

Lawrence Berkeley National Laboratory

LBL Publications

Title

Cyber—Physical Modeling of Distributed Resources for Distribution System Operations

Permalink

<https://escholarship.org/uc/item/3ej5s95t>

Journal

Proceedings of the IEEE, 104(4)

ISSN

0018-9219

Authors

Chatzivasileiadis, Spyros

Kiliccote, Sila

Bonvini, Marco

et al.

Publication Date

2016

DOI

10.1109/jproc.2016.2520738

Peer reviewed

Cyber physical modeling of distributed resources for distribution system operations

Spyros Chatzivasileiadis, *Member, IEEE*, Marco Bonvini, Javier Matanza, Rongxin Yin, Zhenhua Liu, Thierry Noudui, Emre C. Kara, Rajiv Parmar, David Lorenzetti, Michael Wetter, and Sila Kiliccote, *Member, IEEE*

Abstract—Co-simulation platforms are necessary to study the interactions of complex systems integrated in future smart grids. The Virtual Grid Integration Laboratory (VirGIL) is a modular co-simulation platform designed to study interactions between demand response strategies, building comfort, communication networks, and power system operation. This paper presents the coupling of power systems, buildings, communications and control under a master algorithm. There are two objectives. First, to use a modular architecture for VirGIL, based on the Functional Mock-up Interface (FMI), where several different modules can be added, exchanged, and tested. Second, to use a commercial power system simulation platform, familiar to power system operators, such as DigSILENT Powerfactory. This will help reduce the barriers to the industry for adopting such platforms, investigate and subsequently deploy demand response strategies in their daily operation. VirGIL further introduces the integration of the Quantized State System (QSS) methods for simulation in this co-simulation platform. Results on how these systems interact using a real network and consumption data are also presented.

Index Terms—Co-Simulation, Functional Mock-up Interface, Modelica, Demand Response, Load Flow, DigSILENT Powerfactory, OMNET++

I. INTRODUCTION

Moving towards “smarter” grids, power systems complexity increases through the embedding of communication networks, demand side management, electric vehicles, and the stochastic nature of several renewable energy sources (RES). Simulation platforms specialized in power systems can no longer handle in an adequate way the increasing interdependencies with systems such as communications, buildings, and electric vehicles. More detailed simulation tools are necessary to study the system interdependencies and determine the appropriate control strategies for optimizing power system operation. An option is to extend the existing power system simulation tools by incorporating the dynamics of such networks inside the same simulation platform. On the other hand, research in the respective fields has developed highly detailed and reliable tools, which can simulate the behavior and control of such systems. This paper adopts the co-simulation approach, where highly developed and reliable simulation tools, specialized in the respective fields, are merged in a common co-simulation platform to study the interdependencies between systems and identify appropriate control strategies.

Although co-simulation has found a lot of applications in e.g. the automotive industry or building controls (e.g. BCVTB [1]), in power systems it is a relatively recent field which has seen some development during the last 8 years. The approach followed in this paper is to couple a commercial power system simulation platform, widely used by power system operators, with advanced modeling tools for buildings and communication networks. The goal is to determine the impact the demand response strategies have on the network and determined optimal algorithms to utilize flexible loads for power system operation.

Two are the main objectives. First, we aim at reducing the barriers for adoption of novel demand response and other control strategies in the daily power system operation. Coupling a trusted power system simulator, with which several power system operators are familiar, with other advanced modeling tools will help towards a wider adoption of such tools. Testing and becoming familiar with the impact of different strategies on the power system will allow the wider deployment and utilization of the energy reserves “stored” in buildings, e.g. in the form of thermal inertia. Second, we need a modular co-simulation architecture, which will allow the easy exchange and test of different simulation modules, as well as the easy extension with e.g. electric vehicle simulators, optimization tools, hardware-in-the-loop, etc. For this reason, we use the Functional Mock-up Interface standard, which provides a standardized interface for the coupling of several different tools.

This paper describes the Virtual Grid Integration Laboratory (VirGIL), which couples a commercial power system simulator with models for buildings and communication networks. The goal is to estimate the impact of demand response strategies on the grid, and to determine optimal algorithms for exploiting flexible loads (for example, the thermal energy stored in buildings). To this end, we describe a modular co-simulation architecture that allows the easy exchange of different simulation models, as well as the easy extension with, e.g., electric vehicle simulators, optimization tools, hardware-in-the-loop, and so on.

VirGIL’s architecture is based on the Functional Mockup Interface, which defines a standard interface for exposing the capabilities of a simulation tool [2]. FMI provides for a modular structure that allows the simple exchange and testing of different co-simulation tools. VirGIL is implemented in the Ptolemy II framework, which combines continuous and discrete-event simulation [3].

This paper is organized as follows. Section II reviews existing co-simulation methods in power systems. Section III provides the overview of the proposed co-simulation architecture in VirGIL, while Section IV describes the Functional Mock-up Interface (FMI). Sections V, VI, and VII, VIII present respectively the development of the Power Systems, Buildings, Communications, and Control Functional Mock-up Units (FMUs). Sections IX and X describe the simulation of the model exchange FMUs based on the QSS algorithm and the operation of the master algorithm respectively. Section XI describes simulation results based on real data for the LBNL network. Finally, Section XII concludes this paper and provides an outlook for future extensions of this work.

II. CO-SIMULATION IN POWER SYSTEMS

Over the last years, several cosimulation approaches for power systems have been developed and documented in the literature. One of the first documented efforts is [4], which co-simulates power and communications systems. The authors advocate the use of already existing simulation tools that excel in their respective fields instead of creating new simulation platforms (“federated approach”). In that work, power systems are simulated with fixed step through PSCAD/EMTDC and PSLF while communications simulations are carried out on the discrete event simulator ns-2. The two tools are synchronized at specific “synchronization points” without an implementation for a rollback function, which results in accumulation of synchronization induced inaccuracies over time. The authors improved this approach in [5] where they use a master algorithm with a common timeline for both modules. There are no “synchronization points”, instead both simulators evolve synchronously in time.

Most of the co-simulation approaches for power systems combine power system with communication network simulation (examples for distribution networks are [6], [7]). Ref. [8] reports a co-simulation approach for power systems and EV charging and control, where they also use FMI for the coupling of one of the simulation tools to the master algorithm. A survey of the latest simulation tools that are used for co-simulation in power systems is reported, among others, in [9].

This work focuses on the interactions of building models for energy consumption with distribution system models for power system operation.

Among the tools used for co-simulation, Gridlab-D is probably one of the most widespread [10]. It has a flexible environment, which incorporates advanced modeling techniques, efficient simulation algorithms, but most importantly provide a simulation environment not only for power systems, but also incorporating detailed load modeling, rate structure analysis, distributed generator and distribution automation.

In this paper, a commercial power system software, DigSilent Powerfactory, is used for power systems simulation. Building a co-simulation platform incorporating Powerfactory, a tool that several utilities trust and use in their daily operation, decreases the barriers for wider adoption of co-simulation tools from the industry. Power system operators can incorporate their version of Powerfactory with the co-simulation platform

to investigate in more detail the effect of demand response signals, decide and subsequently deploy the most appropriate in real-time operation. Powerfactory has the additional advantage of being capable to model both AC and DC systems. A co-simulation approaches incorporating Powerfactory has also been documented in [11]. However, this is the first time that a modular co-simulation architecture, based on FMI, is implemented for coupling Powerfactory with the rest of the simulation tools.

Besides the development of the appropriate models and controls within each simulation tool, the focus in this paper is on the development of the wrapper functions which will make the modules compatible to the FMI standard for co-simulation. FMI provides for a modular structure of the co-simulation platform which allows the simple exchange and testing of different co-simulation tools. VirGIL’s master algorithm will be Ptolemy II, which can combine both continuous and discrete-event simulation. At the same novel simulation algorithms are implemented in Ptolemy II, such as QSS (Quantized-State-Simulation) which allow for higher efficiency and faster execution times.

III. VIRGIL

VirGIL (Virtual Grid Integration Laboratory) is a modular co-simulation platform that currently couples models of power systems, buildings, communications, and has the potential to integrate other sources or sinks on the electrical grid, including electric vehicles. The platform will facilitate developing novel control algorithms, and optimizing power systems, buildings, communications and EV charging.

Fig. 1 shows an overview of the VirGIL co-simulation architecture.

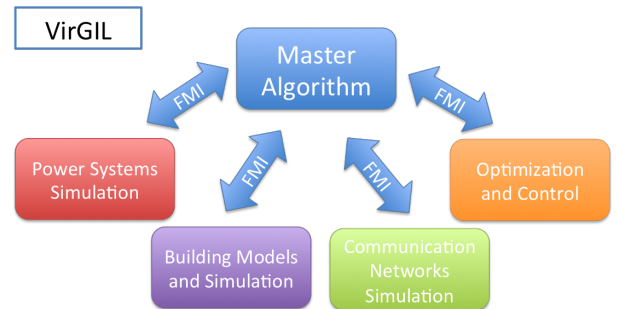


Fig. 1: Overview of the VirGIL co-simulation architecture. Modules include tools for simulating power systems, buildings, and communications.

Figure 2 presents in a schematic way the interactions between the different VirGIL blocks and the potential inputs and outputs of VirGIL. The communications modules, which is fully integrated into VirGIL, is represented in this figure as red blocks of communication delays.

IV. FUNCTIONAL MOCKUP INTERFACE (FMI)

VirGIL’s master algorithm coordinates all modules through the Functional Mockup Interface (FMI) [2]. FMI defines a tool-independent standard for exchanging models and running

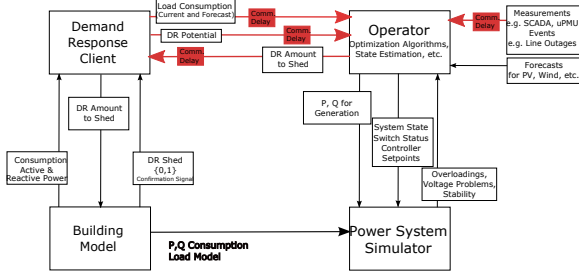


Fig. 2: Interactions between VirGIL Modules

standalone simulation tools. In principle, this allows VirGIL to integrate any FMI-compliant tool. For example different power system simulation tools can be exchanged and tested without requiring changes to any other simulation module, or to the master algorithm.

A simulation model exported according to the FMI standard is called a Functional Mockup Unit (FMU). An FMU is a zip file that contains the source or object code needed to execute a model, plus text files that describe the model's capabilities. The FMU may also contain additional resources, such as documentation and auxiliary input files.

The FMI standard distinguishes between Model Exchange and Co-Simulation. See Fig. 3. The FMI for model exchange represents a dynamic component directly, using differential, algebraic, and discrete equations. Therefore the master algorithm must provide the necessary solvers. By contrast, the FMI for co-simulation defines an interface for coupling independent simulation tools. Under co-simulation, the FMU itself provides the associated solvers. In both cases, the master algorithm coordinates time, and exchanges inputs and outputs, between FMUs.

VirGIL can integrate both types of FMU. For example, the power system model uses the FMI for Co-Simulation, while the buildings model uses the FMI for Model Exchange.

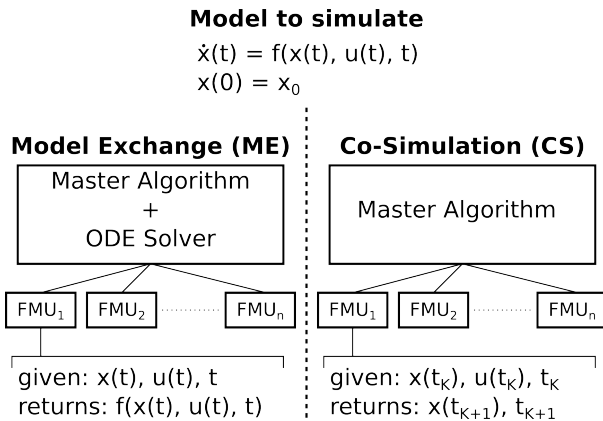


Fig. 3: FMI for Model Exchange and FMI for Co-Simulation

V. POWER SYSTEMS FMU

We chose DigSILENT PowerFactory after reviewing several power system software packages. The main focus was on established commercial power system software, in order to demonstrate how co-simulation enables the use of familiar specialized simulation tools. For this project, PowerFactory's scripting interfaces, for example to C++, C#, Python, and Matlab/Simulink, made it especially attractive. In our implementation, all VirGIL modules, except for the power systems part run on Linux. DigSILENT Powerfactory runs only on Windows. As a result, we implemented a socket communication between Windows and Linux. In Linux the PowerFMU implements all functions necessary for the FMI standard and calls their counterpart in the Windows implementation through the socket. The Windows FMU, in turn, calls the Python functions that start and stop PowerFactory, parameterize the simulation, set the inputs, and get the outputs.

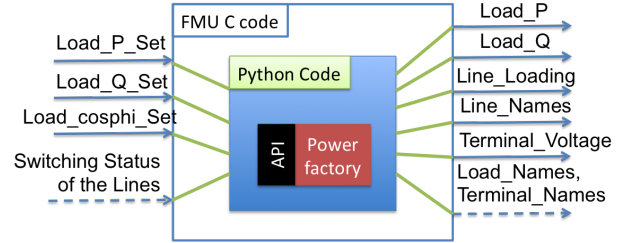


Fig. 4: An FMU wrapper for PowerFactory. The arrows represent variable names.

Fig. 4 shows the structure of the FMU for power systems simulation. The FMU maps the C-language functions defined in the FMI standard to calls on PowerFactory's Python API. For example:

- *fmi2Instantiate()*: start PowerFactory, Activate Project.
- *fmi2SetReal()*, *fmi2SetInteger()*, *fmi2SetString()*: Set the values of variables and parameters.
- *fmi2DoStep()*: execute load flow.
- *fmi2GetReal()*, *fmi2GetInteger()*, *fmi2GetString()*: Get the values of variables and parameters.

VirGIL's initial focus is on the impact of demand response algorithms in the power system steady-state operation, e.g., to investigate line loadings and voltage profiles. Thus the Power Systems FMU runs several sequential load flows, and determines the state of the system after each run. Extending the FMU to handle dynamic simulations is an object of future work.

VI. BUILDINGS FMU

To study how demand response affects the distribution grid, VirGIL requires a building model that can capture the relevant dynamics, without placing undue computational burden on the overall simulation. For example, the model should have sufficient detail to show the effect of DR strategies such as changing temperature setpoints, or reducing fan speeds.

Building energy performance depends on the interaction between many heterogeneous elements, e.g., the envelope, windows, lighting, controls, and the heating ventilation and

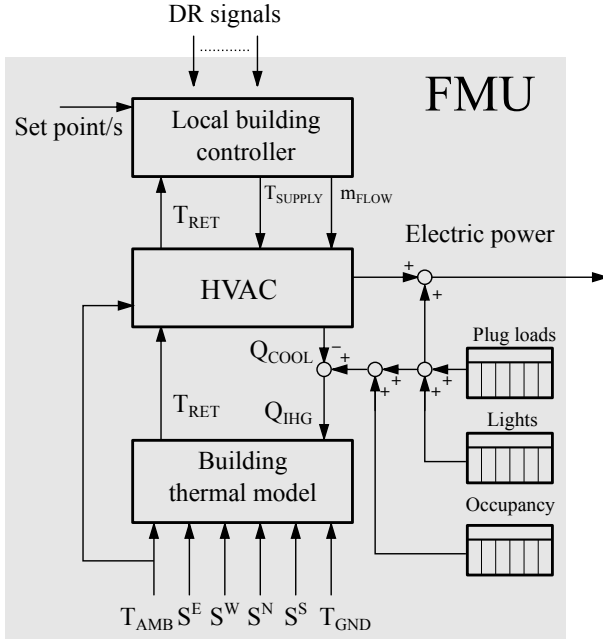


Fig. 5: Overview of the VirGIL building FMU model. The model comprises four main parts: the building thermal model, the HVAC system, the schedules and the control system.

air-conditioning (HVAC) systems. To represent these elements, the building model used in VirGIL comprises four main parts, as shown in Fig. (5): (1) the thermal system that describe the envelope, windows, interior slabs and partitions, and room air; (2) the HVAC systems (e.g., air handling units, fans, etc.); (3) a set of schedules that describe thermal/electric loads such as lights, plug loads and internal heat gains generated by occupants; (4) and the building control systems that manages the HVAC and other assets in order to maintain the comfort levels and receives DR signals.

While VirGIL could incorporate EnergyPlus models directly, using its FMI interface for Co-Simulation [?], a complete EnergyPlus model is too detailed to simulate all the buildings in a complete distribution system. The Building Resistance-Capacitance Modeling (BRCM) Toolbox [12] provides an alternative to overcome the computational burden of a full-building simulation model such EnergyPlus. The BRCM toolbox constitutes a part of the process for creating a building FMU model. Fig. (6) describes the end-to-end process for generating a building FMU model. The following subsections provide a detailed description of each step of the process.

A. Generating the EnergyPlus model

An EnergyPlus whole-building energy simulation model is the first step towards the creation of a simplified building model used in VirGIL. Creating a detailed EnergyPlus model can be a time consuming task, for such a reason the EnergyPlus model have been generated using the Demand Response Quick Assessment Tool (DRQAT) [13]. Alternately, one can use prototypical models [14]. The generated EnergyPlus model contains the entire description of building geometry and other

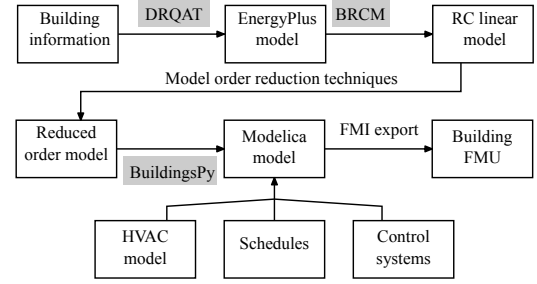


Fig. 6: Description of the end-to-end process for generating the building FMU (Grey boxes are toolboxes or packages used in the process).

physical properties such as the conductivity of the wall layers, their thermal capacitances, the solar heat gain coefficients of the windows, etc. These information will then be used to generate a simplified first-principle model of the building.

B. Converting the EnergyPlus model to RC model

The Building Resistance-Capacitance Modeling (BRCM) toolbox allows to converts an EnergyPlus description of a building's materials and geometry, to a lumped-capacity RC network that accounts for first-principle physical properties. Examples of these properties are the thermal mass and the effect of solar radiation.

For each thermal zone that is described in the EnergyPlus model the BRCM toolbox generates a RC network as shown in Fig. (7). For each zone i the generated RC network contains the thermal capacitance of the air C^i , the thermal capacitance of the internal mass present in the zone C_{IM}^i , and a series of thermal capacitances and resistances for each of the N layers of the k walls surrounding the zone C_W^{kmi} . The heat fluxes q^i , q_{IM}^i , q_{INT}^{ki} , and q_{EXT}^{ki} respectively represent the internal heat gains of the zone (e.g. due to occupants, solar radiation, etc.), the internal heat gains heating the internal mass, the fraction of solar radiation directed to the innermost layer of the walls, and the fraction of solar radiation directed the outermost layer of the walls.

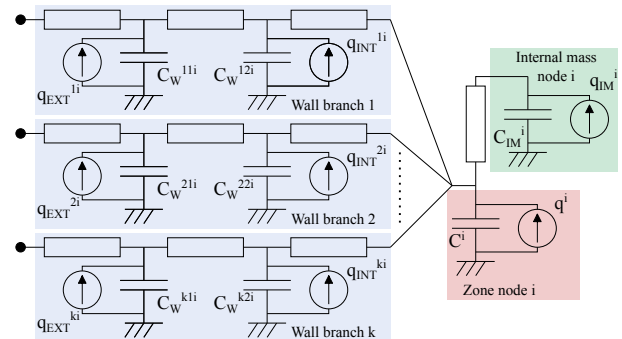


Fig. 7: RC network for a generic zone i . Capacitances represent states (i.e., Temperatures), resistances represent thermal resistances, and current sources represent heat fluxes.

Once the RC network model has been parametrized the model can be written in the following form

$$\dot{x}(t) = Ax(t) + B_u u(t) + B_v v(t) \quad (1a)$$

$$y(t) = Cx(t) + D_u u(t) + D_v v(t) \quad (1b)$$

where $x(\cdot) \in \mathbb{R}^n$ is the state vector containing all the temperatures of the zones, internal masses and wall layers; $u(\cdot) \in \mathbb{R}^m$ is the input vector (e.g., control inputs), $v(\cdot) \in \mathbb{R}^p$ are the predicted disturbances (e.g., external air temperature, solar radiation, internal heat gains, etc.), and $y(\cdot) \in \mathbb{R}^o$ is the output vector.

C. From RC model to reduced order model

The linear model describing the first-principle RC network constitutes a first simplification of the whole-building model. VirGIL requires a model that is detailed enough to correctly capture the thermal dynamics of the buildings and correctly predicts the impact they have on different DR strategies. For such a purpose the model in (1) needs to be further simplified.

Before starting the simplification it's important to define the outputs to be controlled and the input control variables needed to do so. As shown in Fig. (5) the building thermal model computes the temperature of the air in the zones that is then returned to the HVAC system (T_{RET}). The local controller controls the HVAC system in order to maintain the temperature of the air in the building as close as possible to the desired set point. The HVAC system model computes the cooling power to be delivered to maintain the zones temperatures at the desired set point.

This description allows to introduce two simplifications. First, the HVAC and the control system are not part of the building thermal model. They interact with a suitable representation of the building that given the internal heat gains and the other known disturbances computes the return temperature. This allows to remove the HVAC inputs $u(\cdot)$ from the model in (1). Second, the output vector $y(\cdot)$ is equal to the return temperature T_{RET} , that is the weighted average of the thermal zones temperature. After introducing such simplifications the model (1) can be rewritten as

$$\dot{x}(t) = Ax(t) + B_v v(t) \quad (2a)$$

$$y(t) = Cx(t) \quad (2b)$$

$$C = \left(\frac{V_1}{V_{TOT}} \quad \dots \quad \frac{V_{nz}}{V_{TOT}} \quad 0 \quad \dots \quad 0 \right) \quad (2c)$$

where $C \in \mathbb{R}^1 \times \mathbb{R}^n$ is the output matrix, nz is the number of thermal zones (the first nz elements of the state vector $x(\cdot)$), V_i for $i \in [1, nz]$ is the volume of the i -th thermal, and $V_{TOT} = \sum_{i=1}^{nz} V_i$ is the sum of all the volumes. The vector of known disturbances $v(\cdot)$ and outputs $y(\cdot)$ are thus defined as

$$v(\cdot) = \left(Q_{IHG} \quad T_{AMB} \quad T_{GND} \quad S^E \quad S^W \quad S^N \quad S^S \right)^T$$

$$y(\cdot) = (T_{RET})$$

Despite the number of input-output relationship of the model (2) is seven, the number of state variables can be high enough that the simulation speed remains an issue (e.g., a model with ten zones can easily have more than hundred states). For such a reason the model can be further

reduced [15], [16]. The resulting model will have a closely match of the input-output behaviour while reducing the number of states.

D. Conversion to Modelica and generation of the FMU

Once the reduced order model that defines the input-output relationship between the known disturbances and the output is defined, it's possible to express it using Modelica, an object-oriented, equation-based language for modeling multi-domain physical systems.

Then, drawing on the Modelica Buildings Library [17], we add the HVAC, loads, and controls logic. These components predict the active power consumption of the building, and implement a demand response system that adjusts the zone temperatures and airflow setpoints according to DR signals sent by the utility.

Finally, we export the Modelica building model as an FMU for Model Exchange.

VII. COMMUNICATIONS FMU

With respect to the communication modeling, OMNeT++ [18] was chosen among a number of simulation tools. This open-source discrete event environment is a general communication simulator widely used in the research and academic community. In this framework, a basic model is built in a hierarchical manner: first the behavior of simple modules is described in C++; then, these modules are instantiated and tied together using OMNeT++'s Network Description Language (NED) in order to form more complex entities.

Since OMNeT++'s main classes are mainly focused on the implementation of the discrete event machine and the simulation scheduler, it is common to add a number of extensions to the framework in order to upgrade the capability of the model. This is the case of the INET framework, which includes support for IPv4, IPv6, TCP, Ethernet, HTTP and many other used protocols within the Internet. Additionally, there exist other frameworks that implement mobility scenarios (like VNS), wireless sensor network (like WiXiM or Castalia), LTE technology (like SimuLTE), etc. INET counts the all the technologies that are needed for current version of VirGIL.

Other simulator options considered were: ns-2/ns-3 (Network Simulator 2 / Network Simulator 3), JiST (Java in Simulation Time) and OPNET Modeler [®]. Among all of them, OMNeT++ and ns-2/ns3 have extensively been used in co-simulation application for Smart Grids scenarios [19]–[21]. There are several reasons that led us to choose OMNeT++, some of which are detailed in the following lines. OMNeT++ counts with a integrated development environment (IDE) adapted to Eclipse which facilitates debugging and topology creation tasks; other simulators, such as ns-2/ns-3; do not provide any kind of GUI, making debugging a very tedious task. In addition to this, OMNeT++ counts with an extensive and detailed documentation. It does not only provide information for the first steps in running a very generic simulation, but also include specific details in order to built onto the core classes for customized models. In fact, there is a specific section in the documentation on how to embed the

simulation kernel into other applications, which is very helpful for implementing a co-simulation framework such as VirGIL. Once again, ns-2/ns-3 lacks organized documentation for the simulator’s code.

From a more technical perspective, OMNeT++ has shown good agreement with measured data for a number of communication technologies. This is the case of WiFi (IEEE802.11g) or LTE, as reported in [22] and [23] respectively. In terms of performance, both ns-2/ns-3 and OMNeT++ have a similar performance and offer good scalability features, as discussed in [24].

Regarding the demand response application under study, the model counts with three high-level types of actors: server nodes, where information about DR events is stored; client nodes, which try to retrieve this information; and a network, that interconnects all nodes. From a logical perspective, the DR communication infrastructure can be built using these three actors.

Both clients and servers in the network under study will implement Open Automated Demand Response (OpenADR) as an application layer protocol for exchanging messages. OpenADR is a standardized communications data model for sending and receiving DR signals from a utility or independent system operator to electric customers [25], [26].

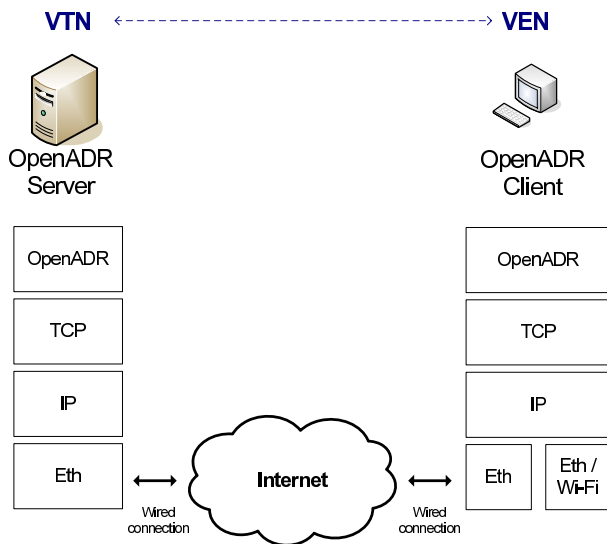


Fig. 8: Communication’s layer stack diagram.

Figure 8 shows a more detailed scheme of the model. The three already mentioned actors can be seen in the figure: an OpenADR Server, an OpenADR Client and an interconnected network (in this case the Internet). Additionally, the figure also shows the implementation of the different communication layers on each of the nodes. In this case, the *de facto* Internet’s layer stack is chosen: TCP as a transport protocol, IP as network protocol and Ethernet as a physical protocol. OpenADR servers and clients use the lower layers to transmit their information. This layered structure, in practice produces a virtual direct communication between pairs of layers.

Additionally, Figure 8 shows the names that OpenADR’s specification gives to the different nodes in the network:

Virtual End Nodes (VEN) and Virtual Top Nodes (VTN). Information flows from VTN to VEN. Additionally, a VEN may also behave as a VTN in order to forward certain data to other nodes.

The implementation of the network is shown in Figure 9. It counts with a DR Server (labeled with `serv`), a DR Client (labeled with `cli[0]`), a number of routers and a cloud network. This scheme represents the communication of both nodes in an interconnected wired network such as the Internet. The figure only shows one client for clarity reasons; however, the number of clients is a parameter for the model. In case that more than one should exist, each one of them would have its own router to connect to the cloud.

Routers pretend to simulate the gateway that each ISP (Internet Service Provider) would provide to a customer in order to connect to the Internet; as such, routers only implement up to layer-3 capabilities. The cloud network models the Internet as a network with a variable delay, transmission speed and error rate. Recalling the classification made in [21] about the network model’s level of detail: the Internet would be modeled as a *black-box* communication network, whereas the rest of the entities would count with a high level of detail (i.e. all layers and communication processes are taken into account).

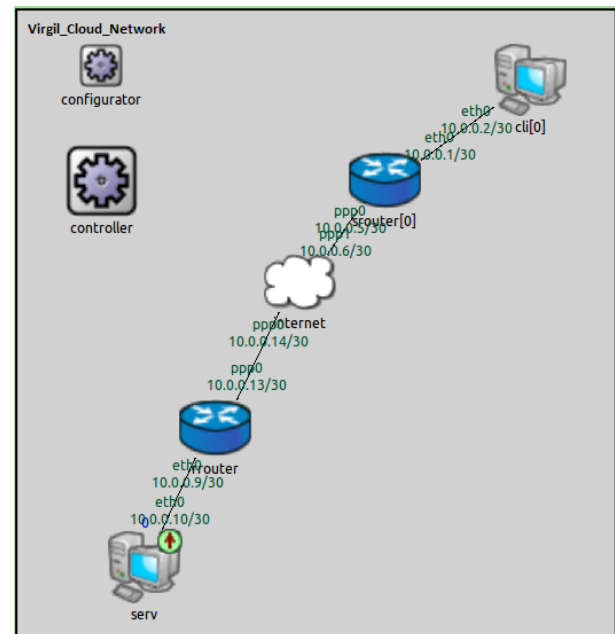


Fig. 9: OMNeT++ implementation of the network.

The structure of the FMU for the communications model is shown in Figure 10. The FMU acts as an interface for the OMNeT++’s API. This API talks directly to the simulation kernel in order to set or get certain variable’s values or messages. As mentioned before, the kernel implement’s some functions needed in the simulation like the message scheduler or the discrete event machine. However, in order to model the Internet layer stack shown in Figure 8, the INET library is used. In addition to INET, the kernel also uses an additional library developed for this study where other capabilities (such as the OpenADR Server and Client) have been implemented.

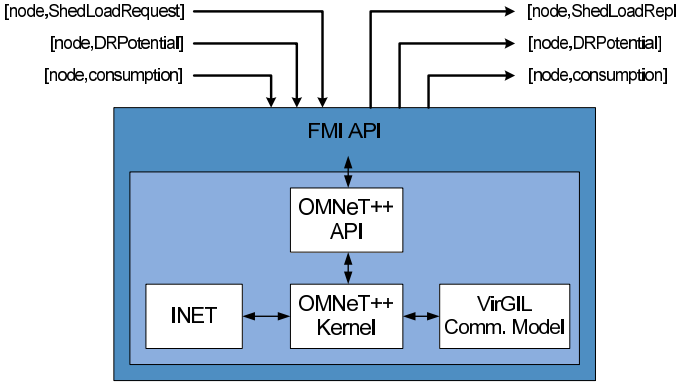


Fig. 10: Structure for the Communication's FMU.

In order to emulate the transmission of information in the network, several parameters are input and output to and from the communication's. As seen in Figure 10, both *Consumption* and *DRPotential* are inserted into the FMU together with a node identifier. The FMU assumes these are values transmitted from client nodes to the server node. The model simulates the transmission and, after a certain amount of time (due to the communication delay) it produces an output directed to the server.

While *Consumption* and *DRPotential*'s information flows in one direction only, communication for DR events goes both ways. Now it is the server node which inserts a *shedLoadRequest* message issued to a given client. Upon reception, the client decides whether to accept or not to participate in the event and replies to the server accordingly. This reply is output from the simulator after the corresponding transmission time.

It may happen that, due to errors in the transmission, some of these messages get lost. However, these lost messages are identified by the automatic repeat request (ARQ) mechanism implemented on the TCP layer of all nodes (see Figure 8). Without going into too much detail about ARQ, whenever a message is lost, a re-transmission mechanism is triggered at the transmitting party. The result is that messages are always delivered even in the presence of errors. The only effect is that erroneous messages are affected by a higher latency (due to the re-transmission).

VIII. OPTIMIZATION AND CONTROL FMU

The control FMU continuously monitors the status of the integrated system and issues control signals, e.g., asking some building to increase/decrease her demand by a certain amount/percentage, charging/discharging energy storage, to ensure the health of the system. The control signal is based on an optimization problem, e.g., optimal power flow (OPF) problem, of the power system. Generally, it is in the following form:

$$\min f(P, Q, V, \theta) \quad (4)$$

$$k(P, Q, V, \theta) = 0 \quad (5)$$

$$\underline{V} \leq V \leq \bar{V} \quad (6)$$

$$|S(\theta, V)| \leq \bar{S} \quad (7)$$

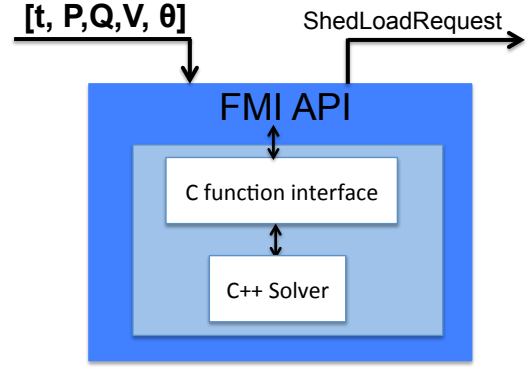


Fig. 11: Structure of the Control FMU.

$f(P, Q, V, \theta)$ is the system cost, which can be the generation costs, system losses, and other control efforts. $k(P, Q, V, \theta) = 0$ represents the Kirchhoff Laws. We also have network constraints, e.g., the voltage constraint and line capacity constraint, as well as other constraints not listed here. Then given system configuration and current status, the control FMU will issue control signals trying to move the system towards the optimal point of the optimization problem. Notably there are lots of challenges in solving the general form of the optimization problem. We employ the state-of-the-art technique [27] to obtain the solution. It is our ongoing work to explore distributed control and decision making under uncertainty.

The basic structure of the control FMU is shown in Figure 11. The control FMU takes the system status at time t , e.g., $[t, V, P, Q, \theta]$, as input, and then employs the C function interface to finally call C++ solver to get the output and issue the *ShedLoadRequest*.

We use the following controllers for our simulations:

- Line capacity controller: make sure the power flow on each line is within its capacity, otherwise issue control signal to shed building load so that all line capacity constraints become satisfied. The shed request can be in either kW or percentage of current building load.
- Volt/var controller: issue control signals to dynamically adjust the reactive power, e.g., from energy storage, to stabilize the voltage on target buses.
- Slope controller: make sure the change slope of power consumption, voltage, and/or current is within acceptable region, otherwise issue control signal to shed building load and/or control energy storage so that the change is not too aggressive, which results in high cost due to reserves in power system.

IX. TIME INTEGRATION OF DIFFERENTIAL EQUATIONS USING QUANTIZED STATE SYSTEM METHODS

As described in Sec. IV, each Buildings FMU defines ordinary differential equations of the form

$$\dot{x} = f(x, u, t) \quad (8)$$

where $\dot{x}(t)$ is a vector of N state variables whose values the solver will predict; $u(t)$ is a vector of input variables which act as boundary conditions; and f is the derivative function.

If Ptolemy coordinates more than one such FMU, then the state variables predicted by one FMU may appear as the input variables of another.

To integrate these equations, we implemented both explicit and linearly-implicit Quantized State System (QSS) methods in Ptolemy [28], [29]. QSS differs from typical integration methods, in that it discretizes the state variables rather than time. Thus Eq. 8 becomes

$$\dot{x} = f(q, \mu, t) \quad (9)$$

where $q(t)$ is the *quantized state*, i.e., a discretized version of $x(t)$. Likewise, $\mu(t)$ is a quantized version of $u(t)$.

Quantization consists of representing a variable as a series of piecewise-continuous polynomials. Component j of the ODE system has a *quantized state model*

$$\hat{q}_{j[\ell]}(t) = \sum_{i=0}^{M-1} q_{j[\ell]}^{[i]} \left(t - t_{j[\ell]}^q \right)^i \quad (10)$$

where $q_{j[\ell]}^{[i]}$ denotes the i^{th} polynomial coefficient for the ℓ^{th} model; $t_{j[\ell]}^q$ gives the *quantization-event time* at which the model was formed; and M gives the QSS method order. For example, QSS1, a first-order method, quantizes the state as a constant, $\hat{q}_{j[\ell]}(t) = q_{j[\ell]}^{[0]}$. Each model holds on $t_{j[\ell]}^q \leq t < t_{j[\ell+1]}^q$ (although $t_{j[\ell+1]}^q$ is not known at time $t_{j[\ell]}^q$).

Integrating the quantized state gives a series of *state models*

$$\hat{x}_{j[k]}(t) = \sum_{i=0}^M x_{j[k]}^{[i]} \left(t - t_{j[k]}^s \right)^i \quad (11)$$

valid on $t_{j[k]}^s \leq t \leq t_{j[k+1]}^s$. Note that the *state-event times* $t_{j[k]}^s$ may differ from the quantization-event times. Fig. 12 shows a block diagram of a QSS integrator.

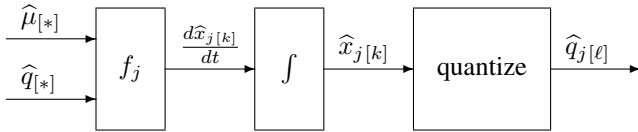


Fig. 12: QSS integration of a component of an ODE system. The quantized state is a piecewise-continuous approximation to $x(t)$. The simulation iteratively updates the state and quantized state models for individual components.

Component j forms a new state model when a quantized input to f_j changes. At the k^{th} state-event time, the new state model is made continuous with the previous one, and its slope found from the derivative function:

$$x_{j[k]}^{[0]} = \hat{x}_{j[k-1]} \left(t_{j[k]}^s \right) \quad (12)$$

$$x_{j[k]}^{[1]} = f_j \left\{ \left(\hat{q}_{[*]}, \hat{\mu}_{[*]}, t_{j[k]}^s \right) \right\} \quad (13)$$

where the models $\hat{q}_{[*]}$ and $\hat{\mu}_{[*]}$ are evaluated at $t_{j[k]}^s$. Index “*” indicates the most recent model for each component. For QSS2 and QSS3, the higher-order coefficients $x_{j[k]}^{[2]}$ and $x_{j[k]}^{[3]}$ are estimated by perturbing the arguments to the derivative function; details are beyond the scope of this paper.

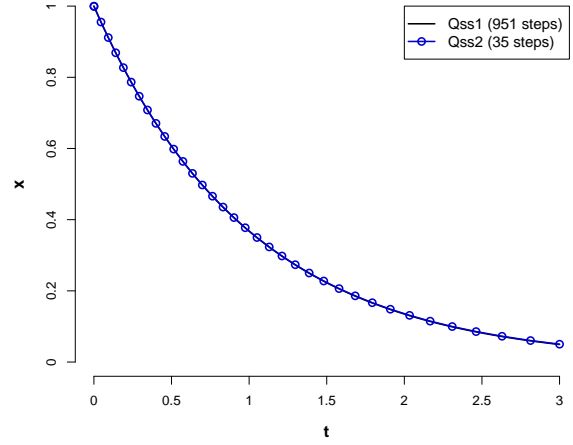


Fig. 13: QSS solution of the exponential problem, $\dot{x} = -x$.

Component j forms a new quantized state model when the current quantized state model differs from $\hat{x}_{j[k]}$ by an amount ΔQ_j , called the *quantum*. In the absence of other events, this happens when

$$\left| \hat{x}_{j[k]} \left(\hat{t}_{j[\ell+1]}^q \right) - \hat{q}_{j[\ell]} \left(\hat{t}_{j[\ell+1]}^q \right) \right| = \Delta Q_j \quad (14)$$

where $\hat{t}_{j[\ell+1]}^q$ is the *predicted quantization-event time* for component j . In practice, ΔQ_j varies with the magnitude of $q_{j[k]}^{[0]}$, according to user-defined tolerances.

At each time step, the simulation advances to the minimum predicted quantization-event time from among all the components. Thus a given global time step may re-quantize only one out of all the components.

When component j does finally experience a quantization-event, it forms a new quantized state model by matching the value and derivatives from the current state model:

$$q_{j[\ell]}^{[0]} = \hat{x}_{j[k]} \left(t_{j[\ell]}^q \right) \quad (15)$$

$$q_{j[\ell]}^{[1]} = \frac{d\hat{x}_{j[k]}}{dt} \left(t_{j[\ell]}^q \right) \quad (16)$$

and so on, for derivatives up to $M-1$ (however, the linearly-implicit QSS methods offset the initial value by up to ΔQ_j). The new quantized state model is then broadcast to any other component whose derivative function depends on x_j . This, in turn, induces state-events in those downstream components.

Fig. 13 shows the QSS1 and QSS2 solutions of the exponential problem, $\dot{x} = -x$, with initial condition $x(0) = 1$. Quantum was chosen as the minimum of 0.001 and $0.001 \cdot |q_{j[\ell]}^{[0]}|$. Compared to the analytical result $x = e^{-t}$, both solutions end at $t = 3$ with a global error less than $5 \cdot 10^{-4}$.

The QSS approach treats every differential equation as a discrete event actor, generating events, and responding to the events produced by other equations. However the Ptolemy implementation currently groups the equations by FMU. Thus if one equation experiences a state-event, it updates the state models for all equations contained in the same FMU.

In addition to the differential equations defined by Model Exchange FMUs, Ptolemy also must handle Co-Simulation FMUs. As suggested by Fig. 4, the Power Systems FMU

defines a static relation, determining the power flows as an algebraic function of its inputs. Since all feedback paths from the Power System outputs back to its inputs pass through the building models, Ptolemy does not have to solve any algebraic loops. To avoid having to call the Power FMU every time a building model updates one of its outputs, we sample the building loads at discrete intervals.

X. MASTER ALGORITHM

To synchronize the data of the different FMUs, we will use Ptolemy II [3]. Ptolemy II is a modular software environment for the design and analysis of heterogeneous systems. It provides a graphical model building environment, synchronizes the exchanged data and visualizes the system evolution during run-time. In Ptolemy II, components are encapsulated as actors which communicate with other actors through ports. A director orchestrates the data exchange between the actors and advances time for the individual actors.

Next, we will discuss the mathematical structure of each FMU, and then discuss how we composed them for a co-simulation. To compose multiple actors in order to conduct a co-simulation, we need to make the distinction between outputs of actors that directly depend on inputs, e.g., they have direct feedthrough, and outputs of actors that do not directly depend on inputs. The latter are for example outputs of explicit time integrators that only change when time is advanced, but not if an input is changed.

The power systems FMU implements an algebraic, time invariant system. Therefore, the outputs of this FMU directly depend on the input values.

The building FMUs take as an input the control signal y_{shed} and produce as outputs the active and reactive power P and Q . Both do not directly depend on y_{shed} . The building FMUs are exported using the FMI for Model-Exchange 2.0 standard. When imported to Ptolemy II, they are combined with QSS integrator, as described in Section IX. For the master algorithms, these QSS integrators can be abstracted as actors that may schedule a time event whenever their input changes, or whenever their state variables change by more than a tolerance. Should the input change prior to such a scheduled event, then the actor may replace this event with a new one that may happen at a different time.

The communications FMU lead to time delays in the signals. They take as inputs signal $u_j(t)$, for some $j \in \{1, \dots, n\}$, where n is a fixed number of channels, and produce after some time delay $\delta_j(t)$ the signal at the output. Hence, the output is $y_j(t + \delta_j(t)) = u_j(t)$. For signal j , the time delay is a function of all signals that have not yet been sent to their output, allowing to model network congestion. In our communication FMU, once $\delta_j(t)$ has been computed, it will not be changed. Therefore, network congestion does not affect signals that have already been received in the communication FMU but have not yet been produced at its output.

The optimization and control FMU has discrete time semantics. For a constant time step $\delta > 0$, and given measurement

signal $u(i\delta)$, with $i \in \{0, 1, \dots\}$, it outputs the control action $y((i+1)\delta) = f(u(i\delta))$.

Figure 14 shows the Ptolemy II system model that combines the power, building, communication and controls FMU. The QSS Director is a new addition to Ptolemy II that we developed in conjunction with the Ptolemy II development team. The QSS director extends the discrete event director, and adds a QSS solver. Thereby, this director allows combining FMUs for model exchange, which will be integrated with the QSS algorithm, with FMUs for co-simulation. In addition, other Ptolemy II actors that work in the discrete event domain can be used in such system models.

XI. SIMULATION RESULTS

A. Calibration of the building model

The case study LBNL's building 71 is a 54,000 ft² two story steel-frame office and laboratory building located in Berkeley, California. The building has a water-cooled chiller system with three cooling towers. The building's operation is typical of office and laboratory, with an operational schedule of 9:00 am to 6:00 pm and high equipment usage during off-hours. The peak electric power demand is over 400 kW during the period of 12:00 pm to 6:00 pm and the average demand during the off hours is about 80 kW.

The building FMU has been generated as described in section Sec. ???. Before using the building FMU model we performed a calibration of the model in order to test the ability of the simplified RC model to replicate the results of the more detailed EnergyPlus model. The main difference between the EnergyPlus model and the simplified RC model are the nonlinear relationships and complex algorithm used by EnergyPlus to compute interior and exterior convective coefficients, solar heat gain coefficients and long wave radiation effects. However, the RC model has a number of coefficients that can be tuned in order to align the simulation results as much as possible. Sensitivity analysis was conducted to rank the key parameters of the RC model that could be tuned. The parameters with higher sensitivity that were used to tune the model are: exterior wall convective coefficient, building solar absorption factor, window heat gain factor and heat transmission value. We therefore used the summer period from May to October to perform a comparison between the RC and the EnergyPlus models. We decided to use such a period because it is of interest for the demand response events as well for critical peak pricing.

Both the RC and the EnergyPlus model have been simulated using standard weather for San Francisco, lighting, plug loads, occupancy and set point schedules for the zones of the building. Given the same boundary condition and operation the models predicted the cooling load required to satisfy the required comfort conditions.

Figure (15) shows the RC model cooling load versus the EnergyPlus model cooling load. Every points represents a simulated data point with a 5 min resolution over the period between May and October. The green, yellow and red area respectively represent a relative error of $\pm 5\%$, $\pm 10\%$ and $\pm 15\%$. As can be seen the highest relative error occurs at low

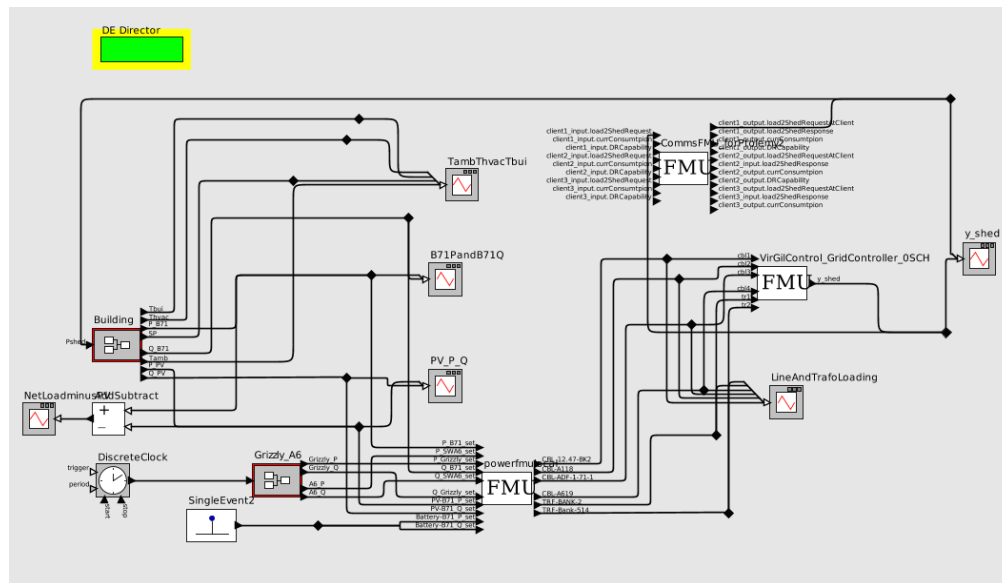


Fig. 14: Ptolemy II model that shows the composition of the different FMUs.

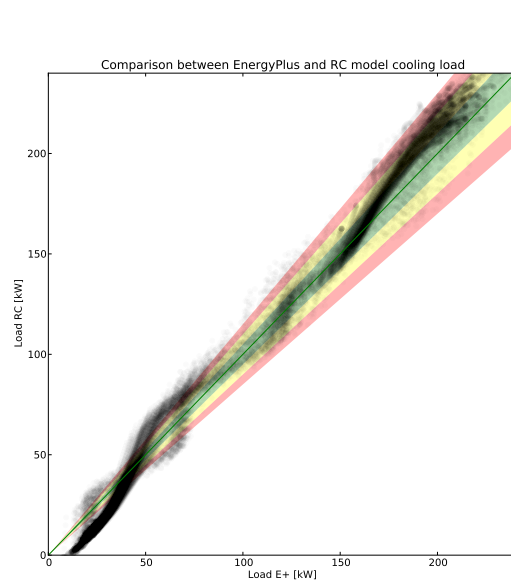


Fig. 15: Comparison of the cooling load predicted by EnergyPlus and the RC model.

cooling load level while when the load is close to its maximum the almost totality of teh points is within the $\pm 15\%$ interval.

Figure (15) shows a comparison across the hour of the day between the cooling load predicted by EnergyPlus and the RC model. All the simulation points for every hour of the day over the simulation period have been collected and their distribution have been compared. The blue boxes shows the cooling load distribution for the EnergyPlus model while the green boxes shows the distribution of the cooling load predicted using the RC model. The highest relative errors evidenced in Figure (15) happen exclusively at night time when the nonlinearities of the long wave radiation exchange that are not captured by the RC

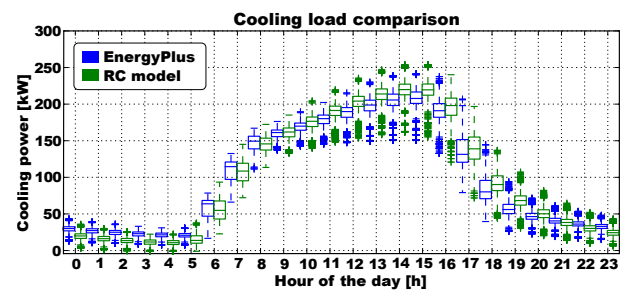


Fig. 16: Comparison of the cooling load predicted by EnergyPlus and the RC model.

model dominate the heat transfer of the building. However Figure (15) help us to put in perspective this error, confirming that even if it's relative high it's impact on the ability of the RC mdoel to predict the cooling load can be neglected.

Under these assumptions it's possible to consider the RC model detailed enough to describe the thermal dynamics of the building to be able to predict with a good accuracy its cooling load. The original RC model generated using the BRCM toolbox had 106 state variables. The model has been reduced to a model that is able to predict the same averaged building temperature as described in section Sec.???. The obtained reduced order model had 8 state variables and its difference between the full RC model is small enough (less than 1% in relative terms) to be neglected for the sake of this study.

B. Overview of the Use Cases

Both cases use the LBNL distribution network and Building 71. As shown in Fig. 17, the LBNL distribution network represents the path from the point of common coupling with PG&E, down to Building 71. The remainder of the distribution system loads are modeled as aggregated loads connected to two switching substations along this path. Real 15-minute data

were used for the two aggregate loads. For Bus SW-A1, real reactive power was used, while for SW-A6, a power factor of 0.94 inductive was assumed. The active power consumption of the aggregate loads in SW-A1 and SW-A6 are shown in Fig. 18.

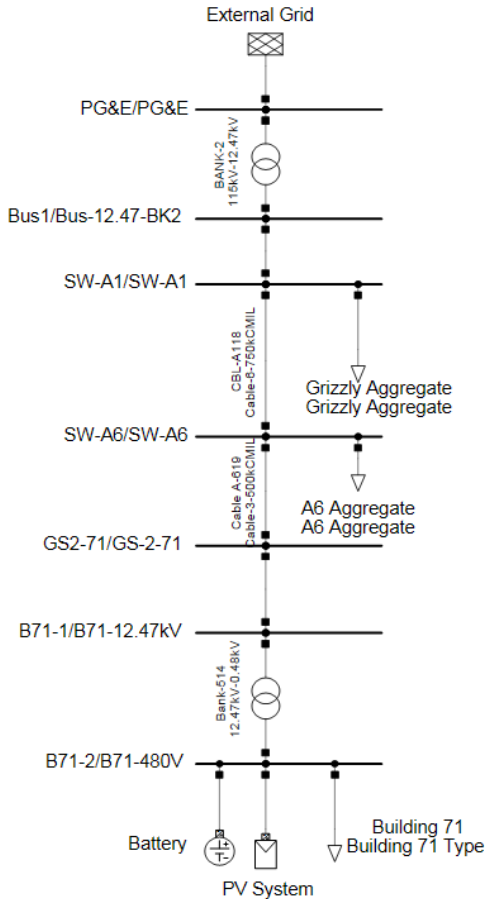


Fig. 17: PowerFactory model of the LBNL distribution system to Building 71. The rest of the LBNL, except for the Building 71, are modelled as aggregate loads on buses SW-A1 and SW-A6.

The modeling of Building 71 has been detailed in Section XI-A. To study the interaction of loads during high penetration of DER, we have assumed a solar PV plant of 340 kWp and a battery connected at the same bus. The active power consumption of Building 71 and the net load (Building 71 and solar PV) demanded at Bus B71 is shown in Fig. 19. As we did not have available data for the reactive power consumption of Building 71, we assumed a constant power factor of 0.96 inductive.

We present two use cases in the following sections to demonstrate the capabilities of VirGIL. The first applies demand response actions in Building 71 to reduce the cable and transformer loading. The second applies Volt-Var control so that the voltage at Bus B71 follow specified setpoints.

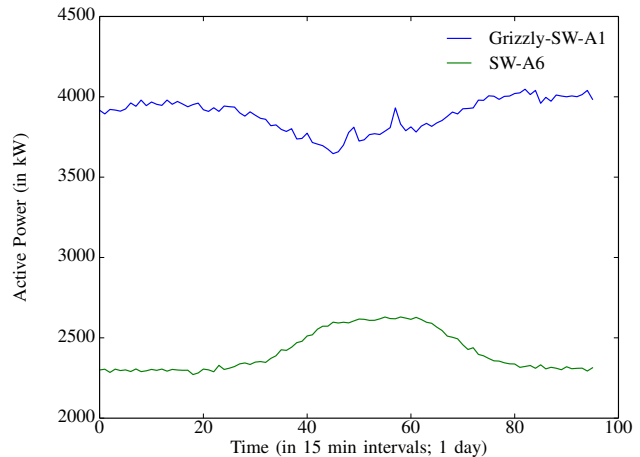


Fig. 18: Active Power Consumption of the Aggregate Loads.

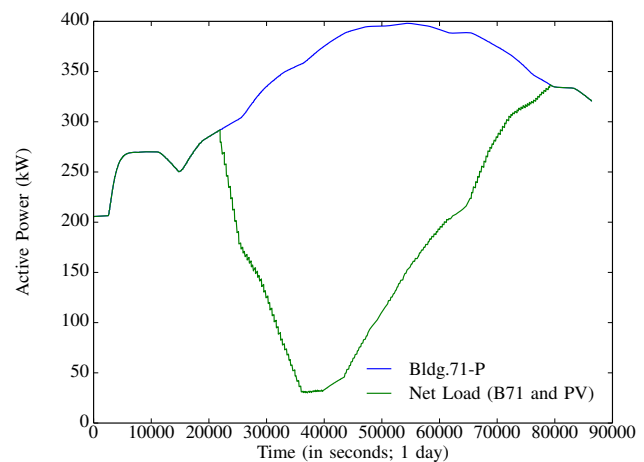


Fig. 19: Active Power Consumption of Building 71 without Demand Response (blue) and Net Load at Bus B71 (green; sum of load and PV infeed).

C. Demand Response in Building 71

Figure 20 presents the transformer and cable loadings of the LBNL network during a typical day, when no demand response actions are taken. As a secure and reliable power supply is central for the operation of a National Laboratory and the experiments that are taking place in it, we observe that the maximum loading of all cables and transformers does not exceed 60%.

In order to demonstrate the capabilities of VirGIL, we set the threshold for demand response actions at 55%. If any cable or transformer loading exceeds the 55% threshold, an automated demand response signal is sent to the building, reducing its power consumption by 20%. The Communications FMU ensures that the communication between the controller and the Building follows the OpenADR standard. Besides plug loads and lighting, which are considered constant during the day in this case, the main building load is HVAC cooling. As soon as the Building receives the DR signal, it has lookup

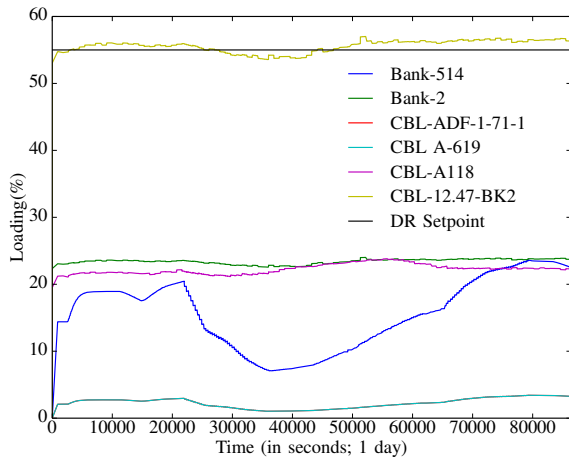


Fig. 20: Cable and Transformer Loading (no Demand Response).

tables that transform the power reduction to increased setpoints for the HVAC operation, as higher operating temperatures reduce the necessary cooling power.

Figure 21 zooms in the most critical cable loading, when no DR action is taken. We expect DR signals to be issued from midnight till 9am in the morning, and then again after 1pm in the afternoon.

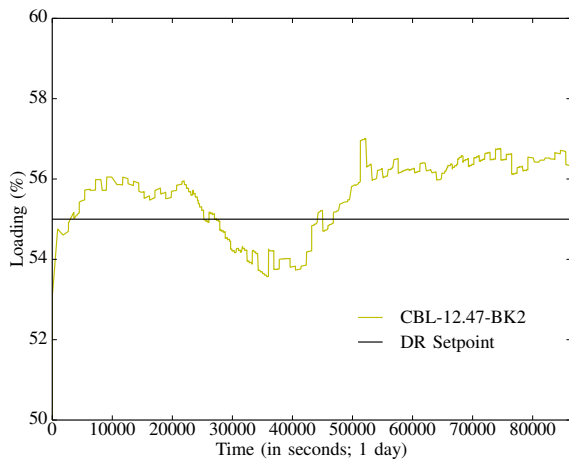


Fig. 21: Loading of the most critical cable and setpoint for Demand Response actions.

Figure 22 shows the active power consumption with DR and how this differs from the baseline for that day. To better understand this figure, we should first examine the sequence of events inside the building. Figure 23 presents the temperature variations in Building 71 for this day. “Tamb” stands for the ambient temperature. “Tbui-setpoint” is the setpoint for the average zone temperature inside the building. In case a DR event takes place, this setpoint changes. In our case, as we have cooling load, the setpoint increases. “Tbui” represents the actual average zone temperature of the building. Ideally, “Tbui” should follow “Tbui-setpoint”. “Thvac” represents the

temperature that is given as input to the HVAC system. This results from the actions of the PID controller which monitors the building temperature and automatically adjusts the HVAC setpoints. As expected, we see that the “Tbui-setpoint” increases during the first 9 hours of the day, then decreases to its nominal temperature of 293 Kelvin (20°C) and increases back again after 1pm. We observe some oscillations in the DR signal at $t=45000$ s. This is because the cable loading is marginally above or below 55%. As the day starts and the ambient temperature increases, we see that “Tbui” increases as well. at about 9am ($t=30'000$ s) it reaches the “Tbui-setpoint”. From that point on the average zone temperature follows the “Tbui-setpoint”. Going back to Fig. 22, we observe that if DR is activated, the building consumes no additional power till about 9am, when the DR signal ends. If the “Tbui-setpoint” remained at 293 K, then Building 71 would have increased its consumption already from about $t=2600$ s. Similarly, we see that after about 1 pm ($t=50'000$ s), when DR is activated again, the building consumption is reduced in comparison to the baseline.

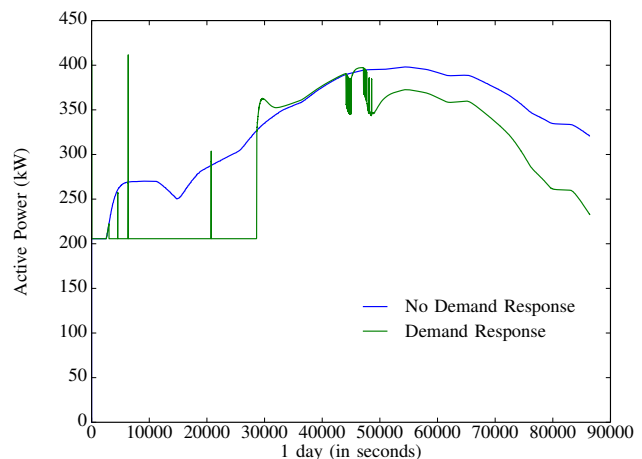


Fig. 22: Active power consumption of Building 71 with and without Demand Response Actions.

Figure 24 compares the active power consumption of Building 71 when there are no communication delays. In our base case, shown in Fig. 22, the polling frequency of the DR client in the Communications FMU was set at 30 s. This means that every DR signal was sent with 30 s. The actual communication delays that were also modelled in this setup did not exceed on average 500 ms. In Fig. 24, we do not observe major differences between the two cases. This is probably due to the fact that the building dynamics are slow enough to not be significantly affected by a 30 s delay. Still, We observe that at about $t=45'000$ s, the oscillations of the active power have a higher magnitude when the signal is transmitted with no delay. This is expected due to the more direct response to the signal.

Concluding this use case, we see how VirGIL is able to accurately model and simulate the interactions between buildings, communication, and power systems. We have observed how a power system event leads to a DR signal, and how this affects the building operation. At the same time, we were

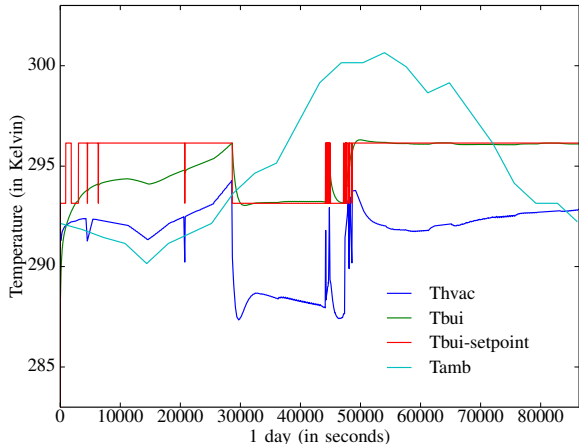


Fig. 23: Temperature variations in Building 71 with active Demand Response. Tamb: ambient temperature; Tbui-setpoint: setpoint of the temperature setpoint inside the building (varies based on demand response actions); Tbui: average temperature inside the building; Thvac: setpoint at the HVAC controller.

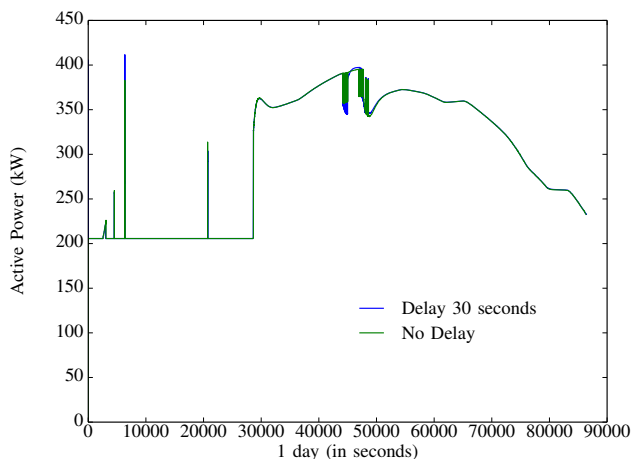


Fig. 24: Active power consumption of Building 71 with demand response, with and without signal communication delays.

able to represent the effect of the communication delays, and measure the effect of the building actions back to the power grid.

D. Volt-Var Control at Bus B71

In this use case, we demonstrate how VirGIL can be used for Volt-Var control. In the LBNL network we have installed three micro-phasor measurement units (μ PMUs) [30]. One of them is located at Bus SW-A6 and one more at Bus B71. μ PMUs are units that can measure with high fidelity voltage, current, and voltage angle.

In this case we assume that we receive as inputs the voltage from the μ PMU measurements at Buses SW-A6 and B71. The

goal is that the voltage at Bus B71 should follow the voltage at SW-A6, by appropriately controlling the reactive power infeed of the battery.

The controller solves the following equation in order to find the necessary reactive power infeed:

$$U_1^2 = (U_2 + (R \cdot P + X \cdot Q) / U_2)^2 + (X \cdot P - R \cdot Q)^2 / (U_2)^2 \quad (17)$$

where:

$$U_1 = U_{SW-A6} \quad (18)$$

$$U_2 = U_{B71} \quad (19)$$

$$P = P_{B71} - P_{BAT} - P_{PV} \quad (20)$$

$$Q = P_{B71} - Q_{BAT} - Q_{PV} \quad (21)$$

$$R = R_{Bank-514} + R_{CBL-ADF-1-71-1} + R_{A-619} \quad (22)$$

$$X = X_{Bank-514} + X_{CBL-ADF-1-71-1} + X_{A-619} \quad (23)$$

Figure 25 presents the voltage at the two buses if no volt-var control actions take place. We can observe how the PV infeed, starting at about 7am, increases the voltage momentarily, while in general the voltage level at Bus B71 is decreasing as the building consumption increases. Once again, we observe that the LBNL network is sufficiently (over)dimensioned so that we do not observe significant voltage drops at the end of the feeders. Still this use case demonstrates VirGIL performance and characteristics.

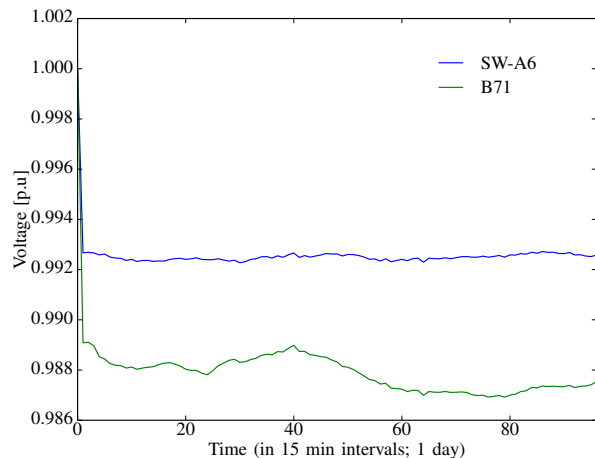


Fig. 25: Without Volt-Var Control: Voltage level of Bus SW-A6 and Bus B71.

Figure 26 presents the same voltages, but with volt-var control, so that V_{B71} to track the voltage V_{SW-A6} at Bus SW-A6. The required reactive power infeed from the battery is presented in Fig. 27. In the same figure, we also present the ΔQ , i.e. the difference by which the Q setpoint should be adjusted from one timestep to another.

XII. CONCLUSIONS

The Virtual Grid Integration Laboratory (VirGIL) creates a modular co-simulation platform for studying in detail the impact of demand response and other controls on power

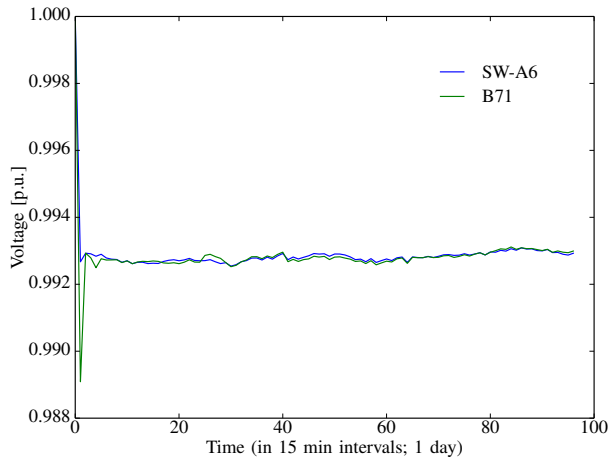


Fig. 26: With Volt-Var Control: Voltage level of Bus SW-A6 and Bus B71.

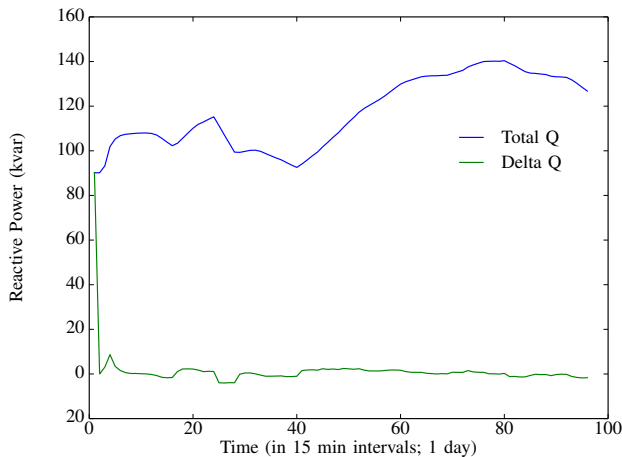


Fig. 27: Reactive power injection of the battery. Total Q corresponds to the reactive power injected. Delta Q is the change in reactive power from the previous timestep.

systems. The platform coordinates commercial software such as PowerFactory, open-source packages such as the Modelica Buildings Library, communication simulation tools such as OMNET++ and bespoke models (such as the Buildings FMU described above). Using commercial and trusted power system software is expected to lower the barriers for adoption of simulation and optimization tools by power system operators, allowing them to test, improve, and deploy new practices, e.g., efficiently integrating demand response in their daily operation. VirGIL uses the industry-standard Functional Mockup Interface (FMI), which encourages a modular approach to instantiating and sharing models.

This paper presented the development of FMUs for Co-Simulation for Power Systems, Communications, and Control, and one FMU for Model Exchange for Building modeling and control. Real network and consumption data were used

as parameters and inputs to these FMUs to model part of the LBNL distribution grid, and to couple the grid to a reduced-order physics-based model of a real building that implements a simple demand response protocol. A full representation of the communications network was also included. To our knowledge, this is the first time that a co-simulation platform couples commercial power system software such as Powerfactory with building and communications models to study the impact of demand response actions on the distributions grid. A further contribution of this paper is the full integration of the Quantized State System (QSS) methods for simulation in VirGIL.

Ptolemy II, the VirGIL implementation framework, handles both continuous and discrete-event simulation, and supports both FMI for model exchange and co-simulation.

Future extensions of this work include electric vehicles, power system optimization, and advanced building controls. Use cases will include demand response applications for volt/var optimization, 3-phase asymmetries, and distribution system planning. Real case studies, such as the demand response potential and impact in the region of the decommissioned San Onofre Nuclear Generating Station (SONGS), will be simulated in VirGIL and presented.

ACKNOWLEDGEMENTS

We thank Edward Lee and Christopher Brooks from UC Berkeley for their support in the FMI and QSS implementation in Ptolemy II. This work was supported by Laboratory Directed Research and Development (LDRD) funding from Berkeley Lab, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

- [1] M. Wetter and P. Haves, "Modular building controls virtual test bed for the integration of heterogeneous systems," in *Proceedings of 3rd National Conference of IBPSA-USA SimBuild 2008, Berkeley, CA, July 2008*, pp. 1–8.
- [2] Modelica Association Project "FMI", "Functional mock-up interface for model exchange and co-simulation," [Online]: <https://www.fmi-standard.org>, July 2014.
- [3] C. Ptolemaeus, Ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014. [Online]. Available: <http://ptolemy.org/books/Systems>
- [4] K. Hopkinson, X. Wang, R. Giovanini, J. Thorp, K. Birman, and D. Coury, "EPOCHS: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components," *Power Systems, IEEE Transactions on*, vol. 21, no. 2, pp. 548–558, May 2006.
- [5] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili, "Power system and communication network co-simulation for smart grid applications," in *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, Jan 2011, pp. 1–6.
- [6] M. Lévesque, D. Q. Xu, G. Joós, and M. Maier, "Communications and power distribution network co-simulation for multidisciplinary smart grid experimentations," in *Proceedings of the 45th Annual Simulation Symposium*, ser. ANSS '12. San Diego, CA, USA: Society for Computer Simulation International, 2012, pp. 2:1–2:7. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2331751.2331753>
- [7] R. Bottura, D. Babazadeh, K. Zhu, A. Borghetti, L. Nordstrom, and C. Nucci, "SITL and HLA co-simulation platforms: Tools for analysis of the integrated ICT and electric power system," in *EUROCON, 2013 IEEE*, July 2013, pp. 918–925.

- [8] M. Stifter, E. Widl, F. Andren, A. Elsheikh, T. Strasser, and P. Palensky, "Co-simulation of components, controls and power systems based on open source software," in *Power and Energy Society General Meeting (PES), 2013 IEEE*, July 2013, pp. 1–5.
- [9] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: Challenges, tools and methods," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 44, no. 3, pp. 318–326, March 2014.
- [10] D. Chassin, K. Schneider, and C. Gerkenmeyer, "GridLAB-D: An open-source power systems modeling and simulation environment," in *IEEE PES Transmission and Distribution Conference and Exposition*, April 2008, pp. 1–5.
- [11] S. Muller, H. Georg, C. Rehtanz, and C. Wietfeld, "Hybrid simulation of power systems and ICT for real-time applications," in *Innovative Smart Grid Technologies (ISGT Europe), 2012 3rd IEEE PES International Conference and Exhibition on*, Oct 2012, pp. 1–7.
- [12] D. Sturzenegger, D. Gyalistras, V. Semeraro, M. Morari, and R. Smith, "BRCM Matlab toolbox: Model generation for model predictive building control," in *American Control Conference 2014*, Apr 2014, pp. 1–5.
- [13] R. Yin, P. Xu, M. A. Piette, and S. Kiliccote, "Study on auto-DR and pre-cooling of commercial buildings with thermal mass in california," *Energy and Buildings*, vol. 42, no. 7, pp. 967–975, Jul 2010.
- [14] Commercial prototype building models. <http://www.energycodes.gov/commercial-prototype-building-models>.
- [15] K. Glover, "All optimal Hankel-norm approximations of linear multi-variable systems and their L-inf error bounds," *International Journal of Control*, vol. 39, no. 6, pp. 1115–1193, 1984.
- [16] K. Zhou, "Frequency-weighted model reduction with L error bounds," *Systems & control letters*, vol. 21, no. 2, pp. 115–125, 1993.
- [17] M. Wetter, W. Zuo, T. S. Nouidui, and X. Pang, "Modelica buildings library," *Journal of Building Performance Simulation*, vol. 7, no. 4, pp. 253–270, 2014.
- [18] Omnet++. <https://www.omnetpp.org/>.
- [19] T. Godfrey, S. Mullen, R. C. Dugan, C. Rodine, D. W. Griffith, and N. Golmie, "Modeling Smart Grid Applications with Co-Simulation," in *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010, pp. 291–296.
- [20] C. Müller, H. Georg, and C. Wietfeld, "A Modularized and Distributed Simulation Environment for Scalability Analysis of Smart Grid ICT Infrastructures," *Proceedings of the Fifth International Conference on Simulation Tools and Techniques*, 2012.
- [21] K. Mets, J. A. Ojea, and C. Devellder, "Combining Power and Communication Network Simulation for Cost-Effective Smart Grid Analysis," *IEEE Communications Surveys & Tutorials*, pp. 1–26, 2014.
- [22] M. Bredel and M. Bergner, "On the accuracy of ieee 802.11 g wireless lan simulations using omnet++," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.
- [23] A. Virdis, G. Stea, and G. Nardini, "SimuLTE – A Modular System-level Simulator for LTE // LTE-A Networks based on OMNeT ++," in *Proceedings of SimulTech*, 2014, pp. 28–30.
- [24] E. Weing, H. Lehn, and K. Wehrle, "A performance comparison of recent network simulators," *IEEE International Conference on communications*, 2009.
- [25] Openadr alliance. <http://www.openadr.org/>.
- [26] Openadr web page in the demand response research center. <http://drrc.lbl.gov/openadr>.
- [27] L. Gan and S. Low, "A distributed algorithm for solving the optimal load control problem in multiphase radial networks," [Online]: <http://www.its.caltech.edu/~lgan/codes/OptimalPowerFlowSolver.zip>, 2014.
- [28] F. E. Cellier and E. Kofman, *Continuous System Simulation*. Springer, 2006.
- [29] G. Migoni, M. Bortolotto, E. Kofman, and F. E. Cellier, "Linearly implicit quantization-based integration methods for stiff ordinary differential equations," *Simulation Modelling Practice and Theory*, vol. 35, pp. 118–136, 2013.
- [30] E. Stewart, S. Kiliccote, C. Shand, A. McMorran, R. Arghandeh, and A. Von Meier, "Addressing the challenges for integrating micro-synchrophasor data with operational system applications," in *PES General Meeting — Conference Exposition, 2014 IEEE*, July 2014, pp. 1–5.