Large-Scale Co-Simulation of Power Grid and Communication Network Models with Software in the Loop

Eric MSP Veith[1], Jawad Kazmi[2], and Stephan Balduin[1]

[1] OFFIS e.V.
Oldenburg, Germany
Email: {veith,balduin}@offis.de

[2] Austrian Institute of Technology (AIT)
Vienna, Austria
Email: jawad.kazmi@ait.ac.at

***Abstract*—Power grids are transitioning from an infrastructure model based on reactive electronics towards a smart grid that features complex software stacks with intelligent, pro-active and decentralized control. As the power grid infrastructure becomes a platform for software, the need for a reliable roll-out of software updates on a large scale becomes evident. In order to validate resilient large-scale software roll-out protocols, corresponding test beds are needed, which mirror not only Information and Communication Technology (ICT) networks, but also include the actual software being deployed, and show the interaction between the power grid and the ICT network during the roll-out, and especially during roll-out failures. In this paper, we describe the design implementation of a large-scale co-simulation test bed that combines ICT and power grid simulators. We pay specific attention to the details of integrating containerized software in the simulation loop.**

***Keywords*—Co-Simulation; Smart Grid; Power Grid Information and Communication Technology; Software in the Loop; Linux Development**

I. Introduction

The transition of the power grid to the smart grid is happening on a large scale. From the first introduction of the term *smart grid* [1], assets in the power grid have evolved into software platforms that feature a vast array of services. Transformers have become tools in asset management [2], while Multi Agent Systems (MAS) represent nodes in the power grid [3].

The numerous use and business cases enabled by this kind of infrastructure obviously require special attention to the software stack deployed on these devices. The life and, hence, innovation cycle in the power grid of 30–60 years that was dominating in the traditional power grid does not hold anymore. As the evolution of energy systems to Cyber-Physical Systems (CPS) based on ICT technologies has happened, so has, with increased complexity, risen the inherent risk of the overall system [4]: Power grids have become a target in terms of cyber security, as proven by the attacks on the Ukrainian power grid between 2015 and 2017 [5][6]. Specifically, software solutions based on Artificial Intelligence (AI) technologies have been regarded as major factors in technical debt, causing frequent updates to be made [7][8].

In a recent literature survey, we noted that the emerging smart grid yields numerous attack vectors, many stemming from the inclusion of ICT, AI technologies or tight market integration [9]. A major research gap exists in AI-based analysis of complex CPS, i.e., the combination of power grid and ICT. Specifically, the interaction of both components has hitherto seldom been discussed. On this basis, Adversarial Resilience Learning (ARL) offers an approach *based* on AI to explore any CPS without domain-specific knowledge and find weaknesses in its configuration [10][11]. This can very well be applied to software roll-out and update processes, too, provided a test bed for this exists. Software update roll-outs are, for a simulation testbed, a special case, as they require the actual software to be deployed within the simulation in order to assess the impact of the roll out.

To this end, we present a co-simulation approach that features power grid, ICT, and software-in-the-loop simulators. We will detail the specific development to facilitate a software-in-the-loop simulation on a large scale. The rest of this paper is structured as follows: Section II provides context for this work. We will detail possible, generalized models for our testbed in Section III. We then offer insights into the ICT co-simulation in Section IV, which accounts for a major portion of this paper. We discuss the overall development in Section V, and conclude with pointers to future work in Section VI.

II. Related Work

Simulators for specific domains exist for many years now, drawing from the standard rationale that, once the system and the interaction of its components become too complex to describe them in terms of formulæ and automatons, a simulation to assert assumptions is in order. For each individual domain, a sound selection of simulators exist, such as *pandapower* by Thurner *et al.* [12] and *SIMONA* by Kittl *et al.* [13] for power grids, or OMNeT++ by Varga *et al.* [14] for ICT simulations.

However, to witness effects of the two domains interacting with each other, none of the two is fully suited. Specifically when smart grid messaging is considered—which is crucial to optimization protocols, such as COHDA [15][16] or Winzent [17]—, this part of the simulation becomes crucial. Previous simulation environments for testing smart grid messaging have focused on other parts of the problem, such as using a Geospatial Information System (GIS) layer to model the feed-in of renewable energy sources [18]. Since a modern power grid has, essentially, become a CPS, ICT has become an integral part. Therefore, the interaction between these two complex domains has become paramount for our research.

 ISBN: 978-1-61208-788-7

The combination of two or more simulators from different domains is facilitated through *co-simulation*. A co-simulator provides an infrastructure to schedule, synchronize different simulators, and enable data exchange between model instances run by the different simulators. One solution is provided by the *mosaik* co-simulator [19]—the one, in fact, used to develop the test bed presented in this paper—; other approaches to co-simulation are employed, e.g., by *OpSim* [20], or *PTOLEMY II* [21]. A co-simulation of power grid and ICT has been described by different authors using different pieces of software [22, 23, 24], but without taking the question of software roll-out into account.

This paper introduces a smart grid software roll-out testbed, based on the idea discussed by Kintzler *et al.* [25]. It details the reasoning behind using a Software-In-the-Loop (SIL) approach—namely, that the software being rolled out itself is complex enough that an approximation through models is not feasible. If the subject to the experiment, i.e., the software, is abstracted away, the result of the roll-out protocol cannot be validated.

SIL co-simulation is not new. Pieper *et al.* [26] use the SIL technique to validate railway controllers; real-time SIL co-simulation in the smart grid for performance measurements is done by, e.g., Bian *et al.* [27]. The OMNeT++ simulation environment offers facilities for Hardware in the Loop (HIL) integration [28][29]. However, when the software itself is subject to change in a co-simulation/SIL scenario, an extension needs to be developed to allow the integration of changing, virtualized software containers. This research gap is addressed by our solution.

III. Modelling Power Grid and ICT

Figure 1 shows the data exchange schema of our co-simulation approach, including all software bridges that connect the simulation with the SIL part. The following sections will refer to the schema when locating individual pieces of software.

A. Power Grid Reference Model

To capture the complex dynamics of and possible effects caused by the large-scale roll-out of smart power devices, a realistic and complete model of the power system model is a necessity. The model needs to be detailed that could later be simulated along with the other related components. It is, therefore, important to choose a modeling and simulation tool that fulfills these requirements. There are many good power system modeling and simulation tools such as *pandapower* by Thurner *et al.* [12] and *SIMONA* by Kittl *et al.* [13] for power grids. After a survey and discussion, *DIgSILENT PowerFactory* was selected for the power system modeling as it meets the selection criteria better than the other available tools. It provides detailed and fine-grained modeling and simulation of many aspects of the power system. The model needs to be simulated in a co-simulation setting and allow to receive the set-point and measurements from and to the coupled sub-system, as shown in Figure 2.

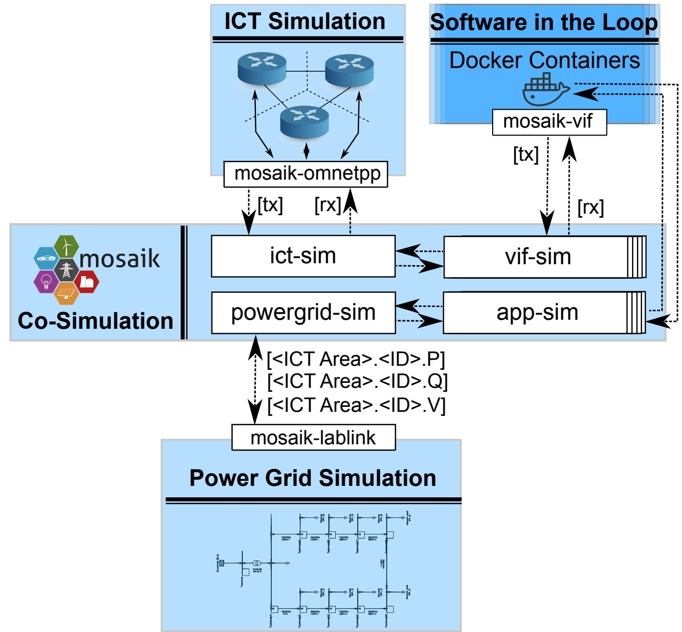

Figure 1. Data Exchange Schema of the Co-Simulation

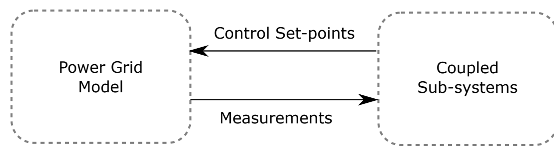

Figure 2. Power grid (co-)simulation design rationale

DIgSILENT PowerFactory is a sophisticated, highly specialized, flexible, and extendable platform for power system modeling and simulation. It supports fine-grained power system modeling and simulation through a combination of both graphical and scripting based methods for almost all the major areas of the power system, including generation, transmission, distribution, etc. There is a large library of models available that can be extended by writing custom components using the DIgSILENT Simulation Language (DSL). For a dynamic simulation of the power system, the tool provides many functionalities including load and power flow calculations, reliability and contingency analysis, and many more. The tool also supports Application Programming Interfaces (APIs) that can be used to communicate with other simulators. It further supports the automation using DIgSILENT Programming Language (DPL).

AIT Lablink is a multipurpose, highly efficient, and distributed middleware for coupling both hardware and software components in a co-simulation. It is used for coupling the individual components and thus makes the power grid simulation flexible and extendable. *AIT Lablink* provides interfaces, simulation control, and data exchange capabilities. By using it, it is possible to do either simulations or an emulation and it has been used extensively [30, 31, 32] for performing various validation and verification activities. A large set of hardware and software components is already supported by bridges that make extending the testbed very easy.

 ISBN: 978-1-61208-788-7

In the present setup, depicted in Figure 1, *AIT Lablink* provides a message bus that the participating components (software/hardware) can connect to through a bridge. This bridge facilitates the data exchange and simulation control, including the synchronization. The bridge and the participating component have a one-to-one correspondence. Two important such bridges are the *DIgSILENT PowerFactory* and *mosaik* bridge. There are some other *AIT Lablink* system components like *Synchronizer*, *Simulation Manager*, etc., that provide useful services, but are excluded here for brevity, as they are part of every setup created with *AIT Lablink*.

As the co-simulation is managed by *mosaik*, *AIT Lablink* coordinates with *mosaik* for data exchange and synchronization of the simulation. All the data exchange requests received from the coupled systems through *mosaik* are forwarded to the respective component (*DIgSILENT PowerFactory* in this case), while simulation synchronization requests are forwarded to *Synchronizer* that takes care of running the co-simulation components in sync.

B. The Communications Infrastructure Model

The ICT model serves to provide a number of realistic network areas to test the software roll-out scenarios. It is independent from the test bed software, i.e., it was developed in parallel as part of the test bed, but can be used on its own, e.g., without software in the loop. It features a number of subnets, with each subnet area designating a certain characteristic network environment, such as a well-built fibre channel network or a spotty wireless area. To this end, it models an Autonomous System (AS) with routers and intra-AS traffic/routing. These subnets have real IPv4 addresses assigned, as the ICT model needs to process actual Internet Protocol (IP) traffic generated by the existing software. The ICT infrastructure network is fully contained in the class C subnet

`10.64.0.0/10` .

Table I contains the relevant subnet specifications for the areas that are described in the following paragraphs.

The reason for choosing this particular kind of subnet is its rather remarkable subnet range and the fact that `10.64.0.0/10` is seldom used as an IPv4 address space. This way, the ICT model does not collide with existing private, class C IPv4 addresses, such as those assigned by Virtual Private Network (VPN) software. This leaves room for 8192 subnets with 254 hosts each in every defined network. The `/24`-subnet should be the only network size, regardless of how many hosts are contained in it. Routers always get the lowest IP addresses assigned, i.e., `.1`, `.2`, `.3`, etc., before the first hosts are added.

The test bed consists of 3 areas, which differ by their Quality of Service (QoS) parameters. We assume that most visible traffic we consider is either based on the Transmission Control Protocol (TCP) or employs similar mechanisms. This especially means that the protocol features a retransmission algorithm. Since packet loss can be caused either by a low-quality link or by network congestion, delay (denoted by d) is, for the purposes of this test bed, the most describing parameter of a link (besides its data rate).

TABLE I. ICT MODEL SUBNET SPECIFICATION

Network	10.64.0.0/10
Network Range	10.64.0.1 – 10.127.255.255
Dedicated Network	10.64.0.0/12
Shared Links Network	10.80.0.0/12
High-Impairment Network	10.96.0.0/12
Misc./Unallocated	10.112.0.0/12

The first area is the *Dedicated Network Area*. The underlying assumption is that of the best possible infrastructure, where a grid operator has deployed dedicated ICT cabling. Thus, the network is of high quality. This does not only create a realistic scenario, but also serves as the test case for the whole simulation infrastructure. The assumed nominal data rate is $1\,\mathrm{GBit/s}$; the delay is modeled stochastically per packet as:

$$d \sim 10 + 50 \cdot f_\lambda(x, 1)\ [\mathrm{ms}]\ . \tag{1}$$

The function $f_\lambda(x, 1)$ denotes the drawing of a random number from an exponential distribution.

The second designated area is the *Shared Links Area*. Here, we assume that a grid operator uses the public infrastructure, such as Internet-facing connections. While we can assume that the necessary security precautions are taken (e.g., by deploying a VPN solution and generally encrypting traffic), other traffic interferes with the QoS of the update traffic we examine. I.e., we can assume that there are occasional packet drops due to congestion. As such, we model the delay as the drawing of a random number from a normal distribution:

$$d \sim \mathcal{N}(250; 20)\ [\mathrm{ms}]\ . \tag{2}$$

The area is well suited for variable-situation test cases. The link data rate is still good, being at $1\,\mathrm{GBit/s}$ nominally.

The extreme end of the spectrum is modeled by the *High-Impairment Area*. It features low-datarate links (configurable from $50\,\mathrm{kBit/s}$ up to $100\,\mathrm{MBit/s}$ with frequent congestion. This area also models the deployment of wireless connections, such as 4G/CDMA 450 or similar technologies. It is characteristic for an area where the development of the infrastructure was hindered by, e.g., existing building situations, harsh terrain, cost constraints, etc. As such, there are frequent packet drops and even connection drops. The delay is modeled as:

$$d \sim \mathcal{U}[100; \infty]\ [\mathrm{ms}]\ , \tag{3}$$

i.e., the drawing of a random number from a uniform distribution with the interval $[100; \infty]$ (inclusive). A delay of infinity means the link is broken.

IV. ICT AND SOFTWARE-IN-THE-LOOP DEVELOPMENT

A. Data Exchange Flows

All simulators appear in the system twice, as Figure 1 suggests. For each simulator process—like the ICT simulation, the power grid simulation, or each containerized SIL—there exists also a representation as an entity object in *mosaik*. This object is responsible for connection data exchange channels as well as communicating with the simulator processes. Overall, there are at least four simulator processes with corresponding entity objects.

The **ICT Simulation** is responsible to run the communication network simulation. Some nodes are also providing an interface to the co-simulation as a bridge between the ICT simulation and the SIL components. I.e., it also injects real IP packets from the containerized applications into the simulation environment and reads packets received from other simulated nodes and transfers them back to the software containers. In Figure 1, it is represented as the `ict-sim` object during a *mosaik* run.

The **Power Grid Simulation** is responsible for calculating load flows and line loads. It receives data through *mosaik* from the actual applications. For example, an application representing an intelligent substation would appear in the power grid simulation as substation; the substation software would receive readings from the power grid simulation and issue setpoints to it. This data flow is depicted in Figure 1 as an exchange between the SIL container, the `app-sim` entity objects, and the `powergrid-sim`.

The **Application Simulators** each represent one containerized piece of software. They are not simulators in the strict sense, but the SIL component. The simulator is responsible for starting and stopping the containers gracefully, and also for setting and collecting data coming from the co-simulation or going to another simulator. Each application container has its own application simulator and, hence, a corresponding `app-sim` entity object.

Application logic will dictate communication with other containers. For example, a distributed real power schedule optimization heuristic like *Winzent* works on MAS basis, and, therefore, requires communication with other applications. I.e., the application containers are the logical connection between ICT and the power grid simulation. For the roll-out scenario, it is not sensible to modify the application software to be part of the ICT simulation directly. Thus, we leave the applications in the container undisturbed, and deploy a virtual network interface to connect the applications to the ICT simulation. This virtual network interface, called *vif* for short, also has a corresponding `mosaik-vif` entity object in the *mosaik* process. Thus, in our scenario, there exist exactly as many `app-sim` objects as there are `vif-sim` entities.

The connection between the virtual interfaces and the corresponding nodes in the ICT simulation is done in *mosaik*. Any ICT-related simulator offers at least one model that represents the respective node. These models have exactly two attributes, `rx` (“receive”) and `tx` (“transmit”). *Attribute* is a *mosaik* term that designates a data exchange interface for a simulator. Referring back to Figure 1, we see that each `vif-sim` has these two attributes. The `rx` attributes always receive data from *mosaik*, whereas `tx` attributes transmit data to *mosaik*. The ICT simulation has more than one `tx`/`rx` pair: one for every node in the simulation for which a corresponding application container exists.

```
someapp_vif_entity = \
    someapp_vif_simulator.vif()
someapp_ict_entity = next(
    x for x in ict_model.children
    if x.eid == \
        'SimulatedNetwork/SomeApp/app-0')

world.connect(someapp_vif_entity,
              someapp_ict_entity,
              ('tx', 'rx'))
world.connect(someapp_ict_entity,
              someapp_vif_entity,
              ('tx', 'rx'),
              time_shifted=True,
              initial_data={'tx': None})
```

Figure 3. Example of a connection between an entity in the ICT simulation and the SIL container

An example of a connection in *mosaik* can be achieved as shown in Figure 3.

B. Virtual Network Adapter & Packet Injection

The code example in the previous section also shows how the hierarchical addressing for entities in simulators in *mosaik* is done. The `vif` entities here denote a SIL entity, i.e., a container with a unmodified piece of software. Each entity denotes two software instances: first, the virtual interface *vif* that exists in a container, and, second, the *vif-sim* that translates data between the container’s networking stack and the *mosaik* co-simulation protocol. These two pieces of software must exists separately as to avoid timing issues. The startup behavior of the container and its software cannot be observed by the simulator; there exists a natural delay between launching the container and being actually able to integrate it in the simulation run, i.e., the containerized application sending and being able to receive packets. Since multiple containers will normally be started, there is a time gap between the first container’s application being online and the last one being ready. As SIL implies no modification on the software, we cannot signal these applications to hold until the simulation is ready to be started; hence, each *vif* must transparently buffer all data until the *vif-sim* is launched by the co-simulator.

In general, the *vif* must act as if it was just a standard network device. For this reason, the Linux kernel’s tun/tap device driver was chosen. It establishes a *tun device* that appears as `tun0` (or any higher index number) in the output of `ip address show`, can carry IPv4 and IPv6 addresses, and can be the subject of the default route. Moreover, the tun device needs no gateway address, i.e., `ip route add default dev tun0` without a `via` stanza is possible. This way, the tun device transparently receives all traffic from the application, which does not need to be changed; the kernel delivers all this traffic to a user space application, i.e., the *vif*. Injecting traffic is done the same way.

Since the userspace application needs to transmit this data to the co-simulator, a second, specific rule for the IP address

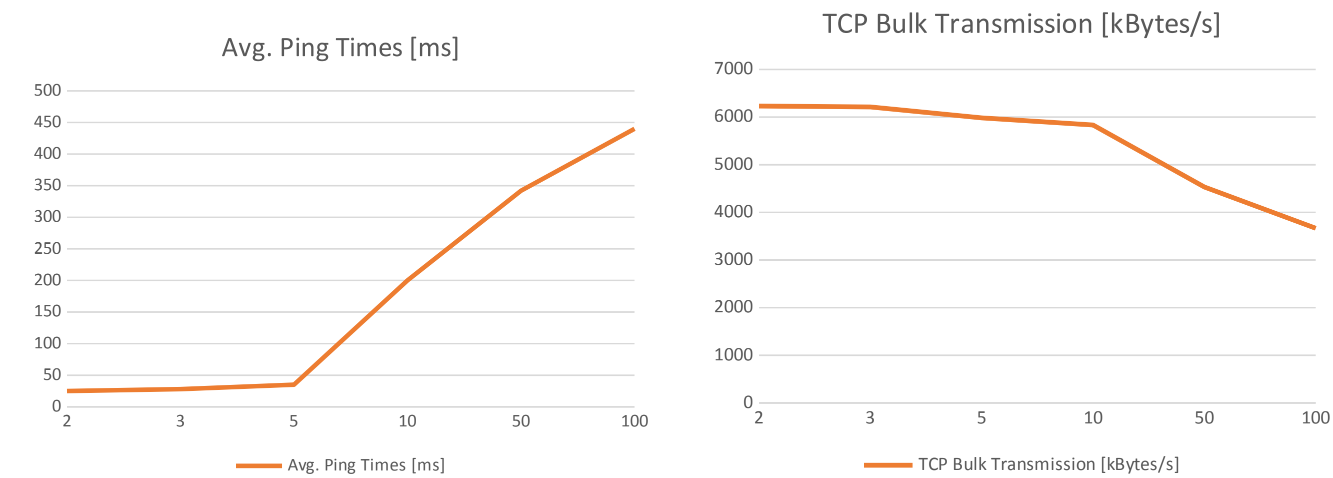

Figure 4. Experimental Measurements of Average Ping Times and Throughput

of the *mosaik* instance is added, so that traffic between the simulator and the *vif* still flows via the standard `eth0` device.

As the tun device now tunnels all regular traffic, the communication protocol between *vif* and *vif-sim* needs to be carefully chosen as to avoid race conditions. Tunneling TCP in TCP is discouraged, as two nested congestion control algorithms interfere with each other, creating cascading time lags that can stall the application, known as *TCP Meltdown* [33]. Since User Datagram Protocol (UDP) needs to be chosen, the external address of the container is not known to the co-simulator, which hinders the ICT simulation from injecting data first before any data is received from the container (and, thus, the container's address becomes known). We solve this be simply sending a burst of zero UDP 'hello' packets to announce the container.

Each byte of packet data received by the *vif* is immediately transmitted to the *vif-sim*, which takes care of assembling the packets. Assuming that the first transmission will contain the start of an IP packet and no intermediate packet data will be lost between container, *vif* and *vif-sim*, the *vif-sim* reads the packet length from the IP header field in order to assemble whole packets. These packets are then encoded in Base64 format so that they can be transmitted to *mosaik* via *mosaik's* Java-Script Object Notation (JSON) communication protocol.

The *vif-sim* as well as the *mosaik-OMNeT++* adapter are single-threaded, but use a cooperative, asynchronous I/O multitasking pattern to handle the communication flow. Under the assumption that these applications are I/O-heavy, but not computationally demanding, the single-threaded, multi-process paradigm where much time is spent in the kernel's I/O space suggests itself [34].

V. Discussion

As the general feasibility of co-simulation has already been established, we focused prominently on the ICT SIL simulations. For this, we have set up a co-simulation with a number of containers in which the *iPerf3* [35] application was running. We have deployed pairs of clients and servers so that an iPerf client can send and receive data from a dedicated iPerf server container. We used this set up to test both the average round-trip times (i.e., ping echo request/echo reply timings) and TCP bulk transfer speeds. All data was routed through the simulated ICT environment, so that the flow of data was as follows: *vif—vif-sim—mosaik—OMNeT++—mosaik—vim-sim'—vif'*. The simulated ICT environment does not impose additional artificial delays in its network model.

Figure 4 shows the behavior for both metrics given a rising number of nodes. Each data point represents a different number of nodes and the average over 100 repeated simulation runs. Delays rise sharply as the number of nodes rises, but not exponentially. With ping times in the area of $23\,\mathrm{ms}$ to $447\,\mathrm{ms}$, we assume that applications that do not rely on real-time or, in general, low-latency communication can be accommodated by this SIL setup. However, the bulk throughput rate between $6102\,\mathrm{kB/s}$ to $3654\,\mathrm{kB/s}$ is far below a characteristic data rate normally achieved by standard Ethernet connections.

We have investigated the reason for the low data rate and have identified three major points. First, *mosaik* currently uses non-compressed JSON messages in a request-reply pattern for data exchange with out-of-process co-simulators. As both the *vif-sim* and the *mosaik-OMNeT++* adapter are written in C++, an additional network round-trip is introduced, even if the simulation runs locally. In addition, *mosaik's* single-threaded request-response communication pattern with its associated simulators means that dependent simulators expect a delay when other simulators are being stepped or queried for data.

Furthermore, *mosaik* has currently no facilities to allow simulators to signal the necessity to be stepped; simulator control is completely in the hands of *mosaik*. This means that *mosaik* must poll all *vif-sims* as often as possible, since the co-simulator has no other way of knowing when data is available from a SIL container. In contrast to the ICT simulation, data from applications arrives in a non-deterministic way. In general, we have observed delays in message processing stemming from the context switches between kernel space and user space that frequently occur as data from the containerized applications travel through several network stacks.

Moreover, we currently launch one *vim-sim* process per container, as this is the easiest way from a software engineering organization perspective. However, this means a separate TCP connection per container, a new process, and a new data stream. We, therefore, plan to implement a multiplexing architecture in the *vim-sim* part in order to reduce the number of processes, and, hence, reduce task and context switches.

We believe that this approach offers great flexibility and ease in modelling ICT networks with SIL. As the development of *mosaik* is open source and already aimed at providing higher throughputs and lower delay in the communication with external simulators—e.g., a ZeroMQ implementation to replace the socket API already exists—, and the co-simulator is extended to allow for event-discrete, non-deterministic simulators as they exist in this scenario, we see an increase in the throughput in the near future.

VI. Conclusion & Future Work

In this paper, we have detailed requirements and issues encountered in a SIL co-simulation of software roll-outs in the power grid. We have shown how an interaction of ICT and the power grid can be simulated and how complete containerized software stacks can be embedded into this co-simulation.

In the future, we expect optimizations on implementation level, e.g., more efficient transports and serialization techniques, as well as implementing zero-copy primitives to reduce the number of copy operations and context switches. On a broader research perspective, we expect that abstracting parts of the system through surrogate models [36][37] will provide for a way to simulate large-scale roll-out procedures.

Acknowledgements

The presented work is conducted in the framework of the joint programming initiative ERA-Net Smart Grids Plus, with support from the European Union's Horizon 2020 research and innovation programme. On national level, the work was funded and supported by the Austrian Climate and Energy Fund (KLIEN, ref. 857570), by German BMWi (FKZ 0350012A), and by the Swedish Energy Agency (Project number 42794-1).

References

[1] S. F. Bush, *Smart Grid — Communication-enabled Intelligence for the Electric Power Grid*, 1st, ser. Wiley IEEE Series. Chichester, United Kingdom: John Wiley & Sons, 2014, ISBN: 978-1-119-97580-9.

[2] H. Ma, T. K. Saha, C. Ekanayake, and D. Martin, "Smart transformer for smart grid—intelligent framework and techniques for power transformer asset management", *IEEE Transactions on Smart Grid*, vol. 6, no. 2, pp. 1026–1034, Mar. 2015, ISSN: 1949-3061. DOI: 10.1109/TSG.2014.2384501.

[3] E. M. Veith, *Universal Smart Grid Agent for Distributed Power Generation Management*. Logos Verlag Berlin GmbH, 2017.

[4] O. Hanseth and C. Ciborra, *Risk, complexity and ICT*. Edward Elgar Publishing, 2007.

[5] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid", E-ISAC, Washington, DC, USA, Tech. Rep., Mar. 2016.

[6] A. Prentice, *Ukrainian banks, electricity firm hit by fresh cyber attack*, Jun. 2017.

[7] D. Sculley *et al.*, "Machine Learning: The High Interest Credit Card of Technical Debt", *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, pp. 1–9, 2014.

[8] D. Sculley *et al.*, "Hidden technical debt in machine learning systems", in *Advances in neural information processing systems*, 2015, pp. 2503–2511.

[9] E. Veith, L. Fischer, M. Tröschel, and A. Nieße, "Analyzing cyber-physical systems from the perspective of artificial intelligence", in *Proceedings of the 2019 International Conference on Artificial Intelligence, Robotics and Control*, ACM, 2019, pp. 85–95, ISBN: 978-1-4503-7671-6.

[10] L. Fischer, J.-M. Memmen, E. M. Veith, and M. Tröschel, "Adversarial resilience learning—towards systemic vulnerability analysis for large and complex systems", in *The Ninth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY 2019)*, vol. 9, 2019, pp. 24–32.

[11] E. M. Veith *et al.*, *Analyzing power grid, ict, and market without domain knowledge using distributed artificial intelligence*, 2020. arXiv: 2006.06074 [cs.CY].

[12] L. Thurner *et al.*, "Pandapower — an open-source python tool for convenient modeling, analysis, and optimization of electric power systems", *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, Nov. 2018, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2018.2829021.

[13] C. Kittl, J. Hiry, C. Wagner, C. Pfeiffer, C. Engels, and C. Rehtanz, "Large scale agent based simulation of distribution grid loading and its practical application", in *25th International Conference on Electricity Distribution (CIRED)*, Madrid, Spain: AIM, Jun. 2019, pp. 1–5.

[14] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment", in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08, Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–10, ISBN: 9789639799202.

[15] C. Hinrichs, S. Lehnhoff, and M. Sonnenschein, "A decentralized heuristic for multiple-choice combinatorial optimization problems", in *Operations Research Proceedings 2012*, Springer, 2014, pp. 297–302.

[16] A. Nieße, J. Bremer, and S. Lehnhoff, "On local minima in distributed energy scheduling.", in *FedCSIS Position Papers*, 2017, pp. 61–68.

[17] E. Veith, B. Steinbach, and J. Windeln, "A lightweight messaging protocol for smart grids", in *Proceedings of the Fifth International Conference on Emerging Network Intelligence (EMERGING 2013)*, Porto, Portugal, Sep. 2013, pp. 1–6.

[18] E. M. Veith, B. Steinbach, and J. Otten, "An open data-based discrete simulation environment for testing smart grid messaging", in *Proceedings of the Fourth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY 2014)*, ser. International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY), Chamonix, France: International Academy, Research, and Industry Association, Apr. 2014, pp. 7–12.

[19] C. Steinbrink *et al.*, "Cpes testing with mosaik: Co-simulation planning, execution and analysis", *Applied Sciences*, vol. 9, no. 5, p. 923, 2019.

[20] S. R. Drauz, C. Spalthoff, M. Würtenberg, T. M. Kneikse, and M. Braun, "A modular approach for co-simulations of integrated multi-energy systems: Coupling multi-energy grids in existing environments of grid planning & operation tools", in *2018 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, IEEE, 2018, pp. 1–6.

[21] J. Eker *et al.*, "Taming heterogeneity-the ptolemy approach", *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[22] K. Hopkinson, X. Wang, R. Giovanini, J. Thorp, K. Birman, and D. Coury, "Epochs: A platform for agent-based electric power and communication simulation built from commercial off-the-shelf components", *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 548–558, 2006.

[23] H. Lin, S. S. Veda, S. S. Shukla, L. Mili, and J. Thorp, "Geco: Global event-driven co-simulation framework for interconnected power system and communication network", *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1444–1456, 2012.

[24] B. M. Kelley, P. Top, S. G. Smith, C. S. Woodward, and L. Min, "A federated simulation toolkit for electric power grid and communication network co-simulation", in *2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, IEEE, 2015, pp. 1–6.

[25] F. Kintzler *et al.*, "Large scale rollout of smart grid services", in *2018 Global Internet of Things Summit (GIoTS)*, IEEE, 2018, pp. 1–7.

[26] T. Pieper and R. Obermaisser, "Distributed co-simulation for software-in-the-loop testing of networked railway systems", in *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, Jun. 2018, pp. 1–5. DOI: 10.1109/MECO.2018.8406023.

[27] D. Bian, M. Kuzlu, M. Pipattanasomporn, S. Rahman, and Y. Wu, "Real-time co-simulation platform using opal-rt and opnet for analyzing smart grid performance", in *2015 IEEE Power & Energy Society General Meeting*, IEEE, 2015, pp. 1–5.

[28] P. Wehner and D. Göhringer, "Internet of things simulation using omnet++ and hardware in the loop", in *Components and Services for IoT Platforms: Paving the Way for IoT Standards*, G. Keramidas, N. Voros, and M. Hübner, Eds. Cham: Springer International Publishing, 2017, pp. 77–87, ISBN: 978-3-319-42304-3. DOI: 10.1007/978-3-319-42304-3_4. [Online]. Available: https://doi.org/10.1007/978-3-319-42304-3_4.

[29] J. Kölsch, C. Heinz, S. Schumb, and C. Grimm, "Hardware-in-the-loop simulation for internet of things scenarios", in *2018 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, IEEE, 2018, pp. 1–6.

[30] I. Ahmad, J. H. Kazmi, M. Shahzad, P. Palensky, and W. Gawlik, "Co-simulation framework based on power system, ai and communication tools for evaluating smart grid applications", in *2015 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)*, 2015, pp. 1–6.

[31] J. H. Kazmi, A. Latif, I. Ahmad, P. Palensky, and W. Gawlik, "A flexible smart grid co-simulation environment for cyber-physical interdependence analysis", in *2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2016, pp. 1–6.

[32] M. Findrik, P. Smith, J. H. Kazmi, M. Faschang, and F. Kupzog, "Towards secure and resilient networked power distribution grids: Process and tool adoption", in *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2016, pp. 435–440.

[33] OpenVPN, *What is TCP meltdown?*, [Retrieved: 2020-07-08]. [Online]. Available: https://openvpn.net/faq/what-is-tcp-meltdown/.

[34] C. M. Kohlhoff, *Boost.asio*, [Retrieved: 2020-05-13]. [Online]. Available: https://www.boost.org/doc/libs/1_73_0/doc/html/boost_asio.html.

[35] J. Dugan *et al.*, *Iperf3*, [Retrieved: 2020-05-13]. [Online]. Available: https://iperf.fr.

[36] S. Balduin, "Surrogate models for composed simulation models in energy systems", *Energy Informatics*, vol. 1, no. 1, p. 30, 2018.

[37] S. Balduin, T. Westermann, and E. Puiutta, *Evaluating different machine learning techniques as surrogate for low voltage grids*, 2020. arXiv: 2006.12389 [eess.SP].