IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 June 2024, accepted 6 July 2024, date of publication 10 July 2024, date of current version 9 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3426333

RESEARCH ARTICLE

A Real-Time T&D Co-Simulation Platform for Testing Grid Services of Distributed Energy Resources

VICTOR DALDEGAN PADUANI[1], **(Member, IEEE), RAHUL KADAVIL**[1], **(Senior Member, IEEE), HOSSEIN HOOSHYAR**[1], **(Senior Member, IEEE), ABOUTALEB HADDADI**[2], **(Senior Member, IEEE), AHMED SAAD**[3], **(Member, IEEE), A. H. M. JAKARIA**[3], **(Member, IEEE), AND AMINUL HUQUE**[3], **(Member, IEEE)**
[1]Advanced Grid Innovation Laboratory for Energy, New York Power Authority, Albany, NY 12203, USA
[2]Transmission Operations and Planning, Electric Power Research Institute, Palo Alto, CA 94304, USA
[3]DER Integration, Electric Power Research Institute, Knoxville, TN 37932, USA

Corresponding author: Victor Daldegan Paduani (victor.daldeganpaduani@nypa.gov)

This work was supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) through the Solar Energy Technologies Office under Award DE-EE0009021.

ABSTRACT This paper develops a real-time (RT) transmission and distribution (T&D) co-simulation platform for testing distributed energy resources management system (DERMS) algorithms. The platform consists of a transmission system modeled within a real-time transient-stability type environment interfaced to an active distribution network modeled within a fundamental frequency phasor-domain platform. The data exchange and time synchronization between the T&D models has been established via MQTT communication protocol, enabling the platform to communicate with a large number of DER models in real-time. The developed platform is generic in the sense that it can integrate an arbitrarily sized transient stability-type model of a transmission system with an arbitrarily sized fundamental frequency phasor-domain model of a distribution system. The platform does not impose an inherent limit on the size of the T&D models, and the size is limited by the computational capability of the simulation computer. This feature enables utilizing the platform for simulation studies of large-scale T&D systems. Furthermore, the developed platform enables both off-line and real-time simulations, extending its application from off-line planning-type studies to real-time operation-type and hardware-in-the-loop (HIL) studies. The paper demonstrates the application of the developed platform in a large-scale case study utilizing behind-the-meter (BTM) DERs to provide grid services when controlled by a DERMS. By including the propagation of dynamics between the simulation domains, realistic DER models, and typical communication protocols that are used in the field, the proposed platform enables the testing of DERMS algorithms in a close to real environment, replicating a wide range of power system phenomena that highlight the impact of a DERMS in an actual power system.

INDEX TERMS Co-simulation, DERMS, distribution system simulator, real-time simulation, MQTT.

NOMENCLATURE

A-DERMS	Aggregator DERMS.
BTM	Behind-the-meter.
DER	Distributed Energy Resource.
DERMS	DER Management System.
DSS	Distribution System Simulator.
Dx	Distribution System.
EITS	Eastern Interconnection Tx.
ESS	Energy Storage System.
GSF	Grid Support Functionality.
HIL	Hardware-in-the-loop.
IoT	Internet of Things.

The associate editor coordinating the review of this manuscript and approving it for publication was Salvatore Favuzza.

L-DERMS	Local DERMS.
MQTT	Message Queuing Telemetry Transport.
PV	Photovoltaic.
RES	Renewable Energy Sources.
RT	Real-Time.
RTS	Real-Time Simulator.
SOC	State of Charge.
T&D	Transmission and Distribution.
TS	Transient-Stability.
Tx	Transmission System.
UDP	User Datagram Protocol.

I. INTRODUCTION

Recent industry trends allow aggregated Distributed Energy Resources (DERs) to participate in capacity, energy, and ancillary service markets operated by Regional Transmission Organizations and Independent System Operators (RTOs/ISOs) [1]. Enabling such services in a reliable and economically efficient manner requires the development of fundamentally new controls and harmonization of Transmission System Operator (TSO), Distribution System Operator (DSO), and Aggregator efforts. Research, development, and demonstration of these concepts requires advanced analysis and simulation capabilities beyond those typically used today.

Traditionally, transmission and distribution (T&D) utilities have independently analyzed their system using a rather simplified equivalent representation of the system of the other entity. Accordingly, simulation tools utilized by T&D utilities have also been inhomogeneous, adapting to the different time scale of the power system phenomena being studied. However, such traditional tools and analysis methods may not be sufficient for studying T&D system services of DERs and their potential impacts on the distribution system (Dx). This study requires a unified simulation platform incorporating both T&D system models, which is addressed in this paper.

Local control functions may not always be sufficient for addressing the challenges of operating a grid under very high DER participation, as it may cause voltage control problems [2]. In addition, having direct control over the DERs operation via a DER management system (DERMS) could improve grid performance by considering DERs' availability, system operator requests, and grid constraints [3]. Therefore, in September 2020, FERC Rule 2222 enabled T&D system operators to utilize behind-the-meter (BTM) DERs to provide services to the power grid [1] by allowing their participation in wholesale markets via aggregators, which was forbidden until then.

In [4], Garg et al. propose a hierarchical DERMS architecture composed by Aggregator DERMS (A-DERMS) and local DERMS (L-DERMS). While L-DERMS agents are responsible for directly controlling BTM DERs setpoints, the A-DERMS is responsible for receiving power requests from system operators, solving an optimization including grid constraints, and issuing power requests to L-DERMS controllers. Figure 1 demonstrates how this hierarchical control structure can be achieved. One of the main challenges in testing DERMS algorithms is the simulation size needed. This is because in order to demonstrate how a DERMS algorithm can effectively provide transmission system (Tx) grid services from BTM DERs, the simulation must be able to include the propagation of dynamics between T&D systems [5]. Furthermore, it must be capable of simulating hundreds to thousands of devices equipped with grid support functionalities (GSFs), as well as the communication links between DERMS and the devices. Currently, typical power system softwares such as PSS/e, CYME, or OpenDSS [6] are not suitable for running a simulation with this level of complexity; therefore, custom made co-simulation models must be developed [7], [8], [9].

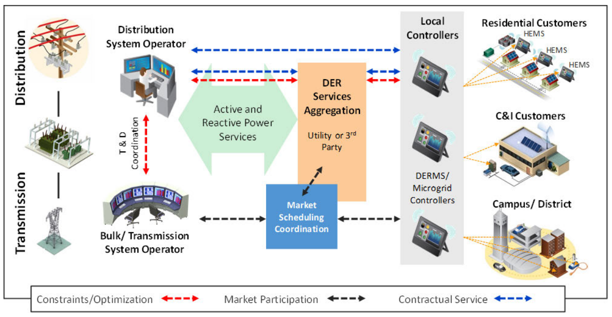


FIGURE 1. Example of the hierarchical DERMS structure proposed in [4] for providing grid services to grid operators.

By including the Tx transient-stability (TS) dynamics as well as the Dx unbalanced grid conditions, T&D co-simulations are a strong candidate for including the propagation of dynamics between Tx and Dx [10], [11]. In addition, if implemented with a real-time (RT) simulator, a co-simulation testbed can not only include the communication links between devices in RT operation [12], [13], [14], [15], but also enable hardware-in-the-loop (HIL) simulations in which a real DER controller or a protection equipment can be included in the loop to test its performance under various scenarios [16].

In [17], Poudel et al. introduced a modeling environment for testing DERMS with the GridAPPS-D platform. As one of the pioneers in this field, the platform demonstrates how DERs can be controlled to provide grid services in RT utilizing the GridAPPS-D platform. However, the proposed method is limited by the GridAPPS-D capabilities, which is focused on Dx systems, and does not include Tx system dynamics. Moreover, the method was validated in a small scenario (IEEE 13-bus feeder model). Therefore, in this work, we present our approach for the development of a RT T&D co-simulation framework that can be used to test DERMS algorithms in large-scale HIL simulations. The DERs are represented by DER emulators developed by the Electric Power Research Institute (EPRI), which are equipped with typical GSFs and are integrated into the Dx simulator via Message Queuing Telemetry Transport (MQTT) communication protocol [18]. The proposed framework is validated in a large-scale use case including a section of the Eastern Interconnection Transmission System (EITS) containing realistic Dx feeders from local utilities. The contributions of the work to the literature are as follows:

- Introduce a RT T&D co-simulation architecture to validate DERMS algorithms in HIL simulations including the propagation of dynamics between Tx and Dx.
- Present methods to address scalability that arise during the development of large-scale scenarios.
- Demonstrate the proposed framework with a large-scale HIL including hundreds of BTM DERs with enabled GSFs in real-life feeder models connected to the EITS.

The work is divided as follows: Section II introduces the RT T&D co-simulation architecture as well as the methods to reduce the implementation efforts, Section III demonstrates the usage of the testbed in a large-scale simulation of a section of the EITS, and Section IV discusses the main findings of the work and future directions.

II. T&D CO-SIMULATION ARCHITECTURE

An overview of the different components constituting the RT co-simulation architecture and their interfaces are shown in Fig. 2. The system consists of the following main agents: (a) a RT simulator including a TS model of the bulk power system, (b) Dx system simulators (DSS), (c) an MQTT broker, (d) DER emulators, and (e) a DERMS algorithm to be tested. In this case, the DERMS hierarchical structure from [4] is utilized. As shown in the figure, the main coupling between the agents is executed by the MQTT broker, which is developed with thread-based parallelism [19] to accelerate the data exchange between RT and non-RT components, maintaining high speed as the number of DERs scales up. This allows for a dynamic and automated approach for introducing DERs into the closed-loop model.

The data exchange between the RT simulator and the external PC is performed via User Datagram Protocol (UDP), whereas the exchange between every other component is performed via the MQTT broker. In addition to the aforementioned agents, the co-simulation includes communication adapter functions built in Python programming language (*mqtt_rts*, *mqtt_dss*), which are responsible for initialization, e.g., defining MQTT clients and topics, handling the exchanged data, and managing the DSS operation.

Since this work utilizes MQTT protocol for the data exchange between different platform agents, it is worth noting the MQTT framework is prone to security issues [20]. Available security features worth mentioning include TLS/SSL encryption [21], credential-based authentication, IP-based whitelisting, and VPN for secure data exchange. However, using these features will bring a trade-off between system complexity and the MQTT lightweight advantages, which may impact the primary goal of ensuring a cohesive real-time T&D co-simulation with the DERs and other agents for HIL testing.

A. REAL-TIME T&D CO-SIMULATION AGENTS

The proposed RT T&D co-simulation framework is comprised by the following main agents:

- **RT Simulator:** operates as the master controller. Responsible for dictating the global timestamp and initializing the operation of each co-simulation timestep. Simulates the Tx system including the feeders P and Q outputs obtained from the DSS in RT.
- **DSS:** simulates the Dx system model considering the Tx system voltages and frequencies in RT. Includes the output from DER emulators. Solves the Dx power flow.
- **DER Emulator:** simulates output from DERs such as photovoltaic (PV), energy storage system (ESS), and water heater units considering Dx system voltages and Tx system frequencies. Includes typical GSFs from grid standards and can control DERs in RT based on control commands issued by the DERMS algorithm.
- **OpenDERMS:** An open-source, web-based platform capable of managing power schedules and issuing commands via HTTP communication protocol to DERs based on solutions provided by the DERMS algorithm under test.
- **A-DERMS:** Solves an optimization problem to achieve day-ahead power requests from grid operators while accounting for system constraints as well as day-ahead forecasts. Each A-DERMS is responsible for issuing power schedules to a group of L-DERMS under its management.
- **L-DERMS:** Solves an optimization problem in RT to find BTM DERs' setpoints based on schedules received from A-DERMS and availability of ESS units. Each L-DERMS is responsible for issuing power commands to a group of BTM DERs under its management. Typically, it will control devices geographically close to each other.
- **mqtt_rts:** Python handling function used to exchange data between the RTS and the MQTT broker via UDP and MQTT communication protocols, respectively.
- **mqtt_dss:** Python handling function used to exchange data via MQTT communication protocol between the DSS, the DER emulators, and the broker. Manages the DSS by issuing Dx system control commands, collecting operational data, and checking for convergence.

The co-simulation closed-loop step can only be completed once the RTS model has received the power flow (PF) results from the DSS solver, which depends on the power outputs from the DER emulators. Meanwhile, the DER emulators' power outputs depend both on the grid conditions and the power schedule issued as commands by the OpenDERMS platform. Therefore, the RT T&D co-simulation agents must be coordinated to exchange data and provide time-synchronization with the minimum amount of delay as possible. Furthermore, an initialization procedure must be established, and an iterative approach has to be developed to address the interdependence between the Dx system PF solution and the DER emulators' power outputs. Section II-B

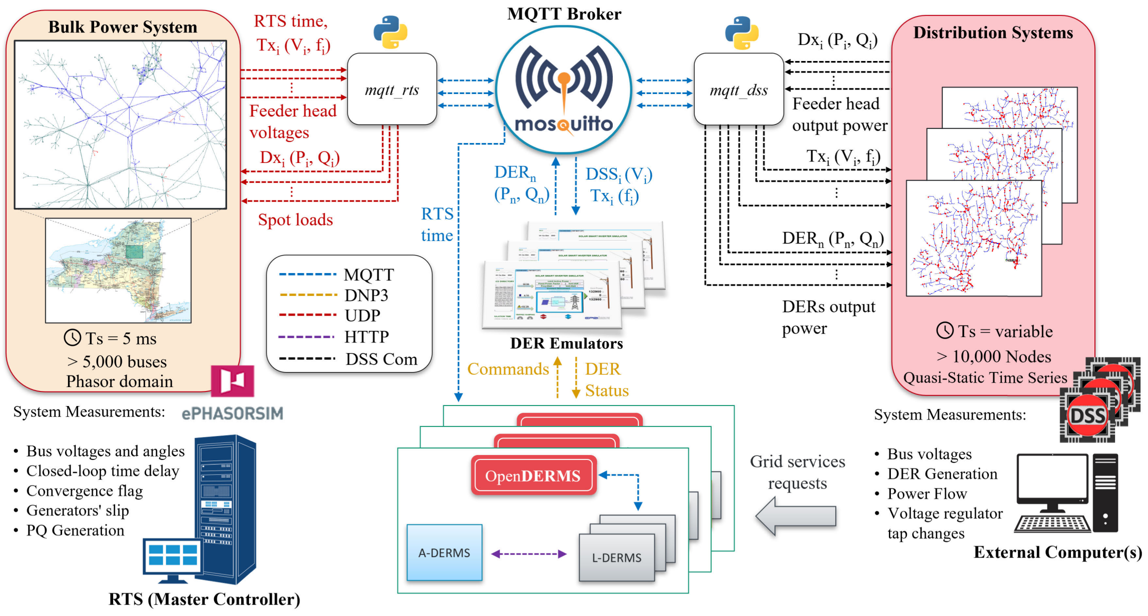


FIGURE 2. Main structure of RT T&D co-simulation framework proposed in this work. Includes communication via MQTT, UDP, HTTP, and DNP3 communication protocols. In this diagram, MQTT, DNP3, and HTTP utilize TCP communication.

discusses how the proposed framework addresses these challenges.

B. REAL-TIME DATA EXCHANGE AND TIME SYNCHRONIZATION

As previously explained, in the proposed framework, the RTS operates as the co-simulation master controller, and it is synchronized to a GPS clock provided by an SEL-2488 Satellite-Synchronized Network Clock. This means the co-simulation closed-loop timestep is initiated when information about the Tx voltage, frequency, and timestamp are streamed from the RTS. And the closed-loop timestep ends when the active and reactive power load consumptions from the Dx feeders (simulated in the DSS) are sent back as spot loads to the RTS. The time difference between the timestamp issued by the RTS and the timestamp returned in the last message received by the RTS including the Dx power outputs represents the total closed-loop delay. Consequently, every calculation and data exchange events that occur during the closed-loop delay must be finished before the beginning of the next closed-loop timestep to ensure no overruns occur in the testbed.

It is worth mentioning the MQTT explorer software tool [22] is utilized for monitoring the data exchange between agents. This opensource software works as a comprehensive MQTT client that provides an easy visualization of the messages exchanged in the MQTT broker.

1) ADAPTER FUNCTION MQTT_RTS.PY

As shown in Fig. 2, the closed-loop timestep starts when the RTS streams UDP data to the adapter function *mqtt_rts.py*, which then actuates by updating the RTS voltages, frequencies, and global timestamp to the MQTT broker. In the proposed testbed, one copy of this function runs in parallel with the simulation for each DSS instance (generally, one per feeder), and it has the following three main tasks. *First*, it provides a socket to exchange raw data between an external computer and the RTS via UDP in RT. *Second*, it publishes the data to a topic in the MQTT broker related to RTS data, which is distributed to any subscribed client (in this case, *mqtt_dss*). *Third*, it subscribes to a topic related to the power output of the feeders, obtained by the DSS PF solution. This means whenever a new PF solution is published by *mqtt_dss*, the function *mqtt_rts* (a client subscribed to those topics) will receive the data and hence stream it to the RTS via UDP. Thus, when the last feeder power output data is returned to the RTS in this manner, the RT co-simulation closed-loop is completed. Note that the *mqtt_rts* publish rate can be adjusted as needed, but should be small enough so that the closed-loop delay remains smaller than the closed-loop timestep at all times.

2) ADAPTER FUNCTION MQTT_DSS.PY

The second main adapter function, *mqtt_dss*, is built to handle the data exchange needed for solving the DSS PF including the DERs outputs. This function creates clients subscribed to the RTS-related topics. Thus, when the RTS data is published to the broker, *mqtt_dss* (a subscriber) is notified, hence utilizing the new data to represent the feeder head conditions to solve the PF and find the new Dx system operating points. Note the *mqtt_dss* processing time is the main bottleneck determining the co-simulation closed-loop delay. Undoubtedly, its processing time is directly affected by the size of the Dx system being simulated. Similar to *mqtt_rts*, one copy of this function must run in parallel with the model for each DSS instance simulated (normally, one per feeder). Its main tasks can be summarized as follows. *First*, it creates clients subscribed both to the RTS-related topics and to DER emulator-related topics (one per DER mRID). Figure 16, in the Appendix, demonstrates part of the code for the creation of a client and its subscription to DER emulator-related topics. *Second*, it initializes the DSS model, and forms the simulation data structure (in a Python dictionary format) that will be accessed when data is exchanged between the DSS and the DER emulators via MQTT communication protocol. *Third*, it is responsible for running PF iterations based on new RTS data received, as well as publishing new PF results back to the MQTT broker. Moreover, as will be discussed in the next section, *mqtt_dss* also performs an iterative PF that is used to find a final solution, in which both the Dx system voltages and the DERs outputs (affected by the grid conditions) have converged to a final solution. This is because the DERs outputs affect the grid, which in turn also affect their outputs due to their enabled GSFs.

To ensure *mqtt_dss* will not delay messages when publishing grid information to each DER, computing parallelization is implemented via the Python threading module [23], [24]. Basically, threading allows multiple tasks to be synchronously queued to be executed one after the other when the CPU becomes available. Note threads could also be assigned over multiple CPUs, but the Global Interpreter Lock will still prevent simultaneous multi-threading. Nevertheless, when designed to execute light tasks (such as I/O functions), the threading speed appears to work as if multi-tasking is occurring. Note threading is more efficient than multi-processing for this application due to the significantly smaller overhead required by each new thread when compared to a new process.

3) DER EMULATOR

Once *mqtt_dss* converges to a PF solution based on the initial Tx data received, it sends via MQTT communication protocol voltage and frequency values for each node to the corresponding DERs topics, and then awaits until active and reactive power and state of charge (SOC) for ESSs from each DER is received. Concurrently, DER emulators also only reply back with DER measurements once voltage and frequency data from each DER topic in its configuration is received. This ensures every DER output to be considered for each PF iteration performed in *mqtt_dss*. Note that for each global timestamp published to OpenDERMS, A-DERMs, and L-DERMS (RTS time in Fig. 2), the DER Emulators' operating time is also updated. This time synchronization is needed because the PVs available power is dictated by irradiance profiles loaded to the emulators during the testbed initialization. The routing of the current time to DER Emulators is published by *mqtt_dss*, as shown in Fig. 4.

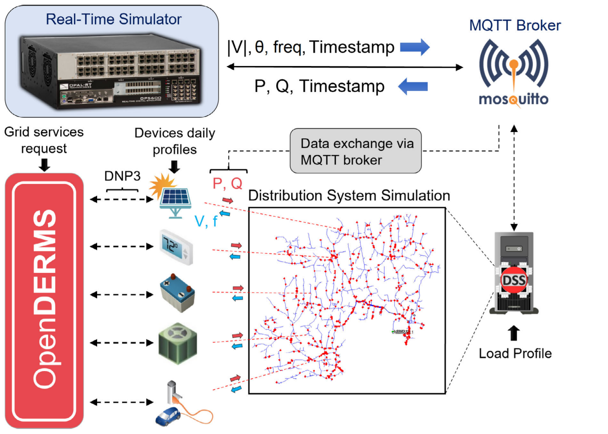


FIGURE 3. High level visualization of the data exchange between the RTS, DSS, and OpenDERMS via the MQTT broker. Modified from [25].

4) OPENDERMS

The closed-loop lock between *mqtt_dss* and DER emulators described above is not needed for the data exchange between DER emulators and OpenDERMS. That is because we can assume every DER will not necessarily receive and send back measurements to its OpenDERMS platform at the same time. Therefore, the time synchronization between the master controller and OpenDERMS is performed as follows. Every second, *mqtt_rts* publishes the RTS global timestamp to OpenDERMS (second-level is sufficient for DERMS operation), then for every new timestamp (RTS time in Fig. 3), if there is a command with a corresponding timestamp in its power schedule stored as an SQL database (sent by A-DERMS or L-DERMS), then it will send a message via DNP3 to its respective DER emulator. This requires the RTS timestamps to be rounded at second or millisecond level to ensure every entry in the OpenDERMS' SQL power schedule will be read.

C. CLOSED-LOOP TIME PROGRESSION

Figure 4 displays the co-simulation timestep sequence. At the start of the co-simulation step ($\Delta t1$), the RT simulator (master controller) streams UDP data to an external computer running both *mqtt_rts* and *mqtt_dss* Python functions (1). Next, *mqtt_rts* receives the data and publishes it to the MQTT broker corresponding topic (2). Once the topic is updated, its subscriber *mqtt_dss* receives the data, and utilizes it to execute a PF in DSS (3). If the DER emulator is disabled, the feeder P and Q consumptions obtained from the PF solution are updated to the MQTT broker corresponding topic (4a), which is then received by *mqtt_rts* and sent as UDP data back to the RT target (5). On the other hand, if the DER emulator is enabled, it receives the PF solution via MQTT (4b), and calculates the DERs P and Q injections based on the grid conditions, what GSFs have been enabled, and the power setpoints issued by OpenDERMS (6). Then, the power injections are updated to the MQTT broker (7), and received by *mqtt_dss* for another PF iteration to test for convergence

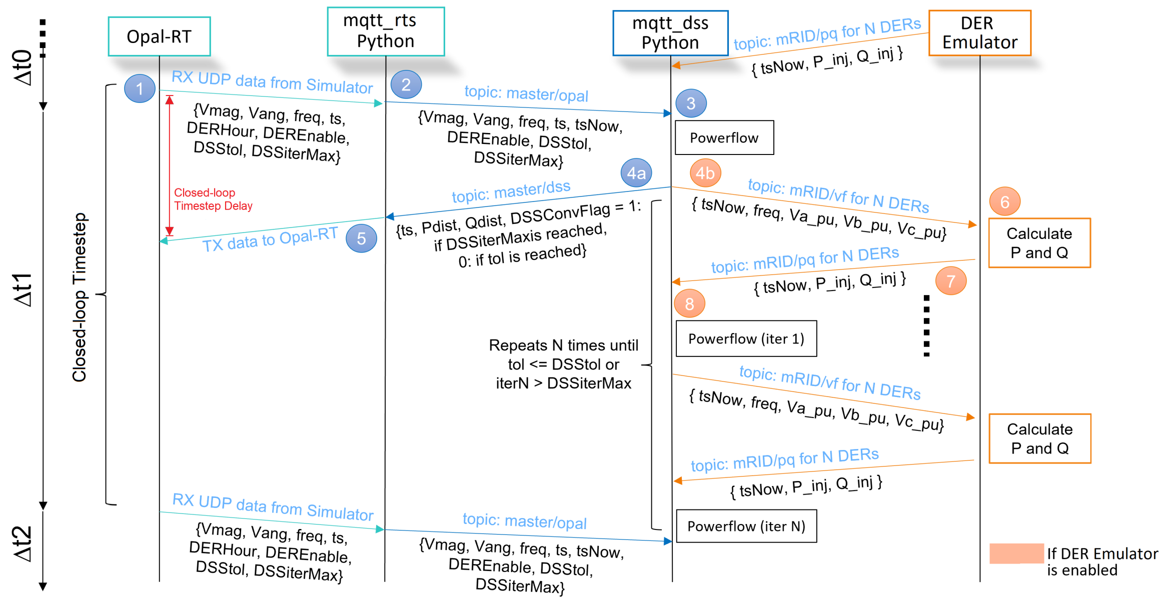

Opal-RT
mqtt_rts Python
mqtt_dss Python
DER Emulator
Δt0
Δt1
Δt2
Closed-loop Timestep
RX UDP data from Simulator
{Vmag, Vang, freq, ts, DERHour, DEREnable, DSStol, DSSiterMax}
topic: master/opal
{Vmag, Vang, freq, ts, tsNow, DEREnable, DSStol, DSSiterMax}
Powerflow
topic: mRID/pq for N DERs
{ tsNow, P_inj, Q_inj }
Closed-loop Timestep Delay
topic: master/dss
{ts, Pdist, Qdist, DSSConvFlag = 1: if DSSiterMaxis reached, 0: if tol is reached}
TX data to Opal-RT
topic: mRID/vf for N DERs
{ tsNow, freq, Va_pu, Vb_pu, Vc_pu}
Calculate P and Q
Powerflow (iter 1)
Repeats N times until tol <= DSStol or iterN > DSSiterMax
Powerflow (iter N)
If DER Emulator is enabled

FIGURE 4. **Co-simulation closed-loop time progression during RT operation. Steps (1) and (5) correspond to UDP data exchange, whereas other steps are carried out via MQTT communication protocol. When the simulation loop includes DER emulators, step 4a occurs only after step 8; otherwise, step 4a occurs after step 3.**

(8). Steps (4b), (6), (7), and (8) repeat for N iterations until either (i) the PF convergence tolerance is achieved, or (ii) a predetermined maximum number of iterations occur ($iterN > DSSiterMax$). In this case, an overrun flag is triggered). The PF convergence is evaluated by checking if the changes from the last PF iteration are below a given tolerance ($tol \leq DSStol$). Therefore, when the DER emulator is enabled, steps (4b) and (5) are only executed after (i) or (ii). Note if the process between the start of step (1) and the end of step (5) takes longer than the co-simulation timestep (Δt1), a closed-loop co-simulation overrun is flagged. For proper operation, the system should be designed to never operate with overruns.

Moreover, when BTM DERs are operating with voltage-support GSFs, there can be oscillations during DSS PF iterations that occur between steps 4b to 8 due to the quasi-static characteristic of the DSS implemented. For instance, suppose that at a given node with a BTM DER initially with zero reactive power output, a PF yields a voltage of 0.9 p.u. at step 4b, causing the BTM DER at that node to provide a reactive power support of 10 kVAr at step 6, and suppose that this reactive power support causes the next PF solution to yield 1 p.u. at that node. This would cause a continuous oscillation in the DER emulator output and the DSS PF solution.

To address this issue a smoothing factor is applied to the output power measurements received from the DER emulators (in *mqtt_dss*) before each PF iteration that occurs between steps 4b to 8. The following equation represents the smoothing factor implemented.

$$Q_{\text{smth}}[i] = \alpha \times Q_{\text{original}}[i] + (1 - \alpha) \times Q_{\text{smth}}[i - 1] \quad (1)$$

With a smoothing factor $\alpha = 0.5$, for example, and assuming a linear reactive power support curve, the previous case would progress as follows: PF solution yields $V[1] = 0.9$ p.u., DER emulator finds a reactive power support $Q_{original}[1] = 10$ kVAr, so a smoothed value used in the next PF becomes $Q_{smth}[1] = 5$ kVAr, assuming linearity, the next PF solution yields a nodal voltage $\text{V}[2] = 0.95$ p.u., which corresponds to a reactive power support of $Q_{smth}[2] = 5$ kVAr, and thus the system would reach convergence.

Note the PF DSS tolerance (DSStol), the maximum number of PF iterations (DSSiterMax), and the smoothing factor (α) are obtained heuristically, and will depend on system size and computational capability. Here we set a tolerance of 0.005 p.u., with a maximum of 5 iterations, and a smoothing factor of 0.3.

D. AUTOMATED PROCESSES TO SETUP CO-SIMULATION MODELS

The RT T&D co-simulation setup including hundreds to thousands of devices can become too laborious due to the need to write configuration files for each device across multiple platforms. For instance, each new DER added to the simulation requires (i) a new entry in the DER emulator configuration files, including DER ratings, GSFs parameters, and initialization values; (ii) a new DER instance defined in the DSS model files, including the DER location and monitors (note every DER is represented in the DSS as a PV power injection, independent of the DER type, since the emulator will already account for the DER operational characteristics based on their type); (iii) a new DER entry in the OpenDERMS platform, including the DER ratings, mRID (master resource identifier), and MQTT topics which

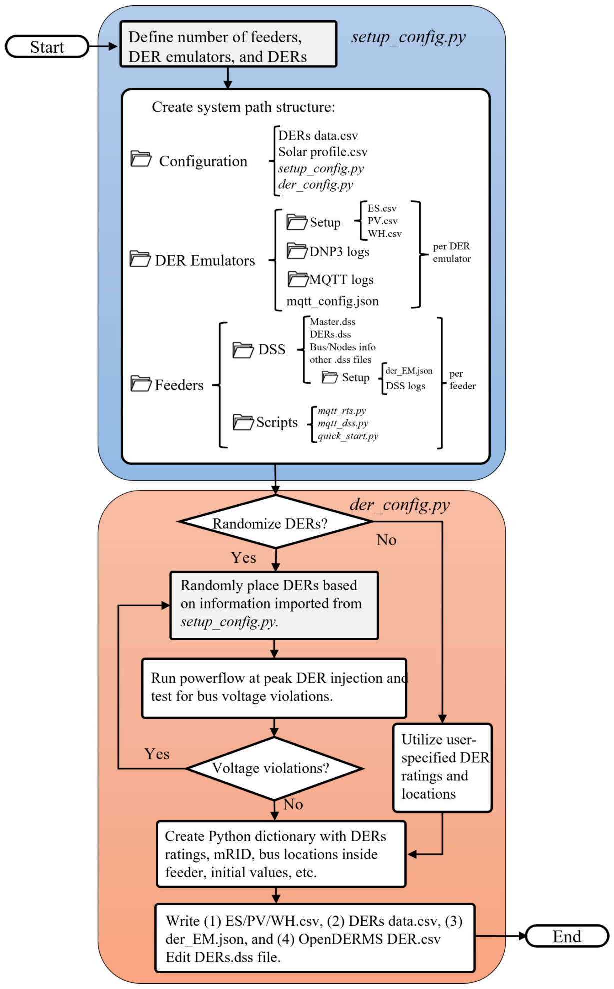

Start
Define number of feeders, DER emulators, and DERs
setup_config.py
Create system path structure:
Configuration
DERs data.csv
Solar profile.csv
setup_config.py
der_config.py
DER Emulators
Setup
ES.csv
PV.csv
WH.csv
DNP3 logs
MQTT logs
mqtt_config.json
per DER emulator
Feeders
DSS
Master.dss
DERs.dss
Bus/Nodes info
other .dss files
Setup
der_EM.json
DSS logs
per feeder
Scripts
mqtt_rts.py
mqtt_dss.py
quick_start.py
der_config.py
Randomize DERs?
Yes
No
Randomly place DERs based on information imported from setup_config.py.
Run powerflow at peak DER injection and test for bus voltage violations.
Yes
Voltage violations?
No
Utilize user-specified DER ratings and locations
Create Python dictionary with DERs ratings, mRID, bus locations inside feeder, initial values, etc.
Write (1) ES/PV/WH.csv, (2) DERs data.csv, (3) der_EM.json, and (4) OpenDERMS DER.csv Edit DERs.dss file.
End

FIGURE 5. Flowchart of the initialization functions required to prepare the testbed configuration files. The system is setup utilizing initialization functions *setup_config.py* and *der_config.py*, which must be executed whenever a new feeder configuration is added to Tx systems.

the platform will use to issue commands; (iv) new topics that *mqtt_dss* must subscribe to, corresponding to the DER active and reactive power outputs, (v) new topics that *mqtt_dss* must publish to, corresponding to the grid voltages and frequency at the bus where the DER is added to, (vi) new contract files that must be added to A-DERMS configuration files, and (vii) a new DER instance in configuration files of the respective L-DERMS responsible for managing the new device.

Furthermore, this editing process must be repeated for each DER based on the number of Dx feeders to be simulated. However, a mistake in any of these steps can result in testbed initialization failure and/or simulation collapse. Thus, significant work has been invested in automation processes that reduce human error during setup. Python functions have been developed to minimize the configuration based on the number of feeders and DERs to be simulated. Figure 5 displays the sequence of steps performed by two main setup functions designed to automate the setup procedure: *setup_config.py* and *der_config.py*. This logic must be done whenever a new testbed scenario or a new feeder is integrated into the testbed.

It is worth mentioning that since the preparation to initialize each simulation also requires significant effort (e.g., initialization of DER emulators and each OpenDERMS via a graphical user interface, start of A-DERMS/L-DERMS instances, etc), Python *subprocess* and *PyAutoGUI* modules [26] were utilized to automate the startup with script *quick_start.py*. This Python function utilizes screen image detection algorithms to automatically navigate through GUIs reducing human effort.

III. HIL SIMULATION RESULTS

A. TESTBED SETUP

In [11], we introduced the coupling between Tx and Dx and studied the propagation of dynamics between the domains. In [25], we expanded the testbed to hundreds of DERs and demonstrated how DERs could be leveraged to provide local frequency support to the Tx. The work also included a scenario regarding frequency and load balance between the Tx and Dx under a severe bulk power system (BPS) contingency. Here, we expand the developed HIL testbed much further by including real-time optimization of DERs dispatch via A-DERMS and L-DERMS to provide day-ahead services to the BPS. The HIL testbed consists of (a) TS positive sequence model of a section of the EITS; (b) Quasi-Static Time-Series models of two 9,500 nodes feeders from local utilities; (c) 182 BTM DERs distributed over 10 DER emulators, corresponding to 48 ESSs, and 134 PV systems, with ratings ranging from tens to about a hundred kW, and (d) the hierarchical DERMS algorithm from [4], corresponding to one A-DERMS, 10 L-DERMS, and 11 OpenDERMS instances. Details regarding the data exchange structure for this use case are presented in Fig. 2. Figure 6 shows how the BTM DERs are distributed across each L-DERMS. Despite having a similar number of ESSs, note the ratio between total energy storage and rated output power from each L-DERMS is designed unevenly to represent a more realistic scenario. For instance, L-DERMS 5 has a large storage compared to its rated power, whereas L-DERMS 8 has more output rated power than energy storage. It is worth highlighting that the Tx model is executed in a positive sequence software tool, hence imbalances are not considered at the Tx level. On the other hand, the DSS includes per phase modeling, and hence can represent the impact of imbalances in BTM DER injections and loads on the Dx operation.

B. DAY AHEAD SERVICE PERFORMANCE

The main novelty of the proposed testbed is the ability to test DERMS in large-scale HIL simulations. To demonstrate such capability, we have developed a use case in which the DERMS algorithm from [4] is utilized to achieve day-ahead power requests sent by a system operator. The simulation is executed with data collected on July 2nd, 2016, and it runs for 12 hours, starting at 10:00 AM. The DERMS optimization

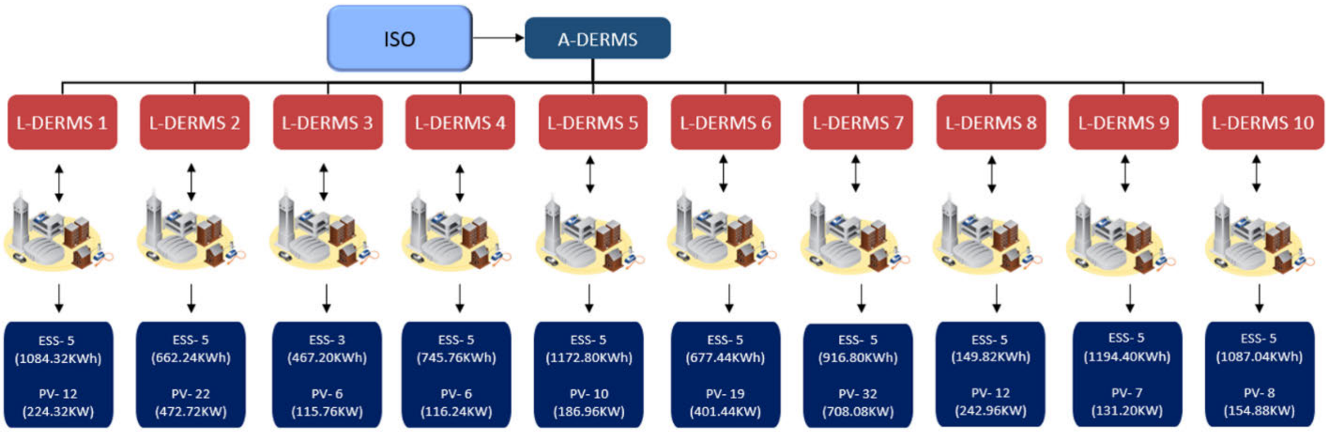

ISO
A-DERMS
L-DERMS 1
L-DERMS 2
L-DERMS 3
L-DERMS 4
L-DERMS 5
L-DERMS 6
L-DERMS 7
L-DERMS 8
L-DERMS 9
L-DERMS 10
ESS- 5 (1084.32KWh) PV- 12 (224.32KW)
ESS- 5 (662.24KWh) PV- 22 (472.72KW)
ESS- 3 (467.20KWh) PV- 6 (115.76KW)
ESS- 5 (745.76KWh) PV- 6 (116.24KW)
ESS- 5 (1172.80KWh) PV- 10 (186.96KW)
ESS- 5 (677.44KWh) PV- 19 (401.44KW)
ESS- 5 (916.80KWh) PV- 32 (708.08KW)
ESS- 5 (149.82KWh) PV- 12 (242.96KW)
ESS- 5 (1194.40KWh) PV- 7 (131.20KW)
ESS- 5 (1087.04KWh) PV- 8 (154.88KW)

FIGURE 6. A-DERMS/L-DERMS and DERs distribution for day ahead grid service scenario under test.

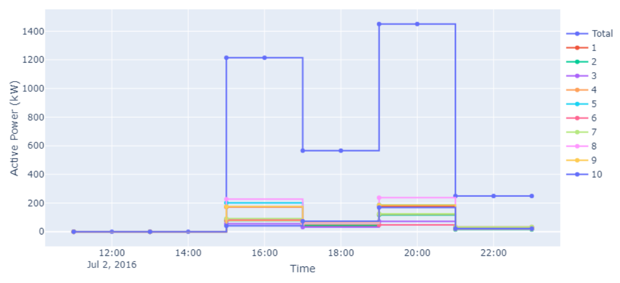


FIGURE 7. Power schedule for each L-DERMS as well as total power schedule request generated by A-DERMS.

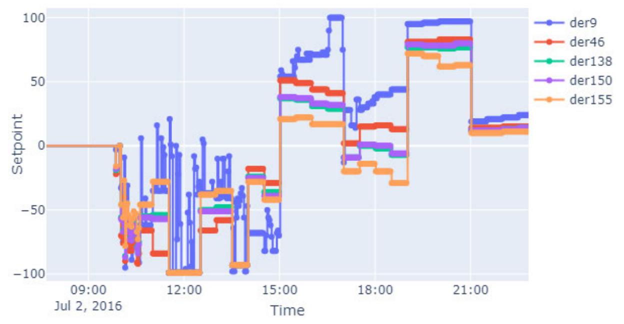


FIGURE 8. Power injection of each BTM ES-type DER in the first L-DERMS.

takes into account contract configuration files, which is established for each BTM DER to represent their availability and willingness to provide services throughout the day.

At the beginning of the simulation, the A-DERMS algorithm reads the contract configuration files from each BTM DER to solve a day-ahead optimization and generate power output requests to be sent to each L-DERMS. This process occurs only once, and after the power request profiles are built, they are sent via HTTP communication protocol (see Fig. 2) to each OpenDERMS platform. Figure 7 displays the power schedule for each L-DERMS prepared by the A-DERMS at the beginning of the simulation.

The L-DERMS algorithm is designed to have a timestep of 30 minutes for all ES devices except for selected pilot units, which have a timestep of 2 minutes. This allows the pilot ES devices to act as fast controllers to handle inaccuracies found in RT operation when compared to the original day-ahead schedule. During RT operation, L-DERMS algorithms issue power commands to their respective OpenDERMS platform to store a power reference schedule for each BTM DER under their jurisdiction.

After the power schedule is updated to OpenDERMS, once the corresponding RT timestamp is published by the RT simulator, each OpenDERMS will send power commands to the BTM DERs via DNP3. This behavior can be observed in Fig. 8, in which DER 9 is the pilot unit.

Every L-DERMS will be operating to achieve its power schedule commanded by their A-DERMS. Therefore, the sum of all L-DERMS injections (similar to the one shown in Fig. 8) has to ultimately match the A-DERMS request from Fig. 7. The total day-ahead requested power and the overall sum of L-DERMS power injections are presented in Fig. 9. Note there are no penalties applied to injecting power above the requested. Hence the PVs might operate at maximum power injection while the ESS were set to charging mode between 11:00 AM to 3:00 PM. Moreover, note the irradiance profiles utilized have a 1-min resolution. Furthermore, every ES is initialized with 70% SOC and is set to charge by the DERMS while the system operator power request from Fig. 9 is null. This can be observed in Fig. 10.

One of the main challenges of operating a Dx feeder under high DER penetration is maintaining acceptable voltage profiles and avoiding over-usage of reactive power support devices. Figure 11 displays the voltage bus profiles measured during the RT simulation. Since the moment of the highest power request occurs at night when there are no PVs, the ES units are solely responsible for achieving the request, causing a large power injection at their point of connection. The abrupt increase in power injection at their connection nodes leads to large voltage violations observed from 7:00 to

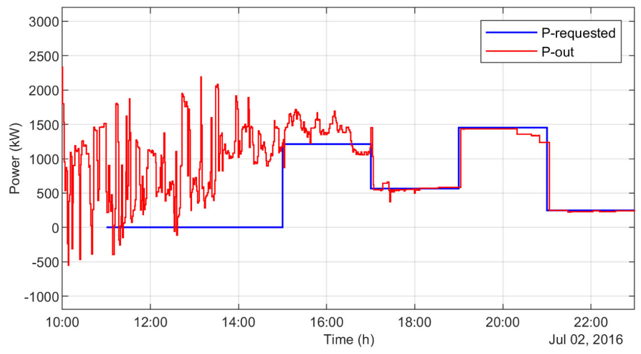


FIGURE 9. Total BTM DERs' power injection.

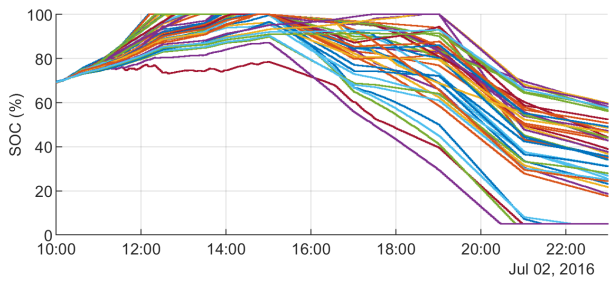


FIGURE 10. ES units SOC throughout the day-long simulation. Note standby battery losses are also included in the DER emulator.

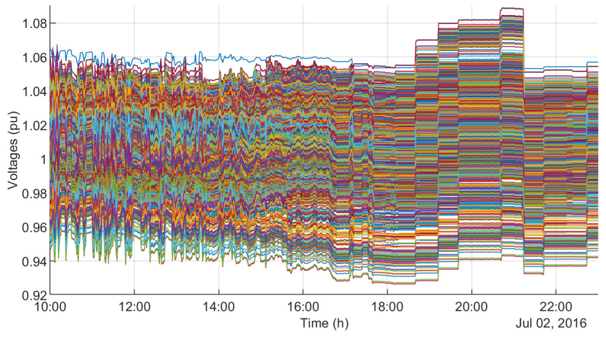


FIGURE 11. Voltage profiles from 3000 randomly selected nodes from the 9,500 nodes within the feeder.

TABLE 1. Settings implemented for the voltage-reactive power characteristic curve from Fig. 12.

Curve	(V_1, Q_1)	(V_2, Q_2)	(V_3, Q_3)	(V_4, Q_4)
Standard	0.92, 44%	0.98, 0%	1.02, 0%	1.08, -44%
Custom	0.94, 44%	0.98, 0%	1.02, 0%	1.06, -44%

9:00 PM. This highlights performance issues expected from a feeder with BTM DERs not equipped with local volt-var control, which is a required functionality defined in standard IEEE 1547-2018.

Therefore, to address this issue, we run another day-long simulation in which the DER emulators have BTM DERs equipped with voltage-reactive power droop local control, based on the standard voltage-reactive power GSF from Std. IEEE 1547-2018 for a category II-B DER. Figure 13 displays the new voltage profiles when the BTM DERs provide local voltage support with standard droop settings. In addition, we also carried out a scenario in which the BTM DERs are equipped with a stronger droop curve. Table 1 and Figure 12 display the droop curve characteristics for each case. With more aggressive voltage-reactive power support characteristics, the feeder bus voltages achieved an even better performance, as shown in Fig. 14.

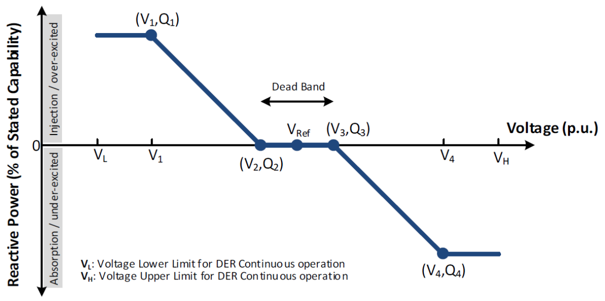

Reactive Power (% of Stated Capability)
Injection / over-excited
Absorption / under-excited
(V₁,Q₁)
Dead Band
V_Ref
(V₃,Q₃)
Voltage (p.u.)
0
V_L
V₁
(V₂,Q₂)
V₄
V_H
(V₄,Q₄)
V_L: Voltage Lower Limit for DER Continuous operation
V_H: Voltage Upper Limit for DER Continuous operation

FIGURE 12. Voltage-reactive power characteristic curve. From [27].

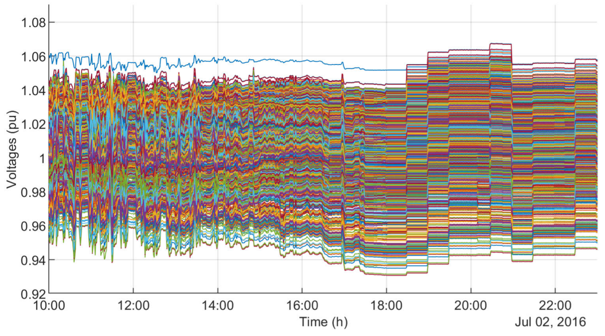


FIGURE 13. Voltage profiles when BTM DERs are equipped with local volt-var control with standard settings.

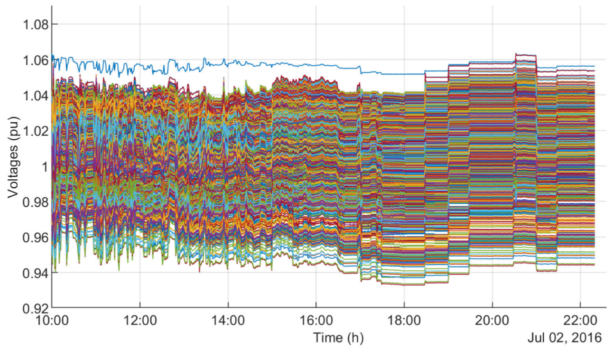
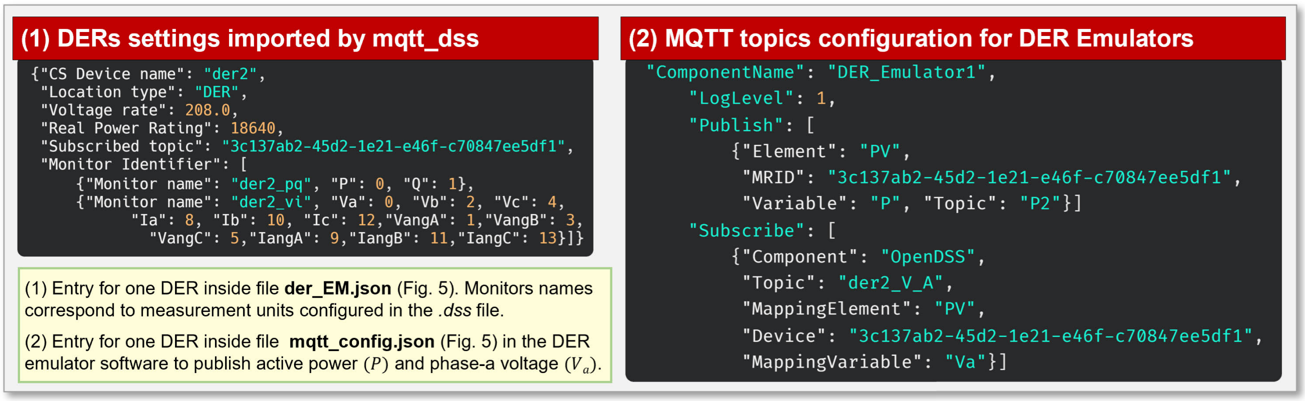


FIGURE 14. Voltage profiles when BTM DERs are equipped with local volt-var control with custom settings (increased support).

Even though voltage violations are still observed in Fig. 13, it is worth mentioning that several studies would be carried out to properly allocate the BTM DERs through the distribution feeder. Since that was not the goal of this work, their locations were randomly based on the scheme from Fig. 5. The important finding here is the demonstration of how



(1) DERs settings imported by mqtt_dss

```
{"CS Device name": "der2",
 "Location type": "DER",
 "Voltage rate": 208.0,
 "Real Power Rating": 18640,
 "Subscribed topic": "3c137ab2-45d2-1e21-e46f-c70847ee5df1",
 "Monitor Identifier": [
     {"Monitor name": "der2_pq", "P": 0, "Q": 1},
     {"Monitor name": "der2_vi", "Va": 0, "Vb": 2, "Vc": 4,
            "Ia": 8, "Ib": 10, "Ic": 12,"VangA": 1,"VangB": 3,
              "VangC": 5,"IangA": 9,"IangB": 11,"IangC": 13}]}
```

(1) Entry for one DER inside file **der_EM.json** (Fig. 5). Monitors names correspond to measurement units configured in the *.dss* file.

(2) Entry for one DER inside file **mqtt_config.json** (Fig. 5) in the DER emulator software to publish active power (P) and phase-a voltage (V_a).

(2) MQTT topics configuration for DER Emulators

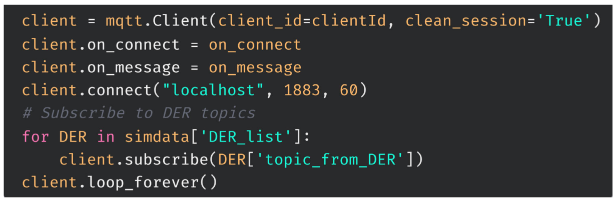
```
"ComponentName": "DER_Emulator1",
    "LogLevel": 1,
    "Publish": [
        {"Element": "PV",
         "MRID": "3c137ab2-45d2-1e21-e46f-c70847ee5df1",
         "Variable": "P", "Topic": "P2"}]
    "Subscribe": [
        {"Component": "OpenDSS",
         "Topic": "der2_V_A",
         "MappingElement": "PV",
         "Device": "3c137ab2-45d2-1e21-e46f-c70847ee5df1",
         "MappingVariable": "Va"}]
```

FIGURE 15. Simulation settings built for each DER as a dictionary for *mqtt_dss*.

the testbed can test a DERMS algorithm in a HIL environment while including propagation of dynamics between Tx and Dx, GSFs from DERs, and the communication links between DERs and DERMS.

IV. CONCLUSION

Traditionally, Tx and Dx simulation studies have been conducted independently using inhomogeneous simulation tools. These traditional tools and practices are not sufficient for studying grid services of DERs which involves potential dynamic interaction between Tx and Dx systems. To research these services, their values and potential system impacts, this paper has developed a first-of-its-kind T&D co-simulation platform. The developed platform allows simulation of large-scale T&D systems which is essential for realistic representation of system impacts. The platform is further capable of both offline and online simulations, providing HIL testing capability for experimental validation of DERs and aggregator controls. These features have been demonstrated in a case study of DER grid services in the NY state grid. In the presented case study, the developed T&D platform runs a 5,560-buses model of the NY state Tx and a 9,877-nodes radial feeder model of a NY state distribution utility including DERs. Moreover, the HIL testing capability of the platform has been demonstrated by testing a proposed DERMS control architecture. The test illustrates the effectiveness and limitations of BTM DER participation in day-ahead energy market. Therefore, the framework for a RT T&D co-simulation testbed proposed here enables testing of DERMS algorithms in large-scale HIL simulations. The cross-platform data exchange and time synchronization is performed via MQTT communication protocol and multi-threading parallelization. Methods are developed to automate the setup of new simulation cases, which can significantly help reduce the burden of developing and/or editing large-scale systems. The proposed method is tested in a realistic large-scale model including a section of the EITS co-simulated with distribution feeders from local utilities and multiple BTM DERs. We demonstrate the proposed framework performance by simulating a day-ahead power dispatch service while including BTM DERs with GSFs to provide voltage support while operating with optimal dispatch.



```
client = mqtt.Client(client_id=clientId, clean_session='True')
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost", 1883, 60)
# Subscribe to DER topics
for DER in simdata['DER_list']:
    client.subscribe(DER['topic_from_DER'])
client.loop_forever()
```

FIGURE 16. *mqtt_dss* function connection via MQTT communication protocol to receive messages published by the DER emulators.

APPENDIX A

Figure 15 demonstrates the initialization process for the *mqtt_dss* function to create a client subscribed to DER topics to receive operational data from the DER emulators. Note the MQTT client is assigned with *on_connect* and *on_message* functions, which are executed when the client receives a CONNACK message from the server, and when a new MQTT message is received by the client, respectively. Moreover, when a client publishes to a topic that does not yet exist, that topic is created.

Figure 16 demonstrates how to create clients to be subscribed to MQTT topics published by the DER emulators, whereas Fig. 17 displays the code for publishing grid data to the DER emulators via MQTT communication protocol. Both are contained within the Python function *mqtt_dss*. Notice the data is converted into a JSON (JavaScript Object Notation) string prior to being published. The function from Fig. 17 is responsible for publishing data to the DER emulators. It can be parallelized, allowing data to be published to multiple devices with minimum time delay. This is accomplished with the code displayed in Fig. 18, utilizing the multi-threading Python module.

APPENDIX B

In this work, the Tx model is based on a 65,000+ buses transient-stability type model of the EITS provided to the

```
def mqtt_data_pub(mqtt_client, der_group, seq, mqtt_payload):
    for deridx in range(seq[0], seq[1]):
        to_dersim = dumps({
            'tsNow': mqtt_payload['from_rts']['tsNow'],
            'freq': mqtt_payload['from_rts']['freq'],
            'Va_pu': mqtt_payload['DER_list'][deridx]['Va_pu'],
            'Vb_pu': mqtt_payload['DER_list'][deridx]['Vb_pu'],
            'Vc_pu': mqtt_payload['DER_list'][deridx]['Vc_pu']
        })
        mqtt_payload['DER_list'][deridx]['update received'] = False
        mqtt_client.publish(topic=mqtt_payload['DER_list'][deridx]
['topic_to_DER'], payload=to_dersim)
```

FIGURE 17. **How *mqtt_dss* function publishes a message via MQTT communication protocol to DER emulators (subscribers).**

```
def client_threaded_publish(mqtt_client, mqtt_payload):
    th = []
    start = time()
    for idx, der_addr_tuple in enumerate(der_group_list):
        th.append(threading.Thread(target=mqtt_data_pub, args=
            (mqtt_client, idx, der_addr, mqtt_payload)))
    for idx in th:
        if not idx.is_alive():
            idx.start()
```

FIGURE 18. **Code for utilizing the threading Python module for task parallelization.**

New York Power Authority by the New York Independent System Operator. The Tx model in use has been reduced to a 5,560 buses system and configured for compatibility with a RTS. Moreover, for the model reduction, the NY state is kept in higher resolution, whereas buses outside the state have their resolution reduced, such that generators located in areas far from the NYISO purview are aggregated mainly based on geographical location. For instance, all units in Florida State (considerably distant from the NYS) are aggregated into one equivalent machine.

On the other hand, the Dx is based on a 9,877 nodes radial feeder model from a local utility in the NYS. The model consists of over 3,500 loads, 2,000 transformer units (including voltage regulators), and shunt reactors and capacitors. Without any DER injection, the total load at the feeder head at a given snapshot is about 7 MW. The feeder is connected to the Tx via a 69/13.8 kV Yg/Yg transformer. In addition, the Tx is represented as a voltage source behind the substation transformer in the DSS, whereas the Dx is represented in the Tx as a lumped load with total power consumption given by the DSS solution. All DERs are represented as PV injectors in the DSS with power injection values given by the DER emulators.

REFERENCES

[1] C. Cano, *A New Day for Distributed Energy Resources*, document FERC 2222, 2020.

[2] H. Sun, Q. Guo, J. Qi, V. Ajjarapu, R. Bravo, J. Chow, Z. Li, R. Moghe, E. Nasr-Azadani, U. Tamrakar, G. N. Taranto, R. Tonkoski, G. Valverde, Q. Wu, and G. Yang, "Review of challenges and research opportunities for voltage control in smart grids," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 2790–2801, Jul. 2019.

[3] N. Singhal, M. Heidarifar, T. Hubert, E. Ela, A. Huque, T. Abate, and P. Shoop, "Grid services in the distribution and bulk power systems," Electr. Power Res. Inst., Tech. Rep., 2021, no. 3002022405, pp. 1–119. [Online]. Available: https://www.epri.com/research/products/000000003002022405

[4] A Garg, T Hubert, Aminul Huque, and A. Renjit, "Distributed energy resource management system (DERMS) control architecture for grid services in the distribution and bulk power systems," Electr. Power Res. Inst., Tech. Rep., 2021, no. 3002022480, pp. 1–70. [Online]. Available: https://www.epri.com/research/programs/067418/results/3002022480

[5] S. M. Mohseni-Bonab, A. Hajebrahimi, I. Kamwa, and A. Moeini, "Transmission and distribution co-simulation: A review and propositions," *IET Gener., Transmiss. Distrib.*, vol. 14, no. 21, pp. 4631–4642, Nov. 2020.

[6] R. C. Dugan and T. E. McDermott, "An open source platform for collaborating on smart grid research," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Jul. 2011, pp. 1–7.

[7] P. T. Mana, K. P. Schneider, W. Du, M. Mukherjee, T. Hardy, and F. K. Tuffner, "Study of microgrid resilience through co-simulation of power system dynamics and communication systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 1905–1915, Mar. 2021.

[8] T. Godfrey, S. Mullen, D. W. Griffith, N. Golmie, R. C. Dugan, and C. Rodine, "Modeling smart grid applications with co-simulation," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, Oct. 2010, pp. 291–296.

[9] J. Matevosyan, J. MacDowell, N. Miller, B. Badrzadeh, D. Ramasubramanian, A. Isaacs, R. Quint, E. Quitmann, R. Pfeiffer, H. Urdal, T. Prevost, V. Vittal, D. Woodford, S. H. Huang, and J. O'Sullivan, "A future with inverter-based resources: Finding strength from traditional weakness," *IEEE Power Energy Mag.*, vol. 19, no. 6, pp. 18–28, Nov. 2021.

[10] H. Jain, B. A. Bhatti, T. Wu, B. Mather, and R. Broadwater, "Integrated transmission-and-distribution system modeling of power systems: State-of-the-art and future research directions," *Energies*, vol. 14, no. 1, p. 12, Dec. 2020.

[11] V. Paduani, R. Kadavil, H. Hooshyar, A. Haddadi, A. H. M Jakaria, and A. Huque, "Real-time T&D co-simulation for testing grid impact of high DER participation," in *Proc. IEEE PES Grid Edge Technol. Conf. Expo. (Grid Edge)*, 2023, pp. 1–5, doi: 10.1109/GridEdge54130.2023.10102714.

[12] V. Paduani, B. Xu, D. Lubkeman, and N. Lu, "Novel real-time EMT-TS modeling architecture for feeder blackstart simulations," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Jul. 2022, pp. 1–5.

[13] A. Shirsat, V. Muthukaruppan, R. Hu, V. D. Paduani, B. Xu, L. Song, Y. Li, N. Lu, M. Baran, D. Lubkeman, and W. Tang, "A secure and adaptive hierarchical multi-timescale framework for resilient load restoration using a community microgrid," *IEEE Trans. Sustain. Energy*, vol. 14, no. 2, pp. 1057–1075, Apr. 2023.

[14] R. Hu, A. Shirsat, V. Muthukaruppan, S. Zhang, Y. Li, L. Song, B. Xu, V. Paduani, N. Lu, M. Baran, and W. Tang, "A novel feeder-level microgrid unit commitment algorithm considering cold-load pickup, phase balancing, and reconfiguration," 2023, *arXiv:2301.08350*.

[15] V. Muthukaruppan, A. Shirsat, R. Hu, V. Paduani, B. Xu, Y. Li, M. Baran, N. Lu, D. Lubkeman, and W. Tang, "Feeder microgrid management on an active distribution system during a severe outage," 2022, *arXiv:2208.10712*.

[16] J. Belanger, P. Venne, and J. N. Paquin, "The what, where and why of real-time simulation," *Planet Rt*, vol. 1, no. 1, pp. 25–29, 2010.

[17] S. Poudel, S. J. Keene, R. L. Kini, S. Hanif, R. B. Bass, and J. T. Kolln, "Modeling environment for testing a distributed energy resource management system (DERMS) using GridAPPS-D platform," *IEEE Access*, vol. 10, pp. 77383–77395, 2022.

[18] (2023). *MQTT*. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3

[19] T. G. Mattson, T. A. Anderson, and G. Georgakoudis, "PyOMP: Multithreaded parallel programming in Python," *Comput. Sci. Eng.*, vol. 23, no. 6, pp. 77–80, Nov. 2021.

[20] F. Chen, Y. Huo, J. Zhu, and D. Fan, "A review on the study on MQTT security challenge," in *Proc. IEEE Int. Conf. Smart Cloud*, Nov. 2020, pp. 128–133.

[21] R. Oppliger, *SSL and TLS: Theory and Practice*. Norwood, MA, USA: Artech House, 2023.

[22] T. Nordquist. (2023). *MQTT Explorer*. [Online]. Available: http://mqtt-explorer.com

[23] J. Palach, *Parallel Programming With Python*. Birmingham, U.K.: Packt Publishing Ltd, 2014.

[24] R. Eggen and M. Eggen, "Thread and process efficiency in Python," in *Proc. Int. Conf. Parallel Distrib. Process. Techn. Appl.*, 2019, pp. 32–36.

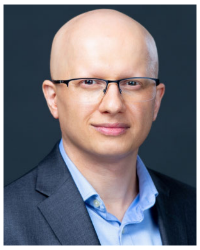
[25] H. Hooshyar, R. Kadavil, V. Paduani, A. Haddadi, A. Jakaria, A. Huque, and G. Stefopoulos, "Grid services by behind-the-meter distributed energy resources: NY state grid case study," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, 2023, pp. 1–5, doi: 10.1109/PESGM52003.2023.10253349.

[26] A. Sweigart. (2020). *PyAutoGUI Documentation*. Accessed: Sep. 30, 2024. [Online]. Available: https://readthedocs.org/projects/pyautogui/downloads/pdf/latest/

[27] *Standard for Interconnection and Interoperability of Distributed Energy Resources With Associated Electric Power Systems Interfaces*, IEEE Standard 1547-2018, 2018.

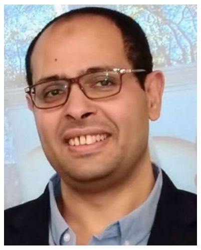
VICTOR DALDEGAN PADUANI (Member, IEEE) received the B.S. degree in electrical engineering from the Federal University of Santa Catarina, Florianopolis, Brazil, in 2017, the M.S. degree in electrical engineering from Villanova University, Villanova, PA, USA, in 2019, and the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 2022. He is currently a Senior Power Systems Engineer with the Advanced Grid Innovation Laboratory for Energy (AGILe), New York Power Authority. His research interests include power system modeling and control, the integration of renewable energy resources, and real-time co-simulation of power systems.

RAHUL KADAVIL (Senior Member, IEEE) received the B.E. degree in electrical engineering from the Fr. C. Rodrigues Institute of Technology, University of Mumbai, India, in 2009, and the M.S. degree in electrical engineering from Colorado State University, Fort Collins, CO, USA, in 2017. He was engaged as a Research Engineer with the Power and Energy Real-Time Laboratory (PERL), Idaho National Laboratory, for modeling and integration testing of hybrid energy storage systems (HESS) using real-time simulation tools. He is currently a Manager with the Advanced Grid Innovation Laboratory for Energy (AGILe), New York Power Authority. He designs and tests hardware in the loop experiments to analyze diverse smart technologies and understand how they complement each other. He is the second inventor of energy storage ramping optimization for aggregating ramping capabilities from multiple HESS. His research interests include power systems digital simulation, digital twins, protection, cyber-physical testbeds, and integration of renewable energy resources.



HOSSEIN HOOSHYAR (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 2012. He has been the Director of the Advanced Grid Innovation Laboratory for Energy (AGILe), New York Power Authority (NYPA), Albany, NY, USA, since 2022. Prior to joining NYPA, he took various research positions with the Electric Power Research Institute (EPRI), White Plains, NY, USA; the Rensselaer Polytechnic Institute, Troy, NY, USA; the KTH Royal Institute of Technology, Sweden; and Lulea University of Technology, Sweden. He is the co-author of more than 100 scientific papers in accredited journals and international conferences. He is the co-owner of a U.S. patent. His research interests include digital simulation of power systems, integration of renewable energy resources, and applications of PMU data for smart grids. He was the co-winner of the Research and Development 100 Awards. He is currently the Chair of the IEEE Task Force on Digital Twin of Large Scale Power Systems.

ABOUTALEB HADDADI (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from McGill University, Montreal, QC, Canada, in 2015. From 2015 to 2020, he was a Postdoctoral Fellow and a Research Associate with Polytechnique Montreal, Montreal, QC, Canada. Since 2020, he has been a Senior Technical Leader with the Electric Power Research Institute, Palo Alto, CA, USA, where he leads research and development activities related to bulk system and distributed renewable energy resources modeling and system integration impacts. His research interests include renewable resource integration, power system protection, and power system modeling and simulation. He is the Chair of CIGRE working Group C4.60 and is further active in a number of working groups of the IEEE Power and Energy Society, the Western Electricity Coordinating Council, and North American Electric Reliability Corporation.



AHMED SAAD (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from Helwan University, Cairo, Egypt, in 2007 and 2012, respectively, and the Ph.D. degree in electrical and computer engineering from Florida International University, Miami, FL, USA, in 2021. He is currently a Senior Research Scientist with the Electric Power Research Institute (EPRI). His research interests include DERMS, microgrid control, power system operation, control and optimization, industrial cyber-security, and digital twin.

A. H. M. JAKARIA (Member, IEEE) received the B.S. degree in computer science and engineering from Bangladesh University of Engineering and Technology, Dhaka, in 2009, and the Ph.D. degree in computer science from Tennessee Technological University, USA, in 2018. After his graduation, he joined the Electric Power Research Institute, Knoxville, TN, USA, as a Senior Scientist. His research interests include information and network security for NFV and SDN. He is also interested in resiliency issues in UAV and the IoT networks. He focuses on the formal modeling of the problems and solving them efficiently for automated synthesis of network topology and management strategies.

AMINUL HUQUE (Member, IEEE) received the M.Sc. degree from Imperial College London, U.K., in 2003, and the Ph.D. degree from The University of Tennessee, Knoxville, TN, USA, in 2010. He is currently a Program Manager with the Electric Power Research Institute (EPRI), Washington, DC, USA. He manages smart inverter and grid support technology research with EPRI that supports safe and reliable integration of renewables on the power gird. He is a key contributor of distributed energy resources (DER) interconnection standards like IEEE 1547. His research interests include gird integration challenges associated with higher penetration of DERs, including solar photovoltaic, energy storage, vehicle-to-grid, and controllable loads.