T-RECS: A Software Testbed for Multi-Agent Real-Time Control of Electric Grids

Jagdish Achara, Maaz Mohiuddin, Wajeb Saab, Roman Rudnik, and Jean-Yves Le Boudec
School of Computer Science and Communication Systems
École Polytechnique Fédérale de Lausanne, Switzerland
{firstname.lastname}@epfl.ch

***Abstract*—Multiple software agents can be used to perform the real-time control of electrical grids. The control performance of such solutions is influenced by software non-idealities such as crashes and delays of the software agents, and message losses and delays due to the underlying communication network. To study the effect of these non-idealities on control systems, we present an open-source software testbed, named T-RECS. It uses software containers to test existing software without modification. The communication network among the software containers is emulated using Mininet framework, which allows for real packets being exchanged. The electric resources in the grid are simulated using state-of-the-art models, whereas the grid itself is modeled in the phasor domain. As control agents are run as is and message exchanges are emulated, T-RECS accurately captures the real-world properties of the control framework. We demonstrate the working of T-RECS with the Commelec control framework and show the effect of network non-idealities on the control performance. We make a beta version available.**

I. Introduction

Real-time control systems (RTCSs) for electric grids [1], [2] use multiple software agents that exchange messages to perform the control actions. The performance of such multi-agent RTCSs is affected by delay and crash faults [3] in the software agents, and by losses and delays due to the underlying communication network. In [4], a delay fault in the energy management system resulted in a cascade of events that led to the failure of the electric grid in Northeast America. As similar incidents are more likely to happen in agent-based control of electrical grids, there is a need for extensive testing of control software under various ideal- and real-world scenarios before their deployment in the grid.

To this end, we propose T-RECS, a software testbed for multi-agent real-time control of electric systems. It targets developers of agent-based real-time control software for electric grids and is designed to test such control software without requiring any modification to it. T-RECS uses software containers to run each software agent in a virtual environment. Not having to change the control software improves the usability of T-RECS, compared to existing solutions [5], [6], [7], [8], [9]. Furthermore, the virtual environment can be used to inject faults such as crash of a controller or to study effects of software non-idealities on the control performance.

In T-RECS, the network between the software agents is emulated using the Mininet framework [10], wherein real packets are exchanged using TCP/IP networking, as opposed to simulation of the communication network done by existing testbeds [6], [9]. This approach enables T-RECS to recreate a wide-range of real-world scenarios by choosing different communication bandwidths, losses and delays. Moreover, the effect of more complex network non-idealities such as network congestion and router failures on control performance of software agents can also be studied. T-RECS is light-weight, compared to running software agents in virtual machines, because Mininet uses process-based virtualization to run hosts.

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

T-RECS models the grid using the three-phase nodal-admittance matrix (Y-matrix) representation. The impact of switching harmonics is not simulated and the frequency is imposed by a single resource. The evolution of the grid is tracked through complex voltage phasors at each bus. These phasors are obtained by performing a load-flow when there is a change in the grid state. Electric resources, such as battery, load, and photo-voltaic (PV) panels, are simulated.

In Section II, we compare T-RECS with existing testbeds, and we present the detailed design of T-RECS in Section III. The implementation of T-RECS is currently in progress but we make available a beta version (https://smartgrid.epfl.ch). Using this version, in Section IV, we run the Commelec control framework with the objective of following a dispatch plan. We show that the dispatch plan is precisely followed in the absence of communication network losses, whereas in the presence of losses, the control framework cannot drive the microgrid to follow the given dispatch plan. T-RECS is currently used to test and analyze the deployment process of Commelec in Rolle, Switzerland, among other places. It is co-developed and used with this control framework.

II. Related Work

Various testbeds or co-simulation frameworks [5], [6], [7], [8], [9], [11], [12] are proposed for control of electric grids. We discuss and compare our proposed testbed with the existing ones, on three different axes: (1) can we easily and accurately study the effect of communication networks on control systems, (2) can we run the control software without requiring the modeling of control systems or modifications to the code, and (3) are they scalable, portable, and distributable.

To study the effect of communication network on control systems, [9] simulates the communication network with OPNET Modeler, whereas in [6], the authors model the low-level network contingencies, e.g., delayed, disordered, dropped, and distorted information flows. T-RECS, on the contrary, emulates the communication network using Mininet. As message exchanges are emulated in T-RECS, it accurately captures the real-time properties of the control protocol. In [7], [8], communication network is neither modeled/simulated nor emulated. In these works, agents talk to each other via inter-process communication and hence, it is not possible to study the effect of network parameters on the control systems. Communication network is a main source of non-idealities in *real-time* multi-agent control systems, and not being able to study its effect is considered a major limitation.

T-RECS provides software containers to run the real control agents without requiring any modifications to them, whereas other testbeds require either modeling of agents [5], [6], [7], [8] or the development of agent software with specific frameworks such as JADE [9]. Testbeds in [5], [11], [12] use physical computing and/or communication infrastructure. This makes these testbeds non-scalable, non-distributable, and non-portable. Moreover, users of such testbeds incur monetary costs. In case of physical communication infrastructure, users can only test the robustness/correctness of their control protocol with a given hardware configuration. As the authors of [11] also admit, 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. In fact, our testbed does exactly what the authors in [11] ask for: We emulate the network infrastructure and run different agents in multiple software containers separated by an emulated communication network. As a result, our testbed does not suffer from these limitations.

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 [13], [14], [15] 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 [13], [14], [16]. With regard to modeling electric resources, we use existing state-of-the-art models and we enable T-RECS users to plug in new models if needed. To simulate the electric grid in the phasor domain, we cannot use PyPower [16] as it supports only single-phase load flow. Also, the grid model provided by GridLAB-D [13] 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 [17].

III. T-RECS Design

As shown in Fig. 1, an RTCS for electric grids can be categorized into layers as follows:

1) Physical layer — consists of the electric grid and resources that are being controlled by upper layers.

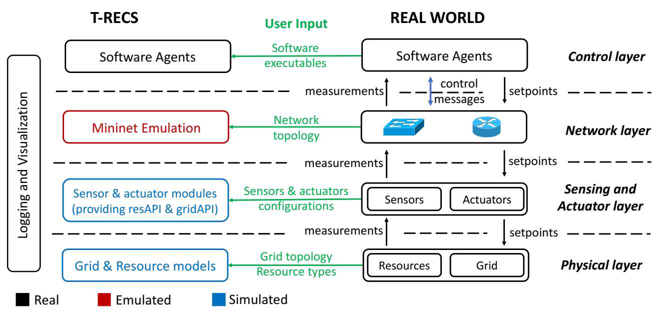

Fig. 1: Design and Architecture of T-RECS. RTCSs typically consists of four layers and the figure shows how these four layers are managed in T-RECS.

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 controllers that use the measurements from sensors to perform computations and output setpoints for the actuators.

Fig. 1 also shows how the different layers of the RTCS are realized in T-RECS. The physical layer is simulated using state-of-the-art models of grid and electric resources, as described in Section III-A. The sensing and actuation layer is simulated through two APIs, namely gridAPI and resAPI. These APIs provide an interface for reading and altering the state of the grid and resources, as described in Section III-B. The network layer is emulated using Mininet, with an exchange of real packets, as described in Section III-C. The control layer is realized in T-RECS without any modification to the software agents, as described in Section III-D. Using the unmodified software agents along with powerful network emulation, enables T-RECS to study the effects of software and network non-idealities on the control mechanism.

A. Physical Layer

To simulate the physical layer, T-RECS takes as an input the three-phase nodal-admittance matrix of the grid along with the location and types of resources. The frequency of the grid is either configured statically or received as an input from one of the resources. Currently, T-RECS provides three resource models: battery, variable load, and PV source. The battery is modeled using the two-time constant model [18]. The variable load and PV source are simulated by replaying a time-stamped trace of the power injections. A change in resource-state is reflected as a change in the state of the bus on which the resource is placed. When the state at any bus changes, the grid model performs a three-phase load flow by using the method in [17] to compute the voltage at each bus in the grid and the current in each line. In this way, the evolution of the grid is tracked throughout the execution.

B. Sensing and Actuation Layer

This layer consists of two APIs, the gridAPI and the resAPI. The gridAPI provides functions to get and set the voltage and power at each bus in the grid. The resAPI provides functions for reading and changing the state of the resource model, such as power injection by a battery. These APIs can be used to create specific sensors and actuators as required by the user's configuration. Currently, as a sensor model, T-RECS provides a phasor measurement unit (PMU) that periodically (every 20 ms in a 50 Hz grid) reads the voltage and current of the bus it is attached to and then, streams them as phasors by using the IEEE C37.118.2 format for synchrophasors [19]. The streamed messages are sent as real packets via the network layer.

C. Network Layer

The network layer consists of the communication infrastructure such as switches, routers, links, etc., that connect the sensors, actuators and control agents. T-RECS takes the topology of the network as input and reproduces it in a virtual environment by using the Mininet framework. This includes the interconnections between the agents, bandwidth and delay of the links, queuing discipline of the routers, etc. This network is used to send real messages between the agents where we can introduce packet drop, delay, or reordering depending on the desired network configuration.

D. Control Layer

The control layer is the same as in the real world. T-RECS takes the executables of the software agents as input and integrates them with the network topology by using software containers provided by Mininet. Just as in a real deployment, the unmodified executables receive messages from sensors, perform computations, and send setpoints to actuators. Thus, T-RECS recreates an environment in which the software agents can be executed and tested without modifying their code.

IV. Use Case Scenario

A. Setup

We have run Commelec [1], an RTCS for active distribution networks, using the publicly provided implementation of T-RECS. Commelec uses two types of software agents: grid agent (GA) is used to control an entire grid whereas resource agent (RA) is responsible for an electric resource. The GA receives measurements of the grid state from sensors every 20 ms. It communicates with RAs every 100 ms to send setpoints and to receive state of their respective resources.

To start the experiment, T-RECS requires as input the software agents, the resource types, the Y-matrix of the grid, and the network topology. Our setup consist of the CIGRÉ benchmark low-voltage microgrid with a battery and a PV resource. As software agents, it has one GA and two RAs. The network topology is such that all agents are in a different local area networks connected through a router.

The resources, namely battery and PV, are simulated using the resource models as described in Section III-A. The PV resource is uncontrollable and characterized by a rated power of 20 kW. We use a power trace from a real PV installation. The battery resource is controllable and its rated power is 20 kW. Additionally, there are five PMUs that send the state of the grid to the GA every 20 ms, as described in Section III-B. T-RECS creates four Mininet hosts (software containers) that run: (1) the GA, (2) the battery RA and the battery model, (3) the PV RA and the PV model, (4) the grid model and sensors. The inter-host communication happens through the network layer (Section III-C). The intra-host communication between the RAs and the resources, the grid and the PMUs happens through the resAPI and the gridAPI, respectively.

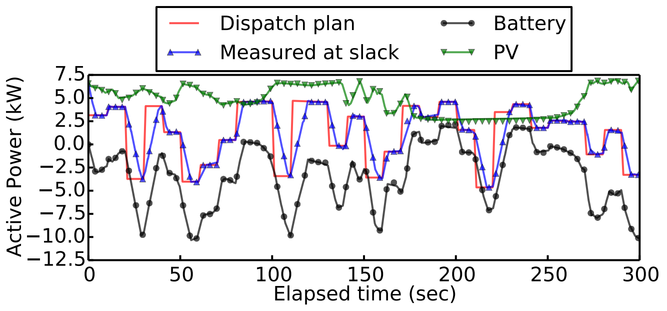

Fig. 2: Tracking the given dispatch plan in the presence of variable PV generation with Commelec. Positive power represents production.

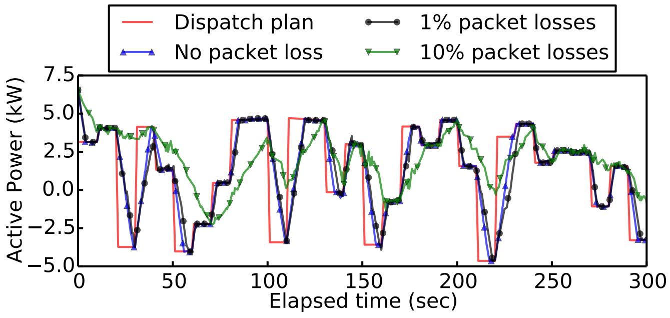

Fig. 3: Effect of network losses on Commelec's ability to track a given dispatch plan. Positive power represents production.

B. Goal

We illustrate the use of T-RECS on the scenario of microgrid dispatchability, by which a microgrid operator has to follow a *dispatch plan* computed day-ahead that specifies the grid prosumption by intervals of, say, five minutes. The goal of Commelec is to follow the said plan as accurately as possible in the presence of intermittent PV generation and to maintain the grid in a feasible state. The maximum tolerable energy mismatch is taken as 50 kWh in a day, which amounts to 0.17 kWh in five minutes, where energy mismatch is defined as the cumulative absolute difference between the dispatch plan and the power prosumed by the grid.

We test how communication network non-idealities between the GA and RAs influence the ability of software agents to follow the dispatch plan. Specifically, we are interested in the percentage of packet loss Commelec can bear while satisfying the 0.17 kWh limit for a duration of five minutes.

C. Results

Fig. 2 shows the dispatch plan and the actual power measured at the slack bus for a duration of five minutes. To track

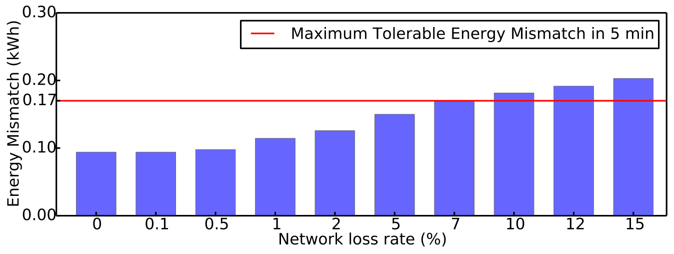

Fig. 4: Energy mismatch between the dispatch plan and the that provided by Commelec during a period of five minutes, as a function of network losses.

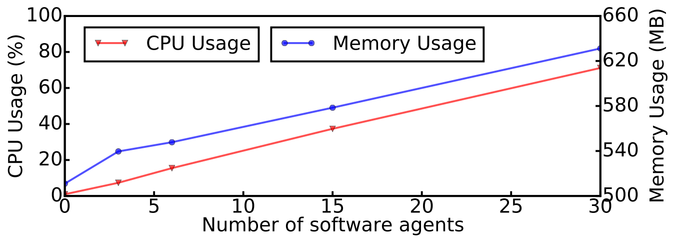

Fig. 5: CPU and memory usage of T-RECS, on a virtual machine with 2 GHz single-core CPU and 1 GB RAM, as a function of number of software agents.

this dispatch plan, the GA accommodates the variations in PV production by changing the power injected by the battery.

Fig. 3 shows the effect of packet losses in the communication between the GA and RAs on the tracking performance. While the requested power is tracked with high accuracy in the absence of packet losses, we see a slight deviation from the dispatch plan with 1% packet loss. This deviation is more profound with a higher loss rate of 10%.

To find the network loss rate at which the constraint of maximum tolerable energy mismatch of 0.17 kWh (Section IV-B) is violated, we vary the network loss rate and measure the energy mismatch. We find that Commelec can follow the dispatch plan with acceptable mismatch up to a packet loss of 7% between the GA and the RAs.

D. *Performance Evaluation*

T-RECS enables testing of wide-range of RTCSs for electric grids without the need for special hardware. In order to demonstrate its usability with large RTCSs, we evaluate the CPU and memory usage of T-RECS with varying number of software agents. For this, we run it in an Lubuntu virtual machine with a single-core 2 GHz CPU and 1 GB RAM, and increase the number of software agents in batches of three (one GA and two RAs).

Fig. 5 shows how the usage scales with the software agents. We find that the memory footprint of T-RECS increases at a rate of 5 MB per software agent, whereas the CPU usage increases at a rate of 2% per software agent. We see that, while running in a modest virtual machine, T-RECS can support up to 30 software agents, including 10 GAs and 20 RAs, without any performance penalty. To put this in perspective, a typical microgrid is controlled with one GA and about five RAs. Hence, we conclude that a control-software developer can use T-RECS to simulate large grids on a single computer.

V. Conclusion and Future Work

We presented T-RECS, a software testbed designed for testing and validating multi-agent real-time control software for electric grids. It supports use of the control software without any modification and emulates the communication network that enables real packets being exchanged. These design choices make it suitable for studying the effect of software and network non-idealities on the control performance.

Using a beta version of T-RECS, which we made available, we study the effect of network losses on the performance of the Commelec control protocol. In the future, we will validate T-RECS by comparing the results of the control performance against those of a real-world control in our on-campus microgrid. T-RECS is open-source and can be extended to include a wide-range of electric resource models. Currently, it is being used in co-development of Commelec, and will be used for testing and analysis of real-world deployment of Commelec.

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*, vol. 125, pp. 254–264, 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] M. Mohiuddin, W. Saab, S. Bliudze, and J.-Y. Le Boudec, "Axo: Masking Delay Faults in Real-Time Control Systems," in *Industrial Electronics Society, IECON 2016 - 42nd Annual IEEE Conference*, 2016.

[4] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou *et al.*, "Causes of the 2003 Major Grid Blackouts in North America and Europe and Recommended Means to Improve System Dynamic Performance," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1922–1928, 2005.

[5] F. Maturana, R. Staron, K. Loparo, and D. Carnahan, "Agent-Based Testbed Simulator for Power Grid Modeling and Control," in *Energytech, 2012 IEEE*. IEEE, 2012, pp. 1–6.

[6] 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.

[7] A. Ravichandran, "Software-Defined MicroGrid Testbed for Energy Management," *Master Thesis*, 2011.

[8] "Mosaik," https://mosaik.offis.de/, [Accessed 04-May-2017].

[9] 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*. IEEE, 2013, pp. 1–5.

[10] "Mininet," http://mininet.org/, [Accessed 04-May-2017].

[11] 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*. IEEE, 2013, pp. 1–5.

[12] 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*. IEEE, 2010, pp. 1–6.

[13] "GridLAB-D," http://www.gridlabd.org/, [Accessed 04-May-2017].

[14] "Modelica," https://www.modelica.org/, [Accessed 04-May-2017].

[15] "OpenDSS," http://smartgrid.epri.com/SimulationTool.aspx.

[16] "PyPower," https://pypi.python.org/pypi/PYPOWER.

[17] 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.

[18] M. Bahramipanah, D. Torregrossa, R. Cherkaoui, and M. Paolone, "Enhanced Electrical Model of Lithium-Based Batteries Accounting the Charge Redistribution Effect," in *Power Systems Computation Conference (PSCC), 2014*. IEEE, 2014, pp. 1–8.

[19] K. E. Martin, G. Benmouyal, M. Adamiak, M. Begovic, R. Burnett, K. Carr, A. Cobb, J. Kusters, S. Horowitz, G. Jensen *et al.*, "IEEE Standard for Synchrophasors for Power Systems," *IEEE Transactions on Power Delivery*, vol. 13, no. 1, pp. 73–77, 1998.