ASGARDS-H: Enabling Advanced Smart Grid cyber-physical Attacks, Risk and Data Studies with HELICS

William Lardier

A Thesis
in
The Department
of
Concordia Institute for Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science (Information Systems Security)
Concordia University
Montréal, Québec, Canada

November 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **William Lardier**

Entitled: **ASGARDS-H: Enabling Advanced Smart Grid cyber-physical Attacks, Risk and Data Studies with HELICS**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair
Dr. Mohsen Ghafouri

_____ Examiner
Dr. Chunyan Lai

_____ Examiner
Dr. Amr Youssef

_____ Supervisor
Dr. Jun Yan

Approved by _____
Dr. Abdessamad Ben Hamza, Graduate Program Director

December 2020 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

ASGARDS-H: Enabling Advanced Smart Grid cyber-physical Attacks, Risk and Data Studies with HELICS

William Lardier

Smart infrastructures are increasingly built with cyber-physical systems that connect physical operational technology (OT) devices, networks and systems over a cyberspace of ubiquitous information technology (IT). A key objective of such interconnection is to offer a data coverage that will enable comprehensive visibility of dynamic environments and events. The arrival of Internet-of-Things, 5G, and beyond in smart infrastructures will enable the collection of unprecedented volumes of data from these various sources for critical visibility of the entire infrastructure with advanced situational awareness. To break the barriers between the different data silos that limit advanced machine learning techniques against cyber-physical attacks and damages and to allow the development of advanced cross-domain awareness models, the thesis tried to develop a modular, complete and scalable co-simulation platform allowing the generation of standardized datasets for research and development of smart distribution grid security. It addresses the lack of realistic training and testing data for machine learning models to enable the development of more advanced techniques. Our contributions are as follows. First, a modular platform for software-based co-simulation testbed generation is developed using the HELICS co-simulation framework. Second, scenarios of instabilities, faults, cyber-physical attacks are built to allow the generation of a realistic and multi-sourced dataset. Third, well-defined datasets are generated from the developed scenarios to enable and empower data-driven approaches toward smart distribution grid security.

Acknowledgments

First of all, I would like to deeply thank my supervisor, Dr. Jun Yan, for giving me this opportunity to work on this thesis in the context of Ericsson's Global Artificial Intelligence Accelerator GAIA, via Mitacs. I would also like to thank him for allowing me to join his laboratory and for the trust he has placed in me on numerous occasions. I have learned a lot from this research experience, and more specifically from innovative and future-oriented fields. I would also like to thank Dr. Mourad Debbabi for his help, confidence and support during this experience.

Second, I would like to thank Saman Bashbagi, Gregory Dutrieux, Jean Michel Sellier and Emmanuel Thepie Fapi for their follow-up and support during my experience at Ericsson. All of them have always been listening and present during the duration of this internship. I also thank all the Ericsson and Mitacs employees who made this experience possible despite complex conditions.

Thirdly, I would like to thank Quentin Varo, Moshfeka Rahman and Antoine Demarquay for their help and support throughout my thesis and for our many exciting and interesting discussions.

Finally, my most sincere thanks go to my parents, without whom I could not have come to this point. Thanks to my father without whom I probably would never have developed such an interest in computer science and cybersecurity and to my mother for her unfailing support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Smart grids play an integral part in critical infrastructures at both local (cities) and national levels. The recent arrival of these new monitoring capabilities also brings many challenges that cybersecurity and physical security experts must face. In fact, these systems must be built on approved standards and must precisely describe the different functionalities of the infrastructure, which combines two very different yet crucial domains, namely electrical grids and communication networks. Although such standards are well established, many threats remain, and guidelines such as the NIST IR 7628 (rev.1) Guidelines for Smart Grid Cybersecurity [1] attempt to provide comprehensive approaches to address these threats and raise awareness of the security threats.

Recently, with the multiplication of communication standards, the arrival of 5G and the constant evolution of interconnections between these networks, we are more and more realizing the benefits of comprehensively enhance the visibility of dynamic environments and events. Increases in computational power and advances in the processing of massive data silos will permit these tremendous volumes of data to be transformed into useful information for monitoring the state of the system and thus protecting it.

Among the emerging and promising techniques to bring security within such cyber-physical systems, Situational Awareness (SA) has emerged as a natural solution for securing CPS, and in particular smart grids. For the specific case of power grids, the U.S. National Institute of Standards & Technologies (NIST) in its practical guide to Situational Awareness for Electric Utilities [2] has confirmed the essential need to provide a perception

of the environment, the understanding of its meaning and the projection of its state in the near future.

In order to address complex threats whose effects might slightly affect each data silo to remain undetected, it is important for learning models to ensure that they will have access to all data and thus not risk a delayed response to major threats [3]. At the same time, the crucial lack of means of access to this type of data makes the development of complex machine learning techniques almost impossible. The majority of real-world scalable solutions are nowadays proposed by private companies, working in collaboration with the government of the state in which they operate. To address these multiple problems, this work, the first part of a larger project as part of a Mitacs agreement between Concordia University and Ericsson, aims to develop a testbed based on open-source technologies and solutions as well as the various standards in place.

1.2 Contributions

How to enable the co-simulation of cyber-physical systems to serve as a basis for the study and generation of data related to cyber-physical security? Through this work, we try to answer this problem by introducing a modular, systematic and highly scalable project allowing the investigation of complex scenarios as well as the future implementation of attacks and events that will serve as a basis for the development of new detection and defense mechanisms for Smart Grid, based on machine learning.

This thesis aims to propose a co-simulation platform enabling the integration of complex cyber-physical attacks, events, faults and instabilities to generate standardized datasets for multiple scenarios. Through this project, we propose the integration of a wide range of technologies in a single co-simulation environment. As the integration of these existing works is only the skeleton of a larger project, many modular ad-hoc implementations are added to create a complete, scalable and modular environment:

- We develop a highly customizable software-based solution for co-simulation testbed generation from a simple power network model;
- The proposed testbed is compatible with Hardware-In-The-Loop (HIL) integration, the addition of control systems, user scripting for real-time integration of machine learning techniques, cyber-physical attacks, faults and events;

- The capabilities of the testbed are multiple: power grid simulation via the GridLab-D simulator, developed by the U.S. Department of Energy (DOE) at Pacific Northwest National Laboratory (PNNL), with the integration of the HELICS co-simulator, the OMNeT++ simulator, developed by OpenSim Ltd. also linked to HELICS and embedding the INET and SimuLTE frameworks, as well as the ad-hoc implementation of multiple protocols and functionalities from smart grids systems. Two federates are also included in the co-simulation: a Meterpreter-like cyber-physical attack federate and a visualization and control federate;

- We generate datasets of multiple source ('csv', 'json', 'pcap', 'vec') from the co-simulations, according to the options defined by the user at the time of testbed generation;

- We study the integration of 5G communications within smart grids in order to develop specific scenarios and highlight the benefits introduced by 5G.

The contributions of the thesis can be summarized in two points. First, we propose a comprehensive and modular co-simulation platform for customizable smart grids co-simulation, which includes capabilities from the power grid model integration to the latency induced by an attacker performing an eavesdropping attack on a specific networked device. Secondly, we propose a realistic environment based on complex scenarios enabling studies of synchronized or desynchronized systems under different conditions and permit, via the generation of standardized datasets, their study by machine learning techniques.

1.3 Ericsson GAIA program

This work is part of the Ericsson Global Artificial Intelligence Accelerator (GAIA). With nearly 300 data scientists and data engineers around the world, GAIA, launched in Montreal in 2019, brings together diverse profiles of researchers. Based in Canada (Montreal), the USA (Santa Clara), Sweden (Stockholm) and India (Bangalore), the clusters are working jointly to meet the various needs of consumers: network performance, new revenues, relentless efficiency and end-customer experience. In the context of the arrival of 5G, this accelerator aims to develop artificial intelligence techniques for the development of new technologies, applications and defenses.

GAIA MITACS consists of 6 main research clusters, with a total of 41 projects for 82 researchers. With 20 Post-Doctoral, 38 PhD and 24 M.Sc. students from 8 Universities, the clusters study various fields, from edge computing to cyber physical systems.

1.4 Thesis Organization

Chapter 2 provides an overview of the concepts discussed throughout this document. Chapter 3 provides a literature review of existing co-simulation solutions as well as the simulation of cyber attacks. Chapter 4 summarizes the co-simulation framework HELICS and presents in detail the co-simulation testbed generation platform, the different sub-parts and protocols/standards used, as well as the different approaches used for the simulation of attacks, events, faults and instabilities. Chapter 5 validate the proposed platform and detail the dataset generation process and implements a first use-case enabling the complete study of our testbed capabilities. Finally, Chapter 6 will discuss this project limitations, and Chapter 7 concludes this thesis and highlights the future work for the improvement of this project and its use. Figure 1 summarizes the thesis organization. Blue dotted lines highlight thesis contributions.

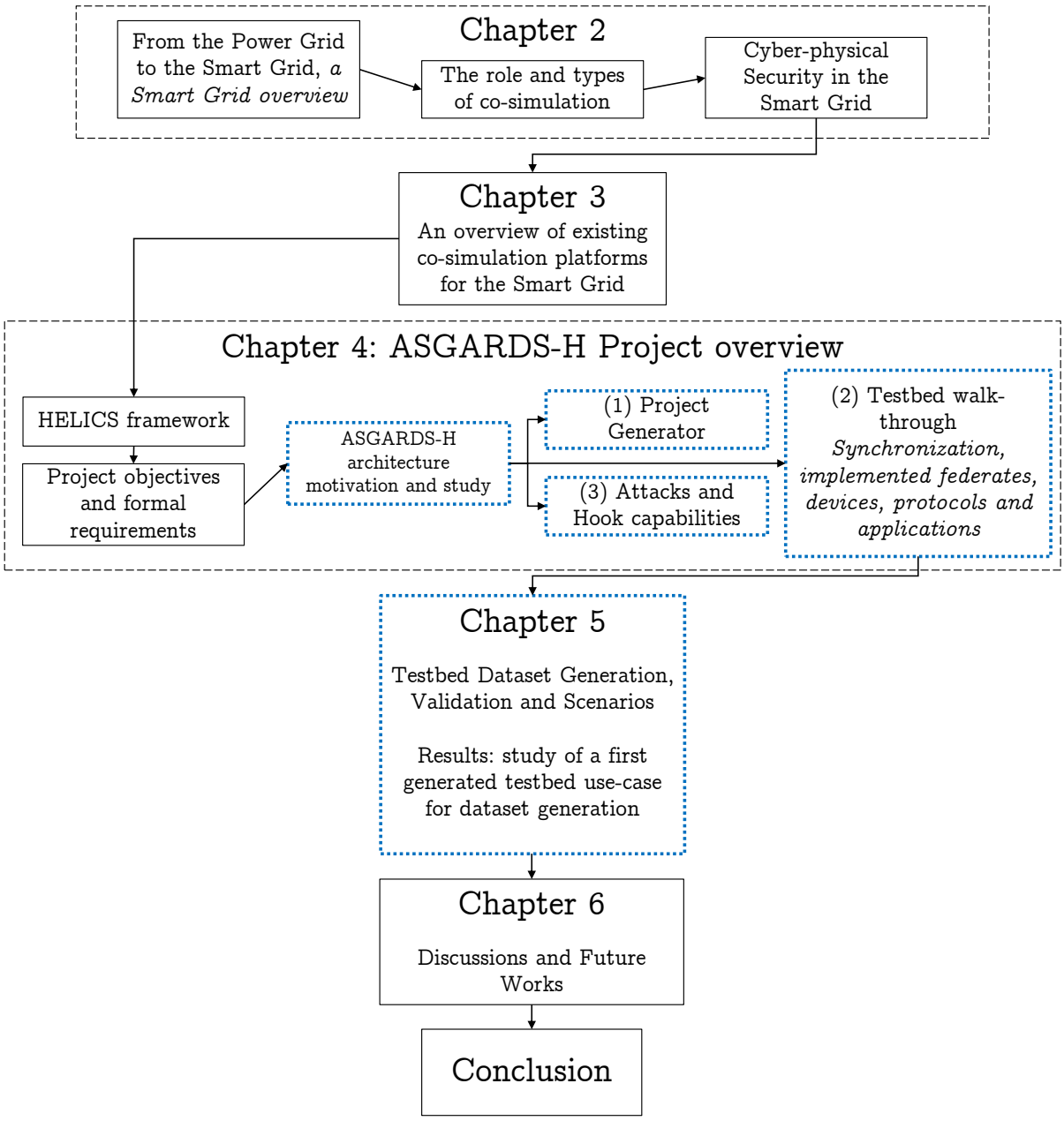


Figure 1: Thesis Organization

Chapter 2

Background

2.1 From the Power Grid to the Smart Grid

Conventional power grids are defined as a centralized power plant delivering energy to end users. Initially, there was little monitoring of power consumption and network status. At the same time, the maturity of technologies has allowed the integration of new equipment into the electricity infrastructure to improve economic development. Since then, administration and control equipment has been developed for the specific monitoring of smart grid-related parameters. They facilitate the monitoring of the electrical architecture and give a modern dimension to the electrical networks - the smart grids. This name encompasses the idea of a monitored electrical network via secure two-way communication improving the interoperability of the electrical and IT layers.

Over time and with the arrival of new supports for large-scale communications such as 5G, smart grids continue to evolve into a global system where mobile entities (electric vehicles) or monitoring of secondary elements (weather conditions), green energy generation (PVs, Wind Turbines, etc.) interact, as well as standardised communication protocols, helping the operator to have a global overview of the system, we speak here of situation awareness.

According to the US Department of Homeland Security (DHS), smart grids are one of 18 critical systems in our society [4]. This type of system is so vital that a widespread failure, global damage or inability to function would cause serious disruption at many levels: from citizens to defense systems to economic stability. Moreover, many critical systems depend on Smart Grids, making this complex system even more special and vital.

For the companies, smart grids have many benefits, i.e. the reduction of electricity production through better management of consumption, thereby lowering costs for customers and reducing gas emissions.

2.1.1 Distribution and Transmission Systems

A distribution system, or electric power distribution, is a part of an electrical grid serving consumers. A distribution system carries electricity from the transmission system (not covered in this work) to the end-user. As part of this project, we consider only the Distribution part of the electrical grid, from the substation to the smart meters in households.

2.1.2 Smart Grid Communications

As summarized in Section 2.1.3, the risks of cyber-physical attacks against smart grids represent a challenge that will be faced by multiple stakeholders, from the operator to the end user. Smart grids are equipped with diversified systems for sensing, controlling and processing large volumes of data:

- Real-time retrieval of physical system status such as frequency or phase angle via Phasor Measurement Units (PMUs) is received by data aggregators (here, Phasor Data Concentrators) before being received by the Control Center, which in turn will analyze this data using a state estimator and intrusion detection system (IDS) in order to send controls to the network and detect illegal attempts to tamper with the system by an attacker;
- This principle of bi-directional communication enables the retrieval, processing and communication of controls in real time via timestamped payloads, permitting the controller to make decisions in an optimized way at different levels of the smart grid system (from the generation to the distribution);
- Finally, the objective is to enable a self-healing capability in response to attacks, various events and faults.

The inclusion of Supervisory Control and Data Acquisition (SCADA) systems that recover and process of data from the network is a major step towards a situational awareness-capable monitoring system. After processing the data, the SCADA system sends controls to the actuators to ensure the functional state of the system. In contrast to the Wide Area

Monitoring, Protection and Control (WAMPC) system, a SCADA system is not limited to process only the Synchrophasors measurements from the PMUs but is also able to collect data from multiple sources.

2.1.3 Cybersecurity Implications

Threats to smart grids are not to be taken lightly. History has already shown that threats can come from the computer network and have only effects, albeit devastating, on the electrical, and therefore the physical layer. The techniques used by attackers are proportional to the means at their disposal. The development of resilient and secure smart grids should allow improvement at various levels, from prevention to system recovery after an event:

- Improves resilience to disruption;
- Set up predictive maintenances enabling self-healing in the face of disturbances;
- Improve cybersecurity.

Unfortunately, the addition of new, flexible and modular technologies brings new cyber security threats for smart grids as the attack surface increases. Also, some attacks, recently developed in the academic literature, tend to show that both physical and cyber attacks can have devastating effects on smart grids, while remaining undetectable with current detection means: they are called cyber-physical attacks (CPA). In 2014, the US Industrial Control Systems Cyber Emergency Response Team noted that nearly a third of incidents were targeting the energy sector [5].

In 2007, Iran's nuclear power station was targeted by a malware-based attack (Stuxnet) that infiltrated the control system in order to slow down the development of the country's nuclear technologies [6]. In 2014, Dragonfly, another malware, was released, targeting over 1000 electrical companies to infiltrate the kernel for the control of the aforementioned systems [7]. In 2015, during the civil war in Ukraine, a Trojan Horse-based attack targeted the power grid and put 80,000 people in the dark [8]. Recently, DHS has noted an increase in the number of cyber attacks against electrical systems, with nearly 4,300 attempts in 2017 against the French electricity grid.

However, as explained in Section 2.3.3, attacks do not always seek to damage or alter the physical system state but can also be used to retrieve sensitive information for later attacks or to gain strategic advantage.

2.1.4 The Role of Co-simulation

To better mitigate cyber attacks, it is important that all stakeholders are well aware of the risks involved in a smart grid context. This includes IT experts, cybersecurity professionals, agencies, companies or government organizations managing smart grids, and even end-users when using smart grid technologies. Indeed, the majority of previously presented attacks have been achieved through some form of social engineering intervention at some point in time, or by exploiting the lack of human vigilance to gain access to the system.

Traditional training methods are limited for a system such as the smart grid: how then to train stakeholders in automation, ICT (networks, cybersecurity) and physical areas, when they are all interconnected? How can stakeholders be trained for a major incident without modeling it in the real world (which would be too costly)? These experiences and skills can only be obtained by training on a real system or, in the case of smart grids, via a **co-simulator**.

Co-simulation addresses the limitations of access to a real smart grid system by integrating various simulators, or federates, into a single synchronized and interconnected environment. It leverages all the capabilities of the various existing simulators into a single, unified platform that can be configured with various scenarios for the simulation of complex events, hence the proposition of the co-simulation as an emerging and promising technology for the study of cyber attacks and their effects in smart grids. Although many co-simulation platforms have already been proposed, few are comprehensive enough to fully explore the effects of cyber attacks.

In this context, we propose a new co-simulation platform named *ASGARS-H* (Advanced Smart Grid cyber-physical Attacks, Risks and Data study with HELICS) to allow the study of complex smart grid scenarios (including weather, instabilities, faults, cyber-physical attacks) and the generation of realistic dataset for the development of detection techniques based on machine learning. Our methodology combines the HELICS co-simulation platform, the GridLab-D simulator, the OMNeT++ cyber simulator and other ad-hoc federates. The following Section presents the minimum necessary concepts for a proper understanding of this thesis.

2.2 Co-simulation

The development of simulation tools depicting complex engineered systems that combine physical, software and network technologies is emerging, due to the increasing demand from industry and academia [9].

Thus, each tool, addressing one particular challenge, is integrated within a consistent system and communicate with the other tools. The integration of such tools within a co-simulation environment is highly dependent on the tool level of modularity, as well as the moment when this communicative capability has been integrated during the development process [10]. Indeed, the development of a co-simulation platform must be holistic: although multi-disciplinary simulators or tools are developed independently, it is crucial for the coherence of the co-simulation that each part is integrated with the others frequently during the development process. Contrary to a classical development project, a co-simulation platform must be validated at each level of the simulation: each simulation, federation, then the whole co-simulation. In addition, at a more technical level, validation, debugging and the usual techniques of a project development lifecycle become more complex as tools from different domains are introduced.

While there are many simulators for the realistic representation of partial systems, few are capable, without major modifications, of being properly included in a holistic development process of a co-simulation platform [11]. It is indeed complex to integrate or replace a specific simulator, on the one hand because it have to be licensed for integration into a larger system and on the other hand because the integration of such a simulator is complex and requires specific knowledge of the system.

Co-simulation then realistically represents coupled system interactions. Although it is complex to obtain a single model to represent a system as a whole, high-fidelity co-simulation enables the study of essential aspects of the whole system, where the capture of interactions between individual tools represents the most important challenge [10].

On a more general point of view, co-simulation is an innovative overlay for controlling and synchronizing federates, or interconnected simulators. The latter are then considered as complete and independent subsystems, accepting input commands, values or controls (subscriptions) and producing output data, events or states (publications), whether periodic (timestep) or dynamic (event-based). These federates can integrate various systems: Hardware-In-The-Loop (e.g., a real Phasor Measurement Unit integrated using FNCS),

solver equations (e.g., a power system simulator), software executing kernel (e.g., OMNeT++ for the network communications simulation).

2.2.1 Simulation and Co-simulation Basics

In this Section, we present the basic concepts of co-simulation. For ease of reading, we illustrate these concepts with a virtual example of a co-simulation between a vegetation simulator and a weather simulator.

Firstly, we speak of a *dynamic system* [10] when a simulator models a real system that uses an internal state and a set of methods to make it evolves over time. For our example, the vegetation simulator could have the states ($flowering$, $growing$, $deciduous$, $wilted$, $dormant$). The evolution methods would represent the step(s) that change the system state to another one, according to specific conditions (e.g., the input temperature and the time).

The *Behavior Trace* [10] is a set of transitions leading to a new state (and thus new outputs) of a dynamic system. For example, the transition set x would here be defined as a mapping between the temperature T_p, the time T and the set of reals $\mathbb{R}$, that is, $x : T_p \cup T \rightarrow \mathbb{R}$.

In order to properly evolve the system over time, a simulator must maintain a global variable, the simulated time $t \in T$, where $T \leftarrow \mathbb{R}$ is the base time. This time is different from *wall clock* time $\tau \in \mathbb{R}$, which is the real-time. The relationship between these two measurements can be summarized as $t = \alpha T$. Users can modify the simulator simulation' speed by modifying the α variable, for example to create a real-time simulator ($\alpha = 1$). It can also be used to accurately measure the performance of the simulator.

Simulation speed is not the only metric to measure the performance of a simulator. The *validity* is a second crucial concept (and a property of a dynamical system) since it represents the ability of the simulator to actually produce the right outputs, given an initial state and a set of commands, controls, or input values. Validity allows dynamic system simulators with a predictability capability. For example, if our vegetation simulator has perfect validity, it is able to correctly predict the correct output state given inputs it has never received before. We also need to dissociate the concept of validity of a simulator from a digital twin, since a high level of validity can be achieved without necessarily having the same routines, functions or protocols as a twin system.

Indeed, a *simulator* is an algorithm that computes the evolution of the state of a dynamic system. In the case of software-based simulators, this calculation is more generally an

approximation (for example the growth speed of a plant), whereas a hardware simulator is able to correctly produce the optimum results, since it will directly exploit the real physical properties. Approximations within software simulators are mainly due to the impossibility to find a solution within a set of infinite size or to calculate a trajectory over a continuum [10]. Finally, we can refer to *accuracy* when a simulator is able to produce results with a normalized error close enough to zero, given an acceptable threshold. Any simulator needs an initial state and a defined trajectory in order to properly start the *simulation*.

A simulation is the generation of a behavioral trace permitted by inputs, depending on the validity and accuracy of the dynamic system and the simulator. [10]. A Simulation Entity (SE) refers to a simulator with a dynamical system, that produces a behavior trace. Now if we consider that each part of a more complex simulated system is represented by one SE. These SE can now be integrated together and communicate to share inputs/outputs, enabling the development of a co-simulation scenario. The co-simulation is a set of federates that represent one or more federations that produces a global behavior trace produced by the SEs coupling. For example, the vegetation simulator is fed by the weather simulator that will influence the growth rate of the plant.

In order to correctly interconnect these SEs which can evolve at different speeds and simulated times (fixed time-step, dynamic time-step, discrete events, time stepped), a scheduler is required. This special module manages the simulated time of each SE and to synchronize their inputs and outputs to guarantee a high level of validity and accuracy. When several schedulers are connected to each other, we refer to a *co-simulation*.

Within a co-simulation, it is important that all P_i properties satisfied by the SEs are also satisfied at the global level. Thus, we talk about additivity of these properties: the properties provided by the co-simulation must correspond to the sum of all the under-validated properties.

The following subsections summarize available all three co-simulation techniques, namely Discrete-Event (DE), Continuous-Time (CT) and Hybrid.

2.2.2 Discrete Event-based Co-simulation

Co-simulation schedulers that uses a Discrete-Event synchronization mode is based on a system design that discretely evolves over time. Examples include a computer communications simulator, or our vegetation simulator. When a system is discrete, then it evolves either at regular intervals (e.g., traffic lights), or as a result of changes in state or the arrival

of new data (e.g., the vegetal dries up due to a drought). Thus, it is possible, in DE-based co-simulation, that the SEs continue to change their states despite the simulation being stopped.

In fact, during a Discrete-Event simulation, it is possible at a given time $t = t_i$, that there is no output for a simulator: this is accepted by the scheduler, since this operating mode relies on timestamped events, which are not constrained to follow a periodic update scheme.

In this mode, if an event appends at a given time, it is automatically processed before the simulation time progresses. It is also up to the scheduler to define the priority of events, when several occur at the same time, in order to guarantee causality.

Finally, DE co-simulation must guarantee the following properties [10]:

- **Reactivity**: the scheduler must process events as soon as they occur.
- **Transiency**: the ability to allow several successive integrations not to increase the simulation time.
- **Ability to predict the next timestep**: each Operating System must communicate to the scheduler its next timestep so that the co-simulation can be correctly synchronized.
- **Determinism**: the same initial states with the same configuration must lead to the same behavorial traces.
- **Distribution**: Run co-simulations with physically remote equipment.

2.2.3 Continuous Time-based Co-simulation

Co-simulation in continuous time mode binds simulators which act as solvers of differential equations and which, therefore, evolve continuously according to the chosen time resolution. The interpolation is then used by the scheduler to calculate all the intermediate values between two timescales. Interpolation techniques are outside the context of this thesis.

The synchronization of this type of co-simulation is more basic, but some constraints will have to be solved, e.g. algebraic loops, where the output depends on the input [12], correct initialization of simulators [13], error control and convergence or noise and delays.

2.2.4 Hybrid Co-simulation

As part of this work, we are setting up a hybrid co-simulation environment. A hybrid co-simulation (HCS) is a mix of the characteristics and constraints of each of the SEs. Unlike DE/CT-based co-simulations, hybrid simulations manage events in a continuous time context (Hybrid CT) and vice versa (Hybrid DE).

The main challenge of hybrid co-simulation is to maintain accuracy and validity, as presented above. Since HCS is highly dependent on the studied system, it is not formally possible to pinpoint the exact requirements. However, most of them will correspond to the scheduler's ability to maintain a coherent co-simulation state by synchronizing simulators that may evolve differently over time. As illustrated in the HELICS platform Section, HCS is currently being actively studied and developed.

2.3 Cyber-physical Security in the Smart Grid

2.3.1 Security Objectives

Securing cyber-physical systems aims to guarantee three security objectives that we detail below: integrity, confidentiality and availability. We also highlight three other specific security objectives related to CPS systems: authentication, authorization and physical security.

- **Integrity**: Ensure that all data measured or generated during the network lifecycle is accurate, complete and was not modified by external parties. Both non-repudiation and authenticity are sub-properties of the integrity objective. Guaranteeing this property is indeed a way to prove that the system is trusted, and that the data is correctly timestamped and of good quality. Integrity can be divided into two categories: integrity against unintentional errors during data transmission, and security against intentional modifications. In the first case, Error Correction Codes (ECC) and Cyclic Redundancy Check (CRC) are common methods for correcting and detecting any changes in the payload of the message. However, since this technique does not authenticate the data, it does not defend against intentional modification, where the attacker would only have to recalculate the CRC value directly. In the second case, integrity can be guaranteed using one-way cryptographic hash functions or Message Authentication Codes (MAC). Targeting integrity of the CPS could aim to achieve

physical damage objectives by preventing any feedback from the sensors to the Control Center or to the actuators to reach their destination unaltered [14].

- **Confidentiality**: Ensure that all data measured or generated during the network lifecycle is not disclosed to anyone else than the legitimate receiver or parties that are authorized to access the plaintext information [14]. For this purpose, cryptographic ciphers that comply with NIST recommendations in terms of the number of security bits are involved [15]. Numerous protocols integrate this confidentiality capability, the best-known being TLS. The standard currently used to encrypt data is the Advanced Encryption Standard (AES), but alternatives are being selected by NIST for Internet of Thing (IoT) related devices, limited in their computing capacity or battery capacity [16].

- **Availability**: Ensure that all authorized parties accesses to the system are ensured in a timely fashion with available services when needed. Network communications channels, physical controls processes and data storing, and processing must function correctly [14]. Availability is the most important security property in the context of smart grids: the loss of availability can lead to a blockage of the real-time decision-making system, highly dependent on the network performance. This loss of property can occur during unintentional events (network congestion) or during specific attacks.

- **Authenticity**: All data, communications, transactions and devices are genuine and able to successfully authenticate and thus trust other parties within the system context [17]. Authenticity of data is enabled through the use of MAC or digital signatures (e.g., RSA). Authentication of other parties can be enabled through certificates such as the x509 standard. This is an essential property for systems such as Smart Grid to prevent any attacker from illegally infiltrating the network to try to tamper with it.

- **Authorization** "is the granting of a right or permission to a system entity to access a system resource. This function determines who is trusted for a given purpose" [17]. Within a CPS context, Authorization, or access control, is a set of operator-defined policies that ensure, restrict and control the data access according to the party's role. In smart grids using networks such as 5G, it is important that these rules are well defined to prevent any attacker from modifying, accessing, deleting data or sending malicious commands that would be accepted by the communication system.

- **Physical security**: To prevent any attacker from tampering with the various pieces of equipment used to monitor the power grid that would allow him or her to take control of it, inject malicious code or paralyze the entire system.

2.3.2 Detection Systems in the Smart Grid and Their Limitations

The main module for intrusion detection within Smart Grid systems is the IDS. There are three categories: signature-based IDS, anomaly-based IDS and hybrid-based IDS [18]. Implemented techniques will mainly base their malicious attempt detections on known threats, so that the models used do not guarantee sufficient security against fully unknown risks or patterns or produce too many false positives. In addition, the multiplication of data sources in different heterogeneous silos, correlated or not, within the rapidly evolving smart grids requires more robust and complete methods.

In spite of the usual ideal cases that are considered when studying CPS attacks and their consequences on Smart Grids, most of the attempts that operators will encounter will be imperfect attacks where the access to information and devices is limited for the attacker, leaving traces that can be detected via the different detection methods implemented. More generally, security should be proportional and adapted to the risks considered.

Academic research is actively studying the use of machine learning techniques to extend the detection capability of known and unknown attacks, taking advantage of the full range of data that a SCADA system can process. However, the lack of fully developed and documented methods for the generation of standardized, realistic and complex datasets is a challenge for researchers implementing advanced detection techniques. In this context, our project aims to create a highly modular platform for the generation of dataset that can be used by these new detection techniques.

2.3.3 Threats

Following the different objectives presented in Section 2.3.1, we elaborate in more detail on the different concrete risks Smart Grids face. Attacks are summarized in Figure 2.

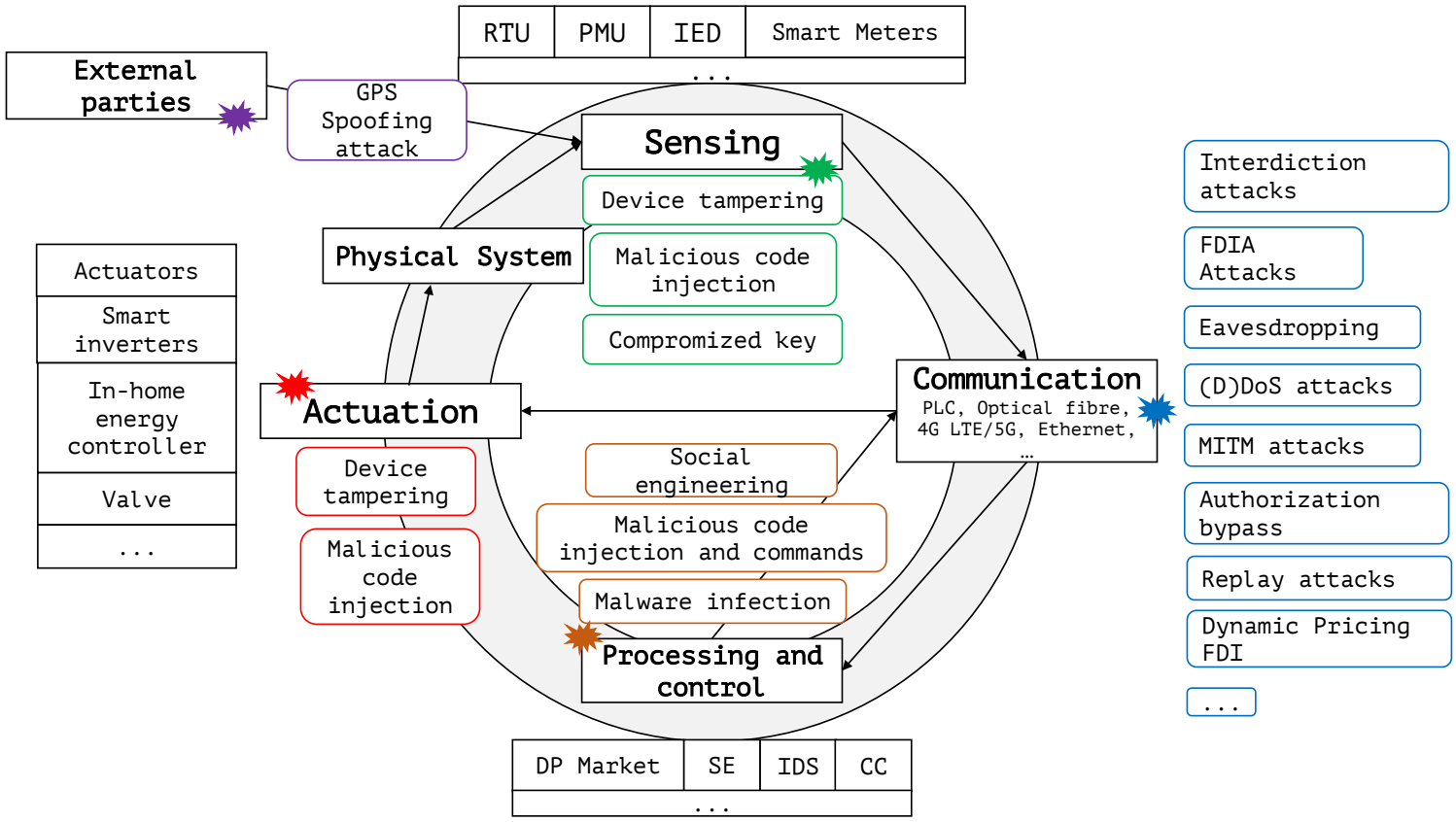


Figure 2: Overview of the Smart Grid CPS threats

2.3.3.1 Physical Security

Although physical security is an issue that is now well understood in the cybersecurity environment, it is of paramount importance in the context of CPS. Beyond the physical security of the system itself, which is a legal matter, it is essential to ensure the trust of the operator in the sensing and control equipment within the system. The use of hardware modules such as the Trusted Platform Module (TPM) is an inexpensive and common method of ensuring physical security of equipment, via secure booting, limiting the possibilities for the attacker, for example in terms of injecting malicious code, also enabling the secure storage of encryption keys and ensuring the authenticity of the data [14].

2.3.3.2 Integrity

Attacks that modify data without authorization are usually aimed at hiding the effects of a third-party attack on the system from the detector. The False Data Injection Attack (FDIA) is a concrete example of an attack that is increasingly studied by the scientific community [19]. For this attack, the attackers alter the measurements (frequency, phase angle, etc.) of one or more sensors within the system in order to modify the SE calculation of the system state in the desired way. With FDIAs, the attacker seeks to maximize the impact of his attack on the physical system by making the most of his knowledge of the system, while remaining stealthy. More recently, this attack has been extended to the market where prices are dynamically negotiated within smart grids, in order to impact, depending on the objectives, the cost for the operator, the customer, or even actively modify the overall power consumption of a market and induce over-consumption at the global level, inducing important financial losses.

Interdiction attacks were one of the first large-scale malicious attacks targeting smart grids and exploiting the integrity property [3]. This attack triggers lines, transformers, generators, buses or substations via manipulated control commands or bad measurements.

The GPS spoofing attack seeks to desynchronize PMUs by broadcasting a malicious (packet forging) or replayed (replay attack) time signals in order to deceive the PMU and modify the time it receives, leading to a modification of the Synchrophasor, representative of the state of the system at this point of the network.

2.3.3.3 Confidentiality

As opposed to an integrity attack, an attacker seeking to breach confidentiality will not attempt to alter the state of the system in this context. The recovery of data, whether from CC controls or SCADA system monitoring, may be highly useful in the development of an attack such as the FDIA in order to know at least partially the state of the attacked system.

The Compromised-Key attack allows an attacker, once the encryption key has been recovered, to decrypt all present (and past if the concept of forward secrecy has not been properly implemented) communications, in a way that is completely transparent to the operator. This attack, undetectable without more advanced systems like quantum encryption with Quantum Key Distribution, endangers the system which may be attacked later and with greater impact. In addition, the attacker could derive the known key to find or compute others within the system, to take advantage of it to impersonate legitimate equipment in the system, or even to carry out a large-scale FDIA. This type of attack is mostly avoided thanks to trusted devices such as TPM [14].

The eavesdropping attack permits the attacker to intercept all exchanges passing through the system. It is a passive attack that can lead to privacy problems (smart homes), if confidentiality is not ensured.

Man-In-The-Middle attack refers to an attack where messages are sent to the Control Center or sensor, usually modified to be malicious and induce erroneous controls or manipulate system equipment.

2.3.3.4 Availability

Typically, an attacker that intends to saturate the network and cause severe congestion will implement a Denial of Service Attack (DoS), which can be amplified by distributing it (Distributed DoS). During a DoS, the attacker will send very large amounts of data within the network in order to paralyze it. The objectives may differ (network congestion, paralysis of a firewall, a router) as well as the techniques used (TCP SYN, ICMP, UDP floods, Ping of Death, NTP amplification attacks, etc.). Blocking communications can be a means of hiding a change in the state of the system from the operator, thus preventing him from taking the appropriate measures to guarantee its steady state.

As part of our work, we also implement two methods that are not strictly real-world attacks as such, but which play on two parameters influencing availability, which are network latency (packet delaying attack) and the correct transmission of information (packet

dropping attack).

2.3.3.5 Authorization

As any system built around different stakeholders with different levels of trust and confidentiality, smart grids are vulnerable to attacks that exploit a loophole in the logic of permissions and roles within the architecture. Some attackers will aim, through some form of social engineering or credentials theft, to take control of a part of the system, a sub-system or to send malicious commands by taking advantage of the rights obtained.

The proper training of the stakeholders then takes on its full meaning, besides, co-simulators can considerably help to secure this type of system.

2.3.3.6 Malicious Code and Malwares

As illustrated in past real-world attacks, targeting the communication network is not always the preferred solution for attackers. Indeed, circumventing the various security mechanisms in place can be too risky or costly for an attacker. On the other hand, everything can easily be simplified if the attacker is able to inject his own code via a malicious program (Malware) in order to take control of the system. Infection procedures are outside the scope of this work but represent a risk for which staff must be rigorously trained.

2.3.3.7 Risks and Adversaries

Finally, the evolution of power grids to the Smart Grid involves defending against attacks that traditionally target cyber systems. However, there is a much more important and crucial dimension here: smart grids are the basis of many vital aspects of a country (health care, transportation, national defense, food and water, financial services, etc. [3]). Attacks can have important and irreversible collateral effects that can affect us all and operators must to put in place adequate measures to secure such systems against multifaceted attackers (skilled hackers, disgruntled insiders with malicious intent, criminal groups, nation-states terrorist groups and possibly state secret services [14]).

Chapter 3

Related work

In this Section, we look at existing power system simulators, communication network simulators and smart grid co-simulation platforms and show their strengths and limitations.

3.1 Smart Grid Simulation Tools

3.1.1 Power Grid Simulators

Power grid simulation has a multitude of solutions available with many programming languages supported. Two emerging solutions for Distribution Systems are receiving increasing interest from the research community: OpenDSS and GridLab-D. Both simulators are actively developed around the concept of co-simulation and Smart Grids, offering innovative methods for advanced simulation and control in the context of co-simulation. For the simulation of Electric Transmission Systems, GridDyn [20], PSST [21] and PyPower [22] are the preferred solutions today.

OpenDSS is a Distribution System Simulator (DSS) providing a modernization effort [23]. Created in 1997, it is an open source project developed by the Electric Power Research Institute (EPRI) and offers advanced simulation of distribution systems. The simulator supports many types of advanced analysis to enable the study of Smart Grid applications in the future [24]. OpenDSS is modular and developers interested in contributing to the project or extend it for specific needs can take advantage of a permissive license and integrate a multitude of modules through the COM interface.

GridLab-D [25] is an open source project introduced by the Pacific Northwest National Laboratory (PNNL) and the US Department of Energy in collaboration with the energy

industry and universities. This project is active developed and is distinguished by its ability to adapt its simulation speed according to needs, from high-precision simulation (Delta-mode) to system studies over a long period of time. GridLab-D embeds modern techniques, modules and technologies for the integration of third-party systems such as a co-simulation frameworks (FNCS, HELICS), but also advanced tools for power system analysis. GridLab-D is developed in collaboration with co-simulation platforms and natively supports methods for the study of advanced scenarios in the context of co-simulation [24].

3.1.2 Communication Networks Simulators

Two main simulators for communications are extensively used today Smart Grids co-simulation: ns-3 and OMNeT++. Both are developed in C++.

ns-3 [26] is a highly modular open source project introduced by the U.S. National Science Foundation (NSF) for use in communication research and education. Introduced as the successor of ns-2, ns-3 offers a complete and validated library of the different communication protocols on the different layers of the OSI model, with the routing, wireless communication or multicast capabilities that can be found in ad-hoc, mobility or smart grids systems. This project uses other programming or scripting languages for the development of simulation models for easy study of specific scenarios.

OMNeT++ is a highly modular and component-based simulator written in C++ and GUI oriented, developed by OpenSim Ltd. [27]. This project aims at providing a unique solution to enable large-scale simulation of multiple scenarios, from ad-hoc networks with mobility to multimedia communication routing. OMNeT++ proposes, beyond the direct C++ implementation of new functionalities, the use of multiple configuration files to dynamically create new devices and define advanced simulation topologies. OMNeT++ can be extended using third-party projects such as simuLTE (4G LTE networks) [28] or the future simu5G [29] (5G simulation), INET [30] (bringing the majority of devices and protocols useful for most simulations) or OpenFlow [31] (Software-defined networks, SDN).

3.2 Existing Co-simulation Platforms

Recent efforts have been made by the academic community to introduce new co-simulation platforms. The Table 3.2 summarizes, for each solution, the different features and functionalities implemented. This list is based on the work of S.C Muller *et al.* [32] and Vogt *et*

al. [33], with the addition of new platforms recently published, compared to our proposed solution.

Recent contributions in the field of CPA co-simulation include ASTORIA and GridAttackSim. We have also proposed a work studying the security of quantum key distribution protocols in smart grids, via the Mosaik platform [34].

EPOCHS, the Electric Power and Communication Synchronizing System is one of the precursors in the co-simulation of smart grids [35]. It synchronizes ns-2 and PSLF (commercial electric simulator) via an agent called AgentHQ acting as a proxy for the co-simulation environment and the RTI module for synchronization. The platform has been developed to study the impact of the control of a SCADA system on an electrical and mechanical network, with the study of the angle of the current and of a rotor. Most of the EPOCHS applications are at the wide-area monitoring level. However, the project has not been updated since 2006.

TASSCS [36], the Testbed for Analyzing Security of SCADA Control System at the University of Arizona is a project synchronizing Power World and RINSE. The platform studies the impact of attacks on the SCADA system, including managing unauthorized access to PLC equipment, spoofing the Master Control Center, device scanning, MITM attack, request tampering, malicious function injection and DoS attack. The solution focuses on the simulation of Modbus communications and dissociates the different attacks according to the zones of the system: corporate, power, process control network, etc. The project has not been updated since 2011 and is not open source.

PowerNet [37] is a closed-source project integrating Modelica and NS-2 and focused on the synchronization of these simulators to propose a realistic co-simulation. The project does not implement the security dimension and has not been updated since 2011.

GECO: Global Event-Driven Co-Simulation Framework for Interconnected Power System and Communication Network [38] is a co-simulation platform studying the realism of co-simulation depending on the resolution used. By integrating PSLF and NS-2, GECO studies the impact of the communication layer on system performance, co-simulation scalability and protection scheme validation. GECO is not available in open source.

INSPIRE [39] integrates DIgSILENT, Powerfactory and OPNET or Modeler to co-simulate smart grid scenarios with advanced performance studies, with an important focus on the study of delays during communications. The project, closed source, integrates new

modules or applications. Globally, the authors show the impact of a high latency communications network on the power grid, via a controller.

Table 1: Overview of the existing co-simulation platforms.

Standalone projects	**Platform name**	**Power System Simulator**	**Network Simulator**	**Co-simulation environment**	**Time strategy**	**Last update**	**Built-in security module with UI**	**Availability**
[35]	**EPOCHS**	PSLF	NS-2	IEEE 1516-2000 (HLA)	Fixed time-step	2006	N/A	Closed source
[36]	**TASSCS**	PowerWorld	RINSE	Ad-hoc	Ad-hoc	2011	Compromised HMI attack, DoS Attack, MITM, TCP attacks	Closed source
[37]	**PowerNet**	Modelica	NS-2	Time stepped	Time stepped	2011	N/A	Closed source
[38]	**GECO**	PSLF	NS-2	Ad-hoc	Discrete event	2012	N/A	Closed source
[39]	**INSPIRE**	DIgSILENT, Powerfactory	OPNET, Modeler	IEEE 1516-2010 (HLA evolved)	Dynamic time stepped	2013	N/A	Closed source
[40]	**GridSpice**	MATPOWER, GridLab-D	N/A	N/A	Fixed time-step	2014	N/A	Open source
[41]	**DACCOSIM NG**	None	Modelica	Time stepped IEEE 1516-2000 (HLA)	Time stepped	2018	N/A	Open source
[42]	**ASTORIA**	PYPOWER	NS-3	Mosaik	Discrete event	2016	Malicious Software Infection Attack, DoS Attack	Closed source

Table 1: Overview of the existing co-simulation platforms.

[43]	**MECSYCO**	Modelica, EMTP-RV	NS-3, OMNeT++, OPNET	DEVS	Discrete event	2018	N/A	Open source
[44]	**SCADASIM**	MATLAB / Simulink	OMNeT++	Ad-hoc	Ad-hoc	2019	DoS Attack, MITM, Eavesdropping, Spoofing	Open source
[45]	**ERIGrid**	MATLAB, PowerFactory	NS-3	Mosaik, FMI-compliant simulation coupling	Dynamic time stepped	2019	N/A	Open source
[46]	**GridAttackSim**	GridLab-D	NS-3	FNCS	Fixed time-step	2019	Channel jamming, Malicious code, Injection attack, Replay of messages	Closed source
	ASGARDS-H	GridLab-D	OMNeT++	HELICS (ZeroMQ, HLA compliant)	Dynamic time stepped	2020	Replay Attack, FDIA, DoS/DDoS, Integrity, Eavesdropping, MITM, Spoofing, TCP attacks, Power attacks, Attack sequences	N/A

Table 1: Overview of the existing co-simulation platforms.

Co-simulation frame-works	**Platform name**	**Power System Simulator sup-ported**	**Network Simulator supported**	**Co-simulation environment**	**Time strategy**	**Last up-date**	**Built-in security module with UI**	**Availability**
[47]	**FNCS**	GridLab-D, PowerFlow	NS-3	ZeroMQ (HLA compliant)	Discrete event	2018	N/A	Open source
[48]	**HELICS**	GridLab-D, MATPOWER, GridDyn, OpenDSS, PSLF, InterPSS, FESTIV, Opal-RT	NS-3, OM-NeT++	Ad-hoc	Dynamic time-stepped	2020	N/A	Open source
[49] [34]	**Mosaik**	PYPOWER, IPSYS, Opal-RT, Jpower, PYPower	MATLAB, OpenFire	Conservative	Discrete event	2020	Quantum key distribution	Open Source

GridSpice [40] is an open-source project integrating MATPOWER and GridLab-D, with the help of a Python REST-based framework for simulation control, allowing the project to run on the Cloud. GridSpice provides a user interface for modeling new models as well as support for security principles (User authentication, Access Control Lists, Cluster Management), databases, and an administration console. Overall, the project enables the advanced study of various control scenarios within a smart grid. However, the realism of communications is not achieved since no computer network simulator is integrated in the project.

DACCOSIM NG [41] is a project developed for the French electricity network operator (EDF), oriented on multithreaded and distributed execution. The integration of various programming languages or tools (Java, Dymola, Matlab, Java) extends the first version of DACCOSIM with the possibility to co-simulate more advanced models. The project responds to concrete problems that operators in the real world have. The project has not been updated since 2018.

ASTORIA [42] is a framework to simulate attacks within smart grids. Built around Mosaik (PyPower) and NS-3, it provides a minimalist platform to exploit the modularity of NS-3 and Mosaik to implement advanced scenarios for the study of attacks. The original publication focuses on the malware infection of the SCADA system and the denial of service (DoS) attack.

FNCS (Fenix Framework for Co-Simulation) [47] is jointly developed with GridLab-D at PNNL and synchronizes NS-3 with GridLab-D to provide an advanced co-simulation environment. Offering complete documentation and support for many languages and operating systems, FNCS is the solution of choice for realistic and fast co-simulation, with a 20% speed gain. Initially proposed as a simple co-simulation framework, FNCS natively integrates many examples for co-simulation of cyber-physical systems such as smart grids. The project has recently been abandoned in favor of a new co-simulation framework, HELICS, which will be detailed in this Section.

MECSYCO [43] (Multi-agent Environment for Complex SYstem CO-simulation) is a co-simulation platform project based on Moedelica and extended during the MS4SG (Multi-agent Multi-Model Simulation of Smart Grids) project to include ns-3 and OMNeT++. The project studies how multi-agent can meet the needs of co-simulation in the energy field, and in particular smart grids. Although the documentation is not complete, the project is proposed in Java and C++.

SCADASim [44] is an open source education-oriented project for the co-simulation of SCADA systems where many sensors are evolving. The platform integrates MATLAB and OMNeT++ as well as a Postgres DB database for dynamic data storage. The project focuses on the simulation of malicious attacks: Denial of Service attack and Spoofing attack scenarios focused on Modbus protocol using PLC communications and HMIs.

ERIGrid (European Research Infrastructure supporting Smart Grid Systems Technology Development) combines ns-3 and Mosaik (PyPower) in order to set up an FMI-compliant interface with ns-3. However, as indicated in [50], the project seems to use only dummy devices to simulate the different interactions (grid delay, congestion), which limits the possibilities permitted by ERIGrid. Many examples are publicly accessible via their GitHub repository.

GridAttackSim [46] is an FNCS-based framework (GridLab-D and ns-3) focused on co-simulating attacks on smart grids, with a strong emphasis on Dynamic Pricing capabilities. The project aims to provide a modular and user-friendly platform to visualize and study the mutual impact of attacks (Channel jamming, Malicious code, Injection Attack, Replay attack) on the computer and power system. This project represents one of the major candidates for researchers interested in the co-simulation of CPS attacks in smart grids.

HELICS [48] (Hierarchical Engine for Large-scale Infrastructure Co-Simulation) is a co-simulation platform developed by the Grid Modernization Laboratory Consortium (GMLC) as an enhancement of FNCS. The project is being actively developed and already proposes the inclusion of dozens of different simulators. The project is distinguished by its scalable aspect (100,000+ federates) and its compatibility with many programming or scripting languages. Not limited to smart grids, HELICS natively co-simulate multiple CPS systems via ns-3, OMNeT++, Gridlab-D and OpenDSS.

Mosaik [49] is a Python alternative co-simulation platform-oriented for smart grids. It natively embeds a kernel for synchronization but also PyPower, Python's electrical simulator. Through a simple and documented set of methods (API), Mosaik allows the integration of many simulators. In the framework of this thesis, as a preliminary work, we were interested in the integration of quantum key distribution (QKD) within a microgrid [34].

3.3 Strengths and Limitations of Existing Platforms

In this Section, we compare the different existing co-simulation platforms and motivate the choice of HELICS as the co-simulation framework for our project.

Figure 3 presents an advanced comparison made against the associated claimed capabilities offered by the different solutions with a focus on Smart Grid applications from the European Technology Platform Smart Grids [51]. Overall, the majority of the solutions have been proposed but have not received an update subsequent to their publication, which, combined with the closed-source aspect, causes the academic community to abandon these solutions. Furthermore, although native support for the study of cyber-attacks against smart grids is increasing over time, this support remains limited and not modular enough, preventing the complex study of systems with realistic scenarios and thus allowing the generation of datasets that can be used for machine learning.

The *Performance* and *CPA studies Completeness* columns have been defined according to the claims of the original, associated research papers, and respectively define the capacity of each platform to allow the co-simulation of large systems and their ability to allow, integrate or study Cyber-Physical Attacks.

Among the most complete solutions today aiming at the same objectives as our project, we can name *GridAttackSim*, *SCADASim* and *TASSCS*. *SCADASim* and *TASSCS* have brought and shown the interest of co-simulation in the context of CPA within smart grids, implementing different attacks and protocols of smart grids, such as Modbus. *GridAttackSim* has enhanced these contributions by providing the first visually usable framework for studying CPA in different aspects of smart grids, with a focus on the education aspect.

However, if we were to highlight the limitations of these projects, we would emphasize the following points:

- The mentioned solutions do not propose an advanced method for the simulation of complex scenarios, mixing device lifecycle, power grid faults, multiple-point, multiple-time cyber physical attacks with support for modularity (i.e., advanced definition of the network capabilities, attacks and events based on temporal and spatial parameters).

- None of the solutions natively proposes a method allowing the study of systems or topologies different from those managed natively.

- The support for the variation of simulation parameters in the context of computer communication networks is limited, without the support of advanced technologies (i.e., 4G LTE) or means to easily parameterize the network and the different protocols used.

- Time management is not modular, and most platforms require a fixed co-simulation speed.

- The lack of modularity of the projects does not help the user easily integrate new simulators or controllers without a thorough knowledge of the system and offers limited methods for real-time control of the system by a user.

Finally, the choice of a co-simulation platform rather than an existing testbed as a basis for our work naturally emerged. Thus, we isolated three potential co-simulation framework solutions: *Mosaik*, *FNCS* and *HELICS*.

Mosaik offers a modular approach, in Python, that integrates multiple simulators and hardware (HIL). We used *Mosaik* in a preliminary project before ASGARDS-H, with the study of quantum key distributions schemes in smart grids, focused on the demonstration of attack detection, in particular eavesdropping.

FNCS is the antecedent of *HELICS*, leading us naturally to abandon it in favor of *HELICS*, actively developed.

The selection procedure of the platform was made according to the following criteria:

- **Maintenance**: The interest of the developers in the project as well as the frequency of updates and the project roadmap.

- **Quality, Capabilities**: The quality of the project's code as well as the different functionalities offered.

- **Documentation**: The presence of a complete and simple documentation to simplify the use of our project by researchers who do not master programming or system configuration.

- **Completeness**: Management of advanced synchronization or data exchange mechanisms.

The main reason why we chose *HELICS* over *Mosaik* is its distributed synchronization capability and its high performance, which is unattainable with the Python language.

HELICS is now the most advanced project in terms of co-simulation, is open-source and have a permissive license. Also, *HELICS* proposes a set of APIs for different languages, allowing the integration of a larger set of simulators.

Note: ● *is better*	Performances	Application timespan (s)	Research Area	Real time	HIL	User Interface	Open Source	Application Grid size (bus)	Maintenance	CPA studies Completness	Capabilities	Research Interest
Project Name												
EPOCHS Electric Power and Communication Synchronizing System	●	◔	State Monitoring; Abnormal conditions, defense and restoration;	○	○	○	○	◑	○	◔	Agent-based power system protection scheme; Protection scheme for transient power system Stability; Wide-area monitoring;	●
TASSCS Testbed for Analyzing Security of SCADA systems	●	◑	Anomaly-based detection of SCADA Attacks, SCADA security analysis and evaluation testbed	●	●	●	○	●	○	◕	PLC simulation (MODBUS Server); MITM; Spoofing; DMZ; Modbus attacks; DoS attack;	●
PowerNet A communication framework for the smart power grid	◔	◔	Basic, real-time synchronization	●	○	○	○	◔	○	◔	Real-time voltage studies and simulator synchronization;	◑
GECO Global Event-Driven Co-Simulation Framework for Interconnected Power System and Comm. Network	●	●	State Monitoring; Abnormal conditions, defense and restoration;	○	○	○	○	◑	○	◔	Protection schemes, co-simulation synchronization methods, Agent-based remote backup relay protection scheme;	●
INSPIRE Integrated co-simulation of power and ICT systems for real-time evaluation	●	◑	State Monitoring; Abnormal conditions, defense and restoration; HVDC Grid based system	●	●	○	○	◑	○	◔	Evaluation of the real-time performance of wide-area monitoring, protection and control; impact of communication delays;	◑
GridSpice A Platform for Modeling, Analysis, and Optimization of the Smart Grid	◑	●	Interactions and responsibilities between distribution grid operators and other stakeholder; Sustainable system operations and low-level system user dispatching; Electric Vehicle (EV) integration into Distribution systems	○	○	●	●	●	○	◔	Optimal placement of solar panels in distribution networks, considering transmission network effects	◑

Figure 3: Co-simulation platform comparison

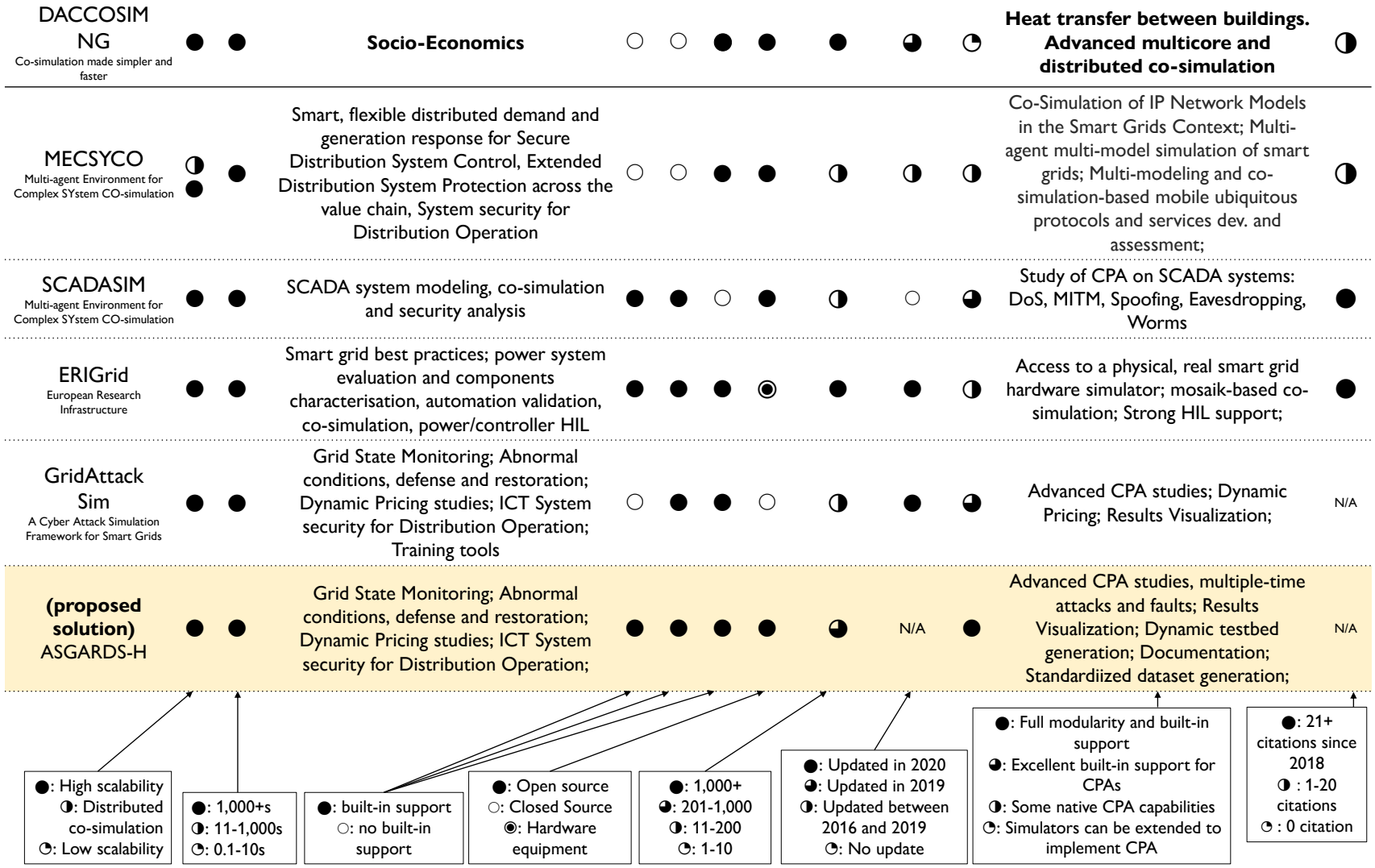


Figure 3: Co-simulation platform comparison

Chapter 4

Project Platform Overview

This Section presents the formal requirements for ASGARDS-H. It first provides an overview of the technologies used and objectives of this project, as well as the various aspects that will be detailed throughout this Thesis.

4.1 HELICS Framework

HELICS is a co-simulation framework developed in C++. It is interesting, before going into the details of the framework, to motivate why we chose HELICS among numerous available solutions and we explain the different parts that constitute an environment of co-simulation with HELICS.

4.1.1 Motivations

In this Section, we illustrate how HELICS performs according to the co-simulation platform selection criteria.

- **Maintenance**: HELICS is actively developed with 22 contributors and more than 5 project updates (commits) every weeks. The project roadmap is now mentioning the future HELICS version 3.x which promises many more improvements for the future.
- **Quality, Capabilities**: HELICS offer strong support for multiple programming language, and offer advanced tools and modules to set-up, configure and run co-simulation. The $GMLC-TDC$ GitHub repository contains 36 sub-repositories offering wide support for different simulators, scenarios and tools.

- **Documentation**: HELICS offers a complete *ReadTheDocs* documentation for both Developers and Users.
- **Completeness**: HELICS supports advanced synchronization mechanisms as well as communication capabilities such as co-simulators filters, delays and routes.

4.1.2 HELICS Terminology

In this Section, we present six HELICS-related concepts.

First, the **Federate** is an individual simulator that is connected to the co-simulation environment and communicates with them, either by sending or receiving data, or both. We highlight here the difference between a simulator and a federate: a federate is in all respects equal to a simulator, except that it is part of the co-simulation environment (networked simulator).

The second element is named **Core**. It is a module added to a federate to connect it to the co-simulation environment. This Core can be developed in any programming language, as long as it is compliant with the HELICS API or the HLA standard.

The **Broker** is an entity that links and coordinates multiple cores or brokers. It is indeed possible to have multiple levels of brokers, allowing distributed co-simulation with different machines.

The **Root Broker** is unique and corresponds to the main broker managing the different federates forming a federation.

A **Federation** is a set of federates operating together in co-simulation. It is possible to have several independent systems consisting of several federates running in parallel within a co-simulation.

An **Interface** is a way for federates to communicate with other federates. These interfaces include Endpoints, Publications, Filters, and Inputs.

The Figure 4 illustrates the above-presented concepts with a realistic architecture of a distributed smart-grid co-simulation with HELICS.

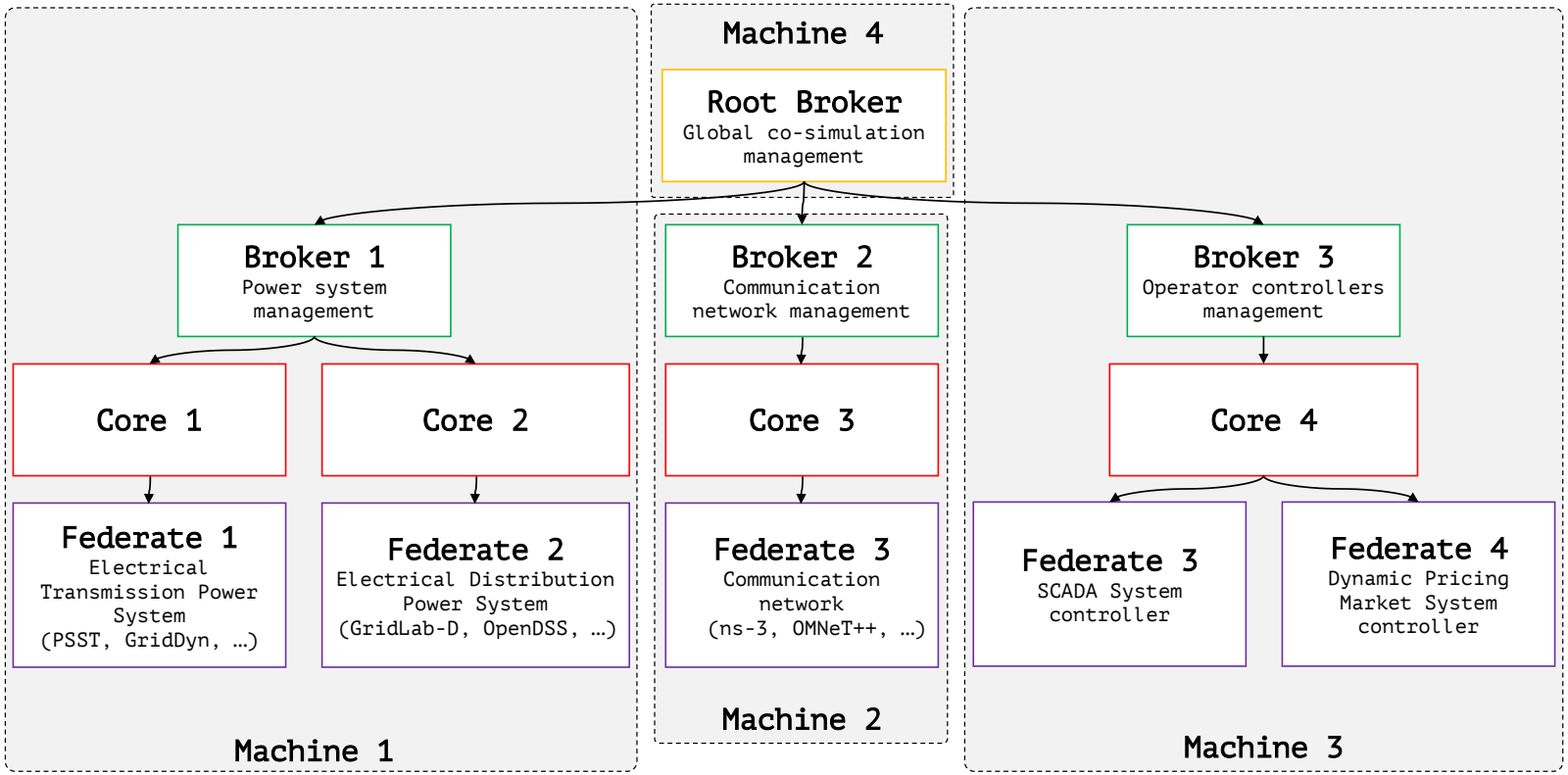


Figure 4: Example of a HELICS smart-grid oriented federation

4.1.3 Federate Communications

There are different types of federates. The first, **Value federates** are fixedly connected to the other federates, while **Message Federate** have no constraints and can be used to integrate an event-based dimension within the co-simulation environment, without timestep constraints. It is also possible to create a federate of both types at the same time. These types will define how the federates will manage to exchange information with other federates.

In the context of **Value federates**, the data (inputs/publications) will follow a fixed route and embed the associated unit. This kind of message is mainly used for sharing simulated data from a simulator such as GridLab-D to OMNeT++. On the other hand, **Message federates** exchanges follow a fixed route during co-simulation and are encapsulated in a blob format (opaque), with a source/destination mechanism similar to what a simulator like OMNeT++ simulates within communication networks. Message federates allows to add events within the co-simulation.

In the context of specific simulations with a realistic aspect, HELICS also introduces a system of filters for the simulation of events at the level of information transmission: time delay, firewall, rerouting, random dropping, etc. For example, a filter can be used to reroute control messages from OMNeT++ to GridLab-D.

To enable communication between different machines, HELICS proposes the use of the ZeroMQ protocol for the transmission of information.

4.1.4 Input and Publication Messages

Inputs and publications make a pair. From a federate point of view, an input corresponds to a value recovered from the co-simulation environment, to which the federate has previously subscribed via its core. A publication will correspond to data that the federate can send to other federates.

At any time within the co-simulation environment, multiple data will transit. The data can be locally or globally accessible, depending on the needs. In addition, different types of labelled data are proposed: integers, booleans, strings, etc., allowing the transport of information in addition to the value exchanged.

4.1.5 Hardware-in-the-loop Design

In the context of co-simulation, it may be interesting to include real equipment in order to study its behaviour in a larger scale simulation. HELICS natively permit the addition of Hardware-In-The-loop (HIL) equipment.

For example, the Pacific Northwest National Laboratory recently demonstrated the use of HELICS with HIL capability. Indeed, HELICS uses ZeroMQ and is compatible with equipment using High Level Architecture (HLA) for the quick the integration of compatible equipment. For this purpose, they have integrated a real building with VOLTTRON [52] in their market-based control technique co-simulation.

4.1.6 Co-simulation Configuration

There are three important parameters for the configuration of federates in HELICS:

- **Uninterruptible**: This parameter will fix the timestep to make sure the federate will operate at fixed intervals even if new data arrives earlier;
- **Period**: Almost all simulators have a minimum time-resolution. This parameter can be used to synchronize the federate by considering its time-resolution. For example, GridLab-D uses a user-settable time-resolution that can be correctly handled by HELICS;
- **Offset**: In some realistic case, introducing synchronization delays to better simulate physical interactions between federates induce time grants that are offset slightly in time.

4.1.7 Synchronization

Dynamic time stepped is a modular way to synchronize simulators that evolve differently over time. HELICS regulates the time of each federate individually via a request system from each federate. It is an iterative process of requesting co-simulation time managed by a central mediator who will control that a simulator will only move forward in time by the amount of time required for the co-simulation to work properly. For example, if GridLab-D is able to simulate the next second but OMNeT++ has not finished and needs the outputs of GridLab-D for the second it has just finished simulating, then the mediator

will pause GridLab-D until OMNeT++ tells the environment that it has finished simulating its second. This process is dynamic and can be split into two mechanisms. Firstly, a dynamic mechanism of queuing federates to ensure consistency of the co-simulation, and secondly the frequency at which these federates will need to communicate with each other. It is possible, for example, that two federates will communicate with each other every second, while one of them will have to communicate every millisecond with a third one. The methodology proposed by HELICS is a dynamic timestep simulation that consistently reports the events taking place within heterogeneous simulators.

When all simulators have indicated to HELICS that their simulation is complete, or that the co-simulation has reached the maximum simulation time configured beforehand, then the simulation will end and a termination cascade will take place to properly finalize the co-simulation.

4.1.8 HELICS Co-Simulation Lifecycle

Every co-simulation lifecycle begins with the model design step. Model design is a crucial step where the architecture and configuration of the co-simulation is realized. In the context of this project, this step is fully supported by our testbed generation interface.

The second step corresponds to the configuration of the federates. Again, our project supports this capability natively. The configuration of the federates defines which are the inputs/publications as well as the different routes that the exchanged data will follow. The configuration of the federates can be completed using a JSON file where are specified the name of the federate, the type of core, the inputs/outputs, as well as the timestep value.

After this preliminary configuration, the co-simulation testbed environment is ready. The next step is managed by the Root Broker. All federates enter *initialization state* and communication channels are established. Also, it is very common that federates already started their simulation to reach a self-consistent state (or steady state) before the co-simulation really starts.

Then, based on the configuration above, all federate will run individually and request time to the Root broker. Depending on the specified timestep, the resolution will change. For example, OMNeT++ will retrieve information from GridLab-D every 0.2 seconds which can be shifted over time as a function of the performance of the communication network, while GridLab-D will update the values every millisecond.

According to the time requests of each federate, the broker will determine which one(s)

will be allowed to run. This is called *time granting*.

When a federate receives the green light from the broker, it can run until the next time request. This would mean that GridLab-D will get 200-time grants when OMNeT++ only receives one, in order to correctly simulate 200 milliseconds in a consistent way.

When the simulation time limit is reached, the federates will left and the broker will ensure that the co-simulation environment is terminating properly.

4.2 Project Objectives

ASGARDS-H is being developed as part of Ericsson's Ericsson Global Artificial Intelligence Accelerator (GAIA) program, and is proposed as a core solution for future studies on the development and detection of physical cyber attacks in smart grids using machine learning. It should allow a user-friendly use with complete and validated co-simulation for the generation of standardized dataset containing data from different sources and simplify the study of complex scenarios mixing cyber physical attacks, control systems attacks, power grid faults and network events. The objectives are as follows:

- **Completeness**: The project must implement all aspects that guarantee the validity of the co-simulation as a whole. This includes its ability to maintain synchronization between federates, the completeness of each simulator in their configuration and interconnections, as well as the integration of protocols, devices, scenarios, attacks and aspects of smart grids.

- **Automatic testbed generation**: In order to permit an easier use of the project for researchers not initiated to programming, the project must generate multiple testbeds. The generation should be based on user-specified inputs, configuration files, and the model of the electrical network that will be used as a basis for the testbed.

- **Accessibility**: The addition and modification of new capabilities must be documented and accessible without requiring too much modification to the project. In addition, the use of a testbed must be possible for each stakeholder without requiring advanced knowledge of cyber attacks, co-simulation or system.

- **Data validation and dataset generation**: The project must generate valid data, i.e. be able to co-simulate a smart grid system correctly and embed all the serializers and

methods to generate files from different sources and formats. The dataset generation must be made possible at the end of a co-simulation, and this in a dynamic way from the output files of the co-simulator.

- **Modularity**: The software architecture of the project must be modular to ensure (1) an advanced use of the implemented capabilities in order to design precise scenarios and (2) to simplify the integration of new federates within the system.
- **Strong events support**: The platform must facilitate the study of complex scenarios that can mix cyber attacks, power system faults and network events. Also, the combination of these elements must be possible and configurable. Finally, the user must be able to interact with the co-simulation in real time through a dedicated federate in order to manually control attacks and events.
- **Good performances**: Although this project integrates many functionalities distributed through different federates, it is important to guarantee, in the worst case, an acceptable performance that is at least as good as real time. This performance can be relaxed in special cases where the scenario is about high time resolution (subsecond) or specific attacks (DDoS).
- **User-friendliness**: The project Generator as well as the visualization interface and the attack federate must be easy to use, modern and accessible for all users.
- **Visualization**: The visualization federate must display in real-time the power grid as well as the computer network in order to visualize the system status over time.

The project is structured around two main components: the project generator and the dynamic co-simulation platform. The next Section details, for each aspect of the project, the various characteristics and requirements of each of the aspects.

4.3 Formal Requirements

The project requirements are detailed here and formally define the objectives we were aiming for this project.

4.3.1 System-level Requirements

These system requirements define the supported operating system, the libraries used and the compiler. The Table 2 summarizes the system-level requirements of our project.

System-level requirements	
Objective: define the operating environment of ASGARDS-H	
Operating System	Linux (Debian)
Programming languages	C++ (federates), Python (Data processing and federates), Javascript (Visualizer) and QML (User Interface)
Third party libraries	Conda for Python 3.7, HELICS shared API, simuLTE, INET, HELICS CLI
Compiler	gcc, g++
Co-simulator Framework	HELICS v2.2.2+

Table 2: System-level requirements

The project is developed for Linux, although it embeds projects that are all equally compatible with Windows. The choice of the Linux platform is here determined by performance needs and better control of the development environment. Also, the QML language has been chosen to realize the User Interface (UI) in order to take advantage of hardware acceleration and to propose a clean, modern and user-friendly UI.

Moreover, we define in Figure 5 the testbed features.

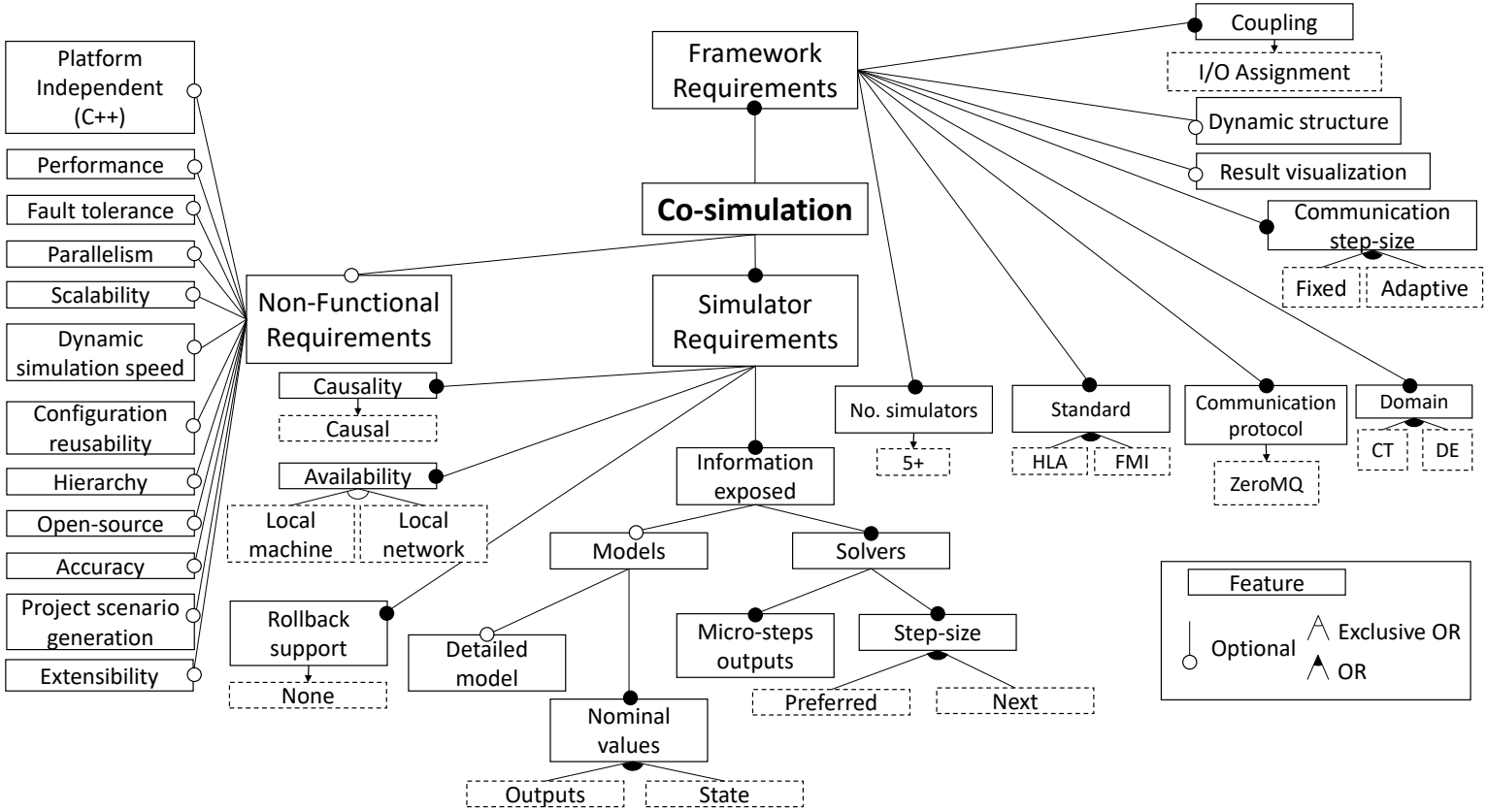


Figure 5: ASGARDS-H features

4.3.2 Co-simulation Architecture-level Requirements

The Table 3 summarizes the architecture-level requirements of our project. The project architecture will be discussed in more detail in the 4.4 Section.

Architecture-level requirements	
Objective: define the federation structure and behavior	
Power Grid Simulation	GridLab-D v4.1+ (develop branch)
Communication network simulator	OMNeT++ v5.6.1+ with INET v4.2.0+ and SimuLTE
Power grid control	Modular python v3.7+ federate with HELICS
Market control	
Visualization (server federate)	
Visualization (front-end)	Javascript using Server-sent-events (SSE) and D3JS
CPS attacks	Multi-processed and multi-threaded python v3.7+ Meterpreter command-line interface program

Table 3: Architecture-level requirements

In addition to the GridLab-D and OMNeT++ simulators, we offer the use of customizable Python scripts to implement any form of mechanism within the various controllers. This choice of design is motivated by the need to include machine learning techniques or attacks directly at the controller level. These special federates are dynamically connected to the co-simulation environment.

4.3.3 Project Generation-level Requirements

The project generator is an advanced wizard that generates a functional and complete co-simulation platform from the user-defined parameters. This aspect of the project is detailed in Table 4 and will be described in more detail in the Chapter 4.5.

4.3.4 Project Capabilities Requirements

We define here the project capabilities requirements. The information proposed in the Table 5 presents the basic functionalities that the project embeds. However, it is possible to extend these functionalities.

Testbed generation-level requirements	
Objective: define the project generator capabilities	
Input	A GridLab-D GLM model file with the support for nodes, triplex nodes, meters, triplex meters, lines, overhead lines, underground lines, line spacing, load, triplex load, Switch, substation, generators, auction, controller and house objects
Output	A fully functional co-simulation testbed with dynamically generated node placement, network topology, federates configuration and input parameters
Parameters	**(1)** Co-simulation global parameters (co-simulation timestep, GLM model, random seed, weather file, federates activation or disabling). **(2)** OMNeT++-specific parameters (OpenADR, Synchrophasor, Firewall, communication data rates, PCAP file generation, random noise, Control Center). **(3)** Co-simulation profiles (operator, defender and attacker) with user-defined accessible and monitorable resources for each profile). **(4)** Output Dataset configuration (recording parameters, frequency and statistics collection). **(5)** Advanced event configuration (power system faults). **(6)** Cyber physical attacks, hooks and specific-node parameters configuration (attack sequences, packet dropping, replay attack, packet delayer, integrity attack, eavesdropping, desynchronization, dynamic pricing attack, statistics outputs)
UI	A Qt QML-based user interface using Python v3.7+ with scenario saving and loading capability in JSON format with a dynamically generated PDF file that summarizes the scenario

Table 4: Testbed generation-level requirements

4.3.5 Devices Requirements

The project must natively facilitate the study of advanced scenarios of smart grids. For this purpose, a number of key features are implemented within the OMNeT++ communication simulator. New equipment can be added simply by leveraging the modularity offered by OMNeT++ and INET. The Table 6 presents the device requirements within the co-simulation.

Project capabilities-level requirements	
Objective: define the considered testbed capabilities that are used for data generation	
Simulation time resolution	Subsecond mode (DELTAMODE), real-time mode and dynamic timestep
Smart grid capabilities	OpenADR protocol (Demand-Response) over HTTP, Synchrophasor protocol (IEEE C37-244-2013), Dynamic Pricing market (auction system), Power grid control
Co-simulation dynamics	Weather capability (TMY3 file), OMNeT++ lifecycle events

Table 5: Project capabilities-level requirements

Devices requirements	
Objective: define implemented devices capabilities	
OpenADR VEN	Virtual End Nodes monitor smart meter-related data as defined in the operator profile. It acts as a HTTP server
OpenADR VTN	OpenADR Virtual Terminal Node is an HTTP browser, either an Aggregator (VTN/VEN) or the Demand-Response Service provider (Control center)
PMU	Phasor Measurement units monitor the power grid frequency, voltage magnitude and phase angle. It uses the IEEE C37-244-2013 protocol and is connected to a Phasor Data Concentrator (PDC)
PDC	Phasor Data Concentrators aggregate all received data from one to many PMUs before forwarding it to the Control Center
Dynamic Pricing	Acts as a basic TCP client that sends captured market bids from GridLab-D to the Control Center. The CC then forward all bids to the Market federate.
Firewall	A multi-protocol rule-based firewall using a ip-table like configuration file. Protects the Control center
Control Center	The Control Center implement both OpenADR, Synchrophasor and Dynamic Pricing capabilities.
eNodeB	In 4G LTE mode, this device is the antenna that receive and transmits messages from the user equipment (UEs) to the Packet Data Network Gateway (PGW)
UEs	UEs correspond to sensors and end-user devices that use the 4G LTE wireless network to communicate with the Control Center
PGW	This equipment routes messages from the 4G network to the operator's network

Table 6: Devices requirements

4.3.6 Networking protocols

Within OMNeT++, we choose to use Transmission Control Protocol (TCP) for data monitoring (HTTP over TCP, and TCP client-servers). However, the UDP protocol is used for wireless communications within the 4G LTE network. Equipment are interconnected using the Point-To–Point (PPP) protocol of the data link layer of the OSI model.

The choice of the TCP protocol is motivated by the realistic aspect of co-simulation. For a physical cyber system such as smart grids where availability is important, it is very unlikely that the UDP protocol, which does not guarantee the acknowledgement of sent messages, will be used. TCP thus makes it possible to study network performance in more detail.

4.3.7 Implemented Model Files

As default study models, we propose and implement four models in GLM format of various sizes. These are summarized in the Appendix A. The integration of new models is possible, provided that they correctly implement the modules defined in the Table 4. Different variants of these models are proposed in order to support a greater variety of scenarios.

4.4 Co-simulation Testbed Architecture Overview

In this Section, we detail the architecture of the testbed as a whole. The detail of the architecture can be found in Figure 6.

First, the **Project Generator** will generate different modules: GridLab-D Federate, OMNeT++ federate, Controller federates and Attacker federate. This generator embeds a set of rules and methods to dynamically generate, in addition to the specific configurations of each federate, the HELICS configuration files in JSON format to correctly interconnect and synchronize the federates between together.

Second, the **HELICS Broker** is dynamically executed when starting the co-simulation with the HELICS CLI tool. The latter controls all the federates and routes the different messages exchanged according to the project configuration. Three profiles are available and are managed invisibly by the broker. The Operator profile is there to define the data considered during the co-simulation. Indeed, adding or removing parameters to the operator will

change the behavior of the smart meters, the visualization federate and impact the data generated during the co-simulation. The Defender profile will mainly impact the visualization by filtering the information available. In the same way, the attacker profile realistically simulates an attacker who has no knowledge or access to the system. The attacker profile directly impacts the output of the co-simulation when using the Eavesdropping attack, the visualization federate, and the Attacker federate.

Around the broker, 6 federates are interconnected and synchronized:

- **Power Federate**: GridLab-D is connected to HELICS through the *helics_msg* library of the *connection* module. It is configured using the *model_file.json* file and offers multiple functionalities described in the Figure 6. This federate directly exchanges information with OMNeT++ and receives commands from both controllers (Market and Power Grid).

- **Cyber Federate**: OMNeT++ is connected to HELICS using the HELICS shared library via a *helics_helper* object which is a singleton shared by all the simulation equipment. This federate is also dynamically configured via a configuration file generated by the Project Generator.

- **Attacker Federate**: Attacker Federate implements a reactive command line interface with auto-completion to let the user launch attacks on the system in real-time. This federate obtains information from GridLab-D and OMNeT++ and the user can exploit it for feedback control.

- **Controller Federates**: There are 2 controllers (Market controller and Power Grid Controller). The first one, dynamically included if the power grid has the capacity of the market, controls the GridLab-D auction object by retrieving data from the OMNeT++ communication network. The Power Grid Controller ensures the same objectives, with the difference that the data retrieved does not come from the Market but from PMU Synchrophasor measurements.

- **Monitoring Federate** or Visualizer, retrieves only the information from the co-simulation (read-only), and display them in real time within a web graphical interface.

Hardware devices can be connected to the co-simulation using HELICS capabilities through their standardised interface or via an ad-hoc developed core. Specific equipment

such as Opal-RT or real power network equipment (power-hardware-in-the-loop, PHIL) are already supported by HELICS. PHIL scenarios can help researchers better study real-world effect of CPS threats and advanced scenario at lower cost and without risk.

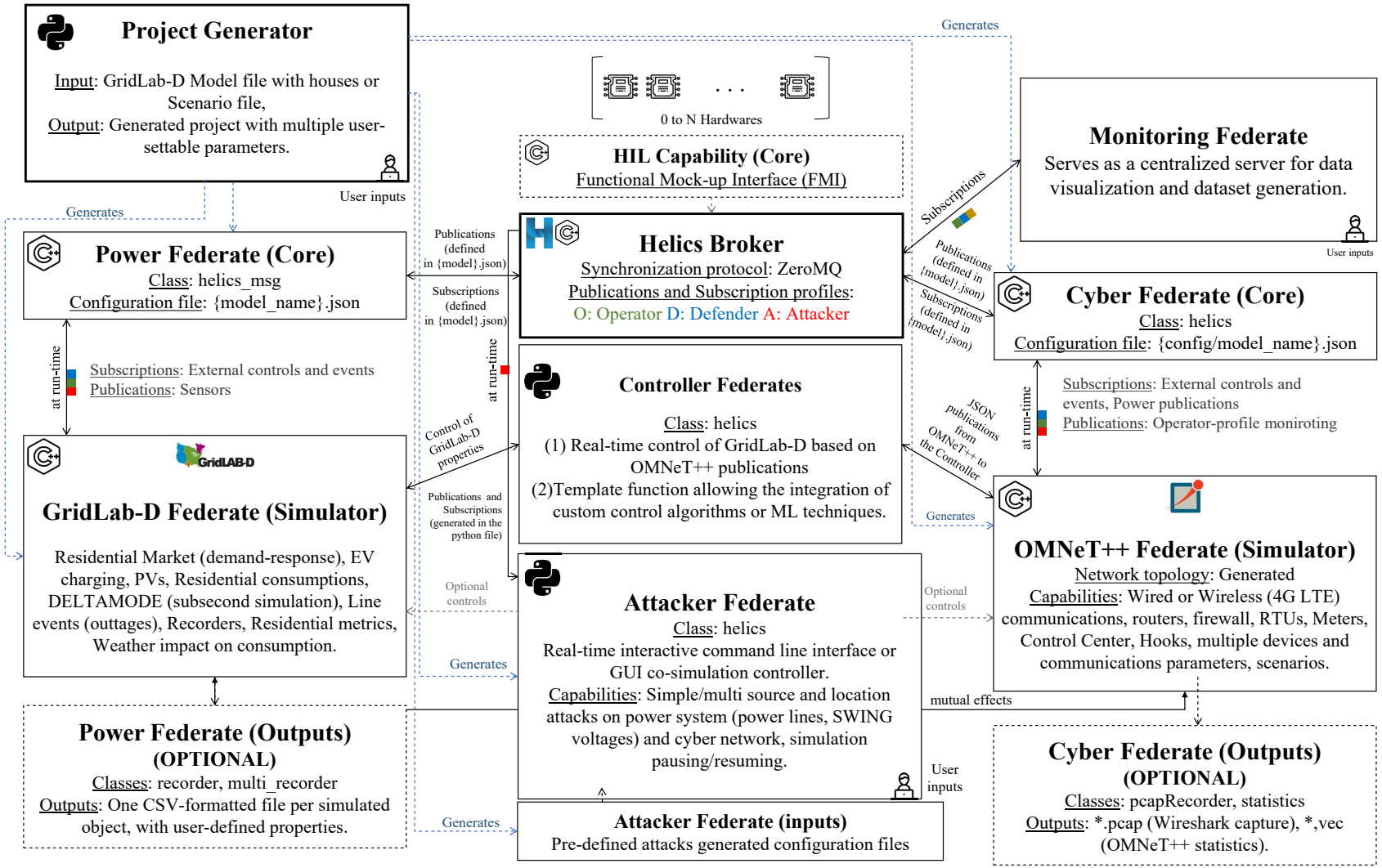


Figure 6: ASGARDS-H project architecture

4.5 Project Generator

The project generator is developed in Python 3.7 and is divided into two main modules:

Front-end Graphical User Interface: The user interface is developed in QML via PyQt5. The latter is visible in the Figure 7. The interface is designed to be very easy to use and is based on Qt's QML technology with the use of hardware acceleration. We can distinguish five menus:

- **Platform Overview** shows some useful metrics and information about the testbed once the project has been generated.
- **Steady-state Configuration** permits the user to configure several parameters for all federates that do not depend on the topology (e.g., the global latency of the OMNeT++ network or the firewall configuration).
- **Profile configuration, dataset generation** is a special menu for profiles configuration as well as dataset generation parameters. Profiles are configured through a matrix of checkboxes.
- **Scenario, instability and events configurations** is a configuration menu of events for GridLab-D (e.g., line faults). This menu is enabled after the project generation.
- **Firewall and Cyber-physical attacks configuration** is a menu for adding, removing or configure cyber attacks and enable the Attacker Federate configuration.

The testbed generation is a two-step procedure comprising the Project Generation step and the Project Build step. The first step objective is to prepare the testbed project folder by parsing the chosen GLM model file and applying steady-state configuration parameters. It updates the GUI by adding model-related information such as the list of network nodes to permit the advanced configuration of the testbed, addition of attacks and power grid faults.

The user can save the current interface settings and generate a PDF file summarizing the defined scenario. The interface uses the JSON format (*.cosim.json*) to store the information.

Back-end Testbed generator: The user interface communicates with a Python module in charge of testbed generation. For this purpose, it uses a common template that will be modified according to the options chosen by the user. The project generator uses mostly

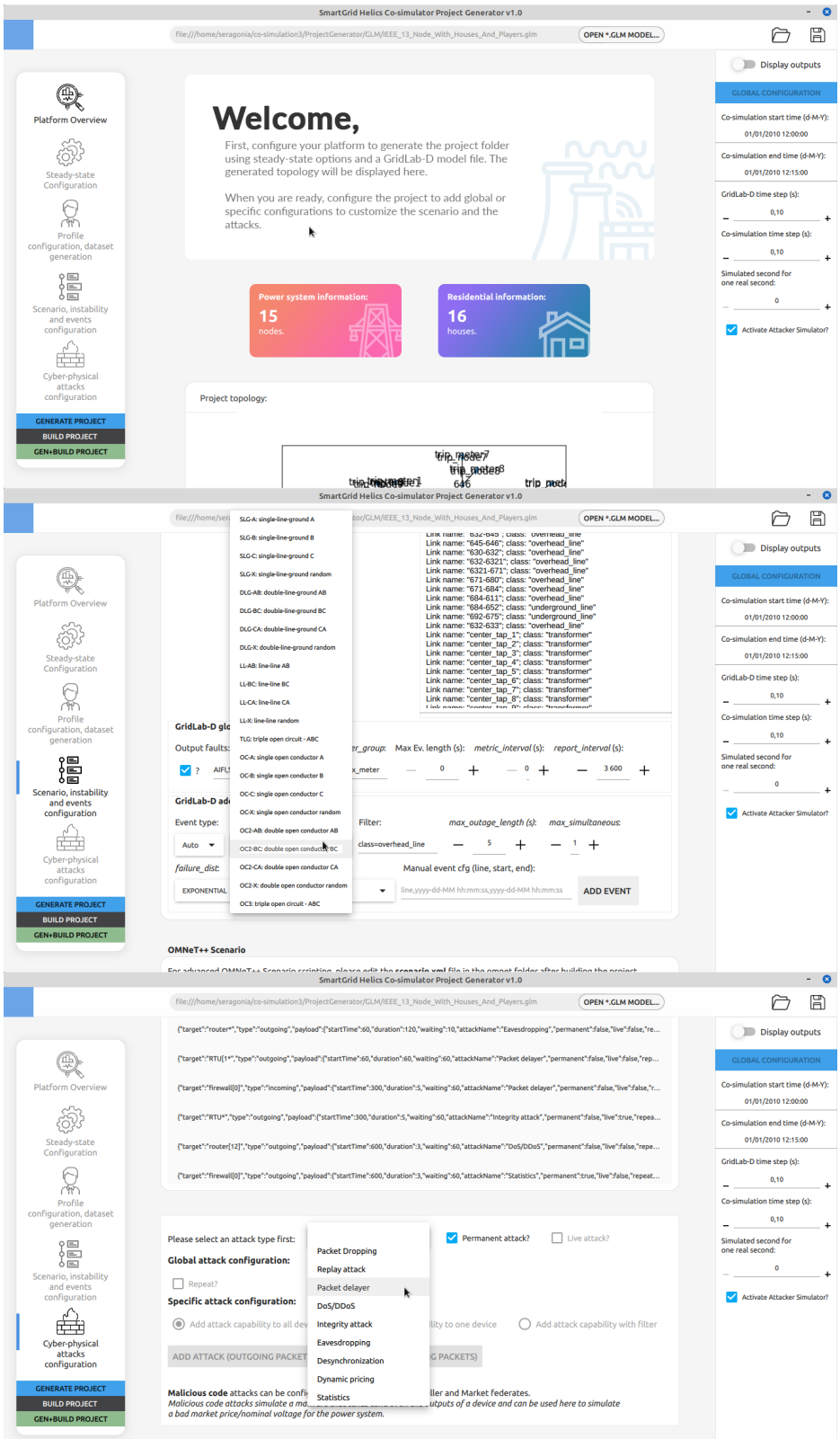


Figure 7: Graphical User Interface main window

dynamic regex to remove, add or modify aspects of the project. The functional structure of the Project Generator is detailed in the Figure 8.

The generation of a testbed relies on a set of options and a power grid model in GLM format (GridLab-D Model). The *Generate* function interacts with multiple *helpers*:

- **GLMHelper**: This is the main function of project generation. It calls the GLM-Parser class in order to retrieve all the information about the GLM file, and then generates multiple aspects of the project (network topology, OMNeT++ configuration, node placement, addition of devices specific to the cyber network, wired or wireless mode).
- **JSONGenerator**: This script generates the JSON HELICS configuration file of GridLab-D and OMNeT++ and also prepares the extraction of the different attack points on the power grid (lines, swing bus, etc.).
- **OMNETHelper**: Writes the JSON configuration file for OMNeT++ and the various parameters to the NETWORK.ned or omnetpp.ini file.
- **ATTACKHelper**: Writes to the federate Attacker to inform about steady-state attacks.

The building via the *BUILD* function adds configurations related to events and specific to the considered electrical network:

- **GLDHelper**: Writes GridLab-D related events to the GLM model file.
- **HookHelper**: Updates the testbed by including all related information about user-specifiec attacks.
- **DatasetHelper**: Appends dataset-related GridLab-D $group_recorder$ objects to the GLM model file for dataset generation.
- **ControllerHelper**: Fills the Controller template with a dynamic dictionary of subscriptions and publications according to the GLM model file.
- **MarketHelper**: If the Market capability is activated, this helper updates the template federate with its subscriptions and publications.

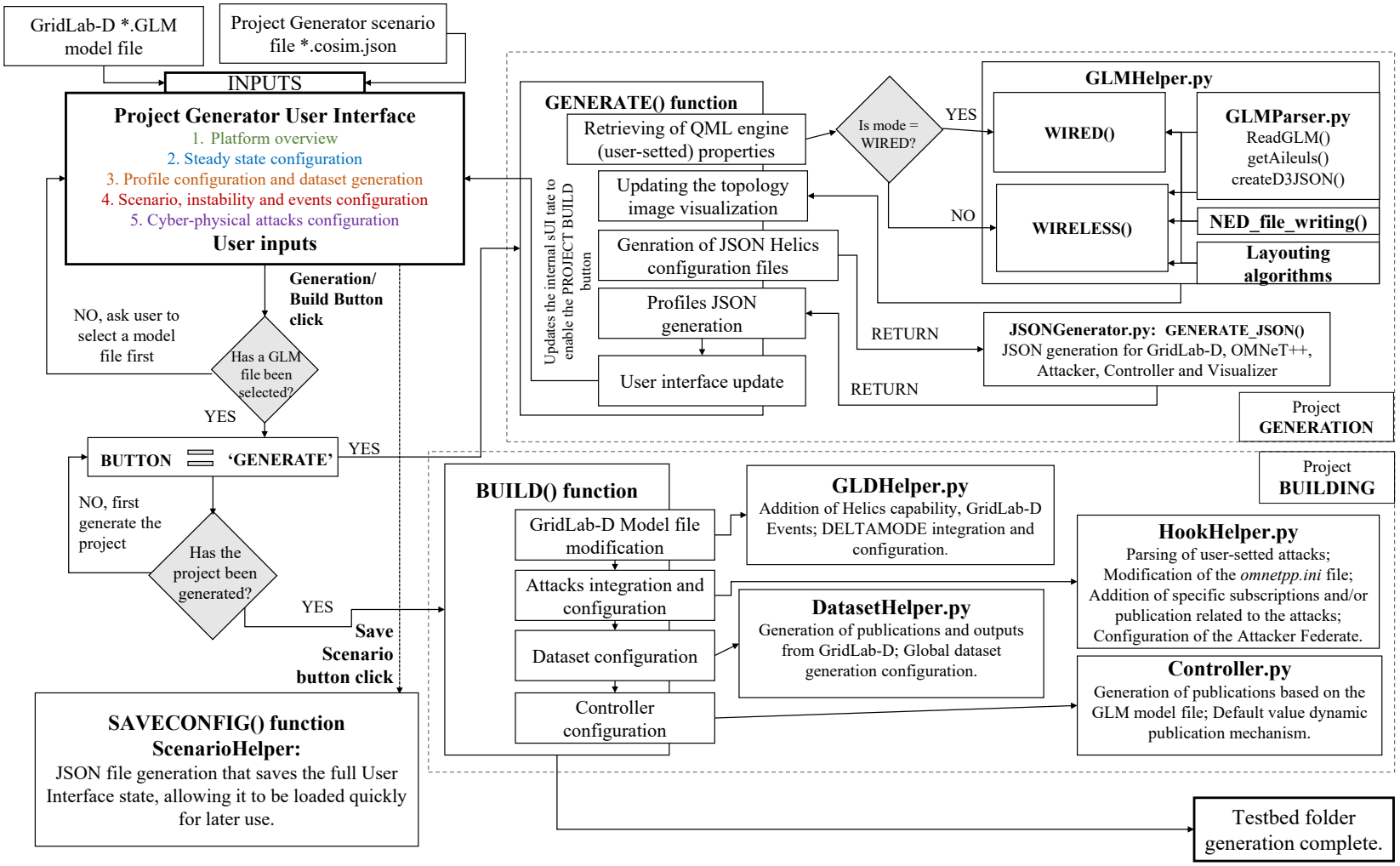


Figure 8: ASGARDS-H project generator architecture

4.5.1 GLM Model File Processing

The parsing of the GLM model used within the testbed represents the most important part of the testbed generation. This step is implemented by the *GLMParser* and is based on the work of J. de Chalendar [53] and extended to meet the functional needs of the project.

The GLM format is similar to the structure found in XML documents with the syntax of the C language, allowing, using regex, to retrieve the whole implemented topology in order to transform it into a complex python object. Thus, it becomes possible to extract each device from the network, and to perform logical operations such as retrieving the node grandfathers used for topology visualization. Globally, the parsing of the model is performed in the following way:

- Recovery of each device and connection lines, with all sub-properties, management of multiple objects and dynamically assigned ID when the *name* field is not filled in.
- Recovery of parents and ancestors
- Dynamic creation of a readable topology in D3JSON format
- Creation of a tree representing the topology and dynamic addition of aggregators according to the number of children implementing a smart meter or a PMU.

Figure 9 shows an example of a topology generated for the IEEE 13 Bus System model. *root* is considered as the main aggregator.

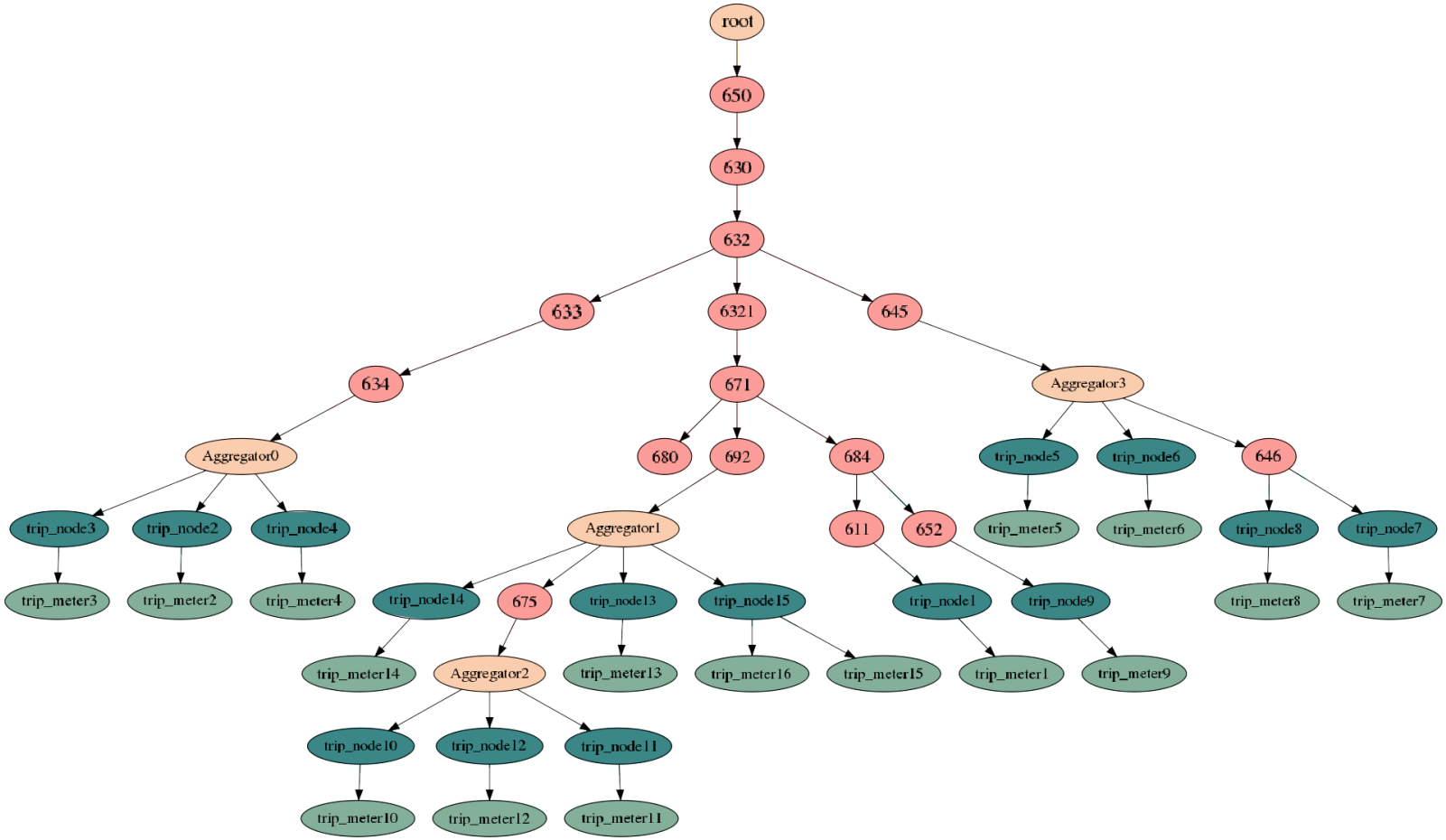


Figure 9: IEEE 13 Bus System generated topology tree with aggregators

4.5.2 Network Topology Generation

The second step in the testbed generation corresponds to the generation of a communication architecture based on the power grid topology. We propose two modes: Wired and Wireless. Each mode follows a common placement logic for each network equipment which is detailed in Figure 10.

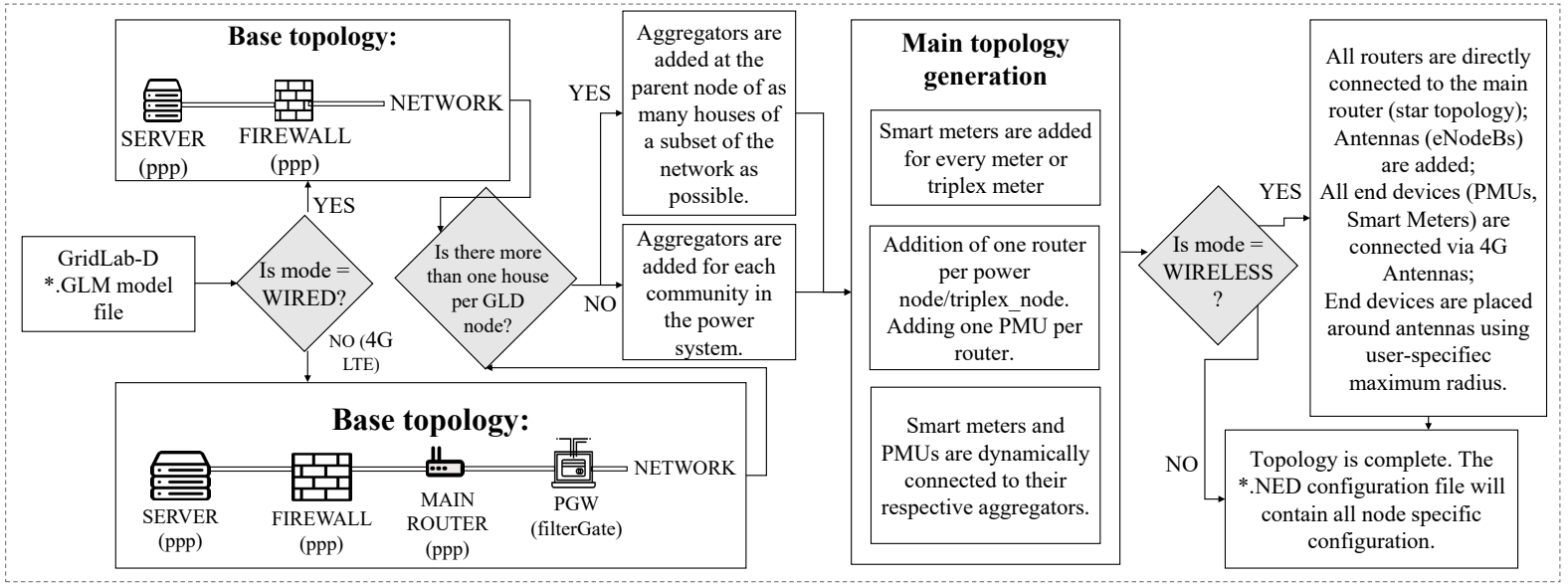


Figure 10: Topology generation logic

Each node of the electrical network is associated with a router, to which a PMU is connected, implemented by the RTU object. Moreover, each *meter* or $triplex\_meter$ of the GridLab-D model is associated to a smart meter within the OMNeT++ simulation. Because a meter can be associated to one or more houses, the simulation can dynamically assign the smart meter to each house to indicate if the smart meter is in charge of more than one house, i.e. one building. The Control Center is always protected by a Firewall, and data is routed through a set of routers connected in the same way as electrical nodes. Finally, all communication (except Firewall-Control Center) uses the PPP protocol for data exchange.

4.5.2.1 Wired Topology

The wired topology is based exclusively on communication cables similar to Ethernet cables. Each communication line extends the *DatarateChannel* class of INET with a realistic calculation of the d delay added during communications based on the distance between each network node:

$$d = replaceUnit(l_{line}/2^8, \text{“}s\text{”}) + \{fixedLatency\} \quad (1)$$

Where *replaceUnit* converts the channel line l_{line} distance to a temporal unit, and $\{fixedLatency\}$ is a user-settable fixed delay.

In the wired topology, nodes are dynamically placed by OMNeT++ without constraint via the $Mobility$ module. This type of network integrates the following equipment:

- *server* (Control Center): A centralized server that aggregate all monitored information. It implements both Demand-Response, Synchrophasor and Dynamic Pricing market capabilities.
- *firewall* (Stateless Firewall): A $HookableDevice$ Iptable-based firewall connected to the Control Center via a 10 Gigabytes Ethernet channel.
- *scenarioManager* (OMNeT++ scenario manager): It is a special object, not connected to the communication network, whose role is to control the lifecycle of the different devices in the network. This module can be parameterized by the user.
- *configurator* (Network configurator): The dynamic wired network topologies does not rely on a network configuration file with IPv4 addresses. The $configurator$ module dynamically assign each device with an IPv4 address.

- *httpController* (HTTP Controller): Dynamically assigns device URI for both smart meters, OpenADR aggregator and Control Center.
- *router* (Router): Networking devices from the INET library.
- *Meter, RTU, aggregator, Attacker, mainAggregator* (Monitoring and end-user devices): Devices that implement the custom *HookableDevice* class. The Attacker is a special class implemented when the DDoS attack is used to generate messages within the network.

4.5.2.2 Wireless Topology

The wireless topology is based on the simuLTE framework developed by the University of Pisa, Italy [54]. It permits the advanced study of LTE and advanced LTE networks (3GPP Release 8+). The dynamic Wireless mode of the Project Generator generates a wireless topology with the antennas and the preservation of the real distances between each node of the power network, as defined in the GLM file. Communications within the LTE network use the GPRS Tunneling Protocol (GTP) User-Plane protocol via UDP and X2 communications supports.

The placement of nodes with constrained link distance is a complex problem of the graph theory (force-directed graph drawing). In the context of computer networks, it is also important to ensure the following properties:

- Minimization of crossover between links;
- Enable an aesthetic rendering;
- Calculate an initial layout in a reasonable amount of time.

We chose to use the Kamada Kawai algorithm [55] for the generation of the first layout, followed by ForceAtlas2 Layout algorithm [56], configured with 1000 iterations and neutral gravity.

Thus, each node is assigned a position on the X and Y axis in real distances (meters). The further generation of the topology follows the same logic as for the Wired topology. Nevertheless, the houses are randomly placed around the antennas according to the radius defined by the user. The equipment added to the Wired network are:

- *channelControl* (LTE Channel Control): A realistic channel model with multiple capabilities (path-loss, shadowing, inter-cell interference, etc.).
- *routingRecorder* (Routing Table recorder): A special recorder saving all $IPv4RoutingTable$ or $InterfaceTable$ changes of all routers and host.
- *binder* (LTE Binder): A global module that stores information about network devices (i.e., reference to nodes).
- *pgw* (PGW Standard): A 4G LTE device that connects a 4G network to the internet or Operator network.
- *eNodeB* (Antennas): 4G LTE antennas. Close antennas are connected through a X2 interface.

4.5.3 Project Parameters and Customization

Beyond the generation of the topology, the Project Generator will apply many changes to the basic template, depending on the user's settings. The Appendix B details all the parameters managed by the user interface, as well as the possible customizations after the testbed generation.

4.6 Testbed Walk-through

Once the testbed has been generated, it will combine many capabilities borrowed from the simulators used, as well as contributions that we have made as part of this project. In this Section, we detail each aspect of the testbed and motivate the different design choices.

4.6.1 Federates Synchronization

Each federate uses a core that implements various functions exported from HELICS to synchronize. The Project Generator has the role of informing the co-simulation duration and the timestep of each one. Thus, the main broker will be in charge of the synchronization of all the federates and the information exchanges. Figure 11 shows an example of synchronization when the co-simulation timestep is $0.1s$ and OMNeT++, the Attacker Federate and the Visualizer synchronize every $0.2s$.

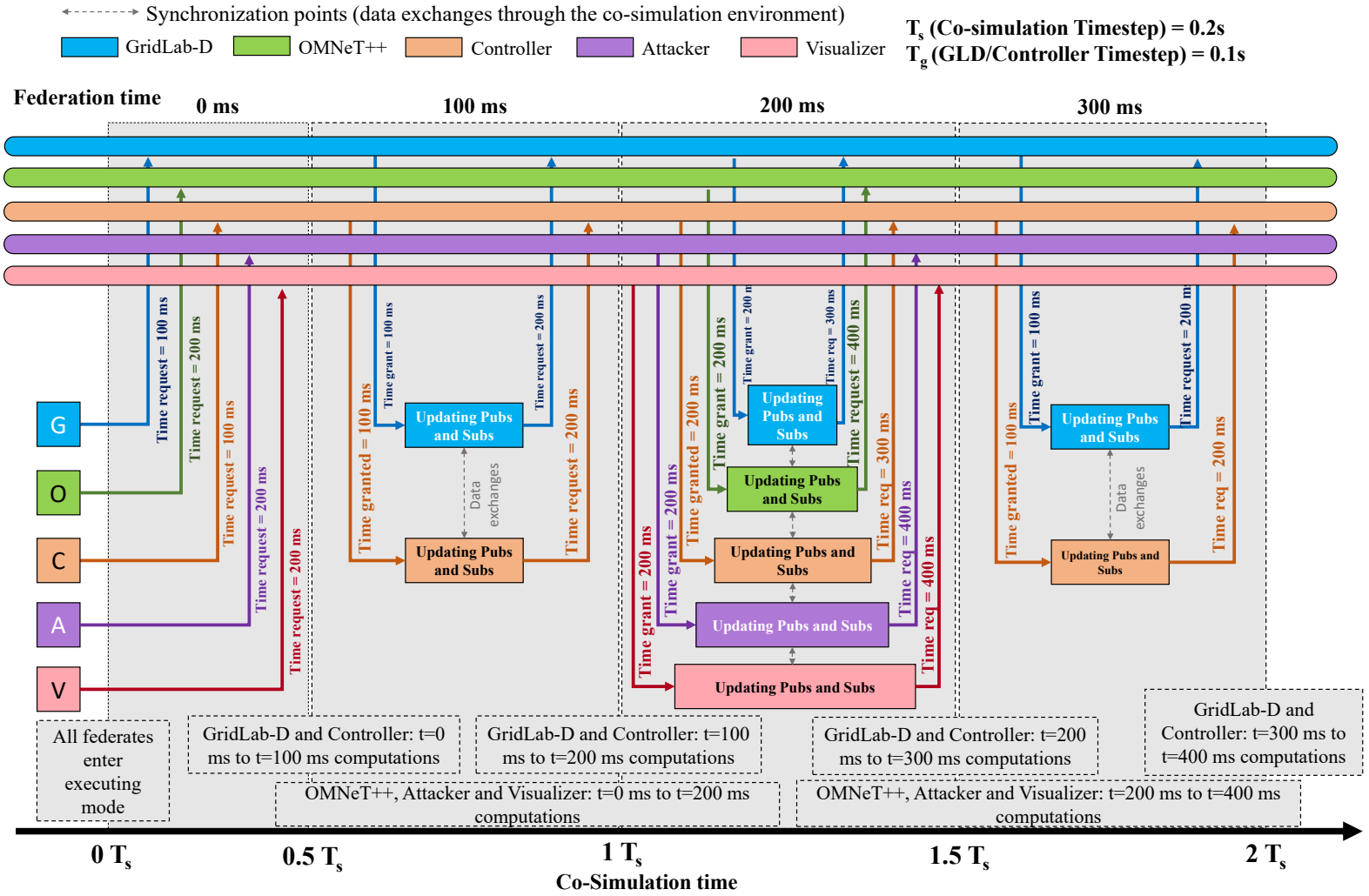


Figure 11: Testbed synchronization

4.6.2 Attacker Federate

The Attacker federate is an implementation of an interactive command line program in Python 3.7+. It has 2 threads and 2 processes:

- The *CLIWorker* implements dynamic updating of the interface display in relation to the co-simulation environment, by rewriting the current time.
- The *helicsWorker* implements the attacker's federate and communicates directly with HELICS via its core.
- The *Prompt* class implemented in the main thread allows the user to enter commands and get help.

It is dynamically synchronized to the testbed and offers a set of attacks generated by the Project Generator. The attacks come from user parameters at the Attacker profile level, from any attacks (hooks) used and from the power network, with default support for line status to simulate physical attacks on power cables.

The Appendix C details the architecture of the federate.

The use of multiple processes and threads is driven by a need to perform multiple operations on the terminal in real time. Indeed, the federate displays in real time the current co-simulation time while updating subscriptions and publications, and let the user asynchronously launch his or her commands. The *atk* command supports autocompletion, extending the federate to mimic a Meterpreter.

4.6.3 Controller Federate

We offer a modular implementation of the power grid control federate. This federate is connected to both OMNeT++ and GridLab-D. It dynamically retrieves OMNeT++ publications aggregating the measurements of the different PMUs within the shared variable "*controller*". Then, for each publication (here, the SWING buses and switches), it updates the values following those currently stored in memory, from the received measurements (e.g., the Voltage A value of the main SWING node). By default, the controller guarantees a stable state by always sending the initial values present in the GLM file but can be extended to let the user implement custom logic.

The algorithm 1 has a pseudocode of the federate. The *do_action* function is here the function that the user can modify. He or she is not limited to this function and can

use external code to perform further calculations, such as State Estimation, IDS, machine learning, etc. The integration of an intermediate federate such as MATLAB is possible.

The *do_action* function receives as input the current co-simulation time, as well as a dictionary linking each node of the power grid with the associated measurement and can change the stored publication values based on the internal logic.

Algorithm 1: Controller federate base code

Input: helics_publication array, values array, timestep, cosimulation duration;
Result: Power Grid CSV file (measurements values)
for *t in linspace(0 seconds int(seconds{timestep}))* **do**
    while *granted time* < *current_time* **do**
        Call HELICS Request Time function;
    end
    Update publications and subscriptions;
    Parse all received data from OMNeT++;
    Call the **do_action** function;
    Update publications and publish values to the cosimulation environment;
    Write useful timestamped information in the CSV file;
end
Exit the Federate *executing mode* before silently terminate the program;

In addition, it is possible to modify the *do_action* function to simulate a malicious code that would perform a malicious action on the power grid.

The integration of machine learning techniques within this customizable function therefore allows to simulate the online deployment of machine learning models and their incremental learning. Thus, the user can (1) simulate the use of pre-trained models using datasets generated by the testbed from previous co-simulations, prior to their direct deployment with a co-simulation or (2) directly integrate Machine Learning models and implement incremental learning.

4.6.4 Market Federate

The Market Federate is based on the same algorithm as presented before. However, instead of processing PMU-related data, it processes JSON-formatted strings of aggregated real bid requests sent by the Houses controllers to the *auction* object. The specific algorithm

for the Market federate is shown below. This federates retrieve values from the OMNeT++ *market* publication.

The *do_action* function receives as input the current co-simulation time, as well as the market aggregated and up-to-date bids, the market statistics and outputs the fixed price. Here, the clearing price is set via the Federate since the *auction* object, previously set in *FIXED_PRICE* mode, will guarantee that the clearing price used corresponds to the current fixed price, here published by the Market federate.

Algorithm 2: Market federate base code

Input: helics_publication array, values array, timestep, cosimulation duration;
Result: Market CSV file (Time, Total Load, Number of Buyers, Mean Load, Total price asked, Cleared price))
for *t in linspace(0 seconds int(seconds{timestep}))* **do**
    while *granted time* < *current_time* **do**
        Call HELICS Request Time function;
    end
    Update publications and subscriptions;
    Parse all received data from OMNeT++ and compute the market statistics;
    Call the **do_action** function;
    Update publications and publish values to the cosimulation environment;
    Write useful timestamped information in the CSV file;
end
Exit the Federate *executing mode* before silently terminate the program;

Figure 12 summarizes the functioning of the Market Federate, with the illustration of the impact of the communication network on the calculation of the clearing price.

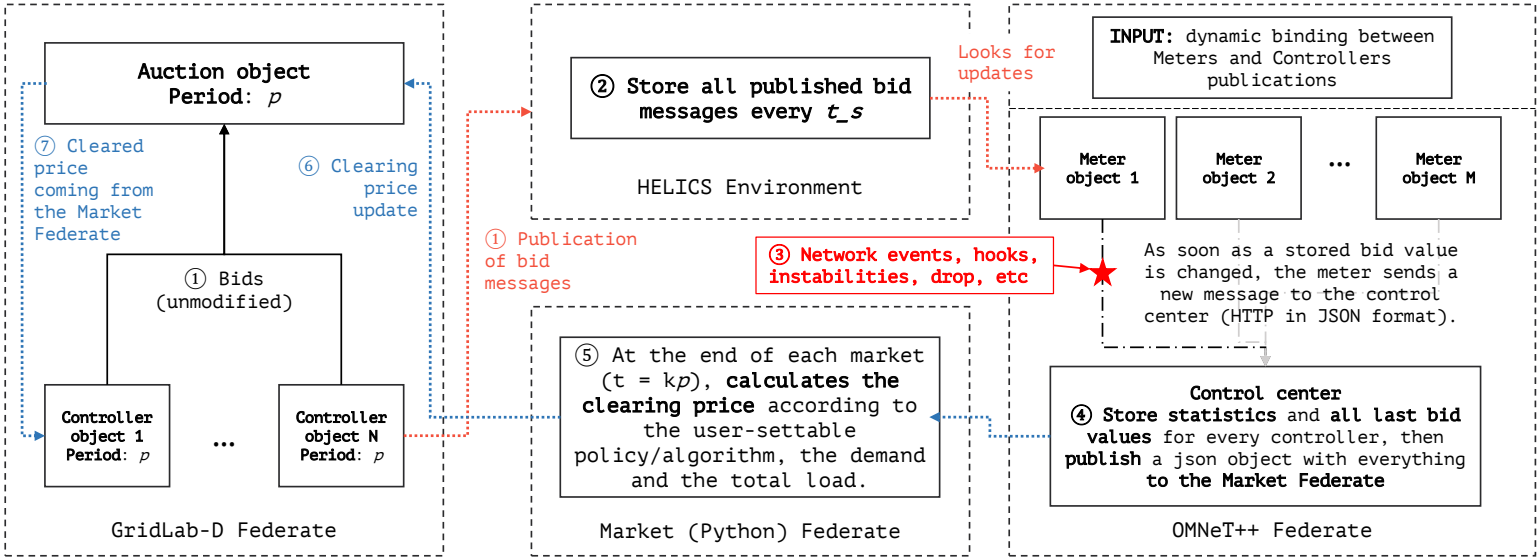


Figure 12: Market Federate overview

4.6.5 OMNeT++ Federate

Within OMNeT++, in order to guarantee a decent support of the capacities of a smart grid, we have developed various specific equipments. Each device is summarized in the Figure below, and in this Section we present the different aspects of each device.

Each device implements a succession of applications distributed over the different layers of the OSI model. Most of the implemented devices embed applications on the Application layer, although Hooks are directly implemented on top of the Physical layer.

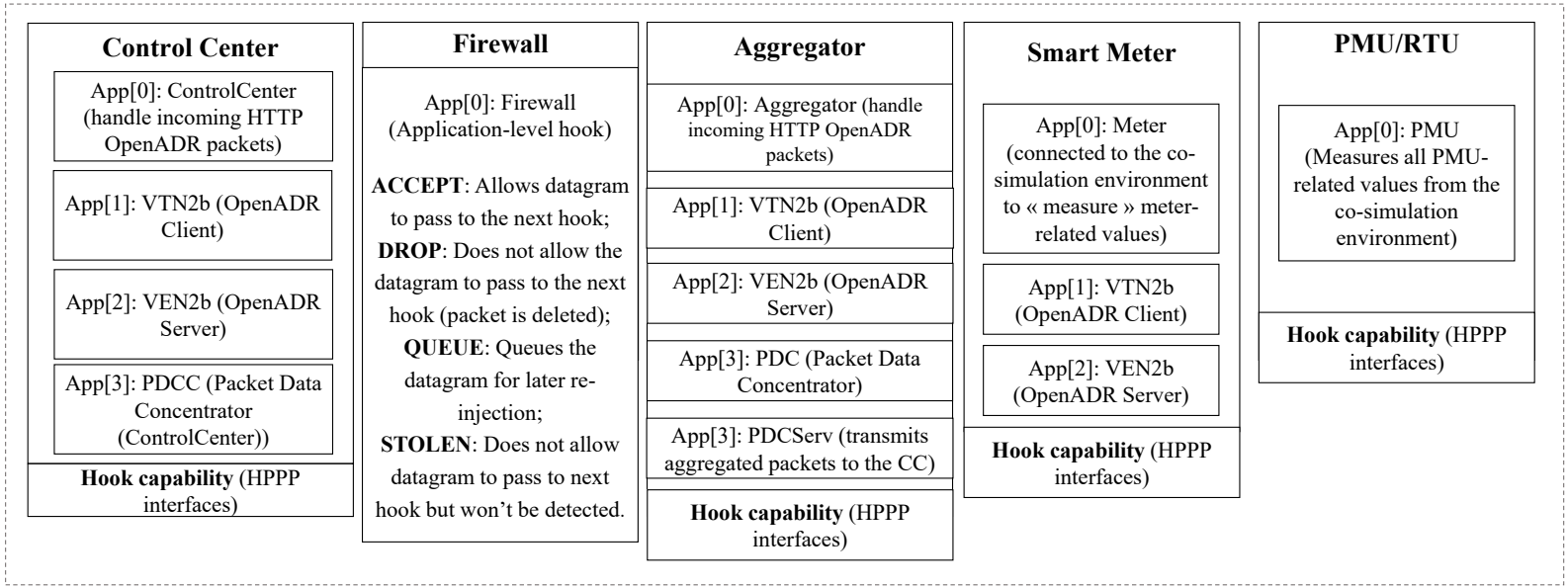


Figure 13: OMNeT++ implemented devices overview

4.6.5.1 HELICS Capability

There is no HELICS module for OMNeT++. It is however possible to use shared libraries to connect the federate to the co-simulation. To do this, we implement a $HELICSHelper$, shared by all the devices of the simulation (singleton) which proposes a set of methods allowing the devices to stay up to date. Thus, we detail the different roles that each device plays in co-simulation:

- **Control Center**: manages the global synchronization of the OMneT++ federate by scheduling self-messages at the same frequency as the co-simulation timestep and by requesting co-simulation time from the broker.
- **Smart Meter**: retrieves on a user-defined time basis the values related to Demand-Response and dynamic pricing using self-messages. Dynamically retrieves the list of local publications and subscriptions.
- **PMU**: this equipment is synchronized with the frequency defined by the user. It dynamically retrieves via $HELICSHelper$ the list of local publications and subscriptions.

Finally, $HELICSHelper$ proposes a conversion between the relative co-simulation time and the global timestamp as defined during project generation.

4.6.5.2 Control Center

The Control Center implements a TCP device with HTTP capability based on the OMNeT++ $cSimpleModule$ and deriving the $HelperBase$ class giving it access to the different applications implemented in the device. The applications implemented in the Control Center are the following:

- **ControlCenter**: Receives OpenADR messages before retransmitting them to the application being loaded. Also manages the equipment lifecycle, sending Dynamic Pricing data to the federate Market, and maintains synchronization of the entire network with HELICS.
- **VTN2b**: This application implements the OpenADR Virtual terminal Node 2.0b,

based on the *HttpBrowser* INET class. The VTN2b serves as a sending application within the context of the OpenADR protocol and allows here to send the configuration frames to the different smart-meters at the beginning of the user-defined OpenADR protocol life cycle.

- **VEN2b**: The Virtual End Node 2.0b of the CC serves as a main, centralized server for the OpenADR-protocol and receives, processes and manages all HTTP messages sent from the Smart Meters and Aggregators.
- **PDCC**: The Phasor Data Concentrator (Control Center) is the main aggregation point where all PMU messages are processed before being sent to the co-simulation environment. It parses each message in order to extract all the information from it.

4.6.5.3 Firewall

The firewall in an ad-hoc implementation of the Linux ip-table stateless firewall. Thus, we permit studies of more advanced filtering or detection methods, since the source code of this firewall is contained within the project.

The role of this equipment is to perform an action on each incoming message (from the Control Center to the network and vice versa), in order to $drop$, $forward$ or $accept$, based on the rules we have defined in the 4.5.3 Section. The global structure of this equipment can be found within the Figure 14. The *PacketDissector* class within a custom, recursive function parses all packets chunks and extracts the information before applying the optional rules.

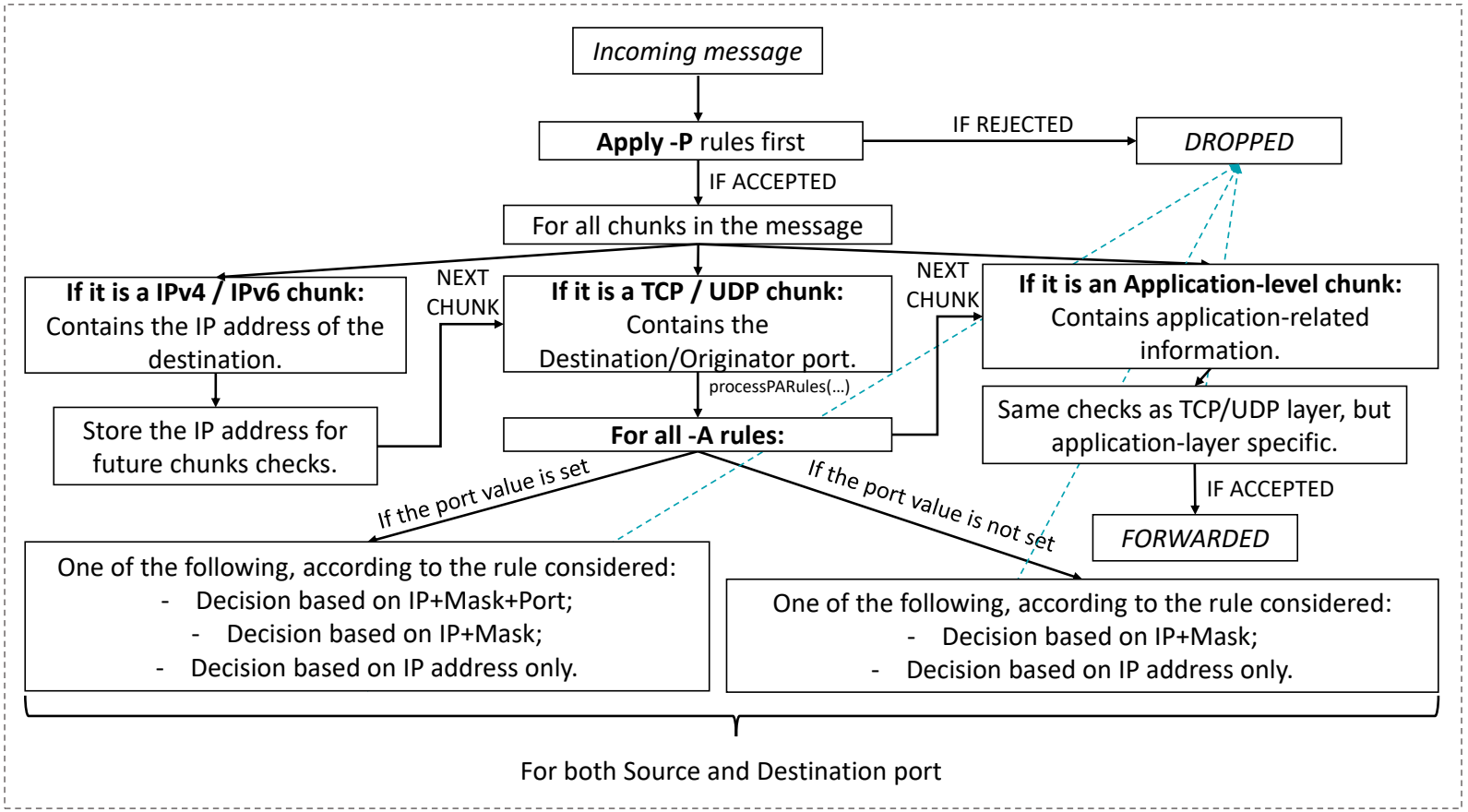


Figure 14: Firewall architecture overview

4.6.5.4 Aggregators

Aggregators are placed dynamically during the project generation process. These are common to the OpenADR and Synchrophasor protocols, since both protocols use the concept of an intermediate data aggregator. Each aggregator implements the following applications:

- **Aggregator**: The application is similar to *ControlCenter*, and serves here as an HTTP server for receiving OpenADR messages.
- **VTN2b & VEN2b**: Both Virtual End and Terminal Nodes 2.0b are implemented. Here, the application is configured to act as an aggregator and will correctly forward or aggregate openADR HTTP packets.
- **PDC**: The Phasor Data Concentrator aggregates all PMU messages that arrived during the PDC time window and send one or many aggregated messages to the Control Center, depending on the number of aggregated messages.
- **PDCServ**: The Phasor Data Concentrator Server is the destination of PMU devices and forwards all PMU messages to the PDC application.

4.6.5.5 PMUs

PMUs are monitoring devices that implements the C37-244-2013 Synchrophasor protocol specifications. The PMUs implement a $TcpBasicClientApp$ TCP client class that we extend and derive the $HelperBase$ class, which allows each PMU to dynamically retrieve the name of the subscriptions to which it is attached, based on the locally known device index. Each PMU acts dynamically according to the global or local configuration, defined during project generation. It is able to dynamically know if it has received local configurations.

During the co-simulation, each PMU is kept in a state of synchronization and will perform periodic measurements thanks to the use of self-messages. During each iteration, the device will retrieve each value from HELICS for the following metrics: $voltage\_A$, $voltage\_B$, $voltage\_C$, $measured\_angle\_A$, $measured\_angle\_B$, $measured\_angle\_C$, $measured\_frequency$, $measured\_frequency\_A$, $measured\_frequency\_B$, $measured\_frequency\_C$. The voltage magnitude is computed as:

$$M_i = \sqrt{voltage_i.real^2 + voltage_i.imag^2} \tag{2}$$

Then, we apply a truncated ($[-5\%;+5\%]$) noise to the frequency as well as the phase angle, if the noise capability is activated:

- **Phase angle**: A random number between -1 and 1 is generated via the C++ generator $default\_random\_engine$ and the standard $normal\_distribution$ class with the parameters defined by the user. If the value is higher than $|0.05|$, we set it to ± 0.05. Then, we apply the angle noise such that:

$$rand_i = truncatedRandomNumber() \in [-0.05; 0.05]$$
$$Angle'_i = Angle_i + 360 * rand_i \quad (3)$$

  Where $rand_i$ is the random number, $Angle'_i$ the noisy angle measurement and $Angle_i$ the initial, perfect measurement.

- **Frequency**: In the same way, we apply a noise relative to the base frequency (60), which we truncate if it exceeds 5% of frequency 60 (i.e. ± 3). Thus, we perform the following calculations:

$$rand_i = truncatedRandomNumber() \in [-3; 3]$$
$$Frequency'_i = Frequency_i + rand_i \quad (4)$$

  Where $rand_i$ is the random number, $Frequency'_i$ the noisy frequency measurement and $Frequency_i$ the initial, perfect measurement.

Finally, each measurement is sent via a $GenericAppPMU$ message we have defined with the following fields:

- **timestamp** (*simtime_t*): The Synchrophasor timestamp (measurement time);
- **measurements** (*double*): All measured values presented above;
- **scheduledTime** (*double*) Initial message sending time used for subsequent attacks targeting synchronization;
- **nodeName** (*string*): Associated Node name used internally.

4.6.5.6 Smart Meters

The Smart Meters implement two capabilities: Demand Response via the OpenADR protocol, and routing of Dynamic Pricing messages from GridLab-D. Each smart meter implements two main functions:

- **MeasureEnergy**: The *measure_energy* function dynamically measures all user-defined parameters related to the House and Meter objects of the GridLab-D federate. It also uses both VTN2b and VEN2b classes to create and send OpenADR XML messages through an HTTP connexion. We support the noise capability by adding a realive noise to all measured values, with the dynamic computation of the $confidence$ field of OpenADR messages.
- **HVACbids**: The Dynamic pricing capability is implemented for HVAC controllers. The Smart Meter object will detect any change in the gridLab-D-published last bid price and quantity sent and will generate a new bid message if it detects any change. We can, this way, simulate the bid message exchanges within OMNeT++ and directly study the effects of network events on the dynamic pricing market. The bid messages are sent via the VTN2b application as HTTP messages using the JSON format.

4.6.5.7 Protocols

In this Section, we detail the implementation of the OpenADR and Synchrophasor protocols.

The OpenADR protocol is based on the OpenADr 2.0b specifications and uses two modules: $OpenADRHelper$ to generate payloads ($EiRequestEvent$, $EiCreateReport$, $DistributeEvent$, $UpdateReport$, $Response$) and $XMLHelper$, which provides a set of methods for editing payloads in XML format to generate valid and complete messages. The two modules VTN2b and VEN2b are shared by the Smart Meters, Aggregators and Control Center, and are dynamically accessible from the main modules of each device. We implement this protocol by means of a state machine defining the messages to be sent or the responses to be made. The role of each device is dynamically known according to the messages received. The exchanges use the HTTP implementation proposed by INET, although we have modified the structure of the HTTP messages to include the $scheduledTime$, necessary for the proper simulation of certain attacks aimed at desynchronization.

We also provide support for the serialization of HTTP messages, which is used when the PCAP mode is activated. OMNeT++ then generates a PCAP dump file representing the communication exchanges in a valid way. Each type of HTTP message is thus correctly serialized and respects the real structure of HTTP messages (header bytes and payload).

The Synchrophasor protocol implements the C37-244-2013 specification. Within this protocol, we do not implement the frame configuration, and we focus exclusively on sending messages unilaterally from the PMUs to the Control Center. The structure of each message is correctly represented via a dynamic serializer, with the CRC calculation, the writing of each measured value, the timestamp and the name of the equipment.

4.6.6 GridLab-D Federate

As described above, we dynamically add capabilities to the GLM models loaded within the Project Generator. Thus, we dynamically add the *DELTAMODE* capability and various recorders that we detail in this Section.

4.6.6.1 Deltamode

The *DELTAMODE* is a capability proposed by GridLab-D to simulate the system with a resolution lower than one second. This mode is particularly interesting for the study of dynamic models powered by generators, or to validate the co-simulation of protocols such as Synchrophasor where the sending frequency is expressed in milliseconds. The Project Generator will thus define the following parameters:

- **simulation_mode**: Set to *DELTA*.
- **deltamode_timestep** (ms): The federate timestep according to the co-simulation timestep (in ms).
- **enable_subsecond_models** (boolean): Enables deltamode for the powerflow module.
- **deltamode_iteration_limit** (integer): The iteration limit when using *DELTAMODE*.

4.6.6.2 Recorders

As part of the generation of output files, we dynamically add recorders within the various GridLab-D objects considered. Thus, we add the recorders defined in Appendix E.

During co-simulation, and according to user-defined interval and recording time parameters, text files in CSV format are generated for each simulated object.

4.6.7 Visualizer Federate

In order to propose a complete modular and dynamic visualization of the co-simulation, we have opted for a web page visualization. It displays, via the D3JS library, of the electrical and communication networks, the whole being synchronized in real time with the co-simulation environment. The viewer also allows the user to dynamically change the co-simulation speed, to pause it, to view the status of each network node and to use the different profiles to access or not to access the information. Figure 15 shows the architecture of the Visualization federate. In this Section we explain how the federate works on the back end and front-end side.

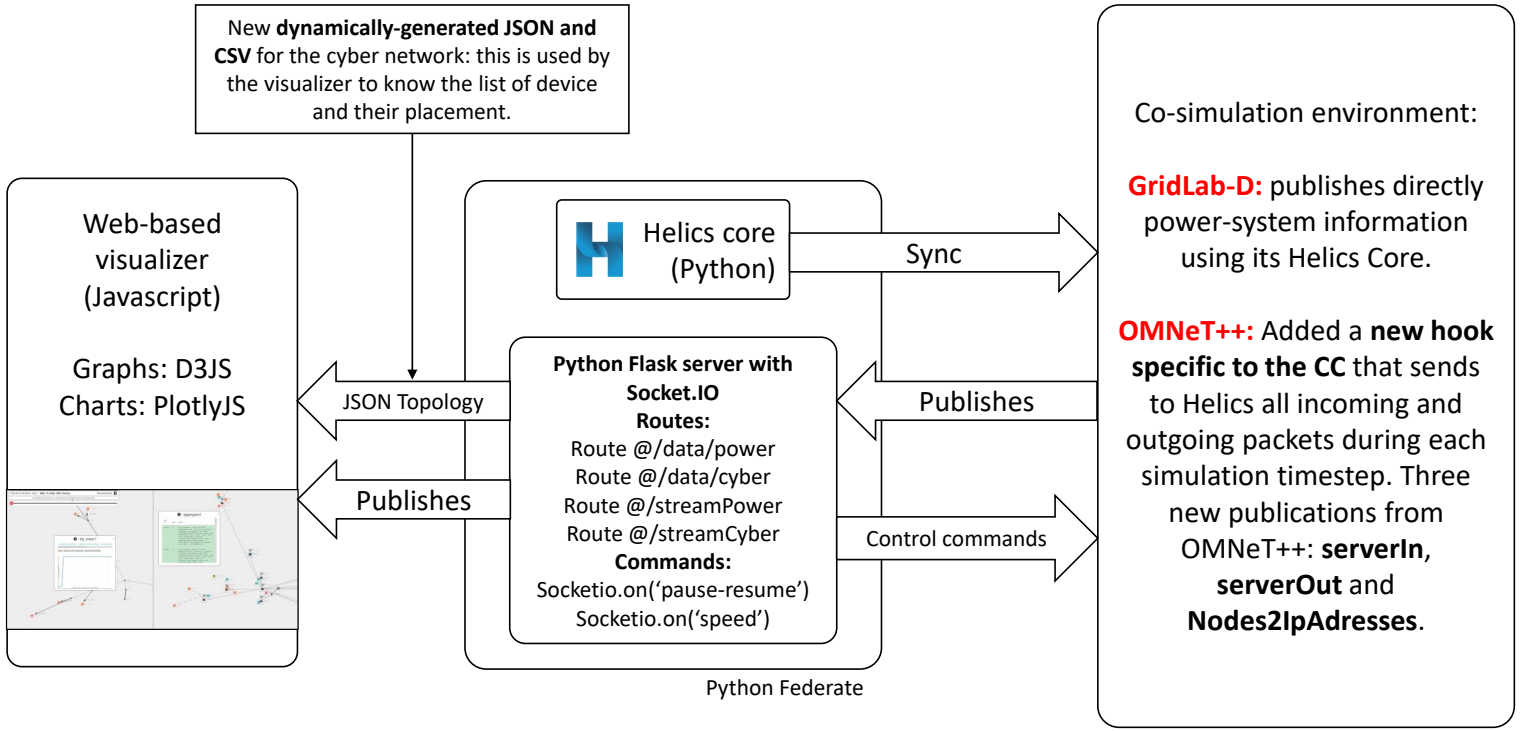


Figure 15: Visualizer architecture overview

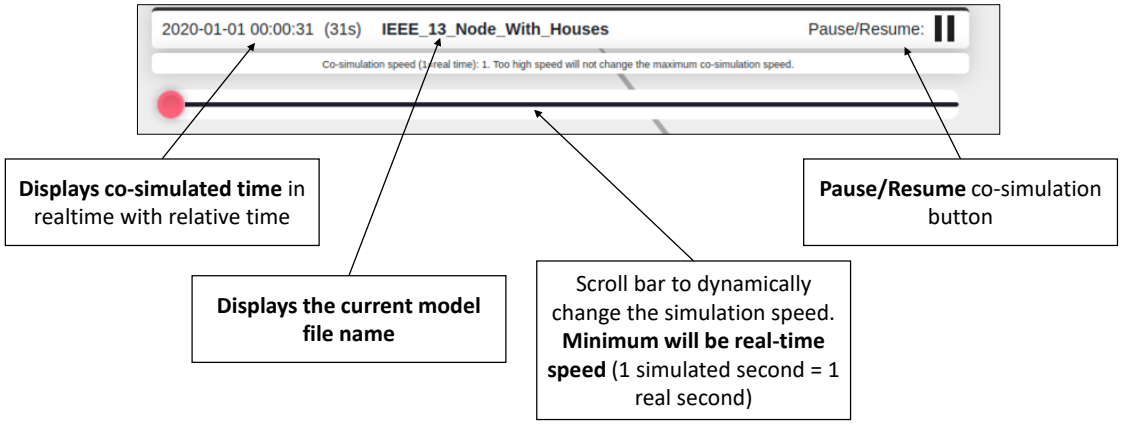


Figure 16: Visualizer control bar

4.6.7.1 Server

The server is developed in python and implements a *Flask* server with Socket.IO and Server-Sent-Events to permit bi-directional communication between server and client. The role of the server is to send the co-simulation topology (power and cyber) to the front-end, to route the data from the co-simulation into different silos which will then be displayed on the front-end side and to control the co-simulation speed and its pausing. The Figure 15 shows the list of existing routes for the different data.

4.6.7.2 Client

The client is developed in JavaScript and communicates with the server automatically. It displays a co-simulation control bar, shown in the Figure 16. In addition, the topology of the electrical and computer networks is displayed. For the electrical topology, the visualizer implements the visualization of real-time curves via the PlotlyJS library and internal pop-ups. On the cyber topology side, the federate displays in a user-readable fashion the last packets exchanged for each device on the network. To guarantee optimal performance, the viewer bases the storage of its data on ring buffers of user-definable size, so as not to cause memory leakage or excessive consumption. The Viewer displays, for each device in the communication network, their IP address for each network interface.

An example of the visualizer web page is shown in Figure 17.

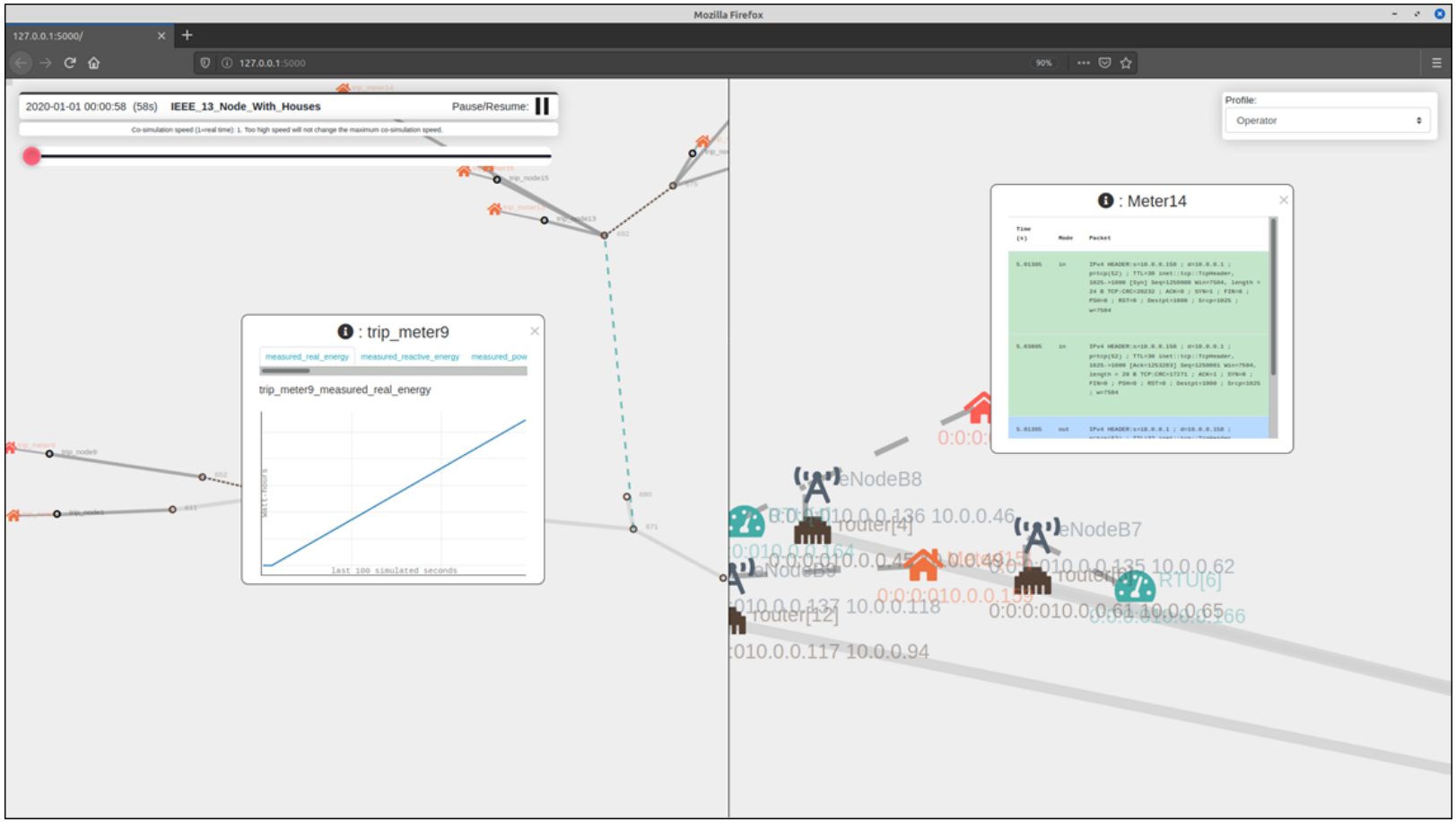


Figure 17: Visualizer example view for the IEEE 13 Bus System

4.7 Attacks and Hooks

Now that we have defined in detail the different aspects of co-simulation, we discuss in this Section the implementation of attacks. Their effects will be validated in the Chapter 5.

4.7.0.1 Hooks Overview

We implement the different attacks in the form of hooks, i.e. intermediate applications that are dynamically added by the Project Generator on top of the physical layer (i.e. on top of the network interfaces) in order to modify, drop, delay or replay messages. Each hook can then be concatenated to another one, to form a sequence of hooks, distinctively for incoming and outgoing messages. The capability implemented here enables the advanced study of complex, single-point, multiple-point, single-time and multiple-time cyber-attacks and their combinations. The user can apply one given hook to one specific device, a set of devices (using a filter) or all devices.

The ability to add hooks has been made possible through the implementation of a *HPPP* (Hookable Point-To-Point) derived network interface, coupled with updated equipment to integrate this new interface (*HookableDevice*). The concrete addition of the hooks is done during the project generation by the Project Generator, according to the user parameters.

The Figure 18 presents a simplified view of the hook sequence principle within the network architecture of the different devices. The Figure 13 presents the devices that have this hook capability. Routers have also been modified to integrate this capability.

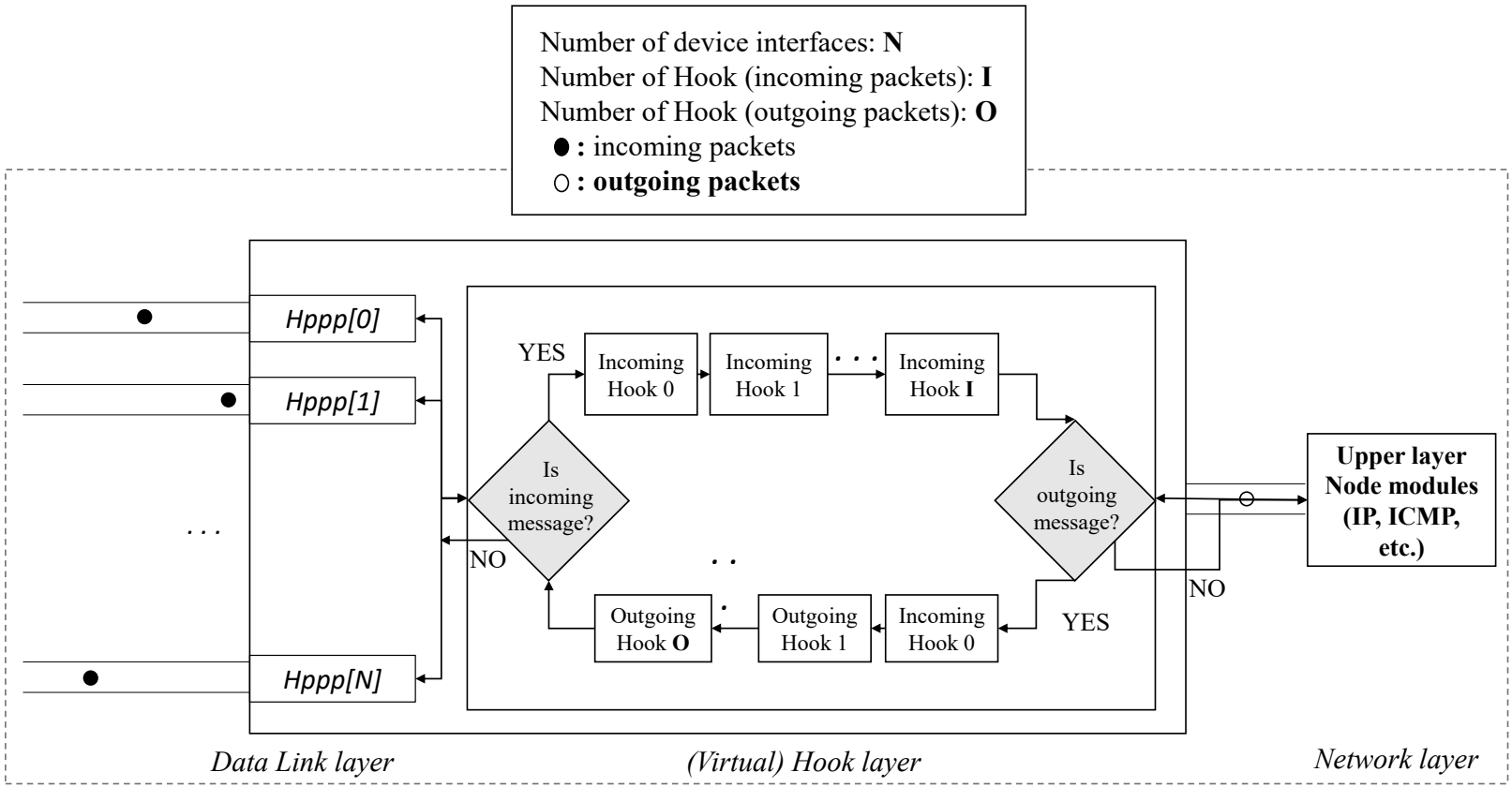


Figure 18: Hook sequences implementation overview

Each hook can be set up in the following modes of operation:

- **Permanent**: The hook will be activated during the whole co-simulation;
- **Live**: The hook is manually launched and configured by the user using the Attacker Federate;
- **Single-time**: The hook starts at the specified time and lasts after a given duration;
- **Multiple-time**: The hook starts at the specified time, lasts after a given duration and is repeated after a given waiting time window.

In the field of co-simulation, the study of attacks is based on the understanding of their impacts to study the primary or secondary effects induced by a malicious action on the system. For example, it is not necessary to fully simulate attacks, but to focus on the effects: an attack aimed at increasing network latency by saturating communication lines can be simulated more simply by adding latency. However, it is still interesting to actually simulate attacks such as DDoS, where the order of packets within network congestion has a direct impact on the attack results.

The following subsections present the different hooks implemented, as well as the ability to include malicious code.

4.7.0.2 Packet Dropping

Hook name	Packet Dropping
Hook target	Availability
Description	Packet Dropping is an attack designed to simulate the effects of an attacker preventing messages on the computer network, thereby jeopardizing the availability property.
Hook Properties	*Dropping_Probability*
	Dropping_Type (ALL, SYNCHROPHASOR, DYNAMIC PRICING, OPENADR)

This hook also has a statistics measurement function, and records the number of packets, number of bits, average throughput and packets per second. Figure 19 shows a logical view of the operation of this hook. Packet dropping is done via the public methods of the *cPacket* under consideration.

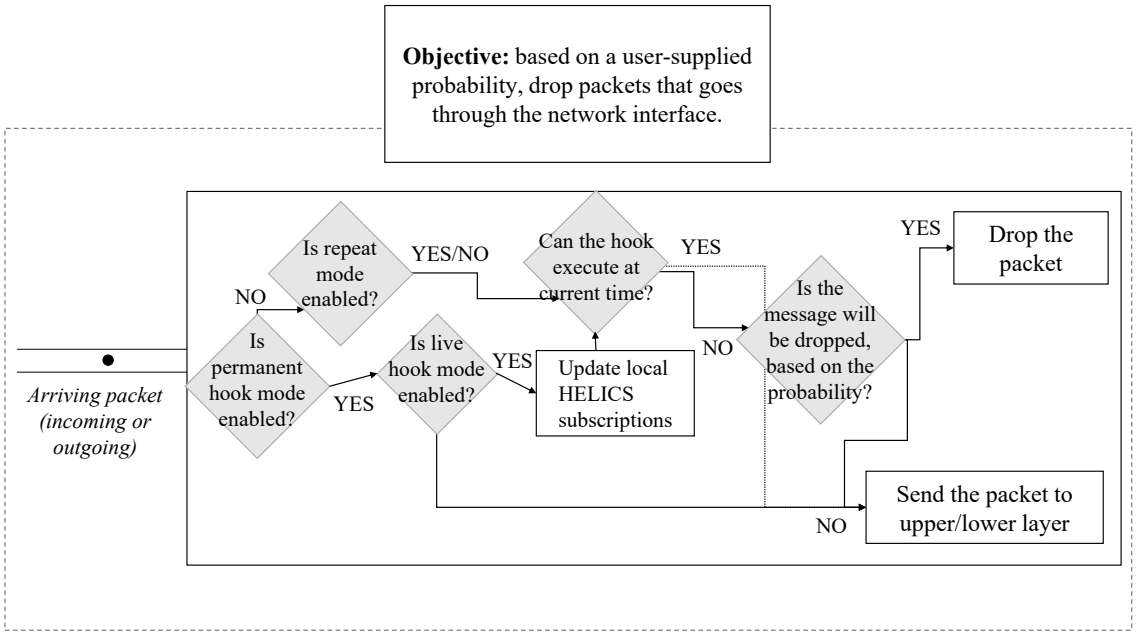


Figure 19: Packet dropping hook

4.7.0.3 Replay Attack

Hook name	Replay attack
Hook target	Integrity
Description	The replay attack is carried out in three phases: recording, waiting and replay. The hook allows the recording and replay of Synchrophasor, OpenADR or Dynamic Pricing messages.
Hook Properties	*Start_Recording_Time*
	Replay_Duration
	Replay_After

The Replay Attack hook implements a message modification mechanism. Since the protocol used is TCP, some constraints apply such as the Sequence Number which must correspond to the receiver window. To overcome this problem, the hook does not actually replay the messages but stores them in vectors in order to replay them in the same order during the replay period. When an incoming (or outgoing) packet arrives during the replay time, the Hook will replace its content with the previously recorded one. As a result, the Sequence Number is kept, the packet is accepted by the receiver, and the CRC is recalculated after updating the payload of the message, via the *PacketDissector*. This hook also has the ability to record statistics during co-simulation. The statistics are then accessible in

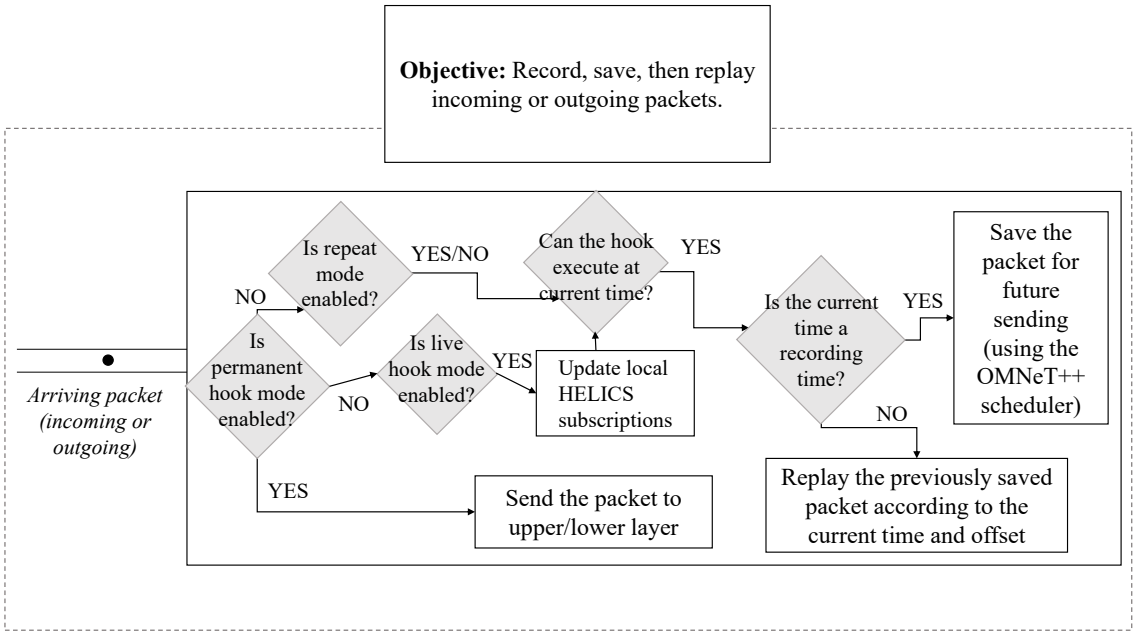


Figure 20: Replay attack hook

the OMNeT++ output *vec* file, readable via the Network Simulator IDE.

4.7.0.4 Packet Delayer

Hook name	Packet Delayer attack
Hook target	Availability
Description	The Packet delayer attack provides the ability to add a fixed or random delay to increase the time between its sending and its reception by the initial recipient.
Hook Properties	*Fixed_Delay_mode* or *Random_Delay_Mode*
	Fixed_Delay
	Random_Delay_Mean
	Random_Delay_Std

This hook performs a scheduling action on the considered messages, i.e. the messages are rescheduled in the future using OMNeT++'s *scheduleAt* method. The Figure 21 presents a logical view of the operation of this hook. Both $Uniform$ and *Gamma* random distributions are supported here, although the user can implement his or her own random distribution.

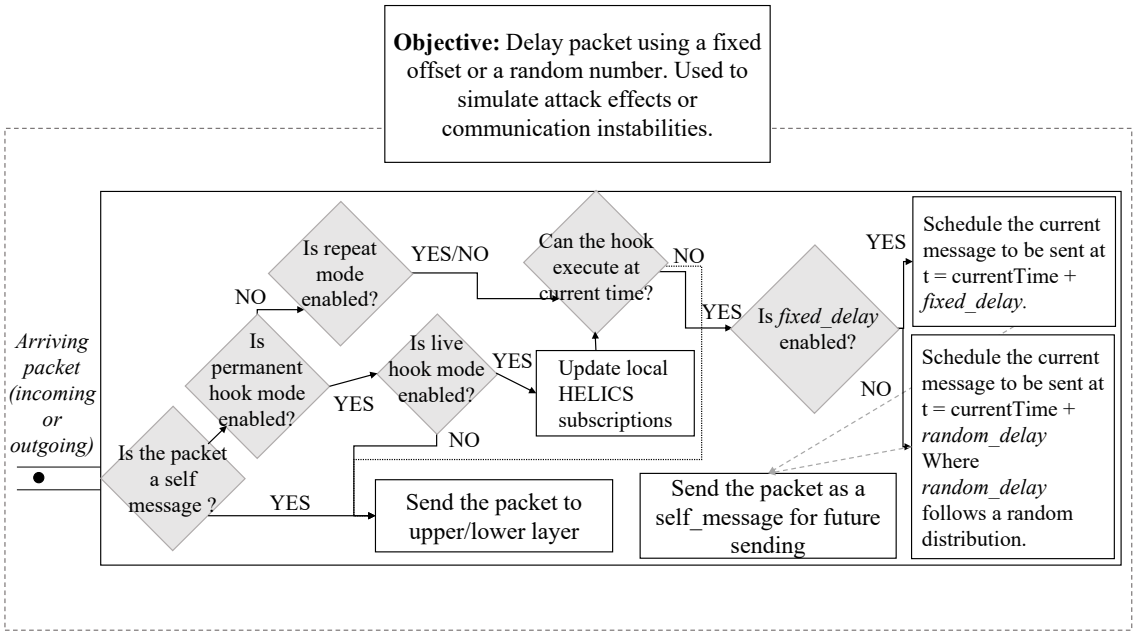


Figure 21: Delayer attack hook

4.7.0.5 DoS/DDoS

The DoS/DDoS attack is a special hook implemented at the application level of the OSI model. It implements a special hardware derived from the INET *ApplicationBase* class and uses the UDP protocol, thus performing UDP flood attack. The attack is fully configurable, and also implement the modification of the IP address of the sender and the receiver with the name of existing network in the system, enabling advanced simulation of network congestion between specific points. This malicious equipment can also be connected to any location on the network and several such devices can be placed within the same network.

Hook name	DoS/DDoS Attack
Hook target	Availability
Description	Malicious equipment connected to a router sends large amounts of data over the network, causing network congestion and blocking the availability of messages from PMUs or Demand Response/Dynamic Pricing.
Hook Properties	*Packet_Size* (Bytes)
	Sending_Delay (μs)
	Target_Device
	Targetted_Port
	Source_Device

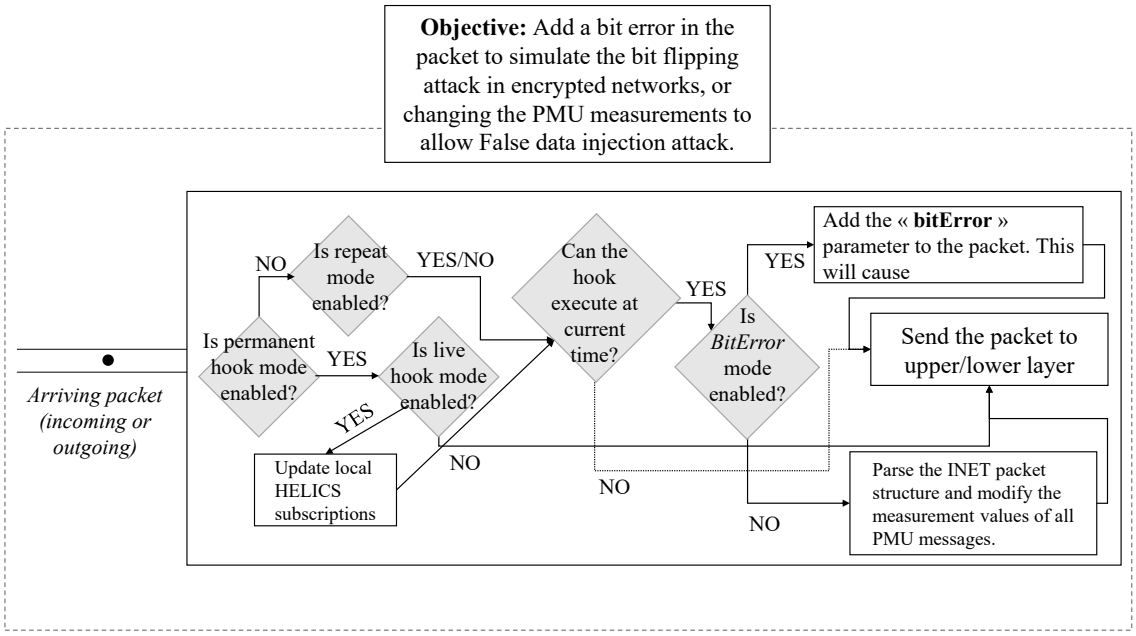


Figure 22: Integrity attack hook

4.7.0.6 Integrity Attack

Integrity attack is a major threat to Smart Grids, as it is the basis for advanced attacks such as FDIA, which can lead the system into a dangerous state. We implement a generic integrity attack for the Synchrophasor protocol and more generally the principle of *BitError* invalidating any MAC or CRC code and causing a rejection of the message by the receiver. For the Synchrophasor protocol, since this attack modifies the payload of the initial message, it also updates the CRC used by recalculating it. Details are shown in Figure 22.

Hook name	Integrity Attack
Hook target	Integrity, availability
Description	An attacker succeeds in modifying the content of the exchanged messages in order to hide an action on the system or to deceive the State Estimator.
Hook Properties	*Bit_Error_Mode* or *Synchrophasor_Mode*
	$False\_Angle\_A,B,C$
	$False\_Magnitude\_A,B,C$
	$False\_Frequency\_A,B,C$

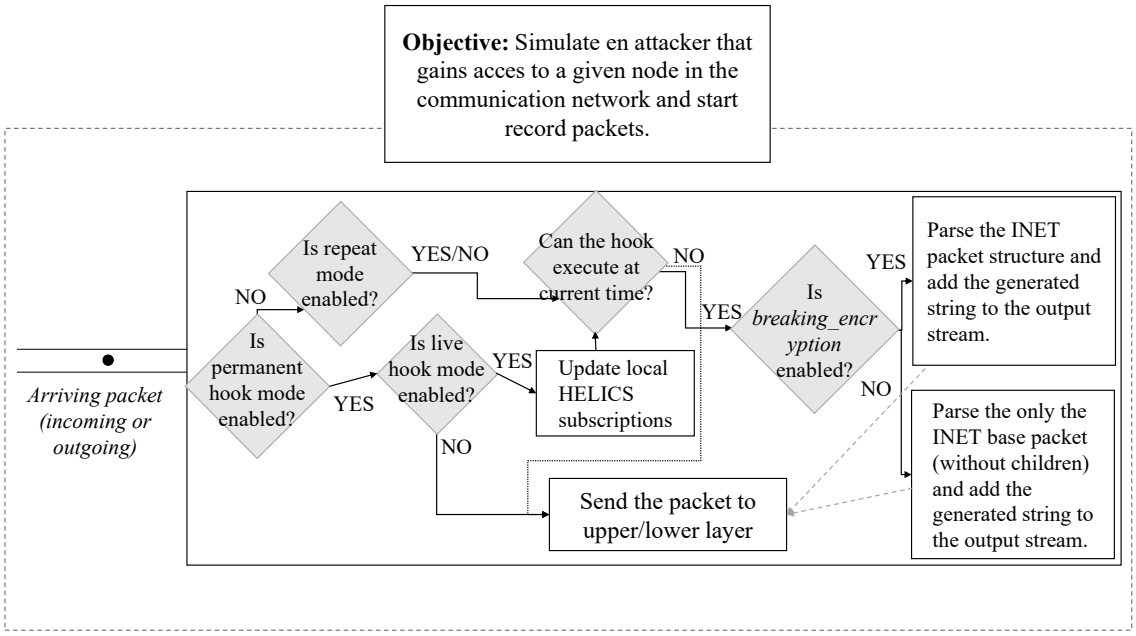


Figure 23: Eveasdropping attack hook

4.7.0.7 Eavesdropping

The eavesdropping attack sets up a passive attacker listening to the network at a given point. He is then able to obtain basic information about the message (if the communications are encrypted) or even the entire data if the confidentiality property is not ensured. In this project, we do not simulate encryption, but we distinguish the information known to an attacker according to the hook's mode of operation. If the encryption is virtually enabled, then the attacker will only have access to a part of the information, such as the packet size, the port used, etc. If encryption is not enabled, the attacker will be able to see all the data. This hook thus generates various text files in real time that can be displayed directly in a terminal.

Hook name	Eavesdropping attack
Hook target	Confidentiality
Description	A passive attacker listen and record packets.
Hook Properties	*Break_Encryption*

The Figure 23 shows a logical view of how this hook works.

4.7.0.8 Desynchronization

The Desynchronization attack simulates a GPS spoofing attack. PMUs in smart grids are synchronized in real time thanks to the temporal broadcast emitted by a GPS. When an attacker attempts to modify the signal to desynchronize the GPS, the PMU makes a measurement error, since the Synchrophasor is timestamped at all points. Each PMU periodically sends different measurements: Frequency, Phase Angle, and Magnitude. When the messages are received, the Control Center will be able to reconstruct a sinusoidal signal based on the magnitude, phase angle, and time contained in the message. This hook aims to modify the time contained in the message in order to alter the sinusoidal signal reconstructed by the Control Center. Details are shown in Figure 24.

The Control Center, knowing the magnitude m, the timestamp t_s, and the phase angle p, calculates the Synchrophasor s using this calculation:

$$s = m * cos(t_s + p) \tag{5}$$

Thus, a change in t_s will result in a shift of the sinusoid to the left or to the right. This hook allows the modification of t_s via a fixed or random time offset t_{error}, via a random generation based on the normal distribution:

$$s_{error} = m * cos(t_s + t_{error} + p) \tag{6}$$

Hook name	Desynchronization attack (GPS spoofing)
Hook target	Integrity
Description	An attacker alter the PMU synchronization by spoofing the GPS signal.
Hook Properties	$Fixed\_Time\_Offset$
	$Random\_Time\_Mean$
	$Random\_Time\_Std$

4.7.0.9 Dynamic Pricing

The Dynamic Pricing attack simulates an attacker modifying the bid prices sent by controllers, thus coming close to an integrity attack. The details of this attack are described in the Figure 25.

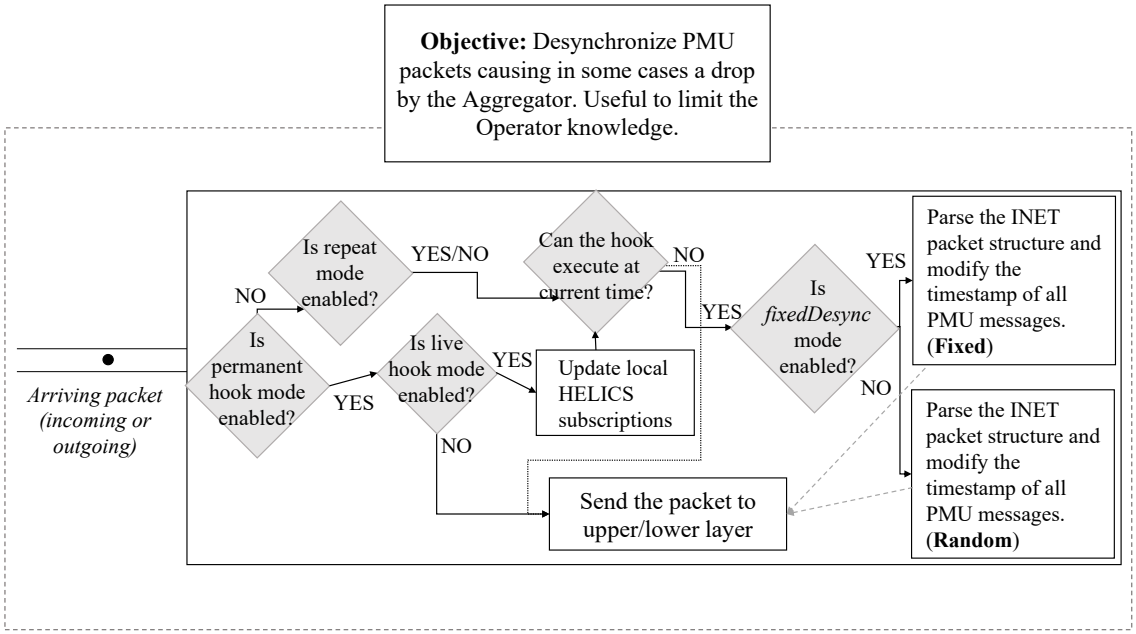


Figure 24: Desynchronization attack hook

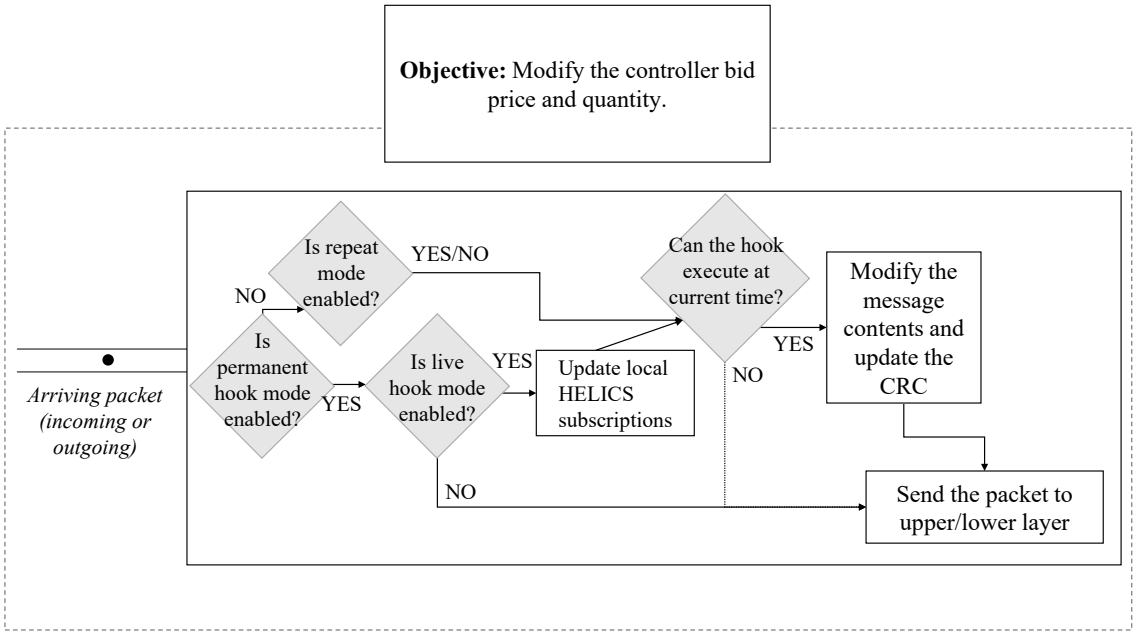


Figure 25: Dynamic Pricing attack hook

Hook name	Dynamic Pricing attack
Hook target	Integrity
Description	An attacker alter the Dynamic Pricing messages Bid Price and Quantity.
Hook Properties	*Bid_Price*
	Bid_Quantity

4.7.0.10 Statistics

This hook does not implement an attack, but generates CSV files representing various useful statistics on the network at a given point:

- **Mean latency**
- **Bits per second**
- **Packets per second**
- **Bytes per second**
- **Bytes per recording interval**
- **Packets per recording interval**

CSV files are dynamically written in the root folder of the OMNeT++ federate.

4.7.0.11 Malicious Code

The malicious code attack is not implemented directly in this project, but can be emulated via the previously described function, *do_action*, letting the user simulate the logic of his choice within the federates controller. The implementation of the Malicious Code is at the user's expense and will allow him to simply study the effects of malicious code that would send false commands.

Chapter 5

Testbed Dataset Generation, Validation and Scenarios

This Section validate the current implementation of ASGARDS-H. We first detail the Dataset generation process, then we highlight the Testbed capabilities by validating numerous aspects permitted by our co-simulation approach.

5.1 Dataset Generation

Dataset generation is the main objective of our project. In order to allow the advanced study of cyber-physical attacks using machine learning techniques, we exploit the different capabilities of the simulators in order to co-simulate a system in the most realistic way possible.

In the context of real systems, all the information accessible from the Control Center comes from the communication network, i.e. from all the protocols used for monitoring and various applications such as Dynamic Pricing or Demand Response. Within the context of co-simulation, we obtain direct information from the electrical and computer network, enabling us to refine the quality of the dataset by offering a set of data that is not altered by the communication network or by possible attacks. Moreover, co-simulation is interested in the mutual effects between simulators, enabling the visualization of the concrete impacts of cyber-attacks on the power grid, modifying the state of the global system and visible from the Control Center.

The generation of our dataset is mainly based on one or more PCAP (Wireshark) files,

containing all the network exchanges between the Control Center and the rest of the network. We also propose the generation of annex files. Finally, we detail the dataset generation process.

5.1.1 PCAP Generation

The *INET* framework supports serializers that transforms simulated messages without binary structure into a payload defined byte by byte and respecting the specifications of real protocols. Thus, the PCAP file generation contains a realistic set of communications in binary format and recognized by third party tools such as Wireshark [57].

The PCAP capability is supported by the Project Generator, which will dynamically define the *PcapRecorders* that will be used by the Control Center, via the *useWireshark* parameter:

```
**.server.numPcapRecorders = {useWireshark}
**.server.pcapRecorder[0].pcapFile = "res/servereth.pcap"
**.server.pcapRecorder[0].pcapLinkType = 1
**.server.pcapRecorder[0].moduleNamePatterns = "eth[*]"
**.server.pcapRecorder[1].pcapFile = "res/serverhppp.pcap"
**.server.pcapRecorder[1].pcapLinkType = 204
**.server.pcapRecorder[1].moduleNamePatterns = "hppp[*]"
```

Since the Control Center communicates both via PPP and Ethernet depending on the operating mode, two types of PCAP files will be generated: the first one, for each *eth* interface, and the second one, for each *hppp* interface, our *PPP* interface that we have modified to allow the integration of hooks.

The list of serializers that we have added or modified to *INET* and *simuLTE* are present in the following list:

- **HttpBaseMessage**
- **HttpRequestMessage**
- **HttpReplyMessage**
- **CustomHttpRequestMessage**
- **CustomHttpReplyMessage**

- **GtpUserMsg**
- **IcmpHeader**
- **IcmpEchoRequest**
- **IcmpEchoReply**
- **GenericAppPMU**

When the *eavesdropping* hook is used, the Project Generator dynamically adds the generation of PCAP files for each of the devices affected by the attack.

5.1.2 Statistics and Power System

By default, OMNeT++ generates a file in *vec* format that contains all the data related to the simulation. This format is shared by all the system's equipment, in particular the equipment related to the 4G LTE of *simuLTE*, which we have extended with the integration of hooks compatible with the generation of statistics. This file defines three types of data: scalar, vector and histogram. The following options are defined by the project generator to allow the complete generation of this statistics file:

```
**.vector-recording = true
output-scalar-file = ${resultdir}/${comp}_${repetition}.sca
output-vector-file = ${resultdir}/${comp}_${repetition}.vec
seed-set = ${repetition}
```

Moreover, thanks to the use of *recorders* within GridLab-D, we generate a set of text files, timestamped in CSV format, which contain all the information defined in the Appendix E.

Finally, the use of the statistics hook allows the generation of CSV files containing useful metrics to improve the visualization of the system.

5.1.3 Standardized Dataset Extraction and Generation

The generation of the dataset is done in several steps.

5.1.3.1 PCAP File Conversion

The PCAP format being binary, we have implemented a mechanism to convert PCAP files to a readable CSV format. This conversion uses *tshark*, the command-line module of Wireshark. Packet by packet, we extract each useful information with the support of the different implemented protocols, using regex.

We thus generate a unique CSV file containing the columns as defined in Appendix D.

This process is performed at least once for the main PCAP file and is also executed for each of the generated PCAP files (*eavesdropping attack*).

5.1.3.2 Power System and Statistics File Extraction

In a second step, the dataset generator will automatically extract all useful files from the co-simulation:

- **CSV power grid files**: All recorder-generated output files from GridLab-D are extracted and moved to the dataset folder;
- **OMNeT++ vec file** The OMNeT++ statistics *vec* file is also moved to the dataset folder;
- **Statistics** Generated statistics file from the Control Center or Statistics hooks are moved to the dataset folder.

5.2 Testbed Validation

We validate our testbed with 13 case studies using the different hooks implemented in different networks. We simplify the validation process using a simplified electrical model, defined in Figure 26.

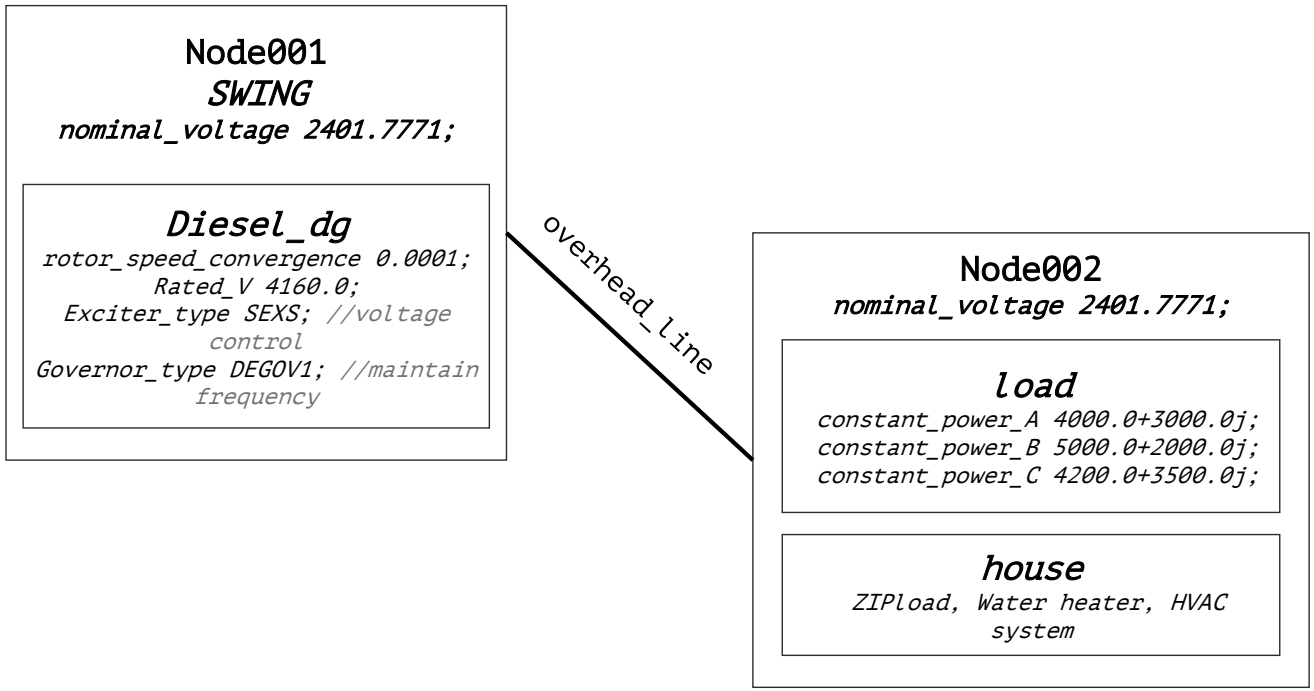


Figure 26: Small power system implemented for the testbed validation

The IEEE 13 Bus System network that we have modified is equipped with 16 houses equipped with ZIP load, Water heaters and HVAC systems.

All the scenarios studied are described in the Figure 27. We highlight the co-simulation results using visualization Python scripts that display several Control Center-monitored values such as the Magnitude, Frequency of Phase Angle. The scenarios we detail in this Section are based on these visualizations. We will present, for each study, the most characteristic results that allow us to validate each of the testbed's capabilities.

#	Model name	Duration	Network speed	Specific configuration
0	Small_system	60 sec	50MBps	No recording noise
1	Small_system	60 sec	50MBps	Recording noise (mean=0.001, std=0.0005). **Parameters used by other scenarios.**
2	Small_system	60 sec	1MBps	**DDoS** on router[1] starts at t=30s, lasts 10s
3	Small_system	60 sec	1MBps	**Packet dropping attack (all PMUs) (probability: 0.91)** starts at t=25s, lasts 10s
4	Small_system	60 sec	1MBps	**GridLab-D manual fault** starts at t=30s
5	Small_system	60 sec	1MBps	**Replay attack** starts at t=1s, record 10s and replays after 20 seconds
6	Small_system	60 sec	1MBps	**Desynchronization attack on RTU[1] (fixed +1.200s)** starts at t=25s, lasts 10 seconds
7	Small_system	60 sec	1MBps	**Integrity attack on RTU[1] (aA: 20.02; aB: 15.02; aC: 0.02; fA: 61; fB: 58; fC: 63: f:61)** starts at t=25s, lasts 10 seconds
8	IEEE 13 Bus System	60 sec	1MBps	**Malicious code** (Controller), multiplies the input voltages by 2 between t=10s and t=20s
9	Small_system	60 sec	1MBps	**Packet delayer attack on RTU[0] (mean=0, std=1.5)** starts at t=25s, lasts 10 seconds
10	Small_system	60 sec	1MBps	**Packet delayer attack on RTU[0] (mean=0, std=1.5)** starts at t=15s, lasts 7 seconds; **Integrity attack on RTU[1] (aA: 20.02; aB: 15.02; aC: 0.02; fA: 61; fB: 58; fC: 63: f:61)** starts at t=30s, lasts 7 seconds **Desynchronization attack on RTU[1] (fixed +1.200s)** starts at t=25s, lasts 15 seconds; **Packet dropping attack (all PMUs) (probability: 0.2)** starts at t=52s, lasts 3s
11	IEEE 13 Bus System v2	2 h	50MBps	No Synchrophasor/OpenADR;
12	IEEE 13 Bus System v2	2 h	50MBps	No Synchrophasor/OpenADR; **Dynamic pricing attack** on Meter[4] (bidP: 20; bidQty: 12.345) and Meter[13] (bidP: 0.5; bidQty: 50) starts at t=1000s, lasts 1700s
13	IEEE 13 Bus System v2	2 h	50MBps	No Synchrophasor/OpenADR; **Dynamic pricing attack** on Meter[4] (bidP: 20; bidQty: 12.345) and Meter[13] (bidP: 0.5; bidQty: 50) starts at t=1000s, lasts 1700s **Packet delayer attack** on Meter[10] (mean=0s, std=0.5s), starts at t=1200, lasts 800s **Packet dropping attack (all PMUs) (probability: 0.91)** starts at t=1200s, lasts 800s

Figure 27: Validation scenarios

5.2.1 Validation 1: Perfect System

In this scenario, we validate the testbed co-simulation under normal, transient to steady states. No noise is added, and the results are shown in Figure 28. We can see the effect of the Diesel Generator start-up on the measured Magnitude and Synchrophasor measurements. The frequency, shown in Figure 29, highlight the dynamics of the system before entering in its steady-state, around a measured Frequency of 59.965.

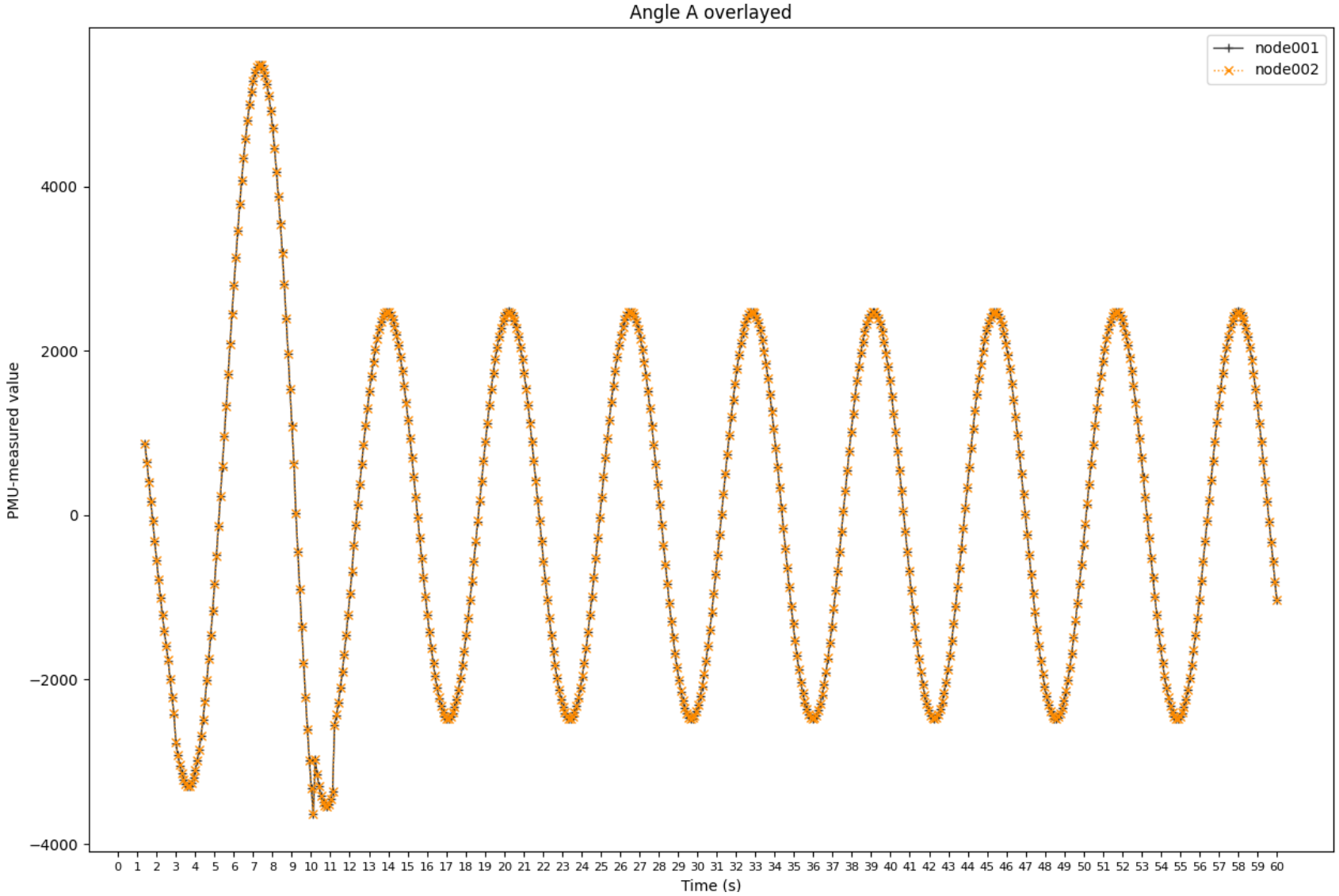


Figure 28: Scenario 1 validation results (Angle A)

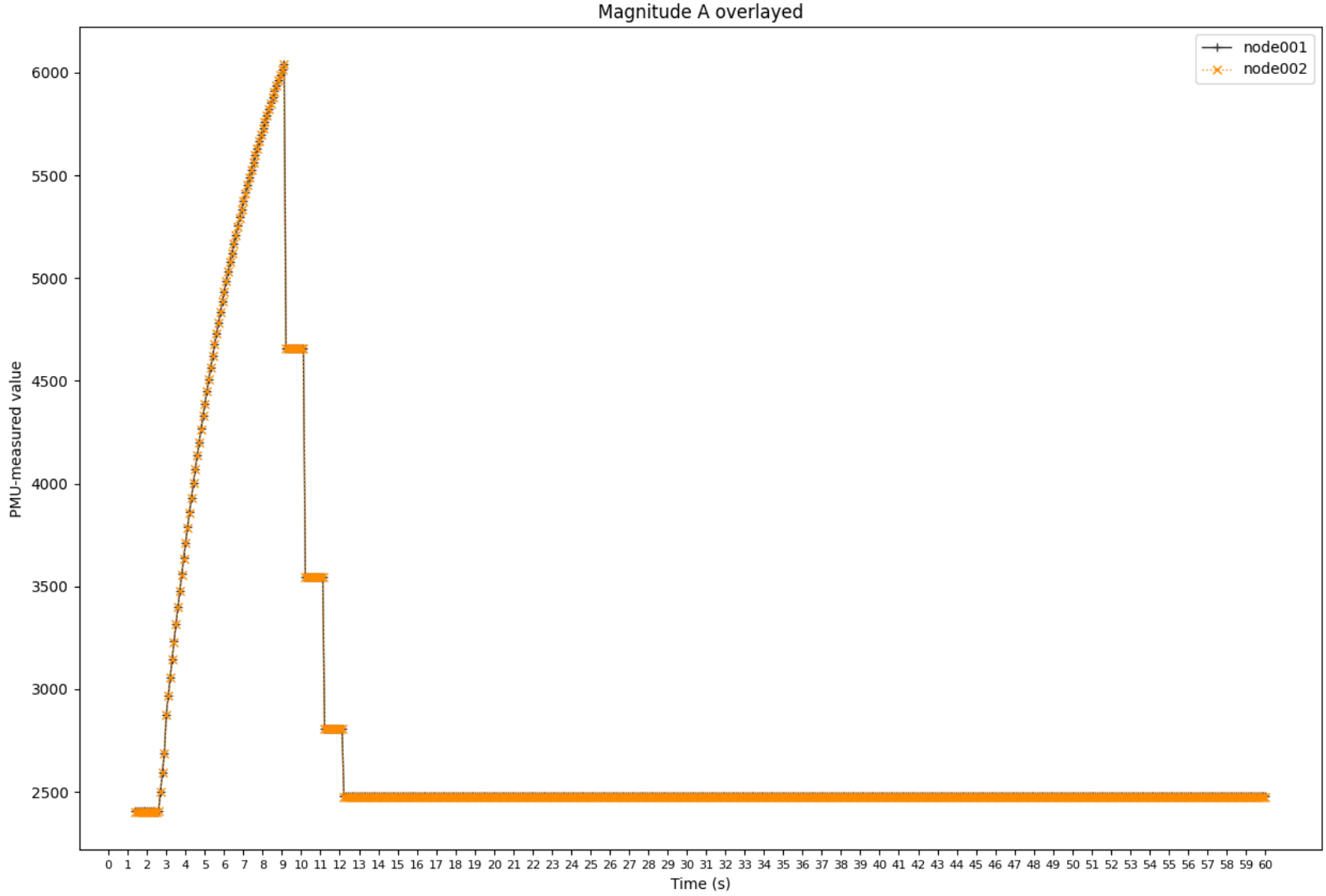


Figure 28: Scenario 1 validation results (Magnitude A)

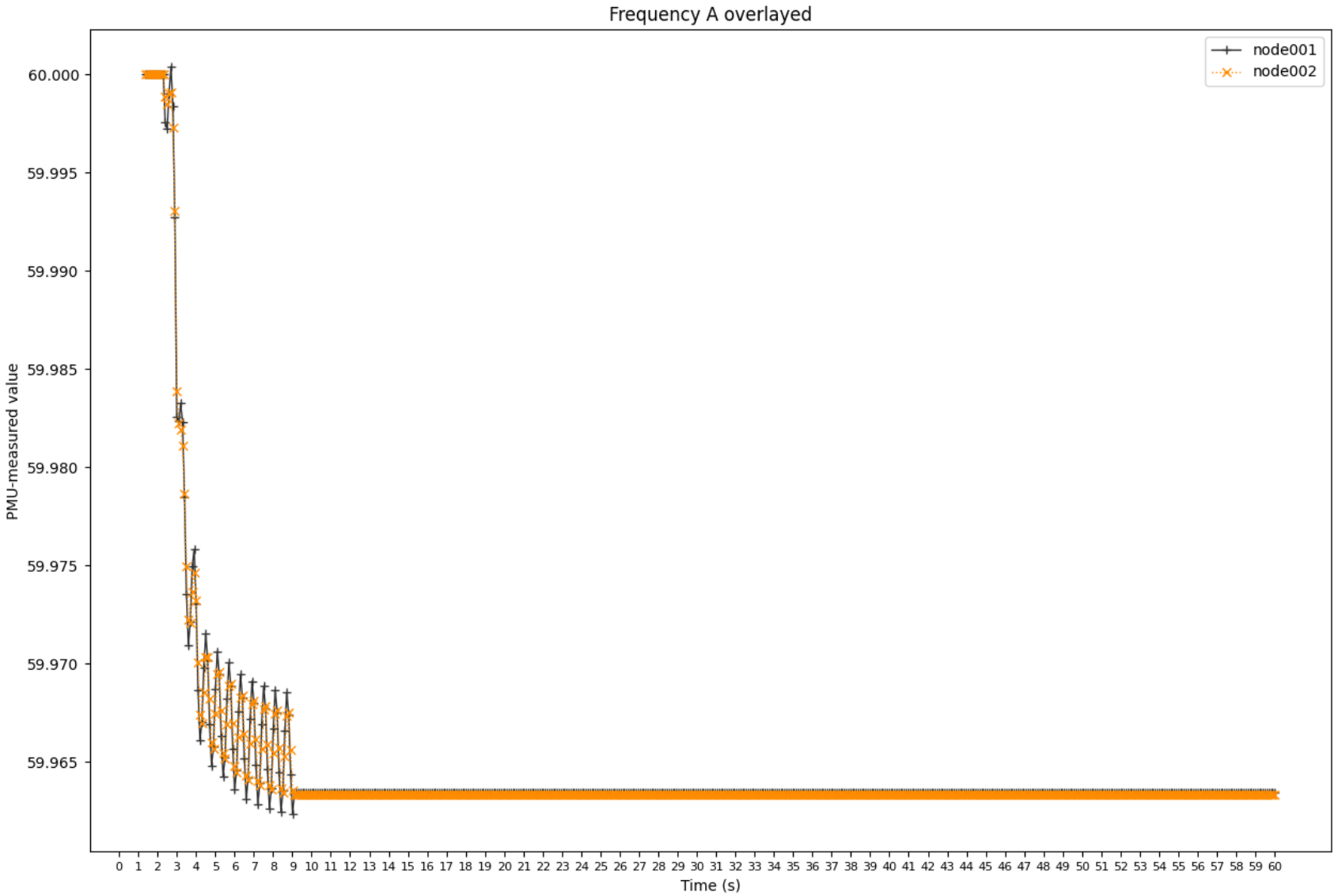


Figure 29: Scenario 1 frequency validation results

5.2.2 Validation 2: Noisy System

In this second scenario, we validate the integration of noise on the measurements. Figure 30 highlights the measurement difference with the previous scenario. Measurements here are noisy and follow the parameters defined by the user, i.e. a noise of average 0.001 and a variance of 0.0005 for the frequency. This noise follows a uniform distribution with known parameters, which can be used by any machine learning algorithms to correctly extract the correct information from these noisy curves.

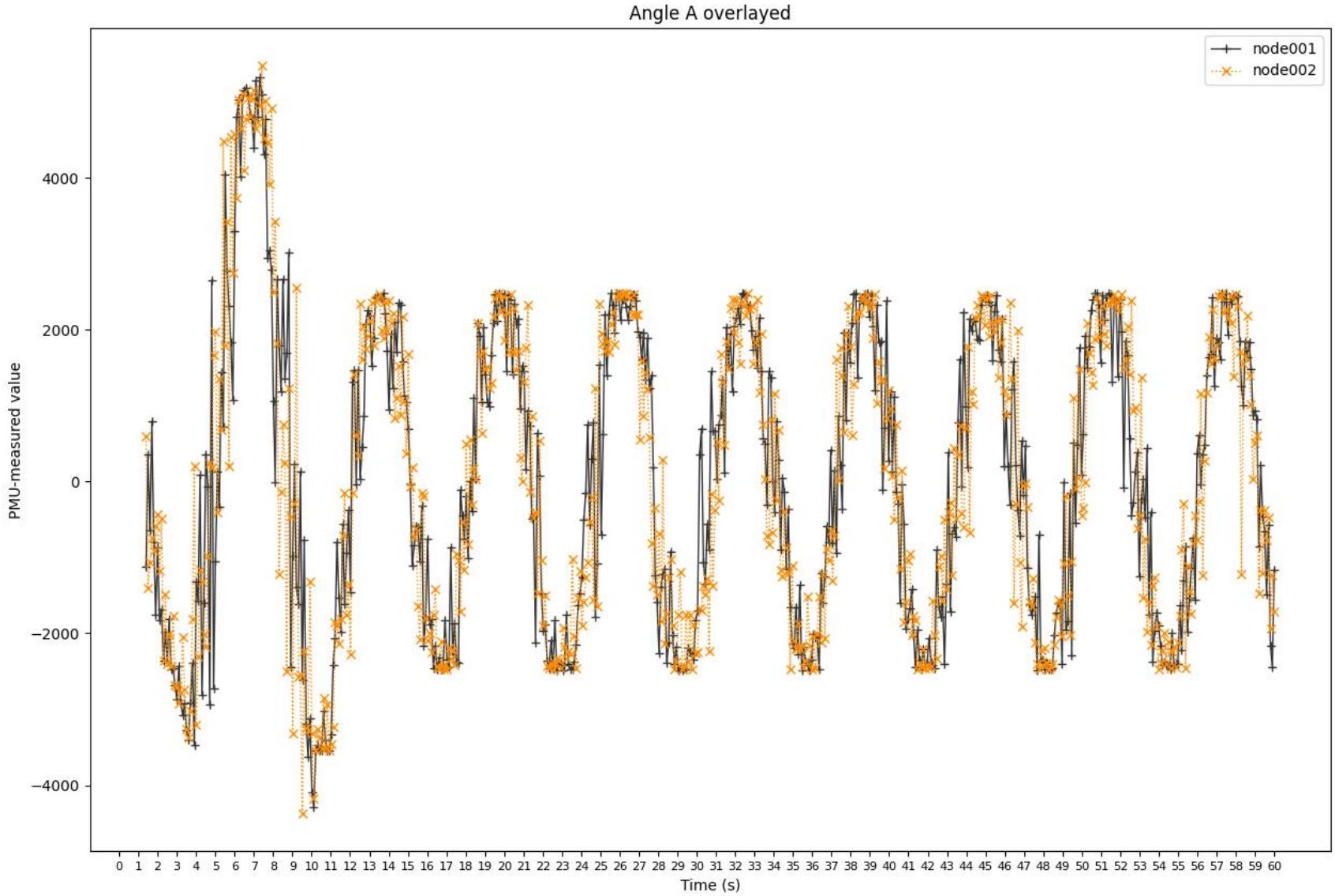


Figure 30: Scenario 2 validation results (Angle A)

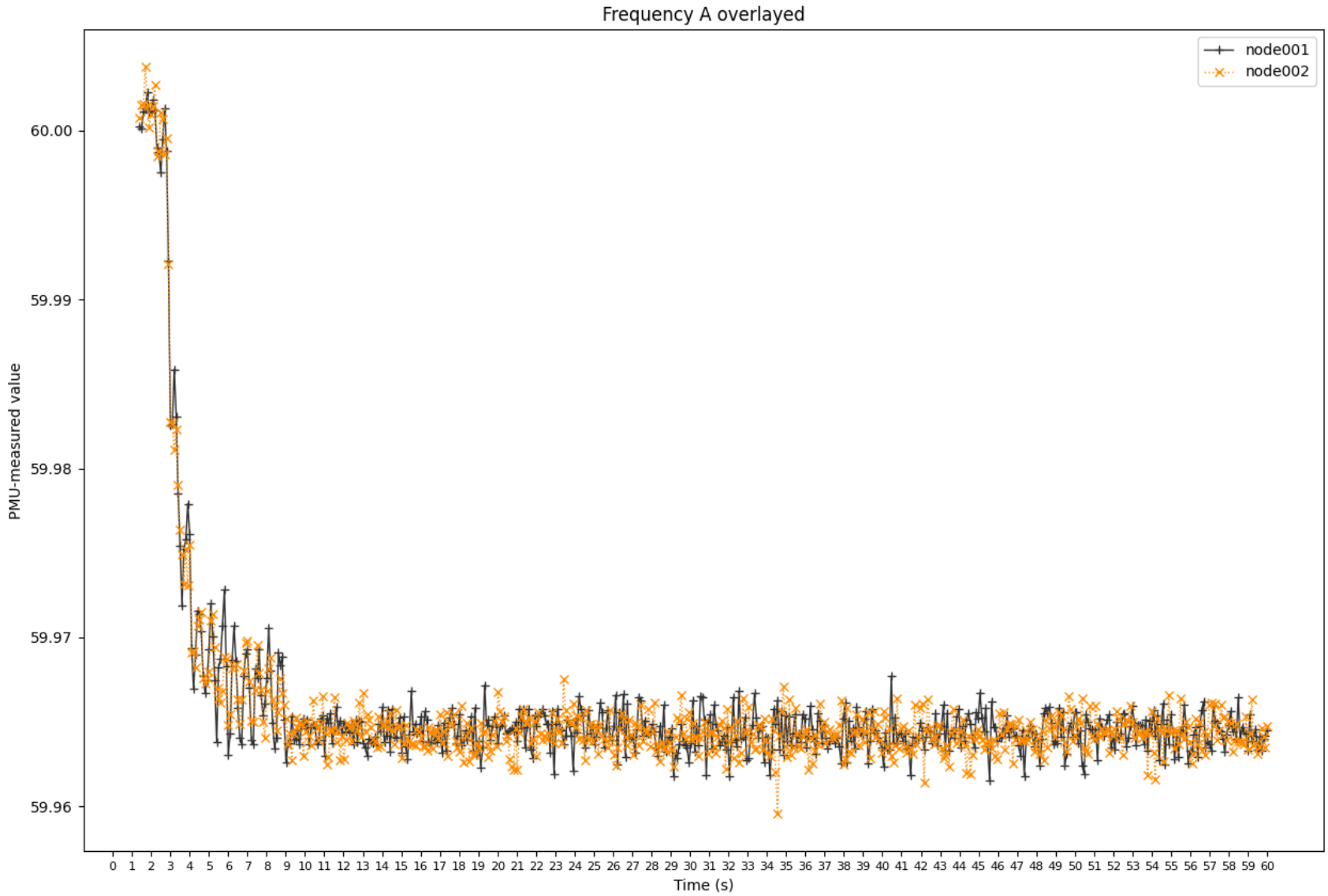


Figure 30: Scenario 2 validation results (Magnitude A)

5.2.3 Validation 3: DDoS Study

Here we study the implementation and impact of the DDoS Hook. In this scenario, a single device is added and connected to the $router[1]$ ($node002$). The attack starts at $t = 30s$ and lasts 10 seconds. The initial parameters of the attack are used, i.e. 100 bytes sent every $10\mu s$, from the $RTU[1]$ to the server on port $1,000$.

The results, shown in Figure 31, show the effects of the attack on the Synchrophasors monitoring. Similar results can be observed for the other curves (Magnitude, Frequency). It can be noted that at the beginning of the attack, a few messages are correctly received by the Control Center but that very quickly the number of messages is considerably reduced. Since the messages are delayed on the network, most of them will be dropped by the aggregator, although some messages manage to be accepted, like the message received at $t = 32.5s$. For a few more seconds after the end of the attack, we can notice that the network congestion is such that it continues to slow down the network for 6 seconds.

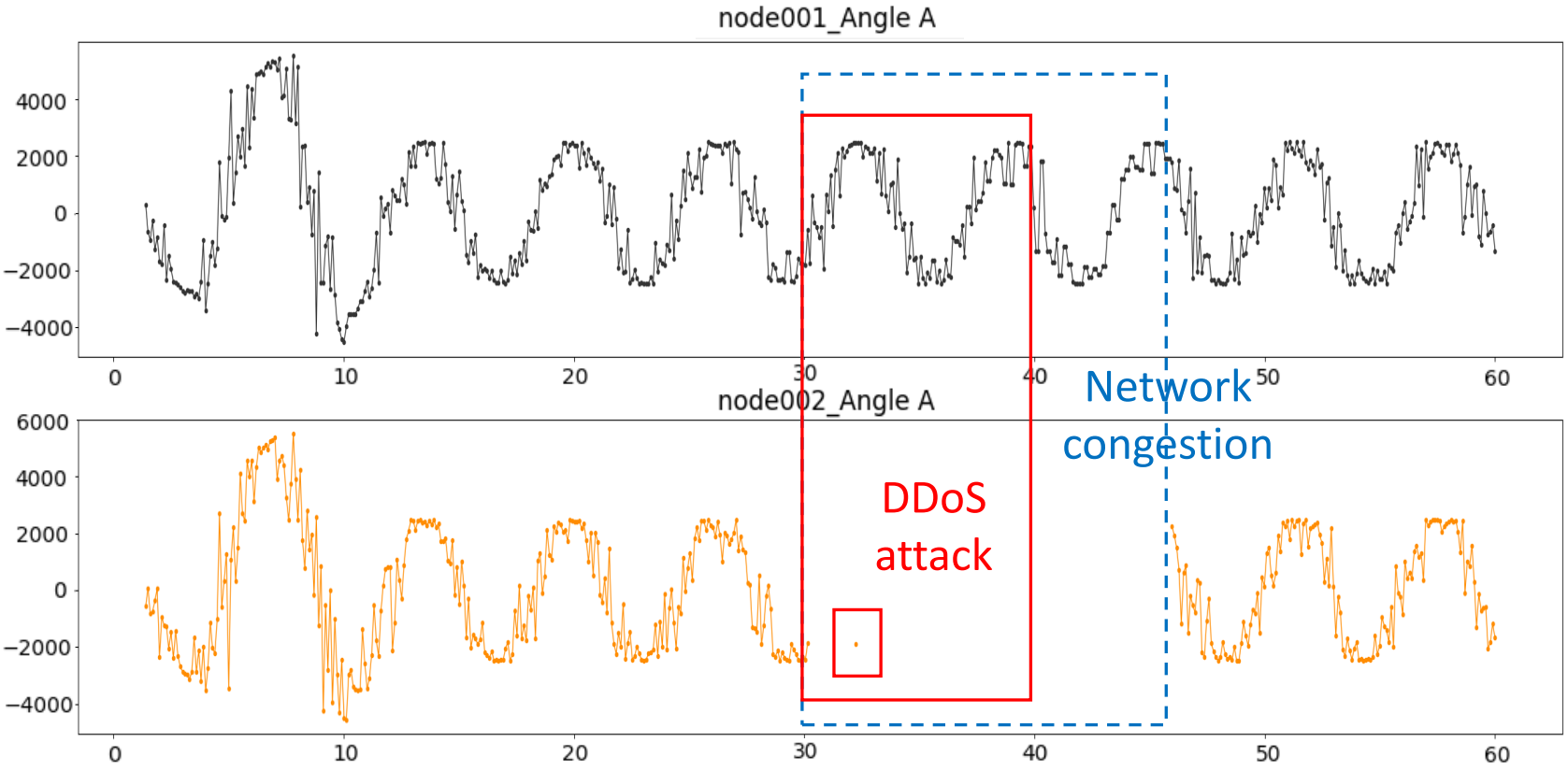


Figure 31: Scenario 3 validation results

5.2.4 Validation 4: Packet Dropping Study

The second validated attack is Packet Dropping. We study the impact of a loss of messages that may be due to network instability or an active attack. Figure 32 shows the impact of this attack which, within a TCP network, causes a multiple sending of the same packet, since the TCP ACK has not been received. Since the attack here is permanent and random, the packets returned will or will not drop again.

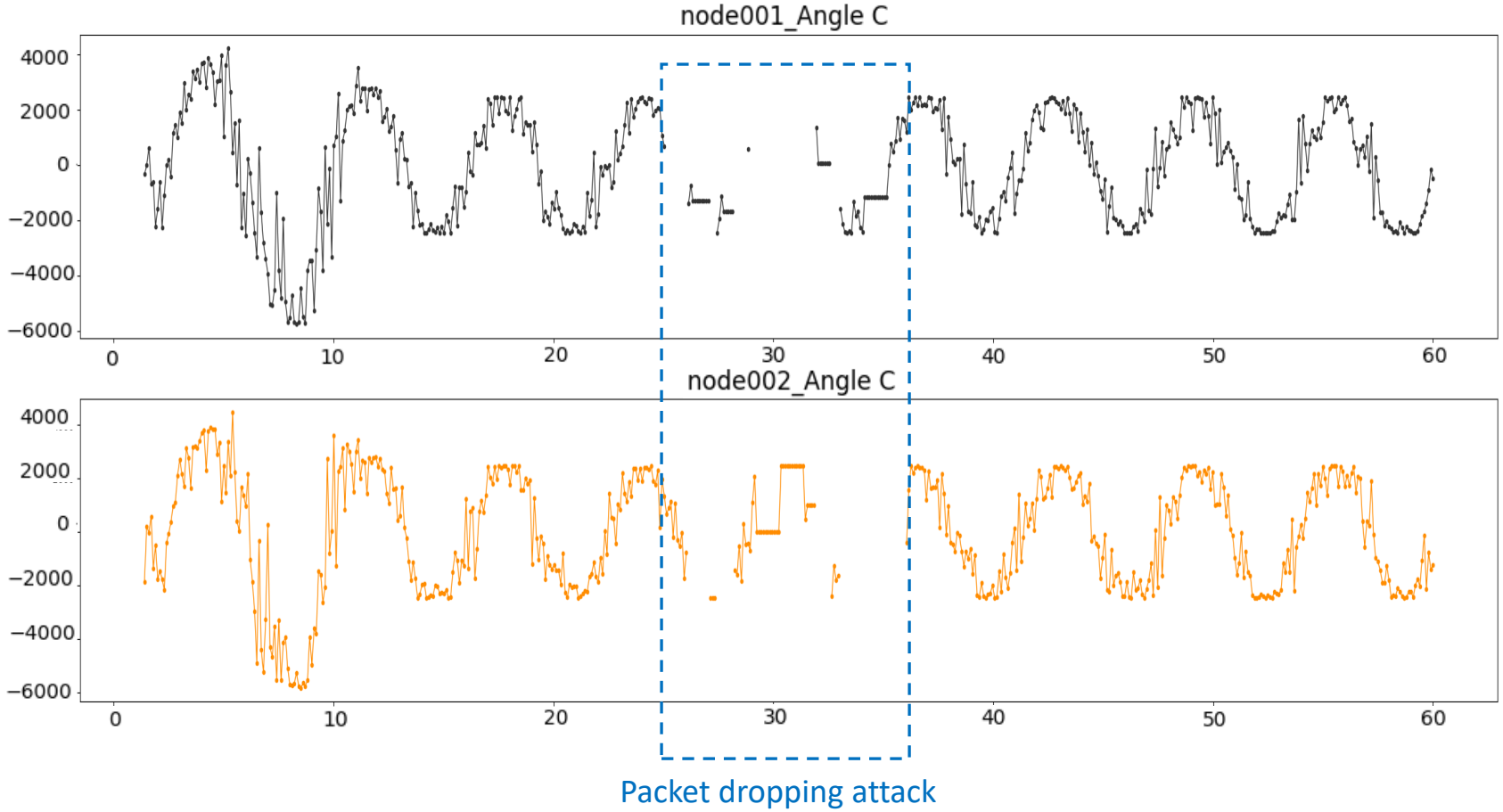


Figure 32: Scenario 4 validation results

5.2.5 Validation 5: Power Grid Faults

The fifth scenario implements a manual fault in the Federate GridLab-D. This fault cuts the overhead line between *node*001 and *node*002, causing an iteration error in the federate, characteristic of a system crash. Figure 33 shows two phases: the first one causing a reset of the monitored values, and the second one, after a few seconds, showing the absence of data caused by the system crash.

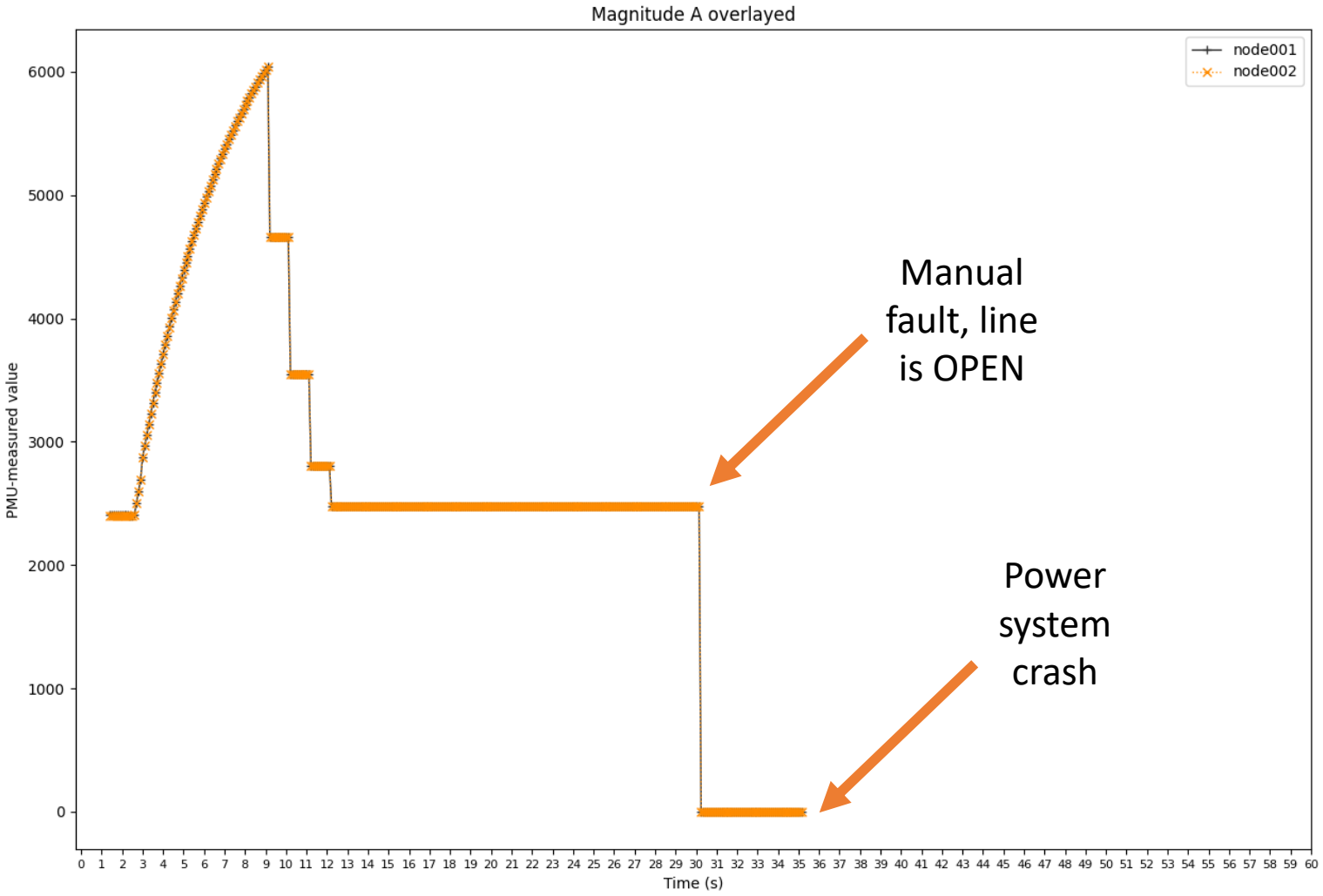


Figure 33: Scenario 5 validation results (Magnitude A)

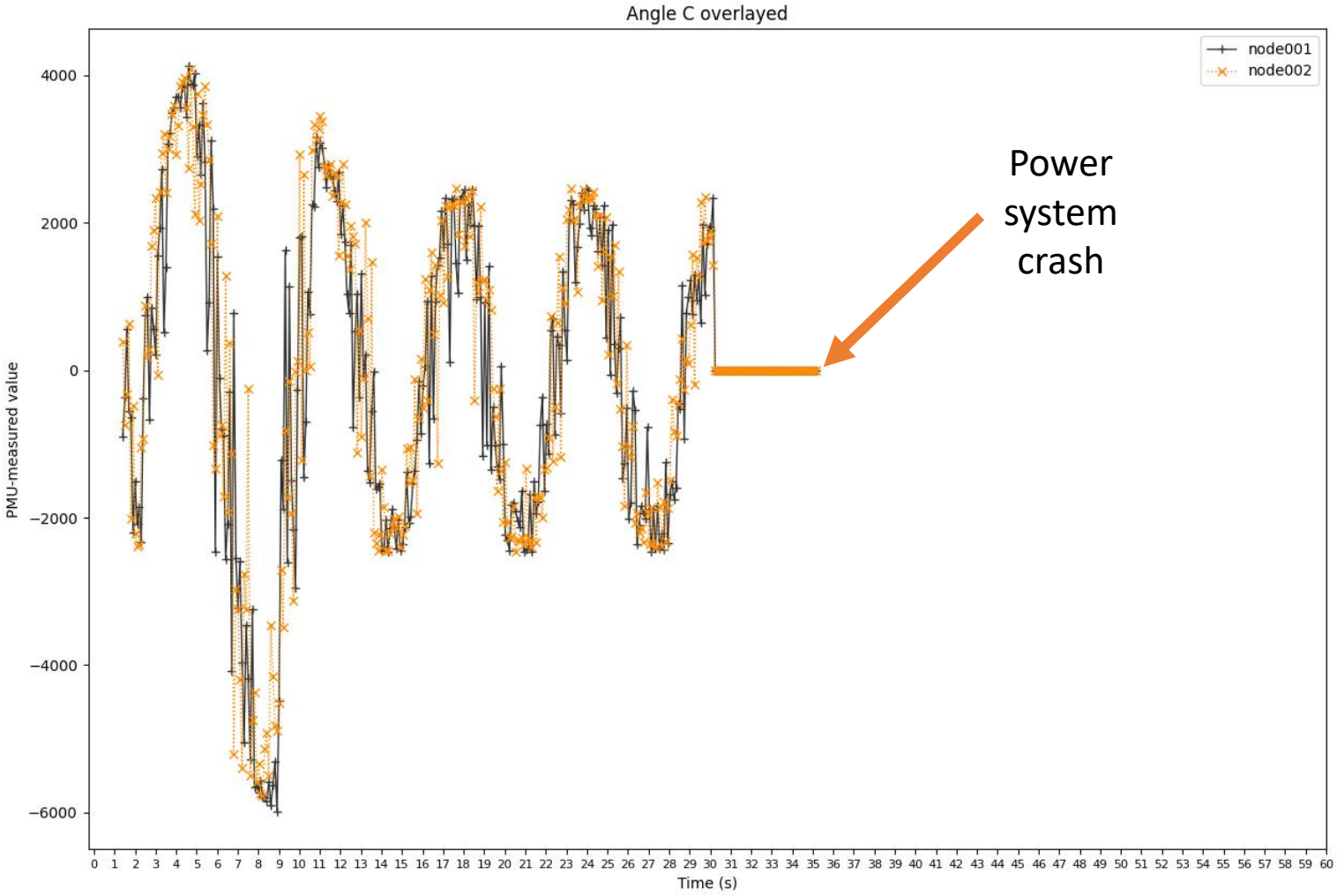


Figure 33: Scenario 5 validation results (Angle C)

5.2.6 Validation 6: Replay Attack Study

We study the replay attack at the level of the Synchrophasor protocol. This attack tries to replay the Synchrophasor stream sent in the first seconds of the co-simulation after $t = 20s$ delay. As shown in Figure 34, the magnitude is clearly visible and different from the Synchrophasor of $node001$ between $t = 31s$ and $t = 41s$.

5.2.7 Validation 7: Desynchronization Attack Study

The desynchronization attack alters the Synchrophasor signal perceived by the Control Center. In this scenario, we apply an arbitrary fixed error of $t = 1200ms$ to study the impact of this error on the Synchrophasor signal. Figure 35 shows a clear shift to the left of the signal from $node002$ during the attack period.

5.2.8 Validation 8: Integrity Attack Study

The integrity attack is validated by means of a fixed value injected into the PMUs for a short period of time. The effects of this attack are in fact directly visible on the curves, as shown in Figure 36.

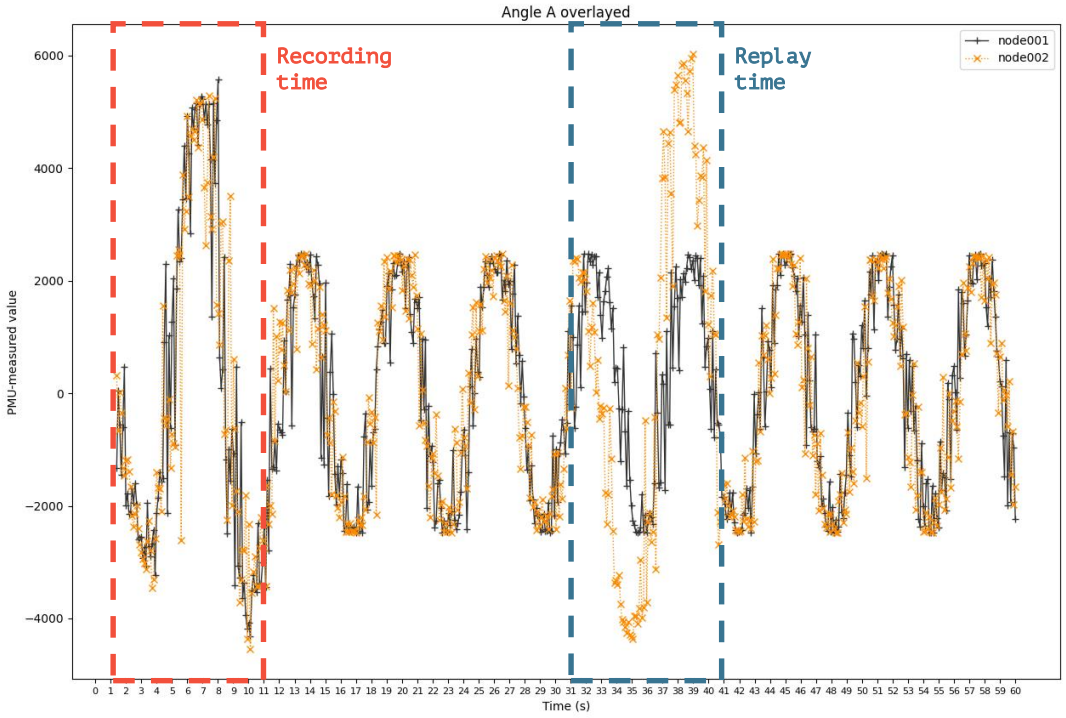


Figure 34: Scenario 6 validation results

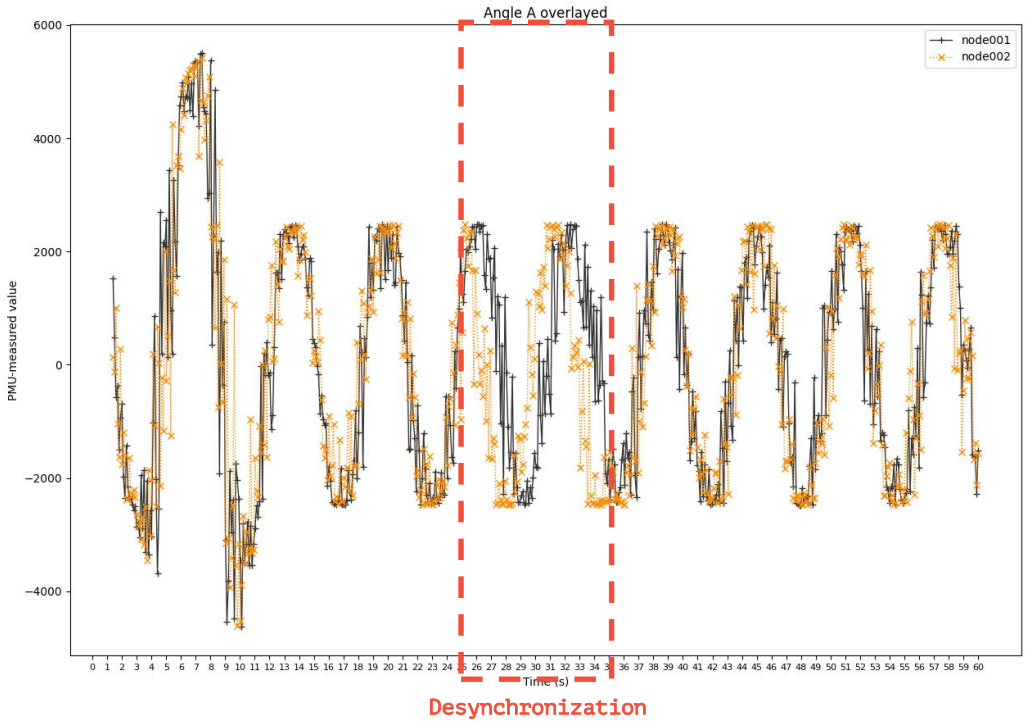


Figure 35: Scenario 7 validation results

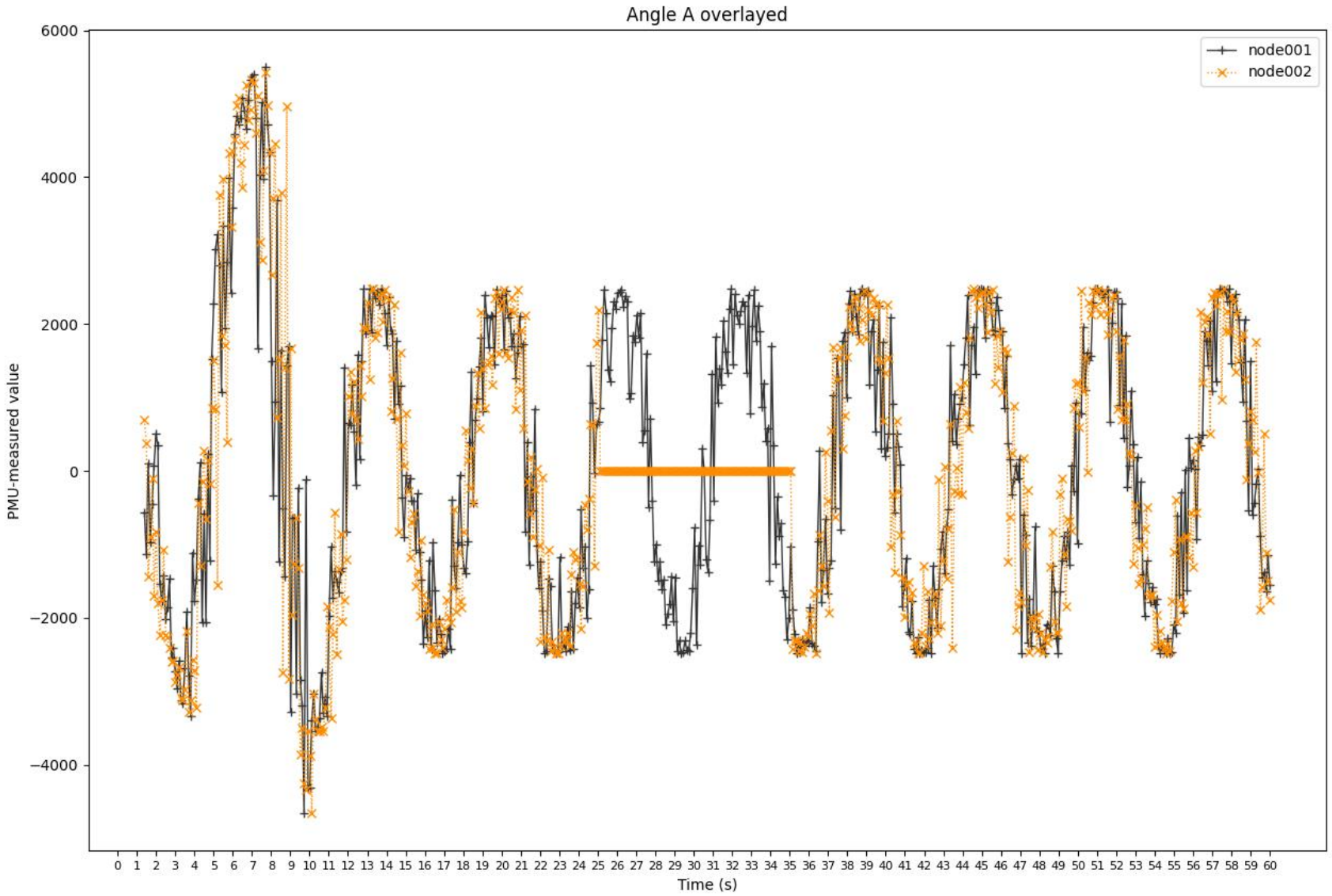


Figure 36: Scenario 8 validation results

5.2.9 Validation 9: Malicious Code Study

We propose in the context of our project a simple implementation of a Power Grid Controller performing a malicious action consisting in doubling the nominal voltage of the electrical system. This example is arbitrary and aims at validating the correlation between the actions of this federate and the effects visible in return by the Control Center. Figure 37 shows the impact of this attack on the IEEE 13 Bus System.

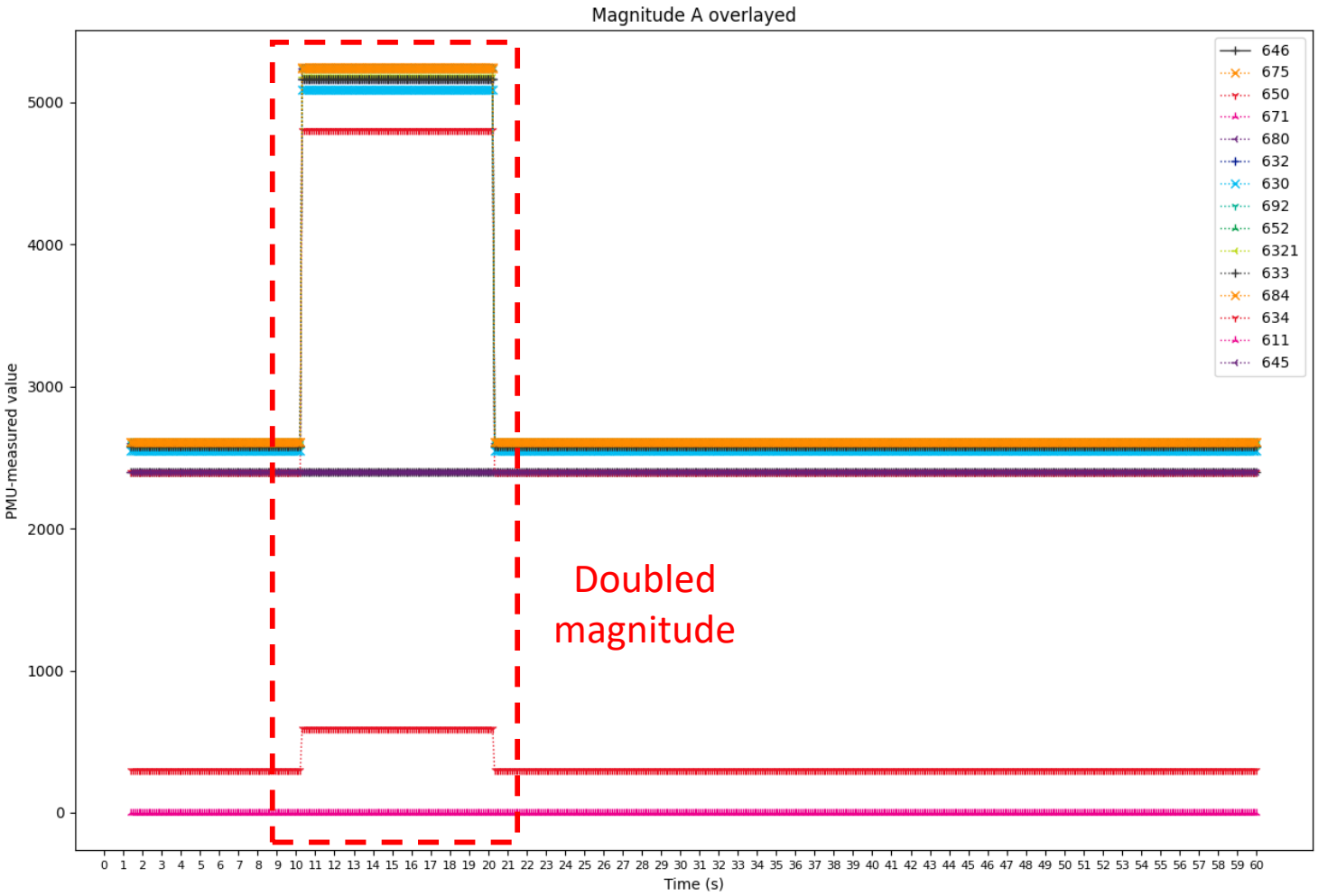


Figure 37: Scenario 9 validation results (Magnitude A)

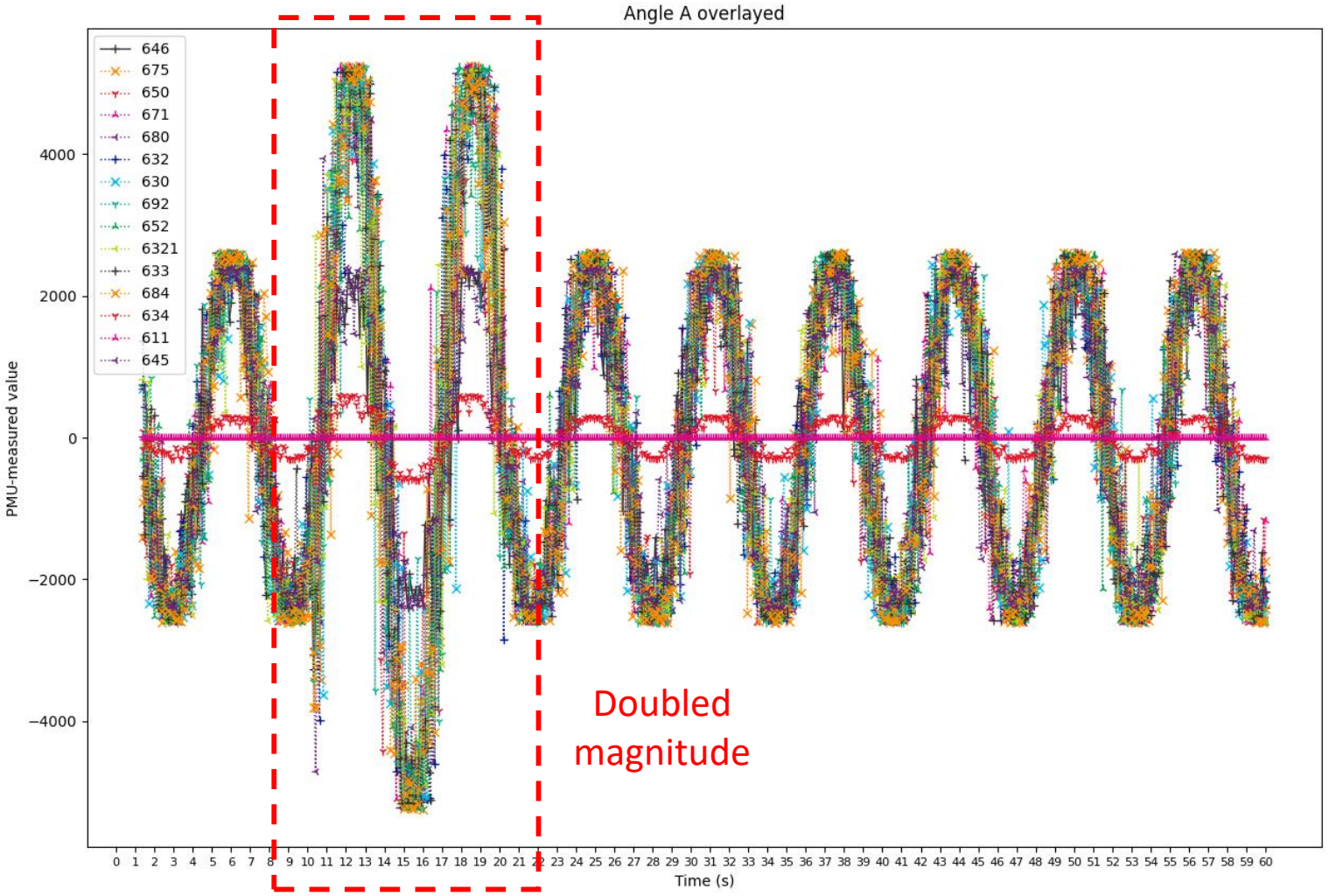


Figure 37: Scenario 9 validation results (Angle A)

5.2.10 Validation 10: Packet Delayer Attack Study

The Packet Delayer is here validated in the same way as the packet dropping attack. Here, the packets will be rejected by the aggregators or, if the delay is sufficient, will be accepted in a later time window than the one initially targeted. Thus, Figure 38 shows the effect of the attack where many messages are rejected and some are accepted with a visible shift to the right with respect to the $node002$ curve.

5.2.11 Validation 11: Multiple Attacks Study

This scenario aims at validating the proper functioning of the Hook Sequences. Here, different attacks specified in the Figure 27 are co-simulated: Packet Delaying, Integrity Attack, Desynchronization attack and Packet Dropping attack. Figure 39 shows the results obtained with the parameters used.

5.2.12 Validation 12: Dynamic Pricing Study

The study of attacks on Dynamic pricing shows the impact of attacks on the communication network on the level of clearing price negotiation by the different entities of the power grid. Here, PMUs and Demand-Response are disabled, in order to allow a fast co-simulation of higher simulation durations. Here $6,900s$ are co-simulated, i.e. 1 *hour and* 55 *minutes*. The Market Federate is enabled and generates curves at the end of the co-simulation representing the following metrics: (1) Cleared price, (2) Mean load, (3) Number of buyers, (4) Total price asked, (5) Total load. For this first scenario, we implement a power consumption of HVAC systems that varies greatly on a few minutes basis, allowing us to obtain concrete effects for a simulation of moderate duration.

Here, and for the other Dynamic Pricing-related scenarios, the Market Federate implements an arbitrary logic that we have developed to study the impact of clearing price on the power consumption of HVAC systems, given the configuration of the deadband thermostats. This federate charge \$0.0379 ($pricePerKW$) for every $kiloWatt$ (kW) below 3000 and \$0.0385 ($priceperKWAbove$) for every kW above:

$$clearing\_price = pricePerKW * 3000 + (total\_load - 3000) * priceperKWAbove \quad (7)$$

Where $total\_load$ is the total load from all controllers during the actual market.

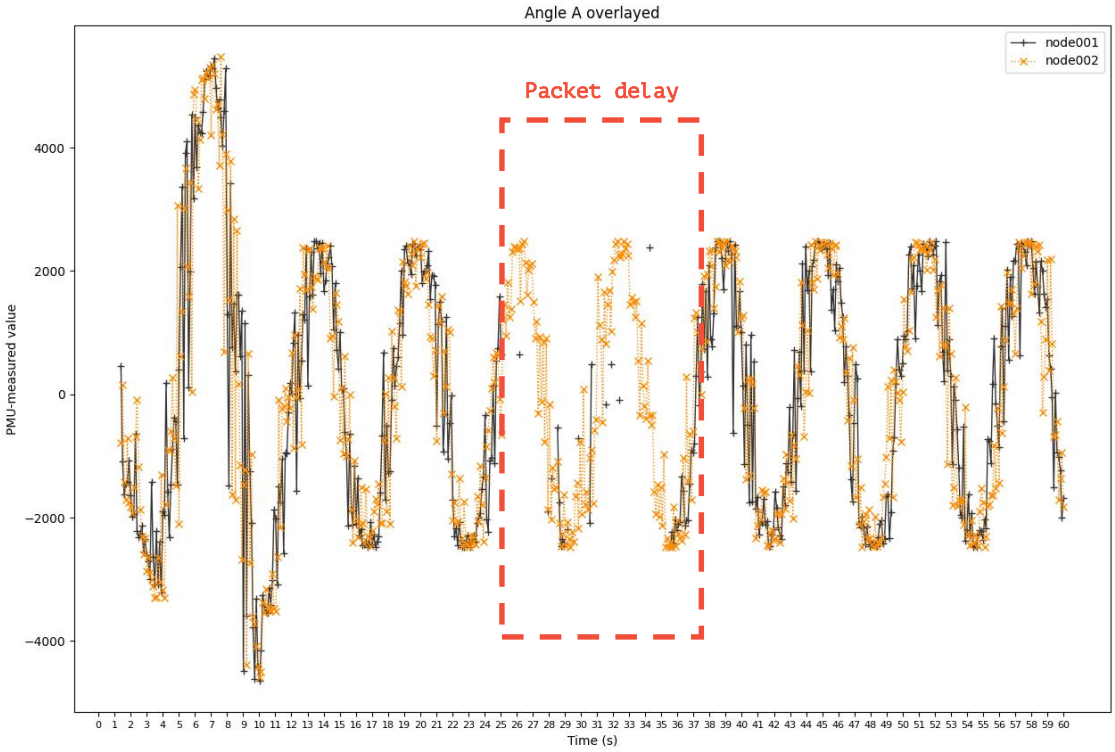


Figure 38: Scenario 10 validation results

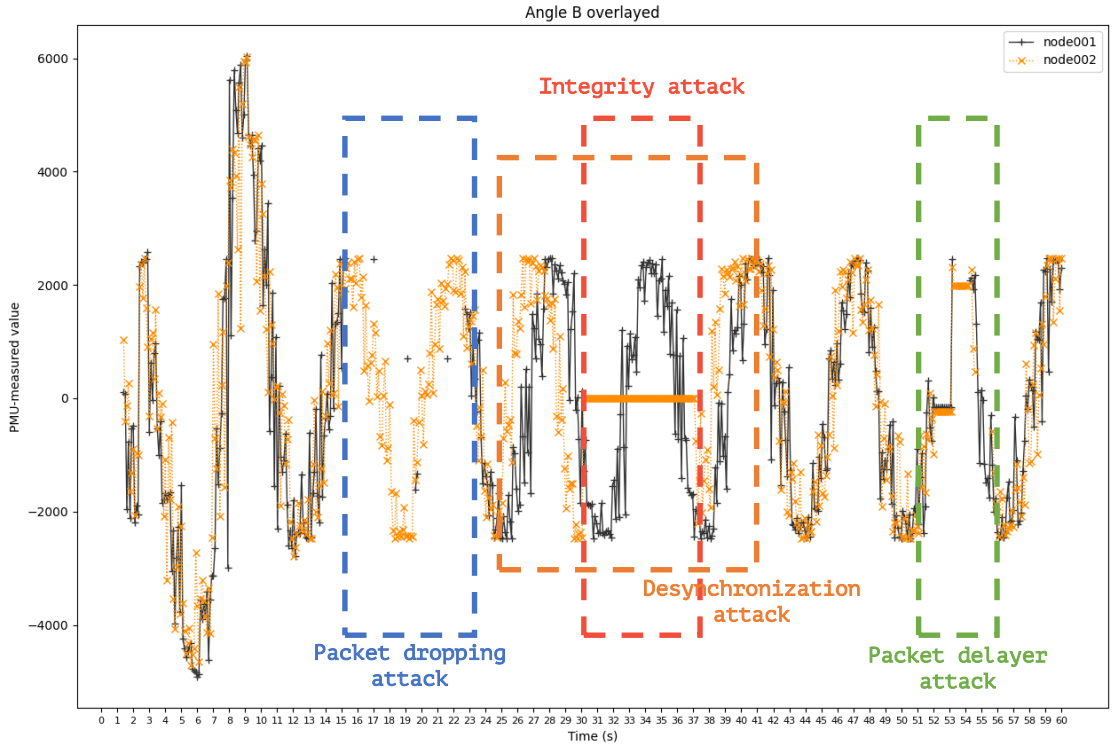


Figure 39: Scenario 11 validation results

For the IEEE 13 Bus System equipped with 14 controllers, Figure 40 presents the results obtained for four metrics.

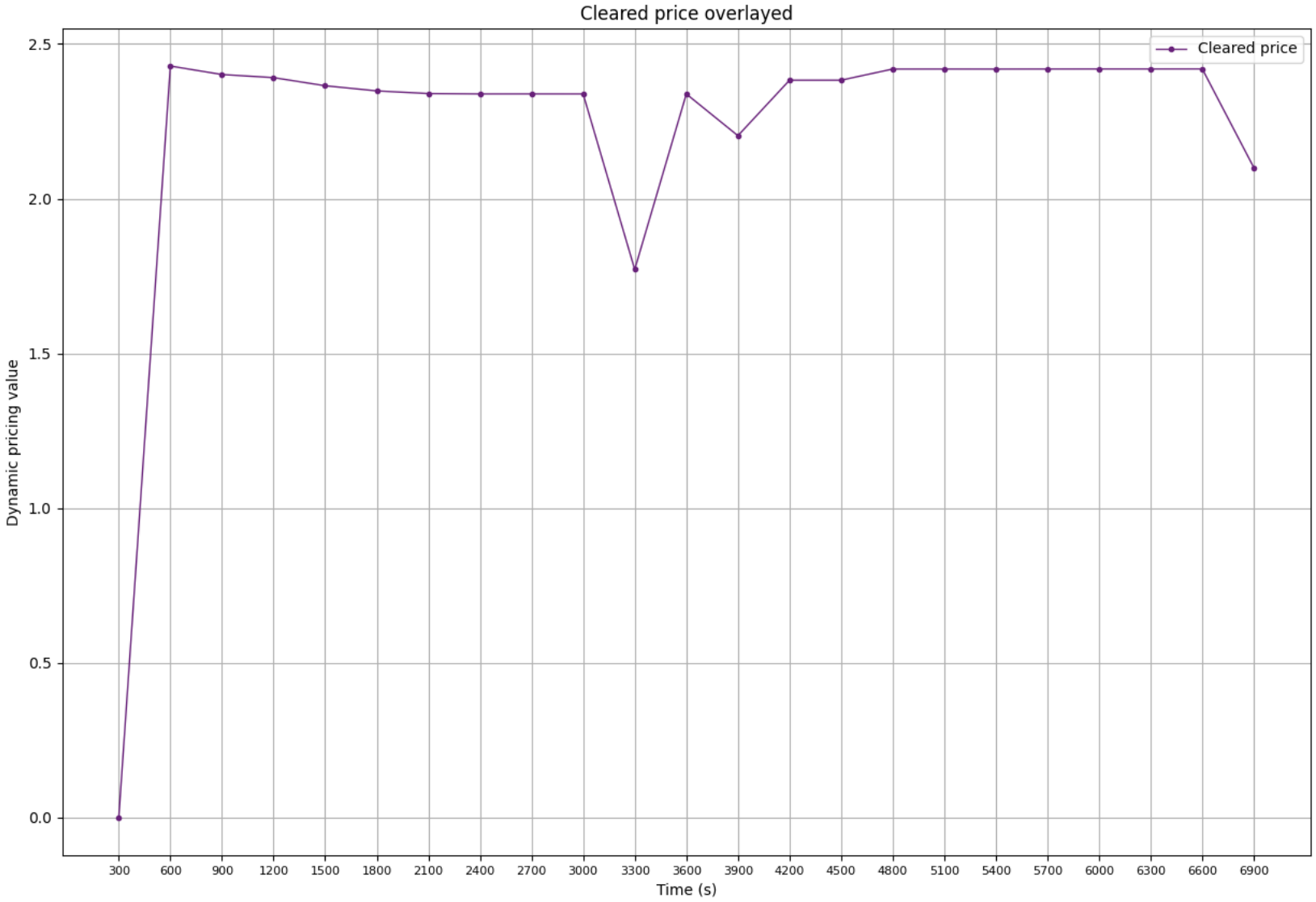


Figure 40: Scenario 12 validation results

5.2.13 Validation 13: Dynamic Pricing Attack Study

This second study validates the hook aimed at the integrity of messages exchanged during Dynamic Pricing. The attack fixes the value in terms of price and quantity. The curves presented in Figure 41 show an increase in the cleared price between $t = 1,200s$ and $t = 3,000s$.

Figure 41: Scenario 13 validation results

5.2.14 Validation 14: Multiple Attacks and Dynamic Pricing Study

Finally, we study the impact of attacks related to network communication (Packet Delaying, Packet Dropping) in order to measure the impact of these attacks on the cleared price determined by the Market federate. We can see in Figure 42 that the number of customers drops from 14 to 9 during the packet dropping attack, causing a decrease followed by an increase in the clearing price due to the use of the Dynamic Pricing Attack. Finally, the total asking price will be directly impacted by the attacks.

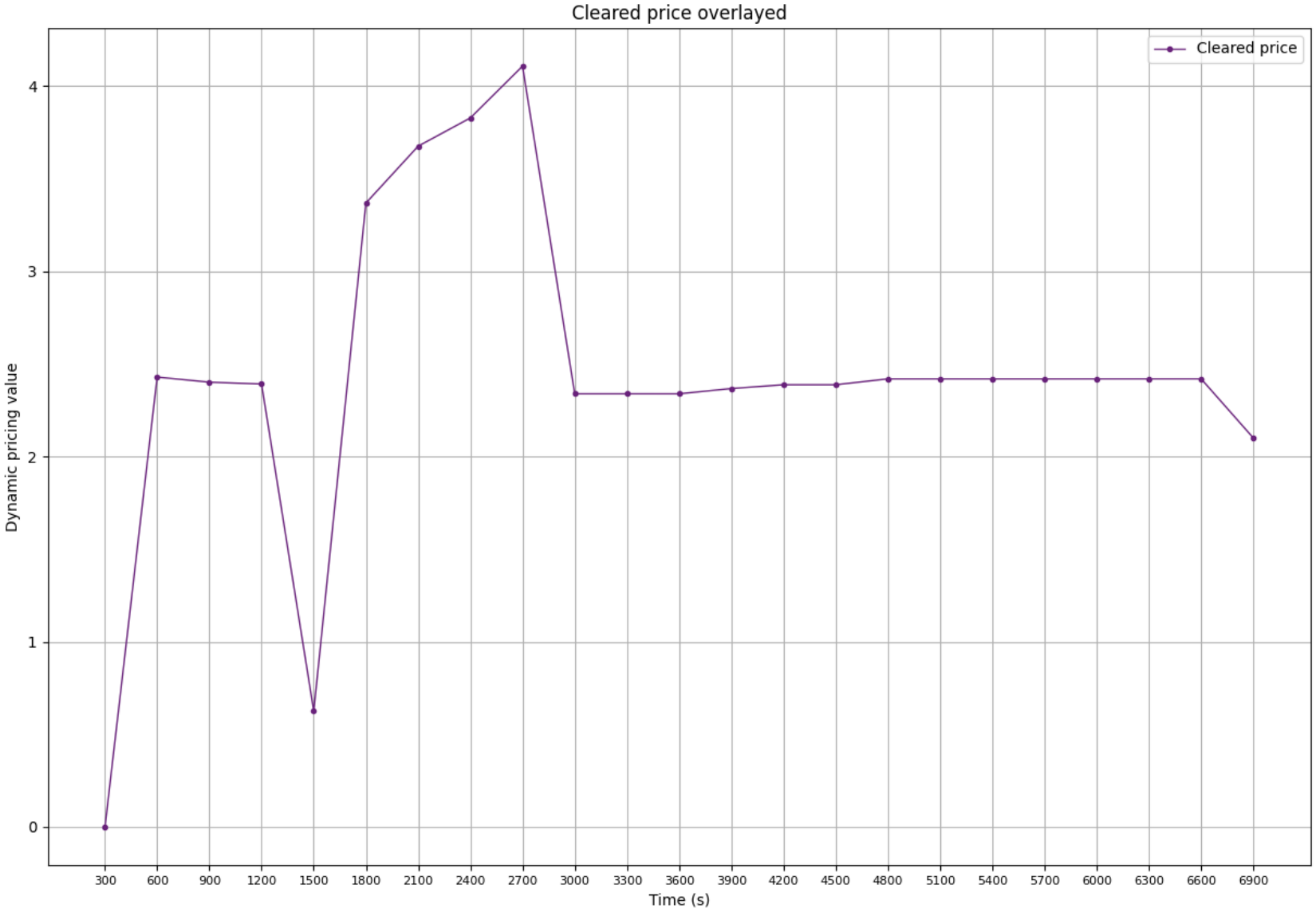


Figure 42: Scenario 14 validation results

5.3 Results

In this Section, we co-simulate the (modified) IEEE 13 Bus System using our proposed framework. We first detail the different parameters and capabilities of the co-simulator involved, then we analyze the results. Finally, we highlight the advances introduced by our project, compared to the various existing co-simulation platforms.

5.3.1 Testbed Configuration

We use the graphical interface of the Project Generator for the generation of a new dataset of the IEEE 13 Bus System. The parameters are summarized in the Appendix F.

This scenario aims to study the behavior of the system under communication attacks but also allows to measure the impact of some attacks on the power grid. To do this, we have implemented a non-realistic Grid Controller whose operation is detailed in the algorithm 3. In addition, all federates outside the Market Controller are activated here.

In order to visually compare the impact of different attacks launched manually during co-simulation via the Federate Attacker, we ran the testbed a first time with the above-mentioned parameters. The second time, we added a manual attack at $t = 73.1s$ targeting the message integrity of $RTU[9]$, setting the frequency to 40. The attack is then stopped at $t = 138.9s$.

Algorithm 3: Power Grid federate custom function use case

Input: Pre-processed data with PMU-related measurements, the current time;

Result: Updates the Power Grid Controller publications

Computes the mean frequency *meanFrequency* from all PMU measurements;

if *Number of PMU-received messages* > *0* **then**

  Update the bus SWING voltage such that

  $voltage_{A|B|C} = (base\_value * meanFrequency)/60.0$;

  Where *base_value* is the stored or default associated value computed by the Project Generator;

  if *meanFrequency* >= *61* **then**

    Open the $Capacitor2$ switches A, B and C;

  else

    Close the $Capacitor2$ switches A, B and C;

  end

end

Return the updated publication values;

5.3.2 Testbed Execution

During this execution, we used a machine running Linux Mint 19.3. The machine is equipped with 32Gb of RAM memory, and data writing is done in real time on a NVMe disk. Each federate run in the same environment and the federates GridLab-D, OMNeT++, Visualizer, Power Grid Federate and Attacker are used. The co-simulated 15 minutes are executed at maximum speed via the speed setting in the Visualizer Federate and are paused at various times to take screenshots or launch attacks.

OMNeT++ Federates, GridLab-D and Power Grid Federate write the outputs of the co-simulation in real time.

Figure 43 shows the Visualizer Federate display for the GLM model used. The second image shows the ability to display detailed information for each communicating or electrical node.

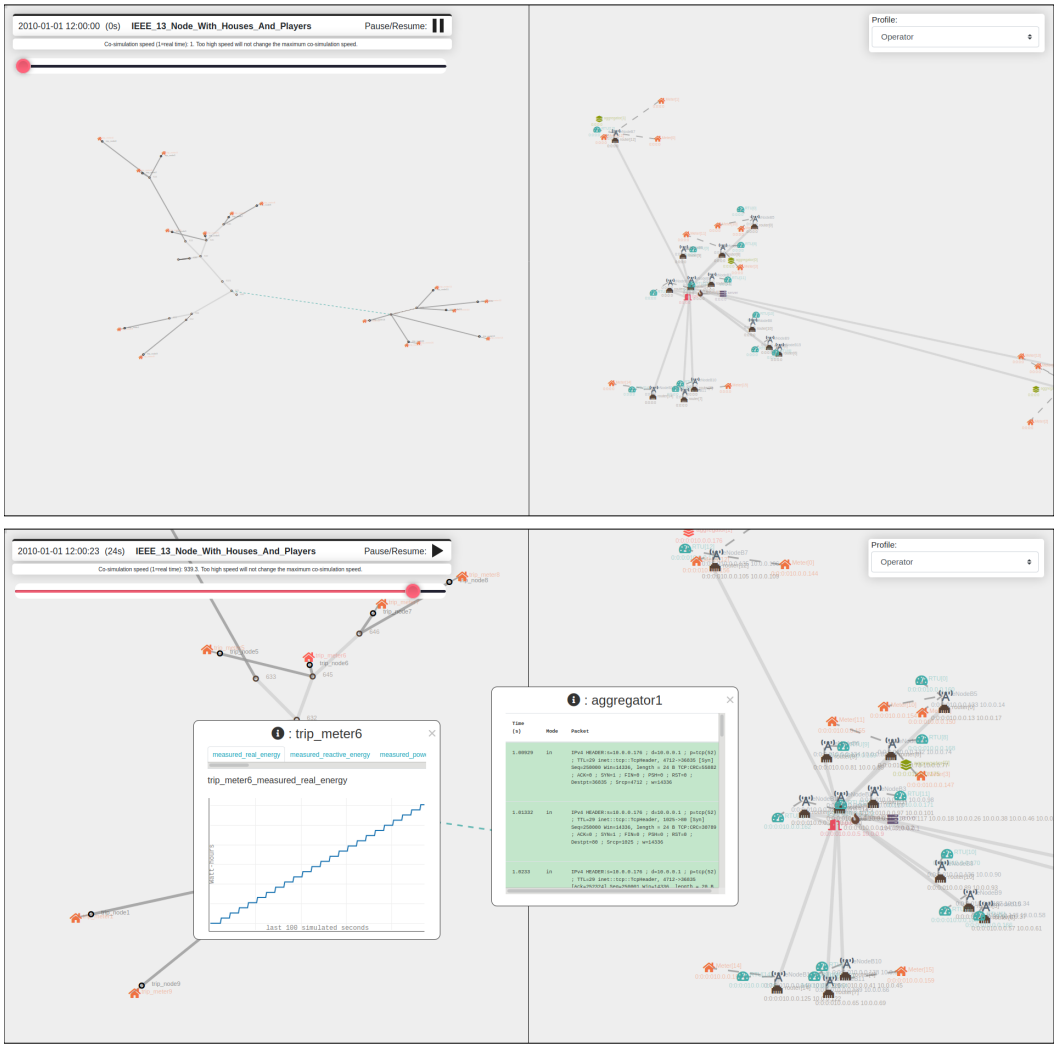


Figure 43: Use case: Visualizer view

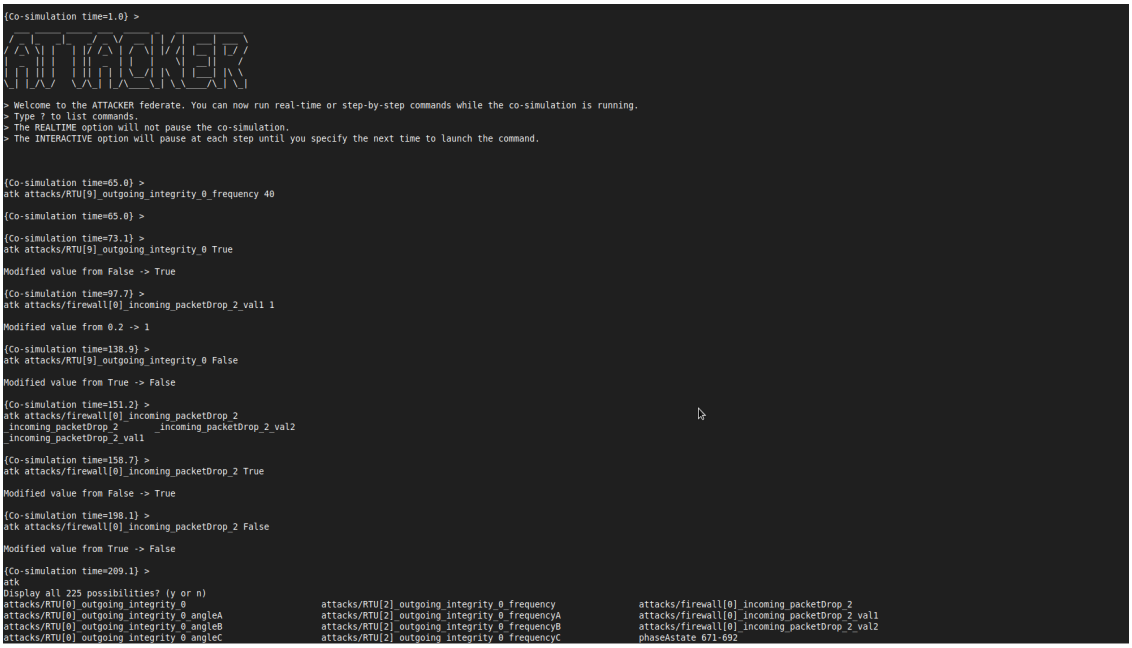


Figure 44: Use case: Attacker view

Live attacks are launched manually via the federate attacker. The Figure 44 presents a concrete view of how this federate works, showing the various interactions we had with it during the co-simulation, via a basic terminal.

Finally, this co-simulation integrates the dynamic placement mechanism of the communication devices via the Project Generator. The Figure 45 shows an overview of the OMNeT++ network, and the Figure 46 the Firewall Hook system, as the latter implements different Hooks.

5.3.3 Dataset Generation and Results

After the full execution of this testbed, we extract different results using python scripts for curve plotting and statistics retrieval. First, we generate the different files that will be used as dataset, using the Dataset Generator. The statistics for the extracted data are presented in the Appendix G. Thus, we generate a file for each device that has been listened to by the attacker, as well as the Control Center, by default.

The Dataset generator also retrieves files from GridLab-D containing direct and unnoised values from the power grid. Among these files are 39 files for the houses, 16 files for the nodes, one file for the switch, 16 files for each of the *triplex_meters* and 15 files for the *triplex_nodes*.

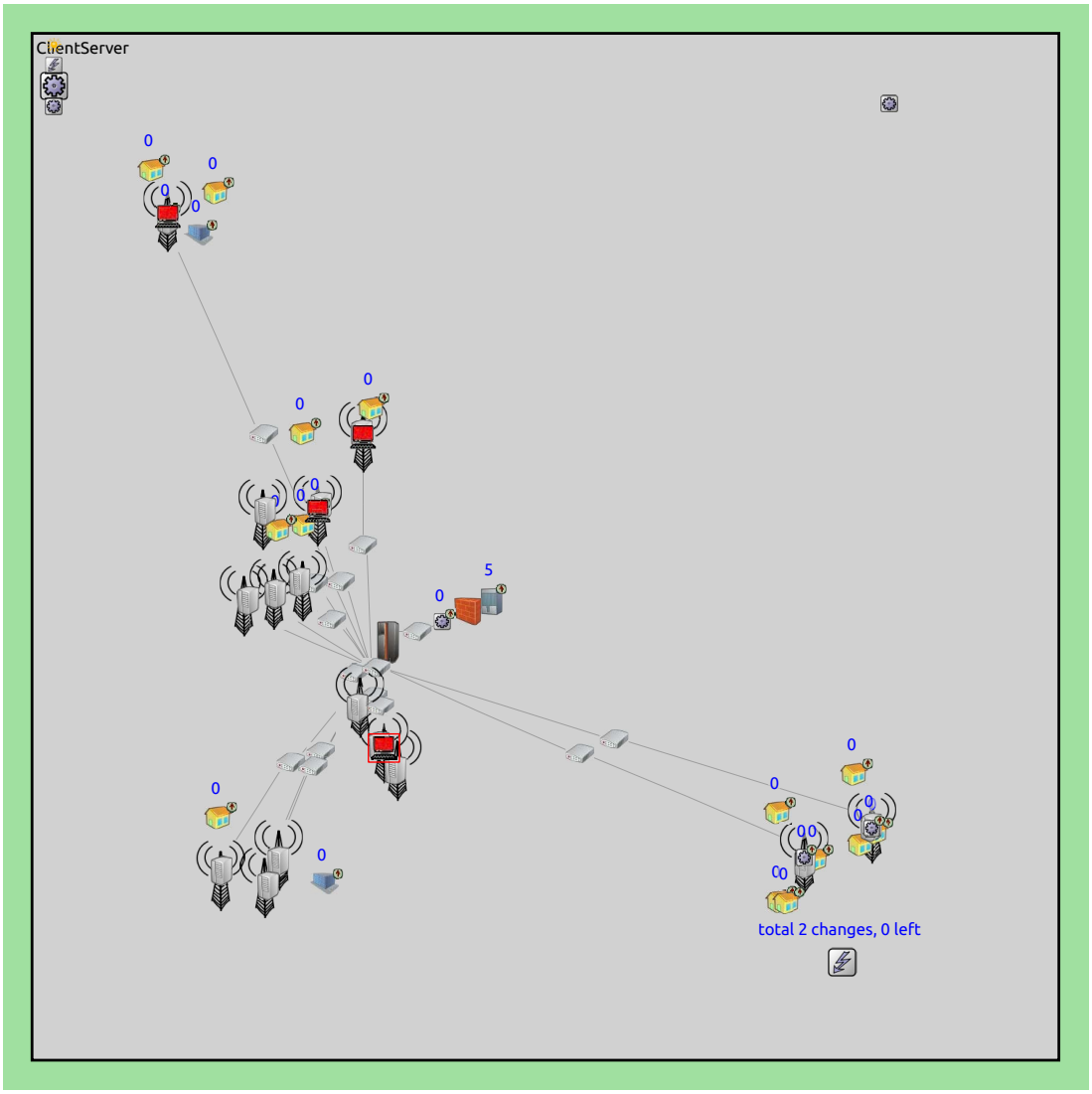


Figure 45: Use case: OMNeT++ topology

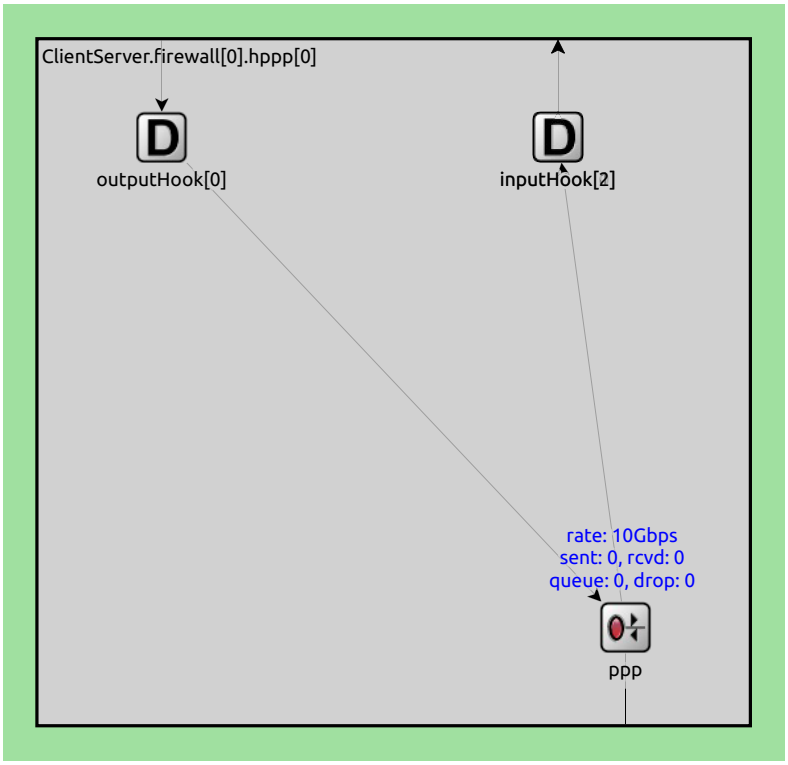


Figure 46: Use case: Firewall architecture

The *vec* statistics file from OMNeT++ has been generated for this co-simulation and weighs 1.3Gb. This file contains all the statistics from all implemented devices from *INET*, *simuLTE* OMNeT++, as well as the hooks.

We study the communication network statistics generated by the *statistics* hook from the firewall point of view. The results, presented in Figure 47, show the expected correlation between the attacks, the network latency, the number of packets exchanged, and the total size of messages passing through this device. Latency here represents the latency added by network congestion, router capacity and possible attacks.

We can notice the periodic variation of the network latency every 60s, caused by the attack 2 repeating itself throughout the co-simulation, causing some packets to be refused by the Phasor Data Concentrator, and others to be accepted, when they arrive at a new time window. Thus, these messages will impact the latency seen by the firewall. Two latency and communication peaks can also be isolated. Between $t = 519.017401s$ and $t = 559.997847s$, then between $t = 599.019114s$ and $t = 614.799273s$, the Control Center enters a saturated state, as shown in Figures 48 and 49 from two generated Wireshark files, where the *Destination unreachable* error indicates that the port 1000 is not implemented

by the system, generating even more communication, as expected. At this time, the OpenADR protocol also starts, and smart meters communicate HTTP messages that will further worsen network congestion.

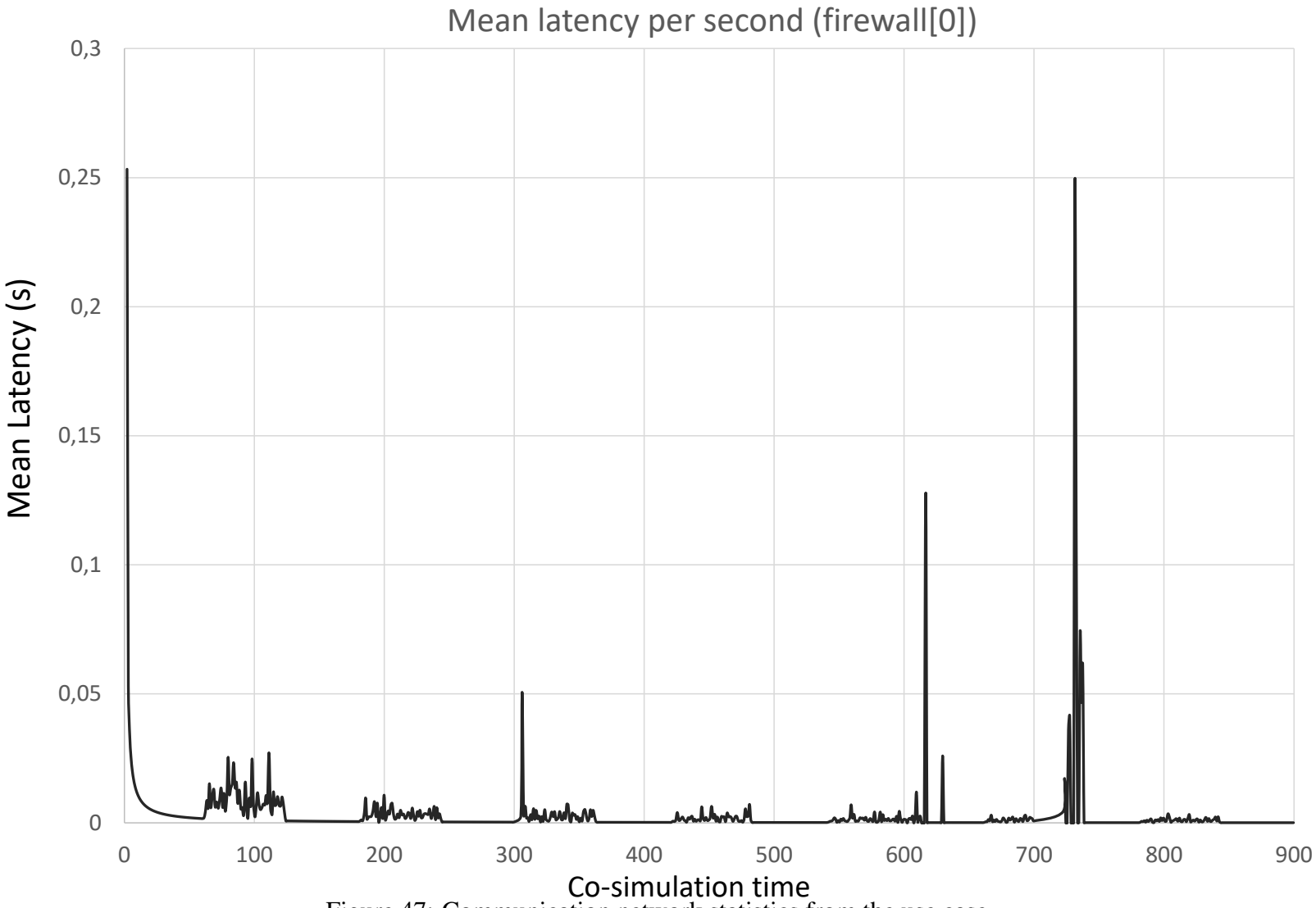


Figure 47: Communication network statistics from the use case

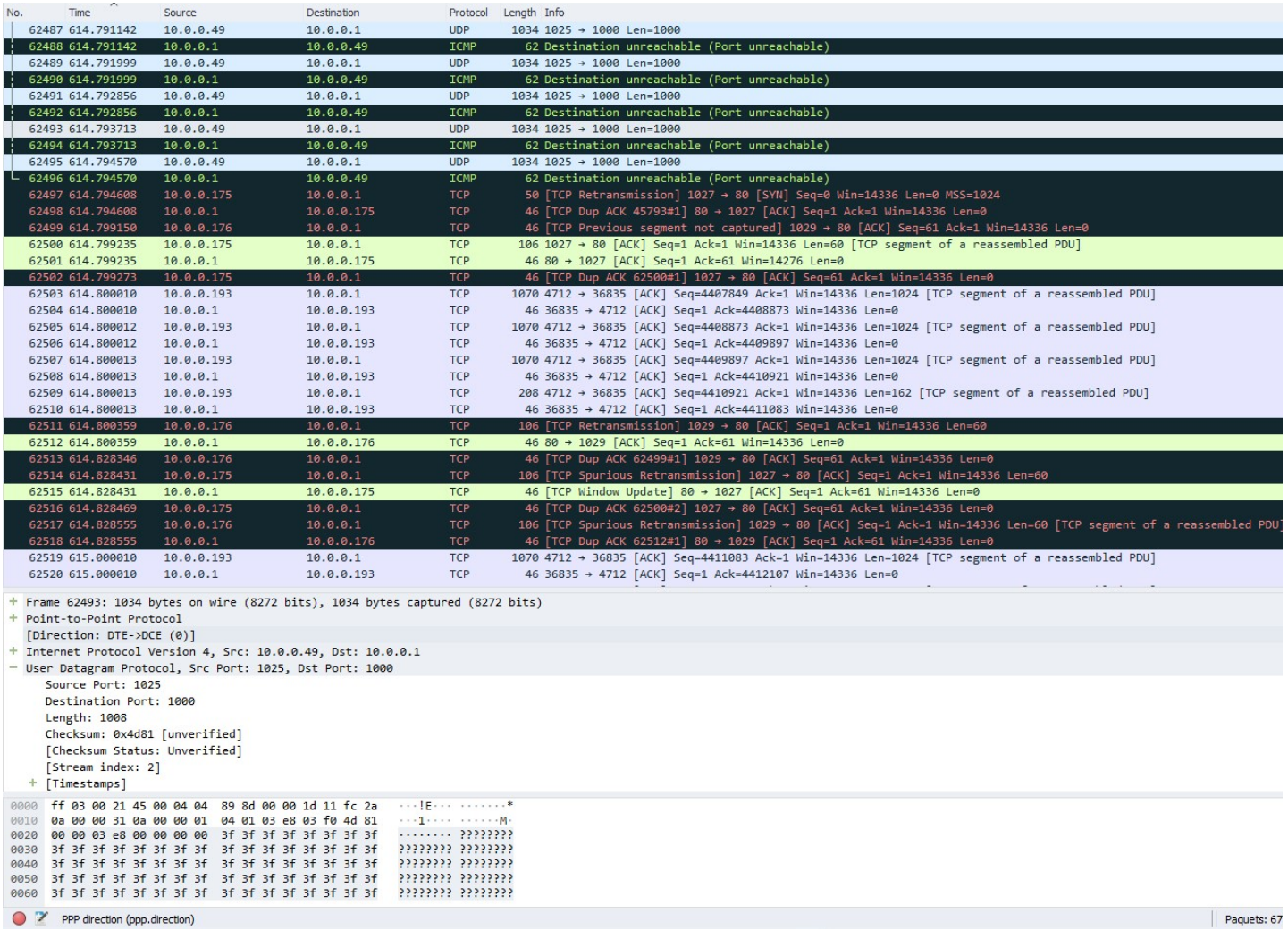


Figure 48: Control Center under DDoS attack

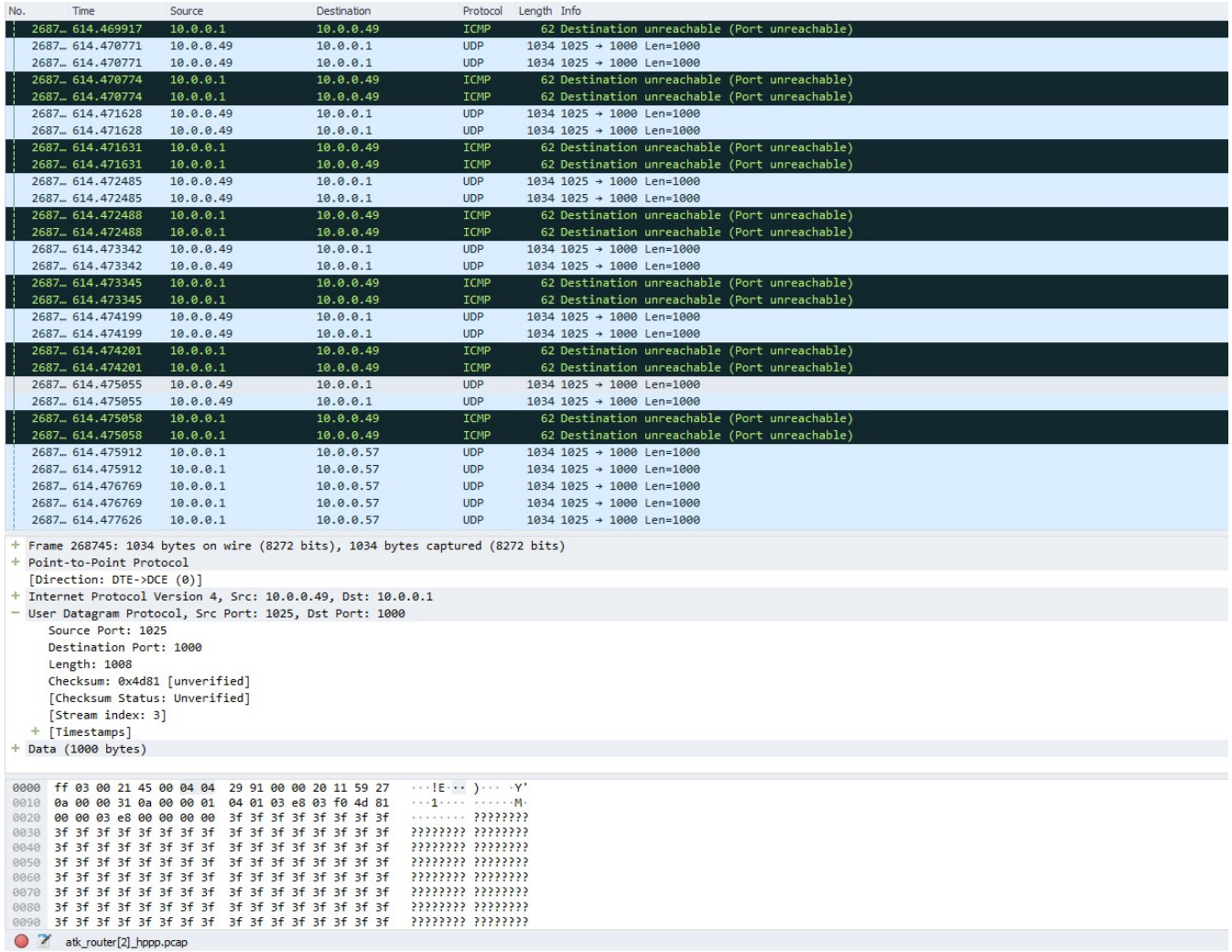


Figure 49: Intermediate router with Attacker-generad DDoS messages

Finally, we study the state of the power grid following the various attacks via the Synchrophasor protocol from OMNeT++, the Power Grid Controller implemented here, and the graphs generated by the Dataset Generator. Figure 50 shows both co-simulations results: the state of the system without attacks via the Attacker Federate, and the evolution of the system following these attacks. Both implement the *permanent*, *one-time* or *repeated* attacks from the Project Generator-defined scenario.

Figure 51 shows the different impacts of the attacks on the Control Center Power Grid State monitoring. We first illustrate the impact of the Packet Delayer hook on communications, causing a decrease in the number of messages received by the Control Center. Then, the *Frequency B* sub-figure illustrates the variations of the measured frequency in the system, with emphasis on the impact of the FDI attack on the values monitored by the $RTU[9]$. The desynchronization attack, despite the presence of noise, shows a variation in the period of the sine function. The sub-figure on *Frequency A* illustrates the impact of the DDoS attack on a system before more bandwidth than during our validation. Here, the DDoS attack impacts the correct exchange of TCP messages and the reception of the associated TCP ACKs, causing the TCP protocol to send the same message again until it receives the TCP ACK. Finally, the last sub-figure shows the impact, also visible in the Figure 50, of the FDIA launched from the Attacker federate. It also shows the impact of the Replay attack.

5.3.4 Comparison of Results with Existing Co-simulation Testbeds

In this Section, we compare the capabilities of our co-simulation with those of other co-simulators, in particular the last two most advanced solutions GridAttackSim and SCADA SIM. We thus show that our project allows a more realistic and in-depth study of complex scenarios, while allowing a large customization and dataset generation.

The Figure 52 presents a comparison of the capabilities of the three solutions GridAttackSim, SCADASIM and ASGARDS-H considering the objectives of our initial project. Overall, our project offers complete solutions that meet all the objectives. We did not include performance metrics, since GridAttackSim is not published as an open-source project.

Our project first allows a more advanced configuration of the system, making it more realistic. As GridAttackSim does not provide a configuration method for co-simulation scenarios, it is difficult for us, without having access to the source code, to measure the

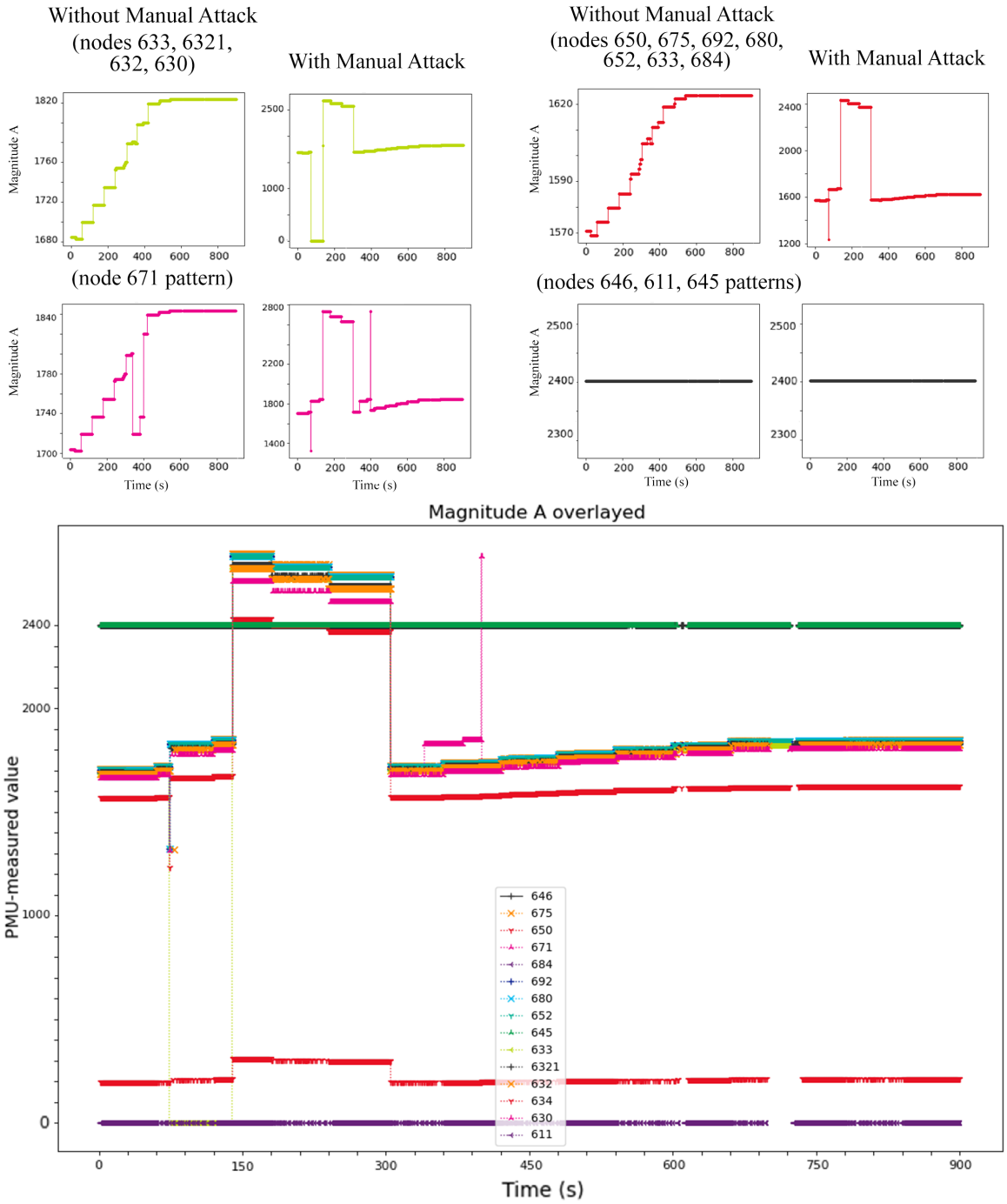


Figure 50: Use case: Power grid states comparison with and without FDIA

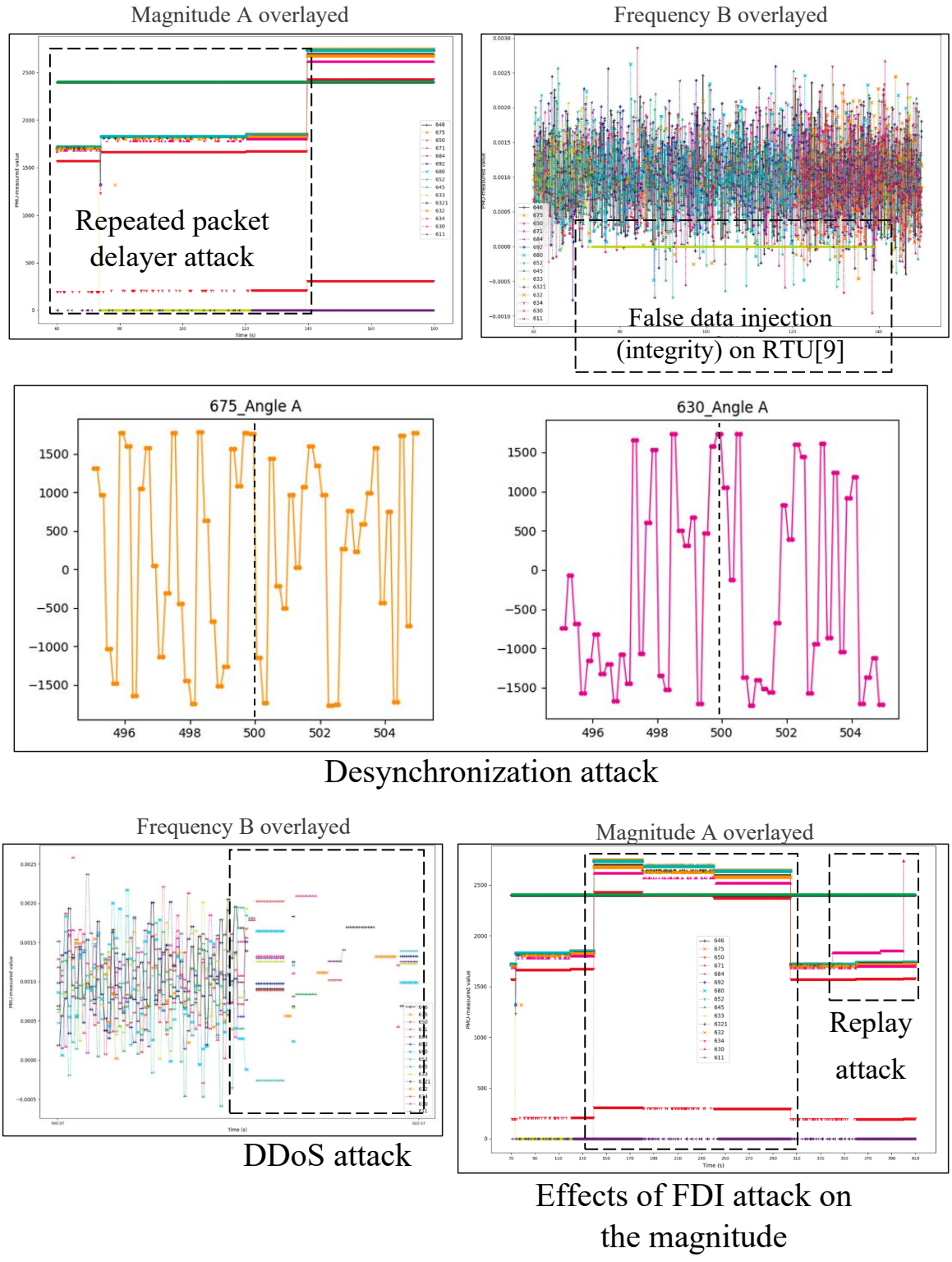


Figure 51: Use case: Attacks impacts on the monitored Power grid state

level of customization of the project. Moreover, despite the generation of some metrics from GridLab-D in the case of GridAttackSim or the statistics generated by OMNeT++ by default in the case of SCADASIM, we propose a complete support of the generation of co-simulation data from each federate, with the implementation of serializers and statistics via the *statistics hook*. Finally, the support of advanced options for co-simulation realism in SCADASIM and GridAttackSim is limited, as illustrated with the absence of the built-in Machine Learning integration feature.

Finally, our platform proposes itself as a credible alternative for the co-simulation of Smart Grids with the future implementation of 5G networks, see [29].

	GridAttackSim (2019)	SCADASIM (2019)	ASGARDS-H (2020)
SG capabilities	Demand Response, Dynamic Pricing	Modbus, SCADA System	Passive Demand-Response, Synchrophasors, Dynamic Pricing, Standard Device communication capabilities
Dataset generation	No, but generates data that are used to draw curves at the end of the simulation	Not implemented, but OMNeT++ can be extended to output PCAP files	Yes (*.pcap, *.csv); Serializers; Statistics
Advanced UI	Limited. Does allow for co-simulation configuration but is limited to application, attack category and attack type (single) parameters	No. Users have access to the OMNeT++ IDE	Yes (Project Generator, Visualizer)
Completness of co-simulation	Basic network simulator models, with UDP, Carrier Sense Multiple Access and one data aggregator; no details about real implemented protocols	Good completeness with a real Modbus protocol integration; advanced attack scenarios	Wired, Wireless modes; realistic generated topology, full exploitation of communication parameters from OMNeT++; advanced control of co-simulation via the Visualizer, Attacker, and Controllers federates
Quick integration of new power grid models	Not implemented, Three models are proposed, but no mechanism or documentation regarding the integration of new ones	Not implemented	Yes (Project Generator module)
Multiple attacks studies	The project allows for multiple CPA studies (one by one) but the authors did not claim the implementation of multiple attacks or advanced configuration	Yes, but not dynamically generated	Yes (Attacks sequences at the OSI Link layer with advanced configuration)
Fault capability	Not implemented	Yes	Yes (Project Generator, GridLab-D)
Quick ML integration capability	Not considered	Not considered	Yes (Controllers)

Figure 52: Comparison of co-simulation testbed capabilities against our initial objectives

Chapter 6

Discussions and Future Works

In this Section, we discuss the limitations of our project as well as future improvements.

We developed this project with scalability in mind, that is, at any point in the development process, to enable and document methods to easily extend the contributions made by ASGARDS-H. As a result, we have focused a lot of effort on the software architecture and user documentation of the project, while ensuring the initial objectives. However, many of the Smart Grids protocols and associated applications are not yet implemented. In the future, this project can be extended in the following ways:

- **Protocols**: OMNeT++ implements the *msg* format that can be used to integrate existing protocols by specifying the exchanges message formats. These messages are associated with Data Serializers and Deserializers. It is then possible to integrate new protocols either by reimplementing the communication logic within OMNeT++, or by directly integrating the source code of these protocols, as SCADASIM has done with the Modbus protocol.

- **Control Center**: The Control Center implements individual applications (Demand-Response, Dynamic Pricing, Synchrophasor). In the case of Demand-Response, the current implementation of the OpenADR protocol is complete but the Control Center does not take any action on the power grid to adapt consumption in real time. This capability can be implemented via the Project Generator and may allow to extend the capabilities of the Control Center. Furthermore, the CC does not implement SCADA systems, since these systems can be implemented directly within the different controllers.

- **Network topology**: Both types of topologies are generated following a logical and unchanging scheme depending on the electrical model considered. The topology for the same model can vary according to the placement of nodes and user parameters, but the use of other types of topologies is not directly possible. For this reason, we propose a simple method to extend the list of topologies and to implement new logics, with no limit in number.

- **5G**: The simu5G project is currently being developed as a successor to simuLTE and will be made public in the year 2020. The integration of this project, proposed as an extension of OMNeT++, can be simply integrated within our platform, allowing the co-simulation of smart grids within an end-to-end 5G context.

- **Electrical vehicles, software-defined networks**: The advantage offered by OMNeT++ today lies in its modularity and the large number of modules that allow its functionalities to be extended. We suggest the use and integration of third-party modules for the study of specific scenarios.

- **SG residential applications**: The level of detail in OMNeT++ allows to set up large scenarios where many protocols interact. For example, ASGARS-H can be extended to include advanced scenarios such as Smart Home, Smart EV charging, and more.

- **Controllers, BDD, SE**: We proposed a customizable implementation of the controllers in Python to allow the user to implement his own logic. It will then become interesting to propose a modular implementation of systems such as the Bad Data Detector or the State Estimator combined with specific actions of the controller to allow a more advanced co-simulation and the study of attacks such as the FDIA.

In order to guarantee support over the project time frame, our documentation aims to explain all aspects of the project, both theoretical and practical, in order to minimize the steep learning curve that ASGARDS-H represents.

A second area of improvement is the visualizer. Our implementation allows a real-time display of both systems, with some controls related to co-simulation. More specifically, we allow the addition of functionalities for the control of these networks, and this improvement would go hand in hand with the addition of an advanced system such as SCADA or BDD/SE.

Finally, we suggest that this project, at least its architecture, serve as a basis for the development of a complete and advanced software for the co-simulation of smart grids and 5G. We believe that this modular approach allows (1) an easier contribution to the research community, (2) the development of a solution of interest to both industry and academia, and (3) a more advanced implementation than the similar projects we have studied in this thesis.

Chapter 7

Conclusion

In this thesis, multiple aspects of co-simulation, smart grids and cyber-physical attacks were studied. Today, co-simulation is proposed as an advanced method for the systematic and safe study of complex cyber-physical systems. The objectives are multiple: to train personnel, to study or even discover new attacks, to develop solutions for education, to develop new techniques for detecting and mitigating attacks, etc. Numerous projects have been proposed, either by academia or industry, or often through collaborations between the two, and each of them has made its own contributions to address the many issues facing smart grids today. However, these projects are struggling to implement mechanisms that are sufficiently complete or modular to allow the realistic study of cyber-physical systems. In addition, cyber-physical attacks are poorly implemented. I believe that the electrical, network and physical cyber-security communities must work together more than ever, with the implementation of complete training courses using this type of co-simulation tools in order to train engineers, researchers, students or technicians to develop their ability to understand the issues, the risks and to act to develop even more complete and secure systems in the future. This thesis objective was to implement a co-simulation testbed for advanced cyber physical attack studies and dataset generation in the context of the Smart Grid, with Situational Awareness consideration, realism and modularity. This testbed must be easily usable by the other users for the rest of the project and must allow the quick integration or use of machine learning techniques. The recent integration of these machine learning techniques for the detection and mitigation of CPA requires new tools for the study and generation of realistic data. I believe that I have provided a complete solution that will be used in the future for advanced scenario study and the development of innovative ML

techniques that can work in the real world. I do not consider ASGARDS-H as the most complete solution today compared to proprietary solutions such as OpalRT, but I believe that ASGARDS-H brings new ways to implement smart grids co-simulations, and will prove its usefulness in future co-simulations studies. I first implemented a co-simulation platform integrating HELICS, OMNeT++ and GridLab-D. This platform then acquired a modular dimension when it was included in a larger project, named ASGARDS-H, allowing advanced generation of co-simulation testbeds, dataset generation and advanced study of multifaceted scenarios (power grid faults, manual cyber attacks, manual physical attacks, configurable cyber-physical attacks, weather, network events). The use of the project has shown a wide variety of co-simulation possibilities, and the validation of each aspect has allowed us to study complete scenarios and generate our first datasets. We were then able to compare our project to similar projects such as GridAttackSim and SCADASIM, and showed the strengths as well as the limitations of ASGARDS-H. For the future, I hope that this platform will be able to meet the expectations of machine learning integration. Also, I wish to see this project fully exploited: many capabilities are frequently added within the integrated simulators, which leaves the user the possibility to extend this project as he wishes. ASGARS-H is now specifically developed for smart grids, but will offer an interesting architecture for other systems. This project will be used and improved as part of the Ericsson GAIA program in Montreal.

Bibliography

[1] V. Y. Pillitteri and T. L. Brewer, "Guidelines for smart grid cybersecurity," Tech. Rep., 2014.

[2] J. McCarthy, O. Alexander, S. Edwards, D. Faatz, C. Peloquin, S. Symington, A. Thibault, J. Wiltberger, and K. Viani, "Situational awareness," *NIST Special Publication*, p. 7B, 1800.

[3] H. He and J. Yan, "Cyber-physical attacks and defences in the smart grid: a survey," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 13–27, 2016.

[4] I. Ghansah, *Smart Grid Cyber Security Potential Threats, Vulnerabilities and Risks: Interim Project Report*. California Energy Commission, 2012.

[5] I. Cert, "Incident response/vulnerability coordination in 2014."

[6] T. M. Chen and S. Abu-Nimeh, "Lessons from stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, 2011.

[7] J. T. Langill, "Defending against the dragonfly cyber security attacks," *Retrieved*, vol. 11, p. 2015, 2014.

[8] T. FoxBrewster, "Ukraine claims hackers caused christmas power outage," *Forbes Security*, 2016.

[9] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.

[10] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: a survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–33, 2018.

[11] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauss, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold *et al.*, "The functional mockup interface for tool independent exchange of simulation models," in *Proceedings of the 8th International Modelica Conference*. Linköping University Press, 2011, pp. 105–114.

[12] R. Kübler and W. Schiehlen, "Two methods of simulator coupling," *Mathematical and computer modelling of dynamical systems*, vol. 6, no. 2, pp. 93–113, 2000.

[13] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical computer science*, vol. 138, no. 1, pp. 3–34, 1995.

[14] E. K. Wang, Y. Ye, X. Xu, S.-M. Yiu, L. C. K. Hui, and K.-P. Chow, "Security issues and challenges for cyber physical system," in *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*. IEEE, 2010, pp. 733–738.

[15] E. Barker, "Guideline for using cryptographic standards in the federal government: Cryptographic mechanisms," National Institute of Standards and Technology, Tech. Rep., 2016.

[16] M. S. Turan, K. A. McKay, Ç. Çalık, D. Chang, and L. Bassham, "Status report on the first round of the nist lightweight cryptography standardization process," *National Institute of Standards and Technology, Gaithersburg, MD, NIST Interagency/Internal Rep.(NISTIR)*, 2019.

[17] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education Upper Saddle River, NJ, USA, 2012.

[18] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," in *Annales des télécommunications*, vol. 55, no. 7-8. Springer, 2000, pp. 361–378.

[19] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 1–33, 2011.

[20] "Llnl/griddyn," https://github.com/LLNL/GridDyn, (Accessed on 08/07/2020).

[21] Ames-Market, "ames-market/psst," https://github.com/ames-market/psst, (Accessed on 08/07/2020).

[22] R. Lincoln, "Github repository for rwl/pypower," *GitHub, November*, 2012.

[23] D. Montenegro, M. Hernandez, and G. Ramos, "Real time opendss framework for distribution systems simulation and analysis," in *2012 Sixth IEEE/PES Transmission and Distribution: Latin America Conference and Exposition (T&D-LA)*. IEEE, 2012, pp. 1–5.

[24] T. D. Le, A. Anwar, R. Beuran, and S. W. Loke, "Smart grid co-simulation tools: Review and cybersecurity case study," in *2019 7th International Conference on Smart Grid (icSmartGrid)*. IEEE, 2019, pp. 39–45.

[25] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "Gridlab-d: An open-source power systems modeling and simulation environment," in *2008 IEEE/PES Transmission and Distribution Conference and Exposition*. IEEE, 2008, pp. 1–5.

[26] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[27] O. Ltd., "Omnet simulator," https://omnetpp.org/, (Accessed on 08/07/2020).

[28] A. Virdis, G. Stea, and G. Nardini, "Simulating lte/lte-advanced networks with simulte," in *Simulation and Modeling Methodologies, Technologies and Applications*. Springer, 2015, pp. 83–105.

[29] G. NARDINI, G. STEA, A. VIRDIS, and D. SABELLA, "Simu5g: a system-level simulator for 5g networks," in *SIMULTECH 2020*. INSTICC, 2020.

[30] "Inet framework," https://inet.omnetpp.org/, (Accessed on 08/07/2020).

[31] D. Klein and M. Jarschel, "An openflow extension for the omnet++ inet framework," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, 2013, pp. 322–329.

[32] S. C. Müller, H. Georg, J. J. Nutaro, E. Widl, Y. Deng, P. Palensky, M. U. Awais, M. Chenine, M. Küch, M. Stifter *et al.*, "Interfacing power system and ict simulators:

Challenges, state-of-the-art, and case studies," *IEEE Transactions on Smart Grid*, vol. 9, no. 1, pp. 14–24, 2016.

[33] M. Vogt, F. Marten, and M. Braun, "A survey and statistical analysis of smart grid co-simulations," *Applied Energy*, vol. 222, pp. 67–78, 2018.

[34] W. Lardier, Q. Varo, and J. Yan, "Quantum-sim: An open-source co-simulation platform for quantum key distribution-based smart grid communications," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2019, pp. 1–6.

[35] K. Hopkinson, X. Wang, R. Giovanini, J. Thorp, K. Birman, and D. Coury, "Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 548–558, 2006.

[36] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri, "A testbed for analyzing security of scada control systems (tasscs)," in *ISGT 2011*. IEEE, 2011, pp. 1–7.

[37] V. Liberatore and A. Al-Hammouri, "Smart grid communication and co-simulation," in *IEEE 2011 EnergyTech*. IEEE, 2011, pp. 1–5.

[38] H. Lin, S. S. Veda, S. S. Shukla, L. Mili, and J. Thorp, "Geco: Global event-driven co-simulation framework for interconnected power system and communication network," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1444–1456, 2012.

[39] H. Georg, S. C. Müller, N. Dorsch, C. Rehtanz, and C. Wietfeld, "Inspire: Integrated co-simulation of power and ict systems for real-time evaluation," in *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2013, pp. 576–581.

[40] K. Anderson, J. Du, A. Narayan, and A. El Gamal, "Gridspice: A distributed simulation platform for the smart grid," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2354–2363, 2014.

[41] J. É. Gómez, J. J. H. Cabrera, J.-P. Tavella, S. Vialle, E. Kremers, and L. Frayssinet, "Daccosim ng: co-simulation made simpler and faster," 2019.

[42] A. G. Wermann, M. C. Bortolozzo, E. G. da Silva, A. Schaeffer-Filho, L. P. Gaspary, and M. Barcellos, "Astoria: A framework for attack simulation and evaluation in smart grids," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 273–280.

[43] B. Camus, V. Galtier, and M. Caujolle, "Hybrid co-simulation of fmus using dev&dess in mecsyco," in *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*. IEEE, 2016, pp. 1–8.

[44] C. Queiroz, A. Mahmood, and Z. Tari, "Scadasim—a framework for building scada simulations," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 589–597, 2011.

[45] T. I. Strasser, E. C. de Jong, and M. Sosnina, "European guide to power system testing: The erigrid holistic approach for evaluating complex smart grid configurations," 2020.

[46] T. D. Le, A. Anwar, S. W. Loke, R. Beuran, and Y. Tan, "Gridattacksim: A cyber attack simulation framework for smart grids," *Electronics*, vol. 9, no. 8, p. 1218, 2020.

[47] S. Ciraci, J. Daily, J. Fuller, A. Fisher, L. Marinovici, and K. Agarwal, "Fncs: a framework for power system and communication networks co-simulation," in *Proceedings of the symposium on theory of modeling & simulation-DEVS integrative*, 2014, pp. 1–8.

[48] B. Palmintier, D. Krishnamurthy, P. Top, S. Smith, J. Daily, and J. Fuller, "Design of the helics high-performance transmission-distribution-communication-market co-simulation framework," in *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, 2017, pp. 1–6.

[49] S. Schütte, S. Scherfke, and M. Tröschel, "Mosaik: A framework for modular simulation of active components in smart grids," in *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*. IEEE, 2011, pp. 55–60.

[50] R. Bhandia, A. van der Meer, E. Widl, T. I. Strasser, K. Heussen, T. V. Jensen, C. Steinbrink, V. H. Nguyen, F. Bourry, M. Syed *et al.*, "D-jra2. 3 smart grid simulation environment," 2019.

[51] S. Smartgrids, "2035," *Summary of Priorities for Smartgrid research topics. V19 June*, 2013.

[52] "Hardy_helics_co-simulation_volttron_users_meeting.pdf," https://bgintegration.pnnl.gov/pdf/Hardy_HELICS_Co-simulation_VOLTTRON_Users_Meeting.pdf, (Accessed on 08/07/2020).

[53] J. de Chalendar, "jdechalendar/glm-plotter," https://github.com/jdechalendar/glm-plotter, (Accessed on 08/07/2020).

[54] A. Virdis, G. Stea, and G. Nardini, "Simulating lte/lte-advanced networks with simulte," in *Simulation and Modeling Methodologies, Technologies and Applications*. Springer, 2015, pp. 83–105.

[55] T. Kamada, S. Kawai *et al.*, "An algorithm for drawing general undirected graphs," *Information processing letters*, vol. 31, no. 1, pp. 7–15, 1989.

[56] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software," *PloS one*, vol. 9, no. 6, p. e98679, 2014.

[57] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, 2006.

Appendices

Appendix A

Natively implemented power grid model files

Object class name	Recorder properties	
IEEE 13 Bus System v1	**Topology**	15 nodes, 9 overhead lines, 2 underground lines
	Houses	16 houses equipped with ZIP load, Water heaters and HVAC systems
	Power solver method	Newton-Raphson (NR)
	Weather file	WA-Yakima
	Model spcifications	13-bus feeder (from PES, Power system Analysis, Computing and Economics Committee)
	Additional parameters	DELTAMODE-compatible
IEEE 13 bus system v2	**Added capabilities**	Dynamic price market controlled by an Auction object, HVAC controllers
DistributionSim B2 G 1	**Nodes**	31 nodes, 33 overhead lines, 18 underground lines

Table 7 continued from previous page

Object class name	**Recorder properties**	
	Houses	497 houses with Water heater, lighting, occupancy and plugs and HVAC controllers
	Power solver method	Newton-Raphson (NR)
	Weather file	AZ-Phoenix
	Model specifications	FY11 Department of Energy taxonomy feeder test case (2016)
	Additional parameters	DELTAMODE-compatible
DistributionSim B2 G 1v2	**Added capabilities**	Fixed HVAC loads and Dynamic Price Market validation
Two Communities	**Nodes**	21 nodes, 30 triplex nodes, 47 overhead lines, 11 underground lines
	Houses	30 houses with HVAC, Water heater and HVAC controllers
	Power solver method	Newton-Raphson (NR)
	Weather file	Wa-Yakima
	Model specifications	Two houses community model file from the *usnistgov* GitHub repository
	Additional parameters	DELTAMODE-compatible
Two Communities v2	**Added capabilities**	Fixed HVAC loads and Dynamic Price Market validation
small testing	**Topology**	2 nodes, 1 overhead line
	Houses	1 house with HVAC and Water heater
	Power solver method	Newton-Raphson (NR)
	Weather file	N/A
	Model specifications	Custom model file for quick validation or long time co-simulation studies

Table 7 continued from previous page

Object class name	**Recorder properties**	
	Additional parameters	DELTAMODE-compatible

Appendix B

Project Generator parameters

- ***fileDialog.fileUrl*** (*stringURI*): The absolute path in the file system of the chosen GLM file. If the latter has additional files (weather, schedules), all the files must be placed in the *GLM* folder of the main folder of the project generator.
- ***seed.value*** (*integer*): Used for reproducibility of results. This seed is used when the user wants to study its system in a more random way.
- ***threads.value*** (*integer*): For compatible GLM models, one can speed-up the co-simulation speed by specifying the number of worker threads to be more than one.
- ***fileDialogTMY3.fileUrl*** (*stringURI*): If the user needs to simulate a specific weather, he or she can specify the file to use here. Note that the file must be place under the GLM folder.
- ***outputRecorder.checked*** (*boolean*): By enabling this parameter, user enables the recorder module in the GLM file to export CSV file(s) of the considered simulation.
- ***gLatMean.value*** (*double*): The global OMNeT++ communications fixed latency. This parameter is proposed as a scaling parameter for the study of more unstable systems.
- ***commDataRate1.value & commDataRate.currentIndex*** (*double*): The speed of communication is expressed here in $bps, kbps, Mbps, Tbps$.
- ***connType.currentIndex*** (*list*): Let the user choose between *Wired* and *Wireless* modes.

- ***useGUI.checked*** (*boolean*) is an option to enable or disable the OMNeT++ GUI during the simulation. If this checkbox is unchecked, a CLI version of the OMNeT++ federate will be launched.

- ***useViz.checked*** (*boolean*) is an option to enable the Visualizer federate. For long-time studies, the visualizer federate may slow-down the simulator speed.

- ***wireshark.checked*** (*boolean*): This parameter will enable or disable the generation of PCAP file from the server.

- ***ueDist.value*** (*boolean*): an option to change the generated distance radius of RTUs and smart meters from their Antenna.

- ***pMULatency.value*** (*double*): Represents the delay at the beginning of the simulation before the RTUs starts to send their data.

- ***pDCLatency.value*** (*double*): The PDC latency represents the delay at the beginning of the simulation before the concentrator starts to aggregate data.

- ***pMUFrequency.value & pDCFrequency.value*** (*double*): Represent the frequency of messages sent by both PMUs to the PDCs or server and the PDCs to the server. The default value is representative of a typical SCADA system.

- ***pDCWaitTime.value*** (*double*): The waiting duration or time window to aggregate PMUs/RTUs messages before sending them to the Control Center (server).

- ***errorMeasurementMean.value & errorMeasurementStd.value & errorFMeasurementMean.value & errorFMeasurementStd.value*** (*double*): Frequency-specific and Phase Angle-specific added noise (mean and standard deviation) using the normal distribution.

- ***ccURI.text*** (*string*): URL used by the meters using the OpenADR protocol to connect to the Control Center. This will impact PCAP output files.

- ***openADRStartDelay.value*** (*double*): Start delay of the OpenADR protocol. This parameter is shared among aggregators, Control Center and Smart Meters.

- ***openADRaggDuration.value*** (*double*): OpenADR aggregators aggregation time window.

- ***openADRsendingAggfrequency.value*** (*double*): Frequency of OpenADR aggregated sending of messages.
- ***openADRsendingMeterfrequency.value*** (*double*): OpenADR smart meter sending frequency.
- ***openADRsendingMeteringMean.value*** (*double*): relative mean noise added to the measurements.
- ***openADRsendingMeteringStd.value*** (*double*): relative standard deviation noise added to the measurements.
- ***adrNoisy.checked*** (*boolean*): Enables or disables the noisy measurements for the OpenADR protocol.
- ***packetDrop.checked*** (*boolean*): Activates the global packet dropping capability using the *globalDROPProbability* parameter. Used to simulate unstable networks.
- ***globalDROPProbability.value*** (*double*): Global probability to drop a message during the co-simulation.
- ***firewall.text*** (*string*): The firewall configuration uses iptables-like grammar to configure the firewall. Available filters are:

  - **-A** to append one or more rules to the end of the rule-chain.

  - **-P** to set the policy for the built-in (non user-defined) chain to the given target.

  - **FORWARD|INPUT|OUTPUT** are filters: INPUT (for packets destined to local sockets), FORWARD (for packets being routed through the box), and OUTPUT (for locally generated packets).

  - **-p** (protocol) is a filter to select specific protocols, such as SYNCHROPHASOR, OpenADR, TCP, UDP, IP, etc.

  - **-d** (ip/mask) is a filter to select specific range of ipv4 address (or specific ipv4 address).

  - **-j** (rule) is mandatory and represents the rule for all packets that are considered with the previous filters used.

- ***profiles*** (*matrixofboolean*): A 3*xN* matrix of all available co-simulation publications from all federates for each profile (Operator, Defender and Attacker). Affects the PCAP generation and the Visualizer federate.

- ***datasetLimit.value*** (*integer*): Defines the GridLab-D recording limit time.

- ***datasetInterval.value*** (*double*): Defines the GridLab-D recording interval.

- ***datasetStatistics.checked*** (*boolean*): Enables the recording of the co-simulation statistic related datasets.

- ***outputFaults.checked*** (*boolean*): By enabling this option, the project generator will add a *reliability* object to the GLM file to output interesting metrics from an end-user point of view.

- ***metricsOfInterest.text*** (*string*) are indicators that represents indices used in the reliability analysis of a distribution system.

- ***customerGroup.text*** (*string*) is a filter for selecting a group of users with conditions.

- ***maxEventLength.value & metricInterval.value & reportInterval.value*** (*double*) are straightforward timing configuration for the metrics.

- ***listOfEvents*** (*array*): A list of user-configurable formatted GridLab-D events. This array is dynamically generated.

  - Event type : Auto or Manual. When using Auto mode, the event will follow a random distribution and then can occur zero, one or multiple times.

  - Fault type : GridLab-D provides a lot of possible events with the *eventgen* module. Check the documentation for more details.

  - Filter (auto mode): GridLab-D filter used in Auto mode to determine the targeted module(s).

  - max_outage_length : Maximum duration in seconds of an event.

  - max_simultaneous : The maximum number of events that can occur at the same time. Set -1 for infinity.

- failure_dist : The random distribution that the event follows. E.g. following an exponential distribution will involve more events at the beginning of the simulation and less at the end of the simulation.

- Manual event configuration (manual mode): user editable text to specify the starting and ending time of the event.

- restoration_dist : The random distribution followed by the reliability module for the restoration after the event.

- ***listOfAttacks*** (*array*): A list of user-defined attacks or hooks. This array is dynamically generated. Available attacks include *PacketDropping*, *Replayattack*, *Packetdelayer*, *DoS/DDoS*, *Integrityattack*, *Eavesdropping*, *Desynchronization*, *Dynamicpricing*, *Statistics*.

- ***popup.listOfSpecificConfiguration*** (*array*): Users can, in addition to global steady-state parameters, configure one by one all OMNeT++ devices and apply specific OpenADR, Synchrophasor, Aggregation or Communication options.

- ***displayLogs.checked*** (*boolean*): A UI-specific parameter to manage the outpout console.

- ***simStartTime.text*** (*timestamp*): The starting date of the co-simulation.

- ***simEnd.text*** (*timestamp*): The co-simulation end time. The total co-simulation duration is dynamically computed from the start and end timestamps.

- ***timeStep.value*** (*double*): The GridLab-D simulation timestep, if advanced studies are needed.

- ***cotimeStep.value*** (*double*): The co-simulation timestep. This represents the minimum valid timestamp for simulator synchronizations. For example, GridLab-D and OMNeT++ will use this co-simulation value, while the Market federate will synchronizes less frequently, according to the market duration.

- ***simSpeed.value*** (*double*): The default simulation speed, e.g. if the scenario requires a real-time co-simulation.

- ***output.checked*** (*boolean*): Enables dataset capability for the testbed.

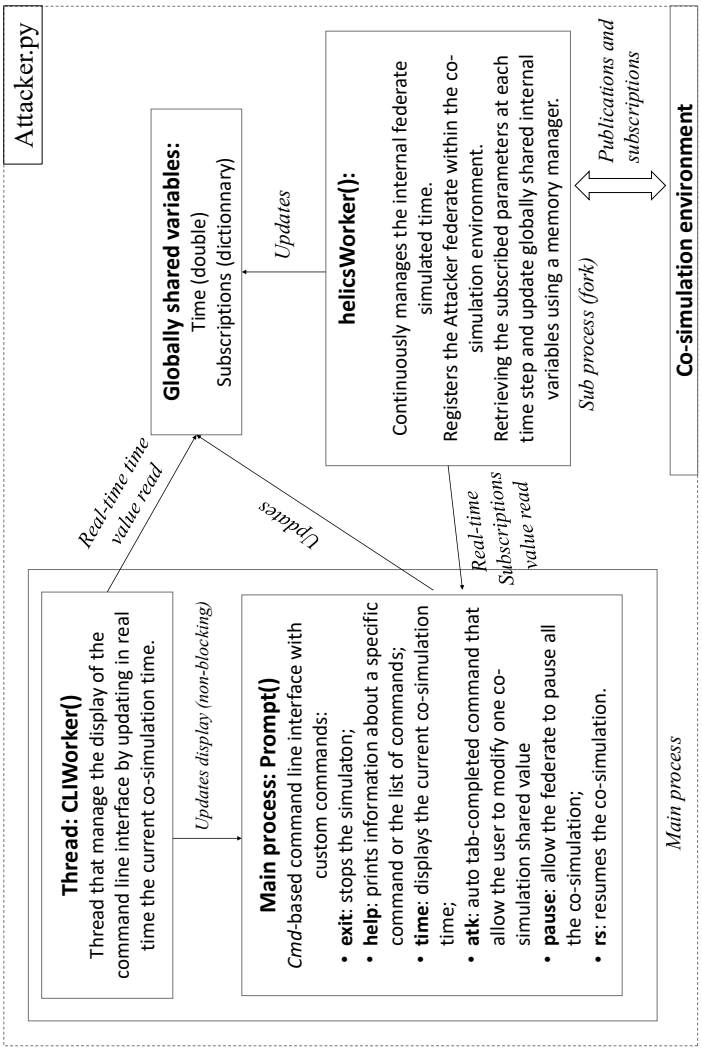
- ***attacks.checked*** (*boolean*): Enables the Attacker Federate.

After generating the project, the user can manually configure the testbed with the following parameters:

- ***Controller federates code*** (Market and Power Grid), in order to change their behavior, add advanced features (State Estimation, Bad Data Detector, IDS), specific attacks (Malicious code).
- ***Customize the OMNeT++ event file*** via the lifecycle configuration file to simulate equipment faults within OMNeT++, disconnections etc.
- ***Add player files*** within the GridLab-D model to manually alter some aspects of the co-simulation and activate the *subsecond* mode.

Appendix C

Attacker Federate Architecture



Appendix D

Dataset default columns

- **Frame Number**: The message frame number;
- **Link Layer Protocol**: *PPP*, *eth*, etc.;
- **Epoch**: The relative time the message was received or sent;
- **Date**: A complete timestamp according to the co-simulation dates;
- **Frame Length**: The packet size;
- **Network layer Protocol**: Either TCP or UDP;
- **IPv4 Source Address**;
- **IPv4 Destination Address**;
- **Time To Live**;
- **Source Port**;
- **Destination Port**;
- **Flags**: TCP or UDP flags of the message;
- **Application Layer protocol**: HTTP, Synchrophasor;
- **RAW Data**: AN hexadecimal view of the raw message data;
- **Synchrophasor/OpenADR extracted Measurements/Text**: *N* columns that contains all extracted measurement values from known monitoring packets.

Appendix E

Built-in ASGARDS-H GridLab-D recorders

Object class name	Recorder properties
load	load_class, measured_voltage_AB.real, measured_voltage_BC.real, measured_voltage_CA.real, constant_power_A.real, constant_power_B.real, constant_power_C.real, constant_current_A.real, constant_current_B.real, constant_current_C.real, constant_impedance_A.real, constant_impedance_B.real, constant_impedance_C.real, measured_power_A.real, measured_power_B.real, measured_power_C.real, measured_voltage_A.real, measured_voltage_B.real, measured_voltage_C.real, measured_voltage_AB.imag, measured_voltage_BC.imag, measured_voltage_CA.imag, constant_power_A.imag, constant_power_B.imag, constant_power_C.imag, constant_current_A.imag, constant_current_B.imag, constant_current_C.imag, constant_impedance_A.imag, constant_impedance_B.imag, constant_impedance_C.imag, measured_power_A.imag, measured_power_B.imag, measured_power_C.imag, measured_voltage_A.imag, measured_voltage_B.imag, measured_voltage_C.imag
house	air_temperature, outdoor_temperature, thermostat_deadband, heating_demand, cooling_demand, total_load, hvac_load

Table 8 continued from previous page

Object class name	Recorder properties
meter	measured_voltage_A, measured_voltage_B, measured_voltage_C, measured_current_A, measured_current_B, measured_current_C, measured_power_A, measured_power_B, measured_power_C, measured_real_energy, measured_reactive_energy, measured_power, measured_demand, measured_real_power, measured_reactive_power, bill_day, price, monthly_fee, monthly_bill, previous_monthly_bill, monthly_energy, previous_monthly_energy, bill_mode, first_tier_price, second_tier_price, third_tier_price, first_tier_energy, second_tier_energy, third_tier_energy
link	voltage_A.real, voltage_B.real, voltage_C.real, voltage_AB.real, voltage_BC.real, voltage_CA.real, current_A.real, current_B.real, current_C.real, measured_angle_A, measured_frequency_A, measured_angle_B, measured_frequency_B, measured_angle_C, measured_frequency_C, measured_frequency, power_A.real, power_B.real, power_C.real, shunt_A.real, shunt_B.real, shunt_C.real, mean_repair_time, bustype, maximum_voltage_error, busflags, voltage_A.imag, voltage_B.imag, voltage_C.imag, voltage_AB.imag, voltage_BC.imag, voltage_CA.imag, current_A.imag, current_B.imag, current_C.imag, power_A.imag, power_B.imag, power_C.imag, shunt_A.imag, shunt_B.imag, shunt_C.imag
fuse	status, mean_repair_time
triplex_meter	mean_repair_time
triplex_node	measured_real_energy, measured_reactive_energy, measured_power.real, measured_demand, measured_real_power, measured_reactive_power, indiv_measured_power_1.real, indiv_measured_power_2.real, indiv_measured_power_N.real, measured_current_1.real, measured_current_2.real, measured_current_N.real, measured_voltage_1.real, measured_voltage_2.real, measured_voltage_N.real, measured_power.imag, indiv_measured_power_1.imag, indiv_measured_power_2.imag, indiv_measured_power_N.imag, measured_current_1.imag, measured_current_2.imag, measured_current_N.imag, measured_voltage_1.imag, measured_voltage_2.imag, measured_voltage_N.imag, bill_day, price, monthly_fee, monthly_bill, previous_monthly_bill, monthly_energy, previous_monthly_energy, bill_mode, first_tier_price, second_tier_price, third_tier_price, first_tier_energy, second_tier_energy, third_tier_energy

Table 8 continued from previous page

Object class name	Recorder properties
switch	voltage_1.real, voltage_2.real, voltage_N.real, voltage_12.real, voltage_1N.real, voltage_2N.real, bustype, maximum_voltage_error, busflags, voltage_1.imag, voltage_2.imag, voltage_N.imag, voltage_12.imag, voltage_1N.imag, voltage_2N.imag
triplex_load	status
node	measured_voltage_1.real, measured_voltage_2.real, measured_voltage_N.real, measured_voltage_1.imag, measured_voltage_2.imag, measured_voltage_N.imag

Appendix F

Testbed use case parameters

Model description	IEEE 13 Bus System with Voltage players and residential module.
Hooks/Attacks statistics	12 hooks
Co-simulation timestep	100ms
Start / End time	From 2020-01-01 12:00:00:000 to 2020-01-01 12:15:00:000
Dataset generation	Yes (PCAP, Statistics, GridLab-D)
Communications	10Mbps Wireless (4G LTE)
Attack 1	**Eavesdropping**. From $t = 60s$ to $t = 180s$, targets all routers incoming/outgoing packets.
Attack 2	**Packet delayer**. From $t = 60s$ for $60s$ (repeat mode every $60s$). Implements a fixed delay using the uniform random distribution (between $0s$ and $0.25s$). Targets all PMUs with the filter $router[1*$.
Attack 3	**Packet delayer**. Targets the firewall outgoing packets. Starts at $t = 300s$, lasts $5s$ with a $5.01s$ fixed delay.
Attack 4	**Integrity**. targets all RTUs in live mode.
Attack 5	**DDoS**. Attacker targets (1) $router[13]$, (2) $router[8]$ and (3) $router[3]$. (1) impersonate $router[6]$ and attacks the server with a $10\mu s$ delay between two packet sending. (2) and (3) impersonate $router[11]$ and $RTU[8]$ and attacks the server with a $07s$ delay between two packet sending. Attack starts at $t = 600s$ and lasts $5s$.
Hook 6	**Statistics**. From the Firewall point of view.
Attack 7	**Desynchronization attack**. Targets all PMUs with the filter $RTU[2*$. Attack starts at $t = 500s$ with a fixed $200ms$ desychronization and lasts $150s$.
Attack 8	**Replay attack**. Targets RTU[5] outgoing packets and starts at $t = 80s$, recording $60s$ and replaying after $200s$.
Attack 9	**Packet dropping**. Targets the firewall, live attack.
Lifecycle	**router[2]** crashes at $t = 50s$ and restarts at $t = 60s$.
Specific configuration	**RTU[0]** sends its messages every $200ms$.

Appendix G

Generated datasets statistics

Device name	Dataset number of packets and size
IEEE 13 Nodes	67,995 packets. 67,450,386-bytes long (67*MB*).
atk_router[0]_hppp	42,493 packets. 44,997,827-bytes long (45*MB*).
atk_router[1]_hppp	11,791 packets. 5,021,487-bytes long (5*MB*).
atk_router[2]_hppp	338,715 packets. 203,376,893-bytes long (203*MB*).
atk_router[3]_hppp	59,391 packets. 40,535,689-bytes long (41*MB*).
atk_router[4]_hppp	17,173 packets. 6,578,495-bytes long (7*MB*).
atk_router[5]_hppp	17,361 packets. 6,687,105-bytes long (7*MB*).
atk_router[6]_hppp	17,173 packets. 6,578,495-bytes long (7*MB*).
atk_router[7]_hppp	17,193 packets. 6,584,667-bytes long (7*MB*).
atk_router[8]_hppp	34,193 packets. 32,159,699-bytes long (32*MB*).
atk_router[9]_hppp	17,173 packets. 46,578,495-bytes long (7*MB*).
atk_router[10]_hppp	11,843 packets. 5,010,895-bytes long (5*MB*).
atk_router[11]_hppp	23,123 packets. 8,770,021-bytes long (9*MB*).
atk_router[12]_hppp	28,807 packets. 30,524,753-bytes long (31*MB*).
atk_router[13]_hppp	331,762 packets. 206,571,669-bytes long (207*MB*).
atk_router[14]_hppp	11,811 packets. 5,001,185-bytes long (5*MB*).