RESERVE

D4.1 v4

Demonstration of prototype of Laboratory infrastructure

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement no 727481.

Project Name	RESERVE
Contractual Delivery Date:	30.09.2017
Actual Delivery Date:	30.09.2017
Contributors:	Steffen Vogel (RWTH), Markus Mirz (RWTH)
Workpackage:	WP4 – Pan-European real time simulation infrastructure and live 5G testing platform
Security:	PU
Nature:	R
Version:	1.0
Total number of pages:	25

Abstract

This document collects the specifications for the laboratory interconnection infrastructure. The report details the protocol adopted for the laboratory interconnection as well as the data structure used for the communication. Requirements in terms of time delay performance are also discussed in relation to the simulation requirements.

Keyword list

Distributed, Hardware-in-the-loop, Real-time, Simulation, pan-European, Infrastructure, Laboratory

Executive Summary

The research of RESERVE touches fundamental concepts of today's energy systems. Decreasing bulk energy generation while increasing the share of distributed energy resources, has an impact on the entire system. The current practice of controlling the grid in a hierarchical way is being steadily replaced with a decentralized approach that is more reliant on communication infrastructure.

Testing new concepts in a large-scale power system scenario is a challenging task for two reasons: costs and control of all system parameters. A test grid would have to be set up separately from the grid that provides energy to customers to ensure quality of service. Moreover, it is not possible to control all test parameters such as the weather, which has a large impact as the share of renewable energy sources increases.

RESERVE proposes a distributed real-time simulation environment which relies on the Internet and utilizes the simulation resources available from project partners to test the concepts developed in the project. Real-time simulation allows for the integration of hardware which enables tests that include real devices and software components.

The pan-European laboratory infrastructure promotes closer collaboration between research communities in Europe and facilitates a more efficient utilization of simulation and hardware-in-the-loop (HIL) resources. This effort based upon the premise that a pan-European laboratory infrastructure will support researchers and industry to work together in new ways which have not been possible before due to the inaccessibility of simulation resources or stringent policies on grid data. Distributed co-simulation allows each partner to maintain control over their data and only provide partners with a black-box view of their system.

As pioneered by Cloud Computing, we envision that simulation can be provided as a service to industry [1]. Such a service not only provides access to simulation targets which are usually quite expensive but also to the knowledge / intellectual property (IP) of the simulation provider. A similar trend has already gained ground in the simulation and verification of integrated circuits. The demonstration of an integrated pan-European laboratory infrastructure is an important step into this direction.

The objective of task T4.1 is the development of an infrastructure for conducting geographically distributed real-time simulation at a pan-European scale. The design and initial implementation of a modular toolset called VILLASframework has been completed. This tool will enable subsequent tasks to validate and study novel control paradigms as developed in work packages WP2 and WP3. New network codes as defined by WP6 will be validated and proposed[1].

The framework consists of three components. VILLASnode is a gateway which implements exchange of simulation data in real-time. VILLASweb is a web-frontend used to monitor and control the simulation. VILLAScontroller is message broker used for the overall orchestration of the simulation. An associated workflow specifies the required steps for conducting a simulation with several involved parties.

It is emphasized that the pan-European laboratory infrastructure uses information and communication technology (ICT) solely for enabling geographically distributed co-simulation. In contrast to the tasks T4.4 and T4.5 where 5G mobile networks are evaluated as part of a Cyber-Physical System (CPS), ICT is considered a tool and not the object of studies.

Nevertheless, the integration of the pan-European co-simulation with the 5G testbed requires interfaces between simulators and industry standard protocols. For this, OpenFMB and VILLASframework will use the industry standard IEC61850-9-2 Sampled Values (SV).

The implementation of the framework is targeted to be completed in M18 (March 2018) in alignment with milestone MS9 and first project review.

[1] http://fein-aachen.rwth-aachen.de/projects/villas-framework/

Authors

Partner	Name	e-mail
RWTH		
	Steffen Vogel	Phone: +49 241 8049740 stvogel@eonerc.rwth-aachen.de
	Markus Mirz	Phone: +49 241 80 49739 mmirz@eonerc.rwth-aachen.de
	Antonello Monti	Phone: +49 241 80 49700 amonti@eonerc.rwth-aachen.de

Table of Contents

1. Introduction

Renewables in a Stable Electric Grid (RESERVE) is a three-year European Commission funded project within the Work Program H2020-LCE-2016-2017. The project officially started in October 2016.

1.1 Task 4.1

This deliverable is the output of task 4.1 in WP4. This task focues on the design of a pan-European laboratory infrastructure which is developed in the framework of RESERVE. This deliverable describes the architecture and software components of the VILLASframework tool suite to facilitate distributed simulation across Europe.

1.2 Objectives of the Work Report in this Deliverable

- To provide a workflow for conducting large-scale distributed simulations
- To develop tools and methodologies for conducting geographically distributed real-time simulations at a European scale
- To describe the architecture and implementation of software components required for these workflows
- Performing geographically distributed simulation adds complexity to simulators, solvers and their interconnection. Inherent communication latencies and the integration of heterogeneous simulation tools in a distributed system are the main challenges which need to be tackled.

1.3 Outline of the Deliverable

This deliverable describes the laboratory interfaces provided by the RWTH Aachen Institute for Automation of Complex Power System (ACS). This work provides the infrastructure to conduct geographically distributed co-simulations. This involves the data exchange between simulators within and between sites, monitoring and collection of simulation results as well as the overall orchestration of the involved resources. ACS developed a suite of software tools named VILLASframework to facilitate these tasks [2].

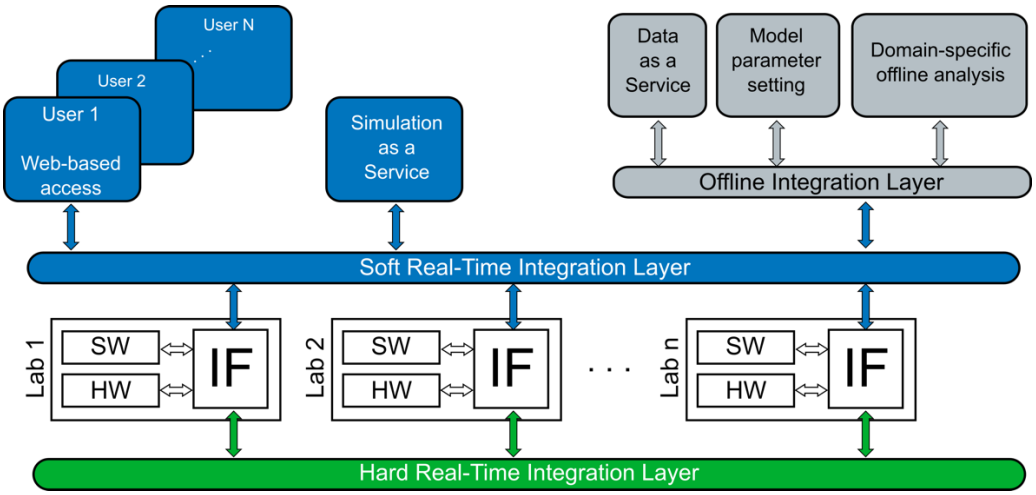

Figure 1: VILLAS integration layers.

Figure 1 shows the integration layers, a concept developed as part of the **V**irtually **I**nterconnected **L**aboratories for **LA**rge systems **S**imulation/emulation project at ACS [3]. A central aspect is the separation of the architecture into hard and soft real-time domains. The hard-real-time layer (red) provides connectivity between closely coupled simulation equipment which is located locally at a site or laboratory. A soft-real-time layer (blue) provides monitoring, data collection and control over an ongoing simulation. An offline integration layer (green) provides access to simulation results and inputs of a completed scenario for further post-processing.

The central component of the framework, the laboratory interfaces (IF) are implemented by a component called *VILLASnode*. This application acts as a gateway for simulation data between the hard and soft real-time domains. It provides a variety of interfaces to connect:

- Digital Real-time Simulators (DRTS)
- Cyber-physical Systems (CPS)
- Hardware-in-the-Loop (HiL)
- Software Modelling Tools
- Web Services and Databases

VILLASnode is designed with focus on very low and deterministic latency to achieve real-time exchange of simulation data where necessary. VILLASnode is and has already been used in several geographically distributed co-simulation scenarios and has been developed further in the context of RESERVE.

Services of the soft-real-time integration layer are accessible over a web interface which is implemented by the *VILLASweb* and *VILLAScontroller* components. They are to be extended in the frame of the RESERVE project which is why this document provides requirements that should be fulfilled by the final version to be used in RESERVE work task T4.3.

1.4 How to Read this Document

This report can be read without prior knowledge of other work within the RESERVE project. D4.1 provides tools which will be used by several other upcoming tasks in the WP 4 and WP 5. Figure 2 shows the relationships between the deliverables of WP 4. Please note that within task 4.1 only tools and methodologies are developed which will be applied by tasks from WP 5 in order to evaluate the control paradigms developed in WP 2 and WP 3.

For further reading, the reader may access the user documentation, an online demo and/or the source code which is freely accessible at the VILLASframework project website:

- http://fein-aachen.org/projects/villas-framework/

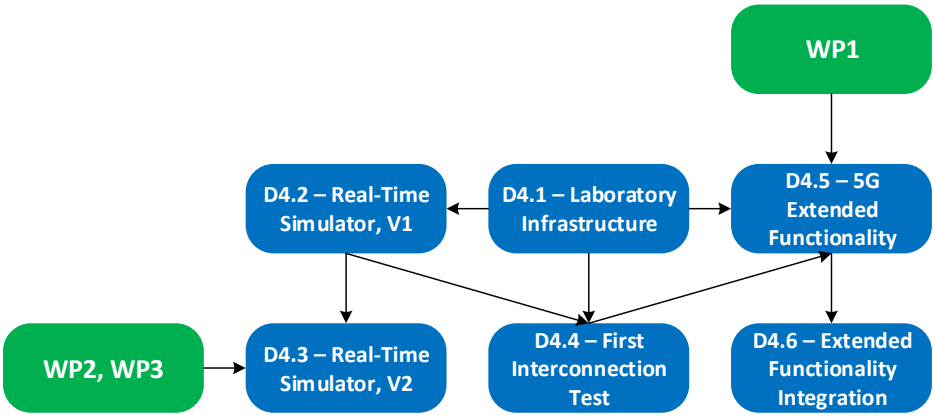

Figure 2: Relations between deliverables in WP4 and other work

1.5 Approach used to Undertake the Work

The initial approach of this framework has been devised by the VILLAS project by ACS [3]. Initially, it was used in a geographically distributed simulation as an integral part of the ERIC lab demonstration [4]. The work of this task is one of many iterations to optimize VILLASframework for increasingly complex co-simulation scenarios. Prior to the design phase, several existing co-simulation frameworks have been compared and evaluated with the result that VILLASframework is the only scalable framework for real-time co-simulation of power systems.

2. Requirements on the Laboratory Infrastructure

Due to the pan-European location of co-simulation sites within the RESERVE project, national research and education networks (NREN) like DFN & GEANT provide the only cost-effective way for interconnection. Long-range micro wave networks which are used for high frequency trading (HFT) in the financial sector provide improved communication characteristics but are inaccessible for research [5]. Existing 4G mobile networks have shown large latencies due to their core network architecture. Next-generation 5G mobile networks will offer latencies which could make real-time simulation feasible for metropolitan area networks.

As a shared medium, delivery of network packets is only guaranteed on a best effort basis without any guaranteed quality of service (QoS) [6]. As a result, packets may be lost or arrive reordered with a normal distributed latency. This imposes the requirement of an interface algorithm (IA) for the laboratory interface **which must compensate the latency introduced by the communication medium**. Common IAs like the ideal transformer model (ITM) or travelling line model (TLM) used in HIL simulations are unsuitable to compensate larger latencies above 10 ms, as they would require systems to be separated by unrealistic long transmission lines.

In past experiments, two alternative IAs based on mean RMS values and Dynamic Phasors (DP) have been used [7], [8]. The first method estimates voltage and current RMS values, their frequency and phase on the decoupling point before sending these values to the remote site for reconstruction. The more recent approach using Dynamic Phasors transforms interface quantities into the time-frequency[2] domain and exchanges phasors for up to 7th harmonics. Both IA are capable of compensating (time-varying) delay in steady state and during slow transients. Maintaining system stability and power balance of the interface is the main objective. At the current state, these IA cannot properly compensate delay during faster transients.

Another important requirement is the **scalability of the co-simulation setup**. To achieve this, we divide the setup into a control and a data plane. For the *data plane*, messaging protocols like MQTT, AMQP or OMA NGSI used by publish / subscribe message brokers are not suited due to their centralized topology, which not always reflects the topology of the power system. The quality of the simulation results is directly dependent on the communication latency and quality of service. Hence, for optimal results all sites must be linked directly by using a mesh of point-to-point connections which match the decoupling points of the simulated power system as close as possible. This eliminates central server / client topologies and limits the choice of available VPN technologies to secure communication.

In section 3.2 of [9], Steinbrick et. al. describe the concept of a co-simulation framework like implemented by OFFIS' Mosaik [10]. It provides interfaces between simulation tools, handles data exchange and acts as a scheduler for the simulation. VILLASnode applies these ideas to distributed real-time simulation by avoiding a central synchronization and replacing the central data exchange with point-to-point links. These changes are required as a central scheduler and broker would increase the communication latency in a geographically distributed simulation.

As the underlying communication network, the Internet provides an unreliable and insecure link between the simulation sites. The laboratory interface must be able to handle the loss of connectivity as well as protect the simulation resources from external threats. Encryption of virtual private network (VPN) solutions further impair the quality of service. Their use is a trade-off between security and the performance of the real-time execution.

To reduce complexity, a single VPN setup for both the data and the control plane is desired. As motivated above, several peer-to-peer (P2P) VPN connections are required to keep the communication latency as low as possible [11]. By using conventional VPN solutions like OpenVPN or IPsec, each of these connections would have to be configured manually and coordinated with the respective IT departments on site.

[2] "A time–frequency representation (TFR) is a view of a signal (taken to be a function of time) represented over both time ***and*** frequency." - Wikipedia

Instead, we propose to use a dynamic P2P VPN solution like Tinc[3] which handles the setup of a fully meshed network and works around firewall restrictions by using NAT-traversal techniques [12].

As a distributed system, the laboratory infrastructure is difficult to control and monitor without suitable tools and user interfaces. While it is important that the data plane is decentralized, and meshed for the performance reasons, mentioned previously, it is desirable to have a central control plane to provide the user with a single point-of-contact to orchestrate the simulation. This requires a method to remotely control simulators and their models as well as the automated deployment and configuration of software components at each site.

Figure 3 shows a full-featured example setup of a laboratory infrastructure built with components of the VILLASframework. There are four geographically distributed simulation sites in this project each with one or more digital real-time simulators (DRTS) and connected hardware in the loop (HIL) setups. A public web-server hosts the VILLASweb component which can be accessible by one or more workstations using a standard web browser. The simulation sites are interconnected via point-to-point connections to guarantee the shortest path between them where needed. In addition, each site streams simulation results to the public web-server for monitoring of live-data.

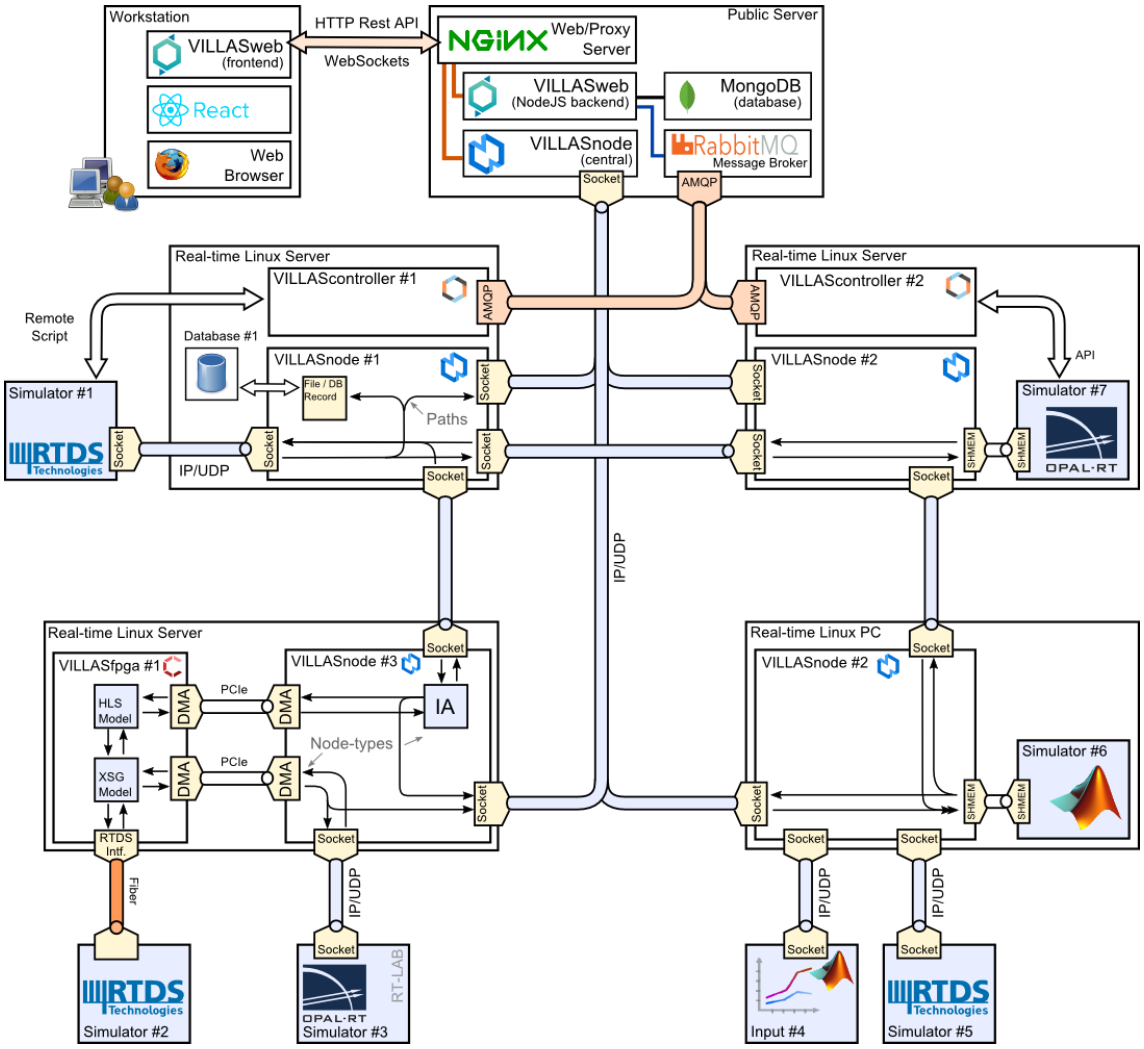

Figure 3: Example setup of VILLASframework.

[3] http://tinc-vpn.org

3. Workflow

RESERVE task T4.1 not only defines and implements components to realize the laboratory interconnection but also introduces a workflow to conduct simulations on this infrastructure. Envisioned is a workflow entirely guided by the web interface which is presented in section 5. This covers the model input supported by a web-based Common Information Model (CIM) editor, mapping of sub-systems onto simulation targets, the execution of the simulation as well as the management of targets, projects, users and simulation data. The web interface provides a single point of contact for all participating researchers and partners. Furthermore, it makes simulation resources available as a service to external parties (Simulation-as-a-service[4]).

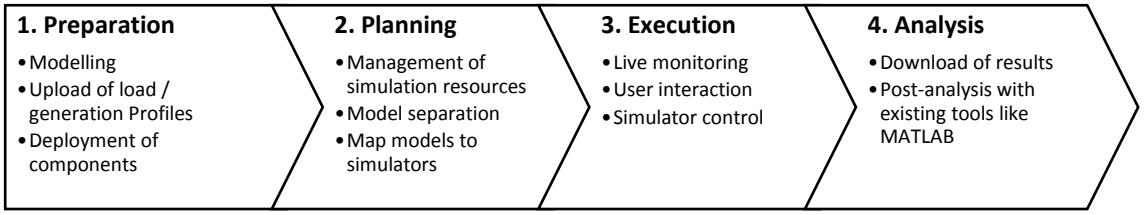

Figure 4: VILLASframework workflow.

Figure 4 presents the workflow, which is divided into four phases. In the preparation phase, users upload their models and associated data profiles to a database. Using the online CIM editor, users could adjust their models or partition them into smaller subsystems. Once all models are available, the user can proceed to the next phase in which models are mapped to simulators, compiled and loaded to the specific target. In phase three, the simulation is started while being controlled and monitored over the web interface. After the simulation has been performed, all simulators upload their results to the global database which is made available to all partners for post-analysis or future simulations.

The realization of this workflow requires the automation of several tasks that are usually manually performed. This is challenging as some of the proprietary simulation tools do not provide the necessary application programming interfaces (APIs). In RESERVE, the automation of this workflow for the DPsim solver, developed in task 4.2, will be presented.

[4] See Software-as-a-Service (Saas), Infrastructure-as-a-Service (Iaas) models used in Cloud Computing.

4. Laboratory Interface: VILLASnode

The laboratory interface, VILLASnode, consists of a server daemon which implements interfaces to several node types. The daemon acts as a gateway which forwards simulation data from input nodes to output nodes. From the viewpoint of the communication partners the gateway ideally is not detectable. This means that simulation data is should be forwarded unaltered and instantly. Hence, it is crucial to keep the added overhead in terms of latency and jitter as low as possible. VILLASnode's main tasks are the collection of statistics, multiplexing of simulation data, collection of results and monitoring of the quality of service.

The daemon is a multi-threaded Linux application. For optimal performance, the server is implemented in low-level C and makes use of several Linux-specific real-time features. The primary design goal was to make the behavior of the system as deterministic as possible. Therefore, it is advisable to run the server component on a `PREEMPT_RT` patched version of Linux. In the simulation environment of WP 4, a Fedora-based distribution is used which has been stripped to the bare minimum to eliminate any impact from other processes, such as a graphical user interface.

4.1 Concept

The VILLASnode software uses a concept of *nodes*, *paths* and *hooks* to build arbitrary topologies. Figure 5 shows an example configuration of a VILLASnode instance with nodes n_i which are connected by paths p_i. Each path connects one or more input nodes to one or more output nodes. A path consists of one queue q_i per input and output node, a register r_i as well as optional hook functions h_i. All paths within an instance are executed **asynchronously**. Each path reads simulation samples for their input nodes which get processed by a series of hook functions before transmitting to the outgoing nodes.

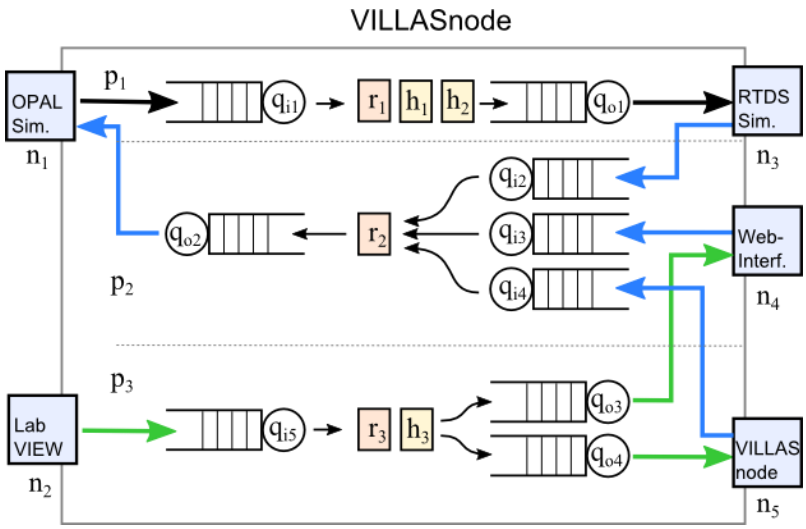

Figure 5: VILLASnode entities, an example configuration.

4.1.1 Nodes

In VILLASnode's terminology, *nodes* are interfaces to simulators, databases or other external devices and services. Figure 7 illustrates a selection of the currently available node types, a complete list is provided in the Annex in Table 1. Most interfaces are capable of sending and receiving batches of samples. A sample is defined as an array of signals with an associated timestamp and sequence number. In most cases, a batch contains only a single sample. However, certain node types can exchange multiple samples in a single packet or request in order to reduce the sending rate while maintaining temporal resolution.

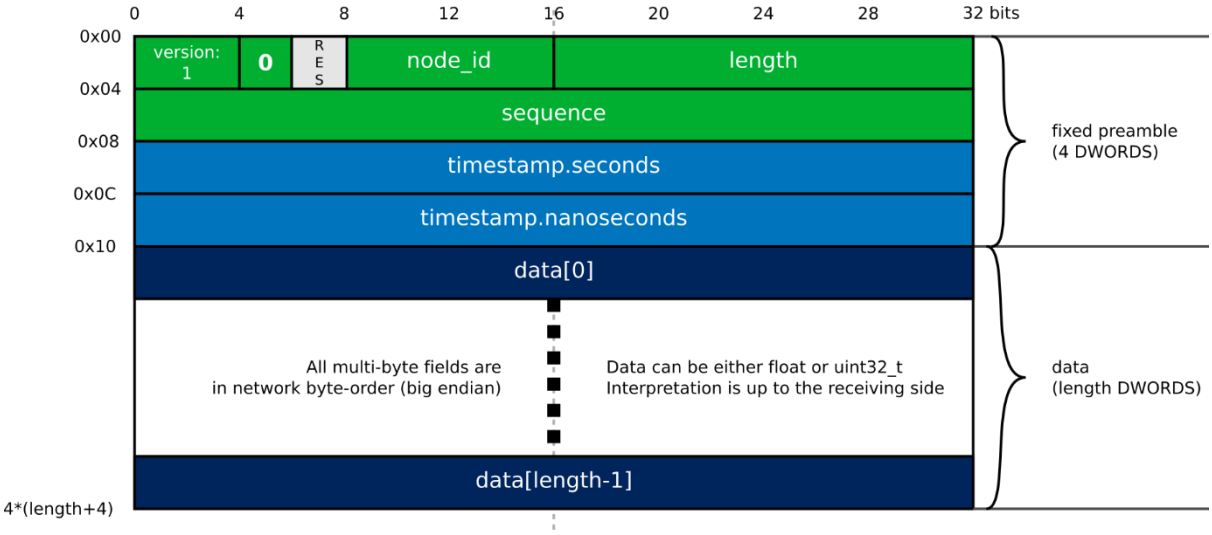

Figure 6: VILLASnode IP / UDP packet format.

In the RESERVE project, simulators will be interfaced by standard IP / UDP sockets and shared memory segments in case the simulation tool runs on the same machine as VILLASnode. Figure 6 shows the UDP packet format which is used to exchange simulation data over IP networks. It consists of a small header including a sequence number and timestamp followed by the raw data in single precision floating-point format. User interaction and monitoring over a web browser will be handled over a bi-directional WebSocket or an AMQP broker.

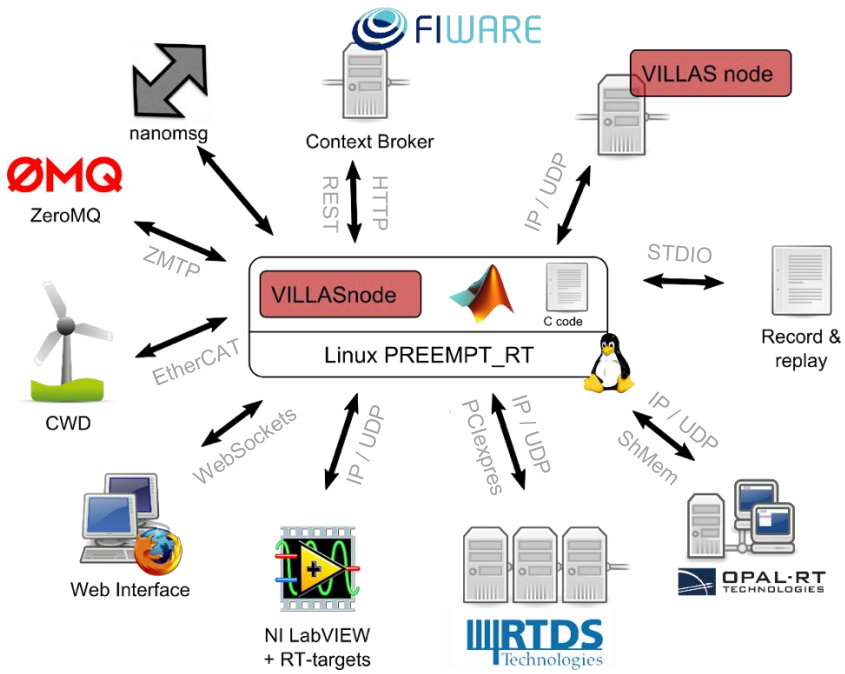

Figure 7: VILLASnode interfaces.

4.1.2 Paths

Paths establish an *n-to-n* relationship between input and output nodes. If simulation samples are received from multiple input nodes, the register should merge those samples with one of the strategies shown in Figure 8.

In the RESERVE project, register modes "*Wait for Any*" and "*Wait for Master*" have been implemented and are ready to be evaluated in task T4.3. More possible modes are shown in Figure 16 of the Annex.

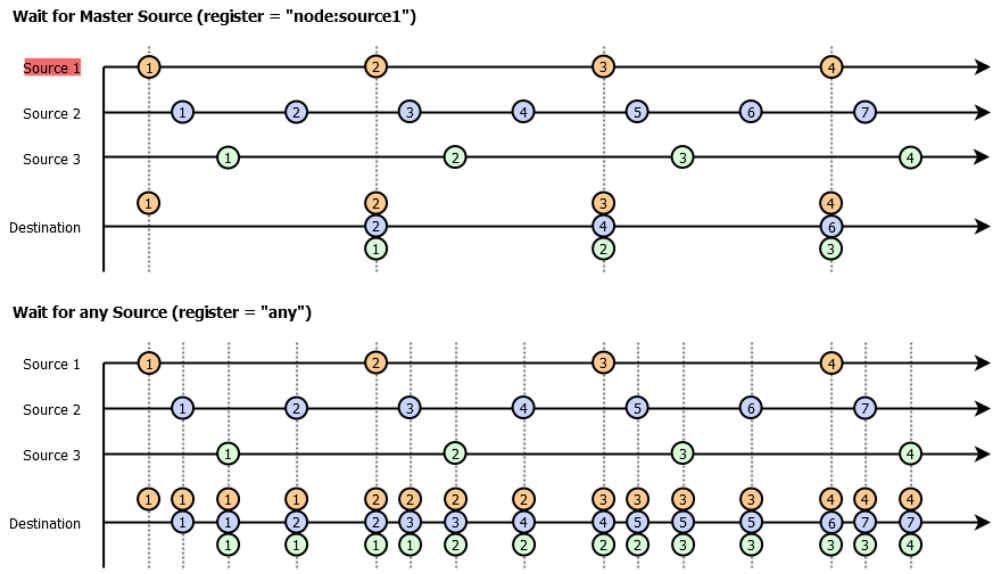

Figure 8: Register modes for paths.

4.1.3 Hooks

Hooks are pluggable entry points for user-defined functions in the usual processing flow. By default, sample data is forwarded without alterations. VILLASnode only validates the integrity of header fields such as a sequence number or possibly a cryptographic signature. In addition, each path can be extended by hook functions to allow further operations on the data. Examples are the collection of statistics or the implementation of interface algorithms. VILLASnode implements a few basic functions which are listed in Table 2. Hook functions are, like VILLASnode itself, implemented in C code.

4.2 Interface Algorithms

For the laboratory interconnection tests performed within the T4.3 of the RESERVE project, communication latencies are expected to be in the range of 10 – 50 ms. Such latencies make it impossible to couple geographically separated sub-systems in the time domain by using current and voltage instantaneous values. The most promising alternative is a coupling based on dynamic phasors (DP) [13].

The DPsim solver developed within the RESERVE task T4.2 already operates in the time-frequency domain and benefits from a simpler Interface Algorithm (IA) to compensate the communication delay.

For electro-magnetic transient (EMT) simulators (e.g. RTDS, OPAL-RT eMEGASIM) a transformation from the time-domain into the time-frequency domain is required. Furthermore, a time-step used in EMT simulations of $50\ \mu s$ would result in a sending rate of $20000\ pkt/s$ which would lead to congestion in the communication network. Based on experience and depending on the available communication network, a sending rate of $1000 - 5000\ pkts/s$ is realistic but must be tested prior to deployment. Both down-sampling and transformation are implemented as part of the IA and can be performed as a hook function in VILLASnode.

Moving the execution of the IA to VILLASnode has the benefit that the simulation models require less adaptations and IAs must only be implemented once for VILLASnode instead for each simulation platform. In addition, VILLASnode has more knowledge of the current QoS of the connection and could adaptively adjust parameters of the IA.

On the downside, VILLASnode and digital real-time simulators (DRTS) are exchanging samples in time domain and require hard real-time capable connections which are not always available due to the lack of required FPGA peripherals. In previous tests the IA has been implemented as part of the simulation models.

5. User Interface: VILLASweb

5.1 Requirements

The web interface should provide a single point of contact for users to control and monitor co-simulations. Therefore, it must be tightly integrated with the VILLASnode and VILLAScontroller components.

5.1.1 Integration with VILLASnode

The web interface can be used to display real-time data in the form of graphical representations available to the user. Data is streamed via WebSockets from a central VILLASnode instance to the web browsers. The central VILLASnode instance collects and merges data streams from the laboratory instances.

5.1.2 Integration with VILLAScontroller

The web interface acts as a central panel for controlling the simulation. VILLASnode instances and simulators can be started, stopped and checked for their status.

Models should be uploaded to the interface and from there deployed to the assigned simulation nodes. The configuration of the simulation is handled by the controller which is described in the following section. In the future, the web interface might also be used to provide a common modelling environment and a way to partition and map the integrated model onto the available simulation nodes.

5.2 Architecture

VILLASweb is built with modern web technologies like HTML 5, SVG, JavaScript, WebSockets, and NoSQL databases. Figure 9 depicts the architecture of VILLASweb which is divided in a React-based frontend application and a backend.

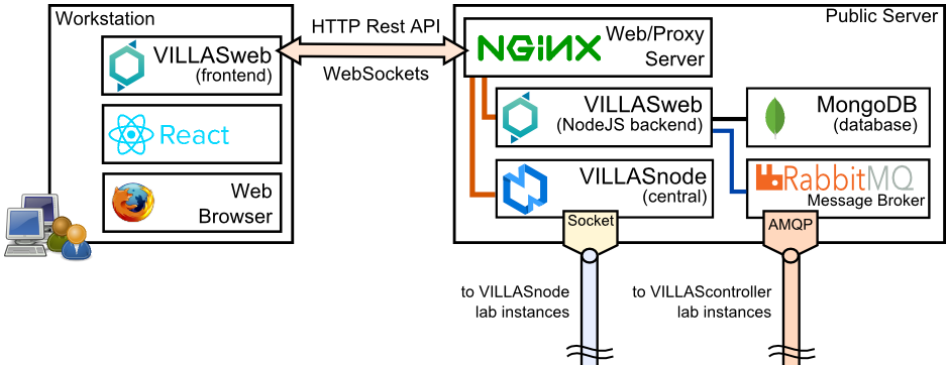

Figure 9: VILLASweb architecture.

The frontend runs within a user's web browser and connects via a HTTP REST API and WebSockets to the backend. The backend consists of a webserver like Nginx which acts as a proxy for API requests, static web content and WebSocket streams. API requests are handled by a NodeJS application which uses a MongoDB database for persistence. Live data is directly streamed from a central VILLASnode without being processed by the NodeJS backend. This central node acts as a collector for all secondary VILLASnode instances. It combines samples from all observed simulators and reduces the update rate. At the same time, the central node can be used to collect and store results for offline analysis.

5.3 User and Data Management

Several user accounts can be created. There are different roles for users such as admin, operator and guest. Administrators allow other users to register and manage roles/permissions.

Each user can create his own projects. A Project consists of all data that is specific for one simulation. This includes time series data that is exchanged with other nodes and simulation specific settings, such as visualizations. A visualization is a user-defined set of graphs, tables, buttons, etc. see section 5.4 for more details. Simulation settings can be edited in the web interface and it is possible to copy and paste projects or parts of the projects e.g. simulation

settings. Project-related data is available after the simulation has finished. Each time series or set of time series data can be downloaded as CSV files or an archive of CSV files.

Project data like visualizations and simulation settings (e.g. connected simulators) can be shared between projects and users, or copied from one place to another, for example, to be able to reuse visualizations in other projects. All data should be secured and inaccessible from outside except for read-only visualizations that are available to the public and do not require a login.

5.4 Visualization

Visualizations can be built in a WYSIWYG[5]-editor. Figure 11 shows a selection of available widgets. One or more variables can be plotted against time either in single or multiple diagrams. If multiple variables are plotted in one diagram they share one y-axis or have one y-axis each. Data can also be displayed in tables.

Furthermore, visualizations offer the means to control parameters of participating simulators. Therefore, buttons, input fields and similar controls are required.

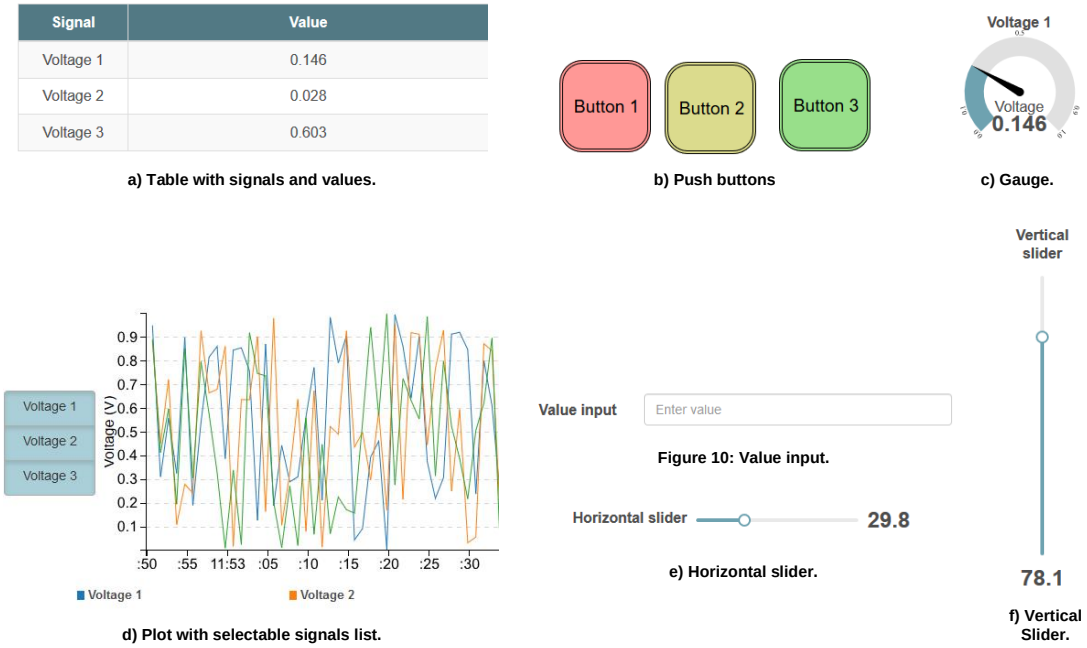

Figure 11: VILLASweb visualization widgets.

[5] What-you-see-is-what-you-get

6. Orchestration: VILLAScontroller

The VILLAScontroller component implements the control plane of the co-simulation infrastructure. A set of Python scripts is planned to interface devices and services with the web interface by using an AMQP broker like RabbitMQ. This component is currently under design. Implementation is planned to commence in Q3 of 2017.

Central orchestration can greatly simplify the procedure of conducting distributed co-simulations. It is an essential component for providing simulation as a service to external partners. However, if unavailable, users always can fall back on the traditional ways of manually controlling simulators from each site.

6.1 Requirements

In comparison to the data plane which is described in section 4, the requirements of the control plane are different. Instead of the lowest achievable latency for the exchange of simulation data, a reliable soft real-time messaging system is needed to distribute control commands. A delayed delivery of control commands must only impact the responsiveness of the system without degrading simulation performance. The control of real-time simulators with attached HIL setups is critical from a security perspective. Connectors must implement an access control mechanism which allows each site operator to define an access control list (ACL) for each of the devices. The controller must provide a common interface to simulators and VILLASnodes. This requires the definition of an abstraction level for these devices / services.

6.2 Architecture

Figure 12 shows the architecture of the VILLAScontroller component consisting of a RabbitMQ message broker and a set of Python scripts which communicate between the simulators and the broker.

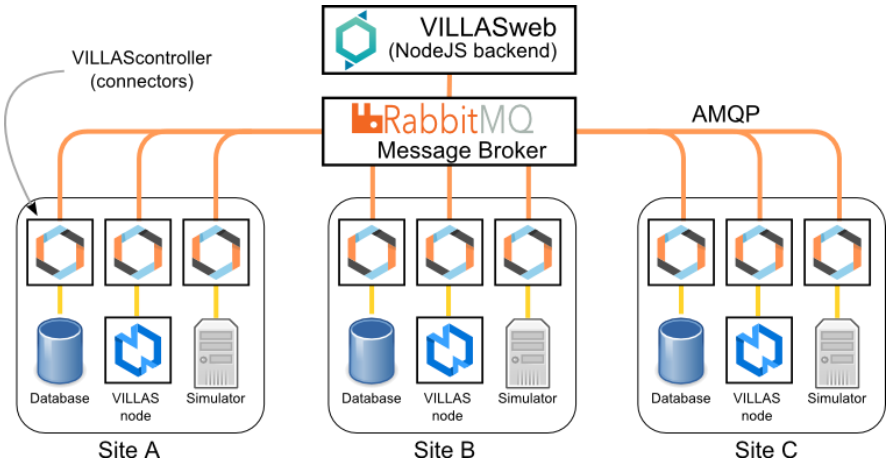

Figure 12: VILLAScontroller architecture.

In a distributed simulation, many different components are involved. The execution and configuration of real-time simulators, gateways, and databases must to be orchestrated and synchronized during the execution. The interfaces provided by these components are heterogeneous, ranging from proprietary tools like RSCAD and RT-LAB to standardized protocols like SSH, Telnet, FTP over to custom software development kits (SDK). The *VILLAScontroller* component will provide an abstraction layer on top of these interfaces in form of a common HTTP REST API. The main client of this API is the VILLASweb interface which allows users to control the distributed system from a central place.

The most critical and complex components in the system are real-time simulators. Their configuration is provided in the form of power system models. These models need to be

compiled and loaded to a real-time target before a simulation can start.

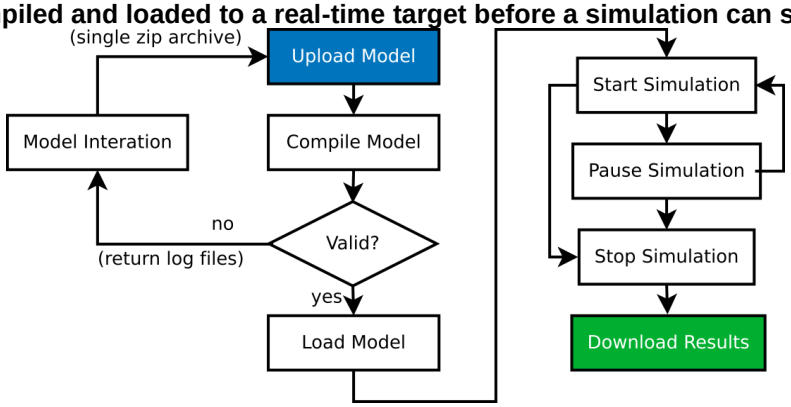

Figure **13** shows a flow which models this procedure. Both OPAL-RT and RTDS require proprietary toolchains for model compilation and simulator control. Instead of using the graphical user interfaces of these tools, VILLAScontroller makes use of APIs which are provided by the vendors to perform the actions programmatically.

The controller abstracts this process by handling the models and result files as opaque archive files whose content depends on the actual simulation tool. If a user does not have direct access to the vendor tools and/or simulator, it is very likely that the models are flawed and several iterations are required.

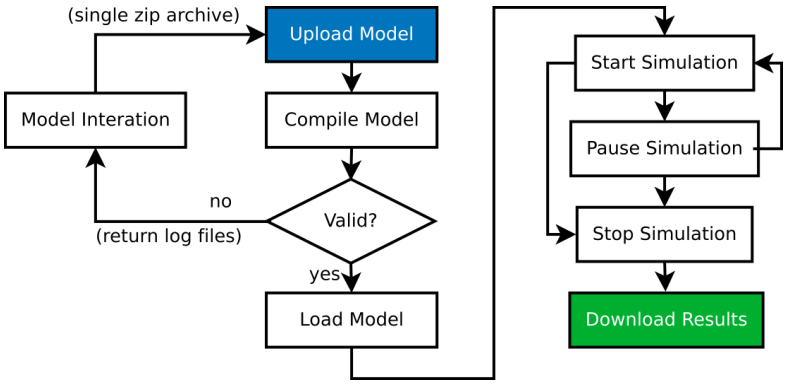

Figure 13: Simulator states.

6.3 Synchronization

The start of a simulation run is synchronized by a global schedule which is distributed to all simulators in advance. After loading the models, each simulator waits for the previously agreed global time of commencement before starting the execution. It is important that this stand-by state is implemented on the real-time target itself in order to avoid unpredictable latencies introduced by the connection between the controller and the target.

To increase timing accuracy, it is advisable to synchronize the time of the simulator to global time sources like GPS. This also prevents drifting of the simulator clocks.

In task 4.3 of the RESERVE project, three simulation platforms will be used:

- RTDS racks will be synchronized with a GTSYNC extension card to a GPS, IRIG-B, PPS or PTP (IEEE 1588) time source.
- OPAL-RT targets can be synchronized using a Spectracom TSync PCIe card to a GPS, IRIG-B or PPS signal.
- DPsim solver instances can be synchronized via PTP (IEEE 1588) using the Linux PTP project [14], [15].

7. Conclusion

This task implements software tools and an associated workflow to perform geographically distributed co-simulation at a pan-European scale.

Original ideas devised from the VILLAS project have been developed further. VILLASframework has been documented and released as open source software under the GNU General Public License Version 3 (GPLv3). Detailed documentation and source code are now available on the project website[6]. We consider the flexibility of these tools and their release to the public one of the contributions of this work package as it will enable future projects to reuse this work.

Other projects like the Global RT-SuperLab, a larger co-simulation scenario between US and EU research institutions or the ERIClab demonstration [4] have already adopted VILLASframework.

7.1 Outlook

The development of VILLASframework is progressing continuously and expected to be completed until the end of the year. The implementation of VILLASnode was accomplished in July 2017. The current work focus is on the development of the web interface and the simulator controller.

First benchmark results are expected to be available with the commencement of task T4.3. Upcoming T4.3 will deploy this framework at four sites across Europe to conduct first communication tests and simulations. Politecnico di Torino (POLITO) will participate by providing simulation capacity using their OPAL-RT simulator. University Politehnica of Bucharest (UPB) is projected to simulate a subsystem with the Open Source Dynamic Phasor Solver developed by RWTH in task T4.2. The Waterford Institute of Technology (WIT) will test interoperability of the new framework with the Open Field Message Bus project (OpenFMB) in order to integrate measurements from SCADA devices as well as the integration of controllers developed in work packages WP2 and WP3 [16]. RWTH Aachen University is hosting the 5G testbed which will be integrated into the pan-European real-time co-simulation during tasks T4.4 - T4.5 and T5.3 - T5.5. Their RTDS real-time simulator will act as a central hub for all other sites.

[6] http://fein-aachen.rwth-aachen.de/projects/villas-framework/

8. References

[1] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu, "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 621–628.

[2] Vogel, S., "VILLASframework," *A Modular co-simulation framework*. [Online]. Available: http://www.fein-aachen.org/projects/villas-framework/. [Accessed: 30-May-2017].

[3] "VILLAS – Virtually Interconnected Laboratories for LArge systems Simulation/emulation - RWTH AACHEN UNIVERSITY Institute for Automation of Complex Power Systems - Deutsch." [Online]. Available: http://www.acs.eonerc.rwth-aachen.de/cms/E-ON-ERC-ACS/Forschung/Forschungsprojekte/Gruppe-Real-Time-Simulation-and-Hardware/~krkg/VILLAS-Virtually-Interconnected-Labora/. [Accessed: 30-May-2017].

[4] "ERIC-lab | European Real-time Integrated Co-simulation laboratory." [Online]. Available: http://www.eric-lab.eu/. [Accessed: 17-Jul-2017].

[5] S. Anthony, "The secret world of microwave networks," *Ars Technica*, 03-Nov-2016. [Online]. Available: https://arstechnica.com/information-technology/2016/11/private-microwave-networks-financial-hft/.

[6] R. Mall, *Real-Time Systems: Theory and Practice*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[7] 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 *PowerTech, 2015 IEEE Eindhoven*, 2015, pp. 1–5.

[8] M. Stevic *et al.*, "Virtual integration of laboratories over long distance for real-time co-simulation of power systems," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 6717–6721.

[9] C. Steinbrink *et al.*, "Simulation-based Validation of Smart Grids - Status Quo and Future Research Trends," presented at the 8th International Conference on. Industrial Applications of Holonic and Multi-Agent Systems, 2017.

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

[11] S. Fey, P. Benoit, G. Rohbogner, A. H. Christ, and C. Wittwer, "Device-to-device communication for Smart Grid devices," in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, 2012, pp. 1–7.

[12] 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*, 2010, vol. 109.

[13] 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. 005020–005025.

[14] K. Ichikawa, "Precision Time Protocol on Linux," presented at the LinuxCon, Japan, 2014.

[15] R. Cochran, C. Marinescu, and C. Riesch, "Synchronizing the Linux system time to a PTP hardware clock," in *Control and Communication 2011 IEEE International Symposium on Precision Clock Synchronization for Measurement*, 2011, pp. 87–92.

[16] "OpenFMB™ - Collaboration Site." [Online]. Available: https://openfmb.github.io/#overview. [Accessed: 16-Jul-2017].

[17] M. O. Faruque, V. Dinavahi, M. Sloderbeck, and M. Steurer, "Geographically distributed thermo-electric co-simulation of all-electric ship," in *2009 IEEE Electric Ship Technologies Symposium*, 2009, pp. 36–43.

[18] E. Bompard *et al.*, "A multi-site real-time co-simulation platform for the testing of control strategies of distributed storage and V2G in distribution networks," in *2016 18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe)*, 2016, pp. 1–9.

9. List of Abbreviations

ACS	Institute for Automation of Complex Power System
AMQP	Advanced Message Queuing Protocol
BSD	Berkley Software Distribution
CPS	Cyber-Physical System
CSV	Comma-separated Values
DRTS	Digital Real-time Simulator
FPGA	Field-programmable Gate Array
GTFPGA	RTDS Giga-transceiver FPGA
GUI	Graphical User Interface
HFT	High Frequency Trading
HiL	Hardware in the Loop
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
IA	Interface Algorithm
IP	Internet Protocol
NREN	National research and education network
OpenFMB	Open Field Messaging Bus
P2P	Point-to-Point or Peer-to-Peer
PTP	Precision Time Protocol (IEEE 1588)
QoS	Quality of Service
RMS	Root Mean Square
RTDS	Real-time Digital Simulator
SV	Sampled Values (IEC 61850-9-2)
TFR	Time–frequency representation
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VILLAS	Virtually Interconnected Laboratories for LArge systems Simulation/emulation
VPN	Virtual Private Network
WYSIWYG	What You See Is What You Get

10. List of Figures

11. List of Tables

12. Annex

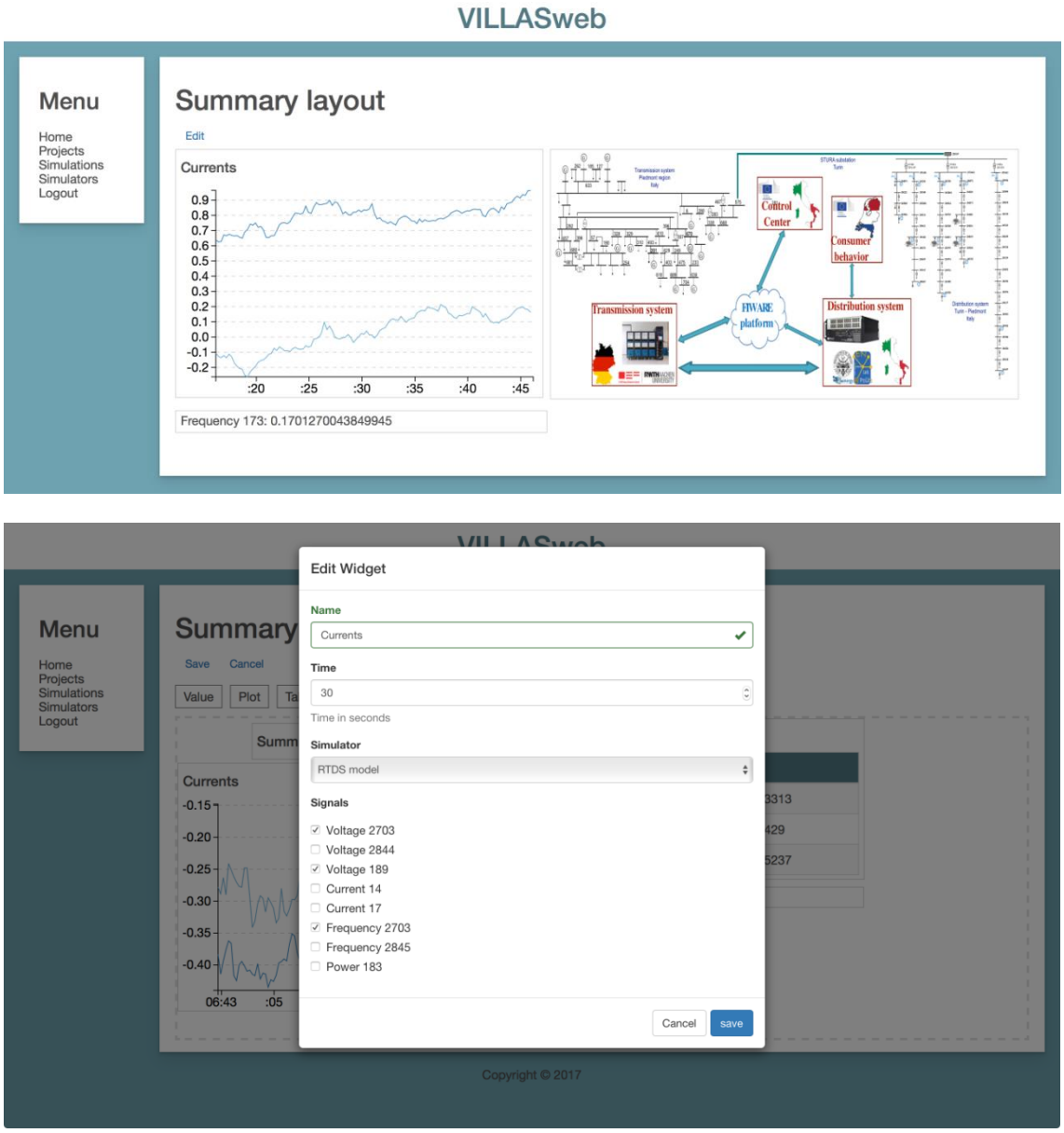

Figure 14: VILLASweb widget configuration dialog.

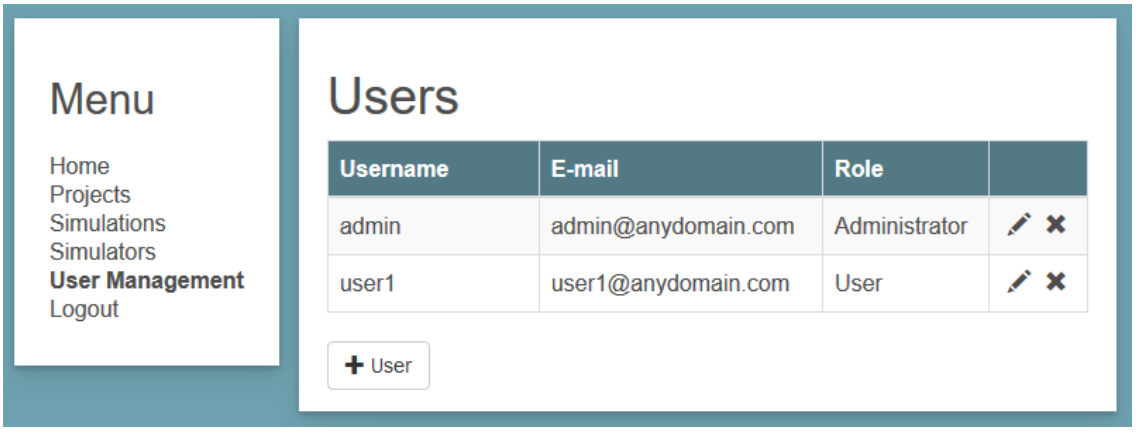

Figure 15: User / Access management.

Name	Description	Dependencies	Status
cbuilder	Cross-compiled RTDS CBuilder models	—	Deprecated[7]
opal	OPAL-RT Asynchronous Process	libOpal*	Deprecated[8]
file	Support for file log / replay node type	libc	Stable
shmem	POSIX shared memory interface with external processes	—	Stable
loopback	Internal loopback using a queued buffer	—	Stable
signal	Configurable signal generator for testing	—	Stable
stats	Send communication statistics to other nodes	—	Stable
test_rtt	Test automation for round-trip time, packet loss, sending rates	—	Stable
socket	BSD network sockets for Packet, IP or UDP layer	libnl3	Stable
ngsi	OMA Next Generation Services Interface 10	libcurl, libjansson	Stable
websocket	Send and receive samples of a WebSocket connection	libwebsockets	Stable
zeromq	ZeroMQ publish / subscribe messaging	libzmq	Stable
nanomsg	nanomsg publish / subscribe messaging	libnanomsg	Stable
fpga	VILLASfpga PCIexpress card	libxil	Beta
gtwif	RTDS GTWIF Workstation Interface	—	Beta
amqp	Advanced Message Queuing Protocol	rabbitmq-c	Planned
mqtt	MQ Telemetry Transport	libmosquitto	Planned
iec61850-9-2	IEC 61850-9-2 Sampled Values	libiec61850	Planned
iec61850-8-1	IEC 61850-8-1 GOOSE Telegrams	libiec61850	Planned

Table 1: List of supported VILLASnode node types.

Name	Description
restart	Trigger restart hooks if simulation restart is detected
ts	Overwrite origin timestamp of samples with receive timestamp
fix_ts	Update timestamps of samples if not set
stats	Collect statistics for the current path
decimate	Down-sampling by integer factor
skip_first	Skip the first samples based on sequence number or timestamp
drop	Drop messages with reordered sequence numbers
convert	Convert message from / to floating-point / integer
shift_seq	Shift sequence number of samples
map	Remap values and / or add header, timestamp values to the sample
print	Print the message to standard output
shift_ts	Shift timestamps of samples by an offset

Table 2: List of supported VILLASnode hook functions.

[7] The cbuilder node-type is running the CBuilder control system model in the same process as VILLASnode.

[8] The `opal` node-type is not synchronized to simulation time-step. A new interface based on the shmem node-type is planned to fix this.

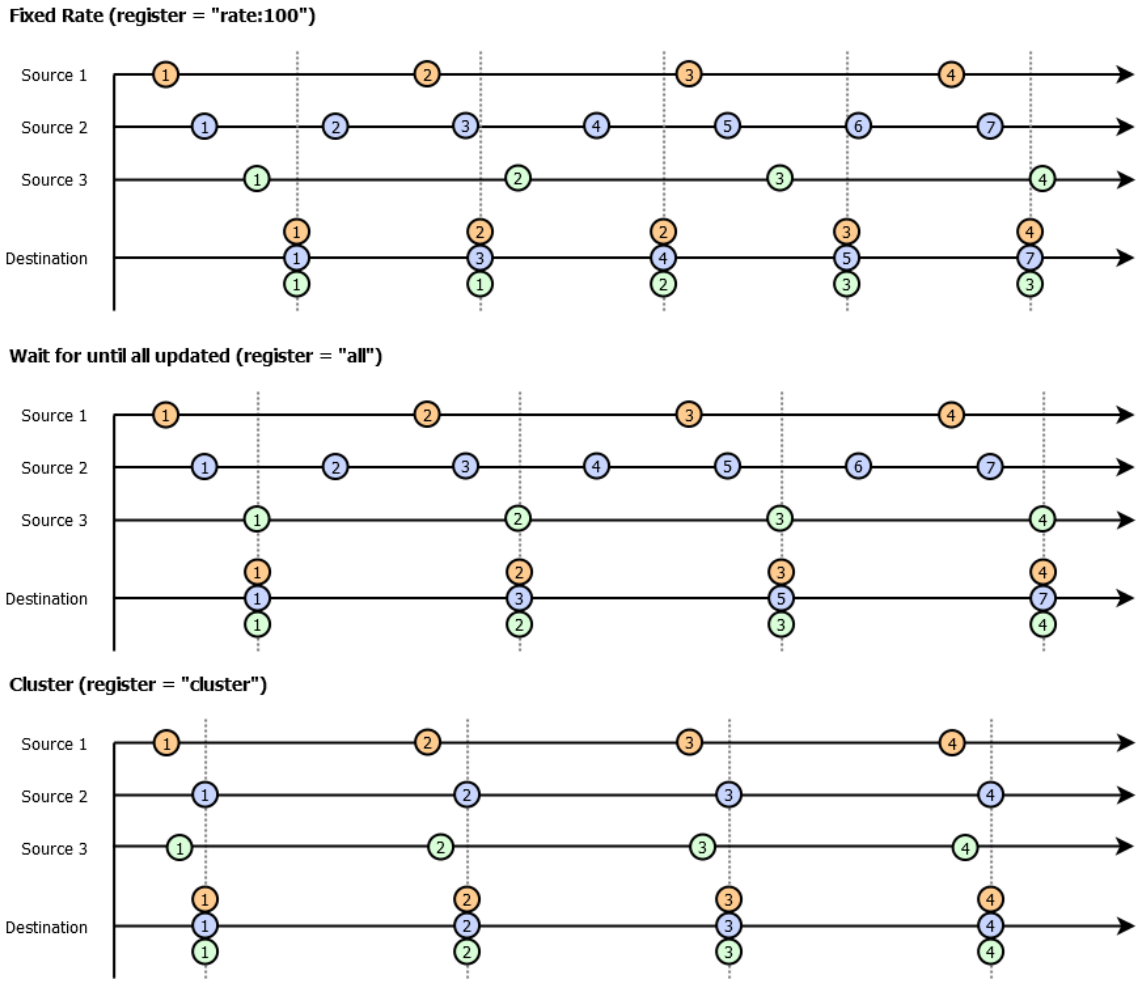

Figure 16: More VILLASnode register modes.