Electrical Power and Energy Systems 104 (2019) 817–826

Contents lists available at ScienceDirect

Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes

Implementation and development of an offline co-simulation testbed for studies of power systems cyber security and control verification

Eman Hammad[a], Mellitus Ezeme[b], Abdallah Farraj[a,*]

[a] University of Toronto, Toronto, ON M5S 3H7, Canada
[b] University of Ontario Institute of Technology, Oshawa, ON L1H 7K4, Canada

ARTICLE INFO

Keywords:
Smart grid
Offline co-simulation
Cyber security
Control
Testbed design and implementation
Cyber-physical systems

ABSTRACT

Smart power systems have recently shifted to accommodate distributed control systems, renewable sources of energy, consumer-centric energy management systems, and active distribution systems. The smart grid evolution is modeled by the integration of power systems and a communication network overlay to facilitate a bi-directional flow of information and energy in the grid. This article provides a detailed description of an offline co-simulation testbed for smart grids, that is developed by integrating well-established power and communication systems simulators. The testbed development approach, setup and implementation are described at a detailed level to enable similar test-bed developments. The developed testbed is envisioned as a tool to help smart grid researchers with the study of relevant research problems such as assessing power system resilience against cyber attacks and threats, and verifying the performance of cyber-enabled control schemes.

1. Introduction

Smart grid stakeholders continue to invest in employing communication systems and infrastructure that enable more flexible, efficient and reliable grid operations and control. By facilitating consumer-centricity and the integration of renewable resources within smart grid systems, the landscape of power systems operation is changing especially in terms of the flow of both information and power. Given the rapid evolution of smart power systems, it is critical that a rigirious investigation of the impacts of (cyber) information and communication technologies (ICT) on (physical) power systems and vice versa be considered by the research community [1–3]. Earlier research studies simplified the impact of communication network on a smart grid system as time delay to be accounted for within the control loop; however, this has proven to be insufficient in terms of investigating the cyber-physical coupling within the smart grid [4,5]. Interfacing existing ICT and power system simulators (termed *co-simulation*) is thought to be a practical and realistic approach to represent their smart grid interactions [6,7].

The introduction of recent interoperability guidelines and standards for smart grid development attests to the critical need for secure and sustainable operation of the cyber and physical subsystems. For example, the IEEE 2030–2011 standard provides principles for smart grid interoperability of power and ICT components [8,9]. This standard presents a *system of systems* view of the smart grid with three parts: power systems, communication, and information technology. As illustrated in Fig. 1, this high-level architecture inspires our proposed co-simulation testbed consisting of analogous abstractions.

Typically, the operation of smart grid power and control components can be described with well-defined mathematical formulations; however, the same cannot be said for the accompanying ICT system because of its often (unpredictable) stochastic behavior and event-driven layered protocol structure involved in data transmission. This motivates the need to combine existing communication and power simulators to enable formal and realistic studies including the verification of cyber-enabled control and analysis of cyber security threats and attacks. Hence, there has been a growing research interest in development of smart grid co-simulators. Approaches for co-simulation development can be categorized into two main approaches from an architecture perspective: 1) tool-based approach; where the test-bed is focused on integrating a specific set of simulators based on the understanding of tools architectures and interfacing capabilities. 2) A platform-based approach; where the focus is on developing a common co-ordinating framework that adopts a more standardized interfacing to support different simulation tools. The first approach is often adopted by researchers who have a known limited set of tools and require more understanding and control over subsystems interactions. The second approach is appealing for researchers with more complex simulation environments and studies that require the flexibility of a systematic

* Corresponding author.
E-mail addresses: ehammad@ece.utoronto.ca (E. Hammad), mellitus.ezeme@uoit.net (M. Ezeme), abdallah.farraj@utoronto.ca (A. Farraj).

https://doi.org/10.1016/j.ijepes.2018.07.058
Received 27 March 2018; Received in revised form 13 July 2018; Accepted 24 July 2018

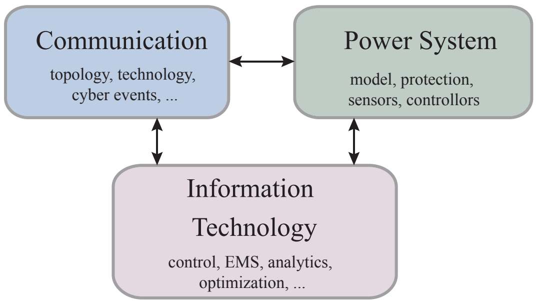
Communication
topology, technology, cyber events, ...
Power System
model, protection, sensors, controllors
Information Technology
control, EMS, analytics, optimization, ...

Fig. 1. System architecture in compliance with IEEE 2030.

interfacing approach without being involved in the details or architecture of either interfacing subsystem.

There has been a momentum of research into both approaches, a survey in [10] elaborates on some of the developed testbeds. Functional mock-up interface (FMI) is a popular framework that is used in platform-based co-simulation approaches [11]. The framework defines a standardized interface to integrate components of complex cyber-physical systems. High-level architecture (HLA) is another standardized co-simulation architecture developed to link different simulators into a distributed environment federation [12]. The ptolemyII platform-based framework is developed using hybrid-systems theory to enable the simulation of heterogeneous systems [13]. Further, other software tools are developed that utilize multi-agent architectures to integrate the simulation of different subsystems. Examples here include Multi-agent Environment for Complex SYstem CO-simulation (MECSYCO) [14], and Mosaik [15] which was developed with a focus on large scale smart grid systems.

In this work, a tool-based approach is adopted because there is a defined set of accessible simulators to use, and a more control on the interfacing between the chosen simulators is preferred. Further, the developed testbed aimed to simplify its approach to enable other research teams to quickly reproduce similar testbeds.

An example subset of previously-developed smart grid co-simulation testbed is captured in Table 1. The table summarizes available tools from both domains (cyber and power) and the research problems studied using these testbeds. The EPOCHS testbed [16] is developed using a multi-agent approach based on the HLA architecture and run-time infrastructure (RTI) middleware [16]. The testbed is among the first to adopt a platform-based approach and is provided as an open-source to interested researchers. GECO [17] presents a tool-based co-simulation testbed that embeds a discrete events scheduler into the power simulator for events coordination and data exchange. A GridLAB-D based integrated simulation engine is described in [18] where a middle interfacing and coordinating layer is developed to be shared between GridLAB-D and NS3. The ORNL power system simulator presented in [19] follows a similar tool-based approach where the ADEVS is developed as a discrete events wrapper around the power simulator THYME. ADEVS defines and coordinates the interfacing between THYME and the OMNeT++ communication simulator. In GridSim testbed [20] the authors develop their own data delivery module denoted GridStat, and integrates the power simulator TSAT within the testbed. A SCADA Cyber Security Testbed integrating PowerWorld and RINSE is illustrated [21]. The testbed implements a protocol converter that is envisioned to enable interfacing with hardware systems/components.

A realtime co-simulation testbed developed in [22] integrated RTDS realtime digital power simulator with hardware controllers, relays and real Ethernet network. The approach is made possible by the interfacing capabilities of the RTDS which do not require additional customization by the testbed developer. Another realtime co-simulation testbed is described in [23] integrating Opal-RT power simulator and OPNET communication simulator. The testbed capitalizes on the features provided by both simulators allowing an easy integration. It is important here to note that realtime co-simulation testbeds often have plenty of features provided by the expensive simulators allowing for seamless integration in many cases. A disadvantage of these testbeds is recognized when considering communication protocols that are not supported by either simulator, where more elaborate development is needed to add that capability [24].

It can be noted that a detailed implementation and setup of co-simulation testbeds are not appropriately detailed in the existing co-simulations literature especially to enable reproducing and comparing research results. The lack of such details forces interested research teams to create, from scratch, their own co-simulators. This paper is motivated by the need for a comprehensive exposition on offline co-simulator development for smart grid applications that can be reproduced or easily modified to support smart grids studies. Offline co-simulation is a cost-effective approach to analysis that is not only more accessible (less expensive), but typically easier to implement while being useful for a variety of “what-if” analysis and analytics for system planning. However, offline co-simulation has few limitations such as it’s extended time of simulation because of the added interactions between subsystems. Moreover, offlines testbeds do not have the capability to integrate hardware systems (e.g. PMU, relays, controllers).

A main challenge of interfacing two simulators, each of distinct characteristics, is to integrate both simulated systems while effectively maintaining the core independence of each. Enabling different simulation platforms to work hand in hand to represent a realistic smart power system behavior requires dealing with synchronization and data exchange challenges [25]. Synchronization issues typically occur between interfaced simulators because of the differences in the size of simulation time-step and the execution sequence [26]. On the other hand, data exchange issues at the interface between the two simulators should be scalable to have a minimal impact on the overall performance of the co-simulation testbed.

Table 1
Sample co-simulation testbeds.

ID	Power simulator	Cyber simulator	Offline/realtime	Investigated problem
EPOCHS [16,29]	PSCAD/EMTDC & PSLF	NS2	Offline	Protection, special protection schemes
GECO [17,30–32]	PSLF	NS2	Offline	PMU based wide area monitoring systems
Integrated simulation engine [18]	GridLAB-D	NS3	Offline	Distribution & demand response (DR)
ORNL power system simulator [19]	THYME model	ADEVS (NS2 & OMNeT++)	Offline	Control & communication
GridSim [20]	TSAT (DSATools)	GridStat	Offline	Wide area control & protection schemes
SCADA Cyber Security Testbed [21]	PowerWorld	RINSE	Offline	SCADA Cyber Security
[33,34]	OpenDSS	OMNeT++	Offline	Electric vehicles (PEV) coordinated scheduling
[35]	OpenVZ(emulator)	S3F(simulator)	Offline	advanced metering infrastructure (AMI) attacks
[22,36]	RTDS	NS3	Realtime	cyber vulnerability & mitigation
[23]	Opal-RT	OPNET[a]	Realtime	communication latency impact on microgrid control
[37]	Opal-RT	hardware network	Realtime	adaptive mitigation of cyber incidents
[38]	Opal-RT	communication adapter	Realtime	energy battery management

[a] The communication network simulator OPNET was later acquired by Riverbed and became known as Riverbed Modeler.

Next, we expand on how synchronization challenges have been approached in example testbeds. [27] sets the communication simulator as the master of event synchronization between the two simulators, but this causes challenges for events generated within Modelica which is used as the power simulator. The GreenBench testbed developed in [28] integrating PSCAD and OMNeT++ attempts to avoid synchronization errors by editing the behavior of OMNeT++ to self trigger at time-steps equal to those of the power simulator. This modified behavior will add additional events in the communication simulator that do not necessarily correspond to power system events, thus resulting in an inaccurate communication representation and dynamics. Coordination via RTI in EPOCHS handles synchronization by marking synchronization points of fixed time separation for both simulators. Both simulators continue execution in parallel until the next synchronization point is reached where data then is exchanged if needed [25]. This approach partly ignores how communication network are unpredictable such as in cyber security studies.

1.1. Contributions

This article describes an offline co-simulation testbed that utilizes TCP/IP sockets and a lightweight database to facilitate data exchange between the co-simulator components. This can guide research communities when considering co-simulation development utilizing their own tools. The main contributions of this works include the following:

1. A cyber-physical abstraction of offline tool-based co-simulation architecture is provided. This abstraction can be extended to other co-simulation testbeds such as intelligent transportation systems.
2. A detailed implementation of the offline co-simulation testbed using selected tools is outlined. The described testbed implementation relies on a system-view understanding of the tools and their handling of events.
3. An efficient event-based scheduler is developed to coordinate events between the co-simulator subsystems. The proposed scheduler bounds the synchronization error to the power simulator time-step, and utilizes a light-weight database to buffer queued events.
4. A customized packet structure is developed for message exchange between the two simulators than can be used to study the different smart grid communication protocols.

The developed testbed can be scaled to any size power system supported by PSCAD and any number of sensors and actuators without an increased overhead due to the co-simulator setup itself. This work is designed, developed and presented to provide comprehensive understanding of one possible approach for offline co-simulation development. The proposed synchronization approach (scheduler) is not matched by other coordination schemes in similar tool-based testbeds.

The next section presents the architecture of the co-simulator and its subsystems. The power system component of the co-simulator is discussed in Section 3. Section 4 details the implementation of the scheduler which handles the coordination amongst different subsystems. The development of the communication simulator is further explained in Section 5. The overall interaction between the co-simulator subsystems is examined in Section 6, followed by an example case study in Section 6.1. Finally, concluding remarks are included in Section 7.

2. Architecture of Co-simulation Testbed

Co-simulation testbeds can be classified as realtime or offline. A realtime co-simulator indicates that the time-step of individual simulators is configured in such away that the runtime of the physical system model simulation is equal to that of the real physical system.[1] For example, a 20-s realtime co-simulation is a simulation of the actual physical system running for 20 s. However, in an offline co-simulation testbed, the time-step of either or both simulation software is (typically) longer than that of the physical system being simulated. Realtime co-simulation testbeds usually include expensive computational engines with dedicated realtime hardware and operating systems. Consequently, for a general-purpose smart grid testbed that is more accessible to research and development teams we consider the development of an offline co-simulation testbed. The testbed adopts a *system-of-systems* view recommended by the IEEE 2030–2011 standard as shown in Fig. 1. If needed, this hierarchical approach allows the interfacing of various information processing tools, such as optimization or data analytics engines.

Illustrated in Fig. 2, the proposed smart grid co-simulator testbed includes:

1. EMTDC/PSCAD [39] module that will be used as a power system simulator,
2. scheduler module,
3. OMNeT++ [40] module that will be used as a communication simulator, and
4. third-party component; for example, an energy management system that is implemented using MATLAB.

In the following sections, we present a detailed description of the implementation steps of each subsystem of the co-simulator.

3. Power simulator: EMTDC/PSCAD

PSCAD provides a graphical user interface for the EMTDC power system simulation engine. A testbed user can utilize PSCAD to model, build, and simulate power systems of interest. For this co-simulation testbed, the power system model is designed in PSCAD with minimal customization. PSCAD provides pre-programmed and user-defined modules. It is also a single-thread application by design; thus, if the execution of the main program is interrupted or stops for any reason, the PSCAD program will pause until the execution returns to the main program. As such, the co-simulation testbed is designed with an additional module (the scheduler) to handle the different interactions between the various testbed simulators to ensure synchronization.

As a power system model is designed in PSCAD, different sensors can be added as measurement points that will be later defined as inputs to the user-defined module. In addition, actuation points can be defined in the power system, where either delayed measurements or control commands are fed to the power system after interacting with a communication simulator or other functions outside the PSCAD environment.

3.1. C++ and C interfaces

PSCAD allows users to build custom user-define modules and embed them into the EMTDC simulation engine. In the presented testbed, a user-defined module is developed to provide an interface to enable data exchange from EMTDC to the rest of the co-simulation testbed and vice versa. As shown in Fig. 2, the main interfaces that are developed in the testbed are:

1. C-subroutine that enables the interaction between the (pre-programmed) PSCAD modules and the user-defined modules, and
2. C/C++ inter-process windows socket interface that enables external interactions between the communication simulator and the

[1] Physical systems simulators rely on mathematical models and computational solvers, where fixed time-steps or variable time-steps can be configured.

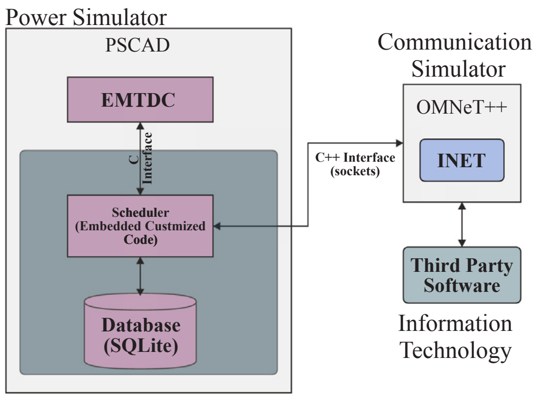

Fig. 2. Co-simulator subsystems.

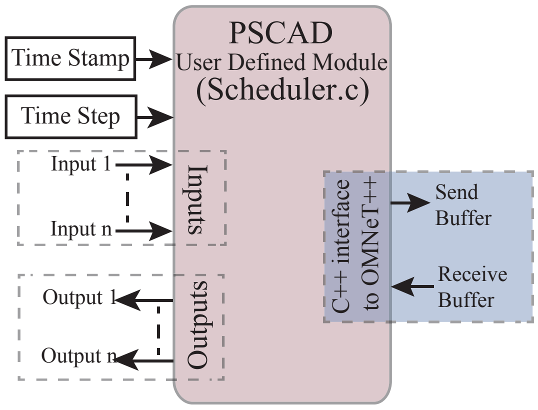

Fig. 3. Scheduler interface block diagram.

PSCAD/EMTDC system.

3.2. Sockets

TCP/IP windows sockets are used as the interface between the different simulators in the testbed due to their low complexity, efficiency, and scalability. TCP/IP sockets are used to couple the two simulators, which requires a reliable protocol (TCP) to ensure messages from one simulator are delivered to the other. Further, the use of TCP/IP simplifies the implementation of the testbed in regular research labs computer networks. The socket interface can be adjusted using the following user-configurable parameters:

- *Sensor sampling rate*: this parameter allows a user to specify different sensors in the power system with different sampling rates, if needed, to enable the study of different power system dynamics of interest.
- *Dynamic buffer size*: this parameter is used to define the size of data packets that are exchanged between the EMTDC/PSCAD simulator and the communication simulator.

The *sensor sampling rate* directly affects how the scheduler subsystem responds to changes. If the power system is equipped with synchronized sensors (for example, phasor measurement units), then the sensor sampling rate can be made uniform for all sensors, and in this case only one value is configured. The *dynamic buffer size* is usually configured depending on the number of sensors deployed in the simulated power system. This parameter can be manually modified through the *system configuration file*.

4. Scheduler

The co-simulation testbed relies on the *scheduler* to coordinate the operation of the different simulators. The basic functionality of the scheduler module is that:

- it creates a discrete-event queue for all events that are generated from the different simulators,
- it sorts these events according to their corresponding time-stamp, and
- it alternates the run of the simulators based on the next event in the queue.

The scheduler operation requires a database to store temporary data fields during the co-simulation run. The used database in this implementation is an open-source lightweight database: *SQLite*, [41]. An advantage of *SQLite* database is that it can be easily embedded in applications because of its small size. In addition to the business logic of the scheduler, the socket code is included. The *SQLite* database provides the scheduler with a C API to establish and maintain a database connection. The *SQLite* database is designed to include tables that correspond to the number of output devices that need to be controlled in the power system.

The input signals from the PSCAD/EMTDC simulator are not stored in the database; rather, these signals create interrupt signals that halt the operation of the power simulator. This is followed by the scheduler handing over to proceedings to the OMNeT + + simulator. When the transmitted signals finish traversing the communication network, OMNeT + + delivers them to the scheduler which in turn utilizes the signals' *destination tag* for storage at the correct database tables. The database tables include two fields to facilitate this process: one field for the time-stamp of each output device being controlled and another field for the control status.

Fig. 3 presents a depiction of the PSCAD user-defined module that contains the scheduler code. The scheduler module contains various functions that are called every time-step during the co-simulator run time. Inputs and outputs of the user-defined module are denoted with respect to the interface between the PSCAD power system model and the user-defined module. For examples, sampled sensor measurements are inputs, and communication delayed actuation/measurements are outputs to be fed into the power system model. For the remainder of this article, the term *actuators* is used to denote the power system model components that rely in their state on the output of the user-defined module (e.g., returned values from the communication simulator).

It is helpful here to define what we refer to as a synchronization window. Let t_{sim} be the simulation time and t_{step} be the time-step of the power simulator. Then, a synchronization window is defined as the time interval between t_{sim} and $t_{\text{sim}} + t_{\text{step}}$. The database can be perceived as a simplified event queueing system; as returned values are stored until they fall in a synchronization windows of the power simulator, where at that time their value is used to update the state of the corresponding actuators, and is then deleted from the database.

The logic included in the scheduler logic can be divided into four sections based on functionality as illustrated in Fig. 4. Specifically, the configuration section holds global variables; the database section includes functions used by the scheduler to interact with *SQLite*; the networking section has the TCP/IP socket code to interface with OMNet + +; and the scheduling section has the core logic for the scheduler as will be explained below. A flowchart of the implemented scheduler is depicted in Fig. 5.

4.1. Scheduler functions description

A brief description of the scheduler main functions is included below. Please note the correspondence between the function index in the flow chart in Fig. 5 and the index below:

1. *create_table()*: during the initialization phase, a check is performed

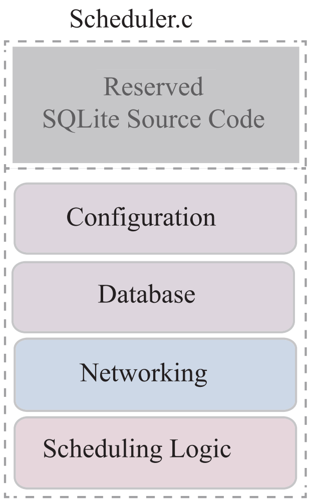

Fig. 4. Scheduler.c file content.

to verify if the current time-step is the first time-step of the simulation. If that is the case, the *create_table()* function creates the corresponding database tables to store the status with its time-stamp for each of the actuators in the power system.

2. *default_status()*: all declared variables are required to have a value before proceeding to the next step of the PSCAD simulation. This function populates the different actuators with default values. The actuators' default values are selected based on knowledge and design considerations of the power system model and expected values.
3. *check_input()*: this function decides when values from every controller and sensor should be collected and sent to the communication network simulator interface. When a sensor requires transmitting a value through the communication network, the sensor sets its corresponding *send_flag* value to 1. At each simulation time-step, the sensor rate dictates whether the *send_flag* for that sensor is set to 1 or not. If the simulation time is a multiple of the sensor rate, then that sensor's *send_flag* is set. Further, this function supports fixed (synchronized) rates and variable rates for different sensors in the system. The rates are typically set by the testbed user in the configuration section.
4. *search_send_flag()*: this function continuously scans the *send_flag* to find if any of the sensors has a set value (i.e., *send_flag* = 1). If no

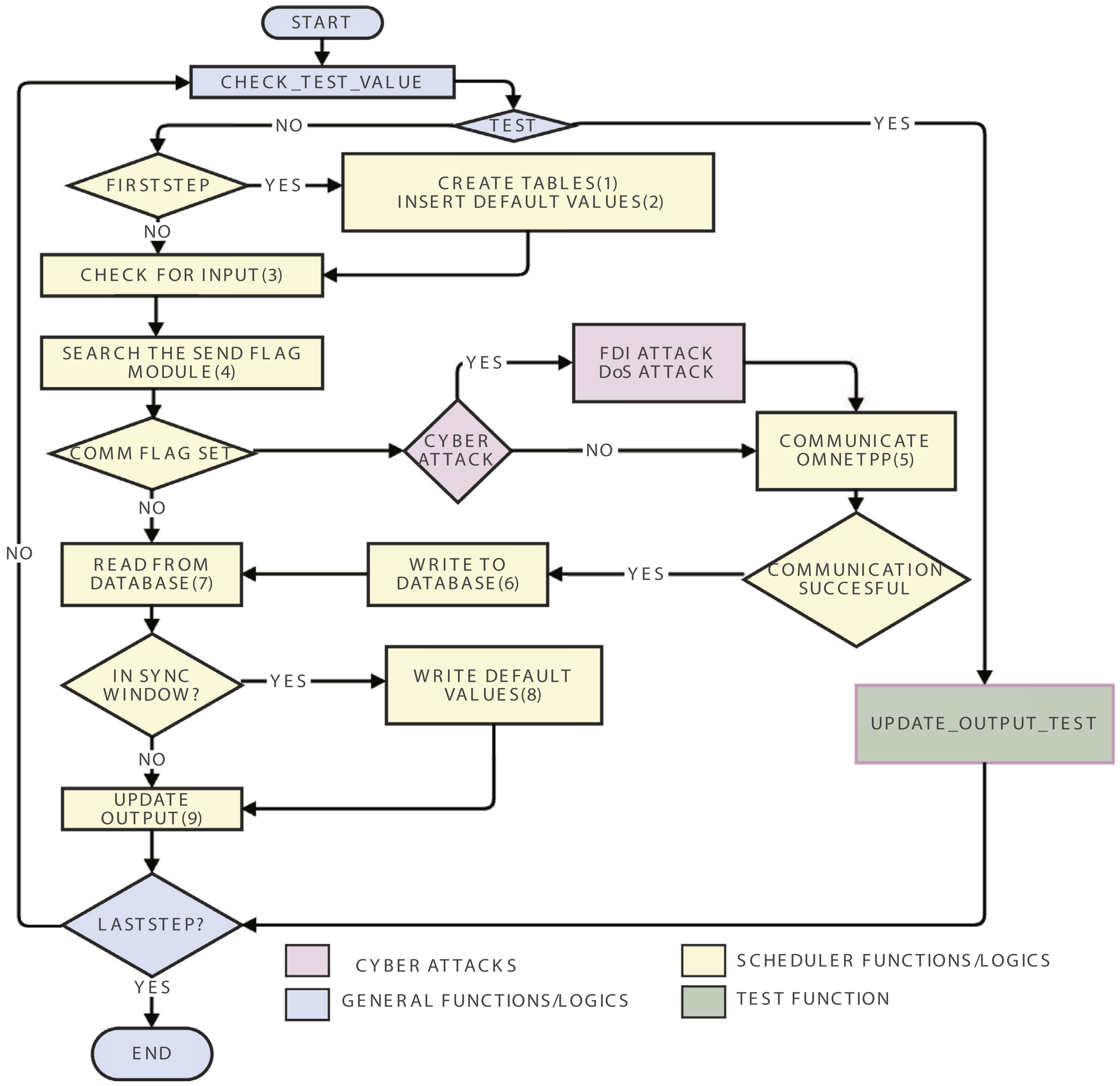

Fig. 5. Scheduler flowchart.

send_flag is set, the PSCAD simulation proceeds. On the other hand, if a *send_flag* is set, the scheduler initiates the interaction with OMNeT++ simulator using the windows socket interface. However, since both the communication and power simulators are single threaded, this action will automatically pause the operation of PSCAD/EMTDC until the scheduler decides to return the execution to it.

5. *client_socket()*: this function is used to establish the data exchange and interaction with the communication simulator module. It creates a *socket* that binds to the *socket server* at the communication simulator. The function then uses the socket to exchange data with the communication simulator in the form of *send buffer* and *receiver buffer* (termed *sendbuf* and *recvbuff* respectively). There is no delay between setting the *send_flag* and sending the measurement to the communication simulator because the two events are executed at the same time-step of the simulation.
6. *write_database()*: if the *client_socket()* function is called and a successful communication is established, the scheduler uses *write_database()* to store returned values from OMNeT++ in the database with their corresponding time-stamps. These values could represent delayed measurements or control decisions/commands (for example, a simple *close* or *open* command from actuators to power line circuit breakers or set points decision from centralized control to local controllers).
7. *read_database()*: this function is used during the simulation to scan the database at every time-step for actuators requiring a status update. If the time-stamp of any actuator falls in the current synchronization window, then the stored status of the actuator is retrieved from the database.
8. *write_default_values()*: given that *read_database()* is executed successfully and the actuator entry is retrieved from the database (this implies that the time-stamp falls in the current synchronization window), then the default values file is updated correspondingly.
9. *update_output()*: this function is used to update the statuses of corresponding actuators in the power system model from the default values file. If multiple actuators share the same time-stamp, these actuators will be updated at the same time. This function enable the power simulator to resume execution.

Further, the scheduler flow chart includes two additional functionalities:

1. *update_output_test()*: this loop tests the scheduler logic without interacting with the communication simulator; hence, the function updates the user-defined module output values from the input. The test loop restricts the number of inputs to be greater than or equal to the number of outputs.
2. to enable cyber security impact analysis, cyber attacks can be implemented using two functions: *falseDataRand()* and *denialOfServiceRand()*. More details on the implementation are provided in Section 5.3.

It is important to note here that the PSCAD/EMTDC module of the testbed is capable of interacting with other communication simulators, provided that the communication simulator supports: 1) embedded sockets, and 2) a mechanism for messaging between the embedded socket and the network model.

5. Communication simulator: OMNeT++

A communication network model is defined by the testbed user. The communication model should have a correspondence to the power system model under study and its deployed communication-enabled points. The communication simulator enables the modeling of communication delays and link congestion. In addition, from a cyber security perspective, different cyber events can be included such as denial of service attacks. OMNeT++ provides the interface for the design and simulation of communication networks. The testbed utilizes OMNeT++/INET framework which is a collection of different C++ modules (libraries) used for network modeling and simulation. Within this setup, a communication system is defined that is equivalent to the corresponding PSCAD/EMTDC power system model, and a C++ interface code is included to interact with the scheduler via sockets. Further, the developed software enables using another socket interface for interacting with third-party software such as MATLAB if needed.

OMNeT++ employs a network description language termed NED. NED lets users declare modules, connect them, and assemble them into compound modules, of which some can be labeled as networks [42]. The network model is constructed in the NED file at the top root of the project file hierarchy. Every module that is contained in the NED network must have their own corresponding NED file used to define and build that module. Further, every NED module must have implementation files which are declared and implemented in the corresponding C++ header and source files respectively.

5.1. Interface description

The relationships between the different software components implemented in the OMNeT++ subsystem are shown in Fig. 6 via a unified modeling language (UML) class diagram. The names of few classes have been adjusted in the UML diagram to be appended with "Ned" to indicate that the class is of a NED file type. The figure captures different classes which can be classified into two categories:

1. **Standard classes** from OMNeT++/INET:
   - *StandardHost*: the class can be used as either a server or a client.
   - *DatarateChannel*: one of the classes for defining communication links between network nodes.
   - *cPacket*: the class is used for messaging between components.
   - *cSimpleModule*: this is the base class in INET, which all other modules inherit from.
   - *ITCP*: the class is the INET implementation of TCP.
   - *NodeBase*: this class is an INET compound module[2] that contains the common lower layers of different types of network nodes such as Router, StandardHost, etc.
2. **Subclassed classes**; developed classes with the testbed and are shaded in blue in Fig. 6:
   - *clientMsg*: this is a subclass of *cPacket*. The user will need to define the contents of the message definition file *clientMsg.msg*; upon a successful build, the corresponding source and header files are automatically generated.
   - *InterfaceModule*: this module is a subclass of *cSimpleModule* and acts as the interface between PSCAD and the OMNeT++ network.
   - *ClientApp*: this class is also a subclass of *cSimpleModule*, but it represents the sensor/actuator nodes on the network.

The following scenario helps in conceiving the process abstracted by the class diagram. Define a PSCAD model with a certain number of sensors, one actuator, and a control center. In OMNeT++ define the corresponding communication network in a NED file with a network name *smartgrid* as an example. In this network model, the user defines the types of network nodes, their NIC interfaces, and types and parameters of communication links. The example network will contain a number of *ClientNed* nodes, one *ControlCenterNed*, one *InterfaceModuleNed*, and communication links which in this scenario are *FiberLine*. Data exchange between network nodes requires a special class *ClientMsg* to define the packet format.

[2] In OMNeT++, a compound module groups other modules into a larger unit [42].

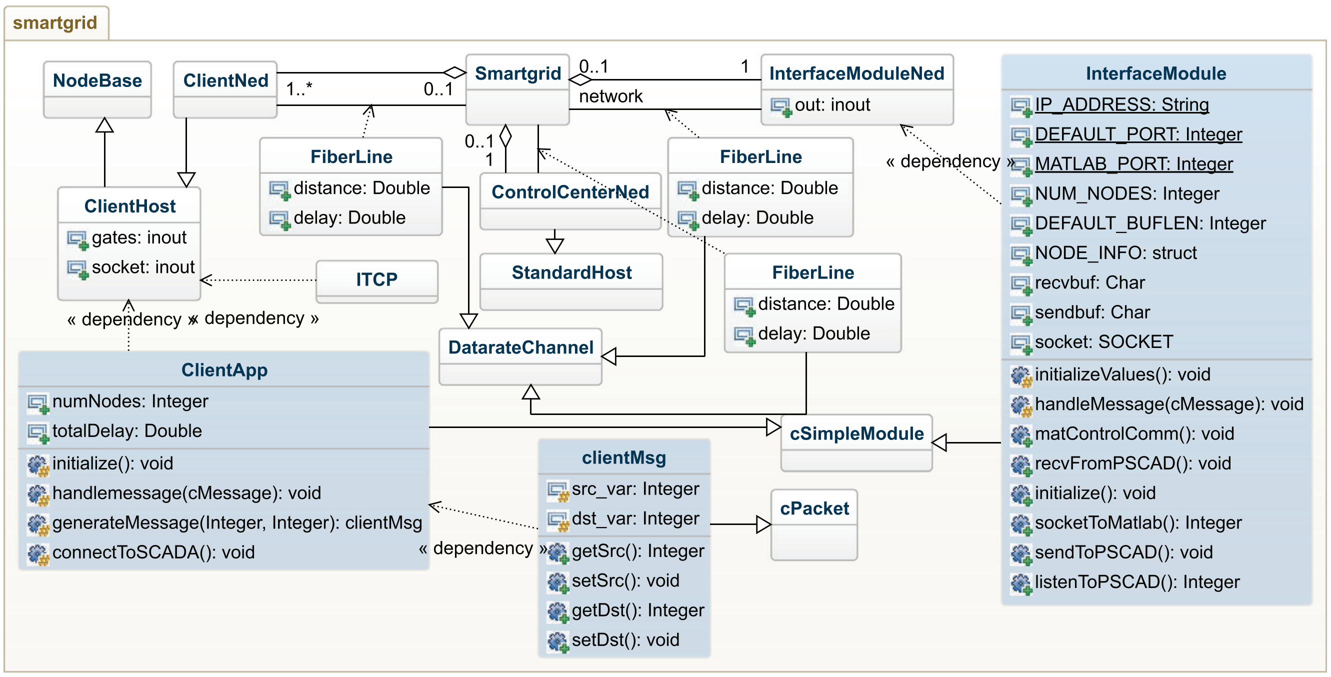

Fig. 6. OMNeT++ co-simulator class diagram.

The *InterfaceModule* receives a packet interrupt from PSCAD via a listening socket *ListenToPSCAD()*. It communicates the messages *ClientMsg* to the appropriate client nodes using the method *handleMessage()*, which in turn calls *recvFromPSCAD()*. Each client node then forwards the message to *ControlCenterNed* which calls its *connectToSCADA()* function. At the *ControlCenterNed* a control logic can be implemented, and then the message is propagated to the destination node. The destination node sends the final message to *InterfaceModule*, where it is finally sent to PSCAD via *sendToPSCAD()*.

5.2. Delay configuration in OMNeT++

OMNeT++ defines three types of channels: ideal channels that have zero delay, delay channels whose model account for propagation delay but not for transmission delay, and data-rate channels which model both propagation and transmission delays. In the developed testbed, data-rate channels can be used and correspondingly channel parameters can be configured to obtain the delay model of the communication channel of interest. A testbed user can configure the following parameters:

1. *Byte-size*: this parameter specifies the number of bytes that are generated by a node per transmission; this parameter can be varied according to a random distributions. For example, a user can use a uniform distribution to sample the byte size. In this case, the parameters of the random distribution are configured in the *omnetpp.ini* file.
2. *Data-rate*: this parameter defines the maximum number of bytes that can be carried by a communication channel in a second. This parameter can be configured in the network section of the *omnetpp.ini* file.
3. *Propagation Delay*: this parameter is related to the distance between the communication nodes and the propagation speed as

$$\text{Delay}_{\text{Propagation}} = \text{Distance}/\text{Speed}_{\text{Propagation}}.$$

Hence, given specific *Data-rate* and *Byte-size* values, the transmission delay can then be calculated using

$$\text{Delay}_{\text{Transmission}} = \textit{Byte-size}/\textit{Data-rate}.$$

Then, the total delay of the communication link is computed as

$$\text{Delay} = \text{Delay}_{\text{Transmission}} + \text{Delay}_{\text{Propagation}}.$$

The total communication delay can also be captured through the simulator API using the defined methods in the *cPacket* class by calculating the difference between packet arrival time and packet creation/sending time. To do that, *getCreationTime* method retrieves the time a packet was created and *getArrivalTime* method returns the time a packet arrived.

5.3. Cyber attacks

Denial of Service (DoS) attacks aim to impact availability of information by either imposing congestion on communication channels and network devices or by causing failure to obstruct information transmission. The impact of DoS attacks is typically modeled as variable delay imposed on the measurements. False Data Injection (FDI) attacks impact the integrity of information to either cause failure/damage or deceive the system to operate in an alternate state that is more beneficial to the adversary [43,44]. FDI attacks include inducing a bias in the measurements, injecting malicious code into controllers or deceiving the system using purposefully delayed measurements (replay attacks). Any cyber attack targeting cyber-physical systems can be described by either DoS,FDI or a combination of both. The developed testbed enables the study of a wide range of cyber attacks.

For simplicity, the proposed testbed supports modeling both DoS and FDI attacks at the interface between PSCAD and OMNeT++ as illustrated in Fig. 7. The cyber events are directly implemented on the corresponding buffer data. The developed co-simulator offers the following implementations of cyber attacks in PSCAD, which are equivalent to attacks implemented in OMNeT++ from an impact perspective on the cyber-physical system:

- DoS attacks are implemented by their impact on the measurements, where measurements are delayed or obstructed. The co-simulator user can specify a range of time delays from which a random delay is selected and is added to the time-stamp of the measurement. This logic can be activated on the send buffer (*sendbuff*) for DoS attacks targeting measurement links, and on the receive buffer (*recvbuff*) for

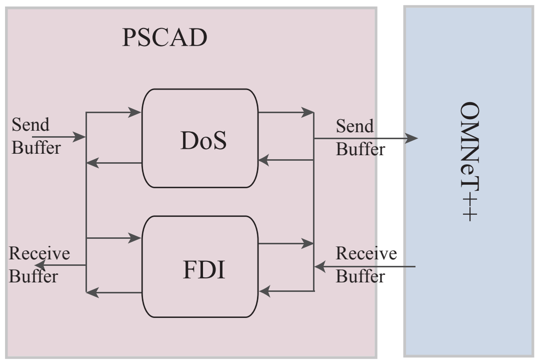

Fig. 7. Cyber attacks implementation.

DoS attacks targeting control and commands links (assuming the user has defined the communication network in a similar fashion).

- FDI attacks translate to a bias introduced into data values. Similar to DoS attacks, FDI attack logic can be activated on the send buffer (*sendbuff*) and/or the receive buffer (*recvbuff*).

Alternatively or concurrently cyber incidents can be modeled in OMNeT + + directly on communication channels and communication network devices.

5.4. Third party software

Third-party software interface enables utilizing extra intelligence such as analytics and optimization tools to improve the cyber-physical system operation. The interface with third-party software such as Matlab has been included in the co-simulator design, and is implemented via a separate network socket initialized from OMNeT + +. The *InterfaceModule* function *socketToMatlab()* establishes a socket communication with Matlab via an intermediate compiler (Visual Studio). The data to be exchanged with Matlab through the socket is handled by *matControlComm()*. The third-party software interface can be disabled if not needed.

The developed testbed can interface with any third-party software provided it can be called and executed from a C + + program. For example, on a Windows based system, this can be implemented in OMNET + + using CreateProcess() or system() to call the third party software.

6. Subsystems interaction

Fig. 8 shows graphical illustrative example to clarify the different interactions between the co-simulation testbed subsystems. The diagram focuses on the sequence of events' interaction between the power simulator, scheduler, and communication simulator as the simulation progresses. The following notation is used:

NU indicates no change in the status of the power simulator actuators. In this case, the progress of the simulation depends on previously-stored values.
t_i represents a time instant in the simulation run time, where $i = 1 \cdots n$ where n is $\leqslant$ end of simulation time.
t_{i+} represents the time-stamp of the communication packet, including total delay, at the end of the communication simulation. Let total delay be denoted as t_{comm} then $t_{i+} = t_i + t_{\text{comm}}$.
E_i represents an event that is generated at time instant t_i from the power simulator and is passed to the communication simulator via the scheduler module.
R_i represents the control command information that is returned from the communication simulator and is meant to be applied on the power simulator at time instant t_{i+}. Let $t_{\text{simulation}}$ denote the power systems simulation time, then R_i may be stored in the database if $t_{i+} > t_{\text{simulation}}$ and is applied when $t_{i+} = t_{\text{simulation}}$.
$\downarrow$ indicates either the scheduler handing over co-simulation to the communication simulator or the case when the power simulator hands over to the scheduler in order to make a decision.
$\uparrow$ indicates either the scheduler handing over co-simulation to the power simulator or the communication simulator handing over to the schedule.
$\rightarrow$ shows progress of the simulation in the testbed simulators or buffer (database) updates in the scheduler module.
$- - \rightarrow$ is the "catch-up" in simulation time in which the power simulator has to run in order to reach to the simulation time at which the scheduler hands over to it.
0|1 the 0 indicates that the sensing device has no information to pass to the communication simulator, and 1 indicates otherwise.

The power simulator, scheduler, and communication simulator interact sequentially via network sockets, where the power simulator via its embedded scheduler acts as a client in the network sockets setup, and the communication simulator is configured as the server. This setup enables the scheduler to control when to interact with the communication simulator. In this section the third-party software subsystem is not included for simplicity.

The diagram in Fig. 8 is based on a power system model with two sensor nodes and one actuator node, and it traces the sensor and actuator *send_flags* and time-stamps. Further, a network model to represent the communication between the three nodes and an additional control node is defined by the testbed user in the communication simulator. The sequence of interactions in the diagram can be described as follows:

1. Sensor S_1 sampling rate indicates a measurement is to be sent to the communication simulator, so it sends an event labeled E_1 to the scheduler at time t_1. Hence, *send_flag* is updated to $(1, 0, t_1)$.
2. The scheduler immediately forwards E_1 to the communication simulator at the same time instant t_1 and the event is removed from the scheduler. The power simulator is *paused* and the scheduler *hands over* simulation to the communication simulator.
3. The communication simulator simulates the communication and returns the response R_1 at time instant t_{1+} to the scheduler. The scheduler stores it in the database and *hands over* simulation to the power simulator.
4. The power simulator resumes simulation, and at the following time-step it checks whether R_1 time-stamp is within the synchronization window; since it is not, the power simulator updates the actuator from the stored default values (history), and there is no new system state update (NU).
5. The simulation of the power simulator proceeds until at time t_2 both

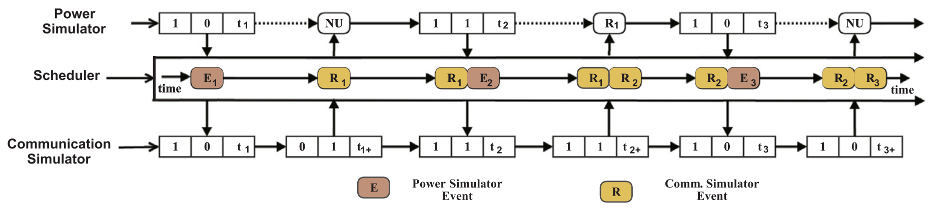

Fig. 8. Subsystems interaction diagram.

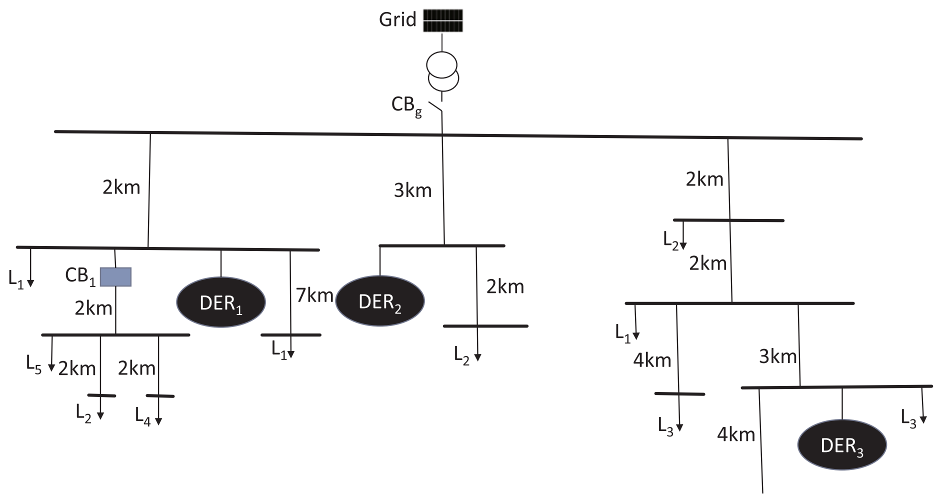

Fig. 9. Microgrid Model.

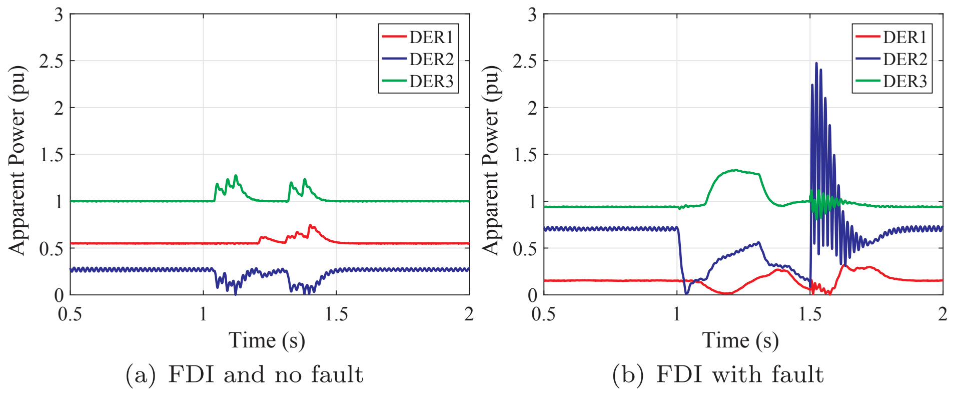

Fig. 10. Impact of cyber attacks.

sensors S_1, S_2 indicate that they wants to communicate with the communication simulator with an event E_2. Thus, *send_flag* is set to $(1, 1, t_2)$.

6. Power simulator pauses and scheduler forwards the event E_2 to the communication simulator (at the same time-step) which *takes over* the simulation.
7. At t_{2+}, the communication simulator returns a control decision R_2. At this same time instant, R_1 time-stamp is found to be within the synchronization window and its value is used to update the state of the actuator. Therefore, only R_2 is left in the scheduler.
8. At t_3, the power simulator generates another event from S_1 and passes E_3 to communication simulator via the scheduler.
9. Finally, at t_{3+}, the communication simulator finishes the simulation and returns the response R_3 which is stored alongside R_2 in the database.
10. The power simulator resumes simulation and checks for an actuator update from the database (an entry with a time-stamp within the current synchronization window); however, since there is none, no change of actuator states and we have no update (NU).

The subsystem interaction model described, illustrates how the developed scheduler is able to effectively coordinate between the two simulators. However, it is noted that the time-step of the power system simulator dictates the upper bound of the synchronization error of the testbed. PSCAD supports time-steps in the range of (1–250) microseconds, hence approximations due to co-simulation interactions are induced, where the power system model may be instructed to proceed without the measurements update until the following time-step. Nonetheless, power system applications that utilize communication are typically concerned about communication delays that are comparable to the power system dynamics being studied. Most of these studies consider a time scale of milliseconds and above, hence the approximations introduced by our testbed synchronization errors are negligible.

6.1. Offline co-simulator case study

The case study considered here is based on the microgrid model in [45] which is illustrated in Fig. 9 and includes three 0.6-kV dispatchable electronically-interfaced DER units and eleven loads which are connected to a 13.8-kV distribution system. The microgrid is operated in islanded mode, i.e., the circuit breaker CB$_g$ is open. A centralized controller is implemented that processes signal measurements of total power to actuate the *set-points* of the different DERs. The impact of an FDI attack is investigated where the adversary aims to manipulate targeted DER operating points above capacity while maintaining controller observed total power unaffected. Two scenarios are considered:

- the attack is activated within the interval 1.1–1.3 s while the microgrid is operating in steady state. Simulation results are shown in Fig. 10(a). The FDI attack causes the DER 2 output to initially fall, rise up and fall again while DERs 1, 3 increase output to balance the load demand. However, this forces DER 3 to operate above its 1.0($p.\ u$) capacity, which could consequently activate the generator protection and drive the microgrid further into instability.
- the adversary monitors and waits for the microgrid to be in a transient or weak state before activating the FDI attack. In this case, FDI is activated at 1 s following the disconnection of a large load. As

DERs are reducing their outputs to adjust to new load demands, the adversary exploits the transient state of the system and is able to drive DER 3 above the $1(p.\,u)$ limit as shown in Fig. 10(b) while the rest of the generators lower their outputs to balance the total power output of the system.

7. Conclusion

Co-simulation of cyber-physical power systems is an important tool to evaluate various cyber-enabled power system components such as distributed control. The implementation of a tool-based offline smart grid co-simulation testbed is described in this work to enable practical cyber security and control studies. The detailed architecture and subsystems interactions of developed testbed are described to facilitate a more involved understanding of co-simulation development concepts. The power simulator PSCAD and open source communication simulator OMNeT++ are integrated using a discrete-events scheduler. A case study on control performance and cyber attacks impact on microgrids using the developed testbed is illustrated.

References

[1] Güngör V, Sahin D, Kocak T, Ergüt S, Buccella C, Cecati C, et al. Smart grid technologies: communication technologies and standards. IEEE Trans Industr Inf 2011;7(4):529–39.

[2] Sabbah A, El-Mougy A, Ibnkahla M. A survey of networking challenges and routing protocols in smart grids. IEEE Trans Industr Inf 2014;10(1):210–21.

[3] Farraj A, Hammad E, Kundur D, A cyber-physical control framework for transient stability in smart grids, IEEE Trans Smart Grid.

[4] Hammad E, Farraj A, Kundur D. Fundamental limits on communication latency for distributed control via electromechanical waves. In: IEEE International Conference on Communications (ICC); 2017. p. 1–6.

[5] Farraj A, Hammad E, Kundur D, A systematic approach to delay-adaptive control design for smart grids. In: IEEE International Conference on Smart Grid Communications (SmartGridComm); 2015. p. 768–73.

[6] Yang C-h, Zhabelova G, Yang C-W, Vyatkin V. Cosimulation environment for event-driven distributed controls of smart grid. IEEE Trans Industr Inf 2013;9(3):1423–35.

[7] Kosek AM, Lunsdorf O, Scherfke S, Gehrke O, Rohjans S. Evaluation of smart grid control strategies in co-simulation integration of IPSYS and mosaik. In: Power Systems Computation Conference (PSCC); 2014. p. 1–7.

[8] IEEE Standards Association. IEEE2030; 2016. < grouper.ieee.org/groups/scc21/2030/2030_index.html > [accessed March 2018].

[9] Basso T, DeBlasio R. Ieee smart grid series of standards ieee 2030 (interoperability) and IEEE 1547 (interconnection) status. Grid-Interop 2011:5–8.

[10] Palensky P, van der Meer A, Lopez C, Joseph A, Pan K. Applied cosimulation of intelligent power systems: implementing hybrid simulators for complex power systems. IEEE Ind Electron Mag 2017;11(2):6–21.

[11] T.M. Association, FMI: Functional Mock-up Interface; 2018. < https://fmi-standard.org/literature/ > [accessed June 2018].

[12] Georg H, Wietfeld C, Müller SC, Rehtanz C. A HLA based simulator architecture for co-simulating ICT based power system control and protection systems. In: Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on, IEEE; 2012. p. 264–69.

[13] Brooks C, Derler P, Lickly B, Lee E. Ptolemy Project; 2018. < http://ptolemy.berkeley.edu/ptolemyII/ > [accessed June 2018].

[14] de Lorraine U, Inria. MECSYCO: Multi-agent Environment for Complex SYstem CO-simulation; 2018. < http://mecsyco.com/ > ; [accessed June 2018].

[15] Mosaik OeV. 2018. < https://mosaik.offis.de/ > [accessed June 2018].

[16] Hopkinson K, Wang X, Giovanini R, Thorp J, Birman K, Coury D. Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components. IEEE Trans Power Syst 2006;21(2):548–58.

[17] Lin H, Sambamoorthy S, Shukla S, Thorp J, Mili L. A study of communication and power system infrastructure interdependence on PMU-based wide area monitoring and protection. In: IEEE Power and Energy Society General Meeting; 2012. p. 1–7.

[18] Fuller JC, Ciraci S, Daily JA, Fisher AR, Hauer M. Communication simulations for power system applications. In: Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES); 2013. p. 1–6.

[19] Nutaro J. Designing power system simulators for the smart grid: combining controls, communications, and electro-mechanical dynamics. In: IEEE Power and Energy Society General Meeting; 2011. p. 1–5.

[20] Anderson D, Zhao C, Hauser C, Venkatasubramanian V, Bakken D, Bose A. A virtual smart grid. IEEE Power Energy Mag 2012;10(1):49–57.

[21] Davis CM, Tate J, Okhravi H, Grier C, Overbye T, Nicol D. Scada cyber security testbed development. In: Power Symposium, 2006. NAPS 2006. 38th North American; 2006. p. 483–88.

[22] Reddi R, Srivastava A. Real time test bed development for power system operation, control and cyber security. In: North American Power Symposium (NAPS); 2010. p. 1–6.

[23] Guo F, Herrera L, Murawski R, Inoa E, Wang C-L, Beauchamp P, Ekici E, Wang J, et al. Comprehensive real-time simulation of the smart grid. IEEE Trans Ind Appl 2013;49(2):899–908.

[24] Hegazi O, Hammad E, Farraj A, Kundur D, IEC-61850 GOOSE Traffic Modeling and Generation. In: IEEE Global Conference on Signal and Information Processing (GlobalSIP); 2017. p. 1100–1104.

[25] Hopkinson K, Wang X, Giovanini R, Thorp J, Birman K, Coury D. Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components. IEEE Trans Power Syst 2006;21(2):548–58.

[26] Lin H, Veda SS, Shukla SS, Mili L, Thorp J. Geco: global event-driven co-simulation framework for interconnected power system and communication network. IEEE Trans Smart Grid 2012;3(3):1444–56.

[27] Liberatore V, Al-Hammouri A. Smart grid communication and co-simulation. In: Energytech, 2011. p. 1–5.

[28] Wei M, Wang W. Greenbench: a benchmark for observing power grid vulnerability under data-centric threats. In: IEEE INFOCOM; 2014. p. 2625–33.

[29] Ross K, Hopkinson K, Pachter M. Using a distributed agent-based communication enabled special protection system to enhance smart grid security. IEEE Trans Smart Grid 2013;4(2):1216–24.

[30] Lin H, Veda SS, Shukla SS, Mili L, Thorp J. Geco: global event-driven co-simulation framework for interconnected power system and communication network. IEEE Trans Smart Grid 2012;3(3):1444–56.

[31] Lin H, Sambamoorthy S, Shukla S, Thorp J, Mili L. Power system and communication network co-simulation for smart grid applications. In: IEEE PES Innovative Smart Grid Technologies (ISGT); 2011. p. 1–6.

[32] Lin H, Deng Y, Shukla S, Thorp J, Mili L. Cyber security impacts on all-PMU state estimator-a case study on co-simulation platform GECO. In: IEEE International Conference on Smart Grid Communications (SmartGridComm); 2012. p. 587–92.

[33] Lévesque M, Xu DQ, Joós G, Maier M. Communications and power distribution network co-simulation for multidisciplinary smart grid experimentations. In: Annual Simulation Symposium; 2012. p. 2.

[34] Levesque M, Xu DQ, Joos G, Maier M. Co-simulation of PEV coordination schemes over a FIWI smart grid communications infrastructure. In: IECON Annual Conference on IEEE Industrial Electronics Society; 2012. p. 2901–906.

[35] Jin D, Zheng Y, Zhu H, Nicol D, Winterrowd L. Virtual time integration of emulation and parallel simulation. In: ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS); 2012. p. 201–10.

[36] Srivastava A, Morris T, Ernster T, Vellaithurai C, Pan S, Adhikari U. Modeling cyber-physical vulnerability of the smart grid with incomplete information. IEEE Trans Smart Grid 2013;4(1):235–44.

[37] Poudel S, Ni Z, Malla N. Real-time cyber physical system testbed for power system security and control. Int J Electr Power Energy Syst 2017;90:124–33.

[38] Bottaccioli L, Estebsari A, Pons E, Bompard E, Macii E, Patti E, et al. A flexible distributed infrastructure for real-time cosimulations in smart grids. IEEE Trans Industr Inf 2017;13(6):3265–74.

[39] Divisions of Manitoba Hydro International Ltd. PSCAD; 2016, < hvdc.ca/pscad/ > [accessed March 2018].

[40] O. Ltd. omnetpp; 2016. < omnetpp.org/ > [accessed March 2018].

[41] SQLite.org. sqlite; 2016. < www.sqlite.org/ > [accessed March 2018].

[42] OpenSim Ltd. OMNeT++ Simulation Manual; 2016 < https://omnetpp.org/doc/omnetpp/manual/ > [accessed March 2018].

[43] Farraj A, Hammad E, Kundur D. A distributed control paradigm for smart grid to address attacks on data integrity and availability. IEEE Trans Signal Inf Process Networks 2018;4(1):70–81.

[44] Hammad E, Khalil AM, Farraj A, Kundur D, Iravani R. A class of switching exploits based on inter-area oscillations. IEEE Transactions on Smart Grid.

[45] Etemadi AH, Davison EJ, Iravani R. A generalized decentralized robust control of islanded microgrids.