Master thesis : Decentralization of control operations carried out on the decentralized production units connected to the ORES network

Auteur : Escalona Coronel, Saul Jose
Promoteur(s) : Ernst, Damien; 12789
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en informatique, à finalité spécialisée en "management"
Année académique : 2021-2022
URI/URL : http://hdl.handle.net/2268.2/16299

University of Liège - Faculty of Applied Sciences
in collaboration with ORES

Decentralization of monitoring operations carried out on the decentralized production units connected to the ORES network

Supervisor:
Prof. D. Ernst

Jury:
Prof. B. Cornélusse
D. Vangulick
Ph.D. E. Karangelos

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science & Engineering
by Saul Jose Escalona Coronel

Academic year 2021-2022

Abstract

Grid monitoring is an important task for operators of the distribution grid, as it allows them to identify and diagnose problems in the system. This can save energy and money, as well as improve system reliability.

There are a number of different methods that grid operators can use to monitor the system. One popular approach is to use data collected by sensors located on the distribution grid. However, the data from these sensors can be rather expensive to collect and store; therefore they may not be suitable for smaller network operators. Some operators also use electrical measurements taken directly from the grid's conductors, but these are also not practical for smaller networks.

One potential use for blockchain in grid monitoring is to store data about the production of electricity on the distribution grid. This data could be used to improve system reliability and optimize energy use. We developed a proof of concept software called "MonitORES" that could utilize this technology to achieve these goals.

The proof of concept developed was based on Hyperledger Fabric and showed that monitoring and control operations can be achieved by blockchain. Analyzing the probabilities of successful attacks on the current system and on the proposed architecture, the results showed how a distributed system reduces the probabilities of such attacks.

Acknowledgements

This thesis is dedicated to my family, who have always been supportive of my pursuits in academics.

Special thanks to David Vangulick, my company supervisor, for the time and patience he dedicated to coach me and follow me through this adventure.

I would also like to thank my jury members and my academic supervisor Damien Ernst, who offered their time to evaluate this thesis.

I am grateful to all the friends and people I met during my studies.

Finally, I would like to thank my girlfriend and his family for their unwavering support and love, which have helped me maintain a positive attitude throughout this long journey.

Contents

Acronyms

ANM Active Network Management. 8, 9, 11, 13, 18, 24, 37, 39, 49, 50, 52, 57, 72

API Application Programming Interface. 4, 52, 53, 57, 59–65, 67

DER Distributed Energy Resource. 3, 8–11, 21, 37, 39, 47

DLT Distributed Ledger Technology. 3, 9, 10, 12, 13, 26, 39, 40, 47, 48, 72

DSO Distribution System Operator. 3, 8–11, 13–21, 24–26, 38, 39, 47–51, 54–56, 64, 67, 72

FDIA False Data Injection Attack. 9

GUID Globally Unique Identifier. 66

HMI Human to Machine Interface. 18

HV High Voltage. 15, 23

IBFT Istanbul Byzantine Fault Tolerance. 33, 34, 41, 44

LV Low Voltage. 16, 37

MV Medium Voltage. 8, 10, 15, 16, 19, 22, 23, 37, 39

OOSDN Optimal Operation for Smart Distribution Networks. 18, 24–26

PBFT Practical Byzantine Fault Tolerance. 32–35, 44

PGU Production Generation Unit. 4, 25, 48–51, 53, 54, 57, 58, 60, 61, 63–67, 71, 74

PLC Programmable Logic Controllers. 18, 24–26, 38

RTU Remote Terminal Unit. 17, 18, 24–26, 38, 39

SCADA Supervisory Control And Data Acquisition. 11, 17–19, 24, 25, 38

SONET Synchronous Optical NETwork. 18

TSO Transmission System Operator. 8, 10, 15, 18, 25, 47

VPN Virtual Private Network. 18, 38

WAN Wide Area Network. 18

Chapter 1

Introduction

Throughout the human existence, energy management has been a time-consuming puzzle. As population grew, we developed methods to enable an open access energy to everyone. Each industrial revolution came with its paradigm of managing energy. From steam machines to the first electricity grids, maintaining and ensuring such infrastructure was a key element to unleash the full potential of these technologies.

Driven by the absolute need to reduce CO_2 emissions and to keep global warming at a descent level, the energy sector is facing one of its greatest transitions in history. According to the World Resources Institute, in 2018, energy related emissions represent 71% of the total belgian human made CO_2 emissions. In particular, the electricity and heat sector are responsible for 25% of the mentioned percentage [1]. This makes it the second main source of CO_2 emissions in the energy sector, transportation being first. To reduce those emissions and respect European objectives, Belgium has committed ambitious goals in term of renewable energy production and energy consumption efficiency [2] [3] [4].

By consequent, we observe an increasing amount of new solutions such as decentralized generation, electricity usage for mobility and heating, and electricity grid balancing. From 2018 to 2020, renewable energies installed capacity experienced a growth of 39% [5]. Photovoltaic panels and windmills capacity is expected to double by 2023. However, managing such sources of energy which are intermittent by nature is not an easy task. Additionally, the belgian government plans to stop nuclear energy production by 2025 [1], which represented about 50% of the current electricity generation in 2021 according to FEBEG([5]). Power systems need to progressively acquire an increase in flexibility and dynamism to face this evolution.

[1] postponed to 2035, `https://www.letemps.ch/monde/belgique-repousse-dix-ans-sortie-nucleaire`

Nowadays, electricity flows through a complex network shared between multiple stakeholders. By consequent, changing the behaviour of flows in the system, has an impact on the management of its infrastructure. Three main stakeholders will play an essential role in this evolution. Transmission System Operator (TSO), Distribution System Operator (DSO), and Distributed Energy Resource (DER). TSOs operates the transmission of electricity across long distances. They manage the extreme-high to high voltage lines and the high to medium voltage transformers. Connected to them, we find DSOs. They are in charge of the distribution system, which is composed of medium to low voltage networks. Customers are directly attached to their networks. Those customers can be electricity producers called DERs whose power capacity is under 25 MW. Above this value, producers are directly linked to the transmission system. DERs are key to the energy transition, as this is where the increasing in renewable capacity happens. Understanding how we can adapt the whole network to the mentioned changes is a collateral challenge that comes with all these innovations.

In order to maintain stability across the network and face the risk of congestion in the short to medium term, old practices of investing in increasing network capacity are no longer effective and new actions have to be taken by the system operators (i.e. TSOs and DSOs). Congestion is a state of the network where the power flowing through the grid is close or at the limit of what it can handles from a security perspective, impacting decisions on how generation and loads should continue to work. For example, a Medium Voltage (MV) network depends on two transformers interfacing with the transmission system, and could be in congestion if the production of this network exceeds the capacity of the transformers. This can happen when one transformer fails or is in maintenance, as well if generators producing simultaneously at full capacity inject power above transformers limits. Not managing congestion could lead to power outage in the corresponding network but also in other parts of the grid as it will disturb the balance of the system. In this work, our focus is set to congestion due to an increasing generation capacity with respect to current consumption.

To overcome this situation, DSOs are currently implementing Active Network Management (ANM) strategies and methods. ANM at DSO level relies on two main pillars: controlling of the end devices (consumption or production) and monitoring the system.

Regarding the control, ANM optimizes the power flow on the network and face possible congestion providing operators with actions that reduce risk of damaging assets while minimizing operational cost and maximizing energy generation [6]. Those

operations are carried in a real-time or near-real-time framework. One common action is to dynamically curtail energy production when a congestion situation is foreseen. DSO establishes with producers flexibility and permanent power intervals which gives him control possibilities. However, choosing the optimal limit in congestion situation is not trivial. In fact, the curtailed generator might be a renewable source of energy where all potential energy should ideally be harvested. Finding this optimal value is an optimization problem combined with statistical methods to predict right amount of curtailed power [6] [7]. Curtailment strategies will be the only ones taking into account in this thesis.

Concerning the second mentioned pillar of ANM, monitoring means gathering reliable amounts of data about the system in order to perform well. This implies also reliable interactions with different endpoints of the system with high level of automation. As mentioned, limits imposed by those strategies are to be followed and verified as the risk of not execution leads to unstable states. Power systems are being upgraded with communication and measurement systems to cope with this observability and control problem. Cyber-physical devices enables network components to remotely interact with control centers through public and private networks. For example, some substations in distribution network, possess the capabilities of remotely being controlled and transmit measurements to the control center. By consequent, computer systems involved in the process, interact between each other and are constantly monitoring the distribution system, triggering actions on equipment exposed to risk [8]. In the last decade, power systems have faced wide area outages due to attackers, such as the Ukranian blackouts in 2015 [9]. This has highlighted the importance of ensuring data integrity in remote cyber-physical equipment. For example, attacks such as False Data Injection Attack (FDIA) are possible [10] [11] [12] [13]. FDIAs aims to manipulate input data in the system, in order to lead the control center to malicious actions without being detected.

Automate and ensure contractual data monitoring between DSOs and DERs is then a key task that must be accomplished reliably. This is where innovative technologies such as distributed ledger technologies can hop in and be useful tools to cope to the use case. They have proven to reinforce integrity and reliability among other security properties in distributed environments such the financial sector. For instance, blockchain is the most popular DLT being discussed nowadays. Along with encryption, hashing and consensus mechanisms, it provides a distributed temper-proof database. At first, it was presented as a decentralized way to exchange value with a first application called Bitcoin [14]. However, the capabilities to self execute code when specific conditions are met (called smart contracts) provide a new dimension of actions by enabling automation to manage

its distributed state [15] [16].

Exploring the use of blockchain to store and control contractual data exchange between the DSO and the customers seems naturally interesting. A blockchain-based approach to improve power systems security was already tested in [17]. In [18], blockchain is also used to adapt the regulation and optimization of the systems in a distributed way. Further discussion on related work will be exposed in a later section.

This thesis focus primarily on evaluating the use of a DLT to reinforce data integrity and automation at the cyber-physical edge devices of the ORES distribution system during monitoring operations of decentralized energy resources exposed to congestion risk. Particularly, we carry out an assessment to choose which particular DLT platform suits better the presented use case, and then implement a prototype in the chosen platform. Finally, in this thesis we study and analyse how such technologies hedge better against cyber-attacks than centralized current models. Section 1.1 presents more about the motivation of this work based on the current situation at the ORES network. In section 1.2, we describe the state-of-the-art in DLTs that will be approached in this thesis. Then, Section 3.2 presents explicitly the questions to be answered by this work. Finally, the structure of the thesis is shown in Section 1.4.

1.1 ORES

ORES is the main DSO in the walloon region, managing up to 75% of the walloon municipalities electricity and gas grid [19]. They take care of the distribution infrastructure supporting all medium to low voltage electricity exchanges as well as natural gas operations and public light infrastructure. In the electricity ecosystems they are at the edge of the system, being physically the closest to customers. In contrast, TSOs such as Elia, are in charge of extra-high and high voltage and take care of the transmission of energy across long distances representing the backbone of the power system.

With a turnover of 1.153 M€ in 2020, ORES is responsible for 51.765 km of electricity distribution network and 10.033 km of gas distribution network [19]. Their medium voltage network is experiencing an increase in the numbers of decentralized energy resources (DER). As **Figure** 1.1 shows, ORES expects to double the amount of power injected in its MV network from 2021 to 2024. Moreover, we observe a particular increase in the power range from 1 MVa to 5 MVa and 10 kVa to 250 kVa, which can be translate

to an increase in the stress put on those networks.

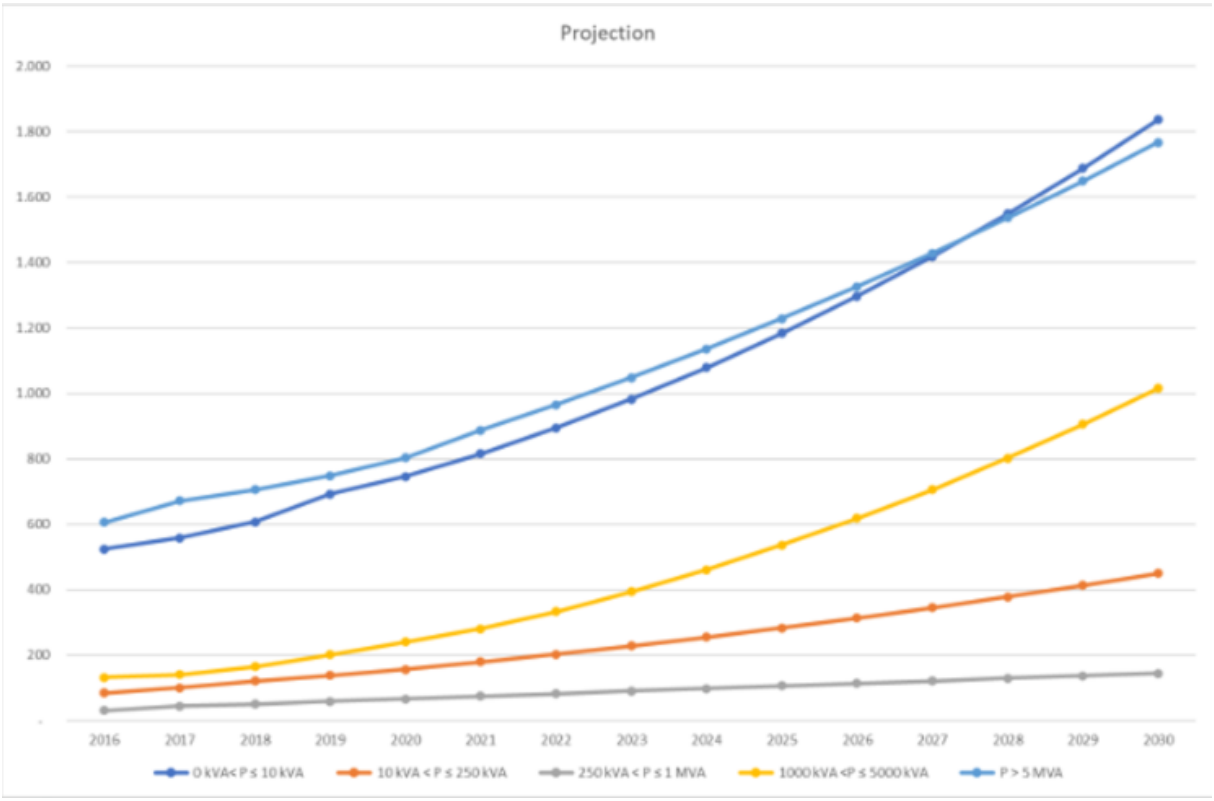


Figure 1.1: Forecasted evolution of DERs attached to ORES network according to government goals

In order to manage the previously mentioned networks, the touch stone of operations is the Supervisory Control And Data Acquisition (SCADA) and related system. SCADA systems relies on devices across the network which measure the system's state and where reliability is not trivially ensured. All information, internal and external, converge at this central point, where they allow risk computation to be achieved in order to generate control actions. This is where ANM takes places, generating and computing all optimizations for the electricity grid. ORES is currently working on the implementation of an ANM service called O-ONE.

When operating the system with such strategies, a legal contractual framework is needed to establish how the DSO can limit a certain generator. Those contracts contain what are the minimum levels of power the DSO is able to receive and the range on which the generator would be constraint. Moreover, once a contract is initiated, DSOs must control the respect of such constraints as disobedience could lead to damage in infrastructure. For instance, monitoring and control actions could be verifying that measures are effectively submitted at specified frequency or checking if the production is

under curtailment limits in the flexibility range.

1.2 DLTs: Decentralized vs Centralized

By definition, a Distributed Ledger Technology (DLT) represents a technological architecture and protocols to validate and manage immutable records of replicated, shared, and synchronized digital data geographically spread across multiple sites, countries, or institutions. Unlike with a distributed database, there is no central administrator.

Blockchain is one well-known DLT, where the exchanges of information called transactions from the ledger are stored in timestamped blocks that are chained with encryption technique. It provides immutability and availability among other properties that will later be discussed.

Nowadays, such systems can be classified into two main groups: permissionless and permissioned. The first group correspond to all DLT systems which operates in totally trust-free environment, where anyone is able to join the distributed system. On the other hand, permissioned systems, stakeholders taking part in it must be identifiable and authorized to interact with the system. Many examples of permissionless systems reside in the cryptocurrency world, such as Bitcoin [14], Ethereum [15] or Cardano [20]. Permissioned systems are more often used in industrial and private environments, such as the supply chain management Hyperledger blockchain created to improve Walmart's fruits traceability [21].

These new technologies are currently experiencing lot of hype, being declared as the solution to all kind of problems. Numerous blockchain frameworks have been released in the past 5 years. Each one proposing its specific features and structure. Choosing the type and framework that suits the most our use case is not trivial.

1.3 Research Question

Now that the basic context of the ORES congestion control and monitoring has been introduced, along with a brief presentation of DLTs. The main question this thesis aims to answer is the following:

On behalf of the DSO, can a DLT monitor the contractual requirements of the generation units within the framework of an ANM scheme and within sufficient resilience to cyber-attacks?

1.4 Thesis Outline

This thesis is structured as follows:

- **Chapter 2: Background**, this chapter introduces all basic knowledge needed to understand how the power distribution systems work as well as DLTs mechanisms.
- **Chapter 3: Problem Statement**, this chapter states the assumptions and the context needed to understand the problem and its decomposition.
- **Chapter 4: Use Case Platform Assessment**, in this chapter the assessment of different DLT platforms is presented for the case of ORES grid.
- **Chapter 5: MonitORES proof of concept**, this chapter describes the software architecture of the demonstrator and the analysis of the results.
- **Chapter 6: Conclusion**, this chapter resumes the main contribution of our work and provides an outlook for further improvements.

Chapter 2

Background

This chapter introduces the theoretical context of this thesis. First, the distribution network is explained as well as the ORES topology along with its monitoring complexity. Then the chapter focus on explaining what distributed ledger technologies are and the different types existing.

2.1 Distribution Systems

In this section, let us focus on the concept of distribution systems and how they are modeled. An introduction to the ORES grid will allow to contextualise the environment in which the project will take place.

A distribution system, in the electrical context, is a network of medium to low voltage that finalizes the delivery of electricity in the power grid. It can be seen as the final stage where the electricity is delivered to the user. The distribution grid comes right after the transmission grid which is in charge of the transport of energy across long distances. It is mainly composed of substations, transmission lines and a variety of devices to manage the energy flowing through the system (e.g. switches, relays, etc.). Generation and consumption devices are attached to this network. These systems are managed by Distribution Systems Operators (DSO). In Belgium, the main DSOs are ORES and RESA in Wallonia, Fluvius in Flanders and Sibelga in Brussels.

This latter part of the grid is sensitive to electricity consumers or producers as it is directly exposed to them. Therefore, DSOs must constantly monitor all the resources in the network.

In order to accurately supervise the distribution network, its state needs to be modeled and tracked. In the following sections, first a description of the network

architecture is given, then the chosen state model is presented.

2.1.1 DSOs & ORES Architecture

A DSO's architecture is here presented in two parts: the physical-electrical infrastructure and the communication architecture. The first part consists on all electrical material that effectively interacts with electricity. For instance, all cables, substations, transformers and other electrical devices belong to this category. On the other hand, the communication architecture is composed of all information systems currently monitoring the system, allowing the DSO to observe and interact electronically with the physical system. Splitting into two mutually connected architectures will ease the analysis and statement of the problem in this work. Indeed, it will expose the security risks in current power information systems while allowing an understanding of what is happening physically.

Let us now dive into each one of the architectures while developing the links between both worlds. In order to illustrate both architectures, MV ORES network is taken as a reference architecture. In fact, other DSOs architectures may not be fundamentally different. Moreover, the developed prototype is based on MV ORES network, allowing the reader to understand later discussions linked to ORES choices.

ORES electrical architecture

From a high point of view, ORES electric network is mainly composed of three elements: Primary substations and PODEs ("*Poste Déporté*"), secondary substations and network lines.

Primary substations and PODEs are in charge of the power connection to the transmission system. Primary substations contains the HV/MV transformers. Limits on ownership of all the primary substations materials vary from a DSO to another, but in ORES case, most of the equipment and building of these substations are owned by the TSO (i.e. Elia). This limit of ownership determines who is responsible and accountable for the substation assets. When the power injection needs to be projected from the transmission system to a specific area where there is no primary substation, ORES uses a PODE (*"POste DEporté"*). PODEs allow the DSO to directly project the HV/MV connection elsewhere.

Then for each primary substation there is a smaller network of secondary substations

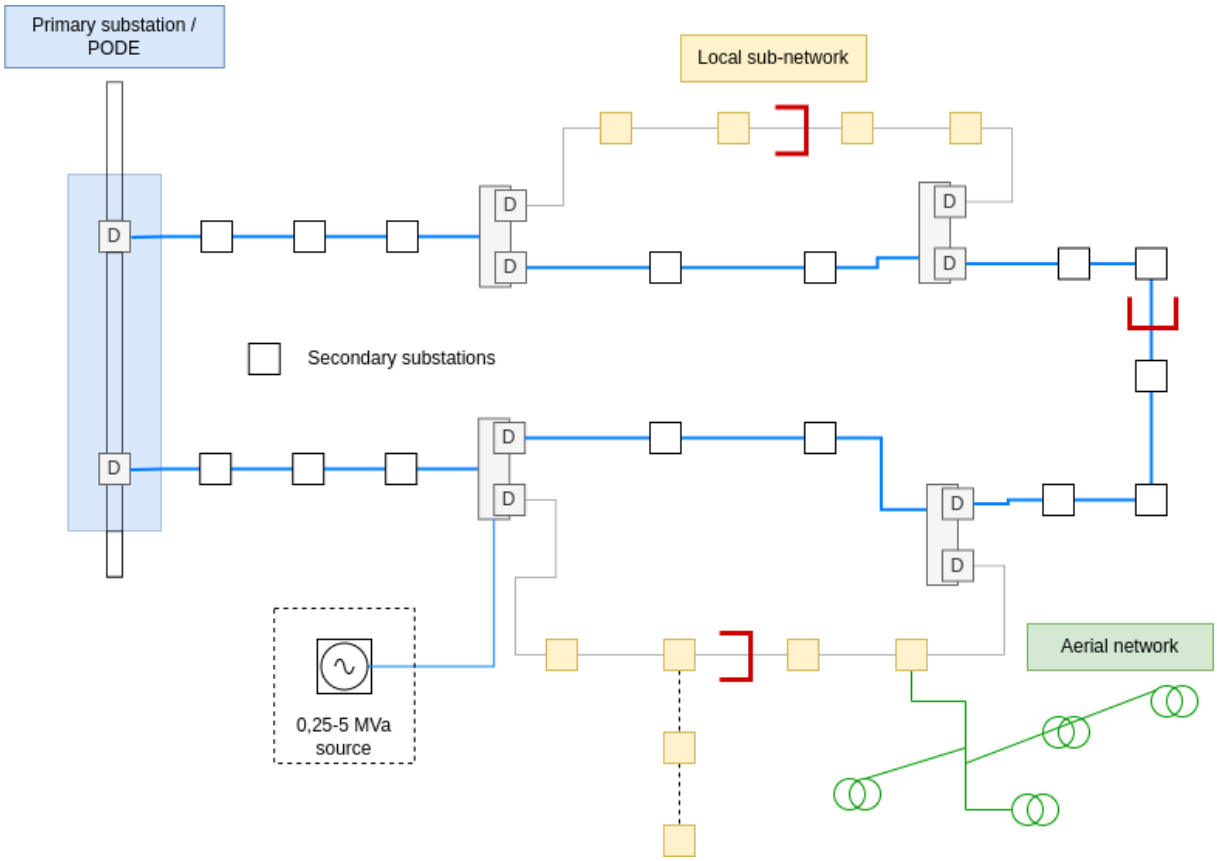


Figure 2.1: MV electric network for DSO

(white squares). Those substations are the MV/LV interface for residential areas or particular loads or generation units. Generation units are attached to this network according to their production capacities. Those rules are specific to ORES but are similar to other DSOs connection policy. ORES counts more than 22000 of them and 20% of these are remote controlled, thus all other substations need to be manually operated. It is possible to find MV sub-networks after a secondary substation as shown in **Fig. 2.1** (yellow squares).

Other possibilities such as directly connected aerial MV networks (green area) for rural areas, exists. However, as they are set to disappear and are not pertinent to this thesis, they are not considered.

Finally, all these elements are interconnected mainly by underground lines and buses which have security constraint, e.g. thermal constrains on cables.

As MV network is the focus of this thesis, there is no further details on LV networks.

Communication Architecture

In order to operate efficiently the system, power systems experience a growth in electronic and digital communication and control systems. In a DSO, the SCADA system is used for centralising and operating all the network information. In this section, **Fig.** 2.2 shows current information architecture in ORES premises. Components can be inside the SCADA center or directly in the physical distribution network. Let us describe each main component shown in **Fig.** 2.2. On this figure, the mapping between the different equipment and the functionalities is illustrated with coloured tags.

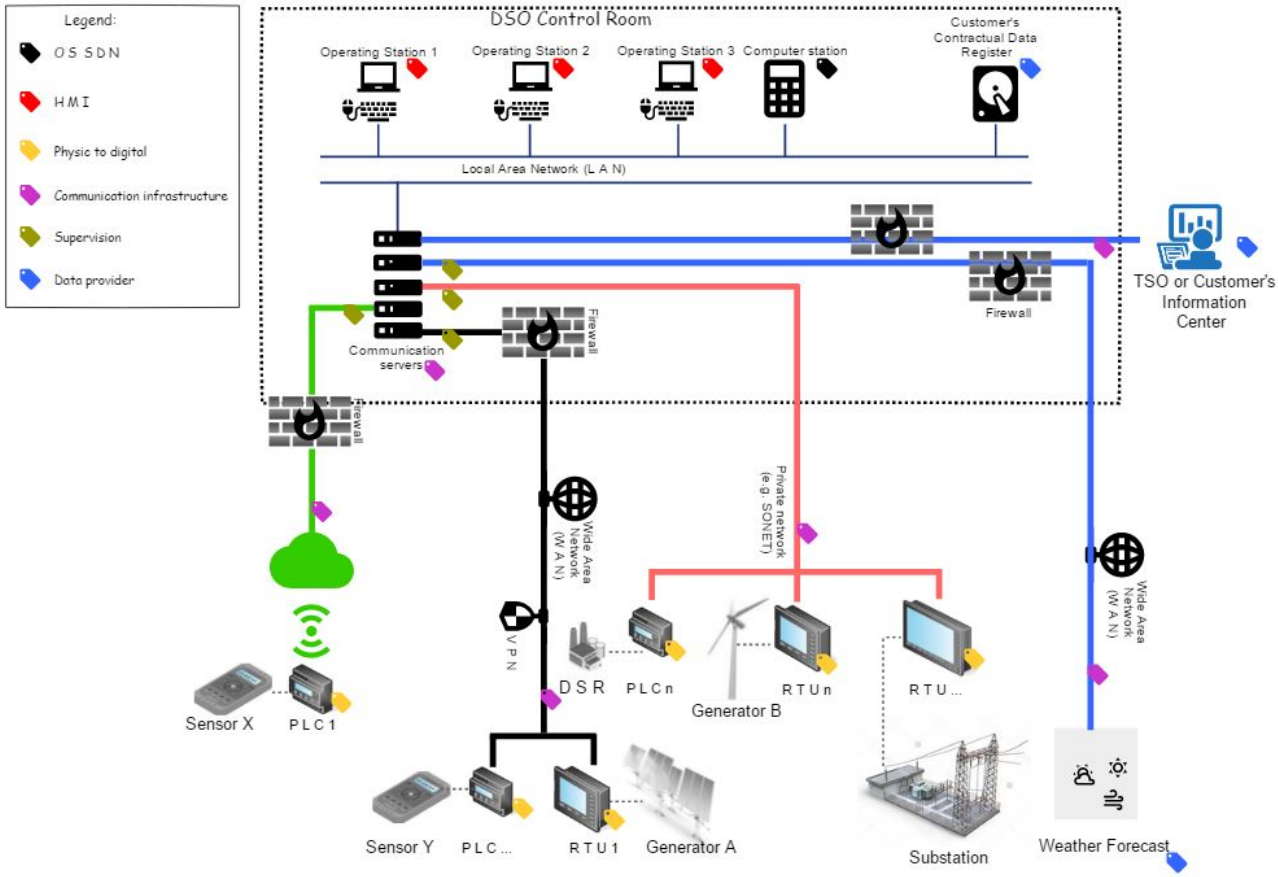


Figure 2.2: SCADA system architecture for DSO

1. **Sensors and local system interface (yellow tag)**: Physical objects are interfaced with SCADA using electronic devices controlled by a microprocessor called Remote Terminal Unit (RTU). These units are used to transmit telemetry data and to receive messages including control signals from the master system to control the objects thus connected. Based on the data received at the RTU level, this device could also autonomously instigate certain control operations. If we only need to collect sensor output signals, convert them to digital data and transmit

them, Programmable Logic Controllers (PLC) are used instead of RTUs because of their higher flexibility, configuration, versatility, and affordability.

2. **Data provider (blue tag)**: We can distinguish three types of data provider. First, the DSO internal data provider, such as customer contractual information, e.g. the maximum installed capacity. Secondly, the DSO external data provider, where information comes from the customer or the TSO information centre that could also be a SCADA system. Finally, the last data provider type is the external web-based information source, like weather forecast, that can be used in order to compute state estimation.

3. **Optimal Operation for Smart Distribution Networks (OOSDN) computer (black tag)**: This is the very hearth of the monitoring system. Firstly, the static information of the grid as the complex impedance of the edges and the mapping functions are digitally recorded in the OOSDN data base. Secondly, the respective limits of every grid elements are also saved in the OOSDN data base. Thirdly, it has access to all information provided by *yellow tags* and *blue tags*. With all this information OOSDN computes the state, the different network state evolution (considering no action has been taken), and the risk assessments. If a risk assessment value exceeds one of the risk limits, it computes the optimal action to take. It is in this component where ANM takes place.

4. **Human to Machine Interface (HMI) (red tag)**: This is an input-output device that presents the process data to be monitored to a human operator. It is possible by connecting to the software and databases of the SCADA system. It is also used to provide the network manager with management information (events logbook), including planned maintenance procedures, trends and diagnostic data for a specific sensor or device. These systems generally allow operating staff to graphically view information.

5. **Communication infrastructure (purple tag)**: ensures that data and commands are correctly exchanged between the different components. Generally, for critical data sources, this functionality is achieved through the combination of radio and wired connections. However, in the case of large systems such as electrical networks and for critical control and/or measurement points, a Synchronous Optical NETwork (SONET) is frequently used. For a less-critical data source, the use of Wide Area Network (WAN) combined with a Virtual Private Network (VPN) is possible.

6. **Supervision** (green tag): The supervision system is firstly used to control the communications between the SCADA system equipment such as RTUs, PLCs and sensors, etc., and the HMI software used in the workstations of the control room.

7. **Cyber security items**: All software and hardware material dedicated to the protection of the SCADA system, e.g. Firewalls.

Electrical and information architectures are tightly coupled as the second one aims to provide a high-fidelity virtual representation and control of the first one. However, as the cost of investment and maintenance of information systems is expensive, all the physical elements composing the electrical network are not perfectly measured. This calls for methods to estimate the state in order to make better decisions when controlling and operating the distribution network. This introduces the next section which presents how the state of a network can be modeled and what is the use of it.

2.1.2 Network Modelling & State Estimation

In this section, the goal is to present how the network modeling and the estimation of its states can be achieved in the context of distribution networks. In **Section** 3.2, the details of which model is given.

The idea behind modeling network state is to decide which variables describes best the situation of the physical network at a given point in time. Then by measuring these values or instantiating a specific configuration of this variables, one can analyse the network. For instance, one can optimize the network for a given situation once part of the state is fixed in order to achieve certain goal under constraint.

Let us now define how the state of the network can be define and how it will be considered in this work. Although methods and tools used in this section can apply to all types of electrical distribution networks (e.g., medium-voltage or low-voltage, private or public networks, etc.), a focus is given on the public medium-voltage (MV) distribution network as economic and social consequences are greatest from a DSO point of view.

Let us consider an MV network as a graph $\mathcal{G} = (\mathcal{B}, \mathcal{L})$, where $\mathcal{B}$ is a set of electrical buses and $\mathcal{L}$ (cables or lines) is a set of oriented edges connecting elements of $\mathcal{B}$. The sizes of $\mathcal{B}$ and $\mathcal{L}$ are M and N, respectively. In a distribution network, we can assume that $N \geq M - 1$ [22].

Each edge $n \in \mathcal{L}$ is defined by its complex impedance of this link. Let $Z_n = R_n + j.X_n$ be the impedance of edge n. N_{rid} is a function that takes the edge as input and the pair of connected buses as output.

On each bus $m \in \mathcal{B}$, one or multiple device(s) $d_i \in \mathcal{D}$ is (are) connected. There

are D connected devices on $\mathcal{G}$ network ($\mathcal{D} = d_1, d_2, ..., d_D$). For each d_i , we associate a complex power vector $S_{d_{i,t}} = P_{d_{i,t}} + j.Q_{d_{i,t}}$. The set $\mathcal{D}$ is composed of two subsets, one which is related to generators and the second one to demand. A device is considered as a generator when $P_{d_{i,t}} > 0$, or as a demand when $P_{d_{i,t}} \leq 0$. Connected to $\mathcal{G}$, there are G generators (denoted g_i with $i = 1, ..., G$) and C demands (denoted c_i with $i = 1, ..., C$). All these devices have a maximum power output, $S_{g_i}^{max}$ for the generator and $S_{c_i}^{max}$ for the demand. We define the function J that maps the devices g_i and c_i to their respective buses on which they are connected. A specific node m_{TSO} is the connection point between the network $\mathcal{G}$ and the transport system operator. This node is called the slack bus. We assume that, at this node, a voltage magnitude (V_{TSO}) is fixed and constant. Given the fact that the power consumed by all c_i is not balanced by the power generated by all g_i (or the opposite) and the impedance of edges creates losses, m_{TSO} provides the necessary active and reactive power to ensure the equilibrium of these powers. All these elements ($\mathcal{B}$ (including m_{TSO}), $\mathcal{B}$, and $\mathcal{D}$) have their own security limits in terms of apparent power and voltage. We define a network state vector $s_t \in S$. For all $m \in \mathcal{B}$ at time t:

$$s_t = (V_{1,t}, V_{2,t}, ..., V_{M,t}, \theta_{1,t}, \theta_{2,t}, ..., \theta_{M,t}) \tag{2.1}$$

$V_{1,t}, V_{2,t}, ..., V_{M,t}$ being the voltage magnitudes and $\theta_{1,t}, \theta_{2,t}, ..., \theta_{M,t}$ the angles at each bus. Using load flow equations and with an exact knowledge of V_{TSO}, S_{d_i} and Z_n which are called state variables, it is possible to compute for every bus m at time t :

- $V_{m,t}$: the magnitude of the voltage;
- $\theta_{m,t}$: the voltage angle;
- for m_{TSO}, the active and reactive power to ensure the power's balance on $\mathcal{G}$.

It is also possible to compute the magnitude of the current in every edge $n(\forall n \in L, I_{n,t})$. However, further details in the network state vector is not in the scope of this thesis.

Despite all modelling, in order to accurately analyse the network, DSOs need to correctly measure the whole systems. In practice, it is not possible to gather all variables confidently. As mentioned in previous section, only 1 in 5 substations possess upgraded communication capabilities. This pushes network operators to create pseudo-measures, which are estimation of measure through probabilistic models.

It is a common practice for distribution and transport system operators to divide time into periods Δt and to estimate the network state vector $s_{t+\Delta t}$ for the period $t + \Delta t$. This practice allows the DSO to receive the information of measured items (e.g. current and voltage magnitude) placed in the network and to be able to realize the power-flow

calculation. During this period the DSO has to determine s_t, to compute the state vector for the next period $s_{t+\Delta t}$, to compute the probability of any constraint(s) violation, and, if necessarily, to take actions to avoid this violation. In the context of this thesis, it is considered that the network topology does not change. In other words, neither the function N_{rid} nor the function J changes.

2.1.3 Network Planning

In order to ensure a high quality service and the supply of energy across the country, DSOs must carefully prepare all futures investments and operations to minimize risk. ORES divides planning into three main horizons: Long-term, Operational and Real-time. This work mainly focus on the short term operations, i.e. real time horizon, however let us briefly describe other horizons to understand the global picture of network planning.

Long-term horizon consist mainly of forecasting the evolution of the network and energy distribution paradigm from 1 month to 20 years ahead. Its mainly about preparing for infrastructure investments. It depends on statistical models determining the behaviour of specific regions of Wallonia. It depends on the innovations in the grid (electrical vehicles, DERs, heat pumps ...). Representative days are then modeled to simulate future situations. Sensitive analysis is made on different scenarios.

Operational horizon is about all decisions taken within a time window of one month to one day. This is where the realization of long-term planning is achieved, e.g. if a line needed to be replaced, it is ensured that all conditions are met. If there is a storm forecasted, DSOs might adapt the schedule of the month. In this horizon, ORES develops a traffic light systems which dictates the flows ahead of time as shown in **Fig.** 2.3. To implement the traffic light system, ORES takes into account consumption from representative days, but works further on the generation forecast, fine tuning predictive models with forecasted weather, production schedules and flexibility.

Finally, the real-time or near-to-real-time horizon covers all operations made below 15 minutes ahead of time. Here, the goal is to dynamically ensure the resilience of the systems when short-term decisions need to be taken. For example, if a tree falls and damage a substation, actions are to be taken to isolate the problem and reduce impact on customers. Those actions are automated or manually done. To efficiently manage real-time operations, it is convenient for DSOs to count on flexibility and congestion management on the devices attached. Let us dive deeper into these two concepts.

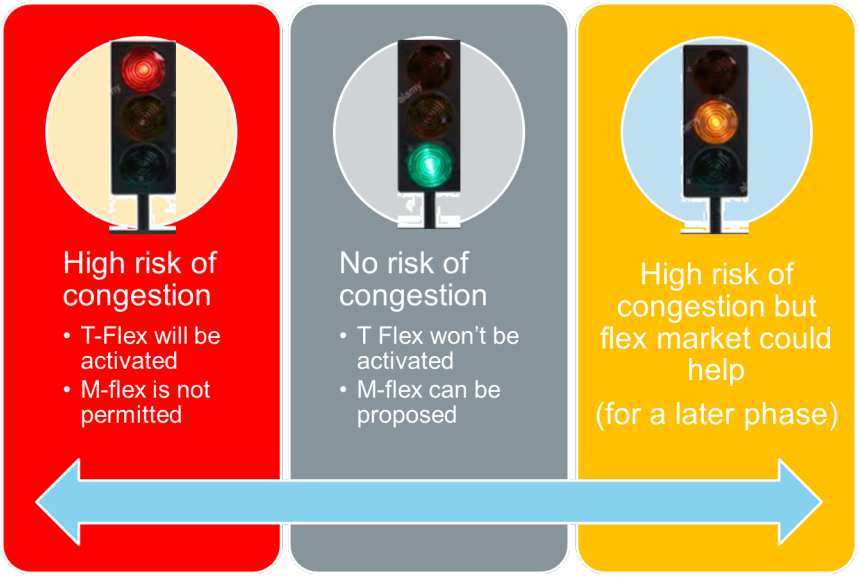


Figure 2.3: Operational planning traffic lights systems at ORES

Congestion and flexibility

As mentioned in the previous section, congestion and flexibility are key points to understand the real time operations.

Electricity grid congestion management is a critical task for ensuring reliable, affordable service to end-users. Grid congestion occurs when the net current exceed security limits of infrastructure such as transformers or cables. The impacts of increased load on the transmission and distribution networks include higher operating cost, increase in consumer prices, reduced reliability, and possible service interruptions. Therefore, effective management of grid congestion is critical to ensuring the continued stability of the electric grid system.

Flexibility is a characteristic that enables a system to adapt or compensate for changes that occur over time. In power systems, flexibility is the ability of the system to respond to the varying demands placed on the system in real time. The ability to respond quickly to changes in demand is essential for maintaining a reliable energy supply while also meeting future demand for increased energy use from commercial and residential customers. Flexible electricity generation and delivery technologies can help to securely manage congestion, e.g. a wind energy producer changing the direction of windmills to reduce production.

To illustrate congestion, **Figure** 2.4 shows an example of MV network depending on two 20 MVA transformers where the total consumption is 11 MVA and the total

installed production capacity is 36 MW. In this case, if one of the transformers fails or if another production source would be added, a risk of congestion could be faced. In fact, if all production sources run at maximum capacity, a total of 25 MW would flow through the transformer. This would cause damage in the transformation system, putting at risk the stability of the MV network. As mentioned before, this situation can also happen if another production source is added. This is a potential brake on renewable and distributed energy growth.

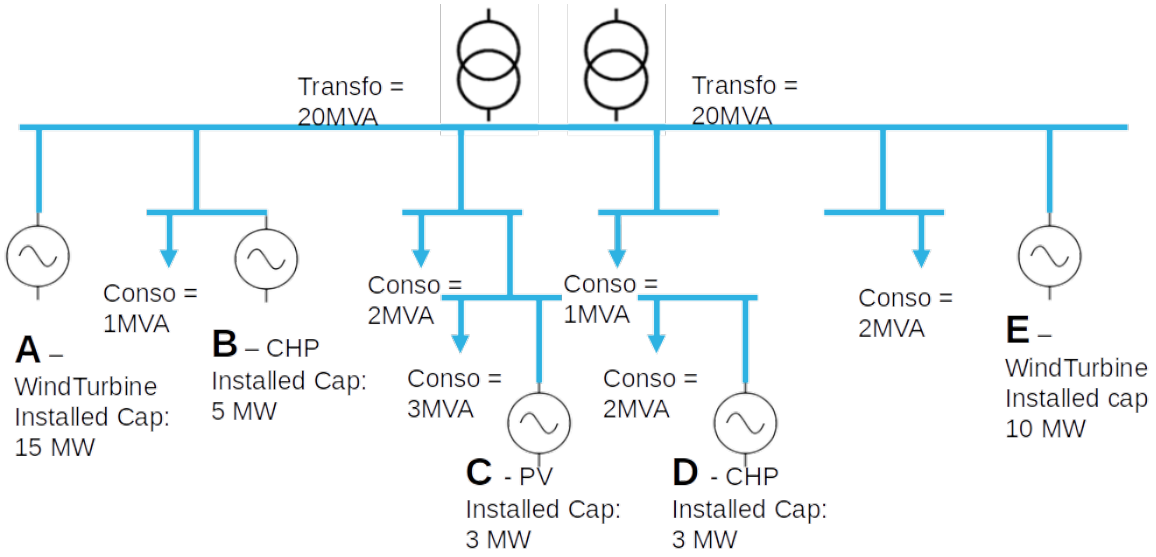


Figure 2.4: Illustration of congestion

In order to establish a legal contractual framework for the actions that ORES can carry when operating the grid, the regulatory entity (CWAPE) and ORES came up with a series of rules where flexibility and priority is explicitly stated. The idea is that each producer gets its potential production divided in two categories, permanent power and flexible power. Permanent power corresponds to the minimal service that could still provide in case of failure in the network, e.g. one HV/MV transformers fails. Flexible power corresponds to the part of energy production that could be asked to curtailed in case of congestion. Non-renewable producers will be curtailed first, and will not be redeem for a loss in potential production, whereas renewable energy producers are prior and will receive a compensation for the loss of potential production. The idea behind this system is to promote and prioritise renewable energies, while ensuring a minimal service scenario where producers are sure they will be allowed to produce.

Active network management and monitoring

To prevent distribution networks from congestion and exploit flexibility DSOs implement currently active network management systems. Active Network Management (ANM) refers to practices that enable the optimization of the power system and its operation to respond to real-time conditions. The most common approaches used to enhance network operations include real-time operation dispatch, reactive power control, voltage support, and curtailment capabilities. Each of these techniques allows the utility to operate its system more efficiently and ensure that it operates at optimal levels to ensure reliable supply of power to the consumers. Additionally, some distribution system operators have adopted the use of advanced monitoring and control systems to help them increase the efficiency of their operations and minimize the amount of peak load on the power grid.

The major components of an active distribution network are sensors, actuators, controllers and communication equipment that are connected through communication networks to provide continuous monitoring of the grid's operating conditions. These sensors monitor parameters such as the line current, voltage and phase angle as well as the weather and operational conditions such as line trips or outages. This information is then relayed to a centralized control system that controls the operation of each device in the system such as the transformer and capacitor banks in order to ensure that they operate within their specified limits. The actuators are capable of adjusting parameters of the transmission and distribution systems in response to changes in the operating environment in order to ensure proper operation of the entire system. These include devices such as circuit breakers, voltage regulators and switchers that can be used to open and close electrical circuits or regulate voltages across the transmission lines in order to maintain system stability. As stated, monitoring is crucial as it what allow the system to adapt itself and verify that all constrained are respected.

At ORES, as mentioned before, the ANM system computes the OOSDN inside the SCADA with information coming form the network. It relies entirely on the measurements of the grid elements. Although, even for a relatively small network (greater than ten nodes), DSOs face the contradictory challenge of trying to find a balance between keeping the operational cost of this data capture as low as possible while determining the power flow as precisely as possible. Regarding these operational costs, we can subdivide them into two principal categories.

First, we have the cost to convert physical information into digital data (RTU or PLC) for which reliability is critical. Second, we establish the cost for the communication infrastructure. The complexity to ensure data security and the cost of these systems

are both high. On that point, the rate of information exchange is very important to ensure that the SCADA is provided with the latest, correct data in order to compute OOSDN. Therefore, these RTUs and PLCs owned by the DSO are relatively expensive, for instance in Belgium, the RESA tariff is $10.109,90$ € or the ORES tariff is $12.532,00$ € [23].

Currently, only producers with an installed capacity above 1 MVA are being monitored with RTUs. All other producers are not being directly monitored and could potentially easily disrespect control orders and do not supply useful information for the optimization computation.

2.1.4 Emergency procedures

Wrong network states can be reached, for instance if the OOSDN computer fails (even if there is certain redundancies), or if a RTU or a PLC stops communicating, or if communication between local devices and SCADA could be interrupted. On the network itself, unforeseen events could also occur, a cable may develop a fault and the loads connected to it could become disconnected.

In order to manage these risks, DSOs instigate two emergency procedures. The first emergency procedure called Alert Mode is used when an unforeseen event occurs between time t and $t + \Delta$ (e.g. disconnection of an important load, internal fault of a transformer used at the interface TSO-DSO, etc). This mode is also used by default when the PGU's RTU detects a broken SCADA link or when the OOSDN is defective (in this case, Alert Mode is declared by an operator). In such situation, it is considered that the DSO has no time to compute a new OOSDN or no means to communicate it. In this situation, the DSO imposes a pre-set maximum generated active power for every generators computed in advance by the DSO and imposed in their connection contracts (and recorded as fixed information type), called *Alert capacity*.

The mapping of these pre-set limits for every generators to the evolution of the state is called $a_e = a^{e1}_{g,m,0}; a^{e1}_{g,m,1} ...; a^{e1}_{g,m,G}$ and corresponds to a value at which the DSO is able to ensure a minimum level of service. In the Alert Mode, there is no further optimization.

The second emergency procedure, called Urgency Mode, is put in place when the following situation occurs:

- between the period $[t; t + \Delta[$, a measure shows at least one edge k is outside its limit

$\overline{I_k}$,

- if the protection mechanism (e.g. protection relay combined with circuit breakers) has not worked yet,
- The exceeding of the limit is due to an excessive energy generation.

In this situation the DSO could try to avoid a massive failure of the system by taking immediate action. The maximum generated active power for generators is put at zero.

To summarize, one can define the control policy (CP_i) for a generating unit i, in terms of maximum generated power as :

$$CP_i = \begin{cases} if(Urgency) \mapsto 0 \\ if(Alert) \mapsto a_{g,i}^{e1} \\ if(r_u > R_u) or (r_d > R_d) \mapsto a_{g,i} \\ else \mapsto a_{g,i} = S_{g,i}^{max} \end{cases} \tag{2.2}$$

Where r and R correspond to the current probability risk and the upper and lower probability risk level, and $a_{g,i}$ the OOSDN computed in case of risk.

These policies need to be implemented and maintained in the local RTU/PLC. It is a challenge to ensure that every PLC or RTU has the last good version of the CP_i.

2.2 Distributed Ledger Technologies

In this section, a theoretical review of DLT is given, first by defining what they are and their different types, then by explaining its main components. With a special focus on blockchains which will be used in the prototype. Then a special section is dedicated to consensus which is crucial when it comes to distributed systems. Finally, examples of such technologies are presented.

Distributed Ledger Technology (DLT) are a group of technologies composed of a distributed database on which state is agreed by the nodes of the systems through consensus mechanisms. Once consensus is reached, data is stored in a immutable way. The most popular one are blockchains, but there other types such a acyclic oriented graph DLTs such a Tangle [24] or Hedera Hashgraph [25]. For the rest of the section, a focus is given on blockchain as it will be the tool used in the prototype developed further below.

2.2.1 Blockchain

A blockchain is a distributed database that maintains a continuously growing list of records called blocks. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. This provides a chronological, immutable record of transactions that all parties can view. By design, blockchains are inherently resistant to modification of the data. This means that no single party can ever erase or manipulate the information stored in a blockchain.

Blockchain technologies are nowadays being incorporated in several different industry sectors. The popularity of these technologies rose because of Bitcoin's proof of concept, drawing the whole world's attention [14]. Later, Ethereum made use of an ancient concept called smart contracts [15][16]. This new feature further extended the potential of blockchain networks.

Blockchain can be devided in two broad categories - permissionless and permissioned. A permissionless blockchain is an open system that allows anyone to join the network and participate in the peer-to-peer transactions that are carried out on the network. Examples of permissionless blockchains include Bitcoin [14] and Ethereum [15]. On the other hand, a permissioned blockchain is a closed system that is available only to a limited number of participants. Permissioned blockchains are typically used to provide secure online access to transaction records between businesses and clients in healthcare and government applications. Examples of permissioned blockchains include IBM's Hyperledger Fabric [26] and R3's Corda [27]. They are further explained and compared in the **Section** 2.2.4.

Let us now dive into the different properties of blockchain and then explain what a smart contract is.

2.2.2 Properties

We discuss the mainly properties that blockchains can face. In this thesis, three main categories are considered : **Scalability & Performance**, how blockchains scale up and become more efficient, **Governance**, how the control is managed or who manages it, and **Security & Privacy**, how to ensure the whole system's safety.

Scalability & Performance

Scalability and performance aim to describe the ability of a blockchain system to maintain an efficient and effective level of service under high load. This load can be measured with different metrics. In the following paragraph, a description of two important metrics is given.

First, as high amounts of data might need to be recorded on the blockchain within a certain amount of time. Throughput is a first candidate metric to assess the performance of a system. In fact, permissionless blockchains such as Bitcoin or Ethereum are only able to process one to two figures numbers. In comparison, real world payment applications such as VISA process up to 56 000 Tx/s. A real gap between real world requirements and currently available technology might exist in terms of throughput. However, not all use cases are throughput sensitive since they might not need to process thousands of transactions per second. For instance, a blockchain keeping track of car registrations in a country would not require such a high throughput. In 2018, some 273 million vehicles were registered in the United States [28]. Therefore, as long as the registration of cars is uniformly distributed throughout the year, a rate of approximately 8 Tx/s is sufficient.

Secondly, the memory used to store the blockchain represents another metric that can be crucial when running nodes. In fact, for some IoT applications, where hardware resources can be scarce, this can be critical. In the beginning of 2021, storing the complete Bitcoin blockchain needs up to 317 GB, about 50 GB more compared to the same month, the year before.[29] It increases at a rate that requires the nodes to invest more and more in storage. Regarding permissioned systems, as the whole chain might not be needed to perform the business logic, data can be often stored off-chain.

Governance

The governance in a blockchain is defined as the "means of achieving direction, control, and coordination of stakeholders within the context of a given blockchain project to which they jointly contribute" [30]. By definition, for permissioned blockchains this property resides in its nature. Here, the challenge will be how to manage this loss in decentralization and how to distribute this control power among stakeholders.

Security & Privacy

Security can be split into 6 different goals: Confidentiality, Authenticity, Integrity, Availability, Accountability and Authorization. Let us now go through each of those and link them to the needs of blockchains.

- **Confidentiality**: the ability to hide data from external entities. In blockchains this is often addressed by cryptographical methods. However, as it is a distributed ledger, it can be audited by any node.
- **Authenticity**: it represents the ability to prove the origin of a transaction. In all blockchain systems this is ensured by digital signatures.
- **Integrity**: it refers to the ability to verify if data has been tempered or manipulated. This is directly address by the nature of the blockchain itself.
- **Availability**: it concerns the ability to keep a minimum level of service at all times. This is directly addressed by the decentralized nature of the system.
- **Accountability**: it refers to the ability to trace actions in a system.
- **Authorization**: it refers to the control access layer of the system. When it comes to permissioned systems, this represents a key factor as it determines who takes part in the system.

Another important aspect of security, is the distributed security. It addresses vulnerabilities related to the distributed nature of the system, such as Byzantine fault tolerance. However this topic is addressed later in the consensus section.

Finally, privacy concerns the ability to control which data is visible to whom and to erase or modify private data. This is one of the main challenges since by its nature, where all transactions are broadcast and appended to the blockchain for transparency, hiding data can be difficult.

2.2.3 Smart Contracts

A smart contract is a computer program that automatically executes the terms of a contract without human involvement. Because it is written in computer code, it cannot be changed once it has been created. This makes the program more reliable and secure than traditional paper contracts. It can also be used to distribute goods

and services over the internet quickly and efficiently. Smart contracts are often used for managing complex financial operations such as banking transfers, currency trading and insurance claims. They are also becoming increasingly popular in corporate legal departments as a way to streamline contract negotiations and other business transactions.

Despite their many advantages, smart contracts also have some significant drawbacks. First, they are extremely difficult to change once they have been executed. This can lead to a number of legal disputes that can be expensive and time-consuming to resolve. Second, it can be difficult to verify the accuracy of a smart contract once it is completed. As a result, parties may accidentally agree to terms that are not actually included in the contract. These problems will likely become even more common as the use of smart contracts expands in the coming years.

Although smart contracts have enormous potential to improve business operations and government services, widespread use is still a long way off. Governments must continue to develop and implement regulations to protect the privacy and security of their citizens. Currently, smart contracts can be considered as real contracts in EU because the execution of contractual obligations is performed in the blockchain network and cannot be altered or cancelled by anyone except the owner who is the only person having his private key. However, since there are no clear regulation on the nature of contracts for this area, the legality of smart contracts will largely depend on the courts that are deciding cases on the question at hand. There are also questions regarding the ability of these contracts to compete with traditional contracts in the future. If successful, smart contracts could significantly decrease the number of lawsuits involving breach of contract and provide greater certainty for all parties involved. However, if the technology is not fully developed and standardized, it may still take years to reach its full potential [1].

2.2.4 Permissioned and Permissionless

Permissionless blockchains (e.g. Bitcoin) allow anyone to participate as users, block validators, developers, or community members. All transactions are fully transparent, meaning that anyone can examine the transaction details. One of the main characteristic of permissionless chains is that they have a token associated with them. This token, exchangeable in fiat currency on specific markets, is typically designed to incentivise and reward participants in the network for behaving correctly (e.g. if you create a block

[1] `https://www.theconversation.com/why-smart-contracts-will-be-the-future-of-law-but-will-it-take-50181`

under certain requirements, you receive 25 bitcoins).

A straight forward definition of permissioned blockchains is found in ´Revocable Attribute-Based Signature for Blockchain-Based Healthcare System´ by Su QianQian, describing permissioned blockchains, "as an additional blockchain security system, as they maintain an access control layer to allow certain actions to be performed only by certain identifiable participants" [31]. In other words, participants need consent from other designated participants to join the chain. Other parties outside this blockchain network cannot have access to the transactions that are only available to ecosystem participants. Furthermore, the access control layer allows the creation of separate channels between different stakeholders inside a permissioned system. In these environments, *Tokenisation* is optional. These systems are more often used in industry sectors, e.g. to track supply chain processes[21].

Property	Permissionless	Permissioned
Decentralization	truly decentralized	partially decentralized
Privacy	transparent network	immutable private transactions
Trust	trust-free	trust on authorized nodes
Scalability	difficult	easy
Security	highly secure	less trivial

Table 2.1: Functional comparison between permissionless and permissioned blockchain

As **Table** 2.1 shows, the main difference resides in decentralization, trust, scalability and security. Decentralization is a key element in permissionless systems, however permissioned blockchains are less decentralized since an access control layer might be run by a central authority. Similarly, trust is distributed differently in each system. In permissionless blockchains, nodes are not trusted at all, whereas in permissioned systems the network relies on a minimum of trust based on the authority of nodes. Regarding scalability and security, permissionless blockchains spend energy and time securing the network whereas permissioned systems focus on scaling the performance of the system.

This comparison highlights the well-known scalability trilemma. Experience has shown, that an intrinsic trade-off between scalability, security and decentralization exists when implementing blockchain systems. Permissionless blockchains lay mostly in

the security-decentralization side of the triangle. As mentioned before, they focus on securing a trust-free network in order to keep it fully decentralized. On the other hand, permissioned systems reside mostly in the security-scalability side. In this side of the triangle, a part of the system is centralized (e.g. Authorization) allowing the system to be scalable and secure. However, it implies that peers do not have the same level of participation in the systems, creating distributed security issues, such as a single point of failure.

From a technical perspective, the main difference between permissioned and permission-less systems resides in the consensus mechanism. To summarise this consensus issue, let us consider one of the main risks with a decentralised ledger. Several blocks which contain information / transactions compete because they are sent at the same time. The question is to know which of these blocks should be considered as the reference for the block chain. A consensus algorithm establishes how the different participants agree on this reference block that is also defined as the state of the chain system. As permissionless blockchains work in trust-free environments, consensus is based on anonymous elements such as proof of work(PoW) or proof of stake (PoS). With the PoW method, the nodes that want to create a block have to solve a complex mathematical problem. This resolution can only be achieved by testing all the possible permutations to achieve the required result. The speed of resolution is a function of the computing power that the node makes available and of the complexity of the result to be achieved. The reference block is that of the node that was able to solve the problem first. The main drawbacks of this method are the energy cost linked to the computation power needed to resolve the mathematical problem and the low transaction rate. In the second method, proof of stake (PoS), we can summarise and simplify it like this. The node that creates the reference block is chosen based on a measure of its wealth/stake. The greater the wealth of a node, the greater its chances that its block will be selected. However, if it makes a mistake, such as validating a false transaction, it loses his wealth. Here also, the rates of transactions, though better than in PoW, are low.

On the other hand, permissioned systems are based on proof of authority (PoA) consensus type. Those algorithm, such as Practical Byzantine Fault Tolerance (PBFT) [32], are based on a set of known nodes. The nodes of a PBFT system are ordered sequentially, with one node designated as the leader and the rest are referred to as backup nodes. All nodes in the system communicate with each other. The leading node offers a block to the backup nodes and the block is validated if the backup nodes reach an agreement on its validity using a majority rule ($>2/3$). Then, a similar phase occurs to commit the validated block. Furthermore, they provide a higher efficiency in terms of

transaction rates as blocks are directly settled.

Technology	Permissionless	Permissioned
Cryptography	✓	✓
Consensus	PoW, PoS..	PoA, IBFT, RAFT,..
Transaction and Ledger management	✓	✓

Table 2.2: Technical comparison between permissionless and permissioned blockchain

2.2.5 Consensus

As discussed in the previous section, consensus is a key criterion to differentiate permissionless and permissioned blockchains. Consensus is the mechanism through which the global state of the chain is coherent and maintained. It also has the ability to prevent distributed security issues such as Byzantine attacks. Byzantine fault tolerant systems are able to run correctly even if a certain number of nodes are acting maliciously or randomly. Different consensus algorithms achieve different levels and types of fault tolerance. In this thesis, two main types are mentioned: Byzantine fault tolerance, which was already explained, and crash fault tolerance. The latter refers to the property of a system to tolerate a certain number of nodes to fail, e.g. to be turned off.

As trust is differently distributed in permissionless blockchain, consensus has to be ensured by every participant. Therefore the systems need to be structurally safe and provide incentives to penalize bad behaviour and reward correct one. On the other hand, permissioned systems rely mostly on a Proof of Authority scheme, where the participants take part in the consensus as authorized nodes and focus on keeping and correct state. Several consensus mechanisms appear in permissioned systems, but they are mostly based in PBFT and its variants, such as IBFT [32][33]. This type of consensus allows achieving Byzantine fault tolerance up to one third of the nodes being compromised. However, other consensus algorithms such as RAFT allow to achieve higher throughput but only achieving crash fault tolerance [34]. Let us now briefly explain how this consensus algorithms work.

- **PBFT/IBFT**: it consists of a 3 phase vote consensus. In the first step, a client request is submitted to the primary node of the system. It issues a pre-prepare vote to all nodes. If a node validates the transaction it broadcasts its vote to the nodes. If it gathers more than two thirds of votes, it sends a commit message where the nodes vote for appending the transaction to the ledger. If transaction gets two

thirds of the vote, the transaction is added. If the primary node fails, then a round of votes is triggered to elect a new primary. IBFT differs from PBFT by allowing the set of validators to be dynamic instead of static. This configuration runs without a client, but having all validators propose transactions [32][33].

- **RAFT**: it presents a similar two round vote system like PBFT, with a leader node handling all requests. However, in this mechanism the other nodes do not verify transactions and trust completely in the leader. This improves its speed of achieving consensus but reduces its fault tolerance [34].

The consensus mechanisms mentioned here represent the basic blocks for many other consensus algorithms in permissioned systems (e.g. Tendermint). It shows how time is saved as finality is reached without being exposed to fork issues.

2.2.6 Use Cases & Applications

This section illustrates how permissioned blockchains can be used. Two use cases, along with a specific application, are mentioned. The first concerns the supply chain sector, with an application in the food industry. The second refers to the energy flexibility market, with an application in an energy trading platform.

Supply chains

As supply chains have become a more complex network of suppliers and consumers: traceability of a product turned into a hard task. Nevertheless, delegating the responsibility of tracing to one of the suppliers might create other problems as it centralizes power in the hand of one entity. This is why blockchain along with its distributed and transparent systems has gained popularity in this sector.

Walmart, a chain of hypermarkets, developed a permissioned blockchain solution in order to improve the traceability of food. In fact, it can be crucial to trace the provenance of food when some diseases coming from this product is detected. Thanks to a blockchain based on Hyperledger Fabric, Walmart managed to reduce its tracing time of mangos from 7 days to 2.2 seconds.[26][21] Permissioned blockchains suited this case well, as a limited amount of stakeholders (i.e. Walmart, farmers, transport companies..) are the only ones who need to consent on information.

Energy exchange

As the world of energy production is becoming more and more decentralized with renewable energies, the need for decentralized coordination has grown. The objective being to improve the self-consumption of producers, reducing risk of congestion in electric grids.

The Enerchain project develops a blockchain based solution for a energy trading system between peers prosumers.[35] The solutions is based on a Tendermint consensus which is a PBFT variant. It shifts some security issues from the blockchain layer to the application layer making use of cryptography methods.

2.3 Related Work

In this section, previous research in the field of prototyping blockchain systems to enhance communication systems are presented.

In [17], a blockchain framework is presented, discussing about how it can improve robustness and security in power systems. They proposed a blockchain where smart meters are the nodes taking part in it. They then compare centralized systems to the blockchain based one based on probabilities of success attacks on the grid. This paper clearly shows how distributed systems can reduce the attacks in a systems composed of hundreds of entry points. However, it lacks of a proof of concept software to demonstrate the implementation of such a system.

On the other hand, [18] shows how to increase self consumption inside a community equipped with solar panels. They implemented a distributed algorithm to solve the optimization problem based on game theory. They concretely deployed and tested the blockchain based system and obtained an increase of self-consumption rate on the local simulation grid. Here the focus is not on compliance monitoring but how energy is consumed and balance between users while maximizing their utility function.

In [36] authors highlight the importance of visibility, automated analysis and response time of distribution systems and proposed a blockchain based system to enhance power communication systems. As in the first mentioned paper, they did not implement the system and show how it would behave dynamically.

Lastly, [37] is not a paper related to power systems, but related to the field of multirobot systems instead, demonstrate how using a blockchain based communication

system can ensure integrity and compliance inside of a group of robots which need to be coordinated and respect directives.

Chapter 3

Problem Statement

In this chapter, the problem and objectives of this work are defined along with the exact context and assumptions. First let us remind the two main motivations in the following sections and then formulate the research question as a list of steps to address the problem.

3.1 Congestion management & DERs contract compliance

Currently the process of integrating a DER in the MV network is plenty of non automated steps and not easily verified and controlled. Moreover, when a DER is up and running, the compliance to congestion management through ANM techniques such as power curtailment are not easily being tracked. To ease the procedures, ORES would like to automate and simplify those process which will improve the quality of the supplied service along with the reliability of the congestion management reducing risk of infrastructure damage.

As mentioned, this thesis focus only on the generators nodes of the network directly connected to the MV stations, i.e. all electricity producers with at least 250 MW of producing power. Even if smaller producers inside à LV network could be aggregate to represent a global generation producer, focusing on single entities that manage enough power through a single legal contract allows to easily interact with them when flexibility is needed. **Figure** 3.1 shows then how the architecture simplifies, from the left network to the right network.

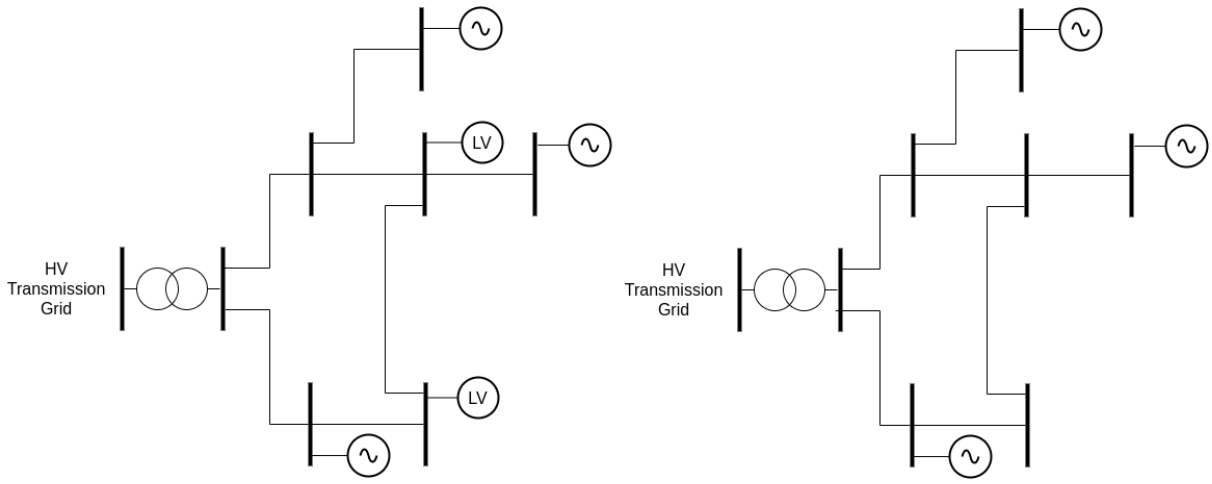


Figure 3.1: Network simplification for this thesis prototype

3.1.1 Communication security and cost

Having a central control service controlling and storing all information related to the grid represents of a single point of failure. Moreover, these monitoring and controlling activities often implies sensitive data. Hence, not only a reliable and resilient system needs to support all these functions where accountability can easily be established.

We distinguish three types of cyber-attack. The first aims to block the computer system of the DSO so that it can no longer operate. This kind of attack is usually coupled with a ransom demand to stop the attack. Solutions like a VPN, firewall, and special supervisory control (e.g. traffic flow analyse) exist. Unfortunately, the more the system is connected to different information sources, the greater the number of opportunities for a cyber-attack. The second type of cyber-attack targets the decentralised information asset. Here, the main openings for a cyber-attack are the RTU and the PLC. For example, in mid-June 2010, a computer virus specially designed to attack Windows-based industrial computers (STUXNET) took control of PLCs, influencing the behaviour of sensors, sending false information, and leading to frequency instability phenomena. The third is data theft or data alteration when the information is sent to the DSO's SCADA. To achieve data takeover, attackers must, in summary, perform three operations: (1) access the equipment where the data is stored and exchanged, (2) access the data, and (3) ex-filtrate the data. If the attacker also wants to modify them, after ex filtration, he must also modify data and store them in places of the initial data. If we are able to detect and stop the attacker at one of these stages, then we can consider the system to be (relatively) secure. It is important that the information captured from the local device is send to the SCADA system in an immutable manner.

3.1.2 Production monitoring

Monitoring producers has a crucial importance to ensure the reliability and security of the system. Since not all producers are directly monitored, the generation units being monitored could be lead to cooperate less as they see other unit not being monitored. In numerous studies and experiments, the general trend showed a decrease in contributions when people who cooperated realised that there were people not cooperating to increase their benefits. Eventually, more people would stop contributing and there would be no benefit to it, hence leading to zero contribution in public goods and services. Currently, ORES only monitors with RTUs some ranges in MV networks (i.e. $>$ 1000 kVa), creating a gap in monitoring procedures between different producers.

3.2 Research question objectives

Let us now remind the research question, and then transform it to specific goals.

Research Question

> On behalf of the DSO, can a DLT monitor the contractual requirements of the generation units within the framework of an ANM scheme and within sufficient resilience to cyber-attacks?

To tackle this question, five intermediate objectives are to be fulfilled and analysed :

- Choose a DLT platform that suits the use case of DERs monitoring at ORES,
- Specify the monitoring and control functions,
- Design the DLT systems in the chosen platform for the mentioned functions,
- Implement a proof-of-concept prototype simulating the the proposed design,
- Simulate specific scenarios, report and analyse results.

Chapter 4

Use Case Platform Assessment

This chapter exposes how the choice of the DLT platform was made and which one was retained to implement the prototype. Let us first defined the criteria and metrics on which the benchmark is made. Finally, the platforms candidate are presented along with their score.

4.1 Criteria and Metrics

As highlighted before, the monitoring process needs to be efficiently done. Nodes might be running on edge devices, therefore hardware constraints must be taken into account to ensure correct service. Smart contracts support has to exist as it will allow automation tasks. Security, and more specifically authorization and privacy are also required as ORES needs to control how and who access the system. Lastly, popularity and support are also to be taken to account as it ensures and eases the maintenance of the project in the long term.

Therefore the set of criteria reflecting the needs of the use case and used to compare the solutions are the following: Consensus, Language Support, Transaction Rate, and Security.

4.2 Platform Assessment

Now that the criteria of the use case are set, let us assess the different platforms. The retained choice will be later use to implement the prototype. Let us notice that all permissionless are directly discarded as they are not able to provide authorization and privacy. Therefore, the comparison is based on the following permissioned systems.

- Quorum
- Corda
- Hyperledger Fabric

The following part of this thesis will focus on a benchmark analysis of common enterprise solutions to provide a better overview of the state of the art. Therefore we will first introduce three established solutions and compare them to each other.

Quorum

Quorum was founded by the J.P. Morgan bank and is a soft-fork of Etherum. [38] Thus, its native cryptocurrency is Ether. Another side effect of being a soft-fork of Etherum is, that the only supported programming language is Solidity. A Quorum Node consists of 3 main components: Transaction Manager, Crypto-Enclave, and Network Manager. **Figure** 4.1 shows the Quorum architecture.

The Transaction Manager provides access to the encrypted data of private transactions. Moreover, he also manages the local data store and communicates with other Transaction Managers on different nodes. Encryption and decryption, as well as managing the private key is the responsibility of the Crypto Enclave. Lastly, the Network Manager controls the access to the network. This last component makes Quorum a permissioned Blockchain.

In Quorum, nodes in the network are called parties. To reach consensus in Quorum a majority of parties need to agree on a block in order to persist it. As consensus algorithms, Raft and IBFT are supported.

Corda

Corda was established in 2016 by R3 with the goal to combine the advantages of interoperable public blockchains with fully private enterprise blockchains.[27] It is not soft-forked from any other blockchain and only supports so called FIAT-currencies. FIAT-currencies are government-issued and are not backed by assets such as gold (e.g. Euro, Dollar). One of the main advantages of Corda is the support of widely practiced programming languages, namely Java and Kotlin.

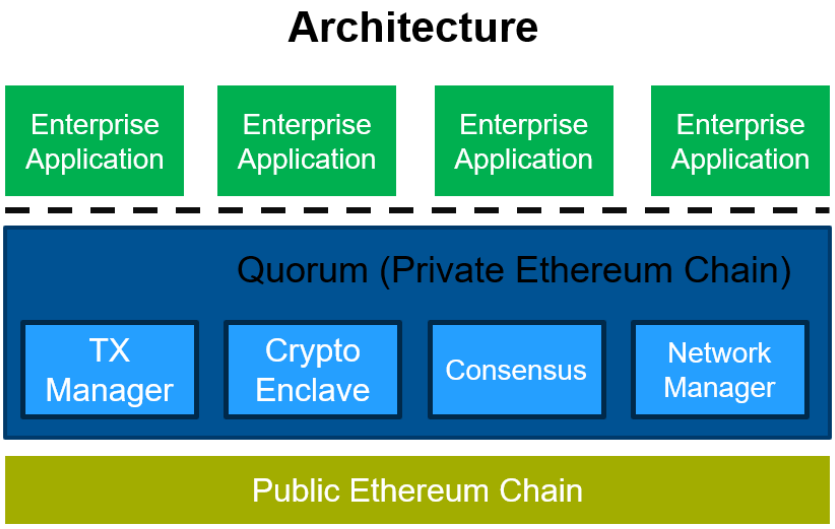


Figure 4.1: Architecture of Quorum. Adapted from Quorum White-paper v0.2 [38]

To achieve the goal of combining public and private blockchains, the global Corda network, as seen in Figure 4.2, consists of multiple permissioned subnetworks. Every node can be part of multiple subnetworks and furthermore a node can decide if it wants to offer its service only to members of the permissioned network, or to all other nodes.

Consensus in Corda is split between transaction validity and transaction uniqueness. Transaction validity is reached between all required signers of a transaction by executing and verifying the deterministic chain code. Transaction uniqueness defines the constraint, that a transaction is the only consumer of its given input states. To achieve that Corda introduced so called Notary pools. Every subnetwork can have its own Notary Pool, it is even possible to introduce multiple Notary Pools in the same subnetwork. A Notary Pool consists of several mutually distrusted predetermined observers who consent on the uniqueness of input states. [27] This consensus is pluggable and can be different for every Notary Service. Another advantage of the Notary Pool is, that they only need to know the input states of a transaction, but neither the chain code nor output states. This increases privacy in the Corda network.

Fabric

Fabric was developed by the Linux Foundation in cooperation with IBM. Chain code can be written in either Go, Java or NodeJS. It natively does not have any cryptocurrency but can be extended with the so called FabCoin. Furthermore, Fabric also supports a pluggable consensus protocol.

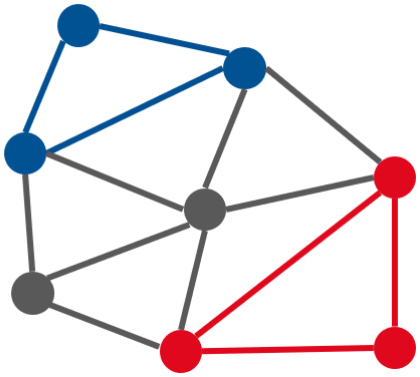


Figure 4.2: Global Corda Network. Adapted from The Corda Platform: An Introduction; Richard Gendal Brown; [27] Figure 6

A notable architecture difference of fabric compared to most other permissioned blockchains is, that transactions are processed in an “execute then order then validate”-way in comparison to the common “order then execute”-way. [26] This means that in Fabric a transaction is first simulated and a ReadWrite-set (rw-set) created. After that, the Ordering Service Network (OSN) orders the rw-set and creates an atomic broadcast on which consensus is achieved. Lastly, endorsements of other peers are validated and in case of a successful consensus, the block is appended to the chain.

Fabric Network Architecture

Let us outline the components of a Hyperledger Fabric distributed ledger system.

- **Peer**. A node which stores the ledger and chaincode. Building block of the Fabric system.
- **Orderer**. A node in charge of ordering transaction into blocks and of consensus mechanisms.
- **Channel**. This represents a logical network hosting a specific ledger. A peer may take part in multiple channels corresponding to each ledger it hosts.
- **Client**. an application that is allowed to invoke chaincode and interact with the ledger without storing all its content.
- **Chaincode/Smart Contract**. Code which determines the rules under which transactions on the ledger may happen. A chaincode is a set of related contracts but it is often used interchangeably with the term "Smart Contract".

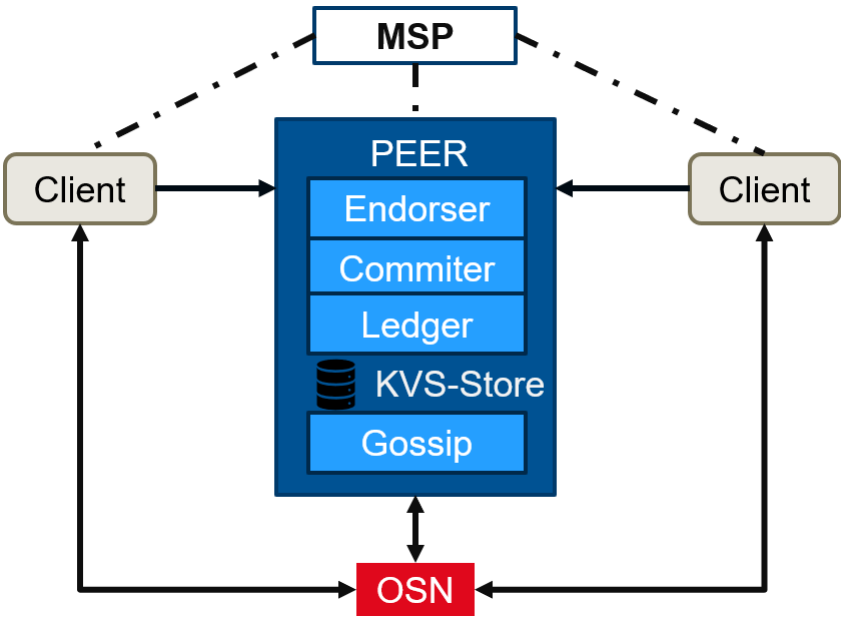


Figure 4.3: Architecture of Hyperledger Fabric. Adapted from Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains; Elli Androulaki et. Al. [26]; Figure 3

4.2.1 Comparison

In this section advantages and disadvantages of each permissioned blockchain solution are inspected. Our choice and comparison is also supported and based by the results of a benchmark of different platforms in [39].

Consensus

The most integral part of every blockchain, either permissioned or permissionless, is the consensus. Although there exists a wide variety of different consensus mechanisms, as seen in Section 2.2.5, in enterprise solutions RAFT or different variations of PBFT are used. [40] The main trade-off in terms of consensus is fault tolerance compared to transaction throughput. This criterion is equally satisfied by Corda and Fabric because they both do not restrict the user to use a certain consensus. Quorum does not fulfill this criterion as well, as the other two. It does support common consensus protocols, like Raft and IBFT, but does not give the user complete freedom. [41]

Language Support

The next comparison criterion is language support. Here a balance must be maintained between providing a safe environment for writing chain code and appealing to a wide range of developers. The latter is especially important for new solutions to become quickly adapted. In this category, Quorum is limited because it is a soft-fork of Ethereum and as

such only supports Solidity. While Solidity is specifically designed for writing chain code, it developer base is still small compared to the languages supported by Corda or Fabric. [42]

Transaction Rate

Often when talking about enterprise blockchain solutions, their transaction-rate is discussed. While this is certainly an interesting criterion, it is very hard to compare results from different papers, due to their varying hardware setups and an often different configuration of the blockchain. Another reason why rating the transaction throughput might be misleading is that for a lot of enterprise use cases the transaction-rate is not the deciding factor. Hence, we will only provide some rough numbers in this paper. According to the article "Menapay Blockchain Tests: Quorum TPS" [43], Quorum has a transaction rate between 90 and 150 transactions per second (Tx/s), which is the lowest transaction rate of those three. Corda increased their throughput with their Enterprise 4 update from 170 Tx/s to around 600 Tx/s [44]. The highest throughput is archived by Fabric with a throughput of around 2500 Tx/s. [41] While there is certainly a great discrepancy between those values it is advisable to have a look into the references of this section and compare the environment in which those rates have been archived.

Security

Lastly, we want to discuss the security measurements in place for those solutions. As all of them support some implementation of BFT, they all have a fault tolerance of $t < \frac{n}{2}$ with t being the number of crashing nodes and tolerance of $f < \frac{n}{3}$ against subverted nodes. With f being the number of subverted nodes. [40] Moreover, all solutions have encryption algorithms in place to protect the integrity of data. Additionally, platforms have other security measures in place. Quorum and Fabric rely on the Zero knowledge proof (ZKP) to hide data. The ZKP guarantees that the probability of finding data on the chain is $\frac{1}{n}$, with n being the number of blocks on the chain. This means, that the security increases as a chain matures. Corda on the other hand relies on Hardware security modules (HSM), e.g. Intel SGX. Those modules consider everything outside the chain as a possible attacker, hence Corda is only hackable from within using potential loopholes in the chain code. This strategy is called Doorman strategy. [41]

4.2.2 Rating

After discussing every criterion in detail, we can now rank every solution. This is done by first rating every criterion separately and then, calculating the sum of all criteria. Achievable values range from 1 to 3. The rating is adapted from ‘Comparative study of

permissioned blockchain solutions for enterprises‘ by Rana M. Nadir.[41] But as mentioned above, we removed the throughput from consideration. . The final rating can be seen in **Table** 4.1. It shows that Fabric is the dominant solution in nearly all criteria. While Corda is a close second, the gap to Quorum is considerably large.

Criterion	Quroum	Corda	Fabric
Consensus	1	3	3
Language Support	1	2	3
Security	2	3	2
Transaction Rate	1	1	3
Summary	5	9	11

Table 4.1: Rating of permissioned blockchain solutions

Chosen platform : Hyperledger Fabric

Based on the score obtained, the chose platform is Hyperledger Fabric. Its ever-evolving ecosystem which provide development tools and its high-efficiency matches the need of a reliable prototype. Furthermore, the community behind allows to easily find support and documentation which can also ease the development of a blockchain.

In our case, Fabric can be combined with Fablo, which allows to build Fabric networks locally in a reliable and simple manner. More details will be exposed in the implementation section of next chapter.

Chapter 5

MonitORES proof of concept

The following chapter describes first the specifications of the prototype being built, then explains how it was implemented and tested. Finally, a discussion on security and communication goals with respect to the developed prototype is presented.

5.1 Functional Analysis

In this section, the analysis of the prototype is presented as an exhaustive description of all the functional features it should contain.

5.1.1 Overview

MonitORES is a set of services that aim to automate and simplify the monitoring and control of decentralized generation units the ORES distribution grid. It is a prototype based on DLT technology and more specifically based on Hyperledger Fabric. The deployment of such a network would reduce infrastructure costs while giving freedom and autonomy to the producers regarding their on premises measurement obligations.

Stakeholders

- DSO (ORES)
- Energy producers (DERs)
- TSO (reporting and action towards)

Services

- **On-boarding:** When a generator starts up, a service takes care of integrating it and checking the conditions for putting it into production.
- **Control/Commands:** The DSO has the possibility to control the generators by updating their status and by submitting a constraint to them.
- **Monitoring:** A service permanently monitors the respect of the status and constraints through the constant sending of measurements by the producer.
- **Visualization:** It is possible through a web interface for the DSO to visualize the status of the network under study. The producers can visualize the state that concerns them.
- **Interface:** It allows to start the control/command operations from the DSO. It allows the producers to activate the monitoring.
- **Supervision:** Visualization of the DLT status.
- **Data Mining:** The collected data allows us to determine the state of the network and to exploit it for its analysis.
- **Out-boarding:** When a producer leaves the network, a service takes care of the necessary checks.

5.1.2 Functional features

In this section, the details about the different prototype's features are described, highlighting how they interact.

On-boarding

On-boarding concerns the first steps executed when integrating a new production unit to the network. It can be divided into two parts: Power Generation Unit (PGU) creation and initialization tests. Let us now specify what is done during this two processes.

PGU creation consist in creating and attributing an instance of the PGU model to a new customer joining the network. ORES submits the creation containing all contractual information to the blockchain and the producer is then allowed to submit information concerning its PGU. When the PGU is first created, its status does not allow it to directly produce.

Next, initialization tests take place in order to release the PGU to full production. Those tests ensure that the communication is behaving correctly. Three types of tests happen:

- time delay tests: verify that the PGU is able to respect the frequency of measures imposed,
- power limit tests: check that the PGU is able to reduce/increase its production in response to constraint,
- infraction tests: verify that the infraction systems (see 5.1.2) works.

Control/Commands

As mentioned, control is one of the main pillars in ANM. Controlling the generation is available to ORES from two different ways:

- Updating status: Each PGU has an status who determines a set of constraints that can apply to it. The DSO can declare alert state and urgency state. He might also bring a PGU back to initial state.
- Submitting congestion constraint: When the distribution network is in congestion mode, ORES submits an additional constraint to producers operating in normal status. This constraint usually comes from the ANM calculator.

Monitoring

Monitoring is the second pillar of ANM strategies. In our case, monitoring takes place through the automated checks of PGU's measure at a specific time interval, updating the status and infractions of the PGU. Let us now describe the infraction system which impacts monitoring and control functionalities.

Infraction system

A system of infractions is be used and updated in the ledger. The system is based on the tit-for-tat game theory strategy [45] which maximizes everyone's rewards. This strategy consists of penalizing the opponent immediately after a fault but forgiving them after a correct behaviour or when time elapsed.

It consists of establishing three levels of fault: minor, major, and critical, each of them leading to a different punishment. **Figure** 5.1 shows how each fault is triggered, the consequence and its forgiveness rule. Minor faults consist of missing measurement activity from a PGU. In fact, as the ANM computation requires near-to-real-time data, stopping measuring the network could generate a wrong optimal constraint value. They are forgiven after x amount of time, which depends on constraint time parameters. When experiencing a minor fault, it does not attribute any penalty unless more than two minor faults occur in the same forgiveness time window. Major faults are triggered by exceeding constraints submitted by the ANM computer or by accumulating three minor faults. They are forgiven after y amount of time which is strictly greater than x. When a PGU accumulates more than one major fault, alert mode is triggered. Alert mode will limit the production to the maximal capacity allowed and stated in the contractual data. This is corresponds to imposing action $a^{e1}_{g,m,0}$ for the faulty PGU m . The last type of fault is the Critical faults. This is triggered by exceeding the alert mode limit or by accumulating three major faults. They can be forgiven after z amount of time. However, this can only be done after the DSO checks the conditions and reasons for the fault. When more than one critical fault occurs, the PGU gets a total curtailment. This is when the urgency mode $(Urgency) \mapsto 3$ is applied only to this guilty PGU. If the PGU still accumulates critical faults, then it is totally disconnected from the grid.

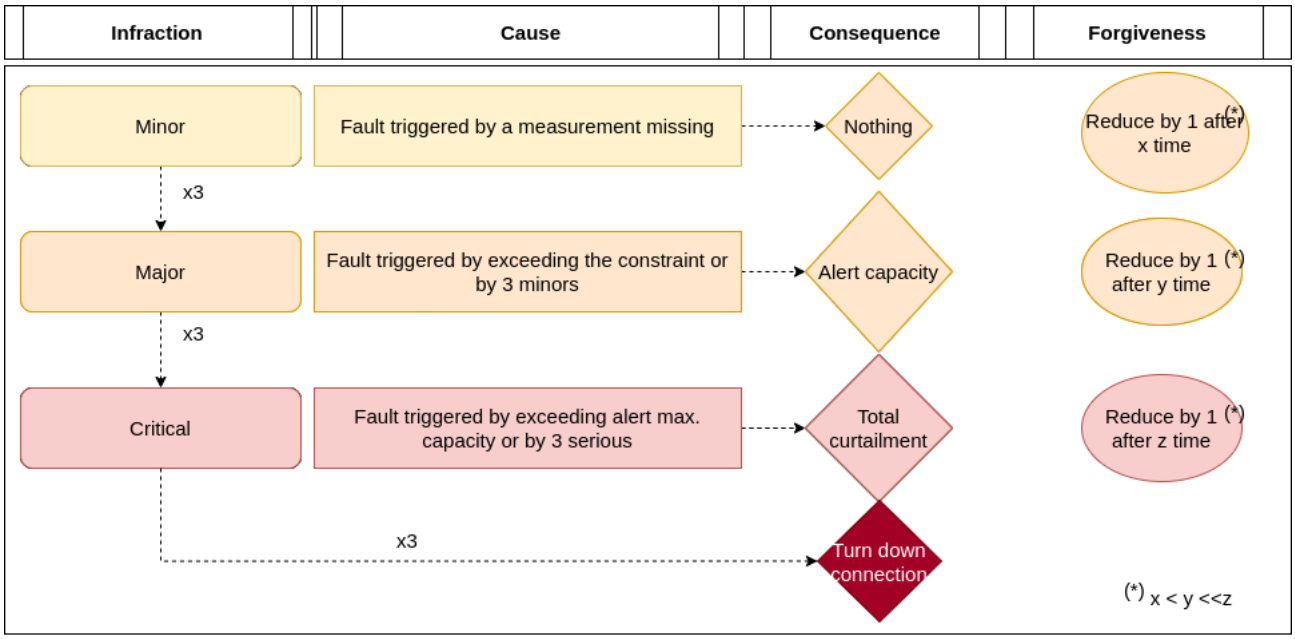


Figure 5.1: Punishment system proposed in the constraint management functionality

Visualization

In order to facilitate data analysis and control by a human operator, a web interface is proposed. It possess the following features:

- Production tracking: The PGU production and current imposed constraint are display in a live chart.
- Infraction tracking: Current infractions displayed along with the last time they occurred.
- Limit simulation: Ability to submit a constraint to the selected PGU, as well as declaring alert or urgency states.
- PGU details: Contract information of the selected PGU.
- PGU list and creation: A PGU creation interface that will be followed by the on-boarding live data. All PGUs will be available in a list where they can be selected.
- User management: Creation and management of the interface users.

Interface

The interface is the set of tools that enables PGUs and the DSO to interact with the main system being the hosted in the blockchain. Compared to the visualization, the interface does not necessarily provide a graphic interface, but provides the endpoints to easily submit or request data from the system. Those endpoints must fulfill three main functions:

- Production submission: Producers send data to the system through it.
- Status collection: Stakeholders can gather data from the current state of the system depending on their authorization level.
- Constraint submission: ORES must be able to connect its current system and submit constraint to the system.

Supervision

From a technical perspective, the operators need to monitor the state of the blockchain layer. An interface where the status of transactions and blocks are displayed.

Data Mining

All data collected in the system is used to improve the evolution of the system as well as the analysis of it. For instance, this is the bridge between the storage system and the ANM system.

Out-boarding

When a producer quits the distribution network, final checks are made and the close of the connection is triggered. Data must be archived and taken out of the system.

5.2 Implementation

Let us now focus on how all features and services were implemented. First going through the global architecture of the MonitORES software as well as the data model chosen for the network elements. Then, we dive deeper into each component of the architecture. In particular, for the blockchain component, it is explained how the smart contracts take in charge the main functions described in the previous sections.

5.2.1 Architecture

To facilitate the illustration and understanding of the MonitORES prototype, we split the architecture into two perspectives. First, the global prototype architecture, abstracting the blockchain into one component. Secondly, the blockchain architecture that will be discussed in its own section.

MonitORES architecture

The global prototype architecture is shown, where the blockchain is seen as whole unique functional service. **Figure** 5.2 illustrate a high-level view of the architecture. In this figure, one can identify four main components that will be later detailed in their own section:

- Blockchain: This component is composed of the Hyperledger Fabric network itself, a REST API to ease queries and a explorer web-view displaying the technical state of the blockchain. The Fablo [46] simulation tool was used to instantiate the proof of concept network.

- PGU Simulator: This service provide the real-time simulation of PGU. Their source can be set to solar or wind. This component is implemented using NestJS [47] micro-services features. Simulation data source can be set to come from an API (OpenWeather API [48]) or from wind historical data from Elia [49].
- API Gateway: To orchestrate all communications and different protocols used, this components facilitate how the PGUs and Web-views trigger actions on the system. It is also implemented using NestJS.
- Web-view: This component is a Single Page Application implementing visualization of data and control actions for admin users. It is implemented using Angular.

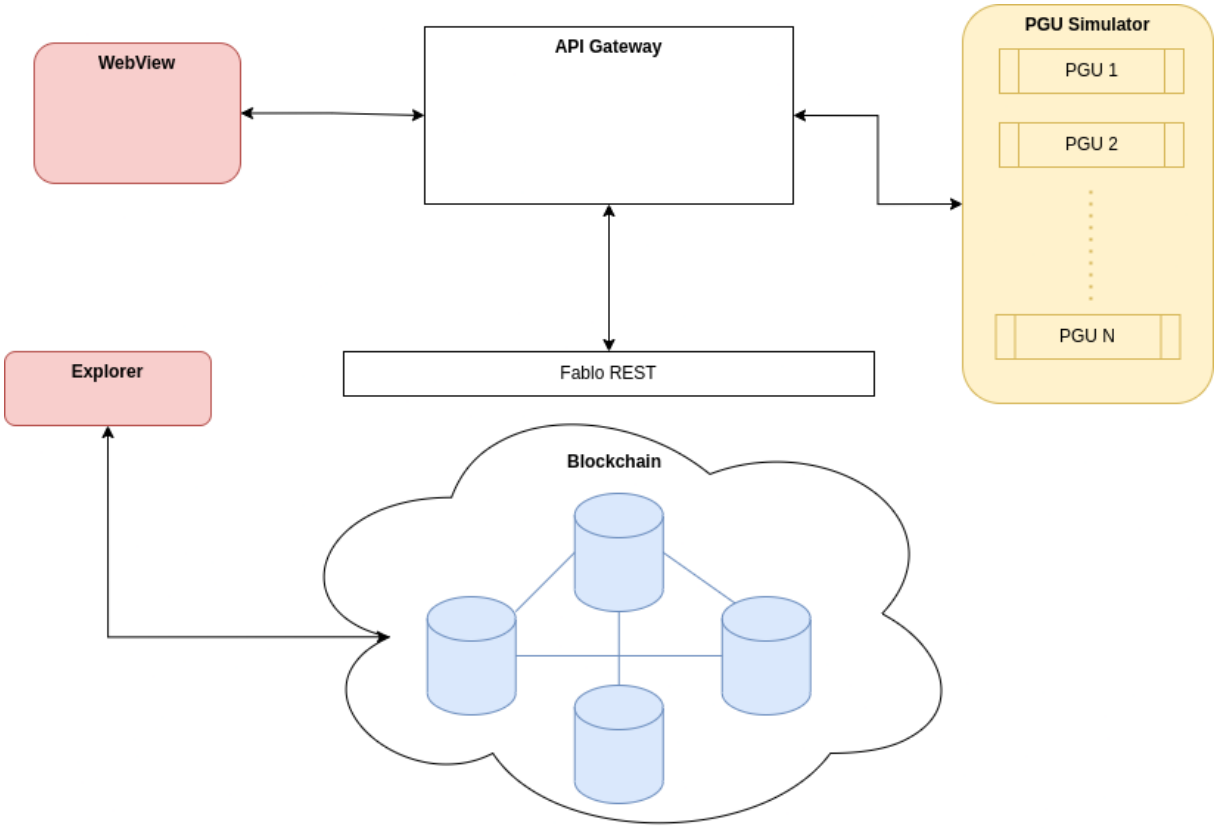


Figure 5.2: Hyperledger functional architecture

5.2.2 Data Model

In this section, the main data models are given. We find four models: PGU, Measure, Infraction, and Constraint.

PGU

a *PGU* is an electrical energy production unit. It is characterized by the following fields:

Field	Type	Description
id	number	unique number that identifies the unit
statusId	number	determines its current state and the actions possible
sourceTypeId	number	production source used by the unit
owner	string	name of the owner of the unit
installedPower	number	value of the installed power of the generation unit
contractPower	number	value of the contractual power of the generation unit

Table 5.1: PGU data model

A specific mapping has been implemented fo the status and source type ids. Those are shown in **Table** 5.2 and 5.3.

Status Id	Status
0	created
1	running
2	alerted
2	in urgency

Table 5.2: Status id mapping

Source type id	Source
0	wind
1	solar

Table 5.3: Source type id mapping

Measure

a *measure* of the energy production. It represents the source data for monitoring and control mechanisms.

Field	Type	Description
id	number	unique number that identifies the unit producing
producedPower	number	value of the power generated for the measurement
time	date	time at which the measurement was taken

Table 5.4: Measure data model

Infraction

Infraction attributed when the rules imposed by the DSO are not respected.

Field	Type	Description
infractionType	string	type of infraction
count	number	number of current infractions
time	date	time at which the infraction was registered

Table 5.5: Infraction data model

Constraint

It represents the constraint that is imposed on a production unit.

Field	Type	Description
id	number	unique number that identifies the unit constrained
limitPower	number	value of the power limit for the production
time	date	time at which the constraint was issued

Table 5.6: Constraint data model

5.2.3 Blockchain

In this section, the blockchain component is described. First, an illustration of the architecture and implementation choices. Then, the smart contracts system is explained.

Blockchain component architecture

Figure 5.3 shows the proposed architecture hosting a single energy production unit. It represents the basic unit to deploy when managing distributed energy resources. In this architecture, each owner has its own channel for transmitting its production. In this prototype, the addition of multiple owners and thus the management of multiple channels is not handled for simplicity and focus on the main functions of the prototype.

This architecture can be divided in three parts: ORES nodes, grid nodes and a monitor interface. Let us explain each one of them and which Hyperledger Fabric components can be found inside them.

ORES nodes represents the nodes running on the DSO premises. They enable one to feed the blockchain with control actions and constraints. As an important stakeholder, the DSO runs a full node (Orderer, Peer and Client) which allows it to store the ledger and be part of the consensus mechanism in several channels. Confidentiality is achieved within a DSO-owner channel, where contractual data stored in ledgers are enforced by

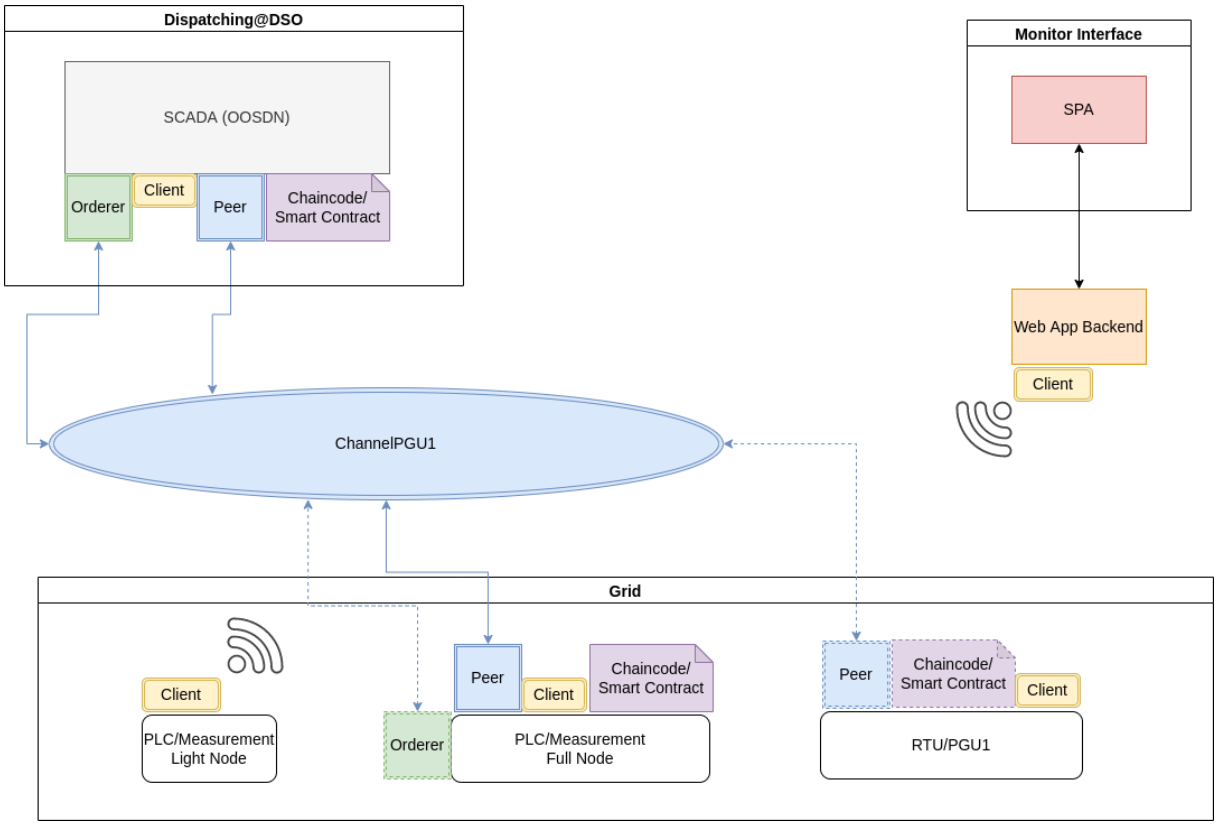


Figure 5.3: MonitORES Hyperledger blockchain architecture

its immutability and reliability.

Secondly, grid nodes represent all nodes physically in the grid. A distinction is made between three elements: light nodes, full nodes and production nodes. Light nodes are client applications that interact with the ledger but do not store it. Their role is to submit grid updates from different points in the grid. On the other hand, full nodes fully participate in the blockchain by hosting a peer and orderer. They are to be hosted in a substation, ensuring one or multiple production channels. Production nodes correspond to Full Nodes on specific substation attached to a energy producer. They differ from normal full nodes on their channel connection: they are only connected to their dedicated channel and do not necessarily possess an orderer node.

Finally, the monitor interface is a web application enabling the monitoring of data within the ledger, as well as blockchain status information. It contains a Hyperledger Fabric client in order to query the ledger, and a web application stack. It provides three different views: supervisor, grid monitor and producer. Supervisor view displays all technical blockchain network data. The grid monitor view shows global information about constraints and infractions associated with all energy producers. This eases the work of monitoring the grid for the DSO monitoring staff. Producer view, allows an

energy producer to see all data related to its channel, such as production or reached levels of infractions.

This architecture is instantiated through the Fablo tool. Fablo is an Hyperledger Fabric network generator using Docker in locally. One needs to specify the configuration of the network in a `fablo-config.json` file and it generates all containers and material for a local test of Fabric. It comes with out-of-the-box with a REST API (Fablo REST [50]) to ease queries to the blockchain. It can be also linked to Hyperledger Explorer.

The use of such tool facilitates the development of a proof-of-concept prototype. However, it comes with several limitations being discussed later.

Smart Contracts

We focus in this section on presenting the smart contract logic supporting the ANM monitoring actions.

Figure 5.4 outlines how chaincode is deployed and used in both production channels. A MonitorPGU contract models interactions with the production channel. Let us now discuss how this contract ensures the functions discussed in the functional analysis.

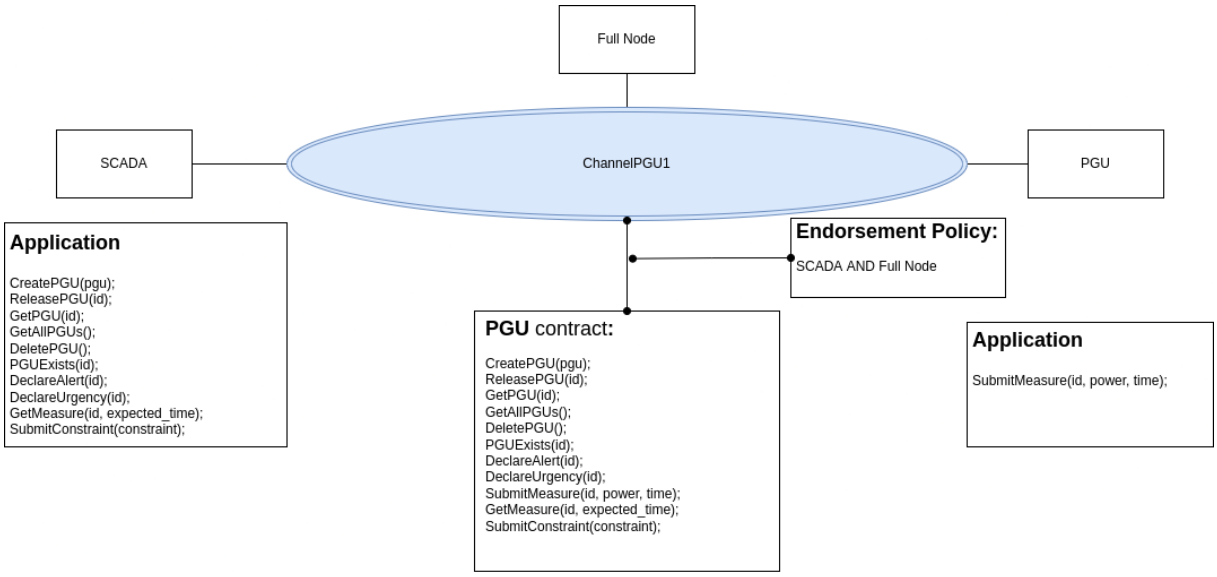


Figure 5.4: Chaincode interaction

On-boarding

As PGUs join the grid, we need to be able to create a PGU state inside the chain that will define the PGU characteristics. As shown in **Figure** 5.4, we have inside the PGU contract a function `CreatePGU` that can be settled when establishing a new connection contract with a PGU's owner. This function initializes the status of the PGU to 0 (created) and takes as argument all contractual information and id of the PGU.

If access policies defined on the Fabric network may change over time (e.g. integrate the Transport System Operator to a channel or change writing permissions of a full node). This is achieved by system chaincode. They provide the transaction logic to update access policies.

Monitoring and control

Monitoring and control are two main functions tied together as one triggers the other and vice-versa. Depending on values monitored control actions are triggered, and once they are applied they must be monitored to be sure they are effective. In the PGU smart contract, four functions ensure this.

First, we have the functions modifying the status of the PGU, `DeclareAlert` and `DeclareUrgency`.

Finally, `SubmitMeasure` and `GetMeasure` provide the methods to manage measures. The first one is used by producers to submit a measure, which will go through a validation procedure. The second one, will be called regularly to check if measures have been correctly submitted. They both receive in input current times as we cannot use "now" inside smart contracts because it would make the code non deterministic. Powers are received in kW.

`SubmitConstraint` provides the procedure to update the power limit to which the PGU will be constrained. It applies five minutes after it was submitted in order to let the PGU adapt to the situation and is only valid during five minutes.

Visualization

In order to visualize current state of the ledger, getters are implemented on the PGU model: `GetPGU` and `GetAllPGUs`. This will return all information about the PGU(s)

requested if it or they exist.

REST API

To facilitate the use of the Fabric blockchain, Fablo can be extended with Fablo REST. This latter tool provides useful REST API endpoints as described in [50]. In this case, the following endpoints were used:

- enroll: enrolls the user, giving it a token to access other functionalities
- reenroll: if a user is enrolled, it can be re-enrolled to extend its sessions
- invoke: invokes smart contract functions that need to modify the current state of the ledger, e.g. `SubmitMeasure`.
- query: invokes smart contract functions that only need to read information from the systmem, e.g. `GetPGU`.

Explorer

A supervisor view can be implemented with existing tools (e.g. Hyperledger Explorer [51]). It consists of a dashboard with all blockchain technical information, such as the number of transactions or blocks. It also provides channel and chaincode visualisation. Authentication and authorization to this interface is managed inside of it.

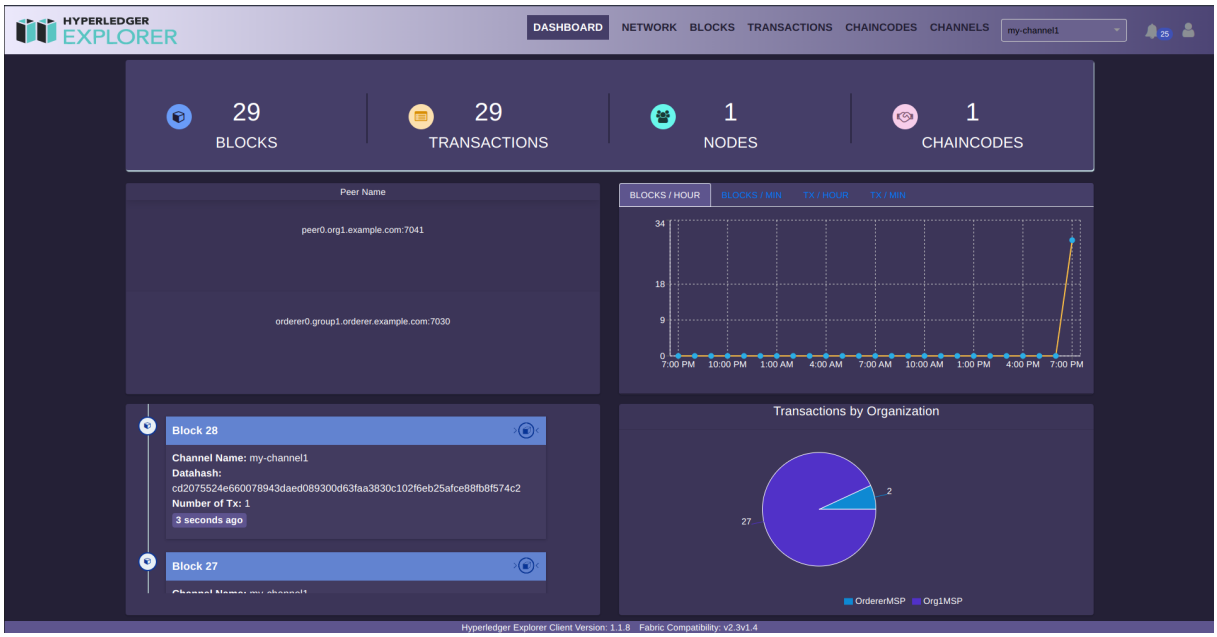


Figure 5.5: Explorer tool

5.2.4 API gateway

An API gateway sits between external clients and the applications running in your data-center and clouds. A Application Programming Interface (API) is a way for software to communicate with each other, without the users having to know where or how it is implemented. This can be done through programmatic interfaces, which are used by developers to create and interact with APIs. These interfaces are created following open standards, so that different applications can interact with each other. As a result, information can be shared between applications and across different software platforms. An API gateway validates incoming requests, dispatch them to the appropriate service based on defined rules, and then returns the appropriate result to the client.

This component is implemented as a NestJS service communicating with all the components, facilitating communication between them. In fact, the prototype uses multiple protocols: HTTP, Kafka, and WebSocket. In **Figure** 5.6, the different communication channels are represented. Having an API Gateway exposes a simpler communication for the different parts of the system. Let us now explain each communication link and the protocol used.

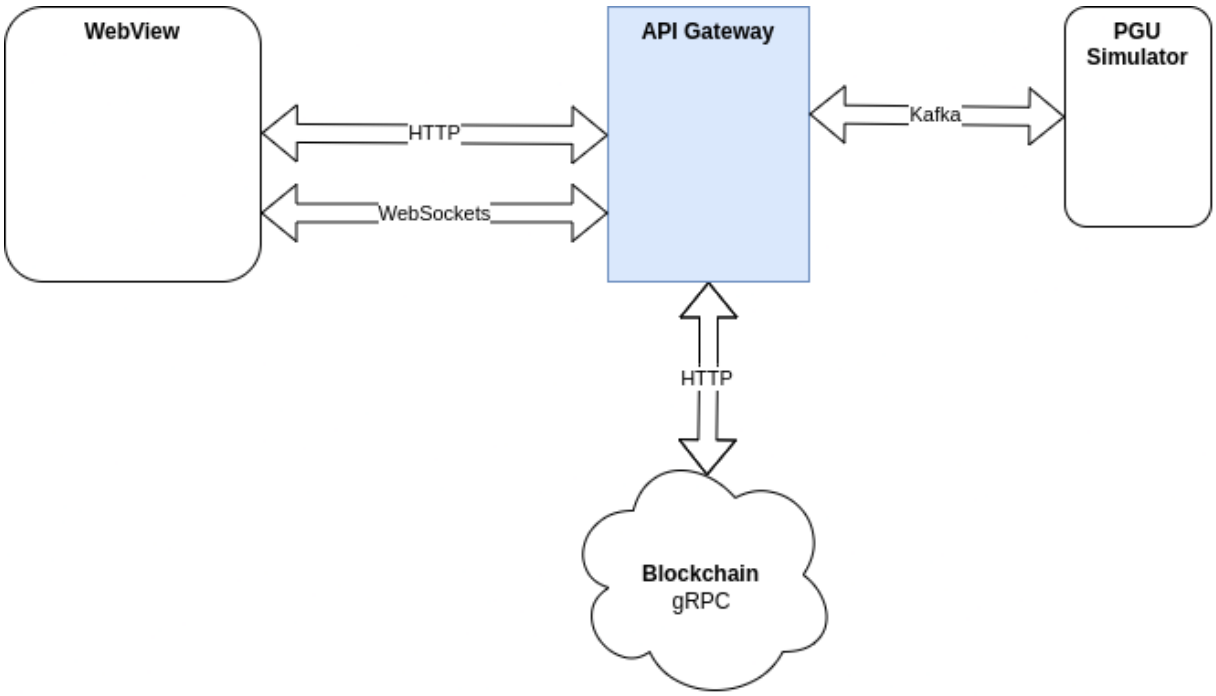


Figure 5.6: API gateway and communication protocols

First, the web interface is connected to the gateway through WebSocket and HTTP. As the web interface needs live data from the network, WebSocket is used to stream production and monitoring data for a given PGU. However, for all other endpoints,

HTTP was used, as it consist on simple requests to the API gateway that manage the translation to Blockchain and Hyperledger terms.

As mentioned in the previous paragraph, the API Gateway is in charge of transmitting requests to the blockchain component. It stores all information about smart contracts invocation, and uses HTTP to communicate to Fablo REST.

Finally, it listens to data from producers (i.e. the PGU simulator) through a Kafka channel. It also uses this channel to trigger control on the PGU Simulation service. This Kafka module is provided by NestJS out-of-the-box and consist in a way to stream data as in WebSockets with a Pub/Sub pattern [52].

To simplify the model only one API gateway was instantiated. However, one can split the gateway into two parts : one in charge of the web interface communication and the other responsible for connecting the PGU production to the system. With both of them communicating through the blockchain.

5.2.5 PGU simulator

This component is responsible for the simulation of PGUs. It has three main parts: creation of the production process, computing the power based on weather or historical data, and emitting the measure value.

The creation process is triggered by the API gateway with a Kafka creation event containing all the information about new PGU being created. Then the PGU service initializes a cron job which executes depending on the type of energy source, the power output calculations at regular specified intervals. A cron job or cron schedule "is a specific set of execution instructions specifying day, time and command to execute"[53]. In other words, it is a tool for task scheduling. This prototype uses the NestJS schedule library, which facilitates the use of such tool.

Generators model

Only two type of producers are taken into account in this work, wind generators and solar panels. Wind generation can be simulated from real-time weather data, or from historical data, whereas solar is only implemented using weather data. Let us now how each type was implemented and their dependence on a weather API.

Wind generators

For wind energy producers, a simplified wind turbine power curve is used. This power function is characterized by 4 zones illustrated in **Figure** 5.7.

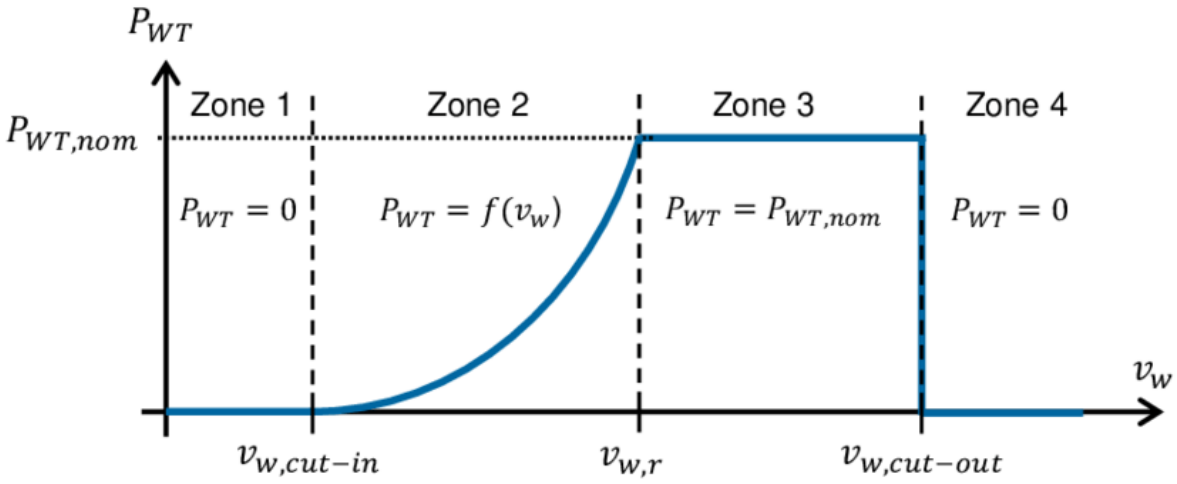


Figure 5.7: Simplified wind power curve [54]

Where the power function $f(\nu_W)$ in zone 2 used in this simulator is shown in **Equation** 5.1.

$$f(\nu_W) = \frac{1}{2}\rho(p,T)Ac_p(\nu_W)\nu_W^3\eta_W \tag{5.1}$$

Where

$$\rho(p,T) = \frac{pM}{RT} \tag{5.2}$$

$$c_p(\nu_W) = \begin{cases} 0.1\ ,\ \nu_W < 3 \\ 0.45\ ,\ 3 \leq \nu_W < 13 \\ 0.15\ ,\ \nu_W \geq 13 \end{cases} \tag{5.3}$$

All parameters are described in **Table** 5.7. For each parameter, the default value is shown. However the wind speed, temperature and pressure values come from an external API given these weather information in real-time. This API is the current weather API from OpenWeather [55]. By specifying a location with longitude and latitude, the API sends the current weather values for that specific place.

Parameter	description	value	variable name
$\nu_{W,cut-in}$	cut in wind speed	1 m/s	`Vcutin`
$\nu_{W,cut-off}$	cut off wind speed	25 m/s	`Vcutoff`
$\nu_{W,r}$	nominal wind speed	17 m/s	`Vnom`
ν_W	current wind speed	value from API in m/s	`windSpeed`
P_{WT}	current power output	given by 5.1	`powerOutput`
$P_{WT,nom}$	nominal power output	2000 in kW	`Pnom`
ρ	air density	given by 5.2	`rho`
p	air pressure	value from API in hPa	`pressure`
R	gas constant	8.3145 J/Kmol	`R`
M	dry air molar mass	0.029 Kg/mol	`AirMolarMass`
T	air temperature	value from API in K	`temperature`
A	windmill surface	5281 m^2	`A`
c_p	power coefficient	given by 5.2.5	`Cp`
η_W	wind turbine efficiency	0.8	`WindEfficiency`

Table 5.7: Wind power parameters

Solar generators

Solar generators follows the same principle as in wind generators. Variable weather data is gathered in real-time from the OpenWeather solar radiation API. In this case, the model is simpler as **Equation** 5.4 shows.

$$P_{sol} = AI_{GHI}PR\eta_{sol} \tag{5.4}$$

Parameter	description	value	variable name
A	cut in wind speed	1 m/s	`Vcutin`
PR	cut off wind speed	25 m/s	`Vcutoff`
η_{sol}	nominal wind speed	17 m/s	`Vnom`
I_{GHI}	current wind speed	value from API in m/s	`windSpeed`
P_{sol}	current power output	given by 5.1	`powerOutput`

Table 5.8: Solar power parameters

Weather API

The PGU simulator relies on a unique service called the weather service which is responsible for pulling real-time weather data for the PGU power computations. Two main functions, one for wind data and the other one for solar data. In the wind function called `getWindWeatherData`, the current wind speed, temperature and pressure of a precise location is pulled. In `getSolarWeatherData`, only one value is asked, the current Global Horizontal Irradiance (GHI) of a specific location. Longitude and latitude are currently set to 3.11 and 51.36 respectively, which corresponds to a location in the

belgian sea. Just like all the other constants and values, the choice of these parameters were arbitrary done based on typical values as the focus of this work is not on perfectly simulating generators. The goal of the simulator is to generate data to control and include in the blockchain.

Historical data

In order to test the prototype with real production data, a function `getHistoricalData` is implemented. This function extracts wind production data from the past. The data used comes from Elia grid data web application [49]. In this work, only was used the onshore wind measured generation directly connected to DSOs. The selected period was from august 7th to august 10th. The production measured were then down scale to the order of kW. The measured are based on a 15 minutes frequency. However, to ease testing and visualisation, the measures are given at a minute rate. The data is repeated when the end of the measures are reached.

Emitting measures and constraint management

In order to manage constraint, simulators check for constraint in the system before submitting a measure. Then depending on the PGU status and respectfulness, they adapt their output to the constraint. Non-respectful PGUs have a probability of 0.1 not respecting the constraint.

The emitting procedure consists on emitting a Kafka event on a the 'pgu_measures' topic, which will be handled by the API gateway to submit measures to the blockchain.

5.2.6 Web-view

The last component to describe is the Web-view displaying the PGU's information. This component can be broken down to three elements: Authentication, Dashboard and Settings. Let us now go through each one of the elements and focus on the dashboard which let the user analyse critical data about what is happening on the electrical network.

Authentication

Concerning the authentication of users in the interface. It is a token based authentication coming directly from the blockchain. This token are issued by Fabric CA according to the MSP and can be retrieve through the enroll API of Fablo REST. Fablo does not allow

yet lot of options with authentication API, therefore parameters such as token live time were not modified and set at default, i.e. 10 min.

Dashboard

Let us now describe the main component of the web-view, which is the dashboard. It represents all relevant information to understand the state of the network and thus the state of PGUs. The dashboard is composed of five sub-components: PGU list, PGU details, PGU constraint, infraction list and energy production.

- **PGU list:** In this component, all PGUs available in the channel are listed and can be selected to change the information displayed in the dashboard.
- **PGU detail:** This component displays all details about a PGU but it can also be used as a form to create a new PGU.
- **PGU constraint:** In this component, the user can submit a constraint to the PGU, declare Alert Mode or Urgency Mode. The last constraint declared is also displayed.
- **Infraction list:** This component contains a table displaying the infraction count per type of infraction, as well as the last time an infraction was made.
- **Energy production:** This component display the energy production of a PGU in a real-time chart. It is implemented using Echarts [56]. Constraints are displayed with a red line and appears 5 minutes before they are applied and stay during their live time. PGU status is also reflected in the colour of the line. Blue line means that PGU is on boarding or running, yellow represents Alert Mode and red is used for Urgency Mode.

5.2.7 Testing

To test our software, the idea is to reproduce the whole life-cycle of a PGU joining ORES grid and watching the behaviour of the system. Each functional requirement needed to be fulfilled. All life-cycle screenshots can be found in **Appendix** B

PGU creation

PGU creation starts at the web interface page for PGU creation as can be shown in **Figure** B.1. When the from is filled the submit button enables and it is possible to create the PGU.

Once the PGU is created, it can be observed that the on-boarding process takes place. A percentage indicator shows the progress of the on-boarding tests. The status correctly displays the on-boarding status.

PGU monitoring and control

When the PGU is allowed to run after passing all on-boarding tests, one can monitor the evolution of production. Three tests are then applied. First, a specific constraint is applied from the interface. This constraint is then displayed in the next minute and to be applied in the next 5 minutes, see **Figure** B.2. Secondly, Alert Mode is declared. The PGU status and the color of the graph are updated, see **Figure** B.2. Finally, to test the infraction system we disconnect the PGU by killing the simulator process in charge of it. Minors infractions start to trigger since there is no measure sent, leading to major and critical infractions. Those infractions change the stats of the PGU correctly and trigger Urgency Mode, see **Figure** B.2.

5.2.8 Bugs, Limitations and improvements

After presenting the software developed, let us discussed about the limitations and drawbacks of the current system, to finally present some hints on the improvements to make.

PGU id management

Currently PGU id is managed through a simple integer counter, being incremented each time a PGU is created. To avoid collisions and trivial IDs, GUID could be used instead. Globally Unique Identifier (GUID) are 128-bit text string to uniquely identify objects and people. They are also known as UUIDs (Universally Unique Identifiers).One of the main benefits of using GUIDs is that they are globally unique. This means that they can be generated anywhere in the world, and they will always be associated with a particular object or person no matter what part of the world they are in. This makes GUIDs useful in many different applications. For example, a software company could use them to identify different users of their software. Another useful property of GUIDs is that they can be used to communicate with computers even when the people involved don't know each other's language.

Blockchain error handling and concurrency

Currently Hyperledger Fabric manages DB requests with MVCC (Multiversion concurrency control) meaning that it uses a lock-free optimistic concurrency, with rollback in case of dirty read/writes. Key collisions needs to be avoided as much as possible. Therefore, implementing logic on client or microservices side should help avoiding these kind of errors.

Crash tolerance of microservices

When one of the NestJS microservices (API gateway or PGU simulator) crashes it does not recover their previous process. Therefore, when the services is restarted, it looses all cron jobs that where running and represented the production and measuring inside the system.

Solar production API

Tests and screenshots of the applications were made using only wind generators as PGUs as the Openweather API for solar production is a paying service and no free API was found at the time of testing. To further analyse the behaviour with solar test, a paying option should be taken or historical solar data should be connected to the simulation as for the wind generation.

User management

As Fablo offers limited access to the permission management system. No user implementation could be achieved. The settings component in the web-view wll host this functionality when implemented.

Data mining and out-boarding

Two of the main services described in **Section** 5.1.2, data mining and out-boarding, were not directly addressed by this prototype. In fact, those were not essential for the research question being answered and required. Those could be addressed in further development of the prototype.

Deployment and real physical infrastructure

The current proof of concept provides a simulation environment to demonstrate how a basic configuration could be achieved for a DSO. However, as all the software was tested running on a single computer, scaling the prototype to a real physical environment could reveal some other aspects not currently taken into account such as real network security management. It might be interesting to dispatch all nodes into different micro-controllers such as Raspberry PI. Currently, Fablo do not support cloud or production deployment, but they stated that they are working on this improvement. Other tools in the Hyperledger ecosystem such as Bevel also could address those questions. Otherwise, one could directly configure and deploy a Fabric network. However, it comes with the complexity of managing all details from this framework. At first, some tries were achieved to manually deploy, but could not be achieved in the required time of the thesis.

5.3 Results Discussion

In this section, the analysis of the produced software and model is presented. Now that the implementation has been reviewed, let us discussed about the model's security and how it performs in terms of communication and compliance, which are the key points to achieve the goals of this thesis.

5.3.1 Security

To assess the security of the prototype system proposed, we take into account three type of attacks: data collection, data transmission and data storage.

Data collection is when the attack happens at the edge device measuring the data. Then, data transmission is when the attack occurs in the communication channel between the device and the storage system. Finally, data storage is when the attack targets directly the storage of data.

To tackle this analysis, a probabilistic approach is taken, inspired from [17]. They assess their blockchain framework by modelling the probability of successful cyber-attacks in two scenarios. A probability of the system being hacked is attributed to the current scenario (normal scenario) and another probability to the scenario with a communication system based on blockchain. For each scenario and attack type, let us model the probability for the system being compromised where n is the number of nodes needed to achieve a successful attack, and N the number of nodes involved in the system. The three different attack types possess independent probabilities.

Normal scenario

In the normal scenario, we attribute to each node a probability of being hacked α_i and a probability to each connection with the central system β_i where $i = 1, 2, ..., N$. Then, a probability η for the central system of being compromised. Then, the probability of achieving a successful attack is given by **Equation** 5.3.1.

$$P_{normal} = \frac{1}{3}\left(\prod_{i=0}^{n}\alpha_i + \prod_{i=0}^{n}\beta_i + \eta\right) \tag{5.5}$$

Blockchain scenario

In the blockchain scenario, we attribute to each node a probability of being hacked $\bar{\alpha_i}$ and each connection with the central system $\bar{\beta_i}$. Then a probability ω_i for each key which encodes the communication of each node. Here, another variable has to be introduced: τ. τ is the voting threshold after which an attacker can control the consensus system. Therefore to compute the probability of the communication attacks we need to define K, which represents how much channels the attacker needs to hack $K = ceil(\tau\frac{N(N-1)}{2})$. Finally, to control the state of the blockchain the attacker needs to take control over consensus, this means hacking into M devices where $M = ceil(\tau N)$. Then the probability of achieving a successful attack is given by **Equation** 5.3.1.

$$\begin{aligned} P_{blockchain} &= \frac{1}{3}\left(\prod_{i=0}^{n}\bar{\alpha_i}\prod_{i=0}^{n}\bar{\omega_i} + \prod_{i=0}^{K}\bar{\beta_i}\prod_{i=0}^{n}\bar{\omega_i} + \prod_{i=0}^{M}\bar{\alpha_i}\prod_{i=0}^{n}\bar{\omega_i}\right) \\ &= \left(\prod_{i=0}^{t}\bar{\alpha_i} + \prod_{i=0}^{K}\bar{\beta_i} + \prod_{i=0}^{M}\bar{\alpha_i}\right)\frac{1}{3}\prod_{i=0}^{n}\bar{\omega_i} \end{aligned} \tag{5.6}$$

Let us now run a Monte Carlo simulation to approximate both scenarios probabilities and compare them. To run the Monte Carlo simulation, N was set to 10 and 1000 simulations were run. α_i, β_i, $\bar{\alpha_i}$, $\bar{\beta_i}$ and $\bar{\omega_i}$ are to be drawn uniformly from [0.9, 1]. η is to be set in the range of [0, 0.1]. τ is set to 51% which represents the lowest threshold for consensus. Therefore, the value of K is 23 and M is equal to 6. Results are shown in **Figure** 5.8.

As it can be observed, probability decreases as the number of devices needed to be hacked increases. This could be expected as a successful attacks implies hacking into more edge systems. However, the probability in the blockchain scenario decreases faster than the one in the normal scenario. This can be explained by the distributed factor, which implies hacking into all devices keys in order to modify the data storage or communication in the system. In the central system model, attacking can be focused on less communications channels and a central database.

5.3.2 Communication and Compliance

One of the main objectives of our prototype is to used blockchain as communication layer to improve contractual producers compliance monitoring.

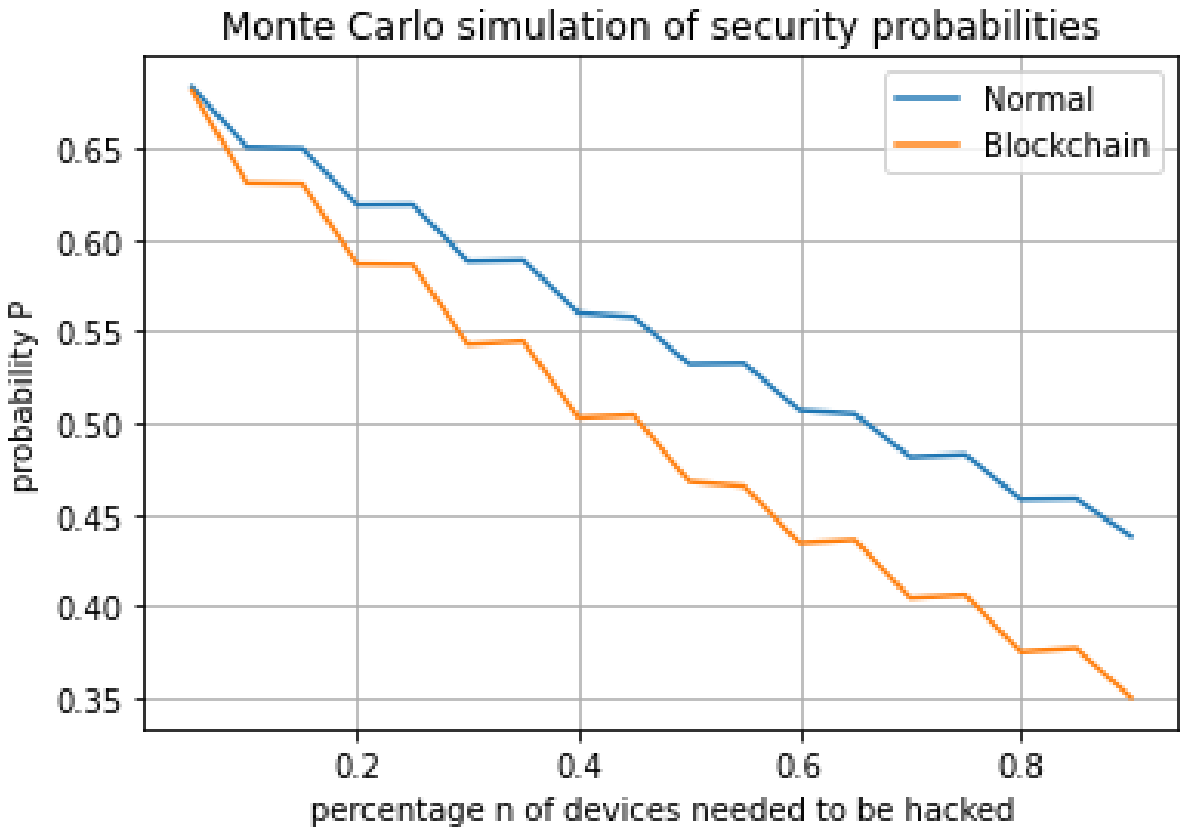


Figure 5.8: Successful attacks probabilities of normal and blockchain scenario based on Monte Carlo simulation

From **Section** 5.2.7, it can be observed that messages pass clearly between components, and monitoring and control are achieved through a blockchain platform. PGUs are directly penalized in the smart contract as they do not respect constraint or do not send measures. Furthermore, an immutable log records all these interactions.

Chapter 6

Conclusion

In this chapter, the conclusion of the thesis is presented along with an outlook and further words for what would be next.

This thesis aimed to demonstrate that a Distributed Ledger Technology (DLT) such as blockchain can be used to monitor contractual requirements of the generation units within the framework of ANM scheme. Based on a proof of concept prototype called "MonitORES" implemented and described in this work, the use case was tested and analysed according to historical and live data displayed in a real time manner. To achieve the main goal, five objectives were fulfilled. First, the framework used to develop the prototype was chosen after a benchmark between current permissioned blockchain platforms, leading to Hyperledger Fabric. In fact, as it offers a higher modularity in terms of scalability and permission control, it suited well our use case. Secondly, a functional analysis was given, explaining what are the functions of "MonitORES". Then, an architecture of a blockchain based system is proposed and implemented. Finally, limitations and a result analysis was presented.

Results show that in terms of security and compliance, a blockchain based system can reduce the probability of attacks and non respects of rules dictated by the DSO. Therefore, a blockchain based can be implemented to increase reliability, security and compliance in decentralised production units in the ORES distribution grid.

While our prototype clearly illustrated the feasibility of a blockchain monitoring system in DSO grids, it raises the question of how well it can be physically implemented and scaled to real world scenarios. The method used to select the blockchain framework helped us to decide, however the blockchain eco system evolved faster each day, so one could reconsider the current criteria.

To better understand the implications of a blockchain system, further studies on physical feasibility and effective cost could be addressed. As this work intentionally focused only on blockchain, it could be interesting to compare our work to other information systems that could addressed the same criteria as we did and how it last in time.

To conclude, this thesis demonstrates how a blockchain system can be used to monitor decentralised production units. In this work, an architecture and limited implementation is proposed and could be the baseline and first step to the exploration of a real word grid monitoring based on blockchain.

Appendix A

Source code

All source code was managed using a private git repository in Azure DevOps and is provided in a compressed file `master-thesis-monitores.tar.xz`. Inside of it, four folders can be found: `monitores-app` for the Angular web app, `monitores-dlt` for the Fablo configuration and chaincode, `monitores-api` for the NestJs API gateway and `monitores-pgu-simulator` for the PGU simulator. A python notebook **monitores-montecarlo** can also be found, where all code related to the Monte Carlo simulation resides.

A.1 Running MonitORES

To run the developed software, a linux environment is needed where Fablo, NestJs and Angular are installed. An additional tool called Kafdrop (`https://github.com/obsidiandynamics/kafdrop`) is needed to ensure the communication between microservices. Then for each component run the following command in this order:

Fablo

```
sudo fablo up
```

Kafdrop

```
cd docker-compose/kafka-kafdrop
docker-compose up
```

NestJs micorservices

```
sudo npm run:start
```

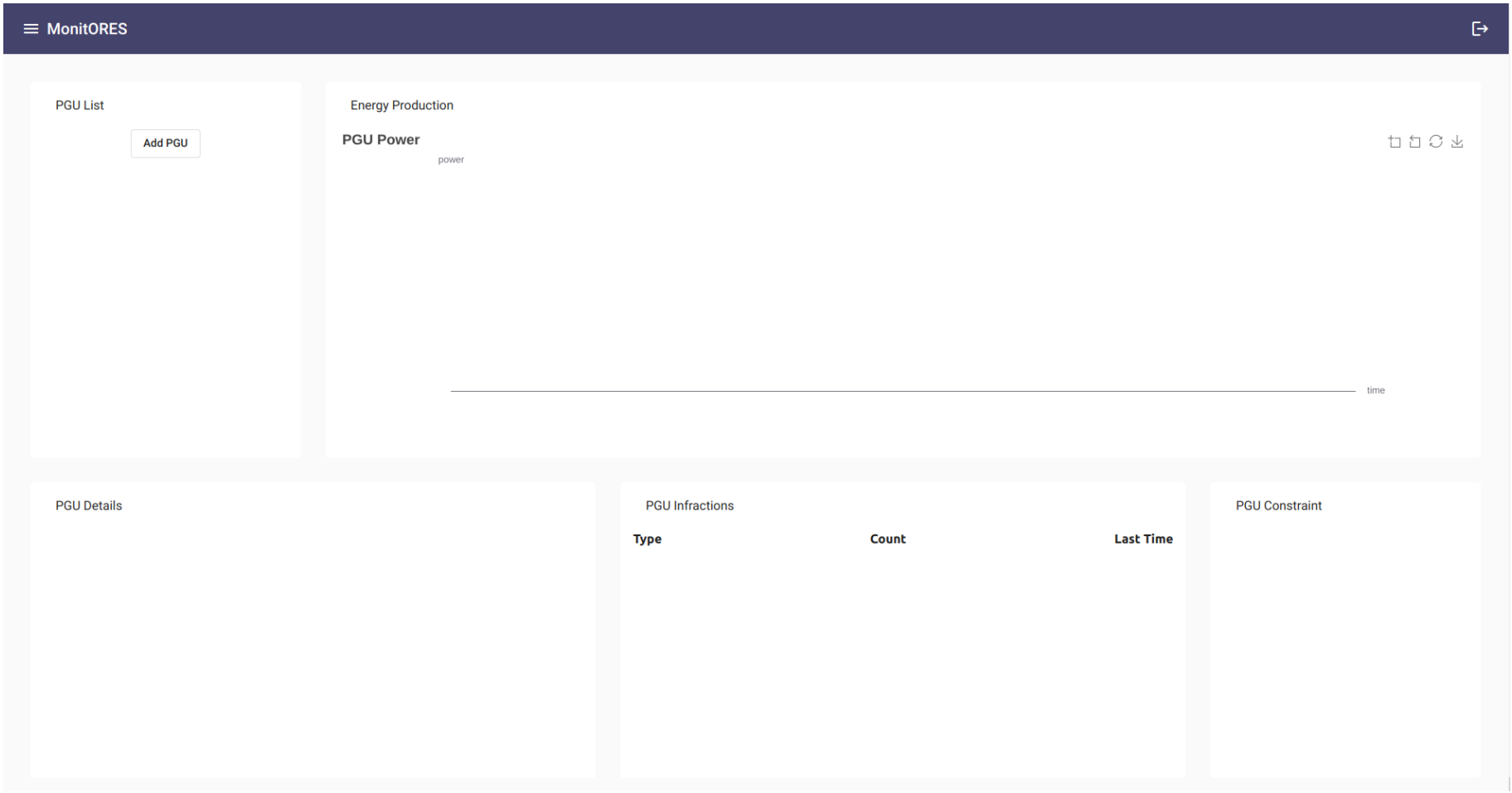
Angular web-view

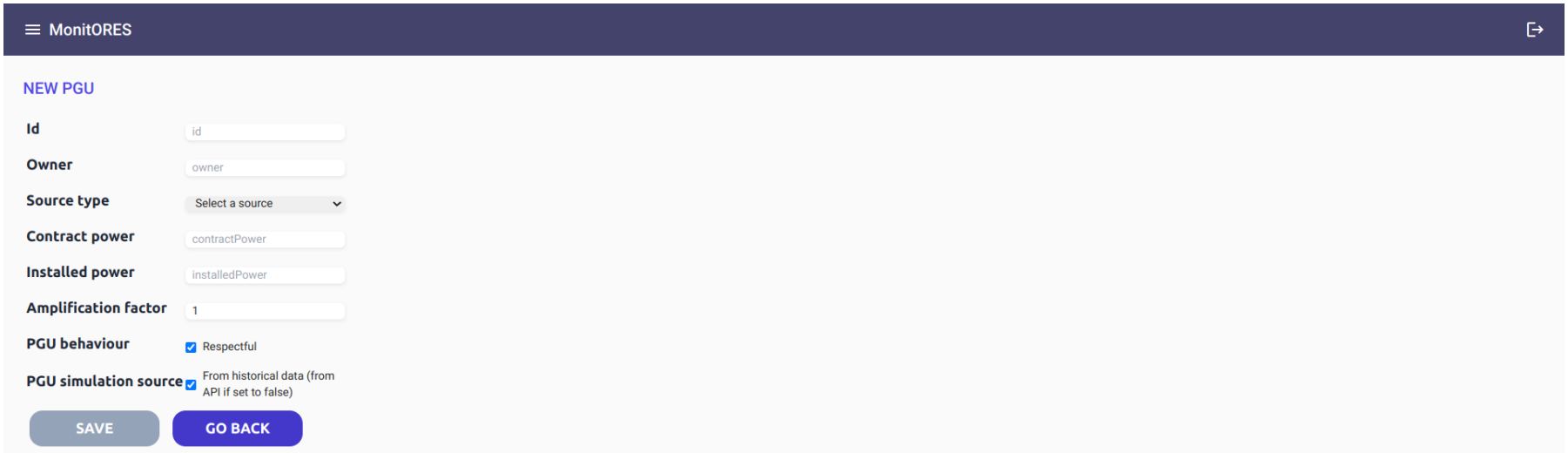
```
ng serve
```

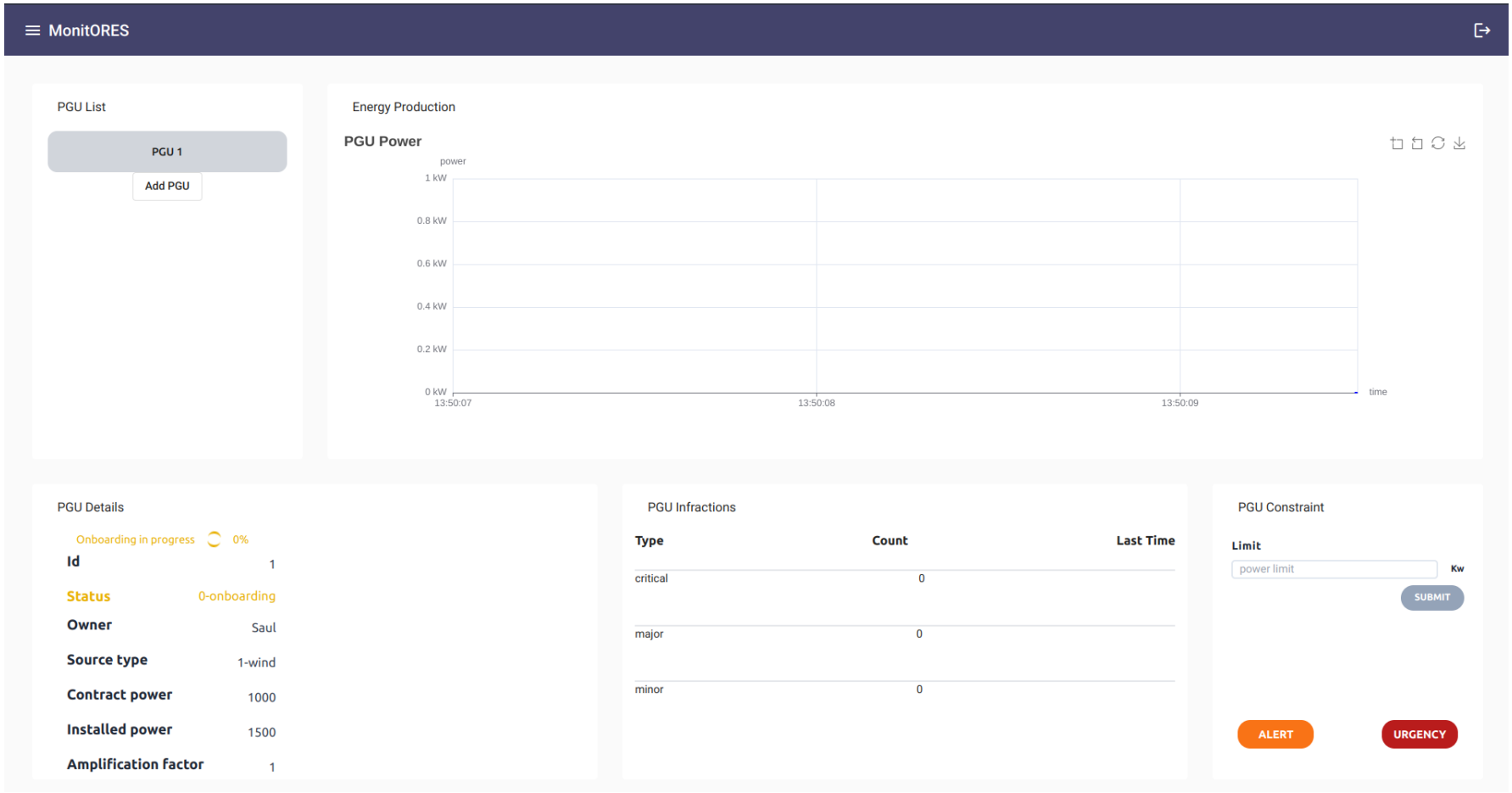
Appendix B

Application screenshots

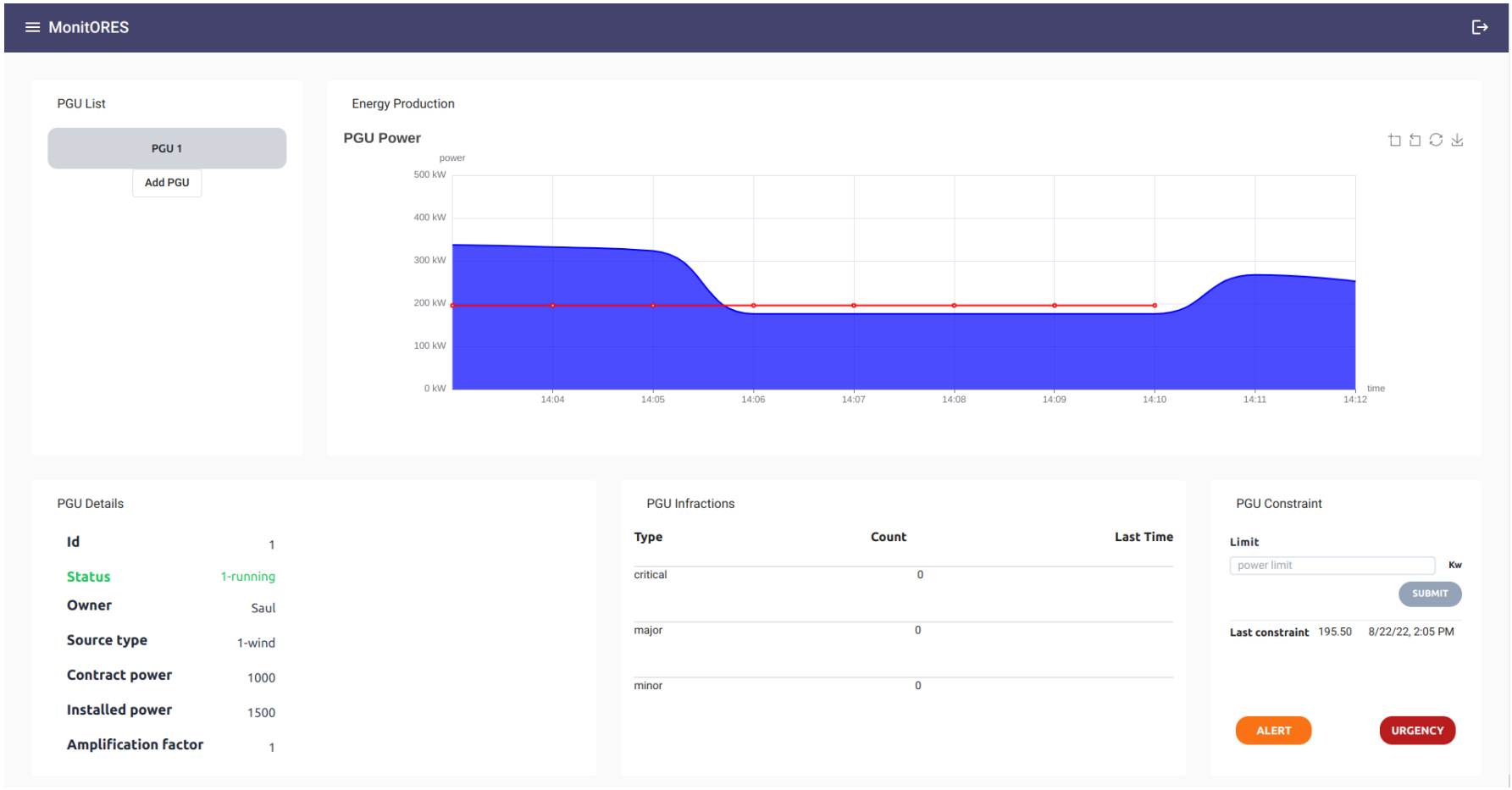
B.1 PGU creation

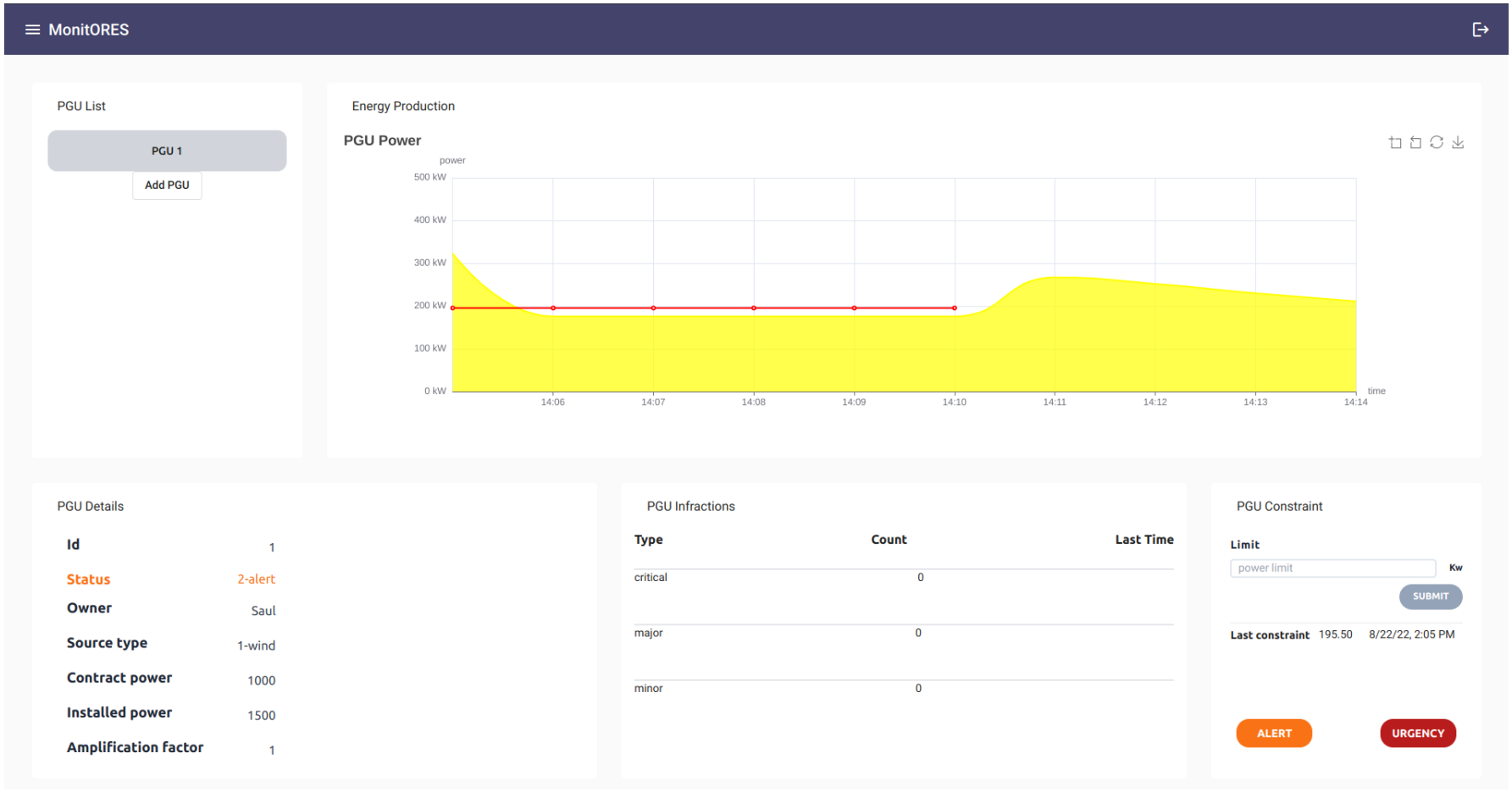

MonitORES
PGU List
Add PGU
Energy Production
PGU Power
power
time
PGU Details
PGU Infractions
Type
Count
Last Time
PGU Constraint

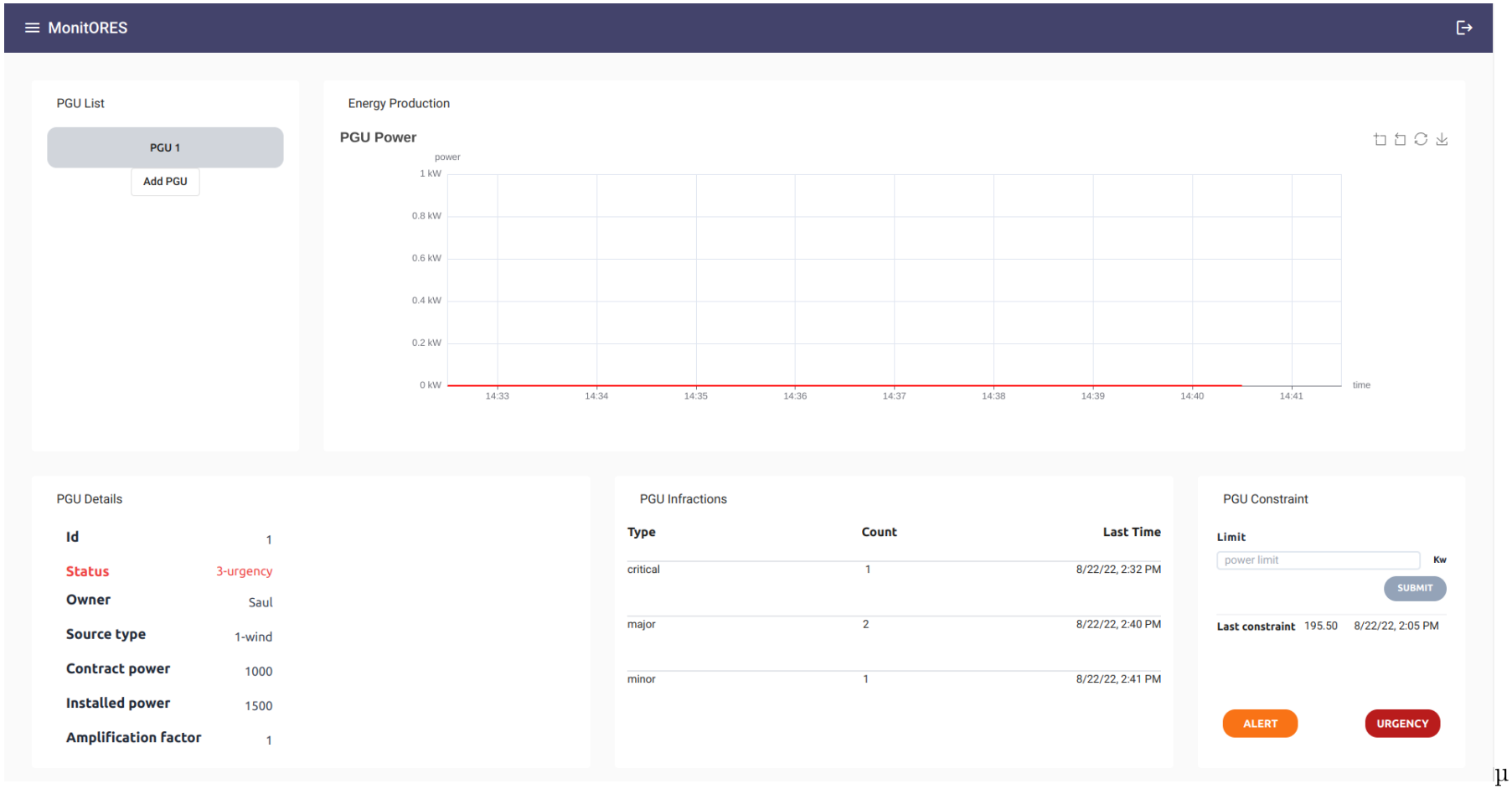

MonitORES
NEW PGU
Id
id
Owner
owner
Source type
Select a source
Contract power
contractPower
Installed power
installedPower
Amplification factor
1
PGU behaviour
Respectful
PGU simulation source
From historical data (from API if set to false)
SAVE
GO BACK


MonitORES
PGU List
PGU 1
Add PGU
Energy Production
PGU Power
power
1 kW
0.8 kW
0.6 kW
0.4 kW
0.2 kW
0 kW
13:50:07
13:50:08
13:50:09
time
PGU Details
Onboarding in progress
0%
Id 1
Status 0-onboarding
Owner Saul
Source type 1-wind
Contract power 1000
Installed power 1500
Amplification factor 1
PGU Infractions
Type Count Last Time
critical 0
major 0
minor 0
PGU Constraint
Limit
power limit
Kw
SUBMIT
ALERT
URGENCY

B.2 PGU monitoring and control


MonitORES
PGU List
PGU 1
Add PGU
Energy Production
PGU Power
power
500 kW
400 kW
300 kW
200 kW
100 kW
0 kW
14:04
14:05
14:06
14:07
14:08
14:09
14:10
14:11
14:12
time
PGU Details
Id 1
Status 1-running
Owner Saul
Source type 1-wind
Contract power 1000
Installed power 1500
Amplification factor 1
PGU Infractions
Type Count Last Time
critical 0
major 0
minor 0
PGU Constraint
Limit
power limit
Kw
SUBMIT
Last constraint 195.50 8/22/22, 2:05 PM
ALERT
URGENCY


MonitORES
PGU List
PGU 1
Add PGU
Energy Production
PGU Power
power
500 kW
400 kW
300 kW
200 kW
100 kW
0 kW
14:06
14:07
14:08
14:09
14:10
14:11
14:12
14:13
14:14
time
PGU Details
Id 1
Status 2-alert
Owner Saul
Source type 1-wind
Contract power 1000
Installed power 1500
Amplification factor 1
PGU Infractions
Type Count Last Time
critical 0
major 0
minor 0
PGU Constraint
Limit
power limit
Kw
SUBMIT
Last constraint 195.50 8/22/22, 2:05 PM
ALERT
URGENCY


MonitORES
PGU List
PGU 1
Add PGU
Energy Production
PGU Power
power
1 kW
0.8 kW
0.6 kW
0.4 kW
0.2 kW
0 kW
14:33
14:34
14:35
14:36
14:37
14:38
14:39
14:40
14:41
time
PGU Details
Id 1
Status 3-urgency
Owner Saul
Source type 1-wind
Contract power 1000
Installed power 1500
Amplification factor 1
PGU Infractions
Type Count Last Time
critical 1 8/22/22, 2:32 PM
major 2 8/22/22, 2:40 PM
minor 1 8/22/22, 2:41 PM
PGU Constraint
Limit
power limit
Kw
SUBMIT
Last constraint 195.50 8/22/22, 2:05 PM
ALERT
URGENCY

Bibliography

[1] World Resources Institute. *Climate Watch Historical GHG Emissions*. 2021. URL: https://www.climatewatchdata.org/ghg-emissions. (accessed: 3.08.2021).

[2] Climate Action Tracker. *EU CO2 Targets*. URL: https://climateactiontracker.org/countries/eu/targets/.

[3] SPW Energie. URL: https://energie.wallonie.be/fr/la-contribution-wallonne-au-plan-national-energie-climat-2030.html.

[4] *Le Plan national énergie - climat 2021 - 2030*. URL: https://www.plannationalenergieclimat.be/.

[5] FEBEG. *Statistiques Electricité*. URL: https://www.febeg.be/fr/statistiques-electricite.

[6] Quentin Gemine, Damien Ernst, and Bertrand Cornélusse. "Active network management for electrical distribution systems: Problem formulation, benchmark, and approximate solution". In: *Optimization and Engineering* 18.3 (2016), pp. 587–629. DOI: 10.1007/s11081-016-9339-9.

[7] Sebastien Mathieu, Damien Ernst, and Quentin Gemine. "Short-term active distribution network operation under uncertainty". In: *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)* (2020). DOI: 10.1109/pmaps47429.2020.9183701.

[8] Chih-Che Sun, Chen-Ching Liu, and Jing Xie. "Cyber-physical system security of a power grid: State-of-the-art". In: *Electronics* 5.4 (2016), p. 40. DOI: 10.3390/electronics5030040.

[9] R. M. Lee and M.J. Assante. *Analysis of the cyber attack on the Ukrainian power grid*. 2016. E-ISAC and SANS., Washington DC, TLP:White.

[10] Gaoqi Liang et al. "The 2015 Ukraine Blackout: Implications for false data injection attacks". In: *IEEE Transactions on Power Systems* 32.4 (2017), pp. 3317–3318. DOI: 10.1109/tpwrs.2016.2631891.

[11] Gaoqi Liang et al. “A review of false data injection attacks against Modern Power Systems”. In: *IEEE Transactions on Smart Grid* 8.4 (2017), pp. 1630–1638. DOI: 10.1109/tsg.2015.2495133.

[12] Yao Liu, Peng Ning, and Michael K. Reiter. “False data injection attacks against state estimation in electric power grids”. In: *ACM Transactions on Information and System Security* 14.1 (2011), pp. 1–33. DOI: 10.1145/1952982.1952995.

[13] Ruilong Deng et al. “False data injection on state estimation in power systems—attacks, impacts, and defense: A survey”. In: *IEEE Transactions on Industrial Informatics* 13.2 (2017), pp. 411–423. DOI: 10.1109/tii.2016.2614396.

[14] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Bitcoin, 2008.

[15] Vitalik Buterin et. Al. *Ethereum Whitepaper 2020*. Feb. 2020. URL: https://ethereum.org/en/whitepaper/.

[16] Nick Szabo. *The idea of smart contracts*. Nick Szabo’s Papers and Concise Tutorials, 1997.

[17] Gaoqi Liang et al. “Distributed blockchain-based Data Protection Framework for modern power systems against Cyber attacks”. In: *IEEE Transactions on Smart Grid* 10.3 (2019), pp. 3162–3173. DOI: 10.1109/tsg.2018.2819663.

[18] Matthieu Stephant et al. “Increasing photovoltaic self-consumption with game theory and Blockchain”. In: *EAI Endorsed Transactions on Energy Web* 8.34 (2021), p. 166770. DOI: 10.4108/eai.27-10-2020.166770.

[19] ORES. *ORES - Qui sommes-nous?* 2022. URL: https://www.ores.be/qui-sommes-nous. (accessed: 18.05.2022).

[20] Aggelos Kiayias et al. “Ouroboros: A provably secure proof-of-stake Blockchain Protocol”. In: *Advances in Cryptology – CRYPTO 2017* (2017), pp. 357–388. DOI: 10.1007/978-3-319-63688-7_12.

[21] Hyperledger. *Hyperledger: Walmart case study*. Feb. 2018. URL: https://www.hyperledger.org/learn/publications/walmart-case-study.

[22] Gomez Antonio Exposito, Antonio J. Conejo, and Claudio Canizares. *Electric Energy Systems: Analysis and Operation*. CRC Press, 2020.

[23] CWAPE. *La Cwape*. 2022. URL: https://www.cwape.be/?lg=1%5C%5C&dir=7.3.8.

[24] Serguei Yu. Popov. “The Tangle”. In: 2015.

[25] Mance Harmon Dr. Leemon Baird and Paul Madsen. *Hedera: A Public Hashgraph Network Governing Council*. 2020. URL: https://hedera.com/hh_whitepaper_v2.1-20200815.pdf.

[26] et. Al. Elli Androulaki Christian Cachin. *Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains*. 2018.

[27] Richard Gendal Brown. *The Corda Platform: An Introduction*. 2018.

[28] Statista Research Department. *Number of motor vehicles registered in the United States from 1990 to 2019*. Feb. 2021. URL: https://www.statista.com/statistics/183505/number-of-vehicles-in-the-united-states-since-1990/#:~:text=How%5C%20many%5C%20registered%5C%20motor%5C%20vehicles,at%5C%206.3%5C%20million%5C%20in%5C%202016..

[29] Raynor de Best. *Size of the Bitcoin blockchain from January 2009 to February 2, 2021*. Feb. 2021. URL: https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/.

[30] Rowan Van Pelt et. Al. *Defining Blockchain Governance: A Framework for Analysis and Comparison*. Utrecht University, 2020.

[31] Su Qianqian et. Al. *Revocable Attribute-Based Signature for Blockchain-Based Healthcare System*. State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, 2020.

[32] Miguel Castro and Barbara Liskov. *Practical Byzantine Fault Tolerance*. Laboratory for Computer Science, Massachusetts Institute of Technology, 1999.

[33] AMIS. *Istanbul Byzantine Fault Tolerance*. Feb. 2017. URL: https://github.com/ethereum/EIPs/issues/650.

[34] Diego Ongaro and John Ousterhout. *In Search of an Understandable Consensus Algorithm*. Standford University, 2014.

[35] Ponton. *ENERCHAIN - DECENTRALLY TRADED DECENTRAL ENERGY*. Feb. 2019. URL: https://enerchain.ponton.de/.

[36] Jianbin Gao et al. "GridMonitoring: Secured sovereign blockchain based monitoring on Smart Grid". In: *IEEE Access* 6 (2018), pp. 9917–9925. DOI: 10.1109/access.2018.2806303.

[37] Eduardo Castello Ferrer et al. "Following leaders in Byzantine multirobot systems by using blockchain technology". In: *IEEE Transactions on Robotics* 38.2 (2022), pp. 1101–1117. DOI: 10.1109/tro.2021.3104243.

[38] chris-j-h. *Quorum Whitepaper v0.2*. Feb. 2018. URL: https://github.com/ConsenSys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf.

[39] Mohammad Dabbagh et. Al. *A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities*. Sunway University, 2020.

[40] Marko Vukoli´c Christian Cachin. *Blockchain Consensus Protocols in the Wild*. IBM Research - Zurich, 2017.

[41] Rana M. Nadir et. Al. *Comparative study of permissioned blockchain solutions for enterprises*. International Conference on Innovative Computing (ICIC). 2019.

[42] Stackoverflow. *Stackoverflow Developer survey 2019*. Feb. 2019. URL: https://insights.stackoverflow.com/survey/2019?utm_source=so-owned&utm_medium=blog&utm_campaign=dev-survey-2019&utm_content=launch-blog#technology.

[43] Kemal Eroğlu. *Menapay Blockchain Tests: Quorum TPS*. Feb. 2018. URL: https://medium.com/menapay/menapay-blockchain-tests-quorum-tps-8aac5f51820b.

[44] Mike Ward. *Transactions Per Second (TPS)*. Feb. 2018. URL: https://medium.com/corda/transactions-per-second-tps-de3fb55d60e3.

[45] Martin Nowak and Karl Sigmund. "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoner's Dilemma game". In: *Nature* 364.6432 (1993), pp. 56–58.

[46] Hyperledger Labs. *Fablo*. URL: https://github.com/hyperledger-labs/fablo.

[47] NestJS. *NestJS*. URL: https://docs.nestjs.com/.

[48] OpenWeather. *Weather API*. URL: https://openweathermap.org/api.

[49] Elia. *Wind power generation*. data retrieved from Elia grid data, https://www.elia.be/en/grid-data/power-generation/wind-power-generation. 2022.

[50] Fablo. *Fablo REST*. URL: https://github.com/fablo-io/fablo-rest.

[51] Hyperledger. *Hyperledger Explorer*. URL: https://wiki.hyperledger.org/display/explorer/Hyperledger+Explorer.

[52] the free encyclopedia Wikipedia. *Publish–subscribe pattern*. URL: https://en.wikipedia.org/wiki/Publish-subscribe_pattern.

[53] *Crontab - Quick Reference*. URL: https://www.adminschoice.com/crontab-quick-reference.

[54] Anthony Roy. *Gestion optimale d'un système multi-sources pour un site isolé en mer*. URL: https://www.researchgate.net/figure/6-Courbe-de-puissance-dune-eolienne_fig39_340262383.

[55] OpenWeather. *Current Weather API doc*. URL: https://openweathermap.org/current.

[56] Echarts. *Echarts Documentation*. URL: https://echarts.apache.org/en/option.html.