time in which the CPU is not idle and the system does not have an outstanding disk I/O request. Memory usage simply accounts for the non-free memory at a given time. Applications were executed in their standard operating conditions, e.g., PMU producing data at a constant rate, while the PDC receives, processes and stores the above data flow. Tests have been carried out for at least 4 hours to demonstrate that applications have a consistent behavior over time and to collect enough data to perform significant statistical analysis, identifying confidence intervals and outliers.

Reaction times tests have been carried out using helper `bash` scripts to poll system events and store their timestamp for subsequent analysis. Each single case will be explained more in-depth in the dedicated section.

B. Containerization overhead

This test quantifies the additional resources required by containerization for the same application installed on bare metal. The PDC service has been containerized using an Alpine base image on arm64 and an Ubuntu base image on x64, which resulted in the most efficient CPU and memory usage in their respective environments. However, this experience suggests that the base image to be used when containerizing a service cannot be given for granted and should be properly assessed in a real production environment.

Fig. 2 shows that the CPU overhead added by the container environment is negligible both in case of x64 and arm64 (actually, CPU consumption even improves in case of arm64), whereas the additional memory is in the order of a few megabytes (53MB for x64 and 40MB for arm64). This confirms that containerization overhead is almost negligible – regardless of the device architecture – even for edge devices with limited computational power.

C. Orchestration and distributed storage overhead

This test quantifies the resources required by K3s, i.e., the cost of the orchestration (without any additional workload), differentiating between worker and master nodes. Next, it analyzes the cost of running also a redundant storage service.

Fig. 3 shows the resource requirements of the different setups. As expected, the master node results in a higher CPU and memory footprint with respect to worker nodes, because of the additional requirements of the control plane components; the use of Longhorn further increases the footprint in master nodes on arm64 devices (5% of CPU and 30% of memory).

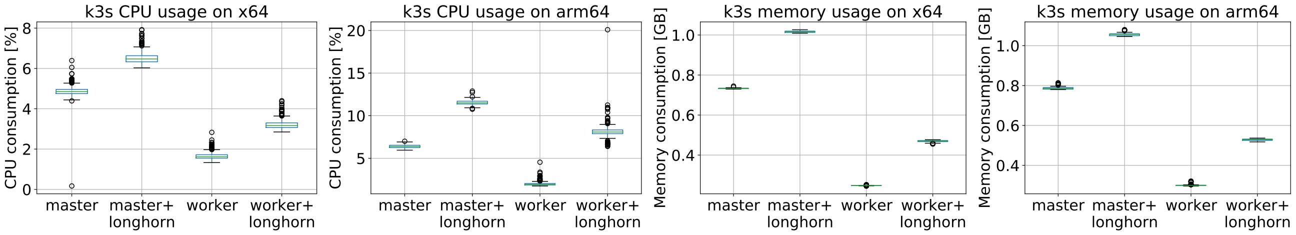

Fig. 3: Orchestrator CPU and memory requirements on K3s master and worker nodes, with and without Longhorn.

Although the orchestration overhead is no longer negligible as for containerization, K3s and Longhorn can provide a good balance between resource usage – still tolerable also for a low-end device such as a Raspberry Pi – and the advantages brought in by an orchestrated system, which provides enhanced data and service resiliency.

D. Orchestrator reaction times

In Kubernetes the reaction time upon node/service failures can be dramatically reduced using *replicas*, i.e., N instances of the same service are executed simultaneously, guaranteeing service resiliency in the event of $n < N$ failures. Although appealing, replicas cannot provide benefits in our case. In fact, multiple PMUs instances would share the same physical measurement device, hence resulting in hardware contention. PDCs are not suitable either, as the default Kubernetes load balancing mechanism sends data to either one of the replicas. This results in data partitioned across all the PDC instances and the consequent necessity of data re-aggregation before further processing, which would not satisfy our requirement of leveraging existing applications. Therefore, only single replica services are evaluated.

Our test evaluates the orchestrator reaction time upon the occurrence of two possible failure events: (*i*) container restart after unexpected execution failure (simulated by forcibly sending a *kill* signal within the container, hence killing the process delegated to the synchrophasor exchange and triggering the re-scheduling policy of the orchestrator); and (*ii*) container re-instantiation in a healthy node after a node either fails or becomes unreachable.

We used the *tcpdump* command to measure the time required by PMU and PDC to actively restore the synchrophasor exchange process, monitoring then the actual network packet exchange between the components. Results are then compared to the widespread `nginx` web server, which provides insights on the maximum performance of a cloud-native application. The entire process is repeated 10 times to obtain statistically relevant data.

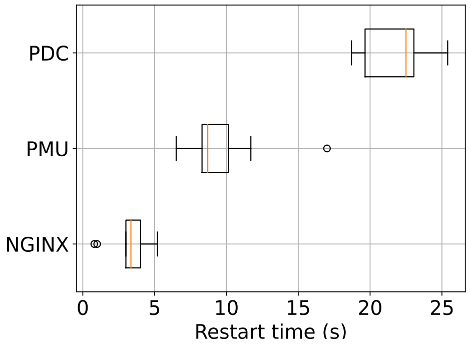

Fig. 4: Data flow restart time interval in case of nginx, PMUs and PDCs.

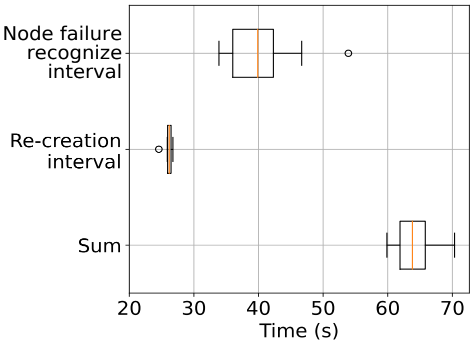

Fig. 5: Time required to recover services on a disconnected node.

Results, depicted in Fig. 4, show that Nginx, designed to be fully cloud-native, experiences the fastest restart time, rarely exceeding $5s$. Instead, PMU and PDC are not designed to operate on a Cloud environment and the measured restart time ranges between $6 - 12s$ for the PMU, and between $18 - 25s$ for the PDC. The proposed results raise some additional considerations: (*i*) In our configuration, the Kubernetes control plane checks every $5s$ the state of the specific service (e.g., healthy, unhealthy). Therefore, the contribution of the orchestrator on the final restart time cannot exceed $5s$ in the worst case and can be further reduced upon configuration. The remaining time is thus related to the service control logic to re-instantiate the communication and can be reduced only with proper code refactoring. (*ii*) As of today, the recovery upon monitoring service failure is not automated, and in many cases still requires manual intervention. This implies that monitoring services have different resiliency requirements, compared to control services, and can withstand longer service disruption (e.g., minutes), without compromising the smart-grid operability.

The node failure is emulated by pushing a set of firewall rules in `iptables` to isolate the target node from the rest of the infrastructure. The isolation process is repeated multiple times, in order to cover any possible failure, as the application placement within the infrastructure is (almost) completely delegated to the orchestrator and may vary over time. The K3s master is continuously polled to check when the target node is marked as unreachable and measure the time required to re-instantiate the containers running on it. Fig. 5 shows the three critical reaction intervals: (*i*) time required for the master to recognize a failed node and set its status as NotReady/Unreachable, (*ii*) time to re-create all the containers hosted in the failed node and to restore the application data flow, and (*iii*) the total time to have the service restored on the remaining running nodes. Specifically, the total re-creation interval is bounded to the slowest-restarting service (i.e., usually the PDC), and the time interval required to identify a node failure strictly depends on Kubernetes control logic and experience high variability, depending on the moment of the failure and the next node health check. Still, even in the worst case, the proposed infrastructure can recover to node failure event within $70s$, well below the requirements.

V. Conclusions

This paper proposes a novel, scalable architecture to deliver critical IT services to the geographically distributed power grid infrastructure. This requires the definition of a Kubernetes cluster, with autonomous orchestration capabilities, on each site, due to the inherent resiliency provided by local master nodes. The deployment of additional services in the telco Point of Presence helps to drastically reduce the communication latency between services in the distributed infrastructure, as well as limiting the impact of possible IT network outages, hence, improving monitoring procedures. The management of geographically distributed clusters can be carried out by Rancher Fleet, avoiding a per-site definition that may be unfeasible given the number of sites involved. Storage resiliency has been taken into account and considered a critical part of the architecture. The evaluation focused on the overhead brought by lightweight virtualization and especially by the orchestrator, as well asthe time required by Kubernetes to react to simulated faults and the consequent application restore time. The results brought up relevant information necessary to design edge clusters both in hardware resources and the number of nodes. The analysis of reaction times gives a general view of the behavior of Kubernetes and highlights where the tuning should be carried out to improve the reactions in case of faults.

Acknowledgments

Authors would like to thank Claudio Lorina, Claudio Usai and Sebastiano La Terra for their precious help in modelling and validating the proposed architecture, and Antonio Manzalini for his comments. Stefano Galantino acknowledges the support from TIM S.p.A. through the PhD scholarship.

This work has been financed by the Research Fund for the Italian Electrical System under the Contract Agreement between RSE S.p.A. and the Ministry of Economic Development - General Directorate for the Electricity Market, Renewable Energy and Energy Efficiency, Nuclear Energy in compliance with the Decree of April 16th, 2018.

References

[1] M. A. e. a. Lopes JAP, "Wires energy and environment," in *The future of power systems: Challenges, trends, and upcoming paradigms*, vol. 9, no. 3, 2020.

[2] F. Ahmad, A. Rasool, E. Ozsoy, R. Sekar, A. Sabanovic, and M. Elitaş, "Distribution system state estimation-a step towards smart grid," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 2659–2671, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032117310134

[3] D. Falcao, F. Wu, and L. Murphy, "Parallel and distributed state estimation," *IEEE Transactions on Power Systems*, vol. 10, no. 2, pp. 724–730, 1995.

[4] A. von Meier, D. Culler, A. McEachern, and R. Arghandeh, "Micro-synchrophasors for distribution systems," in *ISGT 2014*, 2014, pp. 1–5.

[5] J. De La Ree, V. Centeno, J. S. Thorp, and A. G. Phadke, "Synchronized phasor measurement applications in power systems," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 20–27, 2010.

[6] A. Bose, "Smart transmission grid applications and their supporting infrastructure," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 11–19, 2010.

[7] U. D. of Energy (DOE), "Advancement of synchrophasor technology in projects funded by the american recovery and reinvestment act of 2009," https://www.smartgrid.gov/document/Synchrophasor_Report_201603.html#, 2016.

[8] M. Pignati, L. Zanni, P. Romano, R. Cherkaoui, and M. Paolone, "Fault detection and faulted line identification in active distribution networks using synchrophasors-based real-time state estimation," *IEEE Transactions on Power Delivery*, vol. 32, no. 1, pp. 381–392, 2017.

[9] R. Puddu, K. Brady, C. Muscas, P. A. Pegoraro, and A. Von Meier, "Pmu-based technique for the estimation of line parameters in three-phase electric distribution grids," in *2018 IEEE 9th International Workshop on Applied Measurements for Power Systems (AMPS)*, 2018, pp. 1–5.

[10] M. P. Kwaye and R. Lazzari, "Topology identification in distribution network based on phasor measurement units," in *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, 2021, pp. 1–6.

[11] "Ieee guide for synchronization, calibration, testing, and installation of phasor measurement units (pmus) for power system protection and control," *IEEE Std C37.242-2021 (Revision of IEEE Std C37.242-2013)*, pp. 1–98, 2021.

[12] F. Luo, Z. Y. Dong, Y. Chen, Y. Xu, K. Meng, and K. P. Wong, "Hybrid cloud computing platform: The next generation it backbone for smart grid," in *2012 IEEE Power and Energy Society General Meeting*, 2012, pp. 1–7.

[13] F. Luo, J. Zhao, Z. Y. Dong, Y. Chen, Y. Xu, X. Zhang, and K. P. Wong, "Cloud-based information infrastructure for next-generation power grid: Conception, architecture, and applications," *IEEE Transactions on Smart Grid*, vol. 7, no. 4, pp. 1896–1912, 2016.

[14] E-distribuzione, "Piano di sviluppo 2020-2022," https://www.e-distribuzione.it/content/dam/e-distribuzione/documenti/e-distribuzione/Piano_di_Sviluppo_2020_22_30giu2020.pdf, 2020.

[15] P. Kansal and A. Bose, "Bandwidth and latency requirements for smart transmission grid applications," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1344–1352, 2012.

[16] A. Zaballos, A. Vallejo, and J. M. Selga, "Heterogeneous communication architecture for the smart grid," *IEEE network*, vol. 25, no. 5, pp. 30–37, 2011.

[17] M. Iorio, F. Risso, and C. Casetti, "When latency matters: Measurements and lessons learned," *SIGCOMM Comput. Commun. Rev.*, vol. 51, no. 4, p. 2–13, dec 2021.

[18] S. Böhm and G. Wirtz, "Profiling lightweight container platforms: Microk8s and k3s in comparison to kubernetes." in *ZEUS*, 2021, pp. 65–73.

[19] B. Gou, "Generalized integer linear programming formulation for optimal pmu placement," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1099–1104, 2008.