T-RECS: A Virtual Commissioning Tool for Software-Based Control of Electric Grids – Design, Validation, and Operation

Jagdish Prasad Achara
EPFL, Switzerland
jagdish.achara@epfl.ch

Maaz Mohiuddin
EPFL, Switzerland
maaz.mohiuddin@epfl.ch

Wajeb Saab
EPFL, Switzerland
wajeb.saab@epfl.ch

Roman Rudnik
EPFL, Switzerland
roman.rudnik@epfl.ch

Jean-Yves Le Boudec
EPFL, Switzerland
jean-yves.leboudec@epfl.ch

Lorenzo Reyes-Chamorro
EPFL, Switzerland
lorenzo.reyes@epfl.ch

ABSTRACT

In real-time control of electric grids using multiple software agents, the control performance depends on (1) the proper functioning of the software agents, i.e., absence of software faults, and (2) the behavior of software agents in the presence of non-ideal communication networks such as message losses and delays. To evaluate the control performance of such systems, we propose T-RECS, a virtual commissioning tool. T-RECS enables testing the performance of software-based control in-silico (before the actual deployment of software agents in the grid), saving both time and money. Developers can run the binaries of their software agents in T-RECS where these binaries exchange real messages by using an emulated network and simulated models of the electric grid and resources. Consequently, the control of an entire microgrid can be tested on a standard computer. In this paper, we first describe the design and the open-source implementation of T-RECS. Second, we measure its CPU and memory usage and show that our implementation can accommodate eight software agents on a standard laptop computer. Third, we validate the simulated grid used in T-RECS by replaying data collected from experiments performed in a real low-voltage microgrid. We find that the average error is 0.037% and the 99^{th} percentile of the error is less than 0.1%. Finally, we present some typical use-cases of T-RECS such as performance evaluation (1) under extreme grid conditions and (2) with non-ideal communication networks. The former, i.e., performance evaluation under extreme grid conditions, is difficult to test in the field due to safety concerns.

CCS CONCEPTS

• **Networks** → **Network performance analysis**; • **Computing methodologies** → **Real-time simulation**; • **Software and its engineering** → *Empirical software validation*;

KEYWORDS

virtual commissioning, real-time systems, software-based control, electric grids, network delay, software crash

ACM Reference Format:
Jagdish Prasad Achara, Maaz Mohiuddin, Wajeb Saab, Roman Rudnik, Jean-Yves Le Boudec, and Lorenzo Reyes-Chamorro. 2018. T-RECS: A Virtual Commissioning Tool for Software-Based Control of Electric Grids – Design, Validation, and Operation. In *e-Energy '18: The Nineth International Conference on Future Energy Systems, June 12–15, 2018, Karlsruhe, Germany.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3208903.3208928

1 INTRODUCTION

In this paper, we consider real-time software-based systems for control of electric grids. Such systems have the core of their control logic in software that is executed by multiple agents [1–4]. These agents are typically distributed all over the grid and communicate using a communication network. They usually either control other lower-level agents or directly control different resources such as a battery, a super-capacitor, or an array of solar panels. The rate of control varies depending on the system but for real-time systems, such as [1], it is typically sub-second.

1.1 Problem

As developers of these systems need to test their software agents before the actual deployment in the field, various testbeds [5–11] are proposed in the literature. For real-time software-based systems, the testing mainly consists of (1) the correct implementation of their distributed control logic and (2) the reliable communication among software agents. We find, however, that these testbeds are not appropriate for such testing.

First, current testbeds cannot test the final executables of software agents that are going to be deployed in the field. Instead these testbeds require either modeling of the control logic or the development of the control system in their testbed. Second, for simulating the electric grid, these testbeds use physical equipments or hardware-in-the-loop, e.g., OPAL-RT eMEGAsim simulator or real-time digital simulator. This incurs a high cost and imposes serious limitations on the ease of use of such testbeds. Moreover, physical equipment cannot be used to study the grid in extreme conditions as this could cause potential damage.

To summarize, the main requirements of such a testbed are: (1) ability to use existing software agents with minimal modifications, (2) avoid use of physical equipment, (3) allow for inducing non-ideal communication and software. A testbed that satisfies these properties can be used by developers of software agents to design, test, and commission the agents before actual deployment in the field. As such tests can be performed entirely in-silico, we term such a testbed as a virtual commissioning tool.

1.2 Proposed Virtual Commissioning Tool

We propose a virtual commissioning tool, called T-RECS, for developers of multi-agent software-based control of electric grids. A preliminary version of T-RECS was presented in short in [12]. The design of T-RECS is divided into four layers: (1) physical layer, (2) sensing and actuation layer, (3) communication layer and (4) control layer.

The first and second layers in T-RECS are simulated in software. The physical grid in the first layer is modeled using the three-phase nodal-admittance matrix (Y-matrix) representation. The evolution of the grid is tracked through complex voltage phasors at each bus. These phasors are obtained by performing a load flow whenever there is a change in the grid state. Electric resources in the first layer, such as battery, load, and photo-voltaic (PV) panels, are also simulated using state-of-the-art models, e.g., a battery model proposed in [13]. Sensors at the second layer are modeled in such a way that they can read the state of the grid from the simulated grid in layer one and then, can send this state to a software agent.

The third layer, i.e., communication network layer, is emulated using the Mininet framework [14]. This enables real packets to be exchanged between software agents, and we can easily study the effect of communication bandwidth, losses, and delays on the control performance. For the fourth layer, T-RECS provides users with virtual containers where software agents can be run without any modifications. Therefore, using T-RECS, we can verify whether the final executables of software agents are free from software bugs and if they correctly implement the control logic. The use of virtual containers also gives us the possibility to simulate the crashes of agents or other software-related issues, hence enables developers to quickly investigate their effect on the control performance.

As described in Section 3, these four layers form the basic architecture of almost all multi-agent software-based control systems for electric grids. As T-RECS applies to all control systems that adhere to this architecture, it can be used seamlessly with a wide-range of control systems [1–4].

Besides satisfying the requirements of a virtual commissioning tool listed earlier, T-RECS is designed to support real-time control systems. This entails fast updates of the simulated physical layer to reflect the changes in the grid and the electric resources, due to sub-second rate of control. This is possible due to our implementation of a fast, recent algorithm for solving the load-flow problem [15].

1.3 Contributions

Our main contribution is the design and implementation of T-RECS, the first virtual-commissioning tool for software-based control of electric grids. T-RECS is implemented entirely in software and therefore reduces the barrier to study software-based control of electric grids. We also make our implementation publicly available[1]. It is worth noting that the contribution of this paper is the integration of several existing concepts such as Mininet [14], fast load-flow [15], and resource models, to obtain a usable and high-performing tool. This is particularly challenging in terms of interoperability between layers due to the heterogeneity of each layer. The detailed contributions are below:

(1) We identify the various layers of a multi-agent software-based control system. Not only does this enable us to design T-RECS and make it generic to support a wide-range of control systems for electric grids, but also this paves way for users of control systems from other domains to design their own virtual commissioning tools.
(2) We present the detailed design of T-RECS and make our Python-based open-source implementation publicly available. To the best of our knowledge, T-RECS is the first virtual commissioning tool that enables studying the performance of real-time software-based control systems, entirely in-silico.
(3) We implement a fast load-flow algorithm [15] and validate its results by comparing with measurements from a real low-voltage microgrid. We find that average error is 0.037% and the 99^{th} percentile of the error is less than 0.1%. This successful validation confirms that T-RECS can be used to replicate results of experiments in real electric grids, thus supporting reproducible research.
(4) We evaluate the CPU and memory usage of T-RECS, in order to show that it can easily support control systems with multiple software-agents in a standard computer.
(5) We present various use-case scenarios where T-RECS can show its value. We study the behavior of control software in extreme conditions of a grid, which is difficult and costly to perform in a real grid. Also, we study the effects of non-ideal communication networks on the control performance. Through these studies, we show that T-RECS can be used throughout the development of software agents, i.e., their design, test, and commission.

The structure of this paper is as follows. In Section 2, we compare T-RECS with the state of the art. We detail the design of T-RECS in Section 3. In Section 4, we present results from validation of T-RECS' grid model. In Section 5, we evaluate the performance of T-RECS. In Section 6, we present two use-case scenarios of T-RECS, taking COMMELEC [1] control system as an example. In Section 7, we conclude our work and gives future perspectives.

2 RELATED WORK

Table 1 presents a comparative summary of requiremetnts (specified in Section 1) that are satisfied by proposed testbed T-RECS and other existing testbeds. We see that none of the existing testbed satisfied all the three requirements of a testbed for software-based control of electric grids. Below we individually disucss the advantages and limitations of each testbed, and compare it with T-RECS.

A testbed for decentralized control of active distribution networks is proposed in [7]. It consists of three main layers. The first layer performs the real-time simulation of physical power system elements in the OPAL-RT eMEGAsim simulator. The second layer requires the development of the multi-agent control system in the Java Agent Development Framework (JADE). Finally, the third layer models and simulates the communication network with OPNET Modeler. T-RECS also has these three layers, but they are managed

[1]https://smartgrid.epfl.ch/?q=t-recs

Table 1: Comparative summary of requirements (specified in Section 1) satisfied by T-RECS and other existing testbeds for software-based control of electric grids.

Testbed	Allows for inducing non-ideal communication and software?	Able to use existing software agents with minimal modifications?	Avoids the use of physical equipments?
[5]	✗	✓	✗
[6]	✗	✓	✗
[7, 8]	✗	✗	✗
[9]	✗	✓	✓
[10, 11]	✗	✗	✓
T-RECS	✓	✓	✓

differently. The first layer, i.e., simulation of physical power system elements, is done in T-RECS using software models instead of using the OPAL-RT eMEGAsim simulator. This has both an advantage and a drawback. The advantage is that T-RECS is inexpensive, scalable, portable, and easily distributable as it does not require the physical equipment (OPAL-RT eMEGAsim simulator). The drawback is that T-RECS cannot study the effect of system transients or switching harmonics on the control software. This is because, in T-RECS, the software models of both the physical grid and electric resources are modeled in the phasor domain. The second layer in [7], i.e., running multi-agent control software, is managed in T-RECS by using software containers provided by the Mininet framework. These containers can directly run existing or developed executables of software agents hence, as opposed to [7], T-RECS permits testing these final agent executables. Finally, the third layer, i.e., communication network layer, is emulated in the software using the Mininet framework. As the emulation of communication networks exchanges real packets, T-RECS enables easy and accurate study of the effects of different network bandwidths, losses, and delays in the communication network on the control performance.

Another testbed is proposed in [9]. Like T-RECS, this testbed is completely software-defined and does not involve physical equipment. However, it is not possible to test the effects of network/ communication technologies on the performance of software agents. This is because the communication network is neither simulated nor emulated. As today's distributed software-based control systems heavily rely on communication among different agents, the communication network is the main source of unexpected behavior of such agents, and not being able to measure it is a limitation of this testbed. Furthermore, as opposed to T-RECS, this testbed does not provide users with software containers, hence executables of multiple software agents cannot be directly run and tested with it. For example, in [9], the authors implemented their energy management software in one of the components of the testbed itself.

Another multi-agent testbed for power systems is proposed in [5]. This testbed is composed of a power-system simulator, computational platforms, and a data-communication infrastructure. As the testbed uses real hardware (computation platforms and communication infrastructure), it is neither inexpensive, portable, easily deployable, nor scalable. According to the authors of [5], these limitations can be removed if the computation platforms can somehow be virtualized or be placed in software containers and if the communications infrastructure can be emulated in the software. However, as noted in Section 3, this exercise poses several challenges due to the heterogeneity of the different components. In T-RECS, we divided the control framework into four manageable layers. This allows us to emulate the network infrastructure (as network layer) and run different agents in multiple software containers (as control layer). Thus, we overcome the limitations of [5].

A real-time testbed for operation, control and cyber-security of power systems is proposed in [6]. It targets the testing of low-level power-system control mechanisms, such as system monitoring and fault detection. However, as opposed to T-RECS, this testbed is not software-defined and consists of many hardware devices such as the real-time digital simulator, the programmable logic controller, NI-PXI controller, and the Ethernet network. Although hardware-in-the-loop might have some benefits, it is not necessary if we target a testbed for the evaluation of effects of software and communication non-idealities on the control performance using software agents. This is the reason T-RECS is designed completely in software and has all the benefits of a pure software solution. Additionally, we find that these hardware devices, used in [6], run modified software (as compared to what runs in the real grid) or run software specifically developed for testbed purposes. This means that the testbed in [6] cannot test and validate the real software that is going to be deployed in the grid. On the contrary, T-RECS runs unmodified executables of software agents, hence a T-RECS user can easily figure out the runtime behavior and bugs of these software agents. Moreover, T-RECS support reproducible research.

In [8], the authors developed an agent-based testbed simulator for power grid modeling and control. They model the agents of the grid, instead of running the real agents in the testbed. As modeling of software agents puts an additional burden on the testbed user and can test only the correctness of the logic, it is not enough to assess the correctness of software agents. The proposed testbed is hybrid: a part of the testbed is in the software, but other parts require the presence of some minimal hardware and actual I/O signals. According to the authors, the hardware-in-the-loop is a complicated architecture and is therefore, not well suited for testing and validating the software-based multi-agent control systems that extensively rely on computational and communication technologies.

To investigate the effects of cyber-contingency on power system operations, a co-simulation model, based on information flow, is proposed in [10]. The authors model the network contingencies at a low level, e.g., delayed, disordered, dropped, and distorted information flows. The authors claim that these low-level parameters are easier to model than high-level network parameters such as DoS, CLO, and MITM. In contrast to this work, where they simulate the communication network with these low-level parameters, T-RECS emulates the communication network by using Mininet, which enables us to study the effects of different network bandwidths, losses, and delays corresponding to multiple real-world scenarios. As message exchanges are emulated in T-RECS, it accurately captures the real-time properties of the control protocol. Another important distinction of this work with T-RECS is that, in [10], the decision-making layer, i.e., software agents, is also simulated,

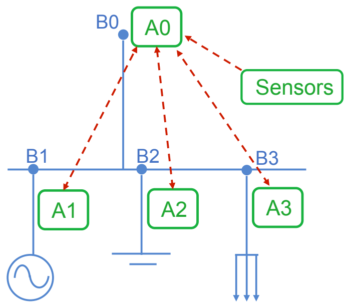

Figure 1: Architecture of a real-time software-based control system for electric grids

whereas T-RECS can run the real software agents without requiring the development of models of software agents.

To run software agents, Mosaik [11] uses process-based software containers, based on SimPy (a discrete event simulation platform). However, it has two drawbacks. First, it does not model or emulate the communication network between different agents, thus it cannot be used to study the effect of non-ideal communication networks on the software-based control systems. Second, the control agents need to be re-written using Mosaik API, whereas T-RECS does not suffer from these limitations.

Next we discuss our choices for selecting different software solutions for T-RECS layers. To simulate the physical grid in the phasor domain, we use the three-phase load-flow algorithm proposed in [15]. We do not use the Newton-Raphson (NR) method for our load-flow computations, because the computation time of the chosen algorithm is faster than the load-flow implementation using NR method (see Section 5). With regard to modeling of electric resources, we use existing state-of-the-art models, e.g., the battery model proposed in [13]. Moreover, T-RECS is designed such that users can plug-in new models of electric resources as/if needed.

Apart from the testbeds designed for control of electric systems, there exists a vast amount of literature on modeling or simulating the electric resources and grid. Authors in [16–18] propose models for the most common resources such as loads, converters, batteries, solar panels, and electric vehicles, whereas the grid is simulated in phasor domain in [16, 17, 19]. With regard to modeling electric resources, T-RECS provides users with some state-of-the-art models but lets users plug in new models as needed. To simulate the electric grid in the phasor domain, we do not use PyPower [19] as it supports only single-phase load flow. Also, the grid model provided by GridLAB-D [16] is not appropriate because (1) it neglects phase-to-ground coupling capacitance, (2) it cannot take as input the new power setpoint at sub-second level on a given node, and (3) the input to the grid model are physical parameters of lines such as type and length of wires instead of a Y-matrix. In T-RECS, to simulate the grid, we use the three-phase load flow algorithm proposed in [15].

3 T-RECS DESIGN

Figure 1 shows the architecture of a typical real-time software-based control system for electric grids. The grid in this example

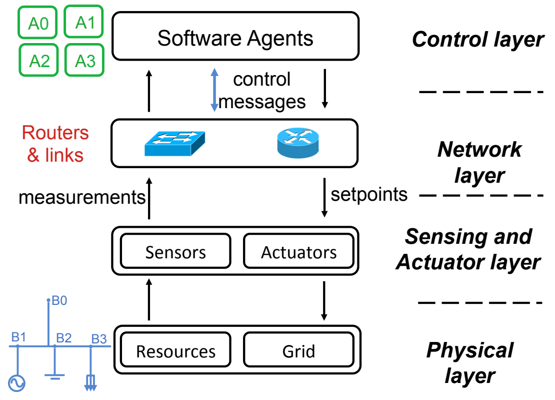

Figure 2: Layers in an real-time software-based control system and their respective elements.

comprises 4 buses, $B_0 - B_3$. Bus B_0 is the slack bus, and buses B_1, B_2 and B_3 have a generator (a producing resource), a battery (a prosuming resource) and load (a consuming resource), respectively. The control of the grid and the resources is performed by software agents $A_0 - A_3$. Software agent A_0 senses the state of the grid (voltage/power at the buses, current at each line) through sensors such as voltage/current sensors or phasor measurement units (PMUs). Software agents $A_1 - A_3$ read the state of the respective resources, namely, generator, battery and load, through a sensor interface provided by the resource. For example, agent A_3 reads the internal state of a load, such as current temperature (for a thermal load), through the on-board thermostat installed on the load. The sensed quantities are exchanged as measurements among the software agents.

In order to control the resources, the software agents perform computations and send out setpoints, thereby keeping the grid in the desired state. The setpoints are implemented by the actuator interfaces at the resources. For example, a setpoint for changing the power injected by a battery is implemented by the converter on the battery, i.e., the actuator. In Figure 1, the software agents $A_1 - A_3$ control the respective resources by sending setpoints to them, whereas A_0 sends setpoints only to other software agents.

Taking a first step towards translating such an control system in silico, we divide the various components into layers as shown in Figure 2. The various layers are:

(1) Physical layer — consists of the electric grid and the resources (load, battery, generators).
(2) Sensing and actuation layer — consists of sensors that read the state of the physical layer and of actuators that alter the state of the physical layer.
(3) Network layer — represents the communication infrastructure among software agents, sensors, and actuators.
(4) Control layer — comprises the software agents ($A_0 - A_4$) that use the measurements from sensors to perform computations and output setpoints for the actuators.

Figure 3 shows the overview of T-RECS design. The physical layer, and the sensing and actuation layer are simulated using state-of-the-art models of the grid and electric resources, respectively.

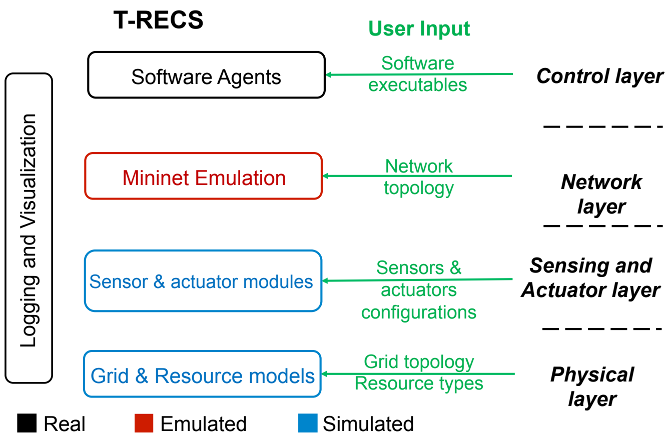

Figure 3: Design of T-RECS.

To this end, T-RECS requires the grid topology, the resource types and their parameters as input. The details of the design of physical layer and actuation layer are described in Sections 3.1 and 3.2, respectively.

The network layer is emulated in Mininet [14], which uses virtual switches and hosts to simulate the switches and routers in the network. For this layer, T-RECS takes as input the topology of the communication network, the bandwidths, the losses, and delays of different links. The detailed design of the network layer is presented in Section 3.3.

The control layer is built through unmodified software agents that are run in Linux containers [20] provided by Mininet. The detailed design of the control layer is presented in Section 3.4.

In line with the requirements of a virtual commissioning tool listed in Section 1, the layers of T-RECS are built in software, thus making it feasible to study all elements of real-time control of the grid within a computer.

3.1 Physical Layer

The two components of the physical layer are the grid, and the resources.

3.1.1 Grid. The grid is represented as a set of complex voltage phasors at each bus. To this end, the grid is modeled by its nodal-admittance matrix. We use a three-phase model in order to be able to simulate distribution networks that are often unbalanced. For example, the grid model for the control system in Figure 1 will take as input the grid topology (connection between the buses and the line parameters) and computes the nodal-admittance matrix. The frequency of the grid is taken as an input and can change during the execution, e.g., could be dictated by the slack bus (B_0) or the generator on bus B_1.

For each change of voltage or power at a bus, the grid model performs a three-phase load flow to obtain the voltage at all the buses and the current in all the lines of the grid. As the speed of the load-flow computation dictates the responsiveness of the grid-model, we chose to implement the method proposed in [15]; it boasts a significant decrease in computation time when compared to classic Newton-Raphson method. In Section 5.2, we study how the computation time of the load-flow evaluation varies with different sizes of the grid.

As the load-flow analysis is a light-weight representation of the grid, it provides the steady-state information of the grid without considering system transient processes. We make this trade-off in order to be able to simulate the grid in-silico. In Section 4, we compare results from the load-flow analysis of the grid-model to those obtained from a real grid and show that the average error is 0.037% and the 99th percentile of the error is less than 0.1%.

3.1.2 Resources. The resources in T-RECS are simulated using existing models of electrical resources. Broadly, there are two types of resources, controllable and uncontrollable. For example, in Figure 1, the battery is a controllable resource, as its output power can be modulated through setpoints, whereas the load could be an example of an uncontrollable resource (assuming it cannot be curtailed).

In our open-source implementation, we make four resource models available: battery, uncontrollable photo-voltaic (PV) panel, uncontrollable load, and a controllable flex-house. The battery is modeled using the two-time constant model described in [13]. The uncontrollable load and PV source are simulated by replaying a time-stamped trace of the power injections. The flex-house model captures the heat dynamics of buildings and is used to simulate controllable thermal loads, as described in [21]. The change in the output of a resource is reflected as a change in the state of the bus on which the resource is placed.

The open-source implementation of T-RECS enables us an addition of new resource models. Each resource model performs three tasks. First, the resource periodically updates its state according to the resource dynamics. For example, the battery updates its state-of-charge (SoC) based on the time-elapsed and output power. Second, when it receives a request from a sensor, it responds with its internal state. Lastly, when it receives a request from an actuator, it changes the output power and the power at the corresponding bus in the grid model.

3.2 Sensing and Actuation Layer

The sensors and actuators are simulated as application programming interfaces (APIs) that act as an interface between the physical layer and the control layer. There are two types of APIs, the grid API and the resource API. The grid API provides methods to get the state of the grid and set the power at each bus (except slack bus) in the grid. The state of the grid consists of voltage phasors and active/reactive power at each bus, and line currents. The resource API provides functions for reading and changing the state of the resource model, such as power injected by a battery. These APIs can be used to create specific sensors and actuators, as required by the user's configuration.

In the current release, T-RECS provides an implementation a simulator of a real-time state estimator that periodically calls the grid API to get the state of the grid and then, periodically sends the state of the grid in a given format to a list of agents specified by the user. We also have implementation of an actuator that alters the state of the battery resource used in the experiments in Section 4 and Section 6.

In Figure 1, the sensor that sends data to software agent A_0 is an example of a use case of such a sensor. The streamed messages are sent as real packets via the network layer.

3.3 Network Layer

The network layer in T-RECS is emulated using virtual switches, routers and hosts provided by Mininet [14]. The main advantages of using emulated network as opposed to simulated network done by other works are: (1) real messages are exchanged in Mininet in contrast with discrete-event simulation of messages, (2) effect of bandwidth-limitations can be accurately studied because real switches are emulated, and (3) newly developed network protocols can be easily studied without any modification to the protocol implementation.

The switches are realized by Open vSwitch [22], a programmable multi-layer switch that can be used to accurately emulate practically any L2 and/or L3 network topology. It can also be interfaced with a software-defined networking controller to emulate a large-scale managed network used in sub-station automation networks.

Using the rich set of tools from the Linux traffic-control suite `tc-netem` [23], we impose bandwidth and delay restrictions on the links to replicate a real-life network topology. Additionally, T-RECS enables us to use message-loss profiles provided by `tc-netem` and by queuing disciplines to capture the real-life network more accurately. This facilitates studying the performance of the control system in non-ideal network conditions, a requirement of such a virtual commissioning tool.

The end-hosts of the network are Linux containers that (provided by Mininet) are used to run the software agents in the control layer. The network topology and the link configurations are taken as an input.

3.4 Control Layer

The control layer is identical to the real world. T-RECS takes the executables of the software agents as input and executes them in the software containers provided by Mininet. Just as in an actual deployment, the unmodified executables receive messages from sensors, perform computations, and send setpoints to actuators. For the example of Figure 1, the control layer is realized by executing software agents $A_0 - A_4$, one in each host, with each one receiving real measurements from sensors and sending real setpoints to actuators, just as they would run in the field. In this way, T-RECS recreates an environment in which the software agents can be executed and tested without modifying their code, as envisioned in our requirements for such a tool.

Although our current implementation runs on a single computer, it is straight-forward to extend to run on a cluster of many computers for very high-scalability. A single computer in a cluster could host several software agents, and the communication links between them can be modified to reflect the real-life network using the same set of tools from the Linux traffic-control suite.

Note that, although the focus of this paper is only control systems for electric grids, the layering scheme described above can be applied to other domains. Consider the example of a self-driving car [24]. The physical layer comprises the dynamics of the car and the dynamics of the environment it runs in, both of which are governed by laws of classic mechanics. The sensing and actuation layer sensors include the speedometer for the current speed, and the actuator includes a PID controller that maintains the speed at a given setpoint. The control layer includes the software agents that detect objects, perform obstacle-avoidance, navigation, etc. In fact, the same design philosophy used in T-RECS can be used to design low-cost, in-silico, virtual-commissioning systems for multi-agent software-based control systems in other domains.

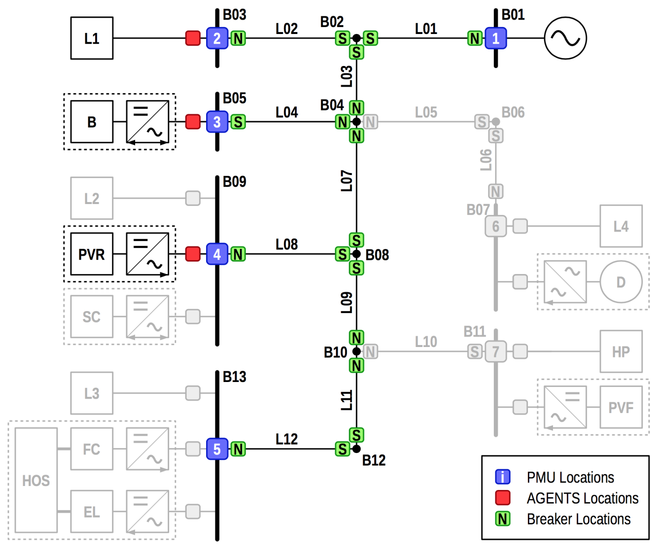

Figure 4: Test setup for validation. All elements used in the experiments are shown in solid points.

4 VALIDATION

In this section, we validate the T-RECS grid-model described in Section 3.1.1. Recall that the T-RECS grid model performs load-flow computation and updates the state of the grid whenever there is a power injection or absorption at a bus. Thus, this section aims to quantify the error committed by the load-flow solver of T-RECS grid model as compared to the measured state of the grid.

Figure 4 shows the topology of DESL-LCA2 microgrid at EPFL. This microgrid is used in the experiments described below. It consists of 13 buses, labeled B01-B13, and reproduces, in real scale, the topology defined by the CigrÃľ Task Force C6.04.02. The resources used in our experiments are shown in solid points in Figure 4 and consists of a 24 kW controllable load on bus B03, a 20 kW, 25 kWh controllable battery on bus B05, and an uncontrollable PV generator of 13 kWp on bus B09. The microgrid is monitored in real-time using phasor measurement units (PMUs), a phasor data concentrator (PDC), and a real-time state estimator (RTSE) [25]. From RTSE, we obtain the timestamped traces of voltage and current phasors at each bus, with one measurement every 20 ms.

The traces are collected during experimental validation of a real-time control framework, called COMMELEC [26]. COMMELEC is a real-time framework that controls the given resources in real-time using explicit power setpoints and software agents [1]. The software agents for the resources, called as resource agents, are

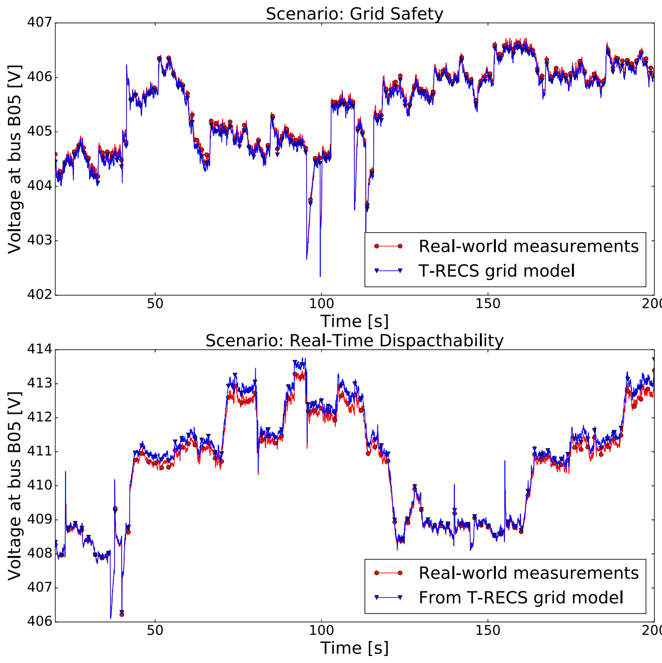

Figure 5: Voltage at the battery bus (B05) obtained from measurements and from grid-model.

co-located with the respective resources and communicate with a centrally located grid agent. The grid agent receives the state of different resrouces from their respective resource agents and sends them setpoints in order to implement a given policy.

We validate the T-RECS grid-model under two different scenarios, with two separate policies from [26]: grid-security and real-time dispatchability. In grid-security, the grid agent maintains the grid in a feasible state, i.e., respects the ampacity limits of all the lines, voltage limits of all the buses, and the constraints of all the resources, in a grid with uncertainty in power prosumption due to the load and the PV. In real-time dispatchability, the grid agent tracks an external dispatch signal while maintaining the grid in a feasible state.

In order to quantify the error between the measurements from the experiments and the output of the load-flow solver in T-RECS, we use measured voltage and power traces from the two experiments as follows. At every 20 ms timestamp, we use the power injections at all the buses and give them as input to the grid-model along with the voltage magnitude at the slack bus as input to the grid-model. The grid-model performs the load-flow computation and returns the voltage at each bus. At each bus, we compare this voltage against the voltage obtained from the measurement traces at the same timestamp.

For the two scenarios mentioned earlier, Figure 5 shows the voltage at the battery bus (B05) obtained from the measurements and from the grid model during a three-minute window. We see that the voltage from the grid model closely follows the measurements. However, we note that the error is relatively higher in the second scenario of real-time dispatchability. This is because the instantaneous grid parameters (resistance, reactance, and susceptance

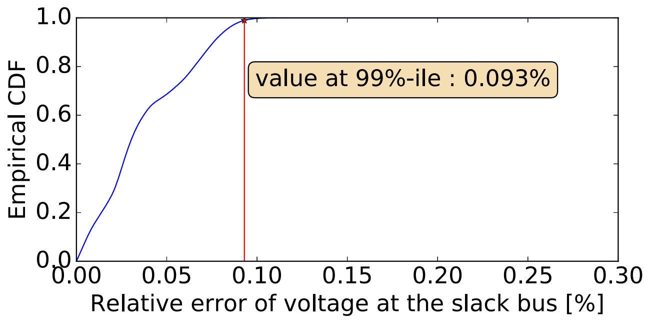

Figure 6: Empirical CDF of the relative error in voltage at all the buses.

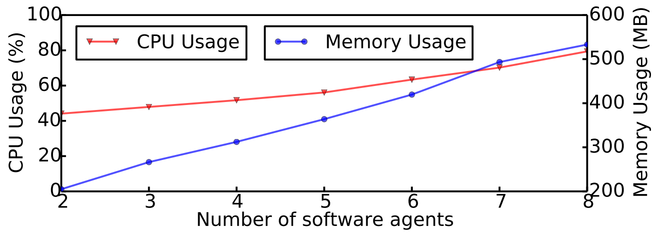

Figure 7: CPU and memory usage of T-RECS, on a laptop with 3.7 GB RAM and a 2.67GHz Intel Core i7 processor, as a function of number of software agents. CPU usage in percentage is cumulative of all four CPUs of the i7 processor.

of lines) are a function of temperature and frequency of line. The voltage error depends on the difference between the instantaneous grid parameters of real grid and the static grid parameters used by T-RECS. The impact of this difference in parameters is higher for higher voltage amplitudes. This error is unavoidable because it is hard to estimate the instantaneous grid parameters, but we see below that it is $< 0.1\%$.

Figure 6 shows the empirical cumulative-distribution function (CDF) of the error between the voltage measured by PMUs at each bus and the voltage obtained by load-flow from the grid-model. The CDF is computed using the entire data from the experiments that amounts to 750,000 data points. We see that the average value of the relative error is 0.037% and its value at 99%-ile is 0.093%. Thus, we conclude that error committed by the grid-model is negligible.

5 PERFORMANCE EVALUATION

In this section, we do two performance studies of our T-RECS implementation. First, in Section 5.1, we evaluate the CPU and memory usage of T-RECS as a function of number of software agents in the control system. The goal of this performance evaluation is to show that T-RECS can easily accommodate several software agents on a standard laptop computer. Then, in Section 5.2, we study the execution time of the load-flow computations by the grid-model for three benchmark grids of different sizes. The aim of this study is to show that due to its ability to quickly update the state of the grid, T-RECS is suitable for real-time control systems for electric grids that have sub-second rate of control.

All experiments are performed on a Lenovo T400 laptop with 3.7 GB RAM and 2.67GHz Intel Core i7 processor. The operating system is 64-bit Ubuntu 16.04 LTS and the Intel virtualization technology is enabled.

5.1 CPU and Memory Usage

We use the COMMELEC control system with the CIGRÉ benchmark low-voltage microgrid with 13 buses [27], same as used in Section 4. Recall from Section 4, that the COMMELEC control system has as software agents, a grid agent (GA) that control one or more resource agent (RA). Moreover, each resource agent is attached to one controlled resource. For example, if COMMELEC controls two resources, then it has to run three software agents: one GA and two RAs corresponding to the two resources.

To better interpret our performance results, it is important to highlight that the COMMELEC control system is run at the pace of 100 ms, i.e., RAs send state of their controlled resources every 100 ms to the GA and GA sends power setpoints to RAs every 100 ms. This has two implications for T-RECS: (1) there is a heavy load of messages to emulate for T-RECS communication layer, and (2) frequent computations of load flow in T-RECS grid module. Additionally, as COMMELEC GA runs at 100 ms pace and requires the state of the grid every 20 ms as input, T-RECS sensor module sends this information to GA every 20 ms.

We run multiple experiments with different numbers of COMMELEC software agents (and corresponding controllable resources) and record the CPU and memory usage of T-RECS in each case. The CPU usage reported below is cumulative of all four CPUs of the i7 processor and is solely of T-RECS processes.

When T-RECS is not running, the CPU usage is 1.5% and the memory usage is 563 MB. Fig. 7 shows how the CPU and memory usage scales with the number of software agents or controllable resources. We find that the CPU usage of T-RECS starts off at 44.1% in case of 2 software agents and increases linearly with a rate of 6% per additional software agent or controllable resource. Moreover, the memory footprint of T-RECS is close to 200 MB with 2 software agents and increases at a rate of around 55 MB per additional resource or software agent.

The initial high CPU usage can be explained by the start of all the T-RECS's components (such as load-flow engine in the grid-module, the sensor module, the network virtualization used by Mininet, etc.). Also, the increase in the CPU usage with the number of software agents is linear, as expected. This is because, with an additional software agent, the burden of T-RECS increases linearly in the following three directions. First, it needs to run one additional resource model of the resource managed by the new software agent. Second, the number of load-flow computations in the grid engine increases linearly with the number of resource model because the number of updates send to the grid module increases by one. Finally, the communication network emulation layer needs to emulate a fixed number of more packets.

We see that, while running in a modest laptop, T-RECS can support up to eight software agents (one COMMELEC GA and 7 RAs). To put this in perspective, a typical microgrid is controlled with one GA and about five RAs. Hence, we conclude that a developer of software-based control systems can easily use T-RECS in a general-purpose desktop/laptop for virtual commissioning.

Benchmark Grid	T-RECS Load-Flow [15]	Newton-Raphson
CIGRÉ 4-bus	0.75 ± 0.02	23.28 ± 0.39
CIGRÉ 13-bus	1.13 ± 0.02	232.19 ± 1.05
CIGRÉ 34-bus	7.42 ± 0.14	1594.26 ± 5.29

Table 2: Average execution times (in ms) of the the load-flow implementation used in T-RECS [15] and the Newton-Raphson method for three different CIGRÉ benchmark grids [28] measured at 95% confidence level.

5.2 Load-Flow Computation

Recall from Section 3.1.1, that every time the power injected or consumed by a resource changes, a load-flow computation is triggered by the grid-model to reflect the effect of the new power injection or consumption on the other buses in the grid. The time taken by the load-flow computation is therefore, the time taken for a given update to reflect in the simulated grid of T-RECS. Therefore, it is important to quantify the execution time of the load-flow performed by the grid model of our implementation. This importance is even more prominent in case of real-time control systems with sub-second rate of control. For example, in COMMELEC, the control is takes place at a pace of 100 ms. Therefore, the time taken for the computation and propagation of grid state from T-RECS grid module to the GA has to be less than 100 ms.

The load-flow computation algorithm used by T-RECS is described in [15]. We compare the time taken by the Python implementation of this algorithm in T-RECS for three different CIGRÉ benchmark electric grids [28]. These grids are of different sizes and consist of 4, 13, and 34 buses, respectively. In our tests, we set the desired accuracy of load-flow results as 1e-6 and the maximum number of iterations are set to 100. To highlight the improvement in computation time over traditional load-flow solvers that use Newton-Raphson method, we have also implemented the load-flow computation by using state-of-the-art Newton-Raphson algorithm in Python. We also run the same test cases with this algorithm.

In Table 2, for the three grids of different sizes, we report the mean computation time and the confidence interval for the mean at 95% confidence level computed from 100 samples. We see that the average computation-time of the T-RECS grid-model for grids of 4, 13 and 34 buses are 0.75 ms, 1.13 ms and 7.4 ms, respectively. This computation time is well-below the required update time (of 100 ms) for a real-time control system like COMMELEC, thereby confirming the real-time capability of T-RECS. We also observe a sharp decrease in computation when compared to traditional load-flow solver that takes 23 ms, 232 ms, and 1.6 seconds for the same grids, respectively. This affirms our choice to use the fast, recent load-flow algorithm proposed in [15].

6 USE CASE SCENARIOS

In this section, we present two use-case scenarios that highlight the range of experiments that can be performed using T-RECS. The first use-case exposes the grid to extreme conditions, with the possibility of an over-voltage, whereas the second use-case deals with non-idealities in the communication network. The COMMELEC control system [1], that was previously discussed in Section 4, is used for both experiments as its implementation is readily available.

6.1 Extreme Grid Conditions

Recent work [29] has shown that COMMELEC performs poorly in the presence of large uncertainties, as it aims to maintain the grid bus voltages within strict ±10% bounds throughout its operation by preventing worst possible scenario. The authors in [29] introduce dynamic bounds that enable the GA to violate the voltage constraints for an interval of 500 ms, as allowed by grid standards [30]. This improves the performance of COMMELEC, allowing for optimal dispatch plan tracking, frequency support, or any other higher level functionality that the GA is performing.

Introducing dynamic bounds requires a software patch to the COMMELEC GA, in addition to tuning certain parameters that vary depending on the grid. One such parameter, α, affects the time of allowed voltage violations. α is the multiplier of the voltage penalty function when the voltage is close to violation. The central idea of this patch [29] is to dynamically increase the cost of the penalty on voltage violation when the voltage is close to the allowed limits. A large value of α leads the GA to return to safe voltage-region quicker. However, during operation, a very large value of α may cause high oscillations in power setpoints computed by GA and hence undesired high oscillations in voltage. In the current state of affairs, there is no theoretical analysis to decide for an optimal value of α; it needs to be tuned for every specific grid. In our experiments, we found that its value typically lies between 10 and 100,000.

Thus, tuning α requires extensive testing that cannot be performed on the real grid, as it involves risk of overvoltages, which can result in severe financial and safety consequences. As T-RECS runs the actual code of the GA, it is directly applicable for performing such tests in-silico, while replicating the grid conditions. This eliminates the risks of performing the same study prematurely on the actual grid.

We consider a scenario in which the grid consists of a 270kW array of PV cells, a 120kW battery, and a 40kW electric vehicle (EV) charging station. The grid is connected to a slack bus, to which it exports 160kW. An EV suddenly disconnects from the charging station after some time, driving the power absorbed down from 40kW to 0kW, and creating a voltage-step. The COMMELEC GA is aware of the possibility of such a disconnection, but due to dynamic bounds, maintains optimal operation prior to the disconnection, although the risk of overvoltages is present. The parameter α dictates how strict the GA behaves after the violation occurs, and how fast it drives the voltage back to safe region.

We use T-RECS to perform this study, by simply applying the software patch to the GA code, and configuring α prior to each run. Figure 8 shows the voltage profile at the bus containing the EV charging station for two values of α. We see that for $\alpha = 78$, the violation lasts much longer than 500 ms, whereas the violation interval for $\alpha = 75$ is within limits. Table 3 shows more results for different values of α, as we performed a binary search in order to find the optimal value.

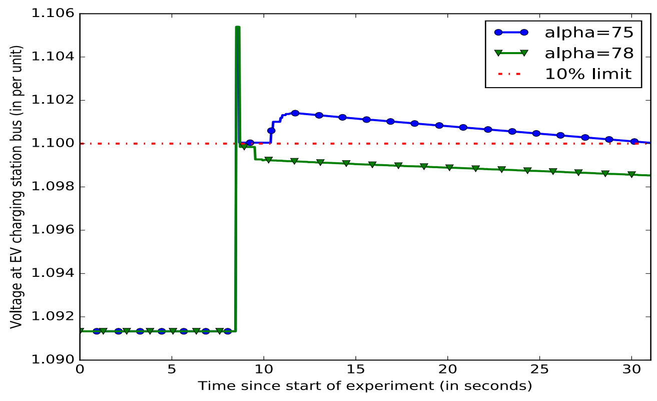

Figure 8: Voltage violations for different values of α

Alpha (α)	Violation time (ms)
75	23004
75.5	22813
75.6	22769
75.8	149
76	132
78	152

Table 3: Voltage violation interval for different values of α

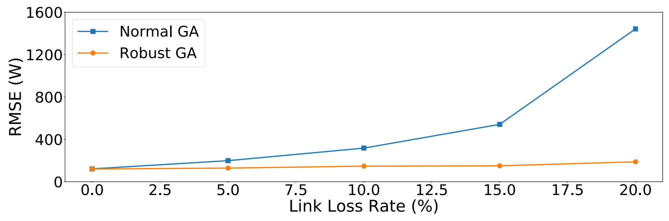

Figure 9: Root-mean-square-error (RMSE) between power at the slack and the reference signal for two versions of the COMMELEC GA at different loss rates

Finally, we can use a value of α for which the violations are within limits, and deploy the dynamic bounds patch on the actual grid without risk. The results on the actual grid will be very close to the results obtained in T-RECS, as seen in Section 4.

6.2 Communication Network Non-Idealities

In the second use-case, we highlight the role T-RECS plays in the co-development of COMMELEC, with a focus on the effect of communication network non-idealities. We use the same setup shown in Figure 4, with a battery, an uncontrollable PV, and a load as discussed in Section 4.

We use T-RECS first to study the effect of message losses on the ability of the COMMELEC GA to track a reference signal at the slack. To do this, we vary the probability of message loss between 0% and 20%, and measure the root-mean-square-error (RMSE) between the power at the slack and the reference signal. The results are

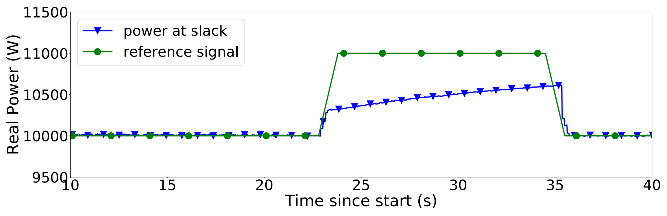

Figure 10: Tracking experiment of Only-long GA with binding grid conditions and a 2% loss rate

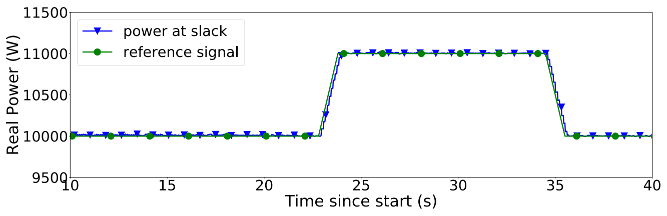

Figure 11: Tracking experiment of Robust GA with binding grid conditions and a 2% loss rate

shown in Figure 9, in which we see that the RMSE of the Normal GA quickly increases.

These results led to the development of an improved version of the GA, dubbed the Robust GA [31]. The Robust GA instructs the RAs to send information about the state of their resource, which is valid for a longer time horizon, in addition to the short-horizon messages originally sent. The long-term messages serve as an input to the GA when it does not have the most recent information about the state of the resources, due to message losses. The short-term messages are still used when present as they are more accurate.

T-RECS is extensively used in the co-development of the Robust GA, as it reduces the time required for the software cycle of design, testing and commissioning. In each iteration of the test, we simply re-run the T-RECS setup with a new version of the GA executable. After several iterations of this cycle, we arrive at the desired implementation of the Robust GA. Figure 9 shows that the Robust GA maintains a low RMSE even under a 20% loss rate. We arrived at the implementation of the Robust GA without the need to have it tested on the real grid. Instead, thanks to the successful validation of the grid-model of T-RECS, the newly obtained Robust GA can be confidently deployed in the real grid, while maintaining the same behavior as shown in Figure 9.

An alternative version of the Robust GA, in which only long-term messages were exchanged in order to save on bandwidth, was initially proposed. This version, Only-long GA, was evaluated in T-RECS, and showed a similar advantage to the Robust GA. However, as extensive testing eventually showed, the Only-long GA fails to track a reference signal when the grid is in binding conditions.

Figures 10, 11 show the results of both the Only-long GA and the Robust GA, respectively, in an experiment where one of the grid lines is close to its ampacity limits. We notice that the Only-long GA, having only long-term information, is very conservative when it comes to injecting power into this line, and consequently fails to track the reference signal. The Robust GA, on the other hand, uses the available short-term messages, and can thus be more aggressive in following the reference signal, while maintaining grid safety.

7 CONCLUSION AND FUTURE WORK

We present T-RECS, a virtual commissioning tool, that is used for design, testing, and validation of multi-agent real-time control software for electric grids. It enables developers to test the executables of their software agents without requring any modifications to them. The effect of non-ideal communication networks on the control performance can be studied using T-RECS as real packets are being exchanged between software agents. This is made possible by emulating the communication network layer in T-RECS using Mininet. T-RECS simulates the physical grid using a phasor-domain load flow solver, and uses state-of-the-art models to simulate the electric resources. To the best of our knowledge, T-RECS is the first virtual commissioning tool for real-time software-based control systems for electric grids.

The main design criteria for T-RECS are the ability to run unmodified code, operate in real-time, and study the effect of non-ideal communication network on the control performance, all without requiring physical equipment. Indeed, T-RECS can be run entirely on a standard PC or laptop. This makes T-RECS the ideal tool for reproducible research in the field of control of electric grids.

We make available an implementation of T-RECS and validate the load-flow solver in T-RECS by running the same experiment in a real microgrid. We find that the tail relative error is less than 0.1% if we compare the results from the two experiments. We show the capabilities of T-RECS through two use-cases that highlight its integral role in the development process of COMMELEC (an in-house control application for electric grids).

We are currently working on the full-stack validation of T-RECS. Our preliminary work shows exact qualitative match between the results obtained from T-RECS and the real-world experiments. We are now working on the quantitative comparison of results from the T-RECS and the electric grid in [26]. One of main challenges we are currently encountering is matching the initial conditions of the various resources in the grid. Furthermore, the validation study is often delayed due to the unavailability of the grid and various resources (they require regular maintenance, resulting in downtime). Indeed, this challenge is a motivation for developing T-RECS in the first place, as it provides a testbed without the issues that come inherently with hardware.

A key feature we wish to incorporate in T-RECS is virtual time. Currently, the pace of the experiments in T-RECS is dictated by the pace of the software agents. For instance, in COMMELEC, the agents communicate every 100 ms. Thus, in order to visualize the impact of COMMELEC after one million runs, it would take more than a day of runtime. With the introduction of virtual time, such an experiment could be run at a faster speed, reducing the time needed by a long-term study.

ACKNOWLEDGEMENTS

This research was supported by the "SCCER - FURIES" project and the "SNSF - NRP 70 Energy Turnaround" project.

REFERENCES

[1] A. Bernstein, L. Reyes-Chamorro, J.-Y. Le Boudec, and M. Paolone, “A Composable Method for Real-Time Control of Active Distribution Networks with Explicit Power Setpoints. Part I: Framework,” *Electric Power Systems Research*, 2015.
[2] Z. Xiao, T. Li, M. Huang, J. Shi, J. Yang, J. Yu, and W. Wu, “Hierarchical MAS Based Control Strategy for Microgrid,” *Energies*, vol. 3, no. 9, pp. 1622–1638, 2010.
[3] K. Christakou, D.-C. Tomozei, J.-Y. Le Boudec, and M. Paolone, “GECN: Primary Voltage Control for Active Distribution Networks Via Real-Time Demand-Response,” *IEEE Transactions on Smart Grid*, 2014.
[4] J. Lin, K.-C. Leung, and V. O. Li, “Optimal Scheduling with Vehicle-to-Grid Regulation Service,” *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 556–569, 2014.
[5] M. J. Stanovich, S. K. Srivastava, D. A. Cartes, and T. L. Bevis, “Multi-Agent Testbed for Emerging Power Systems,” in *Power and Energy Society General Meeting (PES), 2013 IEEE.*
[6] R. M. Reddi and A. K. Srivastava, “Real Time Test Bed Development for Power System Operation, Control and Cyber Security,” in *North American Power Symposium (NAPS), 2010.*
[7] A. Saleem, N. Honeth, Y. Wu, and L. Nordstrom, “Integrated Multi-Agent Testbed for Decentralized Control of Active Distribution Networks,” in *Power and Energy Society General Meeting (PES), 2013 IEEE.*
[8] F. Maturana, R. Staron, K. Loparo, and D. Carnahan, “Agent-Based Testbed Simulator for Power Grid Modeling and Control,” in *Energytech, 2012 IEEE.*
[9] A. Ravichandran, “Software-Defined MicroGrid Testbed for Energy Management,” *Master Thesis*, 2011.
[10] Y. Cao, X. Shi, and Y. Li, “A Simplified Co-simulation Model for Investigating Impacts of Cyber-Contingency on Power System Operations,” *IEEE Transactions on Smart Grid*, 2017.
[11] S. Schutte, S. Scherfke, and M. Troschel, “Mosaik: A Framework for Modular Simulation of Active Components in Smart Grids,” in *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*, Oct 2011.
[12] J. Achara, M. Mohiuddin, W. Saab, R. Rudnik, and J.-Y. Le Boudec, “T-RECS: A Software Testbed for Multi-Agent Real-Time Control of Electric Grids,” in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2017, pp. 1–4.
[13] M. Bahramipanah, D. Torregrossa, R. Cherkaoui, and M. Paolone, “Enhanced Electrical Model of Lithium-Based Batteries Accounting the Charge Redistribution Effect,” in *IEEE Power Systems Computation Conference (PSCC), 2014.*
[14] B. Lantz, B. Heller, and N. McKeown, “A Network in a Laptop: Rapid Prototyping for Software-defined Networks,” in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010. [Online]. Available: http://mininet.org/
[15] C. Wang, A. Bernstein, J.-Y. Le Boudec, and M. Paolone, “Explicit Conditions on Existence and Uniqueness of Load-Flow Solutions in Distribution Networks,” *IEEE Transactions on Smart Grid*, 2016.
[16] D. P. Chassin, J. C. Fuller, and N. Djilali, “GridLAB-D: An Agent-Based Simulation Framework for Smart Grids,” in *Journal of Applied Mathematics*, 2014.
[17] M. S.E., E. H., and J. Broenink, “Modelica - An International Effort to Design the Next Generation Modeling Language,” in *Benelux Quarterly Journal on Automatic Control*, 1998.
[18] “OpenDSS,” http://smartgrid.epri.com/SimulationTool.aspx.
[19] “PyPower,” https://pypi.python.org/pypi/PYPOWER.
[20] Wikipedia, “LXC,” https://en.wikipedia.org/wiki/LXC, accessed: 2018-01-22.
[21] P. Bacher and H. Madsen, “Identifying Suitable Models for the Heat Dynamics of Buildings,” *Energy and Buildings*, vol. 43, no. 7, pp. 1511–1522, 2011.
[22] Linux Foundation, “OvS: Open vSwitch,” http://openvswitch.org/, accessed: 2018-01-22.
[23] S. Hemminger *et al.*, “Network Emulation with NetEm,” in *6th Australia’s National Linux Conference (LCA 2005), Canberra, Australia (April 2005).* Citeseer, 2005, pp. 18–23.
[24] C. Urmson, J. A. Bagnell, C. R. Baker, M. Hebert, A. Kelly, R. Rajkumar, P. E. Rybski, S. Scherer, R. Simmons, S. Singh *et al.*, “Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge,” Robotics Institute , Carnegie Mellon University, Tech. Rep., 2007.
[25] M. Pignati, M. Popovic, S. Barreto Andrade, R. Cherkaoui, D. Flores, J.-Y. Le Boudec, M. M. Maaz, M. Paolone, P. Romano, S. Sarri *et al.*, “Real-Time State Estimation of the Epfl-Campus Medium-Voltage Grid by Using Pmus,” in *The Sixth Conference on Innovative Smart Grid Technologies (ISGT2015)*, no. EPFL-CONF-203775, 2014.
[26] L. Reyes-Chamorro, A. Bernstein, N. J. Bouman, E. Scolari, A. Kettner, B. Cathiard, J.-Y. Le Boudec, and M. Paolone, “Experimental Validation of an Explicit Power-Flow Primary Control in Microgrid,” in *EEE Transactions on Industrial Informatics*, no. EPFL-ARTICLE-234511, 2018.
[27] S. Papathanassiou, N. Hatziargyriou, K. Strunz *et al.*, “A Benchmark Low Voltage Microgrid Network,” in *Proceedings of the CIGRE symposium: power systems with dispersed generation*, 2005, pp. 1–8.
[28] IEEE PES Distribution System Analysis Subcommittee Distribution Test Feeder Working Group, “Distribution Test Feeders,” https://ewh.ieee.org/soc/pes/dsacom/testfeeders/index.html, accessed: 2018-01-22.
[29] R. Rudnik, J.-Y. Le Boudec, A. Bernstein, L. Reyes-Chamorro, and M. Paolone, “Handling Large Power Steps in Real-Time Microgrid Control Via Explicit Power Setpoints,” in *PowerTech, 2017 IEEE Manchester.* IEEE, 2017, pp. 1–6.
[30] IEEE Power and Energy Society, “IEEE Guide for Voltage Sag Indices,” 2014.
[31] W. Saab, R. Rudnik, L. E. Reyes Chamorro, J.-Y. Le Boudec, and M. Paolone, “Robust Real-Time Control of Power Grids in the Presence of Communication Network Non-Idealities,” in *2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, 2018.