A Modern ICT Network Simulator for Co-Simulations in Smart Grid Applications

Fabian Niehaus[1], Bastian Fraune[2], Giacomo Gritzan[3] and Richard Sethmann[4]
FRI, Fakultät Elektrotechnik und Informatik, Bremen, Germany[*]
fri@hs-bremen.de[1]
bastian.fraune@hs-bremen.de[2]
giacomo.gritzan@hs-bremen.de[3]
sethmann@hs-bremen.de[4]

Abstract: The current transformation of power grids towards smart grids and the associated increase in the use of ICT technologies lead to an increased attack surface via the communication network. Holistic, multi-domain co-simulations can be used to evaluate the possibilities for, and impacts of, ICT-based attacks on these grids. This paper presents an ICT network simulator that can be used in co-simulation environments. It outlines requirements for this kind of simulation tool and provides insight into how these requirements can be met. In addition, co-simulation use cases for the simulator arising from different research projects are laid out. Finally, the simulator's functionalities in co-simulation environments are verified with a practical application example. This application shows that our simulator (rettij) is able to exchange data with other simulators through a co-simulation interface. It also demonstrates our simulators capabilities to perform real-world ICT attacks on realistic network topologies.

Keywords: ICT; Cyber Security; Co-Simulation; Network Simulator; Simulation; Smart Grid; Power Grid; Kubernetes; Docker

1. Introduction

Many power grids are currently undergoing a transformation towards smart grids (Doleski, 2017). Smart grids change the current power system from a centralized production approach to a more flexible and distributed integration of renewable energy resources. This results in an increased intertwining of supervisory control and data acquisition (SCADA) systems and information and communication technology (ICT). The communication infrastructure is one of the key drivers towards such a smart grid, as the command and control of many distributed energy resources needs to be managed and coordinated. Consequently, intensive use of ICT is necessary to transform the power grid. This increases the attack surface on the overall grid (M.Igure, et al., 2006) (Yan, et al., 2013). As smart grids have strong interconnections with ICT systems, analyzing the impact of ICT-based attacks becomes a complex task. It is therefore necessary to be able to analyze and understand the effects of ICT attacks on the power grid in greater depth. This can be done by co-simulating power systems and their associated communication networks (Tuinema, et al., 2020). An isolated approach is not sufficient from the perspective of IT security since attack vectors between the systems are increasing due to the large number of small, distributed components. Those ICT attacks generate a demand for simulation testbeds which behave as similar as possible to real ICT infrastructures.

As a result of the issues outlined above, the overarching research question of "How can we holistically simulate future smart grid infrastructures?" arises. In this publication, we aim to propose a possible solution to the issue of how to integrate an ICT simulation with simulations of other associated domains. The requirements for such a simulation tool are derived in part from our federally funded research projects, which address smart grids and ICT attacks and are described in section Use Cases in Smart Grid Research Projects. We identified the following requirements for this ICT simulator:

- Seamlessly and natively work within co-simulations.
- Easily integrate custom and well-known components such as virtual remote terminal units (vRTUs) or web servers.
- Display realistic network traffic without noise from management systems such as Docker.
- Be able to use the Linux network stack.
- Integrate easily configurable and human readable network topologies.
- Support distributed execution.
- Integrate the well-known format of Dockerfile-based container images.

As stated by Muller et al., the development of scalable frameworks that combine ICT and power simulation needs to be improved (Müller, et al., 2019). They have identified three mayor challenges to overcome with in

future research: Scalability, reuse of existing work through open source access, and enabling comparative benchmarking. In order to contribute to solving those challenges, we present our network simulator's capabilities to build network infrastructures and the simple integration of containers that can be used to integrate realistic ICT components for scalable co-simulations together with power domain related simulators.
The rest of the paper is structured as follows. First, an overview of related works is given in section Related Work. Our ICT simulator is presented in section Presenting our Network Simulator, and an overview over its most important features regarding the challenges outlined above is provided. Section Use Cases in Smart Grid Research Projects shows three use cases for our simulator in the context of current smart grid research projects. To verify our simulator's functionality in a co-simulation environment, a practical application example is presented in section Practical Application Case Study. The paper ends with a summary and outlook in section Conclusion.

2. Related Work

In the recent past, many co-simulation approaches have been published, most of them in the domain of power grids. This can be explained by the increase of communication infrastructure within these grids. The continuously increasing attack surface and complexity of cross-domain integration necessitate the ability to carry out the effects and detection of attacks in simulations. This section presents recent publications that are relevant for our work. To the best of our knowledge, none of these co-simulation-capable approaches are using containers to run software or vRTUs within their simulation scenarios.

In a recent publication, Muller et al. list ten co-simulation setups that include ICT networks in the simulations, and compare them in detail (Müller, et al., 2019). It can be observed that ns-2 (the predecessor of ns-3) and OPNET Modeler are among the most frequently mentioned network simulators in these co-simulation approaches. While both concepts and software artefacts for a co-simulation of the power grid exist, they generally lack scalability. Their usability for third parties not involved with the development of the approaches is also in need of improvement, partly due to them often not being open source.

Moulema et al. define various co-simulation scenarios with ICT networks (Moulema, et al., 2015). These scenarios are used to examine the impact of poor communication network performance as well as IT attacks on the energy market price. The co-simulation was implemented using the Fenix Framework for Network Co-Simulation (FNCS) in conjunction with GridLAB-D and MATPOWER. The communication network has been simulated using ns-3. The co-simulation scenarios consider demand and response during False-Data-Injection Attack (FDIA) as well as simulate a low data rate for the communication between smart meters and their corresponding gateways. The authors have shown that FDIAs have an impact on the energy market by increasing prices. However, it remains unclear how the FDIAs have been simulated in detail. It is also unclear how the market application was integrated into the co-simulation. Such an implementation of the co-simulation seems sufficient for their purpose, but it doesn't seem to be sufficient from the IT security perspective for analyzing and evaluating IT security tools like intrusion detection systems in combination with real-world attacks on smart grids.

Another co-simulation architecture is presented by Mirz et al. (Soares, et al., 2008). It also combines the three domains energy market, power system and communication network. The authors propose a model-based topology builder based on the IEC Common Information Model (CIM). These models are used to describe an entire smart grid topology including the corresponding communication and market entities. Their architecture contains one simulator for each domain which are coordinated by the co-simulation framework mosaik. The power system simulation has been done with Modelica, the market simulation focusing Virtual Power Plants (VPPs) has been implemented in Python and the communications network has been simulated using ns-3. While this seems to be a promising approach with regard to the communication network and the IT security aspects, failures in the communications network have been investigated in only one scenario.

3. Presenting our Network Simulator rettij

In this section, we present our network simulator that we have named rettij[1] and its capabilities regarding these previously identified challenges:

- The simulated network and network-connected devices should display realistic behavior.

[1] Open Source rettij Network Simulator on GitLab: https://gitlab.com/frihsb/rettij

- The simulator needs native support for co-simulation interfaces.
- The simulator should be able to run large-scale simulations without the need for costly, specialized hardware.
- Simulated, virtualized and physical domain-specific components should be easily integrable.

Our network simulator aims to provide a solution to these challenges. The basic functionality has already been described by Woltjen, et al. in (Woltjen, et al., 2020). It simulates network-connected devices (*Node*), their network interfaces (*Interface*) and their network connections to other Nodes (*Channel*). These networks are defined via a *Topology*, which describes the components and their connections in a human-readable YAML file format. The Nodes are primarily run as containers, which allows for lightweight, scalable and repeatable execution. Our simulator rettij uses Kubernetes[2] for the distributed orchestration and network attachment of the containers. Users can use their preferred Kubernetes implementation and container runtime, and even scale the cluster into cloud environments. A simple example for such a simulation network is shown in Figure 1.

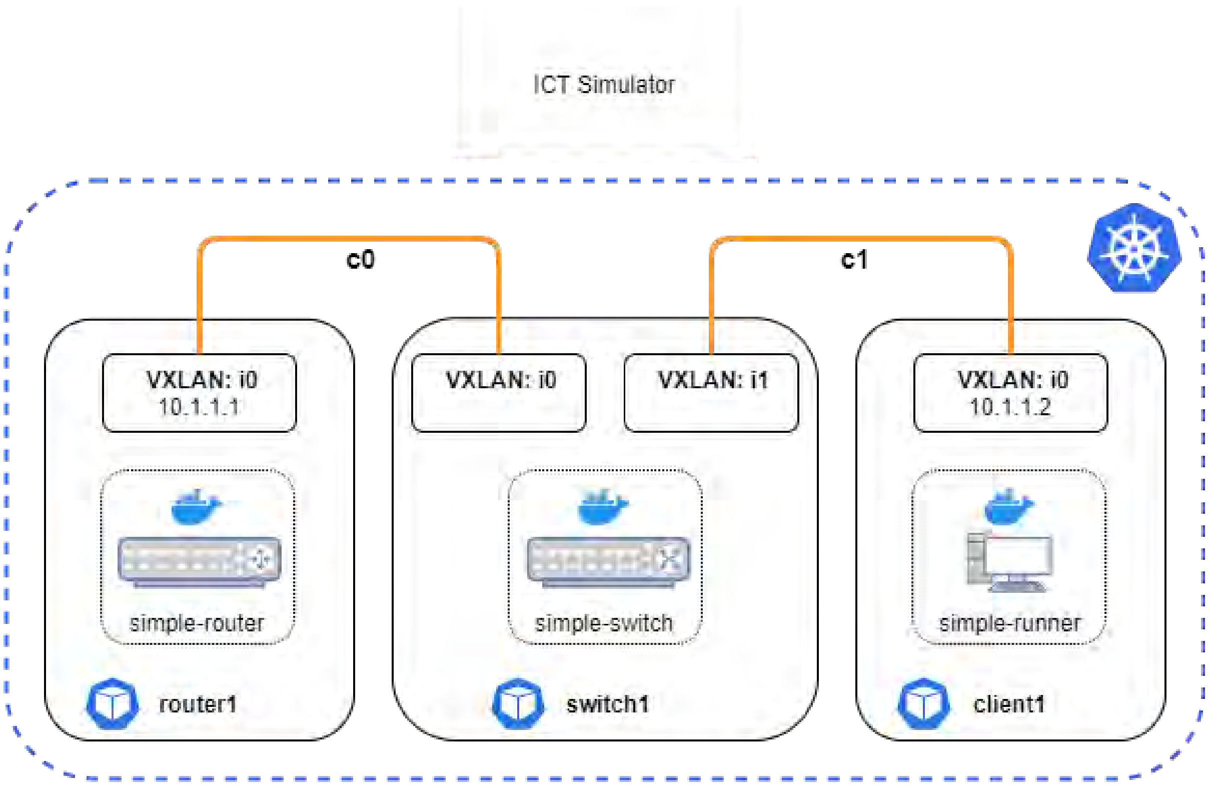


Figure 1: Simple simulation network

As various simulation scenarios require different network devices, our simulator rettij allows the definition of custom components and configurations. These use the Docker Image[3] file format in combination with a specification file and hooks to be executed at certain times during the simulation lifecycle. The use of Docker Images enables users to easily integrate commonly used real-world software images from sources like Docker Hub[4]. Users can also build their own images based on a well-documented file format. The simulator ships with pre-defined components for the most important network devices (hub, router, switch) as well as a basic network client.

3.1 Network Encapsulation

One of the primary challenges when using containerized workloads and the associated orchestrator for network simulation purposes is the presence of noise caused by management traffic on the network. Earlier versions of our simulator employed GreTap tunnels to isolate the simulation network from management traffic and other unrelated messages (Woltjen, et al., 2020). This was replaced by Virtual Extensible LAN (VXLAN)-based tunnels, as GreTap does not allow for redundant point-to-point connections between two network devices due to a lack

[2] Kubernetes: https://kubernetes.io/
[3] Dockerfile reference docs: https://docs.docker.com/engine/reference/builder/
[4] Docker Hub: https://hub.docker.com

of session identifiers. The use of VXLAN tunnels results in clean network captures only containing the traffic associated with the simulation.

3.2 Repeatable Simulation Scenarios

Our simulator allows the execution of fully repeatable simulation scenarios. Repeatable simulation scenarios are an important factor in verifying component behavior and in artificial intelligence (AI) training. This is made possible through the use of simulation sequences and shareable topologies as well as immutable component images. The Sequence defines a timed sequence of simulation steps, which can easily be repeated in every iteration of the simulation. A step can contain various commands, for example for starting a network-based attack. The Topology allows the sharing of a network environment with statically defined components and configurations. Immutable component images ensure that each simulation Node is at a fixed, known state at the beginning of the simulation. The combination of these three aspects ensure that a simulation run is not only repeatable, but also mostly reproducible. Currently, the following non-reproducible aspects are known:

- The system time gets passed from the host system to the Nodes by the container runtime, and subsequently differs between runs.
- Any naturally persistent system, like integrating the host or bridging external networks, cannot be controlled and reset by our simulator.
- The computational resources of the host system might influence the time needed for certain operations.
- The kernel of the host system, e.g. when running on a different machine, can cause slight differences.

3.3 Co-Simulation

One of the primary use-cases for our simulator rettij is to provide a network simulation for co-simulation scenarios managed by mosaik[5]. Our simulator provides an interface for control and data exchange through mosaik, as shown in Figure 2.

For example, this enables a scenario where our simulator and the power system modeling tool pandapower[6] are co-simulating both the ICT and the physical energy component state. Within this scenario, control commands are passed to our simulator through the mosaik data exchange interface, transmitted over the network via a domain-specific communication protocol, and then passed on to pandapower via mosaik. This environment can be used to study the possibilities for and the physical impact of ICT-based attacks like FDIA and denial-of-service (DoS) on energy infrastructure.

In order to facilitate the control and data exchange through mosaik, our simulator implements a step mechanism that accepts input and runs timed commands when triggered. It also defines data retrieval logic. The mechanism is depicted in Figure 3. First, mosaik calls our simulator at a scheduled time, passing both that time and the registered inputs to the co-simulation interface of the simulator.

The received data is refined and passed on to the internal step logic of our simulator. This logic first writes the input data to the corresponding Nodes. Afterwards, the list of scheduled steps is queried for a step at the current co-simulation time. If a step is present, its commands will be executed. Once the execution is finished, our simulator will return its next scheduled time to mosaik. Since the network simulation is continuously running and not discrete, the next scheduled time is simply the smallest available increment of mosaik time units from the last call time (i.e. call time + 1). At the end of each time slot, mosaik will query for a set of data points from its simulators, depending on how they are connected in the scenario. Our simulator retrieves the corresponding values from its Nodes and returns them to mosaik.

[5] Mosaik: https://mosaik.offis.de/
[6] Pandapower: https://www.pandapower.org/

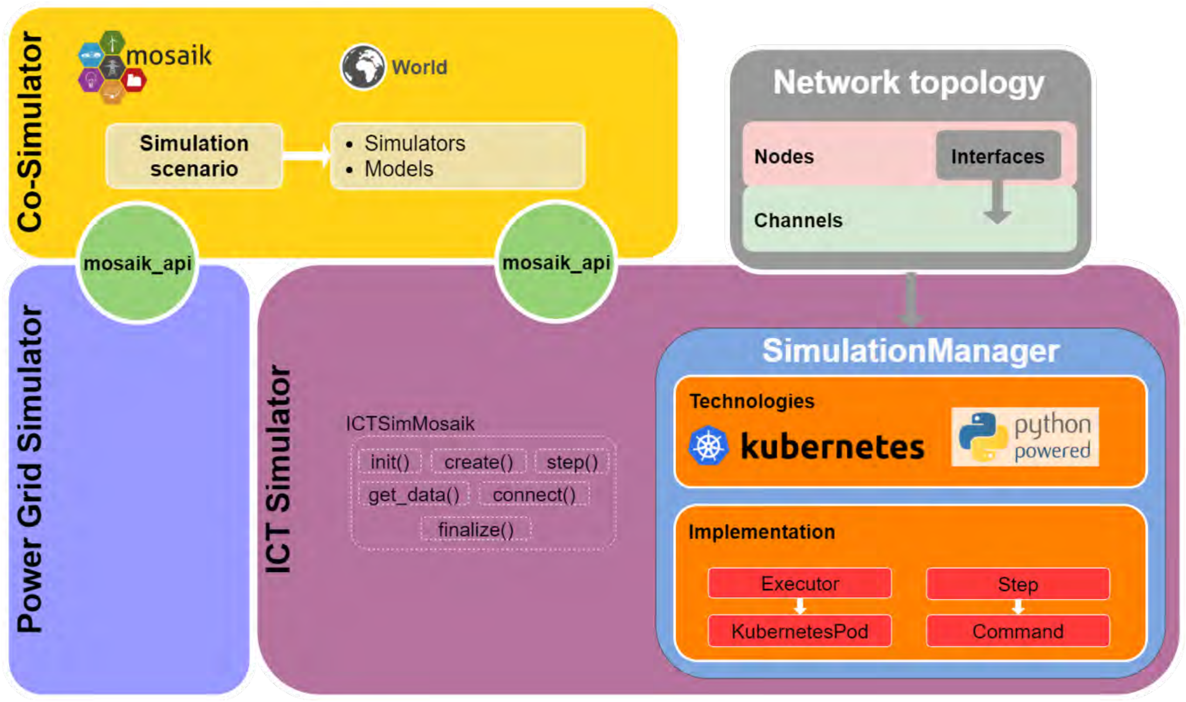


Figure 2: Overview of the interaction between our simulator and the co-simulator

One issue for our simulator that arises from its role as a "courier" in the co-simulation is the lack of information about the destination device for the data received from mosaik. Usually, mosaik connects one entity to another entity directly, and passes data via its data exchange interface. This is not possible with our simulator, as data from one simulator entity destined for another entity is not passed directly, but through the simulation network. To remedy this issue, our simulator provides a connect method similar to that of mosaik. This method allows the user to define (from within the mosaik scenario) where a message should be sent when a specific data point is updated. To facilitate the highest amount of flexibility, this functionality is constructed as a hook, meaning each component has the ability to specify custom logic as to how the information on data points and destinations is used within the simulation.

3.4 Distributed Simulation

One of the distinguishing features of our simulator is the capability to run simulations on distributed computing resources rather than a single device. This has two advantages: scaling, and integration of geographically distributed hardware.

While the simulation capacities on a single device are either fairly limited or come at very high purchasing costs, our simulator allows scaling the simulation over a cluster of various, inexpensive computing nodes. This cluster may be geographically distributed. Hardware labs containing physical systems in one location can be connected to hardware or computing resources in another location through our simulator. Both scaling and geographical distribution are enabled through the use of Kubernetes. The Container Networking Interface (CNI)[7] allows for the creation of overlay networks over various host systems, resulting in a common network for all simulated components regardless of physical location of the underlying node. Depending on the requirements regarding distribution and the used cluster, an appropriate CNI can be used. As Kubernetes is meant for the orchestration of containerized workloads in large clusters, scaling and automated distribution of workloads across the cluster nodes are basic features.

[7] Kubernetes Networking Plugins: https://kubernetes.io/docs/concepts/extend-kubernetes/ compute- storage- net/network- plugins/

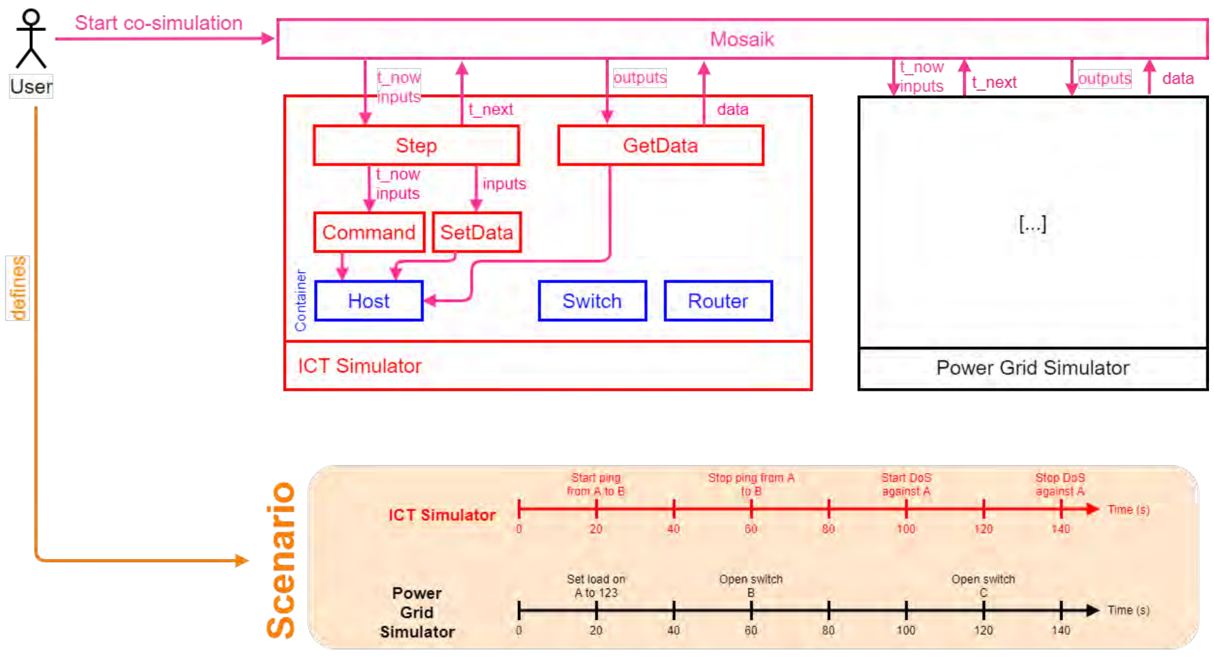


Figure 3: Co-simulation step mechanism

3.5 External Persistent Systems

In addition to container-based simulation Nodes, the systems running the Kubernetes cluster can also be attached to the simulation network via the topology. The host system can be connected via a layer 3 virtual IP interface, or act as a layer 2 bridge between the simulation network and its physical NICs. This allows for the integration of external, persistent networks and systems into the simulation. For example, a physical remote terminal unit (RTU) can be connected to a physical host in the same room, running a grid control center software, via a simulated WAN, eliminating the need for additional hardware. In addition to external persistent systems, Kubernetes-orchestrated virtual machines (VMs) are currently planned as an available node type as well.

4. Use Cases in Smart Grid Research Projects

Several research projects drive the development of our simulator through their requirements. All projects have a high demand to simulate ICT network infrastructures together with power grids. The mosaik co-simulator is used to combine and coordinate the different simulators, with our simulator being one of those simulators. Three of these project-related use cases for our simulator are outlined below.

4.1 Multi-Domain Attack Simulation with Adversarial Resilience Learning

A multi-domain analysis is performed in the project “Polymorphic agents as a cross-sectional software technology for analyzing the operational reliability of cyber-physical systems” (PYRATE). The goal is to analyze the impact of attacks in the domain’s *energy market*, *power grid* and *ICT* (Veith, et al., 2020). In order to systematically analyze how attacks within the three domains can influence each other, different scenarios are run in a co-simulation. One aspect analyzed will be the question of how an attacker can (legally) obtain the highest possible profit on the energy market. To achieve this, an extensive co-simulation is implemented, providing a local energy market, an electricity grid and the underlying communication network. The complete software stack also contains an AI attacker agent based on Adversarial Resilience Learning (ARL), which is used to attack the simulated domains under conditions that are as realistic as possible (Veith, et al., 2020). The attacker has access to sensors and actuators in all domains and is able to read and control them. In order to simulate the consequences of realistic ICT attacks, our simulator provides actuators and sensors to be used by the attacker. Actuators can manipulate data in network traffic or execute an attack such as temporary DoS attacks through a shutdown or reboot of a ICT device during the simulation. The provided sensors allow attackers to read data such as device states and data values that are sent through our network.

For this purpose, the network simulation needs a native co-simulation interface for mosaik and must allow simple integration of domain-specific components in order to provide sensors and actuators. Due to the AI training for the attacker and the number of components, a large simulation is expected, requiring high performance of the simulated ICT devices at low costs.

4.2 Network Data Manipulation in Co-Simulation

The increasing integration of ICT in power systems is also part of the research project “Methods for grid operators to prevent, detect and respond to IT attacks and failures” (MEDIT), (van der Velde, et al., 2020). The projects aims to develop methods for actors in the electric power system to prevent, detect and react to ICT attacks and resulting failures.

In this context, our simulator is part of the research and validation environment to investigate complex interactions between ICT and power systems. Our simulator will be used as part of a co-simulation environment, and will also be integrated into a real power system laboratory. It therefore needs to be able to integrate vRTUs and real hardware RTUs controlling the power flow in the laboratory. In order to develop methods for detecting attacks on the power grid, the simulated ICT devices must behave as close to reality as possible. We therefore need to be able to carry out ICT attacks in realistic environments.

Multiple attack scenarios, such as man-in-the-middle (MITM) attacks on IEC60870-5-104 packets within the network simulation, will be performed in this project. Our simulator receives IEC 60870-5-104 packages from a power simulation and passes it to the corresponding vRTU within our simulator. The vRTU then sends the IEC 60870-5-104 data frames over its simulated network to their destination. On this basis, methods to detect attacks (e.g. intrusion detection system (IDS), monitoring etc.) can be evaluated in a holistic simulation. With this, common ICT attack detection and monitoring systems can be adapted to fit into smart grids, and new attack detections mechanisms can be evaluated for their applicability in the power grid domain.

Our simulator enables such these analysis and evaluations due to its co-simulation interface and the usage of containers that allow the integration of IDS and other systems.

4.3 Blockchain Simulation

Within the research project “Trusted blockchains for the open, smart energy grid of the future” (tbiEnergy), a local energy market is to be developed and implemented with the help of blockchain technology and the usage of additional hardware security. We find the currently available network simulators to be rather difficult to use in this context due to the large number of proprietary blockchain-specific components, which have to be simulated or implemented in the simulator specific language. Our simulator offers a simple integration of components into simulations without the need for extensive adjustment of the original components. In the context of this project, this allows testing the developed blockchain concept before the roll-out on real hardware. Because of the small-scale nature of the systems in the smart grid, and the complexity of interactions between them, hardware requirements for simulation systems are high. With its ability to run simulations distributed on a cluster, our simulator offers an advantage over classic simulators that can only use the computing resources of a single hardware system. Such a cluster could be scaled according to the specific simulation. This scalability in connection with the use of our simulator in co-simulations offers a solution for the simulation of complex scenarios within the smart grid. New approaches and concepts can thus be tested as in a realistic environment, even before the rollout on actual hardware. In the context of this project, our simulator thus offers the possibility to validate the countermeasures, which mitigate threats identified in an initial threat analysis, and to uncover any remaining weaknesses within the system.

5. Practical Application Case Study

In order to verify our simulator’s functionalities in a co-simulation environment, a practical application example was created. This example is meant to show the co-simulation with mosaik as well as our simulator’s networking capabilities.

For this purpose, a minimal scenario featuring a MITM-based message modification attack was implemented. This co-simulation scenario contains our simulator and two simple simulators, one for creating and sending and one for receiving and storing random data values (see Figure 4). The network topology for our simulator consist of a Node each for the sender and receiver simulators which contains a simple TCP sender/receiver program, as well as the network components required for three virtual LANs with routing between them. Additionally, an attacker component is connected to one of the switches. It is used to intercept and modify the traffic from the sender to the receiver Node.

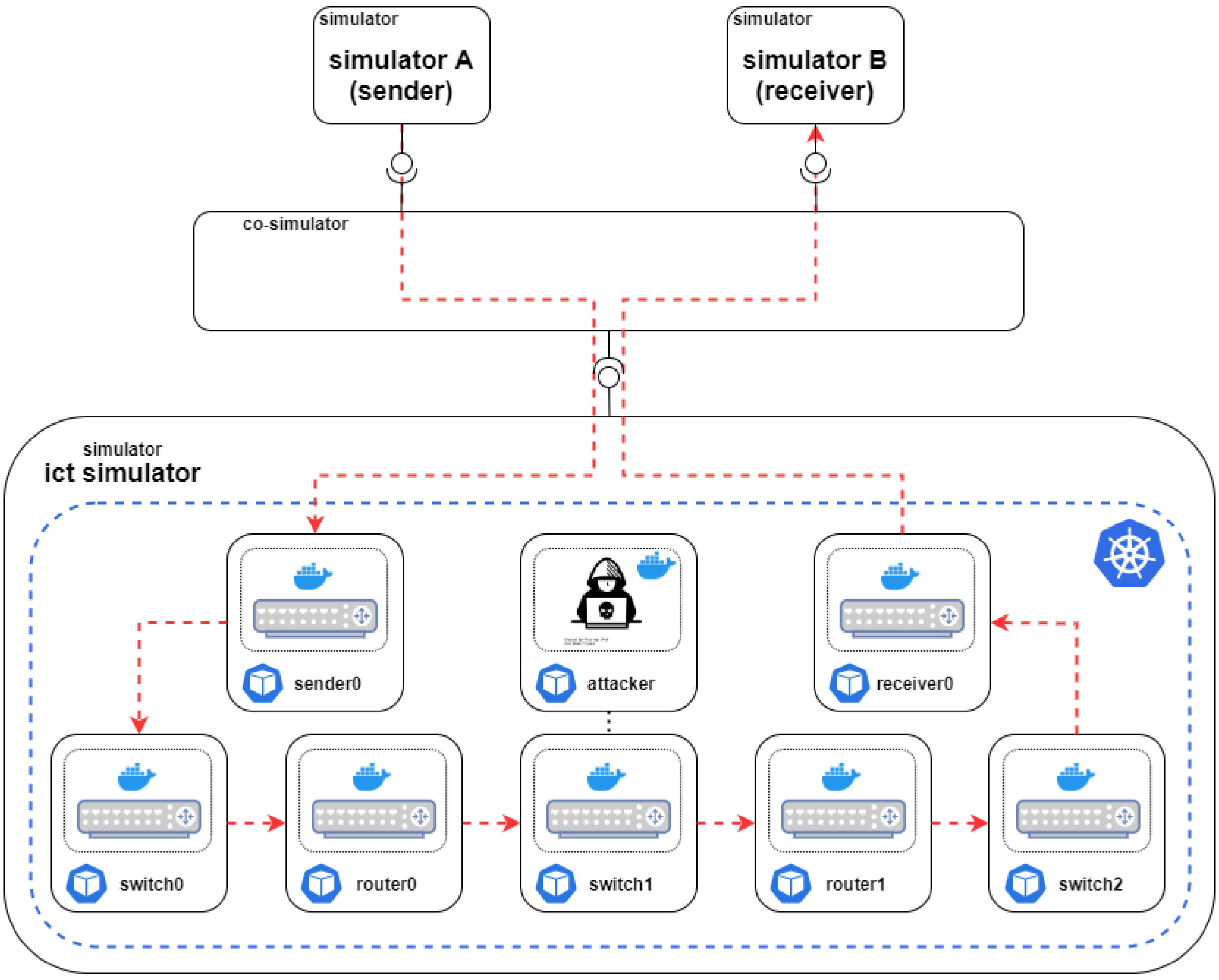
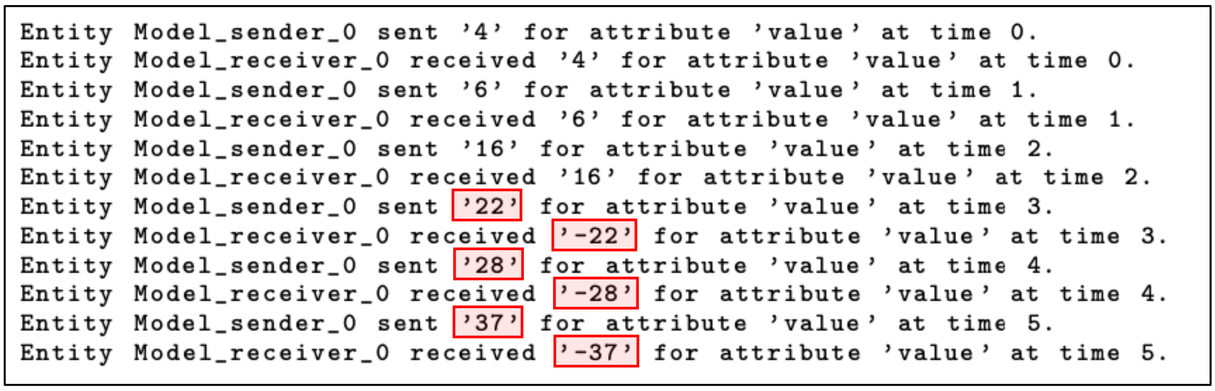


Figure 4: Dataflow co-simulation

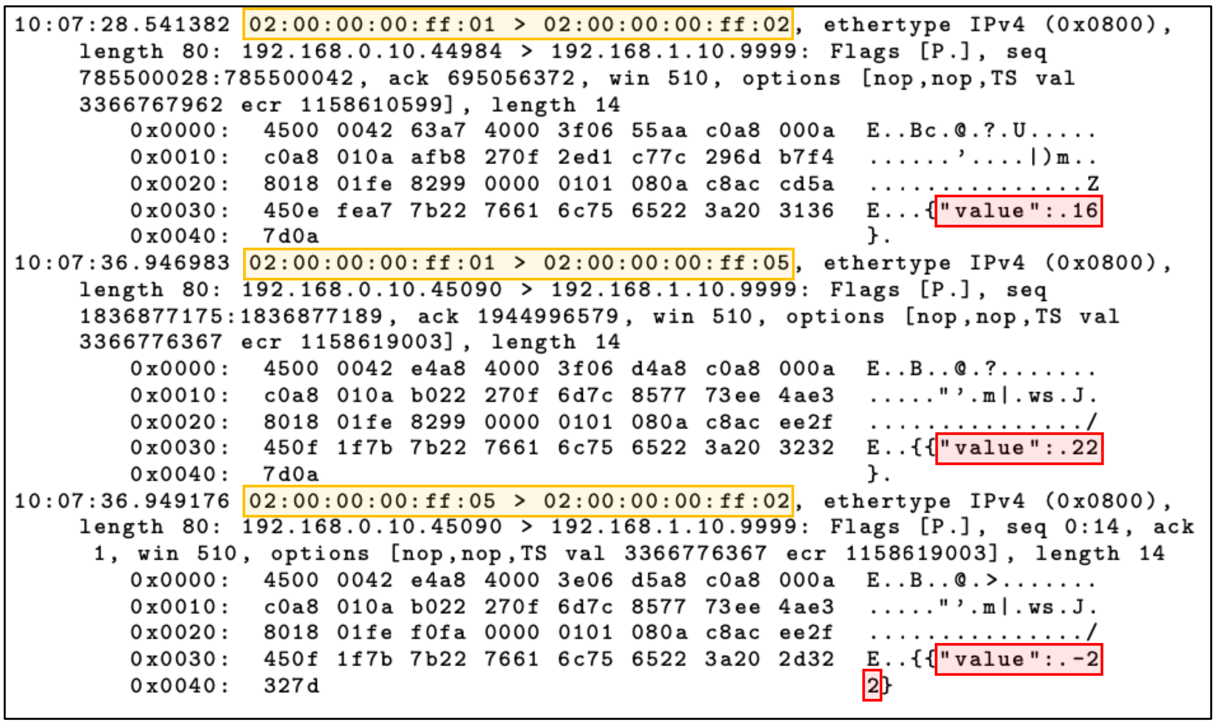
During the simulation run, the sender simulator will continuously create a semirandom, incrementing integer value which is passed on the sender0 Node in our simulator rettij via mosaik. The TCP sender program on the Node then sends the data to the receiver0 Node via the network. The packet is then captured and modified by the attacker Node as part of the MITM-based message modification attack (similar to an FDIA). Once the packet reaches its destination on the receiver0 Node, it will be passed on to the receiver simulator where the data is then stored. In order to establish a baseline for the communication between the sender0 and receiver0 Node, the simulation sequence is defined in such a way that it starts a packet capture on switch1 at the very beginning of the simulation (t=0), and only triggers the attack at t=2. This way, unmodified traffic is captured for the first 2 time increments of the simulation.



```
Entity Model_sender_0 sent '4' for attribute 'value' at time 0.
Entity Model_receiver_0 received '4' for attribute 'value' at time 0.
Entity Model_sender_0 sent '6' for attribute 'value' at time 1.
Entity Model_receiver_0 received '6' for attribute 'value' at time 1.
Entity Model_sender_0 sent '16' for attribute 'value' at time 2.
Entity Model_receiver_0 received '16' for attribute 'value' at time 2.
Entity Model_sender_0 sent '22' for attribute 'value' at time 3.
Entity Model_receiver_0 received '-22' for attribute 'value' at time 3.
Entity Model_sender_0 sent '28' for attribute 'value' at time 4.
Entity Model_receiver_0 received '-28' for attribute 'value' at time 4.
Entity Model_sender_0 sent '37' for attribute 'value' at time 5.
Entity Model_receiver_0 received '-37' for attribute 'value' at time 5.
```

Figure 5: Data send and received by the simulator

The attack was run for 12 time increments, during which the attacker simply inverted the intercepted values (value * -1) and then forwarded the packets to their original destination. A section of the data sent and received by the sender and receiver simulators is shown in Figure 5. There, the beginning of the message modification attack can be seen between t=2 and t=3.



```
10:07:28.541382 02:00:00:00:ff:01 > 02:00:00:00:ff:02, ethertype IPv4 (0x0800),
     length 80: 192.168.0.10.44984 > 192.168.1.10.9999: Flags [P.], seq
     785500028:785500042, ack 695056372, win 510, options [nop,nop,TS val
     3366767962 ecr 1158610599], length 14
         0x0000:  4500 0042 63a7 4000 3f06 55aa c0a8 000a  E..Bc.@.?.U.....
         0x0010:  c0a8 010a afb8 270f 2ed1 c77c 296d b7f4  ......'....|)m..
         0x0020:  8018 01fe 8299 0000 0101 080a c8ac cd5a  ...............Z
         0x0030:  450e fea7 7b22 7661 6c75 6522 3a20 3136  E...{"value":.16
         0x0040:  7d0a                                     }.
10:07:36.946983 02:00:00:00:ff:01 > 02:00:00:00:ff:05, ethertype IPv4 (0x0800),
     length 80: 192.168.0.10.45090 > 192.168.1.10.9999: Flags [P.], seq
     1836877175:1836877189, ack 1944996579, win 510, options [nop,nop,TS val
     3366776367 ecr 1158619003], length 14
         0x0000:  4500 0042 e4a8 4000 3f06 d4a8 c0a8 000a  E..B..@.?.......
         0x0010:  c0a8 010a b022 270f 6d7c 8577 73ee 4ae3  ....."'.m|.ws.J.
         0x0020:  8018 01fe 8299 0000 0101 080a c8ac ee2f  .............../
         0x0030:  450f 1f7b 7b22 7661 6c75 6522 3a20 3232  E..{{"value":.22
         0x0040:  7d0a                                     }.
10:07:36.949176 02:00:00:00:ff:05 > 02:00:00:00:ff:02, ethertype IPv4 (0x0800),
     length 80: 192.168.0.10.45090 > 192.168.1.10.9999: Flags [P.], seq 0:14, ack
      1, win 510, options [nop,nop,TS val 3366776367 ecr 1158619003], length 14
         0x0000:  4500 0042 e4a8 4000 3e06 d5a8 c0a8 000a  E..B..@.>.......
         0x0010:  c0a8 010a b022 270f 6d7c 8577 73ee 4ae3  ....."'.m|.ws.J.
         0x0020:  8018 01fe f0fa 0000 0101 080a c8ac ee2f  .............../
         0x0030:  450f 1f7b 7b22 7661 6c75 6522 3a20 2d32  E..{{"value":.-2
         0x0040:  327d                                     2}
```

Figure 6: Packet capture on switch 1

On the network level, this is also visible in the output of tcpdump on switch1, as shown in Figure 6. While the transmitted value 16 at t=2 is transmitted between router0 (02:00:00:00:ff:01) and router1 (02:00:00:00:ff:02) with no interception, value 22 at t=3 is first passed on to the attacker (02:00:00:00:ff:05) where it is modified and then sent to its original destination via router1.

Using this practical application example, we were able to show that our simulator rettij is able to exchange data with other simulators through the co-simulation interface, perform a real-world ICT attack on a realistic network topology utilizing both switches and routers as well as user-defined components and analyze the resulting traffic without any network noise. In terms of practical applicability, the demonstrated scenario is close to actual attacks on ICT infrastructure in the power grid, during which control commands and value measurements are modified or suppressed in order to influence the associated physical systems.

6. Conclusion

In this publication, the current challenges in the area of smart grid co-simulation of ICT with regard to its security were highlighted. In particular, the increasing attack surface through ICT must be considered in future simulations in order to gain new insights and to be able to adapt and develop new ICT security tools to the needs of the smart grid. This requires observations spanning multiple domains, which in turn requires an ICT simulator that can be used in co-simulations. Consequently, our network simulator was presented, providing a co-simulation interface for mosaik. The network simulator makes use of containers and Kubernetes which allows a simple integration of various software components, such as vRTUs, as well as distributed ICT simulation. Use cases from various research projects are shown to demonstrate the need as well as the benefits of a distributed, co-simulation capable ICT simulator. The introduction of real-world ICT attacks into a multi-domain smart grid co-simulation was demonstrated by a practical example, presenting some of the distinguishing features of our network simulator.

References

Doleski, O. D., 2017. *Wie sich die Energiewirtschaft im Zeitalter der Digitalisierung verändert.* Wiebaden: Springer Fachmedien Wiesbaden GmbH 2017.

Yan, Y., Qian, Y. & Sharif, H., 2013. A Survey on Smart Grid Communication Infrastructures: Motivations, Requirements and Challenges. Band 20, pp. 5-20.

Tuinema, B. W. et al., 2020. *Cyber-physical system modeling for assessment and enhancement of power grid cyber security, resilience, and reliability. In Probabilistic Reliability Analysis of Power Systems.* s.l.:Springer.

Soares, J. et al., 2008. *A Cosimulation Architecture for Power System, Communication, and Market in the Smart Grid,* s.l.: s.n.

Moulema, P., Yu, W., Griffith, D. & Golmie, N., 2015. On Effectiveness of Smart Grid Applications Using Co-Simulation. In: *2015 24th International Conference on Computer Communication and Networks (ICCCN).* s.l.:IEEE, pp. 1-8.

Veith, E. M. et al., 2020. *The Adversarial Resilience Learning Architecture for AI-based Modelling, Exploration, and Operation of Complex Cyber-Physical Systems.* online, IARIA.

Woltjen, T., Gritzan, G., Kathmann, P. & Sethmann, R., 2020. *Simulationsumgebung für IKT-Netze zur Cyber-Abwehr: IKT-Netzsimulation zur Prävention von Angriffen auf das Energienetz.* Berlin, VDE VERLAG GMBH, pp. 233-239.

van der Velde, D. et al., 2020. *Methods for Actors in the Electric Power System to Prevent, Detect and React to ICT Attacks and Failures.* Gammarth, Tunisia, IEEE.

M.Igure, V., A.Laughter, S. & D. Williams, R., 2006. *Security issues in SCADA networks.* 7 Hrsg. s.l.:Computers & Security.

I. T. F. o. I. T. et al., 2019. Interfacing Power System and ICT Simulators: Challenges, State-of-the-Art, and Case Studies. *IEEE Transactions on Smart Grid,* pp. 14-24.