INVITED PAPER

Cyber–Physical Modeling of Distributed Resources for Distribution System Operations

The Virtual Grid Integration Laboratory (VirGIL) is a modular cosimulation platform designed to study interactions between demand-response strategies, building comfort, communication networks, and power system operation. VirGIL introduces the use of the quantized state system (QSS) methods for simulation in this cosimulation platform.

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

ABSTRACT | Cosimulation platforms are necessary to study the interactions of complex systems integrated in future smart grids. The Virtual Grid Integration Laboratory (VirGIL) is a modular cosimulation platform designed to study interactions between demand-response (DR) 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 mockup interface (FMI), where several different modules can be added, exchanged, and tested; and 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 DR strategies in their daily operation. VirGIL further introduces the integration of the quantized state system (QSS) methods for simulation in this cosimulation platform. Results on how these systems interact using a real network and consumption data are also presented.

KEYWORDS | Cosimulation; demand response (DR); DigSILENT PowerFactory; functional mockup interface (FMI); load flow; modelica; OMNeT++

I. INTRODUCTION

Moving toward "smarter" grids, power system 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 cosimulation approach, where highly developed and reliable simulation tools, specialized in the respective

Manuscript received February 15, 2015; revised October 9, 2015; accepted October 26, 2015. Date of publication March 7, 2016; date of current version March 17, 2016. This work was supported by the Laboratory Directed Research and Development (LDRD) funding from Berkeley Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy DE-AC02-05CH11231.

S. Chatzivasileiadis is with the Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA (e-mail: chatziva@mit.edu).
M. Bonvini is with Whiskerlabs, Oakland, CA 94612 USA (e-mail: marco@whiskerlabs.com).
J. Matanza is with Comillas Pontifical University, Madrid 28015, Spain (e-mail: jmatanza@comillas.edu).
R. Yin, **T. S. Nouidui**, **E. C. Kara**, **D. Lorenzetti**, and **M. Wetter** are with the Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (e-mail: ryin@lbl.gov; tsnouidui@lbl.gov; eckara@lbl.gov; dmlorenzetti@lbl.gov; mwetter@lbl.gov).
R. Parmar is with the University of Calgary, Calgary, AB T1Y 4Z9, Canada and also with the Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (e-mail: rajiv1parmar@gmail.com).
S. Kiliccote is with the SLAC National Accelerator Laboratory, Menlo Park, CA 94025 USA (e-mail: SilaK@SLAC.Stanford.edu).

Digital Object Identifier: 10.1109/JPROC.2016.2520738

0018-9219 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

fields, are merged in a common cosimulation platform to study the interdependencies between systems and identify appropriate control strategies.

Although cosimulation 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 eight 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 (DR) 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 DR 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 toward 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 cosimulation 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 mockup interface (FMI) 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 DR strategies on the grid, and to determine optimal algorithms for exploiting flexible loads (for example, the thermal energy stored in buildings).

VirGIL's architecture is based on the FMI, which defines a standard interface for exposing the capabilities of a simulation tool [3]. FMI, which is an open industrial standard, provides for a modular structure that allows the simple exchange and testing of different cosimulation tools. VirGIL is implemented using the CyPhySim distribution of the Ptolemy II framework, which combines mechanisms for continuous time, discrete time, and discrete-event simulation, and quantized state system (QSS) integrators and FMIs [4], [5].

VirGIL is implemented in the Ptolemy II framework, which combines continuous and discrete-event simulation [4].

This paper is organized as follows. Section II reviews existing cosimulation methods in power systems. Section III provides the overview of the proposed cosimulation architecture in VirGIL, while Section IV describes the FMI. Sections V–VIII present, respectively, the development of the power systems, buildings, communications, and control functional mockup 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. COSIMULATION 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 [6], which cosimulates power and communication 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 communication 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 inacurracies over time. The authors improved this approach in [7] 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 cosimulation approaches for power systems combine power system with communication network simulation (examples for distribution networks are [8] and [9]). Stifter *et al.* [10] report a cosimulation 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 cosimulation in power systems is reported, among others, in [11].

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 cosimulation, Gridlab-D is probably one of the most widespread [12]. 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 system simulation. Building a cosimulation platform incorporating PowerFactory, a tool that several utilities trust and use in

their daily operation, decreases the barriers for wider adoption of cosimulation tools from the industry. Power system operators can incorporate their version of PowerFactory with the cosimulation platform to investigate in more detail the effect of DR signals, decide and subsequently deploy the most appropriate in real-time operation. PowerFactory has the additional advantage of being capable to model both alternating current (ac) and direct current (dc) systems. A cosimulation approaches incorporating PowerFactory has also been documented in [13]. However, this is the first time that a modular cosimulation 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 cosimulation. FMI provides for a modular structure of the cosimulation platform which allows the simple exchange and testing of different cosimulation 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 which allow for higher efficiency and faster execution times.

III. VirGIL

VirGIL is a modular cosimulation 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 cosimulation architecture.

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

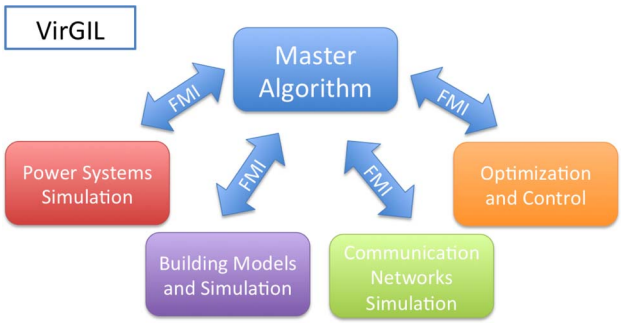


Fig. 1. ***Overview of the VirGIL cosimulation architecture. Modules include tools for simulating power systems, buildings, and communications.***

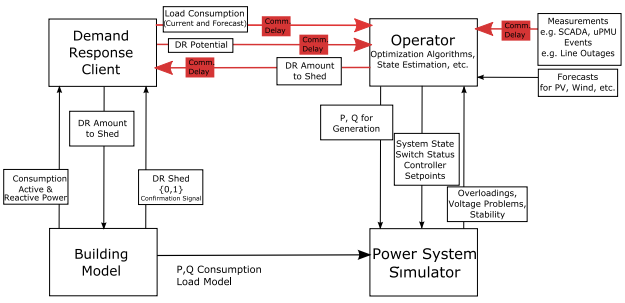


Fig. 2. ***Interactions between VirGIL modules.***

IV. FUNCTIONAL MOCKUP INTERFACE (FMI)

VirGIL's master algorithm coordinates all modules through the FMI [3]. FMI defines a tool-independent standard for exchanging models and running 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 an 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 cosimulation. 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

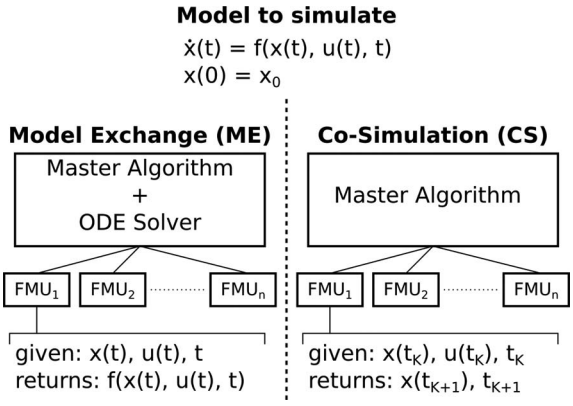


Fig. 3. ***FMI for model exchange and FMI for cosimulation.***

necessary solvers. By contrast, the FMI for cosimulation defines an interface for coupling independent simulation tools. Under cosimulation, 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 cosimulation, while the buildings model uses the FMI for model exchange.

V. POWER SYSTEM FMU

After reviewing several power system software packages, we chose DigSILENT PowerFactory as the power system simulator. The main focus was on established commercial power system software, in order to demonstrate how cosimulation 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 system 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 shows the structure of the FMU for power system 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.

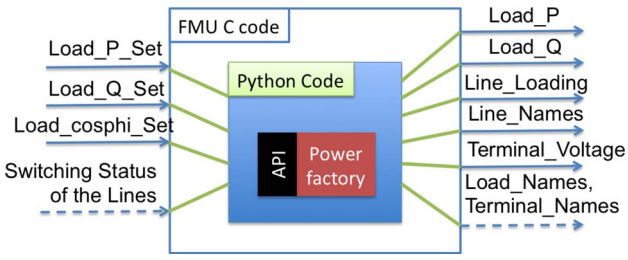


Fig. 4. ***An FMU wrapper for PowerFactory. The arrows represent variable names.***

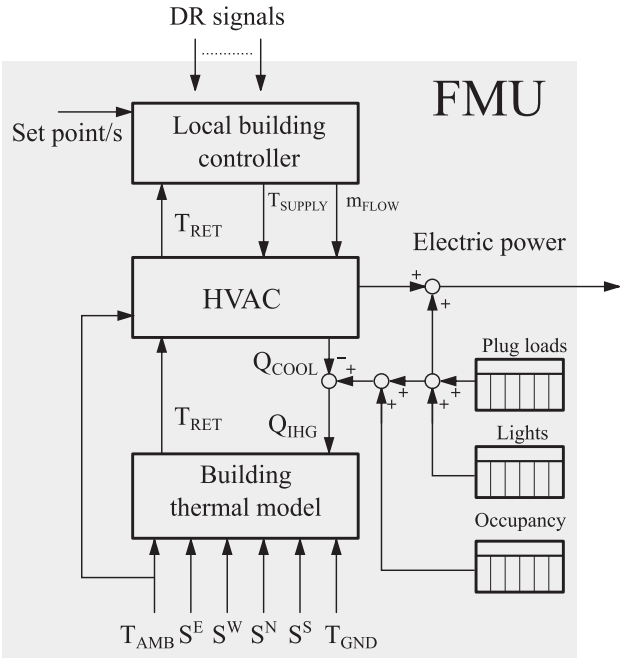


Fig. 5. ***Overview of the virgil building FMU model. The model comprises four main parts: the building thermal model, the HVAC system, the schedules, and the control system.***

VirGIL's initial focus is on the impact of DR algorithms in the power system steady-state operation, e.g., to investigate line loadings and voltage profiles. Thus, the power system 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 DR 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 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; and 4) the building control systems that

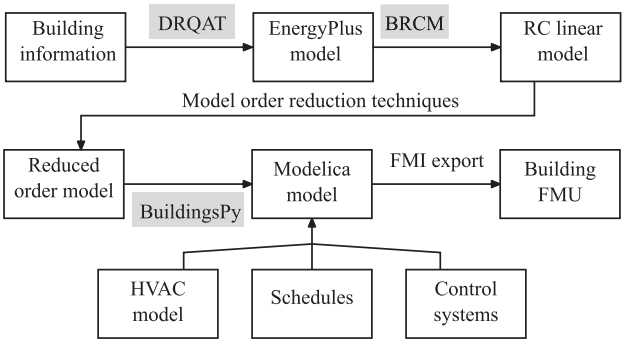


Fig. 6. *Description of the end-to-end process for generating the building FMU (gray boxes are toolboxes or packages used in the process).*

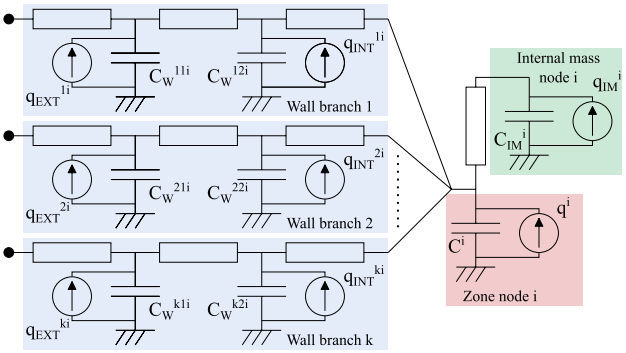


Fig. 7. *RC network for a generic zone i. Capacitances represent states (i.e., temperatures), resistances represent thermal resistances, and current sources represent heat fluxes.*

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 cosimulation [14], a complete EnergyPlus model is too detailed to simulate all the buildings in a complete distribution system. The building resistance–capacitance modeling (BRCM) toolbox [15] 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 sections 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 toward 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 DR quick assessment tool (DRQAT) [16]. Alternately, one can use prototypical models [17]. The generated EnergyPlus model contains the entire description of building geometry and other physical properties such as the conductivity of the wall layers, their thermal capacitances, the solar heat gain coefficients of the windows, etc. This information will then be used to generate a simplified first-principle model of the building.

B. Converting the EnergyPlus Model to RC Model

The BRCM toolbox allows to convert 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 an 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^i_{IM}, and a series of thermal capacitances and resistances for each of the N layers of the k walls surrounding the zone C^{kni}_W. The heat fluxes q^i, q^i_{IM}, q^{ki}_{INT}, and q^{ki}_{EXT}, 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 to the outermost layer of the walls.

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

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

$$y(t) = Cx(t) + D_u u(t) + D_v v(t) \tag{1b}$$

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 is 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 setpoint. The HVAC system model computes the cooling power to be delivered to maintain the zone temperatures at the desired setpoint.

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}, which is the weighted average of the thermal zones temperature. After introducing such simplifications, (1) can be rewritten as

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

$$y(t) = Cx(t) \tag{2b}$$

$$C = \begin{pmatrix} \frac{V_1}{V_{\mathrm{TOT}}} & \cdots & \frac{V_{nx}}{V_{\mathrm{TOT}}} & 0 & \cdots & 0 \end{pmatrix} \tag{2c}$$

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 ith thermal, and $V_{\mathrm{TOT}} = \sum_{i=1}^{nz} V_i$ is the sum of all the volumes. The vector of known disturbances $v(\cdot)$ and outputs $y(\cdot)$ are thus defined as

$$v(\cdot) = (Q_{\mathrm{IHG}} \quad T_{\mathrm{AMB}} \quad T_{\mathrm{GND}} \quad S^E \quad S^W \quad S^N \quad S^S)^T$$
$$y(\cdot) = (T_{\mathrm{RET}}).$$

Here, Q_{IHG} is the heat flux from internal heat gains, T_{AMB} is the ambient temperature, T_{GND} is the ground temperature, and S^E, S^W, S^N, S^S is the solar radiation from east, west, north, and south, respectively.

Despite the fact that the number of input–output relationships of 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 a hundred states). For such a reason the model can be further reduced [18], [19]. As we describe in Section XI-A, the resulting model will have a close match of the input–output behavior 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 relationships between the known disturbances and the output is defined, it is possible to express it using Modelica, an object-oriented, equation-based language for modeling multidomain physical systems.

Then, drawing on the Modelica Buildings Library [20], we add the HVAC, loads, and controls logic. These components predict the active power consumption of the building, and implement a DR 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. COMMUNICATION FMU

With respect to the communication modeling, OMNeT++ [21] 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 R. From them, OMNeT++ and ns-2/ns3 have extensively been used in cosimulation application for smart grid scenarios [22]–[24]. Several reasons led us to choose OMNeT++. From a 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 [25] and [26], respectively. In terms of performance, both ns-2/ns-3 and OMNeT++ have a similar performance and offer good scalability features, as discussed in [27]. Further benefits of OMNeT++ which help substantially in the implementation, are its extensive and detailed documentation, and its integrated development environment adapted to Eclipse which simplifies debugging.

Regarding the DR application under study, the model counts with three high-level types of actors: server

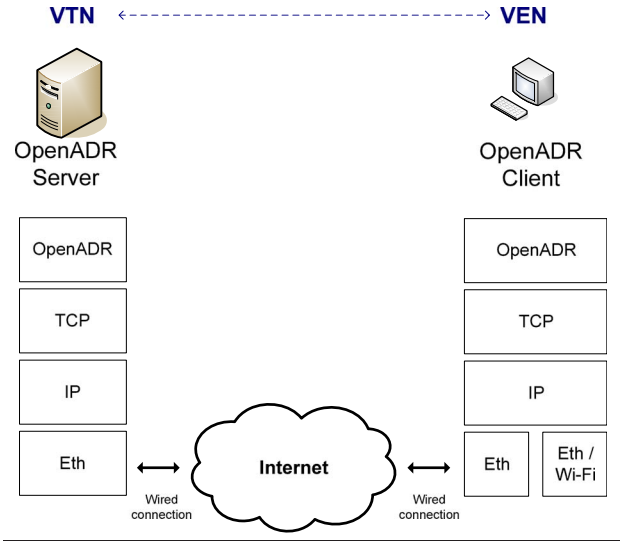


Fig. 8. ***Communication's layer stack diagram.***

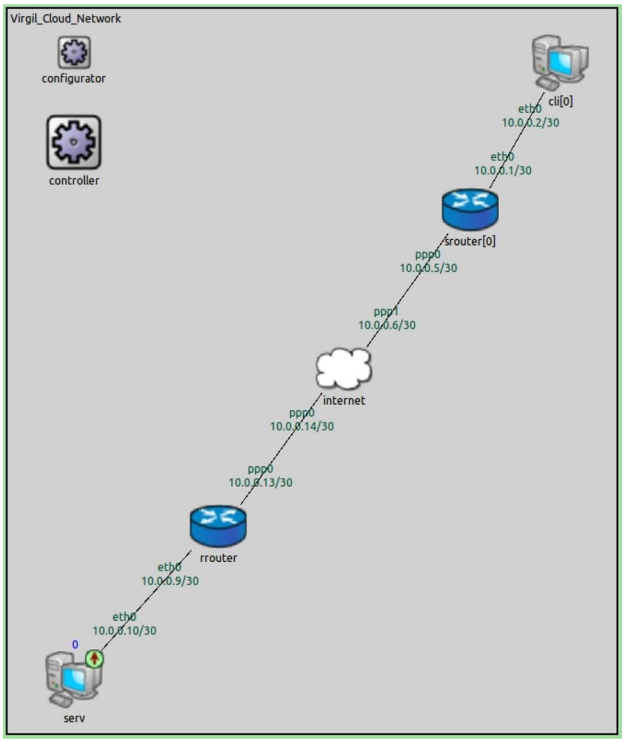


Fig. 9. ***OMNeT++ implementation of the network.***

nodes, where information about DR events is stored; client nodes, which try to retrieve this information; and a network, which 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 communication data model for sending and receiving DR signals from a utility or independent system operator to electric customers [28], [29].

Fig. 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, Fig. 8 shows the names that OpenADR' 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 Fig. 9. It counts with a DR server (labeled with "`serv`"), a DR client (labeled with "`cli[0]`"), a number or 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 Internet service provider (ISP) 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 [24] 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).

The structure of the FMU for the communication model is shown in Fig. 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. 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.

At the current implementation of VirGIL, three types of messages are exchanged between the server and the client: the "load consumption," the "DR potential" of the load, and the "shed load request." "Load consumption"

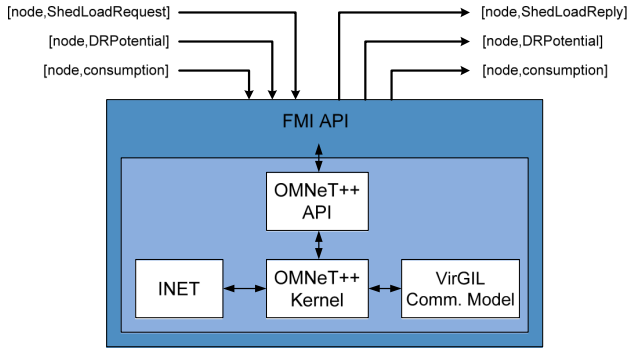


Fig. 10. ***Structure for the communication FMU.***

and "DR potential," along with their node identifier, are transmitted from the client nodes to the server node. The communication FMU simulates the message transmission and, after the calculated communication delay, it produces the output to the server. While "consumption" and "DR potential" flow in one direction only, communication for DR events flows in both ways. The server node sends a "shed load request" message to specified clients. Upon reception, the client replies to the server whether he will participate in the DR event. The communication FMU is responsible for transmitting these messages with the appropriate latency.

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 Fig. 8). Without going into too much detail about ARQ, whenever a message is lost, a retransmission 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 retransmission).

VIII. OPTIMIZATION AND CONTROL FMU

The control and optimization FMU is responsible for monitoring the status of the interconnected systems, and issuing control signals if certain limits are violated. In most of the cases, the control FMU monitors the power system FMU and sends signals to change the electricity demand in buildings, operate energy storage, etc. VirGIL can integrate many different control algorithms, ranging from simple rule-based control, to more sophisticated control laws, and optimization. The control blocks can either be directly implemented in Ptolemy or integrated through an FMI interface. For example, advanced optimization functions, which the owner would prefer to avoid sharing the source code, can be wrapped in an FMU and imported in VirGIL. In future versions, advanced optimal power flow algorithms including convex relaxations will be integrated.

The control and optimization FMU, depending on the functions it carries out, may include a model of the power system or of other components within VirGIL (e.g., buildings). Still, the existence of a separate FMU for power system simulation (and buildings, etc.) remains necessary. For example, the power system FMU contains a detailed power system model and is based on commercial software that the operator uses and trusts. Depending on the implementation, the power system FMU could also include direct SCADA signals for switching operations, etc. As a result, it can determine with much higher confidence the current and future status of the power system. On the other hand, although the control and optimization FMU might as well include a model of the power system (e.g., in case of an optimal power flow algorithm), the level of accuracy and detail would not necessarily be as high as in the power system FMU. In control and optimization, we are mostly interested in arriving to good solutions (e.g., through convex relaxations) and at the same time having a tractable problem, and low computation times. Toward these efforts, we may introduce simplifications to our models. Therefore, it is necessary for the power system FMU to confirm that the determined control actions result to a feasible and reasonable operating point.

For the use cases presented in this paper, we have implemented two controllers: 1) a power system loading controller, which monitors the loading of cables and transformers, and if this exceeds a certain threshold it issues a DR signal (see Section XI-C); and 2) a volt/var controller, which appropriately controls the reactive power injection of a battery storage to track a voltage reference signal (see Section XI-D). First efforts have also been carried out for a ramp-rate controller, where a battery storage assists in the reduction of steep power increases or decreases, and, thus, reduces the need for dispatching expensive power reserves. This will be presented in future work.

IX. TIME INTEGRATION OF DIFFERENTIAL EQUATIONS USING QSS METHODS

As described in Section IV, each building's FMU defines ordinary differential equations of the form

$$\dot{x} = f(x, u, t) \tag{3}$$

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 QSS methods in Ptolemy [30], [31]. QSS differs from typical integration methods, in that it discretizes the state variables rather than time. Thus, (3) becomes

$$\dot{x} = f(q, \mu, t) \tag{4}$$

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

$$\widehat{q}_{j[\ell]}(t) = \sum_{i=0}^{M-1} q_{j[\ell]}^{[i]} \left(t - t_{j[\ell]}^{q}\right)^{i} \tag{5}$$

where $q_{j[\ell]}^{[i]}$ denotes the ith polynomial coefficient for the ℓth 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, $\widehat{q}_{j[\ell]}(t) = \widehat{q}_{j[\ell]}^{[0]}$. Each model holds on to $t_{j[\ell]}^{q} \le t < t_{j[\ell+1]}^{q}$ (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

$$\widehat{x}_{j[k]}(t) = \sum_{i=0}^{M} x_{j[k]}^{[i]} \left(t - t_{j[k]}^{s}\right)^{i} \tag{7}$$

valid on $t_{j[k]}^{s} \le t \le t_{j[k+1]}^{s}$. Note that the state-event times $t_{j[k]}^{s}$ may differ from the quantization-event times. Fig. 11 shows a block diagram of a QSS integrator.

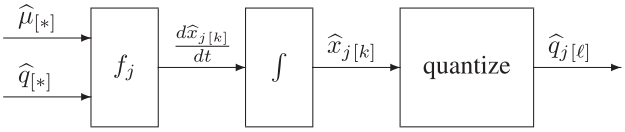


Fig. 11. *QSS integration of a component of an ODE system. The quantized state is a piecewise-continuous approximation to x(t). The simulation iteratively updates the state and quantized state models for individual components.*

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

$$x_{j[k]}^{[0]} = x_{j[k-1]}\left(t_{j[k]}^{s}\right) \tag{7}$$

$$x_{j[k]}^{[1]} = f_j\left\{\left(\widehat{q}_{[*]}, \widehat{\mu}_{[*]}, t_{j[k]}^{s}\right)\right\} \tag{8}$$

where the models $\widehat{q}_{[*]}$ and $\widehat{\mu}_{[*]}$ 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.

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

$$\left|\widehat{x}_{j[k]}\left(\widehat{t}_{j[\ell+1]}^{q}\right) - \widehat{q}_{j[\ell]}\left(\widehat{t}_{j[\ell+1]}^{q}\right)\right| = \Delta Q_j \tag{9}$$

where $\widehat{t}_{j[\ell+1]}^{q}$ is the predicted quantization-event time for component j. In practice, ΔQ_j varies with the magnitude of $\widehat{q}_{j[k]}^{[0]}$, 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 requantize 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

$$q_{j[\ell]}^{[0]} = \widehat{x}_{j[k]}\left(t_{j[\ell]}^{q}\right) \tag{10}$$

$$q_{j[\ell]}^{[1]} = \frac{d\widehat{x}_{j[k]}}{dt}\left(t_{j[\ell]}^{q}\right) \tag{11}$$

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. 12 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 $0.001 \cdot |q_{[\ell]}^{[0]}|$. Compared to the analytical result $x = e^{-t}$, both solutions end at $t = 3$ with a global error less than $5 \cdot 10^{-4}$.

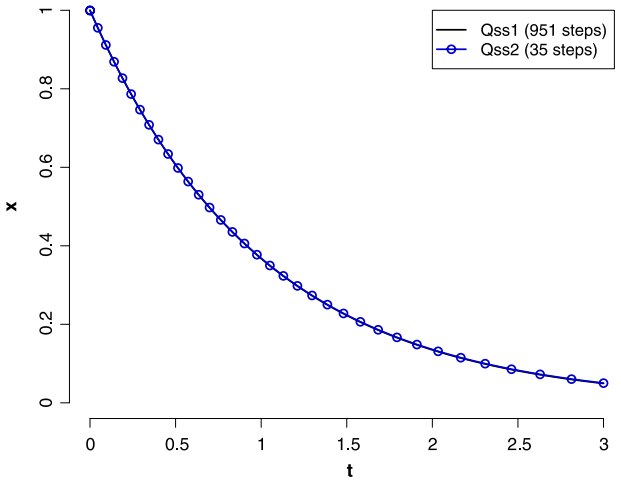


Fig. 12. *QSS solution of the exponential problem $\dot{x} = -x$.*

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 cosimulation FMUs. As suggested by Fig. 4, the power system 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 [4]. 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 runtime. 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 cosimulation. To compose multiple actors in order to conduct a cosimulation, 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 system 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 communication FMUs lead to time delays in the signals. They take signal $u_j(t)$ as inputs, for some $j \in \{1, \ldots, n\}$, where n is a fixed number of channels, and produce the signal at the output after some time delay $\delta_j(t)$. 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 $i \in \{0, 1, \ldots\}$, it outputs the control action $y((i+1)\delta) = f(u(i\delta))$.

Fig. 13 shows the Ptolemy II system model that combines the power, building, communication, and control 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 cosimulation. In addition, other Ptolemy II actors that work in the discrete event domain can be used in such system models.

Such a cyber–physical system has component subsystems that operate with varying time scales. In our model, the power system FMU is currently running as a sequence of steady states (sequential power flows), while the communication and control FMUs are running as discrete-event simulations. In future work, we will also enable dynamic simulations for the power system FMU. The building FMU is originally a continuous dynamic model, but through our QSS implementation we can run it as an event-based system, decoupling the fast from the

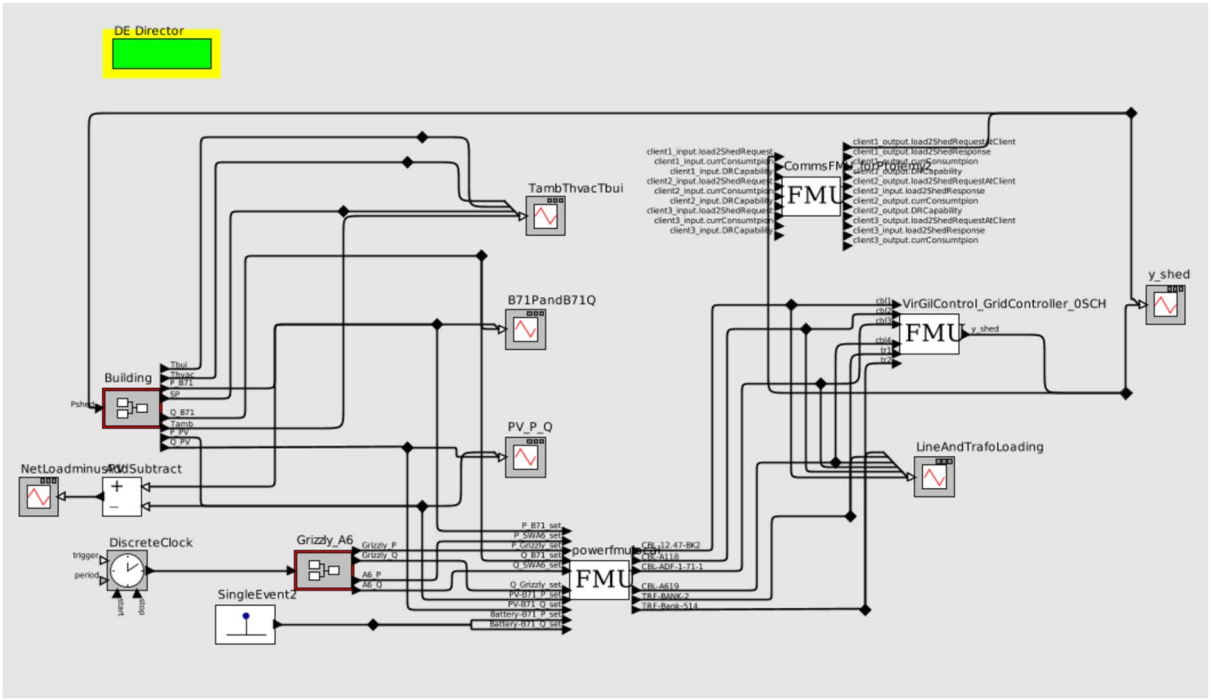


Fig. 13. ***Ptolemy II model that shows the composition of the different FMUs.***

slow dynamics, e.g., change of HVAC electric power (fast) versus change in building thermal mass temperature (slow). As a result, the building model is evaluated when its DR signal changes or when the QSS algorithm requires a new evaluation. Converting the continuous-time semantics to discrete event semantics through the use of QSS allows us to update the states of these subsystems at the rate required by their respective dynamics.

XI. SIMULATION RESULTS

A. Calibration of the Building Model

To demonstrate VirGIL, we used a real building inside the Lawrence Berkeley National Laboratory (LBNL) campus. LBNL building 71 is a 54 000-ft^2 two story steel-frame office and laboratory building located in Berkeley, CA, USA. 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 A.M. to 6:00 P.M. and high equipment usage during off hours. The peak electric power demand is over 400 kW during the period of 12:00 P.M. to 6:00 P.M. and the average demand during the off hours is about 80 kW.

The building FMU has been generated as described in Section VI. 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. Here, we have assumed that the EnergyPlus model is already calibrated against real measured data. As long as enough sensor data become available, the EnergyPlus model is calibrated based on the procedure documented in [16]. 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 DR events as well as for critical peak pricing.

Both the RC and EnergyPlus models have been simulated using standard weather for San Francisco, lighting, plug loads, occupancy, and setpoint 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.

Fig. 14 shows the RC model cooling load versus the EnergyPlus model cooling load. Every points represents a

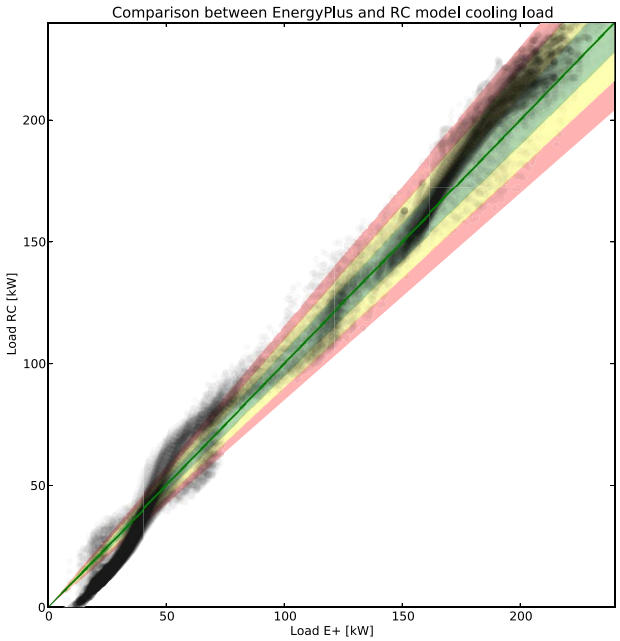


Fig. 14. ***Comparison of the cooling load predicted by EnergyPlus and the RC model.***

simulated data point with a 5-min resolution over the period between May and October. The green, yellow, and red areas, respectively, represent a relative error of ±5%, ±10%, and ±15%. As can be seen, the highest relative error occurs at low cooling load level while when the load is close to its maximum the almost totality of the points is within the ±15% interval.

Fig. 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 were collected and their distribution was compared. The blue boxes show the cooling load distribution for the EnergyPlus model while the green boxes show the distribution of the cooling load predicted using the RC model. The

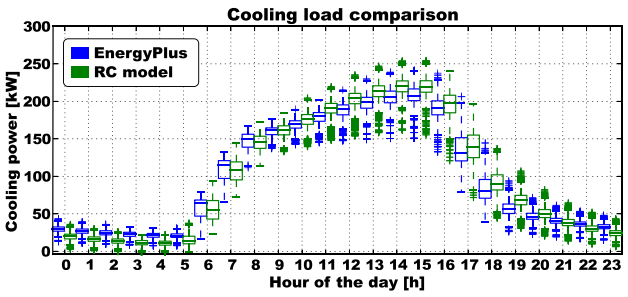


Fig. 15. ***Comparison of the cooling load predicted by EnergyPlus and the RC model.***

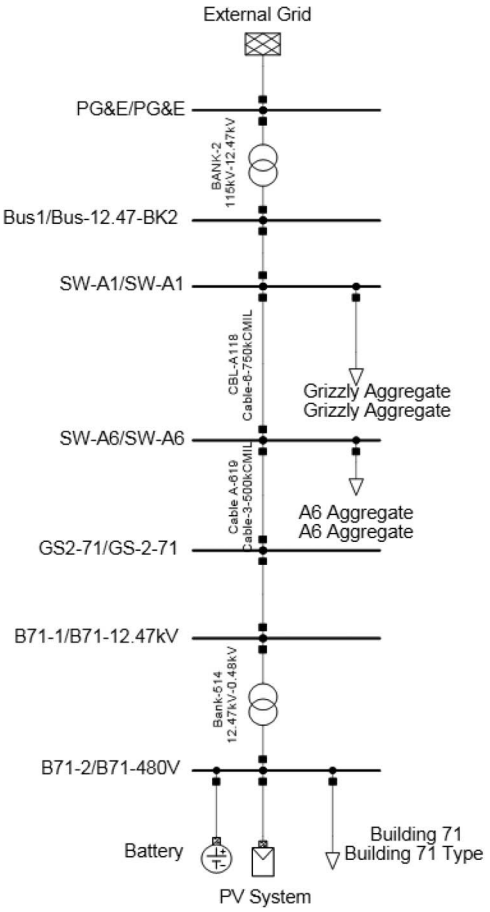


Fig. 16. ***PowerFactory model of the LBNL distribution system to building 71. The rest of the LBNL, except for the building 71, is modeled as aggregate loads on buses SW-A1 and SW-A6.***

highest relative errors shown in Fig. 15 happen exclusively at night time when the nonlinearities of the long wave radiation exchange that are not captured by the RC model dominate the heat transfer of the building. However, Fig. 15 helps us put in perspective this error, confirming that even if it might be considered high in relative terms, its impact on the ability of the RC model to predict the cooling load can be neglected.

Under these assumptions it is possible to consider the RC model detailed enough to describe the thermal dynamics of the building to be able to predict its cooling load with a good accuracy. The original RC model generated using the BRCM toolbox had 106 state variables. Model reduction techniques [18], [19] were used to reduce it to a model that is able to predict the same averaged building temperature as described in Section VI. The obtained reduced-order model has eight 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.

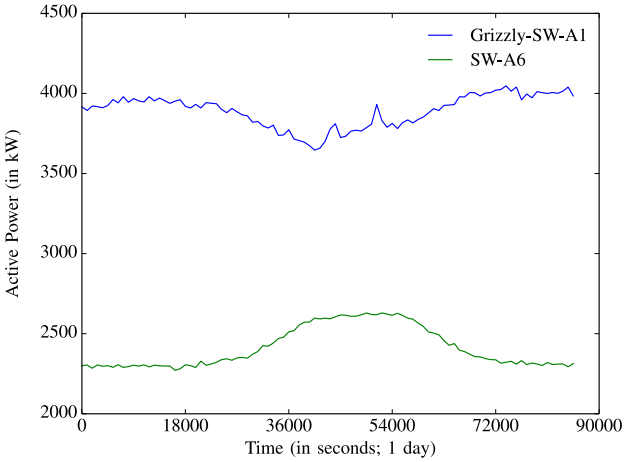


Fig. 17. *Active power consumption of the aggregate loads.*

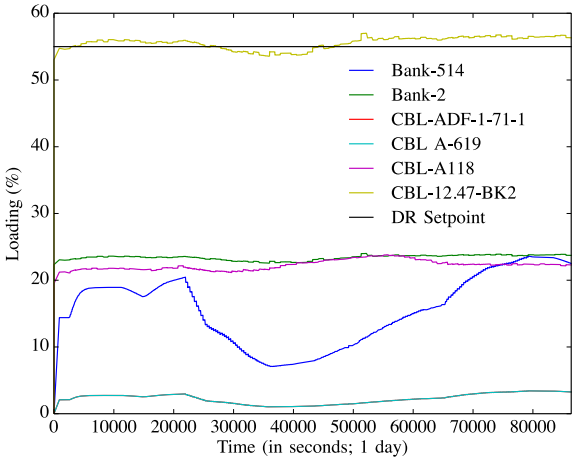


Fig. 19. *Cable and transformer loading (no DR).*

B. Overview of the Use Cases

Both cases use the LBNL distribution network and building 71. As shown in Fig. 16, 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-min 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 consumptions of the aggregate loads in SW-A1 and SW-A6 are shown in Fig. 17.

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. 18. 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.

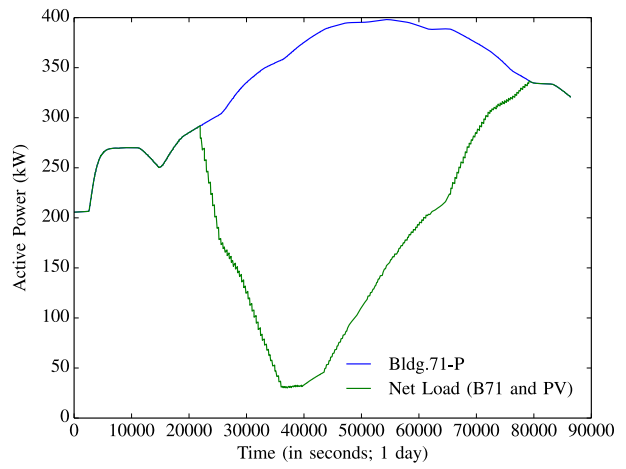


Fig. 18. *Active power consumption of building 71 without DR (blue) and net load at bus B71 (green; sum of load and PV infeed).*

We present two use cases in the following sections to demonstrate the capabilities of VirGIL. The first applies to DR actions in building 71 to reduce the cable and transformer loading. The second applies to volt/var control so that the voltage at bus B71 follows specified setpoints.

C. Demand Response in Building 71

Fig. 19 presents the transformer and cable loadings of the LBNL network during a typical day, when no DR 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 DR actions at 55%. If any cable or transformer loading exceeds the 55% threshold, an automated DR signal is sent to the building, reducing its power consumption by 20%. The 20% value is determined through the DR potential of the building, considering the building materials, use, and geometry, as well as its electricity consumption. Tools such as the DRQAT [16] can be used to determine the potential for DR participation in each building and the corresponding actions to achieve it, e.g., HVAC temperature setpoint change, light dimming, etc. The communication 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

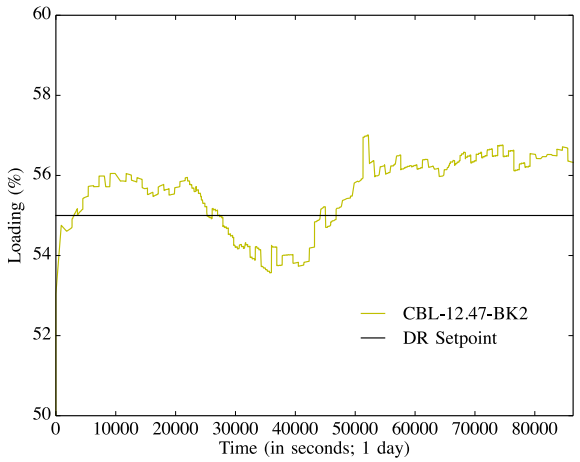


Fig. 20. ***Loading of the most critical cable and setpoint for DR actions.***

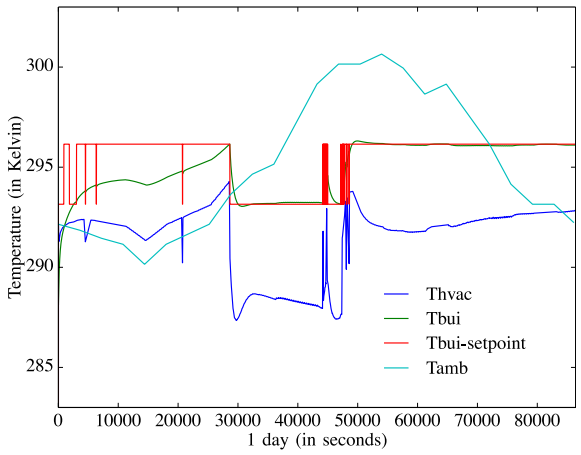


Fig. 22. ***Temperature variations in building 71 with active DR. Tamb: ambient temperature; Tbui-setpoint: setpoint of the temperature setpoint inside the building (varies based on DR actions); Tbui: average temperature inside the building; Thvac: setpoint at the HVAC controller.***

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

Fig. 20 zooms in the most critical cable loading, when no DR action is taken. We expect DR signals to be issued from midnight until 7.30 A.M. ($t = 0$ s to $t = 27000$ s), and then again after about 1 P.M. (from $t = 46000$ s and onwards).

Fig. 21 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. Fig. 22 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 h of the day, then decreases to its nominal temperature of 293 K (20 °C) and increases back again after 1 P.M. ($t = 46000$ s and onwards). We observe some oscillations in the DR signal at $t \in [46000, 50000]$. This is because the cable loading is marginally above or below 55%. Although for a significant change in the loading of cable CBL-12.47-BK2 we would need a larger number of buildings participating in DR, in this case we are still able to observe the interactions between power system (cable loading), building operation, and DR signal dispatch (control) that result in this oscillating behavior. Future improvements in the controller would involve a hysteresis loop to avoid this oscillating behavior. As the day starts and the ambient temperature increases, we see that "Tbui" increases as well. At around 7.30 A.M. ($t = 27000$ s) it reaches the "Tbui-setpoint." From that point on, the average zone temperature follows the "Tbui-setpoint." Going back to Fig. 21, we observe that if DR is activated, the building consumes no additional power until around 7.30 A.M. ($t = 27000$ s), 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 around 1 P.M. ($t = 46000$ s), when DR is activated again, the building consumption is reduced in comparison to the baseline.

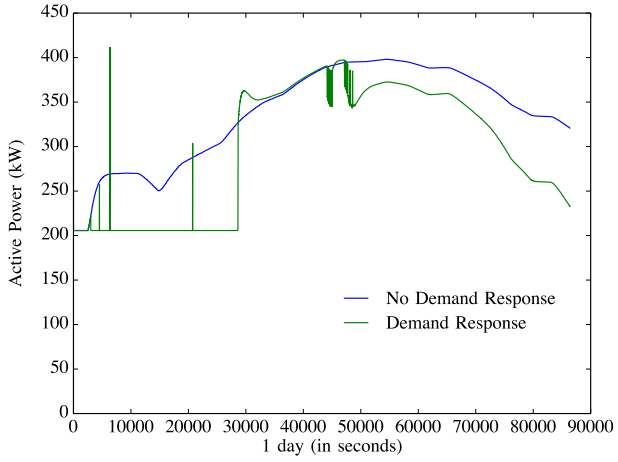


Fig. 21. ***Active power consumption of building 71 with and without DR actions.***

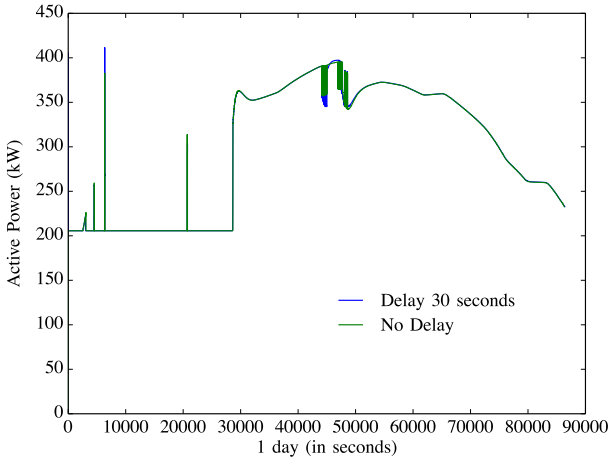


Fig. 23. *Active power consumption of building 71 with DR, with and without signal communication delays.*

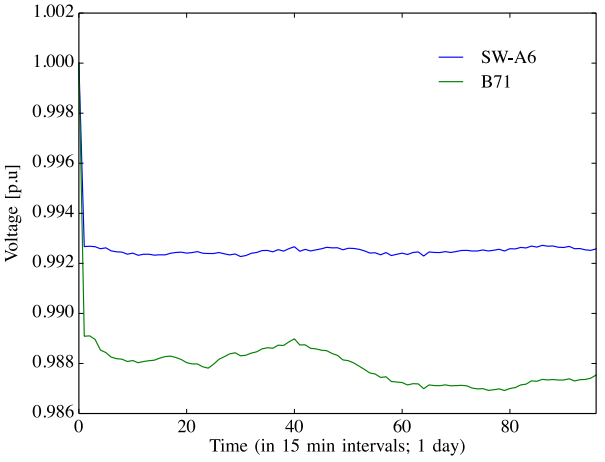


Fig. 24. *Without volt/var control: Voltage level of bus SW-A6 and bus B71.*

Fig. 23 compares the active power consumption of building 71 when there are no communication delays. In our base case, shown in Fig. 21, the polling frequency of the DR client in the communication FMU was set at 30 s, which means that every DR signal was sent with 30 s. The actual communication delays that were also modeled in this setup did not exceed on average 500 ms. In Fig. 23, 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 $t = 45000$ 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 able to represent the effect of the communication delays, and measure the effect of the building actions back to the power grid. In [2], we investigated further the effect of the communication infrastructure in a hardware-in-the-loop setting.

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 microphasor measurement units (μPMUs) [32]. One of them is located at bus SW-A6 and one more at bus B71. μPMUs 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 μPMU 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:

$$U_1^2 = (U_2 + (R \cdot P + X \cdot Q)/U_2)^2 + (X \cdot P - R \cdot Q)^2/(U_2)^2 \quad (12)$$

where

$$U_1 = U_{\mathrm{SW-A6}} \quad (13)$$
$$U_2 = U_{\mathrm{B71}} \quad (14)$$
$$P = P_{\mathrm{B71}} - P_{\mathrm{BAT}} - P_{\mathrm{PV}} \quad (15)$$
$$Q = P_{\mathrm{B71}} - Q_{\mathrm{BAT}} - Q_{\mathrm{PV}} \quad (16)$$
$$R = R_{\mathrm{Bank-514}} + R_{\mathrm{CBL-ADF-1-71-1}} + R_{\mathrm{A-619}} \quad (17)$$
$$X = R_{\mathrm{Bank-514}} + X_{\mathrm{CBL-ADF-1-71-1}} + X_{\mathrm{A-619}}. \quad (18)$$

Fig. 24 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 6 A.M. until about 10 A.M. ($t = 24$ to $t = 40$ quarters), 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 presents the same voltages, but with volt/var control, so that V_{B71} tracks the voltage $V_{\mathrm{SW-A6}}$ at bus SW-A6. The required reactive power infeed from the battery is presented in Fig. 26. In the same figure, we also present

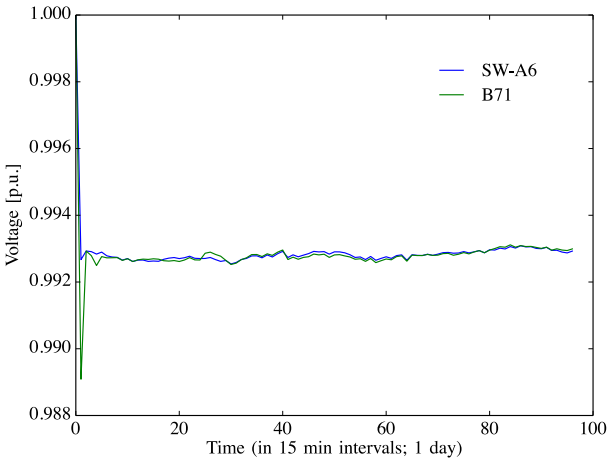


Fig. 25. *With volt/var control: Voltage level of bus SW-A6 and bus B71.*

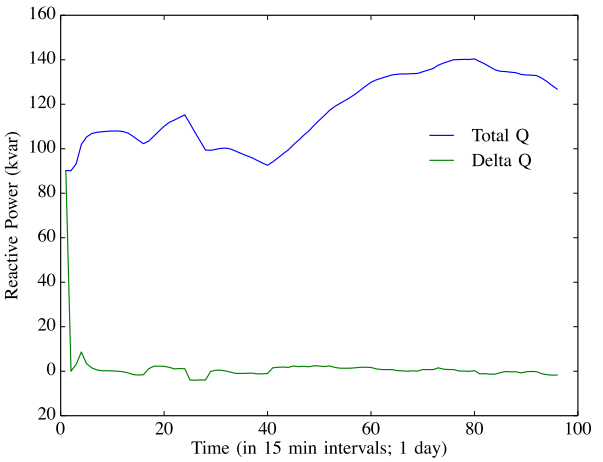


Fig. 26. *Reactive power injection of the battery. Total Q corresponds to the reactive power injected. Delta Q is the change in reactive power from the previous timestep.*

ΔQ, i.e., the difference by which the Q setpoint should be adjusted from one timestep to another.

XII. CONCLUSION

VirGIL creates a modular cosimulation platform for studying in detail the impact of DR and other controls on power 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 DR in their daily operation. VirGIL uses the industry-standard FMI, which encourages a modular approach to instantiating and sharing models, and allows a simple exchange and testing of different cosimulation tools.

This paper presented the development of FMUs for cosimulation 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 DR protocol. A full representation of the communication network was also included. To our knowledge, this is the first time that a cosimulation platform has coupled commercial power system software such as PowerFactory with building and communication models to study the impact of DR actions on the distribution grid. A further contribution of this paper is the full integration of the QSS methods for simulation in VirGIL.

Ptolemy II, the VirGIL implementation framework, handles both continuous and discrete-event simulations, and supports both FMIs for model exchange and cosimulation.

Future extensions of this work will include electric vehicles, power system optimization, and advanced building controls. Use cases will include DR applications for volt/var optimization, three-phase asymmetries, and distribution system planning. Real case studies, such as the DR potential and impact in regions of Southern California, will be simulated in VirGIL and presented. ■

Acknowledgment

The authors would like to thank E. Lee and C. Brooks from the University of California Berkeley for their support in the FMI and QSS implementation in Ptolemy II.

REFERENCES

[1] M. Wetter, "Co-simulation of building energy and control systems with the building controls virtual test bed," *J. Building Performance Simul.*, vol. 4, no. 3, pp. 185–203, Nov. 2011.

[2] S. Rotger-Griful *et al.*, "Hardware-in-the-loop co-simulation of distribution grid for demand response," in *Proc. 19th Power Syst. Comput. Conf.*, Genoa, Italy, Aug. 2016.

[3] Modelica Association Project FMI, "Functional mock-up interface for model exchange and co-simulation," Jul. 2014. [Online]. Available: https://www.fmi-standard.org

[4] C. Ptolemaeus, *System Design, Modeling, Simulation Using Ptolemy II*. Berkeley, CA, USA: Ptolemy.org, 2014. [Online]. Available: http://ptolemy.org/books/Systems

[5] E. A. Lee, M. Niknami, T. S. Nouidui, and M. Wetter, "Modeling and simulating cyber-physical systems using cyphysim," EMSOFT, Oct. 2015. [Online]. Available: http://simulationresearch.lbl.gov/wetter/download/2015-LeeEtAl CyPhySim EMSOFT.pdf.

[6] K. Hopkinson *et al.*, "EPOCHS: A platform for agent-based electric power and communication simulation built from

commercial off-the-shelf components," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 548–558, May 2006.

[7] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili, "Power system and communication network co-simulation for smart grid applications," in *Proc. IEEE PES Innovative Smart Grid Technol.*, DOI: 10.1109/ISGT.2011.5759166.

[8] M. Lévesque, D. Q. Xu, G. Joós, and M. Maier, "Communications and power distribution network co-simulation for multidisciplinary smart grid experimentations," in *Proc. 45th Annu. Simul. Symp.*, San Diego, CA, USA, 2012, pp. 2:1–2:7.

[9] R. Bottura *et al.*, "SITL and HLA co-simulation platforms: Tools for analysis of the integrated ICT and electric power system," in *Proc. IEEE EUROCON*, pp. 918–925.

[10] M. Stifter *et al.*, "Co-simulation of components, controls and power systems based on open source software," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, DOI: 10.1109/PESMG.2013.6672388.

[11] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: Challenges, tools and methods," *IEEE Trans. Syst. Man Cybern., Syst.*, vol. 44, no. 3, pp. 318–326, Mar. 2014.

[12] D. Chassin, K. Schneider, and C. Gerkensmeyer, "GridLAB-D: An open-source power systems modeling and simulation environment," in *Proc. IEEE PES Transm. Distrib. Conf. Expo.*, DOI: 10.1109/TDC.2008.4517260.

[13] S. Muller, H. Georg, C. Rehtanz, and C. Wietfeld, "Hybrid simulation of power systems and ICT for real-time applications," in *Proc. 3rd IEEE PES Int. Conf. Exhibit. Innovative Smart Grid Technol.*, DOI: 10.1109/ISGTEurope.2012.6465734.

[14] T. Nouidui, M. Wetter, and W. Zuo, "Functional mock-up unit for co-simulation import in energyplus," *J. Building Performance Simul.*, vol. 7, no. 3, pp. 192–202, 2014.

[15] D. Sturzenegger, D. Gyalistras, V. Semeraro, M. Morari, and R. Smith, "BRCM matlab toolbox: Model generation for model predictive building control," in *Proc. Amer. Control Conf.*, DOI: 10.1109/ACC.2014.6858967.

[16] 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 Buildings*, vol. 42, no. 7, pp. 967–975, Jul. 2010.

[17] Commercial Prototype Building Models. [Online]. Available: http://www.energycodes.gov/commercial-prototype-building-models

[18] K. Glover, "All optimal hankel-norm approximations of linear multi-variable systems and their L-inf error bounds," *Int. J. Control*, vol. 39, no. 6, pp. 1115–1193, 1984.

[19] K. Zhou, "Frequency-weighted model reduction with L error bounds," *Syst. Control Lett.*, vol. 21, no. 2, pp. 115–125, 1993.

[20] M. Wetter, W. Zuo, T. S. Nouidui, and X. Pang, "Modelica buildings library," *J. Building Performance Simul.*, vol. 7, no. 4, pp. 253–270, 2014.

[21] OMNeT++. [Online]. Available: https://www.omnetpp.org/

[22] T. Godfrey *et al.*, "Modeling smart grid applications with co-simulation," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, 2010, pp. 291–296.

[23] C. Müller, H. Georg, and C. Wietfeld, "A modularized and distributed simulation environment for scalability analysis of smart grid ICT infrastructures," in *Proc. 5th Int. Conf. Simul. Tools Tech.*, 2012, pp. 327–330.

[24] K. Mets, J. A. Ojea, and C. Develder, "Combining power and communication network simulation for cost-effective smart grid analysis," *IEEE Commun. Surv. Tut.*, vol. 16, no. 3, pp. 1771–1796, 2014.

[25] M. Bredel and M. Bergner, "On the accuracy of IEEE 802. 11 g wireless LAN simulations using OMNeT++," in *Proc. 2nd Int. Conf. Simul. Tools Tech.*, 2011, Art. no. 81.

[26] A. Virdis, G. Stea, and G. Nardini, "SimuLTE—A modular system-level simulator for LTE//LTE—A networks based on OMNeT ++," in *Proc. SimulTech*, 2014, pp. 28–30.

[27] E. Weing, H. Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *Proc. IEEE Int. Conf. Commun.*, 2009, DOI: 10.1109/ICC.2009.5198657.

[28] Openadr Alliance. [Online]. Available: http://www.openadr.org/

[29] Demand Response Research Center, "Openadr." [Online]. Available: http://drrc.lbl.gov/openadr

[30] F. E. Cellier and E. Kofman, *Continuous System Simulation*. New York, NY, USA: Springer-Verlag, 2006.

[31] G. Migoni, M. Bortolotto, E. Kofman, and F. E. Cellier, "Linearly implicit quantization-based integration methods for stiff ordinary differential equations," *Simul. Model. Practice Theory*, vol. 35, pp. 118–136, 2013.

[32] E. Stewart *et al.*, "Addressing the challenges for integrating micro-synchrophasor data with operational system applications," in *Proc. PES Gen. Meeting Conf. Expo.*, DOI: 10.1109/PESGM.2014.6938994.

ABOUT THE AUTHORS

Spyros Chatzivasileiadis (Member, IEEE) received the Diploma in electrical and computer engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 2007 and the Ph.D. from ETH Zurich, Switzerland, in 2013, with focus on the integration of HVDC lines in power system planning and operation.

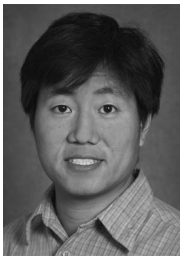
He is a Postdoctoral Researcher at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Until July 2015, he was with Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA, where he worked on dc microgrids and cosimulation tools to integrate demand response in power systems. His research interests are on power system optimization and control.

Javier Matanza received an electrical engineering degree from the Polytechnic University of Valencia, Valencia, Spain, in 2008 and the Ph.D. degree from the Comillas Pontifical University, Madrid, Spain, in 2013.

He is a Research Professional at the Institute for Research in Technology (IIT) and an Assistant Professor at the Comillas Pontifical University. His current interests are in powerline communication technologies and in communication network simulations.

Marco Bonvini received the M.S. degree in computer science and the Ph.D. degree in systems and controls from the Politecnico di Milano, Milan, Italy, in 2009 and 2013, respectively.

Later he worked at Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA, where he developed building simulation models and integrated them with optimization and fault detection algorithms to improve the operation of buildings. Currently, he is a Data Scientist at Whiskerlabs, Oakland, CA, USA, where he focuses on nonintrusive load monitoring.



Rongxin Yin received the M.Sc. degree in mechanical engineering from Tongji University, China, in 2009 and the M.Sc. degree in building science from University of California at Berkeley, CA, USA, in 2013.

He is a Senior Scientific Engineering Associate in the Grid Integration Group, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA. He has been at LBNL since 2007 and focuses his career on building science, energy and buildings, and demand to grid. His research focuses on energy efficiency and demand response (EE&DR) in commercial buildings.

Thierry S. Nouidui received the M.S. degree in electrical engineering from the University of Kiel, Kiel, Germany, in 2004 and the Ph.D. degree in building physics from the University of Stuttgart, Stuttgart, Germany, in 2008.

He is a Principal Scientific Engineering Associate in the Simulation Research Group, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA. His current research is focused on developing and deploying new generation of building performance computational tools for the design and operation of buildings and communities.

Emre C. Kara (Member, IEEE) received the M.Sc. degree in building engineering from Delft University of Technology, Delft, The Netherlands, in 2010 and the Ph.D. degree in civil and environmental engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2014.

He is currently a Postdoctoral Researcher at Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA. His research interests include understanding the energy use of buildings, modeling, and control of building loads to provide demand response services to increase power system reliability.

Rajiv Parmar received the B.Sc. degree in computer engineering and the M.Eng. degree in electrical engineering specializing in energy and environment from the University of Calgary, Calgary, AB, Canada, in 2004 and 2011, respectively.

Until June 2015 he was a Software Developer at Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA. He has several years of experience in the software industry in Canada and the United States, in areas such as energy, demand response, SCADA systems, communications, quality, and security.

David Lorenzetti received the B.S. degree in electrical and comuputer engineering, from the University of Cincinnati, in 1986, and the M.S. and Ph.D. degrees in building technology, from Massachusetts Institute of Technology, in 1992 and 1997 respectively.

Currently he is with Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA. His research interests include whole-building airflow, pollutant transport, and energy; and applying numerical simulation to study problems of life safety and energy conservation in buildings.

Michael Wetter received the B.S. degree in energy and building technologies from the University of Applied Sciences at Luzern, Switzerland, in 1995, and the Ph.D. degree in mechanical engineering from the University of California, Berkeley, CA, USA, in 2004.

He is a Computational Staff Scientist and the Deputy Leader of the Simulation Research Group at Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA. His research includes integrating building performance simulation tools into the research process, as well as their use for design and operation of buildings. At LBNL, he is developing the Modelica Buildings library for building energy and control systems, a prototype implementation of a new computing engine for EnergyPlus, cosimulation tools based on the functional mockup interface standard, the building controls virtual testbed software for cosimulation and model-based operation and CyPhySim, a cyber–physical system simulator based on Ptolemy II. As the cooperating agent of the IEA EBC Annex 60, he leads a research team from 16 countries and 40 institutes in the development of new generation computational tools for buildings and community energy systems between 2012 and 2017.

Dr. Wetter is the youngest Fellow of the International Building Performance Simulation Association (IBPSA), a recipient of the biannual Outstanding Young Contributor Award of IBPSA, the Treasurer of IBPSA, and a Board Member of the Modelica North America Users Group; was President of the U.S. Affiliate of IBPSA; and is a Member of ASHRAE and of the Modelica Association.

Sila Kiliccote (Member, IEEE) received an Electrical Engineering degree from the University of New Hampshire, Durham, NH, USA, in 1994, and the M.S. degree in building science from Carnegie Mellon University, Pittsburgh, PA, USA, in 1995.

She is the Smart Grid leader at SLAC National Accelerator Laboratory, Menlo Park, CA, USA, and holds a part-time position as a demand response expert at Google. Prior to joining SLAC, she spent over ten years at Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA, as a Deputy of the Demand Response Research Center and leading the grid integration initiatives, where she worked with teams to develop OpenADR, Virtual Grid Integration Laboratory (VirGIL), and explored the use of models with data from micro-PMUs for distribution systems.