Interconnecting Geographically Distributed Laboratories for Smart Grid Testing & Validation

Master Thesis

Vetrivel Subramaniam Rajkumar

Technische Universiteit Delft

Interconnecting Geographically Distributed Laboratories for Smart Grid Testing & Validation

Master Thesis

by

Vetrivel Subramaniam Rajkumar

In Partial Fulfillment of the Requirements for the Degree of

Master of Science
in Electrical Engineering

at the Delft University of Technology,
to be defended publicly on Friday July 5, 2019 at 12:30.

Acknowledgements

First and foremost, I would like to thank my *Gurus*, Prof. Dr. Peter Palensky and Asst. Prof. Dr. Miloš Cvetković, under the guidance of whom, this thesis project was carried out. Thank you for your continuous support, invaluable feedback and guidance during this study. I appreciate the time you dedicated in guiding me throughout this study. I have to admit, I have been extremely lucky to have you as my *gurus*. A special shout out to my PhD. Supervisor, Rishabh Bhandia for his guidance and support over the course of this thesis. Thank you for standing by me, through thick and thin.

I am also grateful to the IEPG group for the opportunity to learn and also to contribute to ongoing research. Next, I would like to thank Asst. Prof. Dr. Kai Heussen from Technical University of Denmark (DTU) for agreeing to be part of my thesis committee, and taking time to evaluate the my work. I also wish to express my sincere gratitude to all those who made this thesis possible. First, a special mention to our ERIGrid project colleagues, Petra Raussi from VTT and Oliver Gehrke from DTU for being a part of the joint experiments. Next, the VILLAS4ERIGrid User group, involving Steffen Vogel, Marija Stevic and Ha Nguyen. It was great collaborating with all of you! Furthermore, I would like to appreciate the timely administrative support of our section secretary, Ellen.

Finally, I would like to thank all my colleagues and friends who were a part of my stay in The Netherlands for the past two years. It has been a great learning experience, academically and personally for me, during my time in Delft. This thesis work has been carried out within the framework of the H2020 funded European Union Project, ERIGrid (https://erigrid.eu/) with Grant Agreement (GA) 65411.

Vetrivel Subramaniam Rajkumar
Delft, July 2019

Contents

List of Abbreviations

API	Application Programming Interface
CDF	Cumulative Distribution Function
CPES	Cyber-Physical Energy System
CVC	Coordinated Voltage Control
DER	Distributed Energy Resources
DFT	Discrete Fourier Transform
DP	Dynamic Phasors
EMT	Electromagnetic Transient
GDRTS	Geographically Distributed Real-Time Simulation
HIL	Hardware in Loop
HVDC	High Voltage Direct Current
IA	Interface Algorithm
ICT	Information and Communication Technology
JaNDER	Joint Test Facility for Smart Energy Networks with Distributed Energy Resources
JRA	Joint Research Activity
NREN	National Research and Education Networks
NTP	Network Time Protocol
PCC	Point of Common Coupling
QoS	Quality of Service
RES	Renewable Energy Sources
RI	Research Infrastructure
RTCP	Real-Time Transfer Control Protocol
RTDS	Real-Time Digital Simulator
RTP	Real-Time Protocol
RTT	Round Trip Time
SCADA	Supervisory Control and Data Acquisition
TCP	Transfer Control Protocol
UDP	User Datagram Protocol
VILLAS	Virtually Interconnected Laboratories for Large Scale Simulation
VoIP	Voice-over-IP
VPN	Virtual Private Network
VRI	Virtual Research Infrastructure

List of Figures

List of Tables

Listings

Abstract

There is a convergence occurring between the energy demands of modern society and the sustainability requirements of the environment in which we live in. The combination of these factors is driving the development and implementation of an updated power system with integration of Renewable Energy Sources (RES). This calls for an update of the existing infrastructure, with an additional layer of advanced monitoring, control and Information and Communication Technology (ICT) that is presently only beginning to be applied. With this introduction of ICT into the power system, a Cyber-Physical Energy System (CPES) is formed, that is multi-domain in nature. It encompasses interactions between the power system, communication as well as control and supervisory applications. Thus, the CPES is greater in scale and complexity, in comparison to the traditional electrical power grid. As a result, the study and analysis of this complex and large scale CPES is not feasible in one dedicated facility/Research Infrastructure (RI). Therefore, remotely interconnecting Smart Grid laboratories unlocks the potential to test large scale scenarios through Joint Research Activity (JRA). By conducting joint experiments between different RIs within Europe, resource sharing can be achieved. This enables usage of and interaction between assets located in each RI in a coordinated way. Hence, the application of control algorithms running in one RI for the remote control of devices which are physically distributed in other facilities becomes possible. Thus, this thesis studies the implementation and application of geographically distributed laboratory interconnections for smart grid testing. Broadly, two types of interconnections have been studied extensively, within the scope of this thesis:

- **Remote Hardware and Software Interconnection**
  At present, EU smart-grid laboratories are not connected by a common framework or infrastructure. Various EU labs have their own specialised facilities and interconnecting them will help in having all the required and important features under a common virtual platform. This thesis project work was undertaken for partial fulfillment of the objectives of developing interfaces, supporting software infrastructure for virtual integration under the H2020 funded project, **'ERIGrid'**. Developing an integrated research infrastructure for smart grid systems is the main objective of the ERIGrid project. The interconnection of labs is achieved using the innovative Joint Test Facility for Smart Energy Networks with Distributed Energy Resources (JaNDER) specification over the internet and involves exchange of critical real-time simulation information– measurements, control signals. A case study involving interconnections between TU Delft, The Netherlands, Technical University of Denmark (DTU) and VTT Multipower Laboratory, Finland is carried out. In the studied test case, the Real-Time Digital Simulator (RTDS) at TU Delft has been virtually interfaced with power hardware at at DTU and VTT. The proof of concept of a Virtual Research Infrastructure (VRI) is presented through a joint experiment involving all three labs, to study a geographically distributed power system. Results show the the performance validation of this distributed setup by comparison to a reference simulation.

- **Remote Software and Software Interconnection**
  As part of the ERIGrid Transnational Access Research Exchange, the real-time grid simulators at RWTH Aachen University, Aachen and TU Delft have been interconnected through the public internet. This interconnection is realised through a software tool-set called VILLAS Framework and is applied to carry out a Geographically Distributed Real-Time Simulation (GDRTS). A systematic and comprehensive analysis of a Dynamic Phasors (DP) based co-simulation interface algorithm for GDRTS is studied and its improvements are provided. The obtained results show that, to ensure simulation fidelity in a GDRTS on a shared communication medium, an automated approach to monitor the network and adapt to network congestion is required. Therefore, this thesis introduced the application of a Real-Time Protocol (RTP) in the real-time power system simulation domain.

1 Introduction

The electricity grid is one of the cornerstones of the modern world, without which, life is impossible to imagine. Fossil-fuel based non-renewable energy sources are at the heart of the existing power system. However, their polluting nature and the ever-increasing effects of climate change have necessitated a call for action [1]. Thus, there is an ***"Energy Transition"*** taking place globally, aiming to achieve fully renewable power generation in the near future. Europe in particular has ambitious targets for greenhouse gas reductions and decarbonization by 2020 [2]. This introduction of RES has introduced new challenges in system operation and planning [3]. Hence, advanced ICT based solutions are the need of the hour, to cope up with these challenges, moving forward.

1.1. Background and Motivation

The conventional electrical power grid is undergoing a paradigm shift due to the integration of Renewable Energy Sources (RES). With this integration of renewables and driven by technological developments, an ICT layer is emerging on top of the physical power grid, transforming the existing energy system into a cyber-physical system, which can be defined as a *Smart Grid* [4, 5]. The complexity of the present and future smart power system will require tools that are suited for simulations on a large scale, in order to study their interactions with newer energy sources and inter-operability of new control methods. Advancement in power electronics, driven by the increasing penetration of RES and HVDC calls for high-fidelity simulation tools and a shift from static load flow simulations to dynamic ones. The dynamics and control of such inverter dominated power systems have not been assessed much in detail. Hence, the stability of future power system operation with inverter dominated dynamics requires a matching assessment infrastructure.

As the future power system becomes complex and interconnected to multiple energy sources, a single research infrastructure can rarely provide the required resources to study this complex energy system. The ability to study large energy systems is not only limited by the lack of cross-carrier laboratories but also the absence of detailed large scale models (e.g coupled transmission and distribution system) and expertise. Hence, integrated and unified approaches for analysing and evaluating complex configurations in a cyber-physical system manner are limited.

With the advent of more powerful computational tools : Electromagnetic Transient (EMT) or Dynamic Phasors (DP), large-scale simulation becomes feasible and enables HIL testing through real-time simulation technologies. However, the required level of detail for such simulations cannot be concentrated in a single site for both technical and organizational reasons. Scaling digital real-time simulation in the power system domain to larger systems is a challenging task; not only limited by technical but also by financial and human factors. Detailed models are rarely available for large-scale dynamic simulations. The process for estimating dynamic parameters assumes full knowledge of static models and associated topologies. For confidentiality reasons, this information is usually inaccessible across RIs and partner sites.

GDRTS presents an approach which solves the issue of confidentiality, by distributing both the simulation as well as human work load across a set of participants and Research Infrastructure (RI). Small time steps, often in the range of micro to milliseconds, impose strict real-time computational burdens on the underlying computer and operating system. Thus, differentiating them from offline simulation tools used in other domains. It is an advanced concept which enhances experimental and testing capabilities. Real-time simulation resources, HIL set-ups, novel test benches and hybrid co-simulation frameworks can be interconnected virtually to form a comprehensive VRI platform that allows for sharing of resources present at different geographic locations. In the fields of scientific modelling and simulation, fidelity refers to the degree to which a model or simulation reproduces the state and behaviour of a real world object, feature or condition [6]. Fidelity is therefore, a measure of the realism of a model or simulation. The fidelity of a GDRTS is major challenge, which this thesis aims to address. Figure 1.1 shows the violation of quantities at the co-simulation interface in a GDRTS.

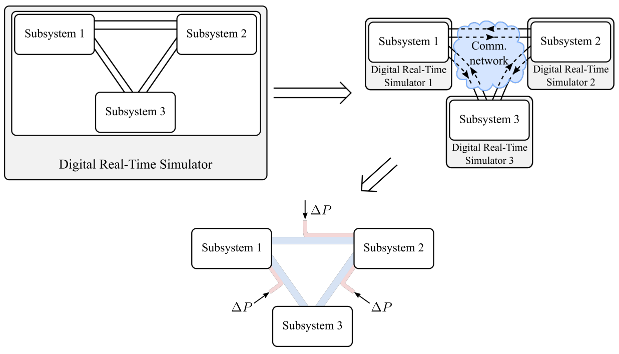

Subsystem 1
Subsystem 2
Subsystem 3
Digital Real-Time Simulator
Comm. network
Digital Real-Time Simulator 1
Digital Real-Time Simulator 2
Digital Real-Time Simulator 3
ΔP
ΔP
ΔP

Figure 1.1: Illustration of GDRTS and its challenges

The key objective of investigating this concept, is the potential development of a software platform for the remote interconnection of smart grid laboratories. This enables the possibility of conducting joint experiments between RIs in Europe, utilizing the different assets provided by each, in a coordinated way. The technical possibility of conducting such joint experiments allows the application of control algorithms running in one research infrastructure for the remote control of devices which are physically located in other facilities. This is shown in Figure 1.2. From the figure, it can be seen that the main advantage of the VRI concept is the possibility for one RI to access the resources located at a remote site - these resources can range from actual hardware devices to real time simulators or Supervisory Control and Data Acquisition (SCADA) systems.

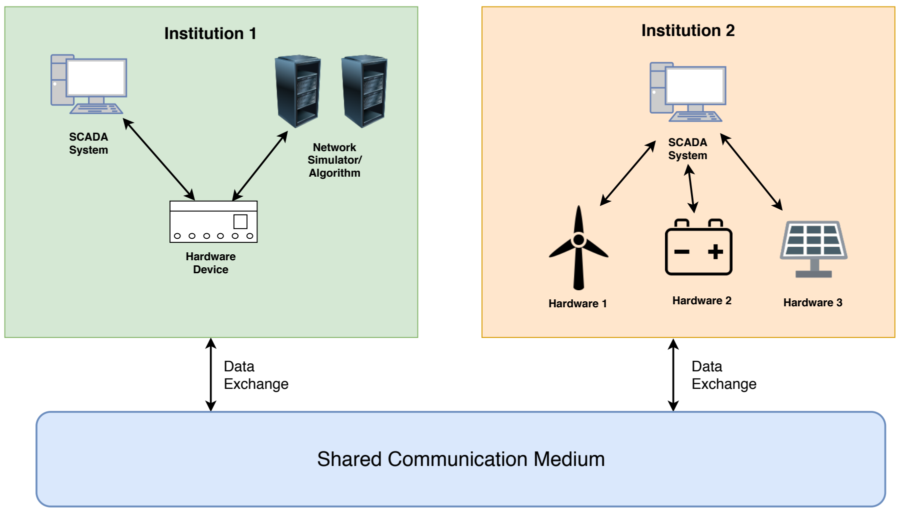


Figure 1.2: Virtual Research Infrastructure concept

1.2. Literature Review

1.2.1. Inter-Connection of Labs

In relation to this thesis, [7] serves as the most important reference. This work describes the efforts toward the realization of a large-scale inter-connected research infrastructure and explains a demonstration of the multi-lab setup for simulation and testing of next-generation global power grids. The idea of virtual integration of labs at different locations was originally investigated in [8]. This paper investigates the feasibility of geographically distributed real-time simulation. Two real time grid simulators located remotely are virtually connected. A simple HVDC point to point link that connects the two AC systems is adopted as a case study.

The proof of concept for virtual integration, however, is presented in [9], which introduces a framework for virtual integration of hardware and software assets at different geographical locations. The framework presented seeks to realise simulation as a service (SMaAS) concept. The paper presents real-time coupling of hardware and software assets located remotely through the internet.

1.2.2. Geographically Distributed Real-Time Simulation (GDRTS)

GDRTS is a novel research topic that addresses the requirements of using real-time simulation resources at multiple locations for joint testing [10–14]. Furthermore, [15] illustrates the development of a real-time power system solver which has been applied to perform GDRTS. A review of existing and emerging cyber-physical system test-beds is carried out in [16].

1.2.3. Co-simulation

In other related work, a novel co-simulation architecture that integrates hardware testing using Power Hardware-in-the-Loop (PHIL) techniques with larger-scale electric grid models is discussed in [17]. In addition to simplifying testing with multiple feeders, the architecture demonstrates additional flexibility with hardware testing in one location linked via the internet to software modeling in a remote location. The feasibility, applicability and benefits of a real-time virtual connection of laboratories for virtual integration of simulators and hardware assets was demonstrated in [18]. A co-simulation platform for analysing electric power grid operation, considering integrated communication systems using OPNET and OPAL-RT is discussed in [19].

A co-simulation framework for wide-area Smart Grid monitoring systems based on phasor measurement units (PMU) is presented in [20]. A simulator having both communication and power system capabilities is presented, based on existing public-domain simulators. OpenDSS and OMNET++ are used for simulation of power systems and communication networks respectively. A study of wireless and wired communication technologies for smart grid implementation is presented in [21]. The analysis shows that wireless technologies have suitability for smart grid applications. However, a combination of mixed wireless and wired technologies may introduce latencies that cannot serve critical functions such as power system protection.

1.2.4. Interface Algorithms for Distributed Simulations

A co-simulation of multi-area power systems was investigated in [22], which dealt with the requirement of large-scale simulation resources using real-time simulation at multiple institutions and the confidence for their grid models. In [23], a GDRTS of HVDC systems were implemented using ideal transformer model interface algorithm. This work showed that simulation fidelity is robust with a small time delay. However, there is room for improvement of co-simulation interface algorithms to guarantee simulation fidelity of all quantities. The use of Dynamic Phasors (DP) as a feasible approach to couple real-time simulators was demonstrated in [13, 24–26]. Similarly, [27] deals with interfacing of Dynamic Phasors (DP) with real-time EMT simulations. A really important reference with respect to this thesis is [28]. It focuses on mitigation of the effects of communication delays between multiple, virtually connected, yet physically separated HIL experiments. The proposed methodology is validated through simulation and hardware experimentation conducted between geographically dispersed laboratories. An approach similar to DP based on shifted frequency analysis is presented in [29], while [30] focuses on the improvement of the accuracy and stability in Power HIL experiments.

[31] deals with the current state-of-the-art in interfacing issues related to real-time digital simulators employed in the simulation of power systems and power-electronic systems. This paper provides an overview of technical challenges encountered and their solutions as the real-time digital simulators

evolved. Hardware-in-the-loop interfacing for controller hardware and power apparatus hardware are also presented.

1.3. ICT tools for Laboratory Interconnections

Literature proposes various methods for the validation of simulation models or systems. This can range from comparison with actual real data, comparison with another model etc [32]. In this thesis, both approaches for lab interconnections are validated by comparison to a monolithic simulation model. Within the scope of this thesis, JaNDER and VILLAS framework have been used extensively, for slightly different objectives as explained below:

1.3.1. JaNDER Specification

The starting point for the design of the JaNDER platform was in DERri (Distributed Energy Resources Research Infrastructure), a European project where the JaNDER concept was defined for the first time [33]. In DERri, JaNDER provided a common interface to control DERs in remote laboratories and measure relevant system data. The implementation of JaNDER in ERIGrid is based on a layered architecture, where each layer addresses a specific aspect and allows for a modular implementation based on the specific test cases which must be implemented as shown in Figure 1.3. As part of this thesis, JaNDER level 0 has been applied to realise a geographically dispersed **Software and Hardware Interconnection**. This is achieved by interconnecting RTDS at TU Delft with power hardware at other european labs as explained in Chapter 3.

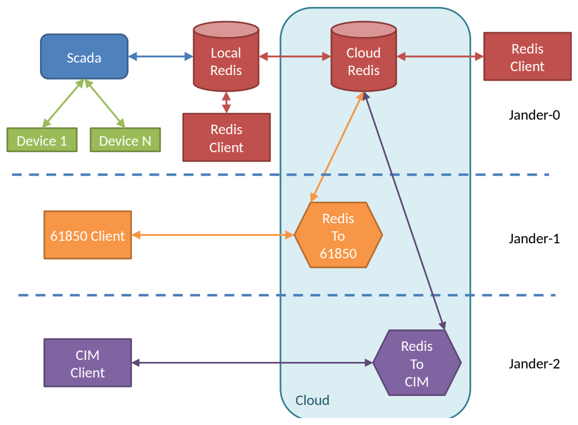


Figure 1.3: JaNDER Modular Architecture

1.3.2. VILLAS Framework

A component of VILLAS Framework, VILLASnode has been used to interconnect the real-time simulators at TU Delft and RWTH Aachen, to carry out GDRTS. This method demonstrates a remote, real-time **Software and Software Interconnection**. The implementation details are covered in Chapter 4.

1.4. Research Questions

The key research questions, this thesis aims to answer are as follows:

- How to interconnect real-time grid simulator at TU Delft with power hardware at other European labs to build a Virtual Research Infrastructure (VRI)? Benefits?
  - Study and validation of a geographically distributed, virtually coupled power system by comparison with monolithic simulation.
- How to interconnect two real-time simulators to perform Geographically Distributed Real-Time Simulation (GDRTS)?
  - Performance validation of a co-simulation interface algorithm for GDRTS based on Dynamic Phasors (DP). Stability and accuracy? Limitations? Improvements?

- What is the effect of the properties and physical limitations of the communication channel (internet) on a networked, distributed power system? Impact on co-simulation interface algorithm for GDRTS?

1.5. Methodology

This thesis work-flow is broadly divided into two parts:

1. **JaNDER based Lab Connections and Testing:** The interconnection of the labs requires development of coupling and interface algorithms which can be standardized across all labs. Currently, JaNDER specification is used for exchange of critical real-time simulation information (measurements, control signals, laboratory asset descriptions). However further testing is required under various complex real-time scenarios before JaNDER can be adapted as a lab interconnection standard. Hence, this thesis aims at studying and validating a distributed power system setup, wherein, resources present at different research labs are interconnected over large geographical distances using JaNDER. This is achieved by conducting joint test experiments with other European labs and institutions, namely – Technical University of Denmark (DTU) and VTT Technical Research Centre, Finland. This is covered in detail in Chapter 3.

2. **GDRTS using VILLAS Framework:** Over the past few years, RWTH Aachen has developed a framework for geographically distributed real-time co-simulation. This framework, called VILLAS framework can be used to virtually interconnect labs in the physical domain to perform larger Distributed Real-time Simulations. As part of the H2020 ERIGrid Transnational Access (TA) research exchange, the real-time simulators at RWTH Aachen and TU Delft have been interconnected to carry out GDRTS. Thus, this thesis will also investigate and validate the co-simulation interface algorithm based on dynamic phasors for GDRTS and contribute to its improvement. This is covered in Chapter 4.

1.6. Thesis Contributions

1. Implementation of JaNDER specification at TU Delft, as part of a joint research experiment, interconnecting labs at TU Delft, Technical University of Denmark (DTU) and VTT Technical Research Centre, Finland. Thus, contributing to the development of a Virtual Research Infrastructure (VRI) platform.

2. Setup of VILLASnode at TU Delft to perform GDRTS. Performance validation of a Dynamic Phasors (DP) based co-simulation interface algorithm for Geographically Distributed Real-Time Simulation (GDRTS). This is carried out by investigation of its stability and accuracy with respect to a monolithic simulation model.

1.7. Outline

This thesis document is organised as follows: This chapter gives an introduction to the project and the intended outcomes. Chapter 2 provides a background of the ICT tools used to achieve lab interconnections within the scope of this thesis. The implementation steps and methodology of JaNDER and VILLAS Framework are covered in Chapters 3 and 4 respectively. Chapter 5 deals with the observations and results of both the methods in depth. Finally, Chapter 6 draws conclusions of this thesis and also discusses future work to be undertaken.

2
ICT tools for Laboratory Interconnections

This chapter covers some of the key details of the tools/specifications used within the scope of this thesis. Both the methods to achieve laboratory interconnections – JaNDER and VILLAS framework are discussed.

2.1. Introduction

This thesis project was undertaken for partial fulfillment of the work to be carried out under the H2020 funded project, ERIGrid [34] under work package 10, JRA 4. This work package aims to implement and demonstrate the functioning of different test cases when various RIs are interconnected to each other. Developing an integrated research infrastructure for smart grid systems is the main requirement of ERIGrid project. Some of its key objectives can be summarized as:

- Supporting the technology development as well as the roll out of smart grid approaches, solutions and concepts in Europe with a holistic, cyber-physical systems approach.
- Integrating and enhancing the necessary research services for analysing, validating and testing smart grid system configurations [35].
- Coupling of research infrastructures for integrated and joint testing [36].

2.2. State of the art

The ERIGrid project deals with the remote interconnection of RI for the purpose of performing joint distributed tests online, thus, building a Virtual Research Infrastructure (VRI). This concept is validated by real implementation and demonstration of different use cases and testing scenarios, by using a dedicated ICT platform which enables remote data exchange between RIs over the internet. The purpose of this thesis is to illustrate in detail, the application of this platform, called JaNDER for remote testing and experimentation, covered in the subsequent section. More specifically, in this project, JaNDER Level 0 (L0) has been implemented and tested.

2.3. JaNDER Specification

The starting point for the design of the JaNDER platform was in DERri (Distributed Energy Resources Research Infrastructure), a European FP 7 project where the JaNDER concept was defined for the first time [33]. In DERri, JaNDER provided a common interface to control DERs in remote laboratories and measure relevant system data. The implementation of JaNDER in ERIGrid is based on a layered architecture, where each layer addresses a specific aspect and allows for a modular implementation based on the specific test cases which must be implemented. The lowest level (Level 0) implements the basic functionalities for remote connection over Internet, and is the basis for all the other layers. The requirements for implementing Level 0 are very minimal and any standard Internet connection can support it. The updated JaNDER platform is very easy to implement; it is modular, so that only the functionalities which are really needed for each test case must be configured, and is expandable by adding new levels. As an added advantage, most of the implemented software is available as open source. The high level architecture of JaNDER level 0 used at TU Delft as part of this thesis is shown in figure 2.1.

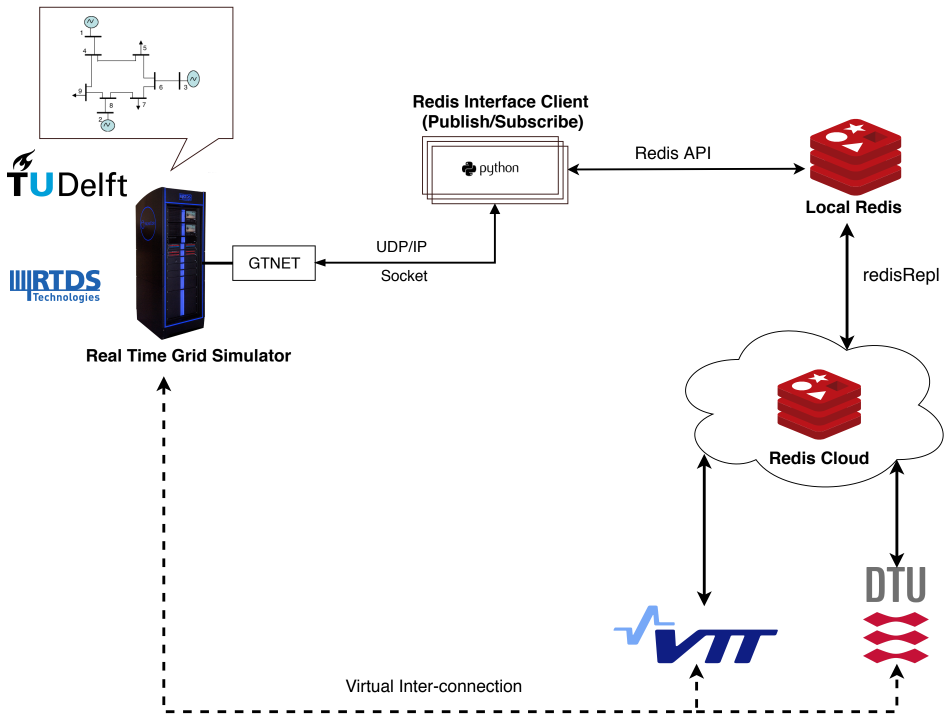


Figure 2.1: JaNDER implementation at TU Delft

2.3.1. Real Time Digital Simulator (RTDS)

A real time grid simulator (RTDS) consists of custom hardware and all-in-one software, specifically designed to perform real-time EMT simulations. It operates continuously in real time while providing accurate results over a frequency range from DC to 3 kHz. This range provides a greater depth of analysis than traditional stability or load flow programs which study phenomenon within a very limited frequency range.

The simulator operates continuously in real time, allowing analytical studies to be performed much faster than with offline simulation programs. Complex networks can be simulated using a typical time step of 25-50μs. Sub-networks consisting of fast switching power electronic devices operating with smaller time-steps in the range of 1-4 μs can also be simulated. RSCAD is RTDS Technologies' proprietary power system simulation software, designed specifically for interfacing to the RTDS hardware.

Figure 2.2: RTDS Hardware at TU Delft

2.3.2. GTNET Block

Figure 2.3 shows the GTNET-SKT component from RSCAD. Data may be exchanged between the RTDS and the external equipment over a LAN/WAN using the GTNET-SKT (SocKeT) Protocol. The SKT protocol is provided as a means to communicate with the RTDS using Transfer Control Protocol (TCP) or User Datagram Protocol (UDP) sockets [37]. A typical connection arrangement for the GTNET-SKT function on the RTDS is shown in figure 2.4. Data is exchanged between the socket communication firmware running on the GTNETx2 card and a RTDS processor card (PB5 or GPC) using a fiber optic connection between the cards. Data is exchanged between the socket communication firmware running on the GTNETx2 card and the remote equipment over a LAN connection. The remote machine may be dedicated hardware, a computer workstation or even another GTNET-SKT module running on another GTNETx2 card. In this case, the remote machine is a computer workstation running the redis database and GTNET-SKT component is configured to be in UDP send/receive mode. The SKT protocol represents both integer and floating point numbers as 4 bytes in the packet. Floating point numbers are single precision(32-bit) and are represented using IEEE754 format. The packet size is an integral multiple of 4 bytes, depending on number of data points transferred.

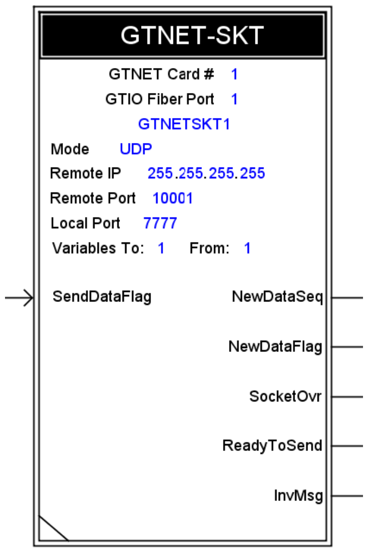


Figure 2.3: GTNET-SKT Component in RSCAD

Once the parameters have been configured and the connection is established, a UDP stream commences between RSCAD/Runtime and the remote machine through a bi-directional channel, also called UDP socket. As explained in Stevens *et al.* [37], most programming languages consider the socket as a file object. This implies that, simulation data can be fed at one end and taken out at the other. Thus, communication between RTDS and a remote machine becomes possible.

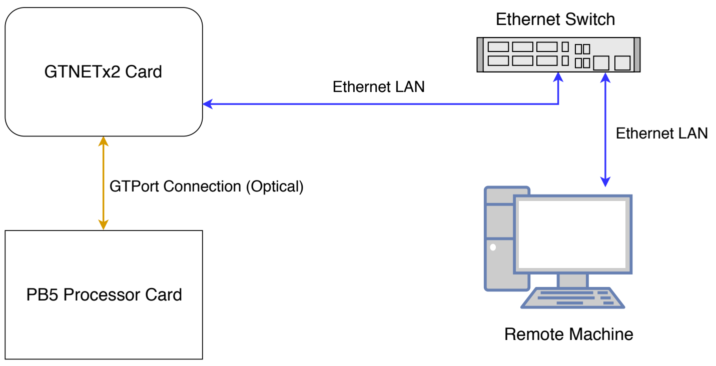
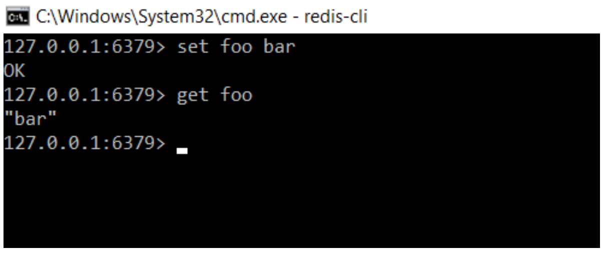


Figure 2.4: GTNET socket Connection

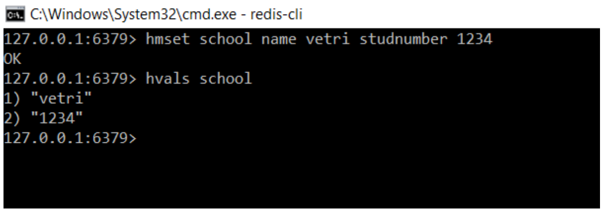
2.3.3. Local and cloud redis

The next step after communication between RSCAD and an external application/machine has been established, is to capture and store the simulation data. As per JaNDER specification, Redis is used as the database. Redis which stands for REmote DIctionary Server is an an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker [38]. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps and geo-spatial indexes with radius queries. Redis has built-in replication, scripting, transactions and different levels of on-disk persistence and provides high availability. Redis has many available data types, however for this thesis, two are of particular importance– key-values and hashes.

Keys are the building blocks of redis and a key is the variable name associated to which data needs to be stored. To this end, redis offers 'strings'. Redis String type is the simplest type of data that can be associated with a Redis key. Redis hashes serve as an extension to keys and can be described as a collection of key-value pairs. This is similar to the python 'dictionary' data type. In other words, redis hashes are maps between string fields and string values, so they are the well suited to represent objects. A hash with a few fields (where few means up to one hundred or so) is stored in a way that takes very little space, so one can store millions of objects in a small Redis instance. While hashes are used mainly to represent objects, they are capable of storing many elements, so they can be used for a wide range of tasks. In this thesis, all measurement values (i.e) data retrieved from simulations are stored in hashes, while control variables are simple key-value pairs. Figure 2.5 shows the working of keys and hashes respectively, as discussed.


```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379> set foo bar
OK
127.0.0.1:6379> get foo
"bar"
127.0.0.1:6379> _
```

(a) Working of redis keys


```
C:\Windows\System32\cmd.exe - redis-cli
127.0.0.1:6379> hmset school name vetri studnumber 1234
OK
127.0.0.1:6379> hvals school
1) "vetri"
2) "1234"
127.0.0.1:6379>
```

(b) Working of redis hashes

Figure 2.5: Redis variables

2.3.4. Redis publish/subscribe client

To store simulation data from the RTDS in the local redis database/instance, an appropriate client is required. It needs to be to interact with both redis and the RTDS. Redis offers scripting through a number of clients, however python is chosen as it open source and also compatible with RSCAD. Python is a free and open source, high-level programming language with a wide range of applications. Of particular interest to this thesis are its socket library and redis API. As described in the previous section, a UDP socket between RSCAD/RTDS and python ensures a bi-directional flow of data. The extracted data is then *published* to the local redis server through the use of the redis API. Similarly, data to be fed back into the simulation is *subscribed* to from the local redis and sent to RTDS. Thus the python script acts as a redis publish/subscribe client. The source code for the python script is provided in Appendix A.3.

2.3.5. Redis Replication (*redisRepl*)

The replication mechanism for JaNDER in ERIGrid has been completely redesigned. The new architecture uses a central master server, placed at one of the RIs and reachable from the public internet, to which all the other RIs connect. This eliminates the need for special types of network setup or firewall exceptions at the client RIs. Outward connectivity towards the public internet is the only requirement, presenting less administrative barriers since it is usually available by default. The main disadvantage of this setup is that a connection-oriented transport layer must be used which leaves TCP as the only transport layer option. Compared to UDP transport, this results in higher latencies and increased jitter due to TCP buffers, re-transmission logic and connection overhead. However, this is not expected to affect overall system performance at the time scale required for the multi-RI test case studied in this thesis.

Hence, the final link in the remote connection process is the replication of the local redis database into the cloud, which can then be accessed or read by other remote partner institutions. For this a small command line program, called 'redisRepl' is used to replicate remotely the commands sent to a local Redis instance [39]. The remote instance is assumed to be running behind a HTTPS interface. The HTTPS connection must use bi-directional authentication, so the appropriate certificate files must be provided via command line parameters. These files have been provided to all partner RIs. It is optionally possible to specify a name space (i.e. a string prefix) in the remote instance to be replicated locally. When started, redisRepl starts listening for events from a locally running Redis instance (in particular, string and hash events): every *'set'* or *'hset'* event is replicated remotely via HTTPS. If the *'-rc'* option is passed to *redisRepl*, then the reverse is also true: every *'set'* or *'hset'* event in the remote Redis instance is replicated locally. The architecture of *'redisrepl'* is illustrated through Figure 2.6.

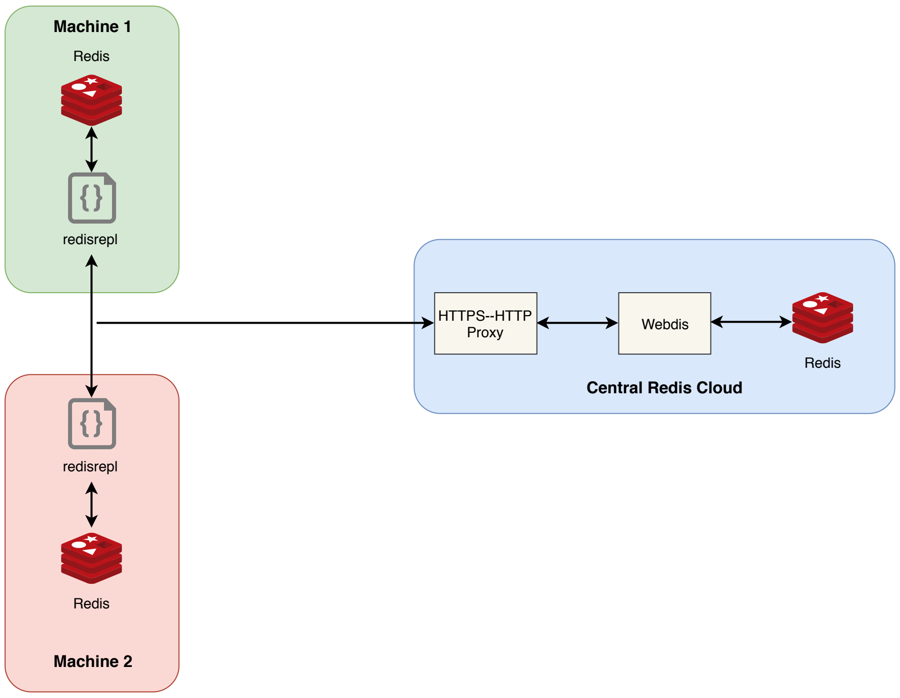


Figure 2.6: Redis replication structure

2.4. VILLAS Framework

VILLASframework is a toolset for local and Geographically Distributed Real-Time Simulation (GDRTS). It is actively developed by the Institute for Automation of Complex Power Systems at RWTH Aachen University [40]. As mentioned earlier, this framework has been used to interconnect the real-time simulators at Aachen and Delft to carry out distributed simulations in real-time. The overall structure and components of the *VILLASframework* can be seen in Figure 2.7. It is to be noted that, VILLASframework consists of several components:

- VILLASnode
- VILLASfpga
- VILLASweb

- VILLAScontroller

However, within the scope of this thesis, the most important component is the *VILLASnode* gateway which is described in the subsequent section.

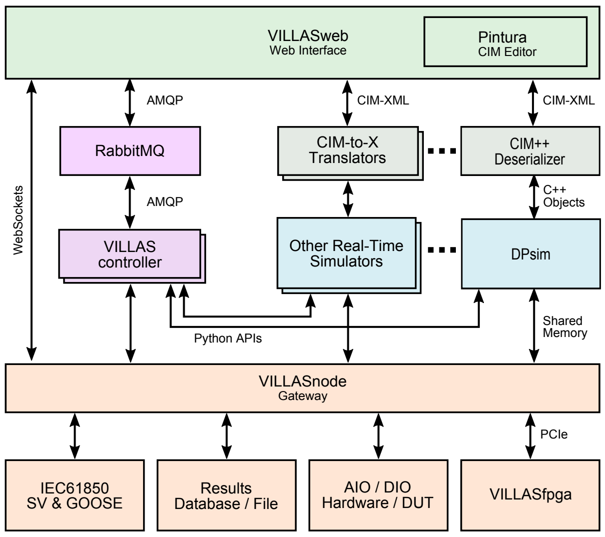


Figure 2.7: Components of VILLAS Framework

2.4.1. VILLASnode

VILLASnode is a modular gateway for simulation data. It offers interfaces to simulation equipment, databases and web services. It is tuned for the real-time exchange of simulation data in different formats and protocols. The software is written in C/C++ using the ISO C11 standard while following an object oriented programming paradigm. It is deployed within each laboratory as a Linux-based gateway server to establish VPN connections and handle data exchange between the local simulators and remote laboratories. The server simply acts as a gateway to forward simulation data from one client to another. For optimal performance, the server is implemented in low-level C and makes use of several Linux-specific real-time features. In this project, it is used on a Fedora-based Linux distribution which has been reduced to the bare minimum, with no GUI and only a few background processes. *VILLASnode* is designed around the concept of *nodes, paths and hooks*. It is the task of the server to forward real-time simulation data between multiple clients. In doing so, the server has to perform simple checks and collects statistics. From the viewpoint of the communication partners the server is nearly transparent. Hence, it's crucial to keep the latency as low as possible.

Nodes and Clients

All communication partners which are interfaced by the VILLASnode gateway are represented as nodes. These nodes act as sinks / sources for simulation data. Every node is an instance of a node-type. In a single VILLASnode instance, multiples instances of the same node-type can be created at the same time. Multiple types of nodes are supported as shown in Figure 2.8. The socket node-type is the most comprehensive and complex one. It allows to send and receive simulation data over the network. Internally it uses the well known BSD socket API. A way to connect simulation equipment is by using a client-application, which itself sends the data over the network to VILLASnode. In this scenario, VILLASnode uses the Sockets node-type to communicate with the client-application.

In addition to the supported Node-types, VILLASnode comes with examples for client applications / and model blocks. These clients usually use the Sockets node-type to exchange data with a VILLASnode instance via UDP packets. With respect to the experiments conducted in this thesis, RTDS acts as a *client* at both the labs and provides real-time simulation data with socket being the node type. The RTDS GTNET card with the SKT (Socket Fimrware) sends real-time simulation data to the VILLASnode as UDP packets.

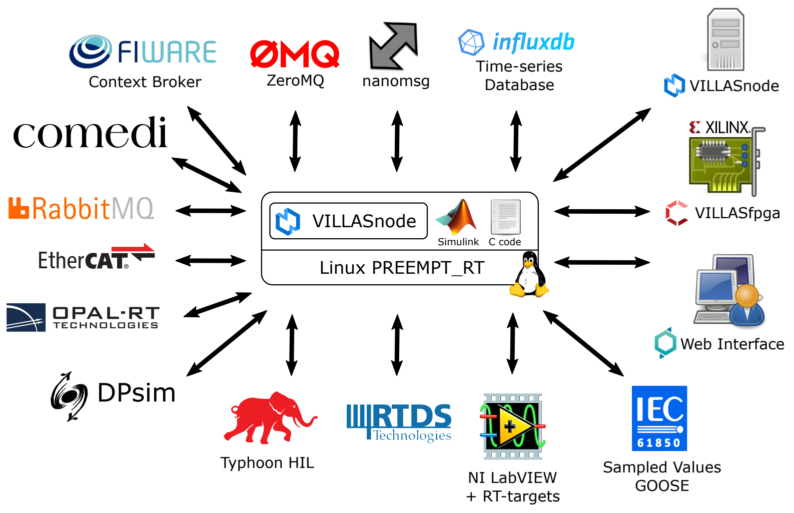


Figure 2.8: VILLASnode supported clients and nodes

Hooks

At times, forwarded sample data needs to be modified or filtered. VILLASnode supports hooks for this purpose. Hooks are simple callback functions which are called whenever a message is processed by a path. There are several built-in hooks for:

- Collecting, showing & resetting statistics
- Dropping reordered messages
- Verifying message meta data
- Handling simulation restarts
- Remapping values of a sample
- Overwriting / updating time-stamps
- Converting data-types

But the main goal of the hook mechanism is to provide extensibility to the end user. Example applications for hooks might be:

- Filter sample values
- Transform sample values: Fourier Transform
- Update network emulation settings based on sample values

Hooks can be added to the processing pipeline in three places:

- Node-read: Every time a sample is received from a node
- Node-write: Every-time a sample is sent to a node
- Path: Every time a sample is processed within a path

Some hooks are built-in, which are enabled by default, without a corresponding section in the configuration file. Usually, built-in hooks have no configurable options.

Paths

A path is a uni-directional connection between incoming and outgoing nodes. It forwards messages from a single incoming node to multiple outgoing nodes. Therefore it represents a 1-to-n relation between nodes. For bidirectional communication, a corresponding path in the reverse direction must be added. By default, message contents are not altered. The server only performs checks for valid message headers (sequence number, time-stamp..). However every path supports optional hook/callback functions which allow user-defined operations on the message contents. Some example node and path configurations are shown below. All the *node*, *hook* and *path* configurations used in this thesis project are provided in Appendix C.1.

```
nodes = {
        "tud_node" = {
                type = "socket",
                vectorize = 10,
                hooks = (
                        {
                                type = "decimate",
                                ratio = 10
                        }
                ),
                builtin = true,
                samplelen = 64,
                signals = (
                        { name = "Va", unit = "Volts", format = "float"},
                        { name = "Vb", unit = "Volts", format = "float"},
                        { name = "Vc", unit = "Volts", format = "float"},
                )
                # type specific settings follow here.
        }
}
```

Listing 2.1: Example node configuration

```
paths = {
                in = [
                        "rtds.data[0-5]",
                        "web.data[0-2]"
                ],
                out = [
                        "broker",
                        "opal"
                ],
                reverse = false,
                mode = "any",
                mask = [ "rtds" ],
                rate = 100,
                original_sequence_no = false,
                hooks = (
                        {
                                type = "print"
                        },
                        {
                                type = "ts"
                        }
                )
        }
```

Listing 2.2: Example path configuration

3 JaNDER Implementation

This chapter discusses the implementation of JaNDER, investigated within the scope of this thesis,for laboratory interconnections and testing. This work is undertaken as part of the **ERIGrid** project. The aim of this particular test is to study a system composed of virtually coupled, geographically dispersed sub-grids. By utilizing remote hardware, a wider range of tests can be performed without a need for additional investments in new equipment. The test also enables integration of remote real system behaviour into a real-time grid simulation.

3.1. Introduction

The overall objective of this test is to demonstrate that, by interconnecting the resources already present in European RIs, it is possible to create an extended Research Infrastructure (RI). With this, new use cases can be developed and studied, without any additional investment in new hardware. All the participating laboratories, linked with a standardized ICT solution –JaNDER, are integrated into a single VRI, encompassing the simulation / experimental potential made available by each lab. Thus, this enables the ability to participate in tests of complex use cases. The virtual emulation of an electrical interconnection is made possible by the communication platform used to exchange data, online. This platform, JaNDER Level 0 has been implemented in each RI involved in this test. Accordingly, this thesis aims also aims to demonstrate the JaNDER-Level 0 platform in a multi-RI experiment.

3.2. Test Setup

As part of the system under test, three RIs, namely – TU Delft, DTU and VTT have been interconnected to form a Virtual Research Infrastructure (VRI), i.e. a set of devices (simulated, emulated or physical) that act together following predefined criteria. In the studied test set-up, TU Delft acts as a grid simulator and simulates a distribution network using a real-time simulator. The other two RIs, namely DTU and VTT represent smaller physical parts of the network that are locally controlled. This section describes the communication and simulation test setups for the implementation of JaNDER.

3.2.1. Communication Network

The JaNDER method of interconnection is a cloud based solution to remotely connect laboratories which are geographically dispersed. As described in Chapter 2, a central cloud based server acts as a broker. This is an Amazon Web Services (AWS) cloud machine EC2 located at RSE in Italy. Consequently, the quality of the communication link between the central cloud and local redis database at every lab is an important performance criterion. Therefore, it is decided to investigate the one way latency of JaNDER Level 0 between the local and cloud redis. In particular, the test has been executed changing the number of measurements exchanged between the the local redis at TU Delft and the central redis cloud. This experiment, however does not consider the load of the different laboratories, but only the latency time (for JaNDER Level 0). The target measure is latency of data updates on the cloud platform. In the performed test, n number of measurements: 1, 10 and 100 were exchanged simultaneously between the local and cloud redis and time-stamps noted. 1000 repetitions were performed for every single type of measurement with a cooling period of 1s. Both the databases are synchronized through a Network Time Protocol (NTP) server in Italy, the same location as the central cloud. Hence, the comparison of time-stamps is indicative of the one way latency.

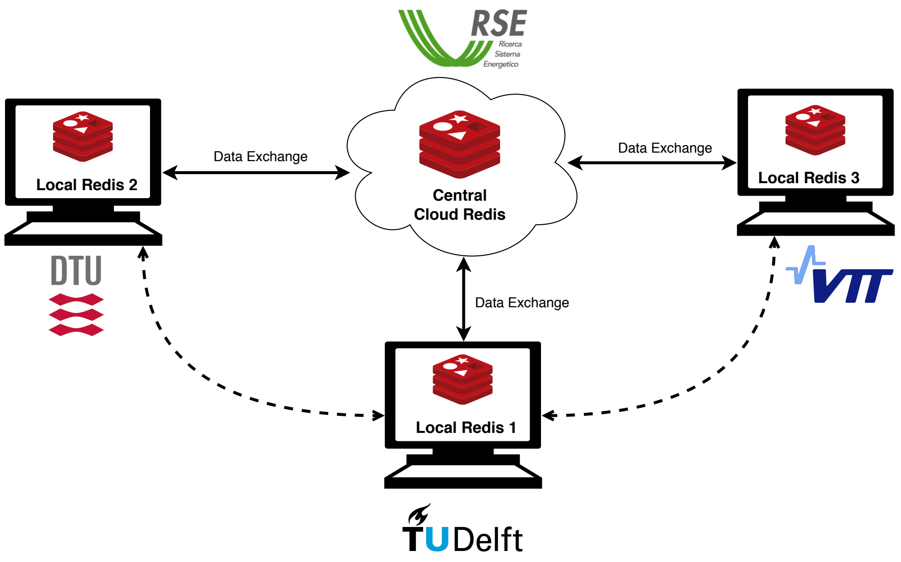


Figure 3.1: JaNDER Communication Setup

3.2.2. Experimental Setup at TU Delft

As part of the experiment, TU Delft provided its Real-Time Digital Simulator (RTDS) and acts as the grid simulator to which other two RIs connect remotely. The distribution network model simulated on the RTDS is a Low Voltage network based on [41, 42]. It is a single phase network with 11 buses, two of which are chosen to act as *Virtual Point of Common Coupling (PCC)*, Buses 8 and 11 respectively. All the loads, PVs and storage are modeled as controllable current sources. The input for their control blocks is a set of P (Active power) and Q (Reactive power) values. The control block then calculates the current magnitude and angle that corresponds to these P and Q values, with regard to the voltage of the bus that the current source is connected to. For buses 8 and 11, these P and Q values are received from the remote RIs, DTU and VTT.

The JaNDER specification has been implemented through a custom python script. This enables bi-directional data exchange between the RTDS and redis database by the use of UDP packets. To this end, the GTNET card of the RTDS in conjunction with the SKT(socket) firmware is used. The GTNET card supports data exchange rates between RTDS and an external machine, in the range of 1 to 20 kHz. This maximum value of 20 kHz corresponds to the default time-step of the simulator (i.e) $50\mu s$. Additionally, up-to 100 data-points from the simulation can be exchanged in every time-step. However, as per RTDS, an exchange rate of 5 kHz and transfer of 50 data points is recommended. Since, the other participating RIs within this experiment supply their hardware, only lower exchange rates between 1 to 10 Hz have been used. Further, the number of variables sent and received are 4 each, respectively. Voltage measurements at the PCC is calculated by the real-time simulator and sent to the remote RIs which act as micro-grids and provide feedback active (P) and reactive power (Q) set-points. This can be visualized through Figure 3.5.

3.2.3. Experimental Setup at VTT

VTT provided a grid emulator and a load within its low voltage network. The VTT RI acted as a sub-network of the main grid simulated by TUD. The virtual PCC between TUD and VTT is realised via a grid emulator, which sets the voltage and the frequency of the network. The range for grid emulator is 0-277 V_{rms} and 10-400 Hz. The grid emulator is connected to several different resistive loads during the testing ranging from 2 to 15 kW. Other devices which are part of this sub-network are two feeder protection and control relays connected to each line of the busbar. The measurements are sent to the substation computer around every 300 ms. The substation computer is connected to the local Redis via a Modbus client program on python. From the local Redis the data is exchanged with the other RIs via the JaNDER-L0 interconnection. The variables used in the experiment are listed in Table 3.1.

The measurements are published to redis in hashes containing the value, time stamp and the quality indicator for each value, while controls are published as key value pairs. The measurements published are active and reactive power, phase voltage and frequency for the phase A. The controls published are the phase voltage and frequency set points. The local Redis is installed on the same server as the redisRepl application for replicating the data from the local Redis to the cloud Redis and vice-versa. The measurements are updated to the cloud Redis every second and controls every second. The internal update time for the controls is 100 ms. In the experiment, VTT received the set points for the phase voltage and frequency at the virtual PCC from main grid simulated by TUD and sends measurements of active and reactive power at the virtual PCC to the main grid at TUD. To avoid damaging the hardware due to the remote connection, all the set points received are reviewed by the Modbus client. If a set-point is not within the limits defined on the Modbus client, the controls are set to the nearest limit. Additionally, all the devices part of the experiment have also own local protection to diminish the risk of damage.

3.2.4. Experimental Setup at DTU

DTU's setup consists of a grid emulator and a branched network with a controllable load and a (non-controllable) wind turbine, each connected at the end of a long line from the grid emulator. Due to operational difficulties with the units scheduled to be used as a grid emulator, multiple candidates for grid emulator were tried over the course of the testing period: A 125 kVA back-to-back converter, the 20 kVA inverter of a redox-flow battery and finally a 15kVA three-phase linear amplifier. Due to the different connection points of these units in the laboratory, the topology of the test system had to be adapted each time.

The grid emulator unit used for the tests is a Spitzenberger & Spies DM-15000/PAS 3x5kW linear amplifier system. It consists of three 5 kVA amplifiers which are controlled through the manufacturer supplied SyCore unit. The latter provides an IEEE488 control interface as the only remote control input. The characteristics of this interface are the primary factor limiting the response time of the unit. A full set-point cycle requires the setting of the main oscillator frequency (20ms) as well as three individual waveform amplitudes (ca. 40ms each), resulting in an overall response time of ca. 140ms (~7Hz).

While the particular amplifier model is equipped with voltage and current feedback which can be read through the IEEE488 interface, active and reactive power measurements are not available due to the lack of a suitable measurement module. The P and Q readings of the amplifier must therefore be taken from the panel instrumentation in the substations which have a lower measurement rate of ca. 1Hz.

The load at the end of the first grid branch (Load 2.1) is a three-phase, 80kW load bank consisting of an array of 3x8 resistors switched by semiconductor relays with zero-crossing detection. The resistors form an exponential cascade with 256 load steps per phase. The worst-case response time of the unit is one half cycle (~10ms). The combined network between TU Delft, DTU and VTT is shown in Figure 3.2. The orange lines represent *virtual PCC* implemented through JaNDER Level 0, while green rectangles illustrate the boundaries of each RI.

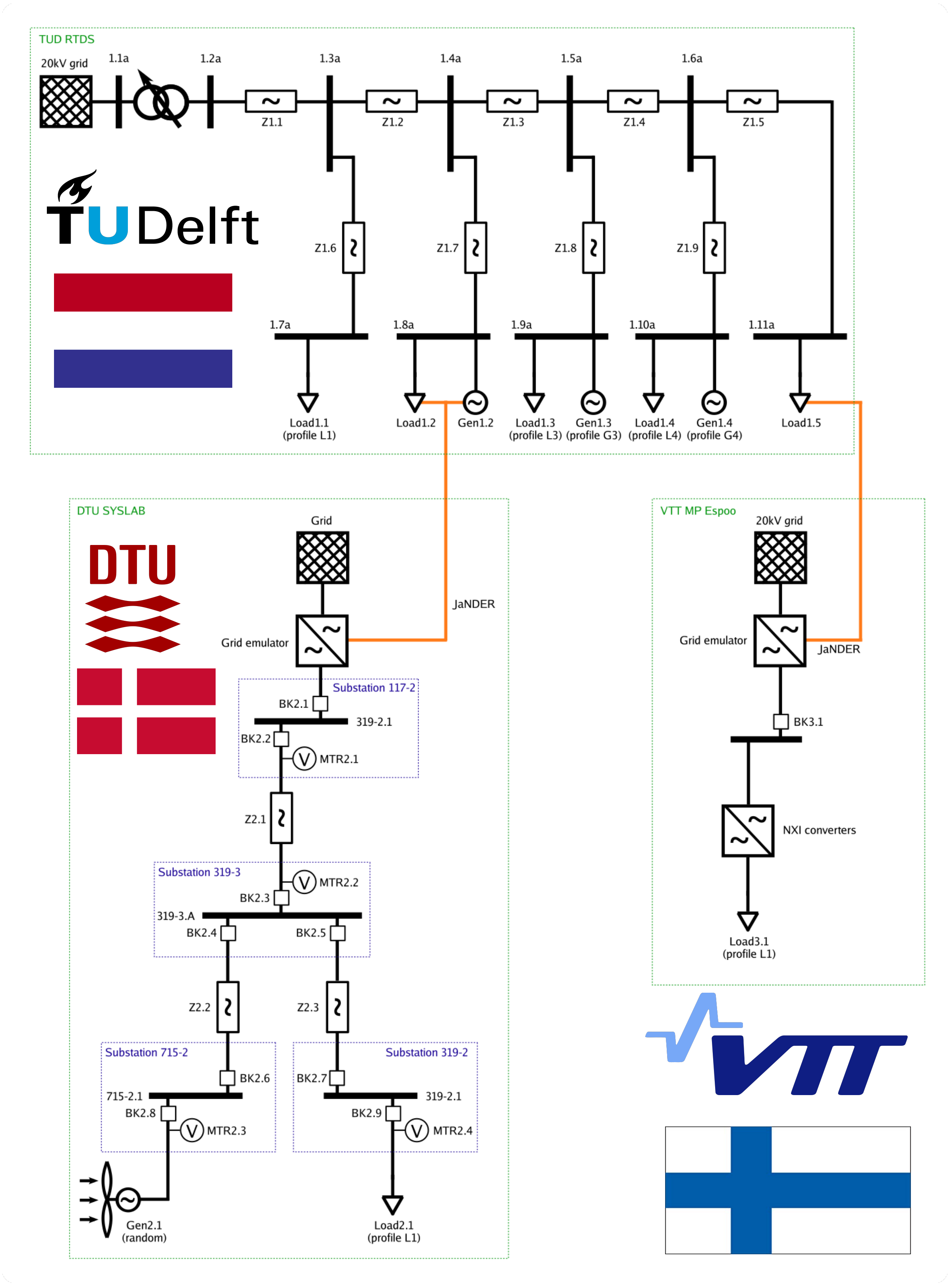


Figure 3.2: Single Line Diagram of Combined System

3.3. Implementation

The implementation of the experiment is carried out in incremental steps as follows:

3.3.1. Data Exchange using JaNDER

This first step of implementation involves connecting all the RIs remotely to verify that they can connect and communicate with each other through JaNDER. This is a fundamental requirement without which, no further testing can be carried out. The chosen test system requires a significant number of signals pertaining to measurements of physical grid quantities to be transferred from the SCADA systems in DTU and VTT to the RTDS at TU Delft, in real-time. Similarly, the grid simulator sends the set-points to all the devices in the other RIs. via redis. To achieve a successful remote connection test, all the participating RIs must be able to read and write into the redis variables. For this, proper naming conventions are to be followed. Table 3.1 lists all the redis variables used in this experiment. The workflow to exchange measurement data and set points is shown in Figure 3.3.

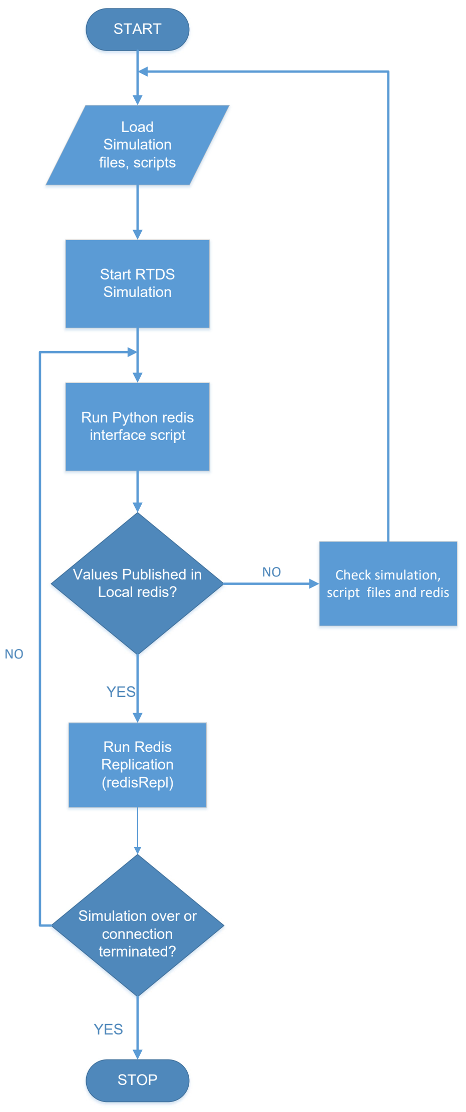

START
Load Simulation files, scripts
Start RTDS Simulation
Run Python redis interface script
Values Published in Local redis?
NO
Check simulation, script files and redis
YES
Run Redis Replication (redisRepl)
NO
Simulation over or connection terminated?
YES
STOP

Figure 3.3: Flowchart of JanDER level 0 implementation at TU Delft for data exchange

3.3.2. Quasi Static Soft HIL

The second step of implementation is the establishment of a *virtual* electrical connection by the exchange of data set points – voltage and frequency from the grid simulator to other remote RIs. In return, these RIs provide power set-points (P and Q) from their physical devices. This data exchange is independent of any control algorithm to observe steady-state system behaviour.

Open-loop configuration

In the open loop configuration, as the name suggests, data is exchanged only in one direction. The RI with the grid simulator, sets the voltage and frequency at the grid interface of the remote RIs, but receives no feedback in the form of P and Q measurements. In order to verify the interconnection between the three labs, JaNDER level 0 is used. All the RIs start two instances of Redis replication: one to publish their measurements and one to read the measurements of the other RIs. Since this test is an open-loop test, the frequency and voltage values obtained from the simulation are set at the grid interfaces of DTU and VTT by TUD. On the contrary, TUD does not send the P and Q measurements back to the RTDS. This can be seen through Figure 3.4.

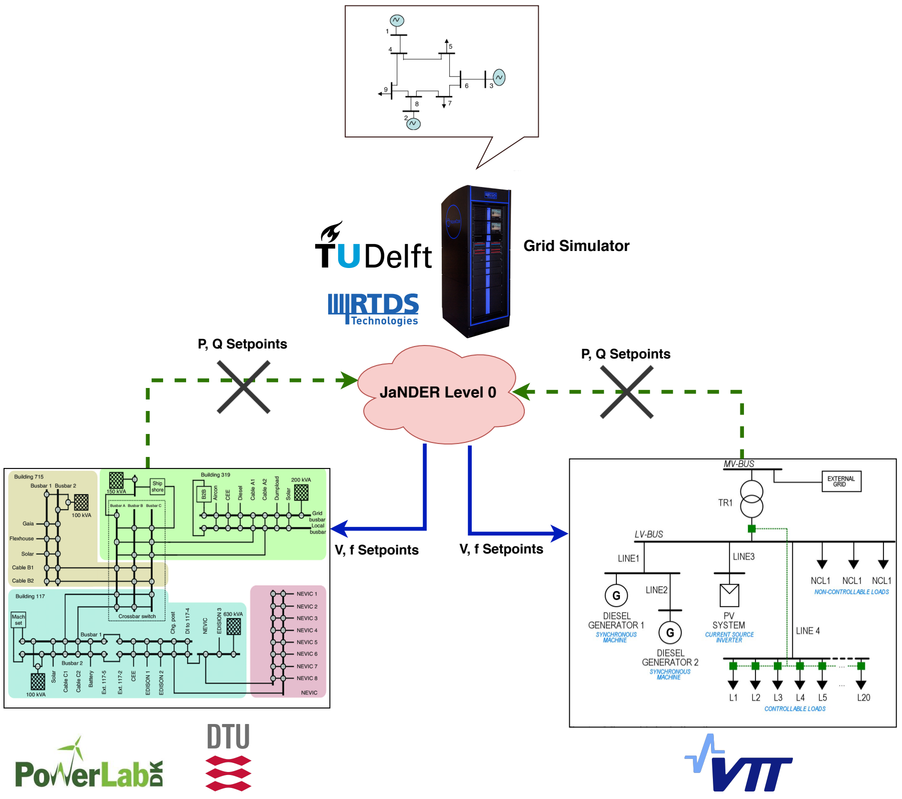


Figure 3.4: Soft HIL Open-loop configuration

Closed-loop configuration

In the closed-loop test, all three RIs exchange data, with feedback P and Q measurements from DTU and VTT integrated into the real-time simulation at TU Delft, thus closing the loop. This implies that, the system under test is now geographically distributed, but virtually coupled. In order to test this system, actual hardware at DTU and VTT is used. The devices used as part of the experiment are controllable loads at both locations. Subsequently, load step events are performed and the resulting behaviour/performance of the geographically distributed, virtually coupled system is noted. The purpose of this test is the characterization and validation of steady-state precision. To this end, the virtually coupled, geographically distributed physical system is also modeled in a real-time simulator as a monolithic reference to validate its performance. The results of all these cases are covered in Chapter 5.

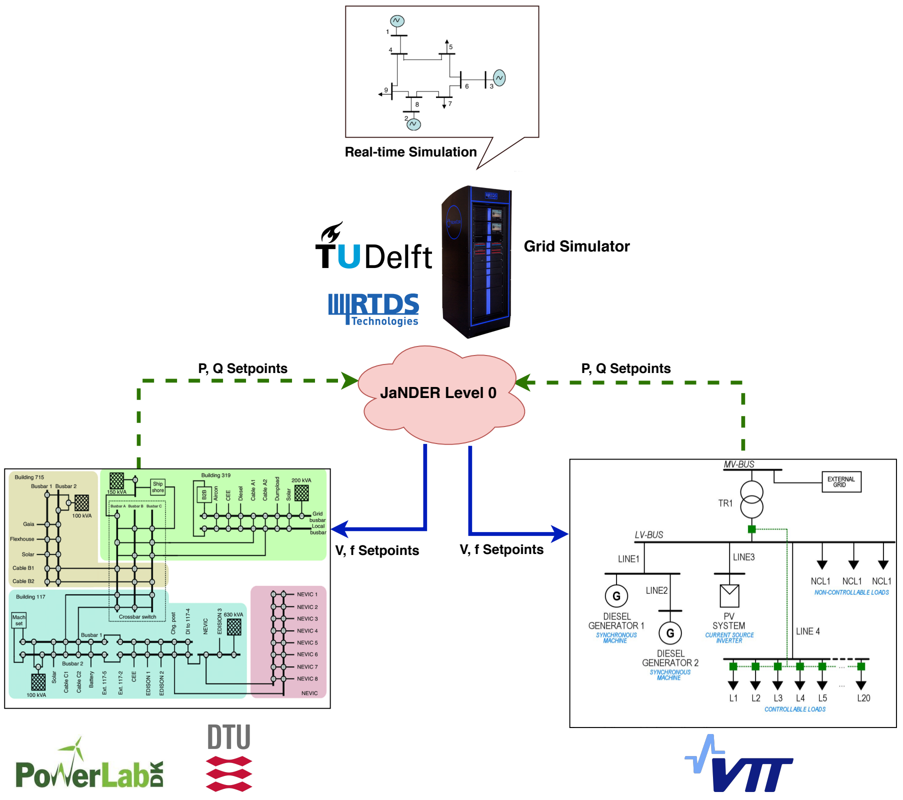


Figure 3.5: Soft HIL Closed-loop configuration

Table 3.1: List of Redis Variables

RI: DTU	
Redis Variable	**Description**
DTU::Storage_Vanadium:S_EA_PPV.instMag	Voltage setpoint for grid interface
DTU::Storage_Vanadium:S_EA_Hz.instMag	Frequency setpoint for grid interface
DTU::Storage_Vanadium:M_EA_Watt.instMag	Resulting active power load on grid interface
DTU::Storage_Vanadium:M_EA_Var.instMag	Resulting reactive power load on grid interface
RI: VTT	
VTT:Source_JK-T.KK-TRB.emulator:C_EA_PhV.phsA.ctlVal	Phase A voltage setpoint for grid interface (0-277 V_{rms})
VTT:Source_JK-T.KK-TRB.emulator:C_EA_Hz.phsA.ctlVal	Phase A frequency setpoint for grid interface (10-400 Hz)
VTT:Source_JK-T.KK-TRB.emulator:M_EA_W.phsA.instMag	Resulting active power load on grid interface
VTT:Source_JK-T.KK-TRB.emulator:M_EA_VAr.phsA.instMag	Resulting reactive power load on grid interface
RI: TUD	
TUD:RTDS_PCC_A:Meas	Voltage and Frequency Measurements at PCC A (Bus 8)
TUD:RTDS_PCC_B:Meas	Voltage and Frequency Measurements at PCC B (Bus 11)

4 VILLAS Framework for GDRTS

This chapter discusses the application of VILLAS framework, a toolset for Geographically Distributed Real-Time Simulation (GDRTS) which has been used to interconnect the real-time simulators at RWTH Aachen and TU Delft. The architecture of the experimental setup and detailed communication and simulation test cases are described. Additionally, study of the co-simulation interface algorithm based on Dynamic Phasors (DP) is also dealt with, in detail.

4.1. Architecture

This section describes the system architecture used for GDRTS through VILLAS framework as illustrated in Figure 2.7.

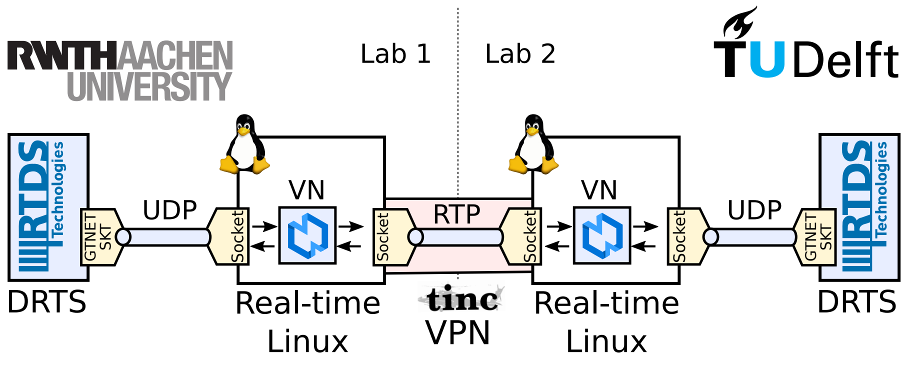


Figure 4.1: Architecture of VILLAS framework implementation

4.1.1. Communication Network

Interface signals from the electrical domain are exchanged over the public internet between both real-time simulators. For best results, both laboratories are connected via their university networks to the National Research and Education Networks (NREN). Namely, Germany's *Deutsches Forschungs Netzwerk (DFN)* and the Netherlands' *SURFnet* are non-profit organizations which are themselves interconnected the the European GÉANT network. In contrast to special purpose computing networks such as the LHC Computing Grid (LGC) by CERN [43], the networks used in this thesis are part of the public internet and thus, shared with other users, which may result in unpredictable communication latencies or packet loss. The DP based co-simulation interface algorithm described subsequently in this chapter, can mitigate the consequences of these disturbances on simulation performance.

Virtual Private Network: Tinc

For GDRTS it is crucial to establish a direct point-to-point connection between the labs to maintain low latency and jitter. As in most cases, computing equipment in both laboratories is protected against external threats via a firewall which filters incoming network traffic. There are two approaches to circumvent this restriction:

1. Virtual Private Network (VPN) can tunnel network traffic between both sites. Most of the time this requires the addition of exceptions to the firewall rule-set.

2. A technique referred to as *Interactive Connectivity Establishment* which facilitates peer-to-peer networking as used by Voice-over-IP (VoIP) or other Video Conferencing Solutions such as Microsoft's Skype [44].

In this thesis, Tinc-VPN software has been used which leverages both above mentioned techniques to establish a fully meshed VPN between multiple labs [45]. It does so by using a publicly reachable node which supports the restricted nodes in their connection establishment. However, after this initial connection establishment, all critical real-time traffic is exchanged peer-to-peer between the laboratories.

4.1.2. Co-simulation Gateway: VILLASnode

Within each laboratory, a Linux-based gateway machine is deployed to establish VPN connections and handle data exchange between the local simulators and remote laboratories. For the data exchange *VILLASnode*, a component of the *VILLASframework*, is used. VILLASnode is a C/C++ application tuned for the real-time exchange of simulation data in different formats and protocols. In this setup, it handles the collection of statistics on the communication link as well as the protocol conversion between the UDP based connection to RTDS' GTNET card and the Real-time Transfer Protocol (RTP) which has been used between the laboratories. Its working has been explained in depth, previously, in Chapter 2.

4.1.3. Real-time Protocol: RTP / RTCP

The rate at which simulators exchange their interface signals is one of the key factors, which affects accuracy of simulation results. The maximum rate is limited only by the available bandwidth of the communication link. As the link is shared with other users, the available bandwidth varies with time, which may cause congestion. Congestion has to be avoided as it leads to packet loss and a degradation of simulation fidelity.

In the past, a static rate has been determined by empirical tests preceding the actual simulation tests. This approach is cumbersome as it needs to be repeated for every new communication link and possibly at different times of a day. To improve this situation, this project implements a congestion avoidance scheme based on a real-time protocol which dynamically adjusts the sending rate.

For data exchange between the gateway machines, RTP and its sibling, Real-Time Transfer Control Protocol (RTCP) are used [44]. Both protocols are used in conjunction: RTP handles data transfer whereas RTCP is used to exchange Quality of Service (QoS) reports which measure packet loss, delay and jitter. These reports are the main advantage of RTP over plain User Datagram Protocol (UDP) packets, as they can be used by the sender to adjust its behaviour.

RTP and RTCP are widely used protocols in real-time multimedia streaming applications such as Voice-over-IP (VoIP) or Video-on-Demand streaming. In these applications, RTCP receiver reports are used to adjust the bit rate of multimedia codecs based on the current network conditions in order to avoid stuttering and lags in the stream. In the context of this work, the receiver reports are used to change the sampling (sending) rate of signals which are a exchanged between the simulators. RTP encapsulates the simulation signals in a raw IEEE 754 single-precision floating format and adds header fields for time-stamps as well as a sequence number. In this project, RTP uses UDP as the underlying transport protocol.

4.1.4. Real-Time Digital Simulators: RTDS

Both labs operate digital real-time simulators from RTDS Technologies. As part of this project, two Novacor chassis at RWTH and one PB5 rack at TU Delft have been used to simulate the system described subsequently. For synchronization, both RTDS installations are equipped with GTSYNC extensions cards which use the Global Position System (GPS) to synchronize the time-steps as well as the simulation start to a common time reference. However, as seen later, synchronization is not an essential requirement for the fidelity of the simulation, but instrumental for the collection and alignment of results.

4.2. Test Cases

4.2.1. Communication Tests

In order to interconnect the real time simulators at both labs, the network topology and underlying characteristics of the communication network have to be studied. The following communication tests are conducted, for the same:

Ping & Trace-route tests

The most basic of communication tests, involve using the standard ping and trace-route tools to gain a basic idea about the network connecting the two sites. For the ping test, 10000 packets in total, in intervals of 10 ms are transmitted from one VILLASnode gateway machine to the other and the average Round Trip Time (RTT) is recorded. This is found to be in the range of 12.7 to 13 ms. Next, using the trace-route tool, the routing path between RWTH Aachen and TU Delft is found with a total number of seventeen hops. Additonally, using the mtr (My Traceroute) command, each individual hop is pinged with 1000 packets in intervals of 100 ms and the resulting statistics of the network are studied. These are presented in Chapter 5.

Between VILLASnode Gateways

In this test, data is exchanged between the *VILLASnode* gateway machines running Tinc VPN in a *loop-back configuration* through UDP. Sine-waves with varying frequencies are generated at either location and every sample is transmitted to the other site, with a time-stamp. The received samples on the other end, are re-transmitted back to the source without any changes to the encapsulated data. By comparing the sent and received time-stamps, the communication latency between the sites is estimated. This test is carried out for the following combinations:

- With a fixed sending rate and variable data size. Data size is varied from 1 to 100 data points in steps of 10 data points.
- With a variable sending rate and fixed data size. Sending rate is varied from 1 packet/s to 20000 packets/s in steps of 500 packets/s.

Between Simulators using GTSYNC

As the final step of communication testing, data exchange between both the real-time simulators is carried out. To achieve this, models which test the communication latency, jitter and maximum packet rate were developed. Using the GTSYNC cards at both labs, simulations are started concurrently. Simulation data is transferred from one RTDS to its local VILLASnode machine through the GTNET card and GTNET SKT (socket) protocol. The SKT protocol represents both integer and floating point numbers as 4 bytes in the packet. Floating point numbers are single precision(32-bit) and are represented using IEEE754 format. The packet size is an integral multiple of 4 bytes, depending on number of data points sent. From there on, this data is sent through the internet to the other lab, where is it received by its VILLASnode machine and transferred to the RTDS using a similar setup. The IP addresses, ports and variables to be exchanged are specified in the RSCAD model files.

4.2.2. Simulation Test Cases

The system under study in this project is based on a simple power system consisting of a voltage source connected to loads through a transmission line as shown in Figure 4.2. This system represents the most simple version of a transmission / distribution network, where the co-simulation interface is located at the sub-station. Initial tests were conducted with an ideal voltage source to validate the co-simulation setup. For validation, three different variants of the model have been compared:

Monolithic model

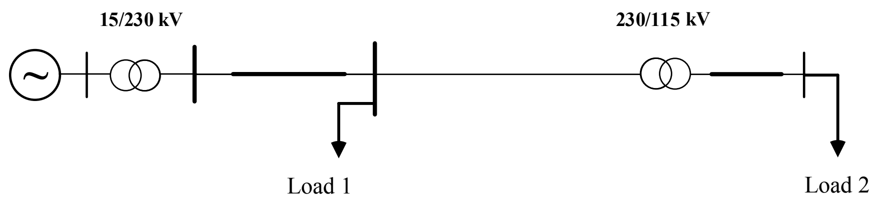
Monolithic model is modelled in real-time simulator using a single RTDS rack where the entire system is simulated in a single subsystem with 50 µs time step as shown in Figure 4.2a. It serves as a reference with no time delay for the comparison of the decoupled and distributed model.

Decoupled model

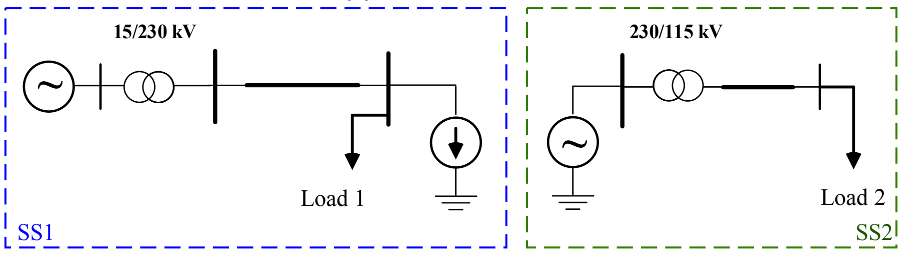
The decoupled model shown in Figure 4.2b is derived from the monolithic model where the system is separated into two subsystems at the 230 kV bus based on Ideal Transformer Model (ITM) approach for co-simulation as described in the next section. The simulation is executed across two RTDS racks but still contained and started by a single RSCAD/draft file. Signal exchange between the subsystems is handled by cross-rack signal import/exports. Additional communication delay can optionally be applied to these cross rack signals by using RSCAD control components.

Distributed model

The distributed model is derived from the decoupled model by moving one subsystem entirely to a separate RTDS simulation. The co-simulation now spans two independent RSCAD/draft files which are started separately. The signal exchange of the subsystems is handled by the VILLASnode gateway machines, as discussed previously.



(a) Monolithic model



(b) Decoupled/distributed model

Figure 4.2: Simple power system diagram

4.3. Dynamic Phasors (DP) Co-Simulation Interface Algorithm

The co-simulation interface algorithm used in this thesis is based on the ideal transformer model (ITM) approach. As illustrated in Figure 4.2b, controlled sources are utilized to impose voltage and current measured at the interface. GDRTS systems are based on Electromagnetic Transient (EMT) simulations, where current and voltage quantities are as instantaneous values. Direct sampling and transfer of these instantaneous values is not preferable. This is because, across a shared wide-area communication network that is characterized by a relatively large and time-varying delay, transfer of these values would significantly deteriorate the wave-forms and simulation fidelity. As a solution, two main approaches are proposed in literature. Representation of interface quantities in the form of Root Mean Square (RMS), Frequency and Phase angle is proposed in [46]. In this work, co-simulation interface algorithm based on representation of current and voltage at the interface in the form of time-varying Fourier coefficients, known as Dynamic Phasors (DP) is utilized [13].

Figure 4.3 illustrates transformation of waveforms to a DP based representation before sampling and sending interface quantities to the remote GDRTS system. At the receiving side, the DP quantities are reconstructed back to the time-domain to serve as reference for the controlled sources as illustrated in the Figure 4.4. DP interface algorithm allows for the compensation of time-varying delay based on phase shift within reconstruction of time-domain waveform. This compensation approach has been proposed for Power Hardware-In-the-Loop (PHIL) in [47] and adapted for GDRTS in [13]. Details of the implementation and comprehensive analysis of the DP-based co-simulation interface algorithm are given in the following Section.

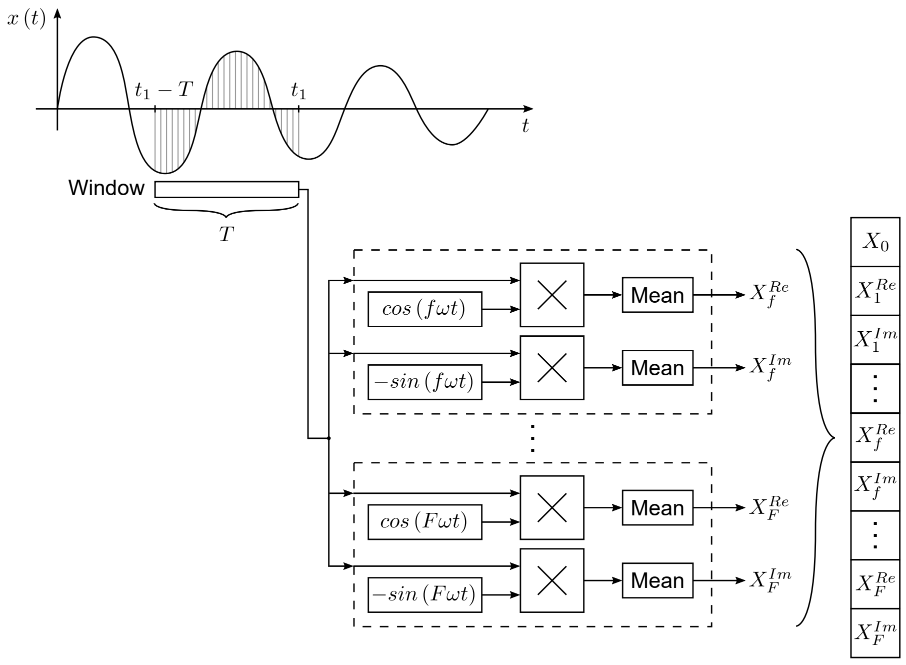


Figure 4.3: Calculation of DP coefficients

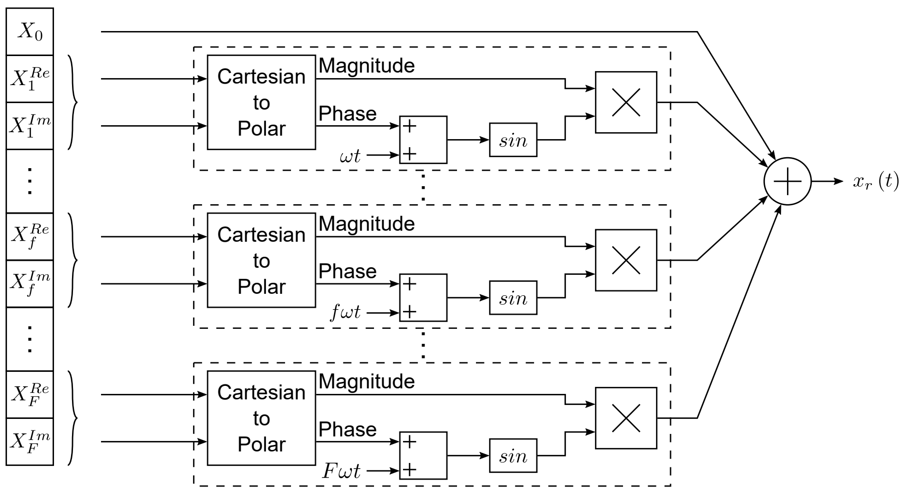


Figure 4.4: Reconstruction of the time-domain waveform based on DP coefficients

4.4. Implementation

4.4.1. DFT Window Length

In a GDRTS, the power system is simulated in the time domain through discrete and equal time steps. As a consequence, the Fourier transform used to calculate dynamic phasors must be a Discrete Fourier Transform (DFT). The DFT is implemented as a series over the moving window of the product of signal $x[n]$ with reference phasors:

$$X_k[m] = \frac{1}{N} \sum_{n=m-N}^{m} x[n] \cdot e^{-j2\pi f_0 kn} \tag{4.1}$$

As a result, the Fourier coefficients describe the harmonic components of the interface quantity. The length of the DFT window is chosen so that it covers one period of the fundamental frequency of the signal. For example, in a 60 Hz system which is simulated with a 50 µs time step T_s, the window has a length of $N \approx (1/f_0)/T_s) = 333,\overline{3}$. As this window length is not an integer number, the resulting Fourier coefficients do not represent the harmonic components of the 60 Hz system accurately. If this error is not properly taken into account, RMS values of interface quantities will not be identical and the power exchanged at the co-simulation interface will be imbalanced even in steady state. As indicated in the Table 4.1, a possible solution to this issue is the adjustment of the simulation time step such that, the DFT window length is closer to an integral number: $T_s = (1/f_0)/334 \approx 49,9$ µs

Table 4.1: RMS values of interface quantities with respect to the DFT window lengths

DFT window			Interface quantity			
			$V_{A,rms}$ [kV]		$I_{A,rms}$ [A]	
T_s [µs]	N	$length$ [ms]	SS1	SS2	SS1	SS2
50	333	16.65	136.7	136.0	51.64	51.9
50	334	16.7	136.6	137.9	52.56	52.05
49.9	334	16.6666	136.6	136.6	52.56	52.56

It is to be noted that, this problem only occurs for systems whose fundamental period is not evenly divisible by the simulation time step. E.g. for simulations with a 50 Hz system frequency and a 50 µs time step, the DFT window exactly spans 400 steps.

4.4.2. DFT Calculation

A critical step for performance of the Dynamic Phasors (DP) Interface Algorithm is the calculation of complex-valued phasors from the real-valued instantaneous voltage and current signals and their subsequent reconstruction to original form. Phasors are calculated by a DFT from which only certain harmonic components are selected. The calculation is continuously updated over a moving window, thereby producing a stream of *dynamic phasor* updates. Different approaches to calculate the phasors and the reconstruction have been considered:

RSCAD: DFT control component

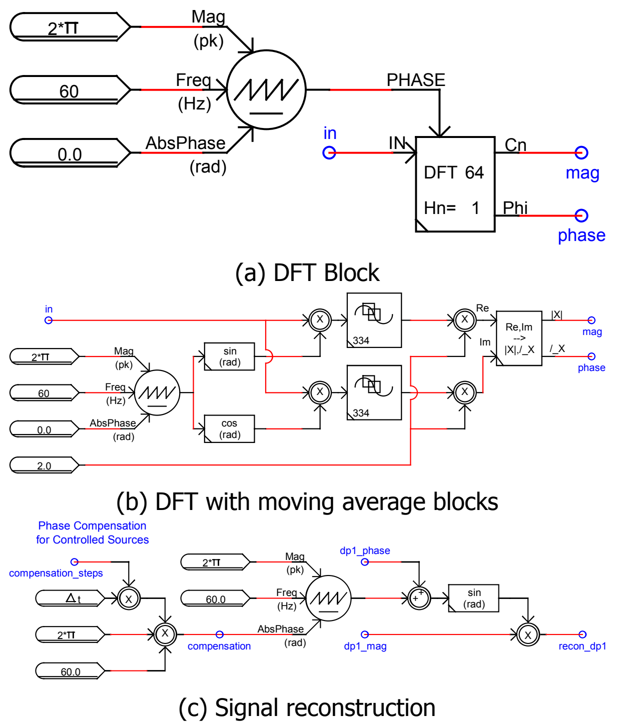
The included DFT block in RSCAD supports a maximum sampling of 64 points per cycle. For the case of a 60 Hz system, this results in a maximum update rate of the phasors to $60\ \text{Hz} \times 64 = 3840\ \text{Hz}$. The execution time required for this block is $0.18\ \mu s \times 64 + 0.5\ \mu s = 12,02\ \mu s$ which is relatively high. In a standard co-simulation, 9 DFT blocks are required (3 phases × 3 harmonic components). Given the execution time, this would result in utilization of several control component processors, which is undesirable.

RSCAD: Moving average window

Figure 4.5b shows a custom implementation of the phasor calculation which utilizes moving average blocks in RSCAD. This implementation produces an updated phasor in every simulation time step and has an execution time which is nearly half of that of the DFT block.

VILLASnode

Both RSCAD based DFT implementations suffer from the disadvantage that, changing the number of harmonic components is not feasible without extensive manual changes to the models and their large utilization of RTDS hardware resources. Therefore, it is desirable to move the calculation of the DP interface algorithm to the VILLASnode gateway and reduce the amount of changes required to the initial model. The only modification, then necessary to adapt a model for co-simulation is the addition of communication blocks (GTNET) and controlled sources. The simulation gateway has been updated to perform calculation and reconstruction of phasors to instantaneous values.



(a) DFT Block

(b) DFT with moving average blocks

(c) Signal reconstruction

Figure 4.5: RSCAD implementation of DP interface algorithm

4.4.3. Signal reconstruction to the time-domain waveform

The reconstruction of the instantaneous voltage/current wave forms is performed by multiplying the dynamic phasor coefficients $X_k[n]$ with the same rotating reference phasor. The resulting signal is then used to directly control a voltage/current source at the coupling point. Due to internals of RTDS scheduling, the resulting reconstructed waveform is always delayed by 1-3 time steps n_c as the control components cannot control the sources in the same time step as they are executed. This demands for a constant phase compensation by $\varphi_c = 2\pi f_0 n_c T_s$ which can be added to the absolute phase of the reference phasor as shown in Figure 4.5c:

$$x[n] = \sum_{k=0}^{K} X_k[n] \cdot e^{j(2\pi f_0 kn + \varphi_c)} \tag{4.2}$$

Table 4.2 shows the steady state power flow with respect to different compensation steps for the voltage and current sources in SS1 and SS2 respectively. It is to be noted that, the apparent power and RMS voltage at the interface are matching in all cases. If internal delays of controlled sources are properly compensated, the power flow is identical for both sides of the interface ($P_{SS1} = P_{SS2}, Q_{SS1} = Q_{SS1}$) which is given for $n_{SS1} = 3, n_{SS2} = 2$.

Table 4.2: Steady-state power flow with respect to phase compensation of source signals.

n_{SS1} $[T_s]$	n_{SS2} $[T_s]$	P_{SS1} [MW]	Q_{SS1} [MVar]	P_{SS2} [MW]	Q_{SS2} [MVar]	S [MVA]	$Vrms$ [kV]
0	0	19.16	9.846	20.0	8.003	21.54	227.7
1	1	19.52	9.118	20.0	8.003	21.54	227.9
2	1	19.69	8.749	20.0	8.003	21.54	227.9
3	2	20.0	8.003	20.0	8.003	21.54	228.1

4.4.4. Update Rate

In steady state, all three approaches provide correct results as the exchanged phasors remain constant. During transients, the update rate of the phasors becomes critical. There are currently two bottlenecks which limit the update rate of the phasors:

1. Firstly, the RSCAD DFT block is limited to 3840 Hz.
2. Secondly, the maximum recommended sending rate by RTDS for the GTNET card is 5 kHz.

The first bottleneck has been solved by using the custom DFT implementation based on moving average windows. The second bottleneck is the current limiting factor and therefore, the simulations in this project have been conducted with an update rate of 5 kHz.

In any case, a sending rate less than the simulation time step ($f_c < \frac{1}{T_s}$) during transients will result in discontinuities in the reconstructed signal. In order to be avoid system instabilities, the signal should be filtered with a low pass filter. Initial tests with a 3rd degree Butter worth filter with $f_c = \frac{1}{2T_s}$ have shown promising results but entail a group delay of the reconstructed signal. However, this delay will add to the existing communication delay.

5 Results and Discussion

The results of JaNDER Implementation and application of VILLAS framework for GDRTS are discussed in this chapter. For both methods, results of the communication tests are presented followed by the results of the simulations.

5.1. JaNDER Implementation

5.1.1. Communication Test Results

A key parameter which must be evaluated for the JaNDER solution is performance : the platform should be "fast enough", meaning that it should be able to support future ERIGrid test cases. However, it should be kept in mind that performance of the JaNDER platform arises from a trade-off between conflicting requirements, in particular:

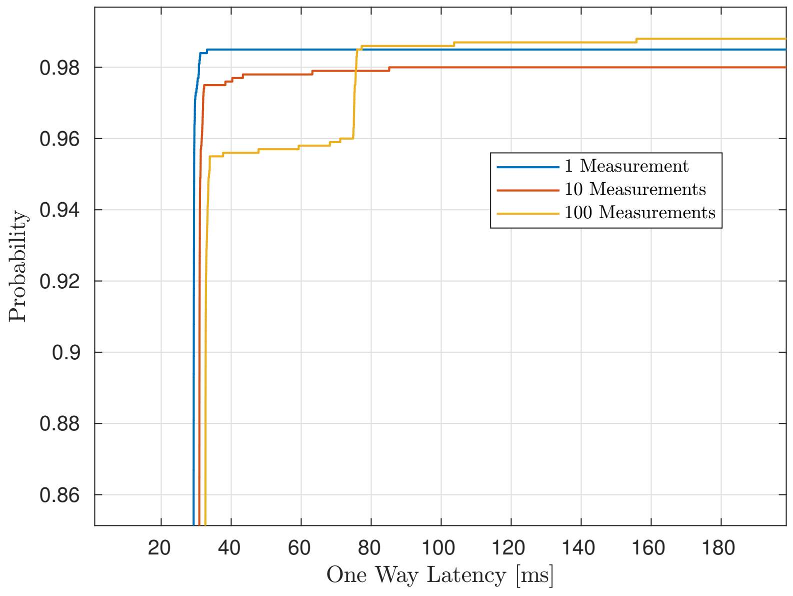
- To guarantee a very high level of cyber security.
- The need to work over the public Internet.
- Be easy to configure and deploy.

Thus, the most important performance factor becomes latency, i.e delays associated with exchange of information or data. To obtain performance which meets the testing needs of ERIGrid, it is crucial to ensure that the appropriate Redis data structure is used in each context (i.e.) plain keys or hashes. In this project, to evaluate latency, simple measurements were conducted at each step of the remote connection, as explained in Chapter 3.

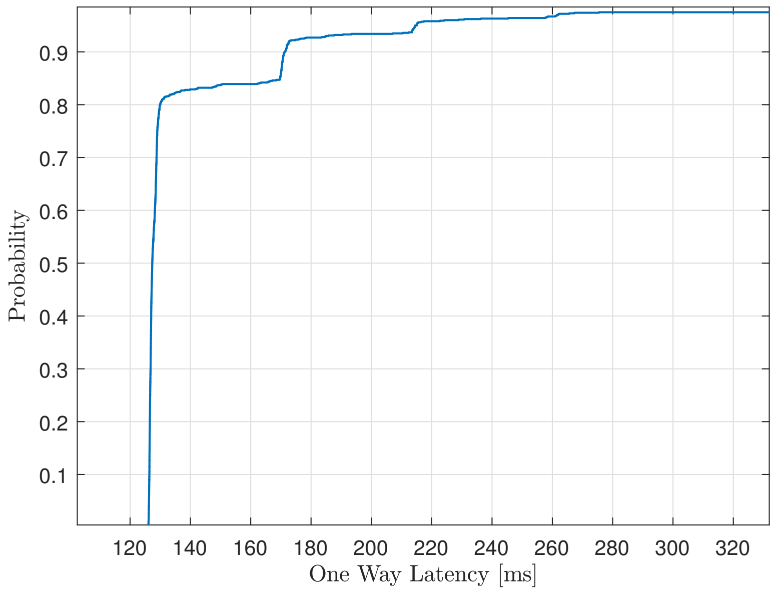
Local to Redis cloud

The most important step in communication testing is investigation of latencies between the local redis and the central redis cloud database. As explained earlier in Chapter 3, redis hashes containing 1, 10, 100 and 1000 measurements were replicated to the cloud. Each of these tests was repeated 1000 times, with the local and cloud redis time-stamps being recorded. The difference in these time-stamps is the latency associated with local to cloud replication.

Figure 5.1a shows the cumulative distribution of the one way delays when replicating hashes with 1, 10 and 100 measurements respectively. It can be observed that, all three cases show similar behaviour with the 95^{th} percentiles for replication of 1, 10 and 100 measurements being 29.20 ms, 30.81 ms and 32.48 ms respectively. Thus, in all these cases, 95% of the values lie below 35 ms. In stark contrast, Figure 5.1b shows the distribution for replication of 1000 measurements. There is a significant increase in delays, with the 95^{th} percentile, now lying at 129.01 ms. This implies, majority of the values are below 130 ms. However, for real-time applications such a large delay is unacceptable. Therefore, number of measurements contained in a hash and replicated during a joint experiment must be limited below 1000.



(a) 1,10 and 100 measurements

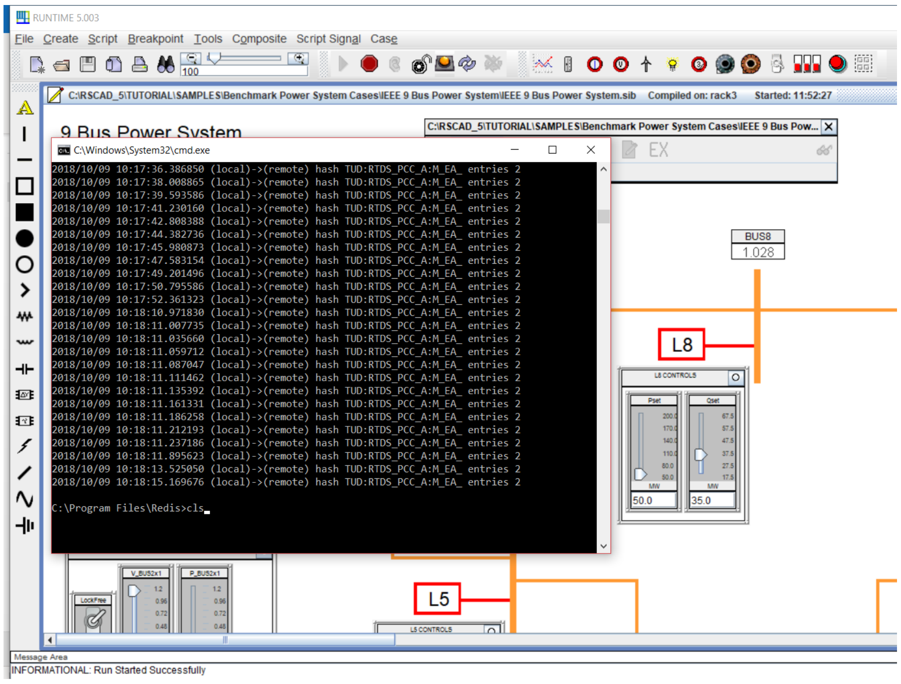


(b) 1000 Measurements

Figure 5.1: Cumulative Distribution of Latencies

5.1.2. Remote Data Exchange using JaNDER

The objective of this step is to show the proof of concept for bi-directional exchange of data between the RTDS at TU Delft and a partner RI using JaNDER level 0. Figures 5.2 shows the working of redis replication and the RTDS simulation. It can be seen from Figure 5.2 that, through the line $local \rightarrow remote$, simulation data stored in local redis as a hash is being replicated to the cloud. It can also be observed that, the slider for controlling active power load is at 50 MW.


RUNTIME 5.003
File Create Script Breakpoint Tools Composite Script Signal Case
C:\RSCAD_5\TUTORIAL\SAMPLES\Benchmark Power System Cases\IEEE 9 Bus Power System\IEEE 9 Bus Power System.sib Compiled on: rack3 Started: 11:52:27
9 Bus Power System
C:\Windows\System32\cmd.exe
BUS8
1.028
L8
L8 CONTROLS
Pset
Qset
50.0
35.0
L5
L5 CONTROLS
V_BUS2x1
P_BUS2x1
LockFree
Message Area

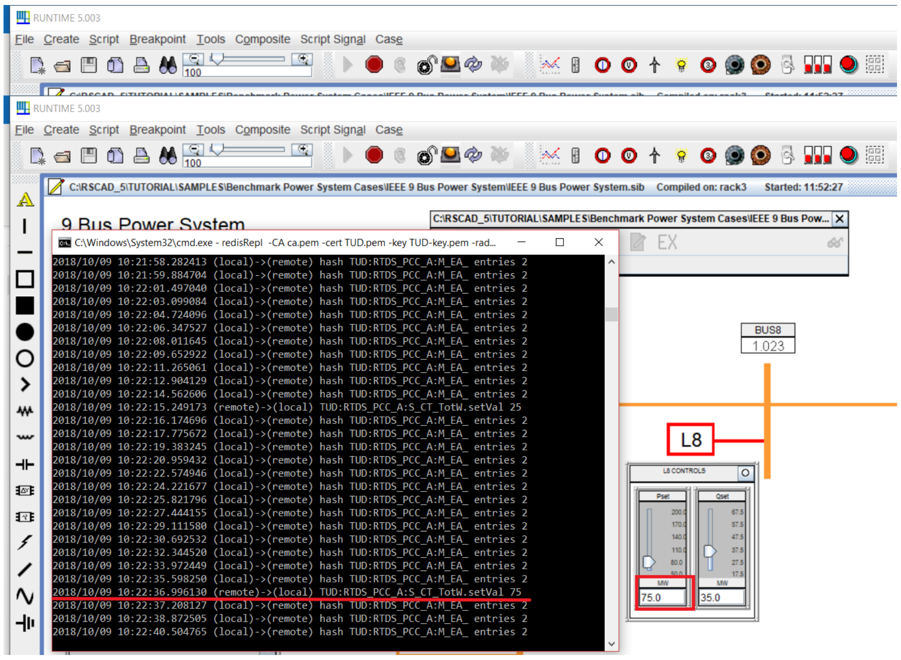
```
2018/10/09 10:17:36.386850 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:38.008865 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:39.593586 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:41.230160 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:42.808388 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:44.382736 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:45.980873 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:47.583154 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:49.201496 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:50.795586 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:17:52.361323 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:10.971830 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.007735 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.035660 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.059712 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.087047 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.111462 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.135392 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.161331 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.186258 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.212193 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.237186 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:11.895623 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:13.525050 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:18:15.169676 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2

C:\Program Files\Redis>cls_
```

```
INFORMATIONAL: Run Started Successfully
```

Figure 5.2: Remote connection test default case

Figure 5.3 depicts the realization of a successful remote-connection. Through the line, $remote \rightarrow local$, the value of the active power set-point for load at bus 8 is modified remotely. This is depicted by the change in slider value on the right from 50 to 75 MW as highlighted in the figure. Thus, remote data exchange using JaNDER is achieved.


RUNTIME 5.003
File Create Script Breakpoint Tools Composite Script Signal Case
C:\RSCAD_5\TUTORIAL\SAMPLES\Benchmark Power System Cases\IEEE 9 Bus Power System\IEEE 9 Bus Power System.sib Compiled on: rack3 Started: 11:52:27
9 Bus Power System
C:\Windows\System32\cmd.exe - redisRepl -CA ca.pem -cert TUD.pem -key TUD-key.pem -rad...
BUS8
1.023
L8
L8 CONTROLS
Pset
Qset
75.0
35.0

```
2018/10/09 10:21:58.282413 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:21:59.884704 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:01.497040 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:03.099084 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:04.724096 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:06.347527 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:08.011645 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:09.652922 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:11.265061 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:12.904129 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:14.562606 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:15.249173 (remote)->(local) TUD:RTDS_PCC_A:S_CT_TotW.setVal 25
2018/10/09 10:22:16.174696 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:17.775672 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:19.383245 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:20.959432 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:22.574946 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:24.221677 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:25.821796 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:27.444155 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:29.111580 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:30.692532 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:32.344520 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:33.972449 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:35.598250 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:36.996130 (remote)->(local) TUD:RTDS_PCC_A:S_CT_TotW.setVal 75
2018/10/09 10:22:37.208127 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:38.872505 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
2018/10/09 10:22:40.504765 (local)->(remote) hash TUD:RTDS_PCC_A:M_EA_ entries 2
```

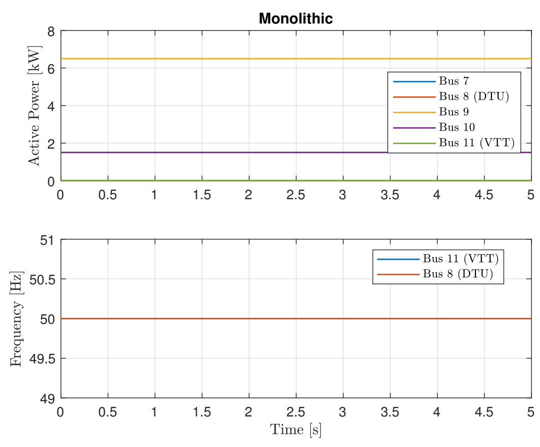
Figure 5.3: Virtual connection with remote changes

5.1.3. Simulation Test Results

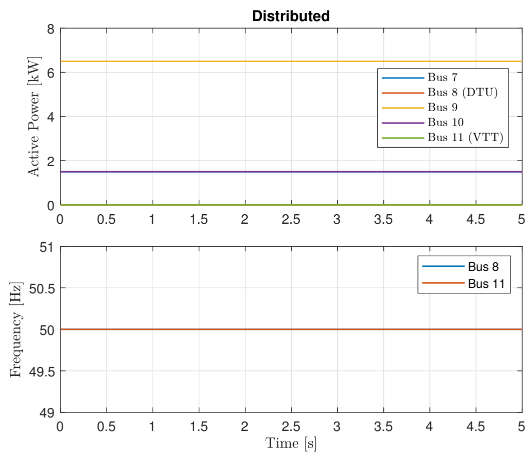
In order to test and validate the geographically distributed, virtually coupled power system, described in chapter 3, it is also modeled as a monolithic reference case in the real-time simulator at TU Delft for comparison. Similar events/scenarios are simulated for both, monolithic and distributed setups.

Steady State Behaviour

Figures 5.4a and 5.5a show the steady state performance of the monolithic model. Since the system is in steady-state, there are no deviations or changes to system parameters. The bus voltages and frequencies are maintained at their nominal values. It is to be also noted that, all the loads kept at zero. The same is observed for the distributed setup as seen through Figures 5.4b and 5.5b. In order to study the quasi-static behaviour of the system, the loads connected to buses 8 and 11 which represent the virtual PCC in the distributed system are changed i.e increased or decreased. The resulting changes to bus voltages and frequencies are observed for both the studied cases. This is explained in the following sections.

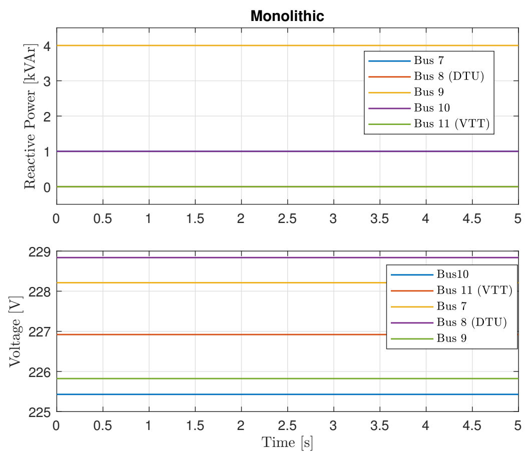


(a) Monolithic

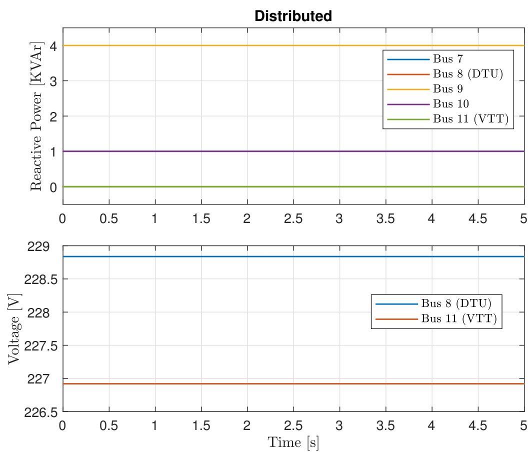


(b) Distributed

Figure 5.4: Active Power and Frequency: Steady State



(a) Monolithic

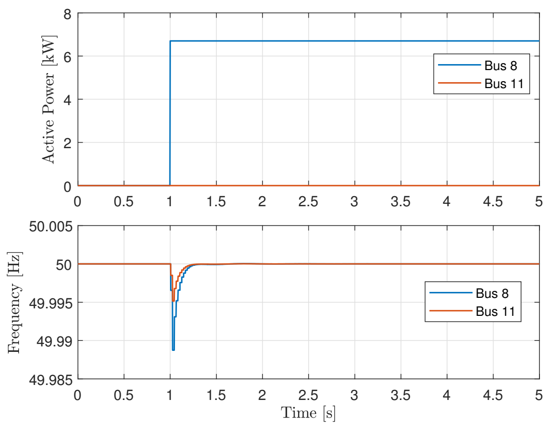


(b) Distributed

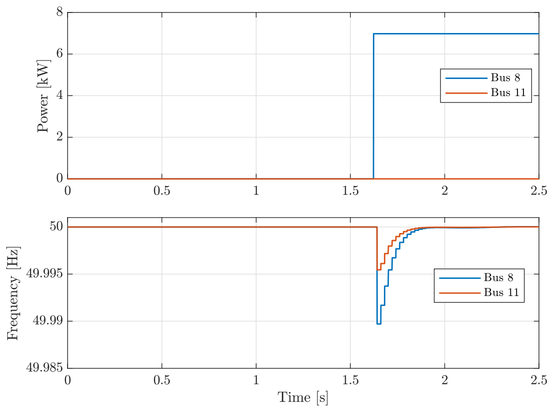
Figure 5.5: Bus Voltages: Steady State

Load Change at Bus 8

To note system performance when subjected to a change, the active and reactive power of the load at Bus 8 (DTU) are increased to 6.7 kW and 10 kVAr respectively. These values correspond to three actual load steps in the controllable load-bank located at DTU in Denmark. The same load step is performed in the monolithic simulation as well. Due to the sudden increase in load demand, a frequency drop is observed from 50 Hz to 49.9 Hz. However, since, the simulated network is a distribution grid, with an ideal voltage source, the system frequency quickly recovers to the nominal value of 50 Hz. Similarly, there is a drop in bus voltages with the maximum effect observed for bus 8 where the load step occurs. In both cases, a voltage drop of about 6 V occurs. The post event voltages settle to a new steady-state value, slightly lower than the pre-event or base values. This can be seen in Figures 5.7a and 5.7b. It is observable from Figures 5.6 and 5.7 that the behaviour of the monolithic and distributed cases is similar.

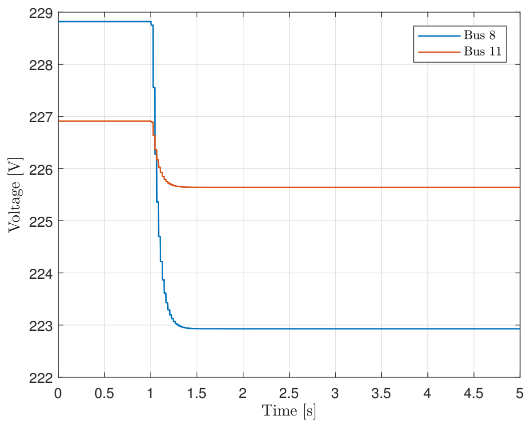


(a) Monolithic

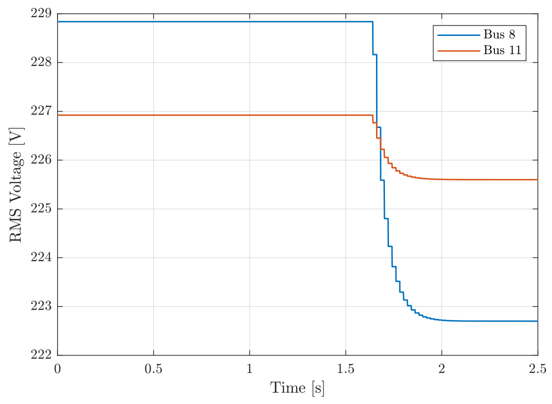


(b) Distributed

Figure 5.6: Active Power and Frequency: Load Step at Bus 8 (DTU)



(a) Monolithic

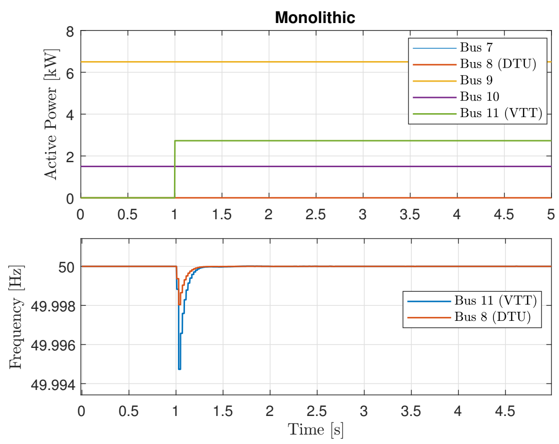


(b) Distributed

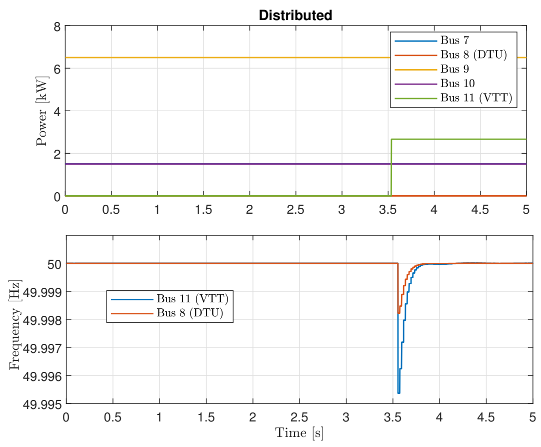
Figure 5.7: Bus Voltages: Load Step at Bus 8 (DTU)

Load Change at Bus 11

The controllable load used at VTT as part of the joint experiments is purely resistive in nature. One load step of this load corresponds to an increase by 2.73 kW. Hence, an increase in active power from 0 to 2.73 kW is simulated at $t = 1s$ in the monolithic model, while the load step event in the distributed setup is triggered at $t = 3.5s$. The frequency nadirs observed in both cases match quite well, with recovery time for both cases being around 600 ms. Due to the increase in load, a voltage drop also occurs. This is shown in Figure 5.9, which highlights the similar behaviour of the distributed system with respect to monolithic case; The voltage drop in both cases is 3 V. It is interesting to note, there is a very small voltage dip of about 0.5 V at Bus 8. This shows that despite being geographically separated, the virtually coupled system behaves similar to an actual distribution grid.

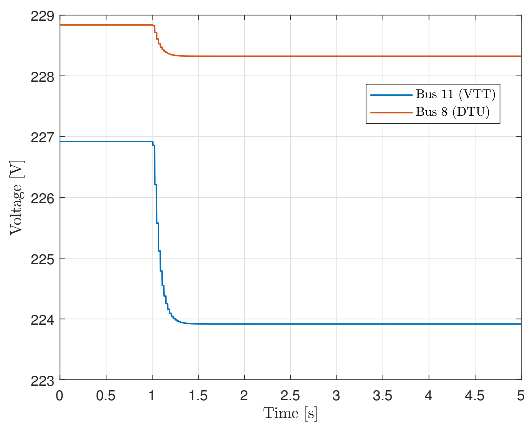


(a) Monolithic

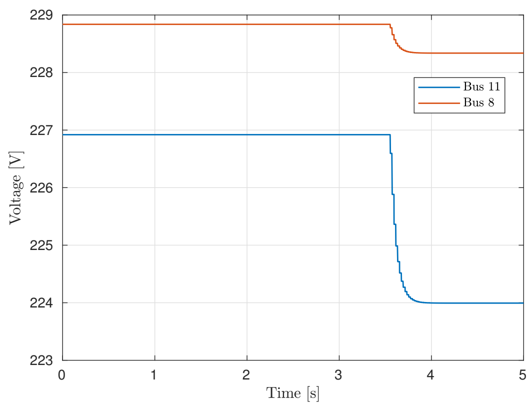


(b) Distributed

Figure 5.8: Active Power and Frequency: Load Step at Bus 11 (VTT)



(a) Monolithic



(b) Distributed

Figure 5.9: Bus Voltages: Load Step at Bus 11 (VTT/PCC B)

5.2. VILLAS Framework

This section delves into the results obtained through the application of VILLAS framework. The communication and co-simulation test results are presented and discussed.

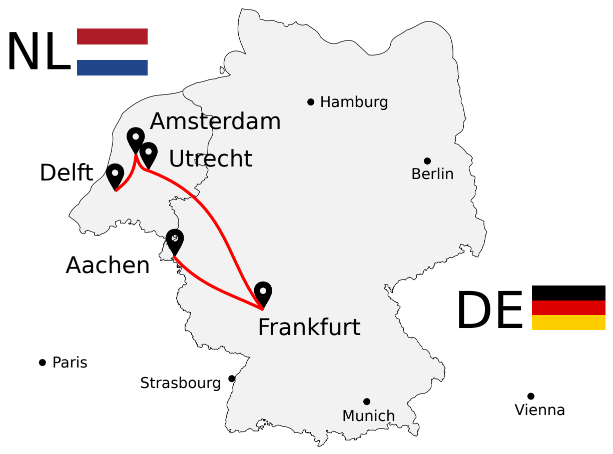
5.2.1. Communication Tests

Trace Route and Ping Tests

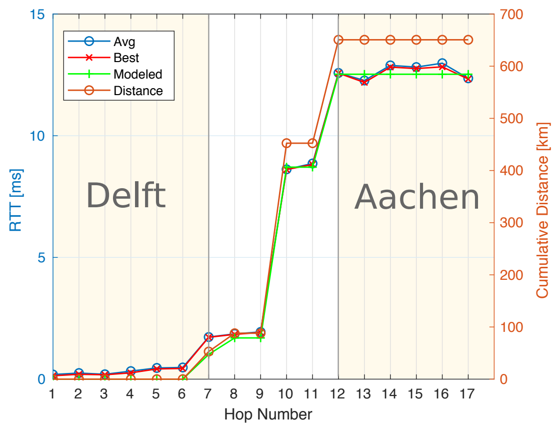
Routing path of packets between Delft and Aachen is shown in Figure 5.10a. A total of seventeen hops are present. Statistical analysis of all individual hops is shown in Figure 5.10b. It can be observed that, there is a positive correlation between the actual physical distance between hops and the modeled latency, calculated using:

$$T_m = a * D * (c/n) \tag{5.1}$$

where, $a \approx 2$ is an empirical air-line distance correction factor, D as the distance, c as speed of light and $n \approx 1,5$ as the refractive index of a fiber optic.



(a) Geographic Map of Trace Route



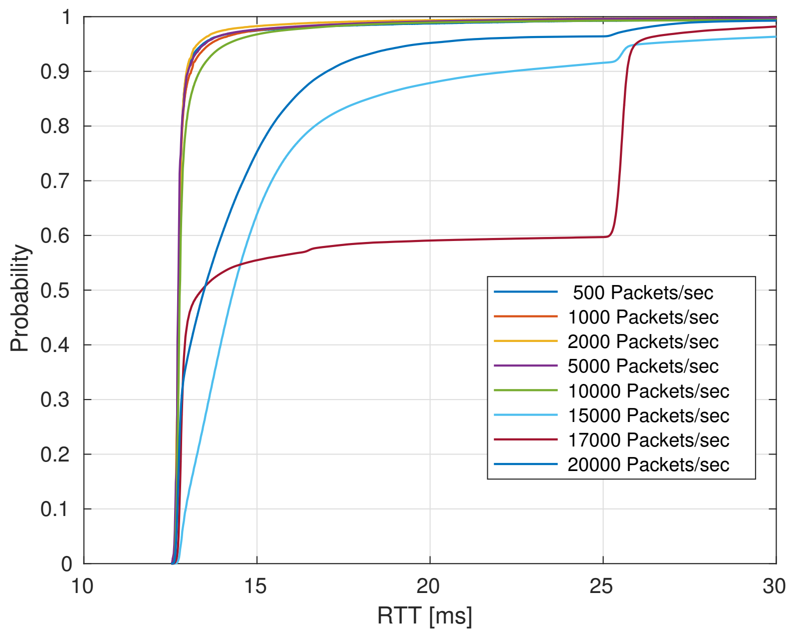
(b) Latency of routing path

Figure 5.10: Ping and trace-route results

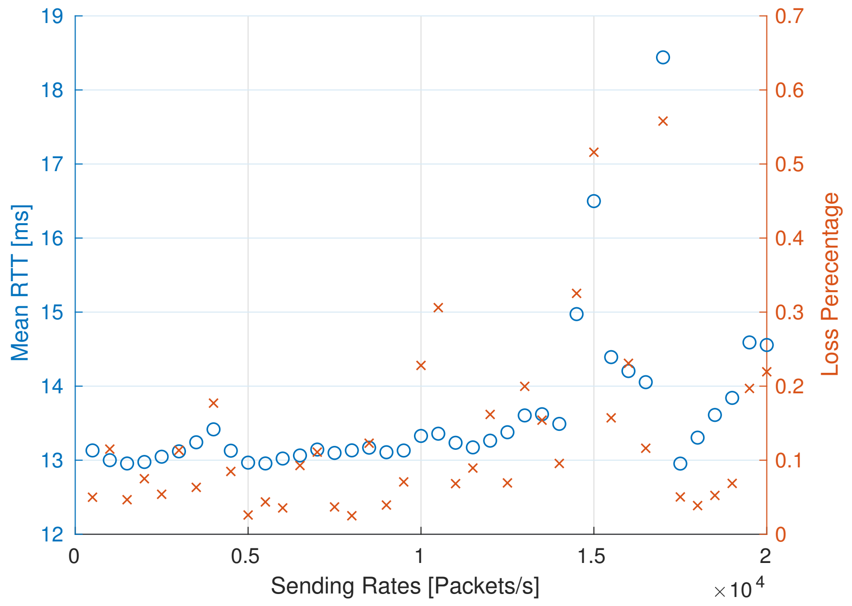
Between VILLASnode Gateway Machines

Figure 5.11a shows the cumulative distribution of Round Trip Times (RTT) for the transfer of a fixed number of data points–24 with variable sending rates between the gateway machines. 24 data points are chosen as they are relevant to the application/use-case. It can be inferred from Figure 5.11b that, for sending rates greater than 10000 packets/sec, RTT is higher than average and they can be treated as outliers. Further, with higher sending rates, average RTT and packet loss also increase, which is undesirable for the case of real-time simulations. For instance, with a sending rate of 17000 packets/sec, there is only a 60% chance of the RTT being lesser or equal to 25 ms. However, the lower sending rates are all tightly coupled with their RTT lying in the range of 12.3 to 13 ms.

A similar analysis is carried out for RTT measurements with a fixed sending rate of 2000 packets/sec and variable number of data points in each packet. Even here, when number of data points transferred is greater than 100, there is a pronounced increase in RTT and packet loss. This is clearly seen through Figure 5.12. Thus, it can be concluded, neither higher sending rates nor large data sizes are preferable. This can be attributed to the fact that, the communication network between the two sites is not a dedicated one, but shared. Hence, this introduces a degree of unpredictability or randomness. Therefore, to avoid congestion and guarantee soft real-time behaviour even on a shared connection, the sending rate and number of data points transferred should be kept below 10000 packets/s and 100 respectively.

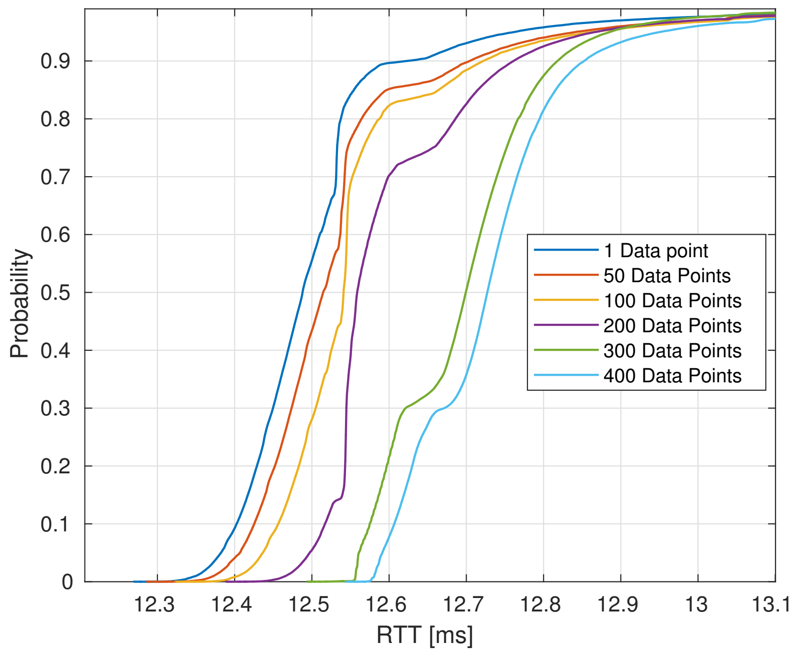


(a) Cumulative Distribution of RTT for variable sending rates

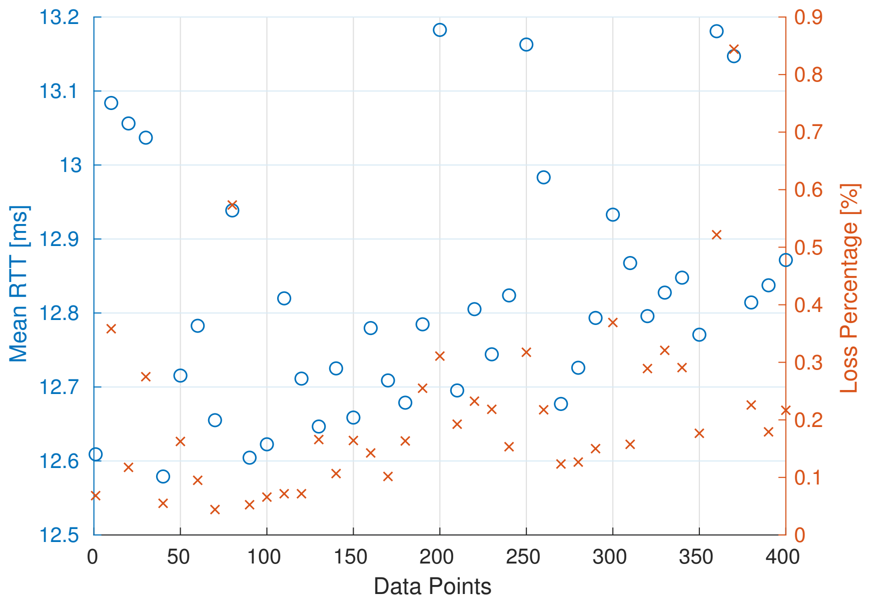


(b) Mean RTT and packet loss

Figure 5.11: RTT statistics for a fixed data size and variable sending rate



(a) Cumulative Distribution of RTT for variable data sizes

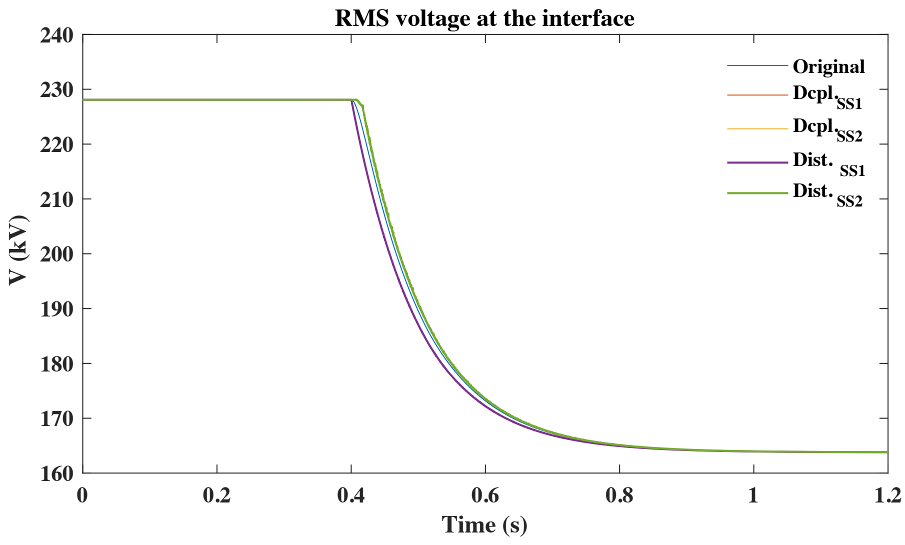


(b) Mean RTT and packet loss

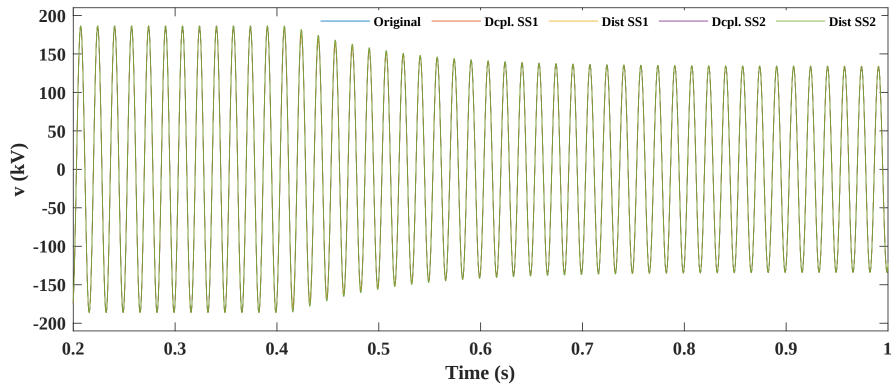
Figure 5.12: RTT statistics for a fixed sending rate and variable data size

5.2.2. Validation of the Co-simulation Interface

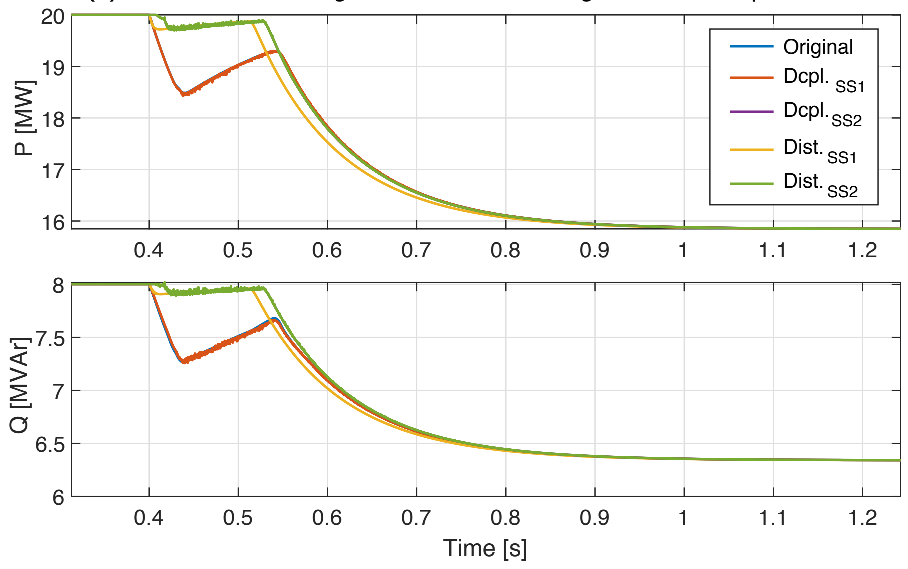
In order to validate the co-simulation interface, outputs from the distributed simulation are compared to the monolithic case. To ensure simulation fidelity, values on either side of the interface must be similar. To test and validate the robustness of the co-simulation interface, the amplitude of the ideal voltage source is decreased from 230 kV to 165 kV as shown in Figure 5.13a. Both, the decoupled and distributed simulation models' RMS voltages on either side of the interface are similar to the monolithic/original model. The results of the distributed model, if compared against the original monolithic model as well as the decoupled version, show only a slight delay of 6 ms. This delay is attributable to the one way communication latency between both the labs. The same holds true for instantaneous voltages and power quantities on both sides of the interface observable in Figures 5.13c and 5.13b.



(a) RMS Voltage at interface for change of source amplitude.



(b) Instantaneous voltage at interface for change of source amplitude



(c) Active / Reactive power flow at interface for change of source amplitude

Figure 5.13: Quantities at the interface

5.2.3. Limitations of the co-simulation interface

To test the limits of the co-simulation interface algorithm, a 90° phase discontinuity of the ideal voltage source is introduced. This causes a transient in the monolithic simulation model, before the system returns to a steady-state. However, the distributed and decoupled system's response to the same event is highly different. This is shown in Figure 5.14. The interface quantities do not change and strongly differ from the monolithic reference waveforms. In the case of the simple and stiff power system which is used in this study, the system eventually returns to a steady state. Thus, there is a strong case for investigation of this improved co-simulation interface when applied to more complex power system test cases.

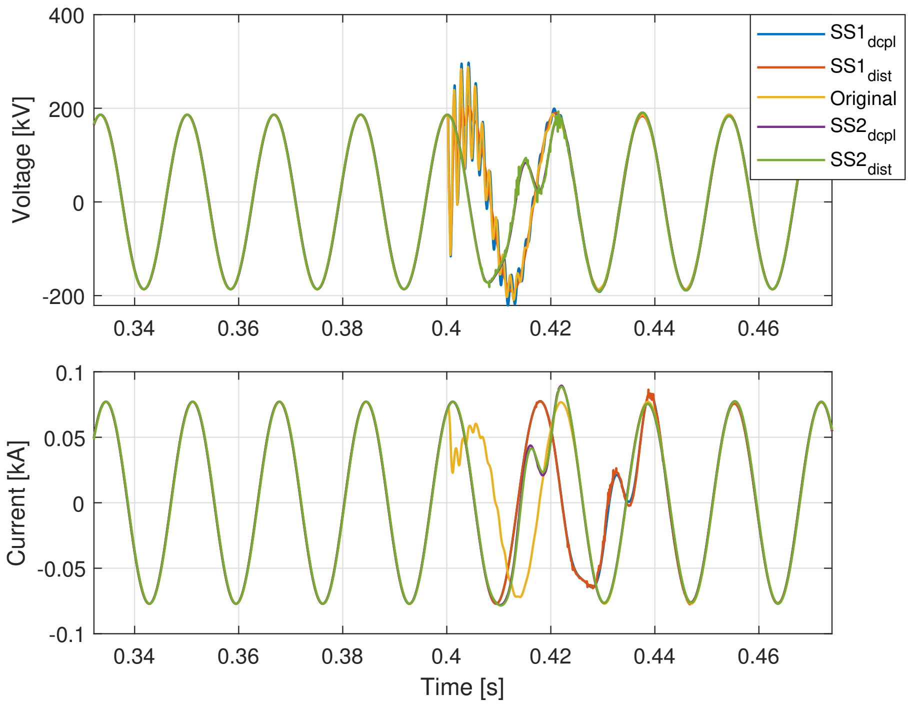


Figure 5.14: Limits of DP-IA interface in case of 90° phase shift

6 Conclusions and Recommendations

This chapter summarizes and concludes the scientific and technical implications of the research findings of this thesis in considerable detail. The answers to the research questions posed in Chapter 1 are presented along with final concluding remarks and specific contributions of this thesis. Last but not least, challenges faced and recommendations for future work are also discussed.

6.1. Answers to Research Questions

1. **How to interconnect real-time grid simulator at TU Delft with power hardware at other European labs to build a Virtual Research Infrastructure (VRI)?**
   The RTDS at TU Delft has been successfully interconnected with hardware at other European labs using the JaNDER platform. This is a software solution to interconnect multiple labs over large geographically distances through the internet.

2. **How to interconnect two real-time simulators to perform Geographically Distributed Real-Time Simulation (GDRTS)?**
   The real-time simulators at RWTH Aachen and TU Delft have been successfully interconnected through the VILLAS Framework. This interconnection was used to perform Geographically Distributed Real-Time Simulation (GDRTS); Thus, extending the simulation capabilities of both labs.

3. **What is the effect of the properties and physical limitations of the communication channel (internet) on a networked, distributed power system?**

   - Using the JaNDER architecture, only *static* experiments can be performed. The term *static* refers to the limitation of being able to only analyze steady state performance. This is due to the data exchange method between the RIs, which is cloud based and centralized. It is known that the internet is not deterministic for networked control systems[48]. Consequently, the latency between the local Redis of each RI can change as a function of many uncontrollable variables.
   - The average one way delay between local and cloud redis for transfer of 10 measurements at one RI is 30 ms. Extending this, the overall RTT for a complete loop-back between two RIs would be $30 \times 4 = 120ms$, which is quite high. Hence, the dynamic behaviour of an electrical system cannot be accurately emulated using this approach.

4. **What is the effect of the properties and physical limitations of the communication channel (internet) on a co-simulation interface algorithm for GDRTS?**

   - The communication link between RWTH Aachen and TU Delft is quite good due to routing of packets over the dedicated National Research and Education Networks (NREN). The average RTT is in the range of 12.5 to 12.7 ms, which is quite low. Detailed testing of this particular link has also revealed that, to avoid congestion and ensure simulation fidelity for a GDRTS, the sending rate and number of data points transferred between the labs should be kept below 10000 packets/s and 100 respectively.

6.1.1. Conclusions on JaNDER Implementation and Testing

- The proof of concept for a geographically distributed, virtually coupled power system is presented and its quasi static steady state performance is validated. This enables the realization of the Virtual Research Infrastructure (VRI) concept, extending the individual capabilities of laboratories.
- The JaNDER Level 0 one way latency is approximately identical for transfer of 1, 10, 100 measurements (29, 30 and 32 ms respectively) to the cloud. However, it is about 129 ms for 1000 measurements. Thus, JaNDER Level 0 is unsuitable in situations where more than 1000 measurements have to be exchanged.
- For replication of measurements, it is better to use hashes since they are updated periodically in a bulk manner. On the other hand, plain keys are more suitable for control variables since these are typically transmitted asynchronously.
- The results obtained from the monolithic simulation and distributed test case match quite well. The system response of the distributed setup to load change events is similar to that of the monolithic case and inline with theory. There are some small deviations, which are to be expected, as the three participating RIs are geographically dispersed and no delay compensation or control was applied.

6.1.2. Conclusions on Application of VILLAS framework for GDRTS

This thesis identified and addressed several issues in the co-simulation interface which has been used so far for GDRTS. These issues have been solved and validated, using a distributed real-time simulation of a simple power system.

- Significant differences in Quality of Service (QoS) of the communication link have been observed in comparison to previous lab couplings. These demand for an automated approach to monitor the network and to adapt to network congestion. Therefore, this project introduced the application of RTP/RTCP protocol in the real-time simulation domain.
- In the current state, GTNET cards cannot achieve the required sending rate, to sample data at every simulation time step. This is necessary to move the Co-simulation Interface Algorithm into the VILLASnode gateway machine. This is a hardware limitation which can be overcome by use of a GTFPGA unit.
- The Discrete Fourier Transform (DFT) window lengths do not always match with the fundamental system period due to the usage of discrete time steps by the simulators. This error has been successfully reduced by adjusting the simulation time step to an integral factor of the time period.
- A constant phase offset caused by scheduling dependencies in the control system and network solution within the RTDS system has been identified. This has been solved by applying a constant phase compensation to the reconstructed signals.

6.2. Contributions

- This thesis has demonstrated and validated the concept of Virtual Research Infrastructure (VRI) through a multi RI experiment. It was achieved by interconnecting geographically dispersed hardware and software assets using the JaNDER level 0 platform. The platform can be used for similar experiments or projects in the future, involving TU Delft.
- Study and validation of an updated co-simulation Interface Algorithm (IA) for Geographically Distributed Real-Time Simulation (GDRTS) by means of the VILLASframework. Hence, TU Delft is now a Point-of-Presence for Virtually Interconnected Laboratories for Large Scale Simulation (VILLAS).
- Another important contribution of this work is the development and release of a reusable library component and its demonstration in a demo model for the RTDS' RSCAD software. It simplifies the procedure to adapt existing models for a GDRTS as the user can simply copy a library block into their models. This thesis also presented the improvements made in simulation fidelity as well as usability for establishing future simulator and laboratory connections using Dynamic Phasors (DP). This is shown in Appendix B.1.

- The research findings of this thesis have been submitted to a special session on *Smart Automation and Digital Control in Power and Energy Systems* at the 45^{th} Annual Conference of the IEEE Industrial Electronics Society (IECON'2019).

6.3. Challenges Faced

- JaNDER Implementation
  - Resolving basic issues related to signal names, formats, units etc. as part of the JaNDER tests took a surprising amount of time.
  - Due to involvement of multiple RIs with different local hardware and software setups, testing and experimentation was slow.
  - Time Synchronization between RIs was an issue.
  - Different data logging formats were time consuming to process.
- VILLAS Framework Implementation
  - Obtaining intricate details related to existing network topology at TU Delft.
  - Wiring and connection changes to the RTDS setup had to be made to ensure a best-effort connection with the VILLASnode gateway machine.

6.4. Recommendations on Future Work

- The improved Dynamic Phasors (DP) based co-simulation Interface Algorithm must be tested for more complex scenarios/models. Therefore, as follow-up work, a complex GDRTS is planned, composed of a transmission (IEEE 9-bus system) and a distribution system (IEEE 34-node test feeder), to be implemented among three real-time simulation laboratories at RWTH Aachen University, Technical University of Denmark (DTU) and TU Delft.
- To add an additional layer of complexity to the JaNDER testing, a control algorithm such as the Coordinated Voltage Control (CVC) applied in [42] can be implemented, to control remote resources in the distributed setup.
- A known hardware limitation in the RTDS hardware is sending/update rate of the GTNET card. The recommended limit of 5 kHz cannot sample data at every simulation time-step. In future tests, GTFPGA cards could be used to work around this issue and obtain more accurate results.

A

JaNDER Implementation Details

A.1. Remote connection test

1. **Title of Test:** Remote Connection of different RI

2. **Test Rationale:** The concept of a Virtual Research Infrastructure (VRI) requires a soft-HIL approach, so there is a "main-RI" that sends the voltage and frequency setpoints to all the other "sub-RIs" virtually connected. All the RIs sends the active and reactive power and the voltage measurements to the RI with the grid simulator. This test aims to verify the remote connection between the RIs.

3. **Specific Test System:** The Test System consists of all RI SCADA/DAQ systems (at least the variables of the power converters, or similar devices, that allows the coupling between the RIs) through JaNDER infrastructure.

4. **Target measures:**

   - Two way data exchange between RIs
   - Data can be deciphered and read in the RIs
   - Divergence between end-to-end values within uncertainty threshold: error between the voltage and frequency setpoint sent by the main-RI and the corresponding measurements at the virtual connection node in the sub-RI; error between the active and reactive setpoints sent by the sub-RI and the corresponding measurements at the main-RI.

5. **Test Design:**

   - Verify the operation mode of each device dedicated to the virtual connection: in the main-RI all the devices at the connection nodes must be in grid-feeding mode (current source) while in all the sub-RI, the devices at the connection nodes must be in grid-forming mode (voltage source).
   - Set a grid configuration of the main-RI in order to have different voltages measurements at the connection nodes with the sub-RIs.
   - Read the measured voltages and frequencies in all the sub-RIs.
   - Evaluate the error between the setpoints and measurements of voltage and frequency in all the connection nodes.
   - Repeat the point 2 and 3 also for the active and reactive power values.

A.2. Quasi Static Software based HIL

1. **Title of Test:** Quasi Static Software based HIL

2. **Test Rationale:** This is the second step of implementation where a virtual electrical connection is established by exchange of data set-points. Voltage and frequency from RI 1 and power setpoints from RI 2 and 3. The data exchange is independent of any control algorithm to observe system behaviour, without any control
3. **Specific Test System:** A test grid is created which spans multiple RIs i.e. some lines and resources in each RI. The test grid is also modelled in a power flow simulator as a monolithic model, without any split.
4. **Domain under Investigation (DuI):** Electric power and ICT domains.
5. **Purpose of Investigation (PoI):**
   - Integrating real physical system behaviour into simulated grids.
   - Proof of concept of soft-real time multi-RI experiment through JaNDER level 0.
   - To characterize the difference in results obtained through experiment carried out within one RI and experiments carried out through a multi RI setup.
6. **Target measures:**
   - Deviations/errors in the multi-RI setup with respect to the monolithic model.
7. **Test Design:**
   - Define data to exchange and corresponding convention.
   - Establish RIs communication through JaNDER-L0.
   - Plan the experimental steps.
   - Run the experiment:
     (a) Load change in DTU
     (b) Load change in VTT
     (c) Voltage change in TUD
   - For each test, transient and steady state values are recorded.

A.3. Script to Interface RTDS through JaNDER

```python
import socket
import struct
import redis
import time
import math


r=redis.Redis('localhost')


dtu=["DTU::Storage_Vanadium:M_EA_Watt","DTU::Storage_Vanadium:M_EA_Var","DTU::
    Storage_Vanadium:S_EA_PPV",
"DTU::Storage_Vanadium:S_EA_Hz"]

vtt=["VTT:Source_JK-T.KK-TRB.emulator:M_EA_W.phsA","VTT:Source_JK-T.KK-TRB.emulator:M_EA_VAr.
    phsA",
"VTT:Source_JK-T.KK-TRB.emulator:C_EA_PhV.phsA.ctlVal","VTT:Source_JK-T.KK-TRB.emulator:
    C_EA_Hz.phsA.ctlVal"]

tud=["TUD:RTDS_PCC_A:Meas","TUD:RTDS_PCC_B:Meas"]


UDP_IP = "10.10.9.1"
UDP_PORT = 7777
gtnet=("10.10.9.2",7777)

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM) # UDP
```

```
sock.bind((UDP_IP, UDP_PORT))


while 1:
  try:
    #send data

      data,(UDP_IP,UDP_PORT)= sock.recvfrom(1024) # buffer size is 1024 bytes
      array = bytearray(data)
      value_rec= struct.unpack('>4f',array) #read from RTDS GTNET and decode
      #print(value_rec)


      #set to local redis
      r.hmset(tud[0], {'voltage':(value_rec[2]*1000), 'freq':(value_rec[0]) ,'tstamp':str(
    time.time())})
      r.hmset(tud[1], {'voltage':(value_rec[3]*1000), 'freq':(value_rec[1]) ,'tstamp':str(
    time.time())})

      #set values at DTU
      r.hmset(dtu[2],{'instMag':(value_rec[2]*1000*math.sqrt(3))})
      r.hmset(dtu[3],{'instMag':value_rec[0]})

      r.set(vtt[2],value_rec[3]*1000)
      r.set(vtt[3],value_rec[1])

      p1=(float(r.hget(dtu[0],'instMag').decode())/1000)
      q1=(float(r.hget(dtu[1],'instMag').decode())/1000)
      p2=(float(r.hget(vtt[0],'VTT:Source_JK-T.KK-TRB.emulator:M_EA_W.phsA.instMag').decode()
    )/1000)

      q2=(float(r.hget(vtt[1],'VTT:Source_JK-T.KK-TRB.emulator:M_EA_VAr.phsA.instMag').decode
    ())/1000)

      if(p1>15.0):
        p1=15

      if(p1<0):
        p1=0

      if(q1>10):
        q1=10

      if(q1<0):
        q1=0

      if(p2>6.5):
        p2=6.5

      if(p2<0):
        p2=0

      if(q2>4):
        q2=0

      data_to_rtds=struct.pack('>4f',p1,q1,p2,q2)
      sock.sendto(data_to_rtds,gtnet)


  except (KeyboardInterrupt,RuntimeError, TypeError, NameError):
    print("bye")
    break
    #r.flushall()
    sock.close()
```

Listing A.1: Python Code to communicate between RTDS and redis

A.4. RSCAD and RTDS Models

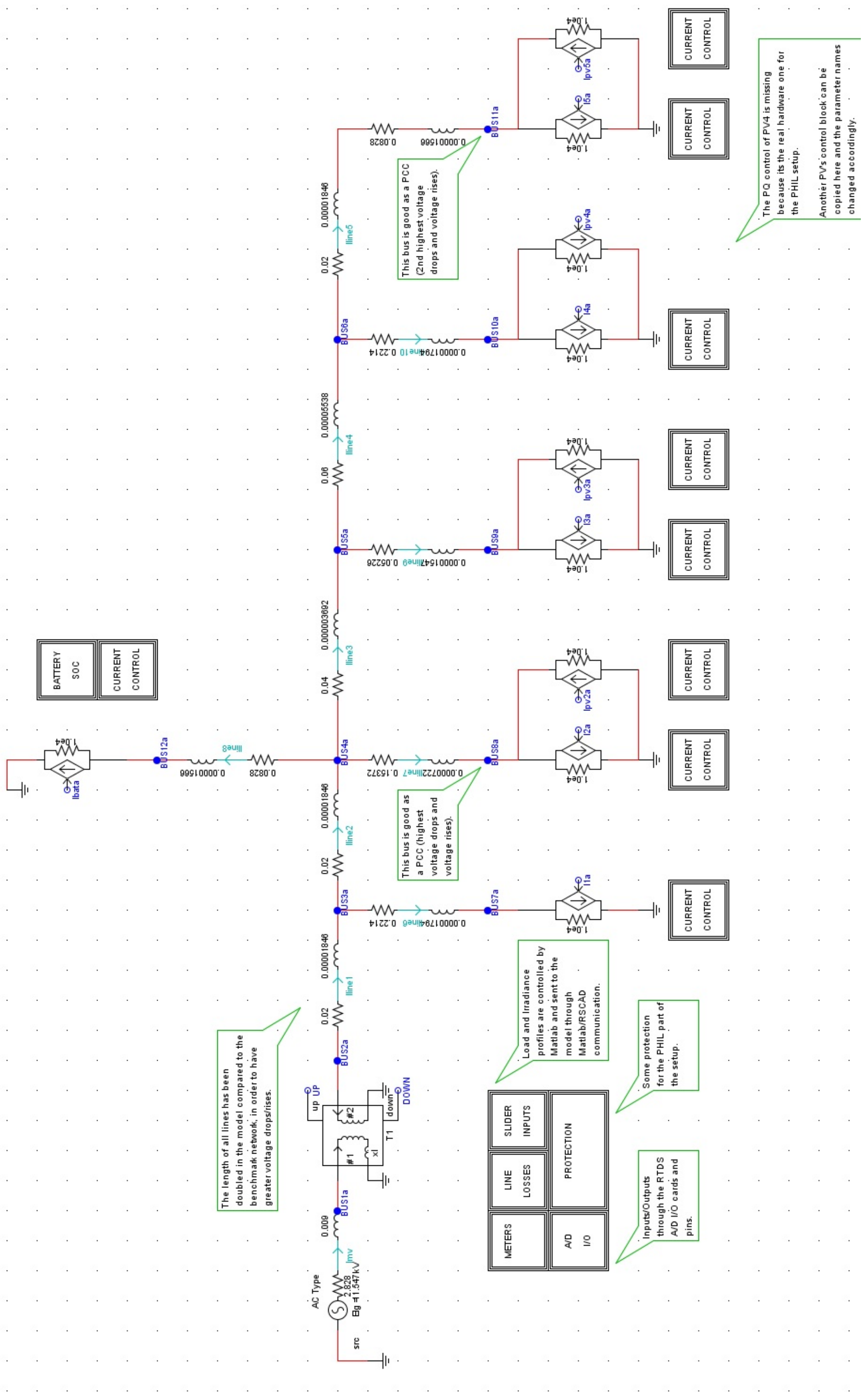

This bus is good as a PCC (2nd highest voltage drops and voltage rises).
The PQ control of PV4 is missing because its the real hardware one for the PHIL setup.
Another PV's control block can be copied here and the parameter names changed accordingly.
BATTERY SOC
CURRENT CONTROL
This bus is good as a PCC (highest voltage drops and voltage rises).
The length of all lines has been doubled in the model compared to the benchmark network, in order to have greater voltage drops/rises.
Load and Irradiance profiles are controlled by Matlab and sent to the model through Matlab/RSCAD communication.
Some protection for the PHIL part of the setup.
Inputs/Outputs through the RTDS A/D I/O cards and pins.
METERS
LINE LOSSES
SLIDER INPUTS
A/D I/O
PROTECTION
AC Type
src
BUS1a
BUS2a
BUS3a
BUS4a
BUS5a
BUS6a
BUS7a
BUS8a
BUS9a
BUS10a
BUS11a
BUS12a
T1
UP
DOWN
Ibata
I1a
I2a
I3a
I4a
I5a
Ipv2a
Ipv3a
Ipv4a
Ipv5a
Iline1
Iline2
Iline3
Iline4
Iline5
Iline6
Iline7
Iline8
Iline9
Iline10
1.0e4

Figure A.1: Simulated Network on RTDS

A.5. Network Data

Table A.1: Component data

Cable/Line	Length [m]	RI	Real/Virtual	R [Ω]	X[Ω]
Z1.1	70	TUD RTDS	Virtual	0.02	1.846E-05
Z1.2	40	TUD RTDS	Virtual	0.02	1.846E-05
Z1.3	40	TUD RTDS	Virtual	0.04	3.692E-05
Z1.4	40	TUD RTDS	Virtual	0.06	5.538E-05
Z1.5	40	TUD RTDS	Virtual	0.08	1.56E-05
Z1.6	40	TUD RTDS	Virtual	0.2214	1.794E-05
Z1.7	60	TUD RTDS	Virtual	0.15372	7.22E-05
Z1.8	60	TUD RTDS	Virtual	0.05226	1.547E-05
Z1.9	60	TUD RTDS	Virtual	0.05226	1.547E-05
Z2.1	700	DTU	Real	0.0854	0.0539
Z2.2	350	DTU	Real	0.10955	0.0273
Z2.3	25	DTU	Real	0.00783	0.00195
Z3.1	50	VTT	Real	0.008	0.004

Table A.2: Load Data

Bus	Max Active Power [kW]	Max Reactive Power [kVar]
7	1.5	1
8 (DTU)	15	10
9	6.5	4
10	1.5	1
11 (VTT)	6.5	4

A.6. Additional Results

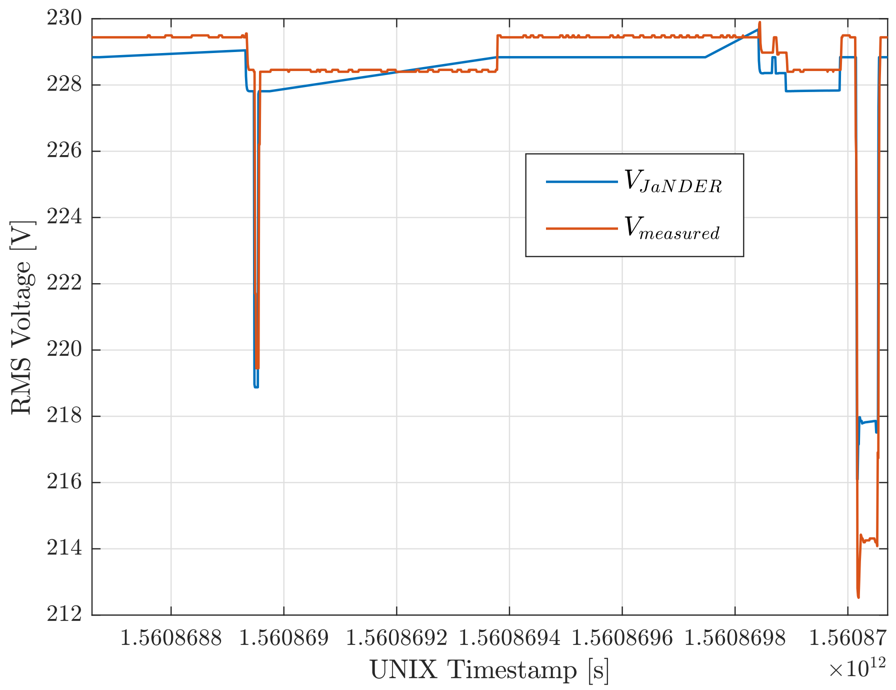


Figure A.2: Comparison of Voltages at DTU

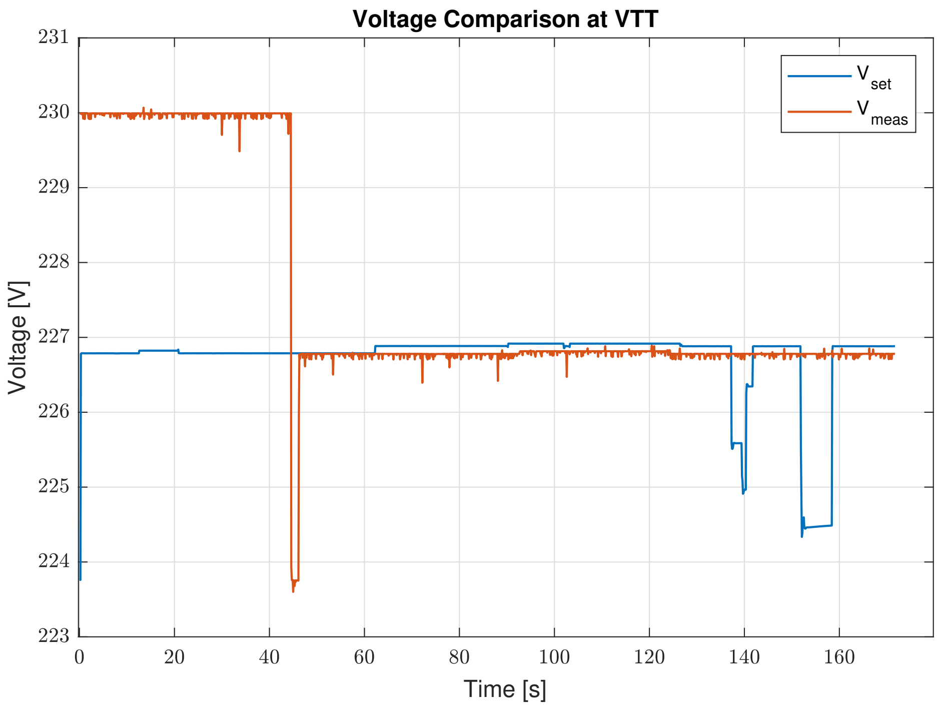


Figure A.3: Comparison of Voltages at VTT

B

Application of VILLAS Framework

B.1. RTDS Models

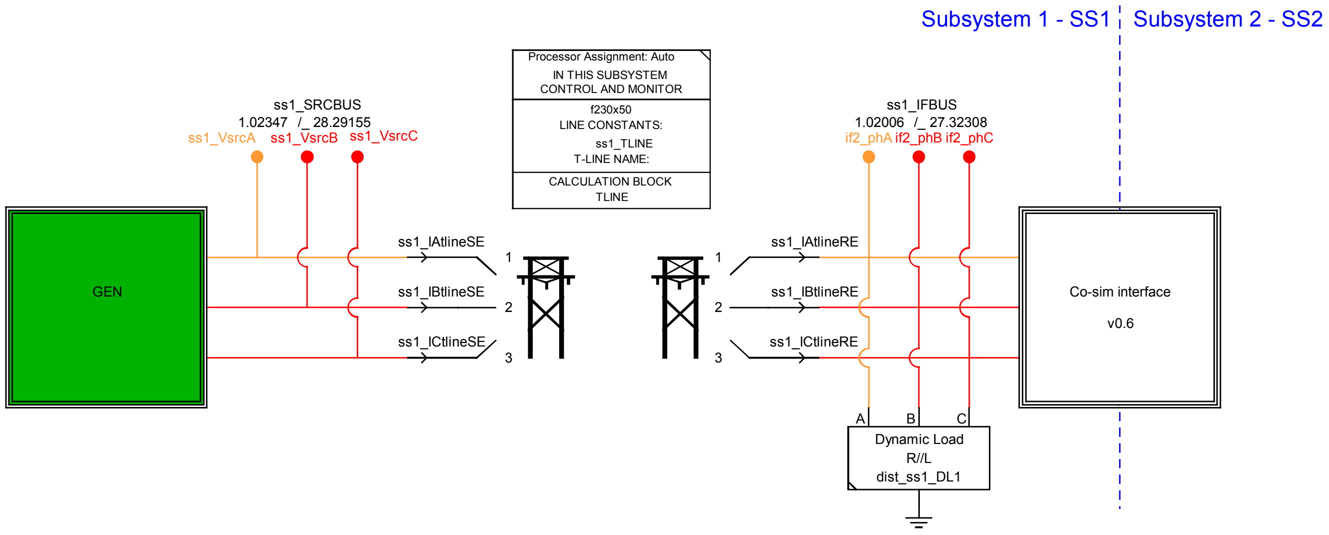
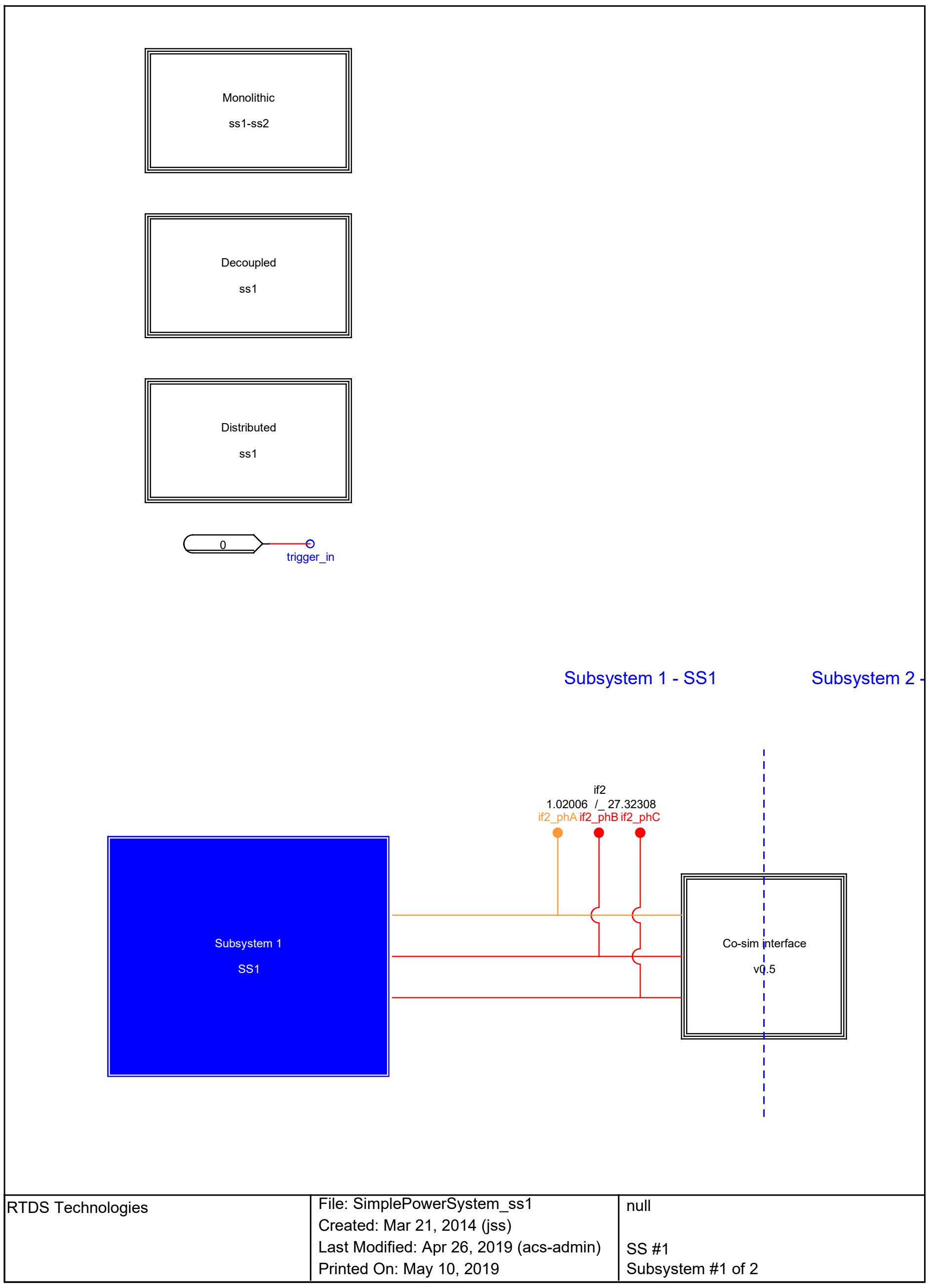


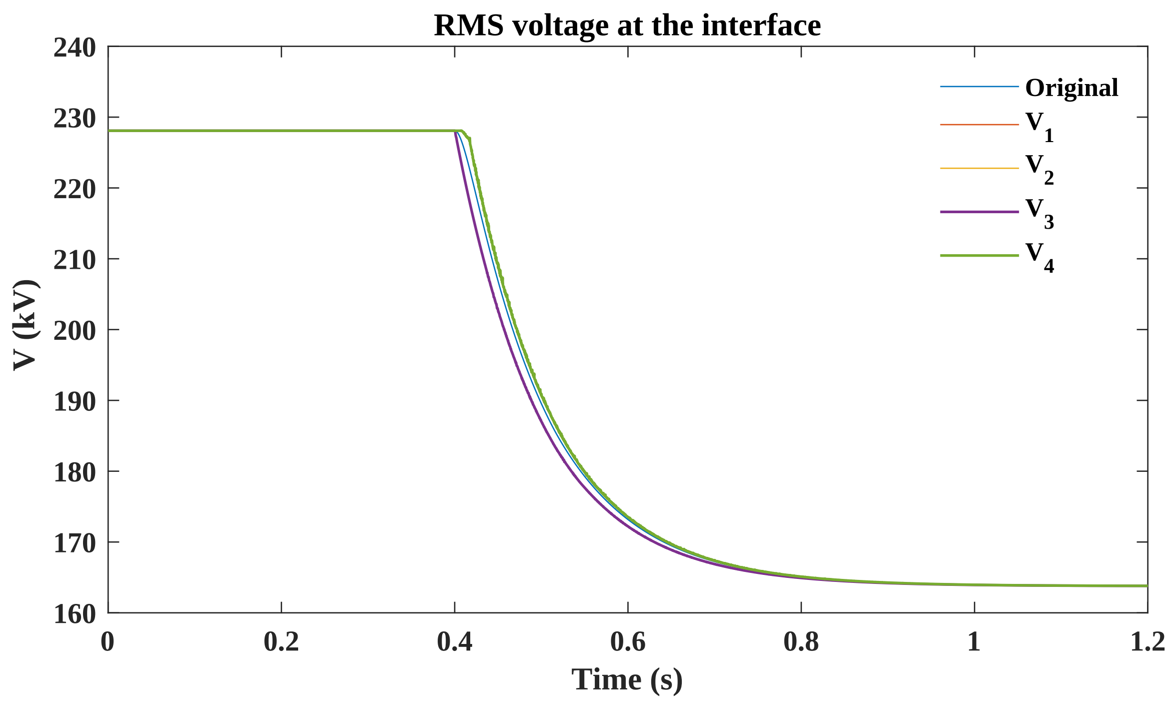
Figure B.1: RSCAD Draft of Simple Power System Model



RTDS Technologies
File: SimplePowerSystem_ss1
Created: Mar 21, 2014 (jss)
Last Modified: Apr 26, 2019 (acs-admin)
Printed On: May 10, 2019
null
SS #1
Subsystem #1 of 2

Figure B.2: RSCAD Draft of co-simulation interface

B.2. Additional Co-simulation results



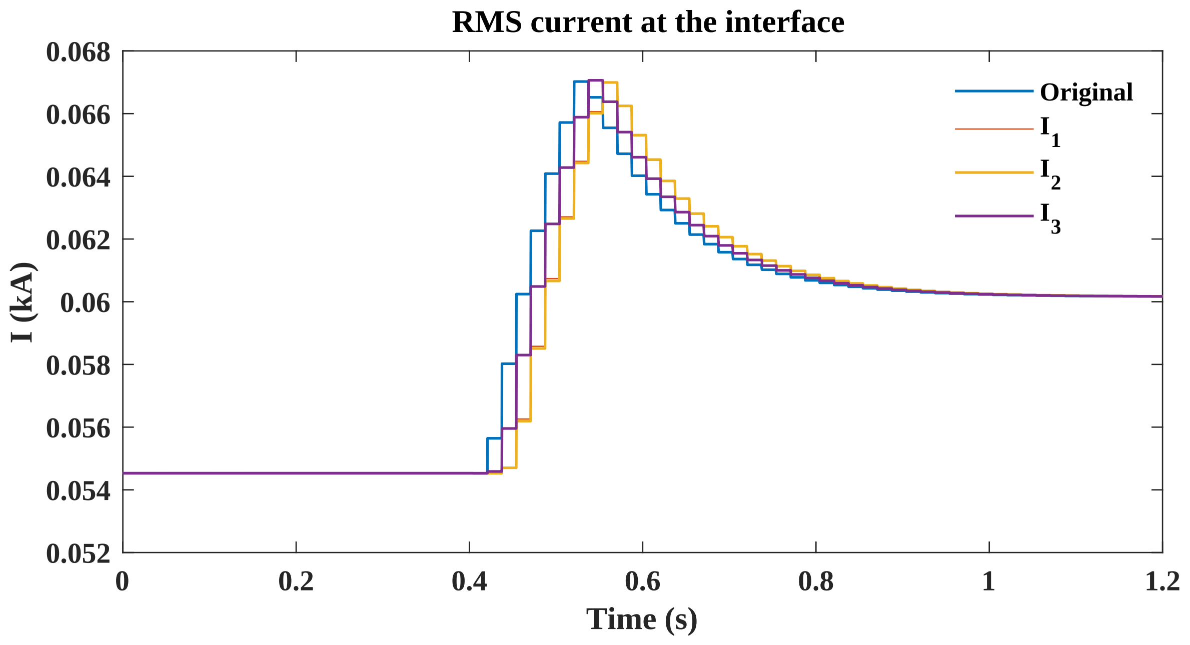


Figure B.3: RMS Voltages and Currents

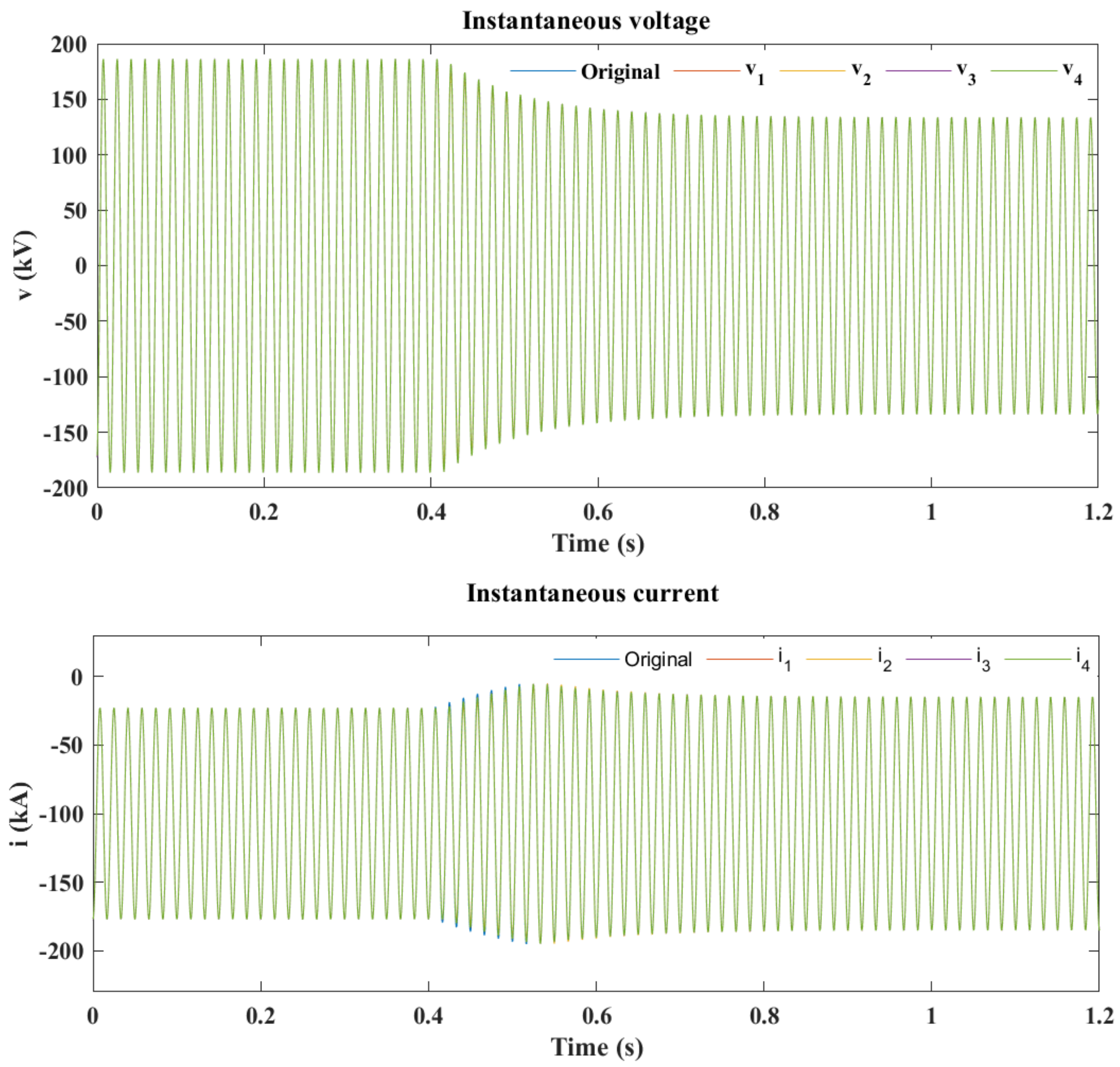


Figure B.4: Instantaneous Voltages and Currents

B.3. Effects of Communication Jitter and Congestion

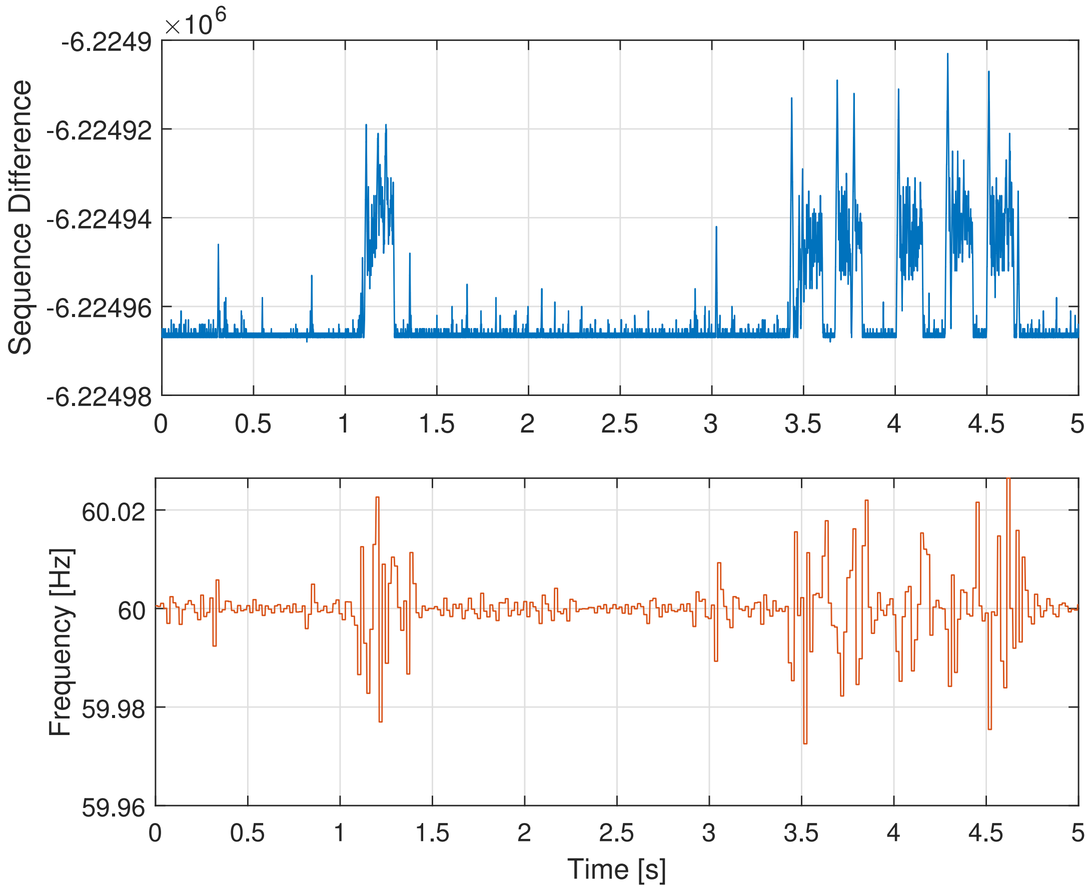


Figure B.5: Frequency variations at the interface caused by spikes in communication latency.

C

VILLASnode Configuration Files

C.1. Communication tests

```


priority = 99
affinity = 0x1c

logging = {
  level = "info"
}

nodes = {

  acs = {
    type = "socket"
    layer = "udp"

    in = {
      address = "*:12000",
      builtin = false

      signals = {
        count = 500
        type = "float"
      }

      hooks = (
        {
          type = "fix"
        },
        {
          type = "restart"
        },
        {
          type = "stats",

          verbose = true
          warmup = 100
        }
      )
    }
    out = {
      address = ”10.10.12.2:12000”
    }
  }

  test_data = {
    type = ”test_rtt”

    cooldown = 2,                            # The cooldown time between each test case in seconds

                prefix = ”ct2_1_%Y-%m-%d_%H-%M-%S”,                      # An optional prefix in the filename
                output = ”/home/iepg/Sciebo/ERIGrid TA/Measurements/CT/CT2”,           # The output directory for all results
                                                            # The results of each test case will be written to a seperate file.
                format = ”villas.human”,                    # The output format of the result files.

                cases = (                                   # The list of test cases
#                        {
#       rates = [
#         1, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500,
#         5000, 5500, 6000, 6500, 7000, 7500, 8000, 8500, 9000,
#         9500, 10000, 10500, 11000, 11500, 12000, 12500, 13000, 13500,
#         14000, 14500, 15000, 15500, 16000, 16500, 17000, 17500, 18000, 18500,
#         19000, 19500, 20000
#       ]
 #                             values = [ 24 ]  # An array of number of values
  #                            duration = 60           # The duration of the test case in seconds (depending on the sending rate)
   #                     },
      {
        rates = [ 2000 ]
#       values = [
#           1,  10,  20,  30,  40,  50,  60,  70,  80,  90,
#         100, 110, 120, 130, 140, 150, 160, 170, 180, 190,
#         200, 210, 220, 230, 240, 250, 260, 270, 280, 290,
#         300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400
#       ]
        values = [ 350 ]
        duration = 60
      }
                )

  }
}


paths = (
  {
    in = "test_data",
    out = "acs"

    hooks = (
#       { type = "print" }
    )
  },
  {
    in = "acs",
    out = "test_data",

    hooks = (
#       { type = "print" }
    )
  }
)
```

Listing C.1: Configuration for communication tests

C.2. Simulation Tests

```

priority = 99
affinity = 0x1c

logging = {
  level = "info"
}

nodes = {

  acs = {
    type = "socket"
    layer = "udp"

    in = {
      address = "*:12000",
      builtin = true

      hooks = (
        {
          type = "stats",

          verbose = true
          warmup = 20
        }
      )
    }
    out = {
      address = "10.10.12.2:12000"
    }
  }

  rtds-tud = {
    type = "socket"
    format = "gtnet.fake"
    builtin = true

    in = {
      address = ”*:12002”

      hooks = (
        { type = ”stats” }
      )
    }
    out = {
      address = ”10.10.9.2:12002”
    }
  }
}


paths = (
  {
    in = ”rtds-tud”
    out = ”acs”
  },
  {
    in = ”acs”
    out = ”rtds-tud”
  }
)
```

Listing C.2: Configuration for simulation tests

C.3. Implementation of Real-Time Protocol (RTP)

```
priority = 99
affinity = 0x1c

nodes = {

  acs = {
    type = ”rtp”

    rate = 20000

    rtcp = {
      mode = ”aimd”
      throttle_mode = ”limit_rate”
    }

    aimd = {
      a = 100
      b = 0.8

      start_rate = 8000
    }

    in = {
      address = ”0.0.0.0:12000”,

      signals = {
        count = 24
        type = "float"
      }

      hooks = (
        {
          type = "stats",

          verbose = true
          warmup = 100
        }
      )
    }
    out = {
      address = "10.10.12.2:12000"
    }
  }

  signal = {
    type = "signal"

    rate = 20000
    signal = "constant"
    values = 24
  }
}


paths = (
  {
    in = "signal",
    out = "acs"

    hooks = (
#     { type = "print" }
    )
  },
  {
    enabled = false
    in = "acs"

    hooks = (
#     { type = "print" }
    )
  }
)
```

Listing C.3: Configuration for RTP

Bibliography

[1] *Paris agreement,* UNTC XXVII 7.d.

[2] E. Commission, *EUROPE 2020 – A strategy for smart, sustainable and inclusive growth*, Tech. Rep.

[3] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, *A survey on smart grid communication infrastructures: Motivations, requirements and challenges,* IEEE Communications Surveys Tutorials **15**, 5 (2013).

[4] ABB Inc., *When grids get smart - abb's vision for the power system of the future,* .

[5] J. Jackson, *Chapter 28 - smart grids: An optimised electric power system,* in *Future Energy (Second Edition)*, edited by T. M. Letcher (Elsevier, Boston, 2014) second edition ed., pp. 633 – 651.

[6] Y. Ben-Haim and F. M. Hemez, *Robustness, fidelity and prediction-looseness of models,* Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **468**, 227 (2012), https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2011.0050 .

[7] A. Monti, M. Stevic, S. Vogel, R. W. D. Doncker, E. Bompard, A. Estebsari, F. Profumo, R. Hovsapian, M. Mohanpurkar, J. D. Flicker, V. Gevorgian, S. Suryanarayanan, A. K. Srivastava, and A. Benigni, *A global real-time superlab: Enabling high penetration of power electronics in the electric grid,* IEEE Power Electronics Magazine **5**, 35 (2018).

[8] M. Stevic, S. Vogel, A. Monti, and S. D'Arco, *Feasibility of geographically distributed real-time simulation of HVDC system interconnected with AC networks,* in *2015 IEEE Eindhoven PowerTech, PowerTech 2015* (2015).

[9] M. Stevic, S. Vogel, M. Grigull, A. Monti, A. Estebsari, E. Pons, T. Huang, and E. Bompard, *Virtual integration of laboratories over long distance for real-time co-simulation of power systems,* in *IECON Proceedings (Industrial Electronics Conference)* (2016).

[10] M. Stifter, J. H. Kazmi, F. Andrén, and T. Strasser, *Co-simulation of power systems, communication and controls,* in *2014 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)* (2014) pp. 1–6.

[11] Hua Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili, *Power system and communication network co-simulation for smart grid applications,* in *ISGT 2011* (2011) pp. 1–6.

[12] R. Bottura, D. Babazadeh, K. Zhu, A. Borghetti, L. Nordström, and C. A. Nucci, *Sitl and hla co-simulation platforms: Tools for analysis of the integrated ict and electric power system,* in *Eurocon 2013* (2013) pp. 918–925.

[13] M. Stevic, A. Estebsari, S. Vogel, E. Pons, E. Bompard, M. Masera, and A. Monti, *Multi-site european framework for real-time co-simulation of power systems,* IET Generation, Transmission Distribution **11**, 4126 (2017).

[14] C. Steinbrink, F. Schlögl, D. Babazadeh, S. Lehnhoff, S. Rohjans, and A. Narayan, *Future perspectives of co-simulation in the smart grid domain,* in *2018 IEEE International Energy Conference (ENERGYCON)* (2018) pp. 1–6.

[15] M. Mirz, S. Vogel, G. Reinke, and A. Monti, *Dpsim—a dynamic phasor real-time simulator for power systems,* SoftwareX **10**, 100253 (2019).

[16] M. H. Cintuglu, O. A. Mohammed, K. Akkaya, and A. S. Uluagac, *A Survey on Smart Grid Cyber-Physical System Testbeds,* (2017).

[17] B. Palmintier, B. Lundstrom, S. Chakraborty, T. Williams, K. Schneider, and D. Chassin, *A Power Hardware-in-the-Loop Platform with Remote Distribution Circuit Cosimulation,* IEEE Transactions on Industrial Electronics (2015), 10.1109/TIE.2014.2367462.

[18] C. Dufour and J. Belanger, *On the use of real-time simulation technology in smart grid research and development,* in *IEEE Transactions on Industry Applications*, Vol. 50 (2014).

[19] D. Bian, M. Kuzlu, M. Pipattanasomporn, S. Rahman, and Y. Wu, *Real-time co-simulation platform using OPAL-RT and OPNET for analyzing smart grid performance,* in *IEEE Power and Energy Society General Meeting*, Vol. 2015-September (2015).

[20] D. Bhor, K. Angappan, and K. M. Sivalingam, *Network and power-grid co-simulation framework for Smart Grid wide-area monitoring networks,* in *Journal of Network and Computer Applications*, Vol. 59 (2016).

[21] M. Garau, G. Celli, E. Ghiani, F. Pilo, and S. Corti, *Evaluation of Smart Grid Communication Technologies with a Co-Simulation Platform,* IEEE Wireless Communications **24** (2017), 10.1109/MWC.2017.1600214.

[22] TUDelft, *Power systems co-simulation,* .

[23] D. Shu, X. Xie, V. Dinavahi, C. Zhang, X. Ye, and Q. Jiang, *Dynamic phasor based interface model for emt and transient stability hybrid simulations,* IEEE Transactions on Power Systems **33**, 3930 (2018).

[24] M. Stevic, A. Monti, and A. Benigni, *Development of a simulator-to-simulator interface for geographically distributed simulation of power systems in real time,* in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society* (2015) pp. 5020–5025.

[25] M. Stevic, M. Panwar, M. Mohanpurkar, R. Hovsapian, and A. Monti, *Empirical study of simulation fidelity in geographically distributed real-time simulations,* in *2017 North American Power Symposium, NAPS 2017* (2017).

[26] M. Mirz, A. Estebsari, F. Arrigo, E. Bompard, and A. Monti, *Dynamic phasors to enable distributed real-time simulation,* in *2017 6th International Conference on Clean Electrical Power (ICCEP)* (2017) pp. 139–144.

[27] K. Mudunkotuwa, S. Filizadeh, and U. Annakkage, *Development of a hybrid simulator by interfacing dynamic phasors with electromagnetic transient simulation,* IET Generation, Transmission & Distribution (2017), 10.1049/iet-gtd.2016.1616.

[28] J. L. Cale, B. B. Johnson, E. Dall'Anese, P. M. Young, G. Duggan, P. A. Bedge, D. Zimmerle, and L. Holton, *Mitigating communication delays in remotely connected hardware-in-the-loop experiments,* IEEE Transactions on Industrial Electronics **65** (2018), 10.1109/TIE.2018.2821618.

[29] A. T. J. Martí and J. Jatskevich, *Transient stability analysis using shifted frequency analysis (sfa),* in *2018 Power Systems Computation Conference (PSCC)* (2018) pp. 1–7.

[30] A. Markou, V. Kleftakis, P. Kotsampopoulos, and N. Hatziargyriou, *Improving existing methods for stable and more accurate Power Hardware-in-the-Loop experiments,* in *IEEE International Symposium on Industrial Electronics* (2017).

[31] W. Ren, M. Sloderbeck, M. Steurer, V. Dinavahi, T. Noda, S. Filizadeh, A. R. Chevrefils, M. Matar, R. Iravani, C. Dufour, J. Belanger, M. O. Faruque, K. Strunz, and J. A. Martinez, *Interfacing issues in real-time digital simulators,* IEEE Transactions on Power Delivery **26**, 1221 (2011).

[32] R. G. Sargent, *Verification and validation of simulation models,* in *Proceedings of the 2010 Winter Simulation Conference* (2010) pp. 166–183.

[33] DERi, *Distributed Energy Resources Research Infrastructure,* Available : `http://www.der-ri.net`.

[34] E. Union, *ERIGrid - European Research Infrastructure supporting Smart Grid Systems Technology Development, Validation and Roll Out - project,* https://erigrid.eu/the-project/.

[35] A. A. van der Meer, P. Palensky, K. Heussen, D. E. M. Bondy, O. Gehrke, C. Steinbrinki, M. Blanki, S. Lehnhoff, E. Widl, C. Moyo, T. I. Strasser, V. H. Nguyen, N. Akroud, M. H. Syed, A. Emhemed, S. Rohjans, R. Brandl, and A. M. Khavari, *Cyber-physical energy systems modeling, test specification, and co-simulation based testing,* in *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)* (2017) pp. 1–9.

[36] T. I. Strasser, C. Moyo, R. Bründlinger, S. Lehnhoff, M. Blank, P. Palensky, A. A. van der Meer, K. Heussen, O. Gehrke, J. E. Rodriguez, J. Merino, C. Sandroni, M. Verga, M. Calin, A. Khavari, M. Sosnina, E. de Jong, S. Rohjans, A. Kulmala, K. Mäki, R. Brandl, F. Coffele, G. M. Burt, P. Kotsampopoulos, and N. Hatziargyriou, *An integrated research infrastructure for validating cyber-physical energy systems,* in *Industrial Applications of Holonic and Multi-Agent Systems*, edited by V. Mařík, W. Wahlster, T. Strasser, and P. Kadera (Springer International Publishing, Cham, 2017) pp. 157–170.

[37] W. R. Stevens, B. Fenner, and A. M. Rudoff, *UNIX Network Programming, Vol. 1*, 3rd ed. (Pearson Education, 2003).

[38] Redis Labs, *Redis(software),* .

[39] *RedisRepl (Software),* Available :https://bitbucket.org/danielePala/redisrepl.

[40] FEIN Aachen e.V., *Villas framework,* .

[41] S. Papathanassiou, N. Hatziargyriou, and K. Strunz, *A benchmark low voltage microgrid network,* CIGRE Symposium (2005).

[42] M. Maniatopoulos, D. Lagos, P. Kotsampopoulos, and N. Hatziargyriou, *Combined control and power hardware in-the-loop simulation for testing smart grid control algorithms,* IET Generation, Transmission Distribution **11**, 3009 (2017).

[43] C. Eck, J. Knobloch, L. Robertson, I. Bird, K. Bos, N. Brook, D. Düllmann, I. Fisk, D. Foster, B. Gibbard, C. Grandi, F. Grey, J. Harvey, A. Heiss, F. Hemmer, S. Jarp, R. Jones, D. Kelsey, M. Lamanna, H. Marten, P. Mato-Vila, F. Ould-Saada, B. Panzer-Steindel, L. Perini, Y. Schutz, U. Schwickerath, J. Shiers, and T. Wenaus, *LHC computing Grid: Technical Design Report. Version 1.06 (20 Jun 2005)*, Technical Design Report LCG (CERN, 2005).

[44] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne, *RTP: A transport protocol for real-time applications,* .

[45] G. Sliepen, *The difficulties of a peer-to-peer VPN on the hostile internet,* in *Proceedings of the Free and Open Source Software Developers' European Meeting-FOSDEM*, Vol. 109 (2010).

[46] R. Liu, M. Mohanpurkar, M. Panwar, R. Hovsapian, A. Srivastava, and S. Suryanarayanan, *Geographically distributed real-time digital simulations using linear prediction,* International Journal of Electrical Power & Energy Systems **84**, 308 (2017).

[47] E. Guillo-Sansano, A. J. Roscoe, C. E. Jones, and G. M. Burt, *A new control method for the power interface in power hardware-in-the-loop simulation to compensate for the time delay,* in *2014 49th International Universities Power Engineering Conference (UPEC)* (2014) pp. 1–5.

[48] R. A. Gupta and M. Chow, *Networked control system: Overview and research trends,* IEEE Transactions on Industrial Electronics **57**, 2527 (2010).