4.2 System overview

Our simulation testbed design is influenced by the simulation components used by Wang et al. [13]. We utilize MATLAB/Simulink software [49] for simulating the smart grid substatation model. A key element in this setup is the OpenPLC_Simulink-Interface (OS-Interface), a Simulink C language script provided by Thiago Alves, a developer of the OpenPLC project [50]. This interface enables communication between the MATLAB/Simulink application and the OpenPLC Runtime environment, which we employ to simulate the IED devices.

To generate GOOSE messages based on commands issued by the IED devices within the OpenPLC Runtime environment, we utilize the lib61850 library [51]. This library processes the data passed to it and converts them into GOOSE protocol messages, simulating communication between publishers and subscribers in our model.

Figure 4.1 presents the one-line diagram of our simplified testbed, which forms the basis of our initial simulation tests for performance evaluation before further expansion in section 4.2.6.

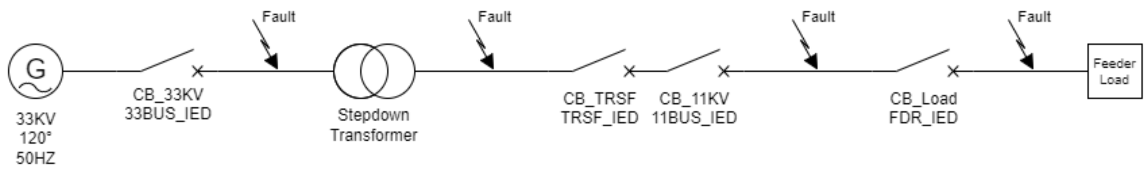

Figure 4.1: 33KV Substation One Line Diagram

The one-line diagram includes the following components and symbols: G represents a three-phase generator rated at 33kV, with a phase angle of 120° and a frequency of 50Hz. CB_33kV is a circuit breaker on the 33kV bus, equipped with IED protection and control capabilities. The Stepdown Transformer reduces the voltage from 33kV to 11kV. CB_TRSF is a transformer-associated circuit breaker with IED capabilities. CB_11kV is an 11kV bus circuit breaker with IED, and CB_Load is a load feeder circuit breaker with IED. The Feeder Load is the final distribution point load, receiving power from the transformer after voltage reduction. Fault Indicators denote potential fault locations in the system, such as short circuits or open circuits.

4.2.1 Testbed Architecture and Design Framework

This section will elaborate on the overall architecture of the testbed, including the hardware and software components, network topology, and the rationale behind the chosen design framework. It will cover the integration of IEC61850 GOOSE protocol elements and how they interact within the testbed environment.

Figure 4.2 illustrates the architecture of the testbed simulation system, demonstrating the interaction between its various components. The testbed consists of two Linux-based virtual machines (VMs). The first VM (VM#1 - 6 Processors @ 3800 MHz / 12GB RAM) hosts the MATLAB/Simulink software, the OS-Interface and the lib61850 subscriber library. The second VM (VM#2 - 4 Processors @ 3800 MHz / 6GB RAM), on the other hand, hosts the OpenPLC-Runtime, the OS-Interface and the lib61850 publisher library. This configuration facilitates effective communication and coordination among the components, ensuring the reliable operation of the simulated smart grid scheme. By closely emulating the operational systems within smart grid networks, our research environment enhances the relevance and applicability of our research findings.

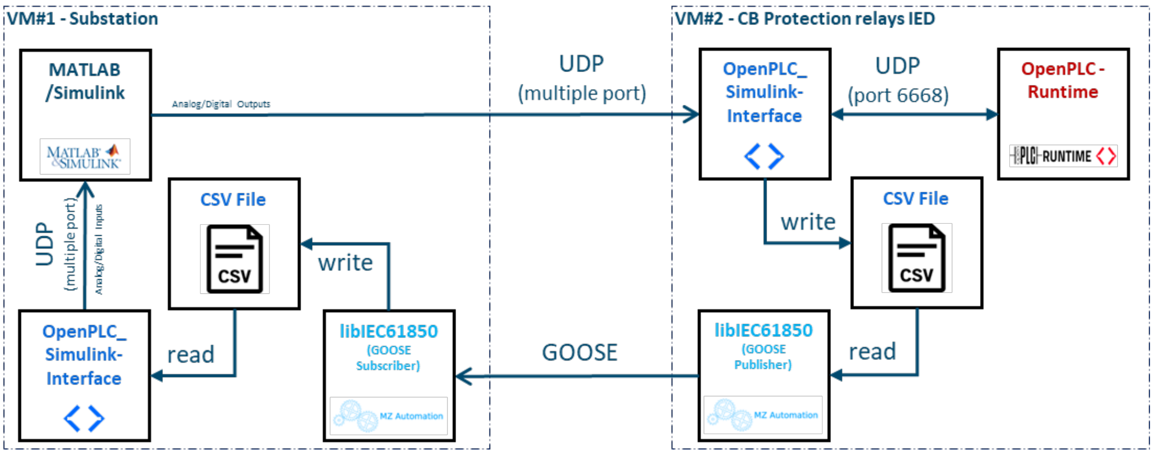

Figure 4.2: Simulation System Architecture

4.2.2 Testbed Setup Configuration

This section will detail the prerequisites for setting up the testbed, including hardware and software requirements, and the step-by-step process involved in preparing the testbed for operation. This includes the configuration of VMs, network devices, IEDs Protection Relays, and simulation software. A copy of configuration files can be found in "https://github.com/ELMASRY-A/Simulation-Testbed"

Testbed Host and Common VMs Configuration

A virtualization software that helps running multiple operating systems in a single host. In our Testbed we will use VirtualBox software [52]. VirtualBox will the host the two VMs running Ubuntu Desktop [53] Linux based OS environment.

For ease on managing the testbed files, we created a /research folder under /home directory, where all the following command/packages installation to be run inside using the the Ubuntu Desktop Terminal.

- git: Distributed version control system [54]
  - sudo apt-get install git
- libiec61850: Open-source library for the IEC61850 protocols [51]
  - git clone https://github.com/mz-automation/libiec61850/
  - cd libiec61850
  - make
- OpenPLC_Simulink-Interfacelibiec61850: Interface between the OpenPLC Simulink driver and a Simulink model using UDP Send and Receive blocks [50]
  - git clone https://github.com/thiagoralves/OpenPLC_Simulink-Interface
- net-tools: networking utilities for Linux including ifconfig [55]
  - sudo apt install net-tools
- tshark: Terminal-based Wireshark [56]

  - sudo apt install tshark

- dbus: Simple interprocess messaging system to call multiple terminals [57]
  - sudo apt install dbus-x11

Testbed VM#1 Setup Configuration

This section will cover configuration applicable only to VM#1 where the following configuration/packages installation are required.

- VM#1 enp0s3 Interface IP Address: 23.23.23.254/24
  - sudo ifconfig enp0s3 23.23.23.254 netmask 255.255.255.0 up

- MATLAB/Simulink: Block diagram environment used to design systems with multidomain models [49]. List of features required when installing MATLAB:
  - Distance Protection Relay
  - DSP System Toolbox
  - Instrument Control Toolbox
  - Parallel Computing Toolbox
  - Signal Processing Toolbox
  - Simscape
  - Simscape Electrical
  - Simulink

Testbed VM#2 Setup Prerequisites

This section will cover configuration applicable only to VM#2 where the following configuration/packages installation are required.

- VM#2 enp0s3 Interface IP Address: 23.23.23.253/24
  - sudo apt install net-tools
  - sudo ifconfig enp0s3 23.23.23.253 netmask 255.255.255.0 up

- OpenPLC Editor on Linux [58]

- OpenPLC Runtime on Linux [59]
  - git clone https://github.com/thiagoralves/OpenPLC_v3.git
  - cd OpenPLC_v3
  - ./install.sh linux

Testbed VM#1 Software Configuration

Once all prerequisites are installed, configuration of installed software has to be carried out. We start with the MATLAB, navigate to /research directory and open Simulink and start a new blank model.

Here is a list of the main blocks and functions we utilised to create a substation model:

- powergui: Set simulation type, simulation parameters, and preferences.
- Three-phase Source: A balanced three-phase voltage source.
- Three-phase Breaker: A three-phase circuit breaker.
- Three-phase Transformer (Two Windings): Uses three single-phase transformers.
- Three-Phase Series RLC Load: A three-phase balanced load.
- Three-Phase Fault: A fault (short-circuit) between any phase and the ground.
- UDP Send: Send a UDP packet to a network address.
- UDP Receive: Receive UDP packets on a given IP port.
- Multimeter: Measures the voltages and currents
- RMS: Compute the root-mean-square (RMS) along the specified dimension

Using the above blocks the following model is created as shown in the 4.3. For configuration of each block parameters and values, refer to Appendix B.

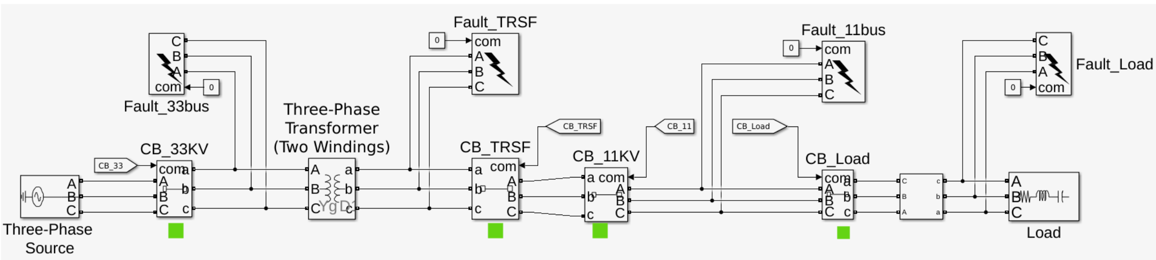

Figure 4.3: MATLAB/Simulink Substation Scheme

At this stage the MATLAB/Simulink substation model is operational and UDP data can be confirmed to be sent out using Wireshark as shown in 4.4

Arrival Time	Source Address	Protocol	Destination Port	Info	Data
22:31:59.873733	23.23.23.254	UDP	16691	59103 → 16691 Len=8	000000000000f03f
22:31:59.871950	23.23.23.254	UDP	16641	59103 → 16641 Len=8	0000000000000000
22:31:59.872026	23.23.23.254	UDP	15591	59103 → 15591 Len=8	000000000000f03f
22:31:59.873733	23.23.23.254	UDP	15541	59103 → 15541 Len=8	0000000000000000
22:31:59.871950	23.23.23.254	UDP	12291	59103 → 12291 Len=8	000000000000f03f
22:31:59.873733	23.23.23.254	UDP	12241	59103 → 12241 Len=8	0000000000000000
22:31:59.871950	23.23.23.254	UDP	11191	59103 → 11191 Len=8	000000000000f03f
22:31:59.873733	23.23.23.254	UDP	11141	59103 → 11141 Len=8	0000000000000000
22:31:59.871950	23.23.23.254	UDP	10000	59103 → 10000 Len=8	0000000000000000

Figure 4.4: MATLAB/Simulink Model Operational

Next we focus on OS-Interface to prepare it to pass data which will received from GOOSE subscriber to MATLAB/Simulink UDP receiver blocks controlling the circuit breaker. As per the system architecture 4.2, we will be reading the GOOSE subscriber data that will be stored in a CSV file to pass it to MATLAB/Simulink UDP Receive blocks. To achieve this, we modified both simlink.cpp and the interface.cfg (copied and renamed to simlink_VM1.cpp and interface_VM1.cfg). OS-Interface simlink_VM1.cpp is modified to read data from CSV (TripFromPRCB.csv), while OS-Interface interface_VM1.cfg is modified to define MATLAB/Simulink UDP Receive blocks parameters. Refer to Appendix C for detailed changes.

Final step is to setup the lib61850 GOOSE Subscriber model to capture GOOSE publisher data and write them to the csv file which will be read by OS-Interface. we navigate to /research/libiec61850/examples/goose_subscriber.c (copied and rename to goose_subscriber_VM1.c) and modify it to write received GOOSE publisher data to TripFromPRCB.csv file. Refer to Appendix C for detailed changes.

Now that VM is setup and ready, the following files need to be put in the /research directory in preparation to start up the whole model; goose_subscriber_VM1.c, interface_VM1.cfg, simlink_VM1.cpp, TripFromPRCB.csv. Then create a run.sh script which need to be run using ”sudo ./run_VM1.sh” to compile and execute the code. Refer to Appendix D for run.sh content.

Testbed VM#2 Software Configuration

Once all prerequisites are installed, configuration of installed software has to be carried out. We start with the OpenPLC Editor, open OpenPLC Editor and start a new blank model and build our Circuit Breaker Protection Relay model. The CBPR model is built using Function Block Program Organization Unit (POU) and will be named CBPR. Figure 4.5 shows the Function Block Diagram (FBD) configuration, which will then be exported to CBPR.st and imported in OpenPLC-Runtime.

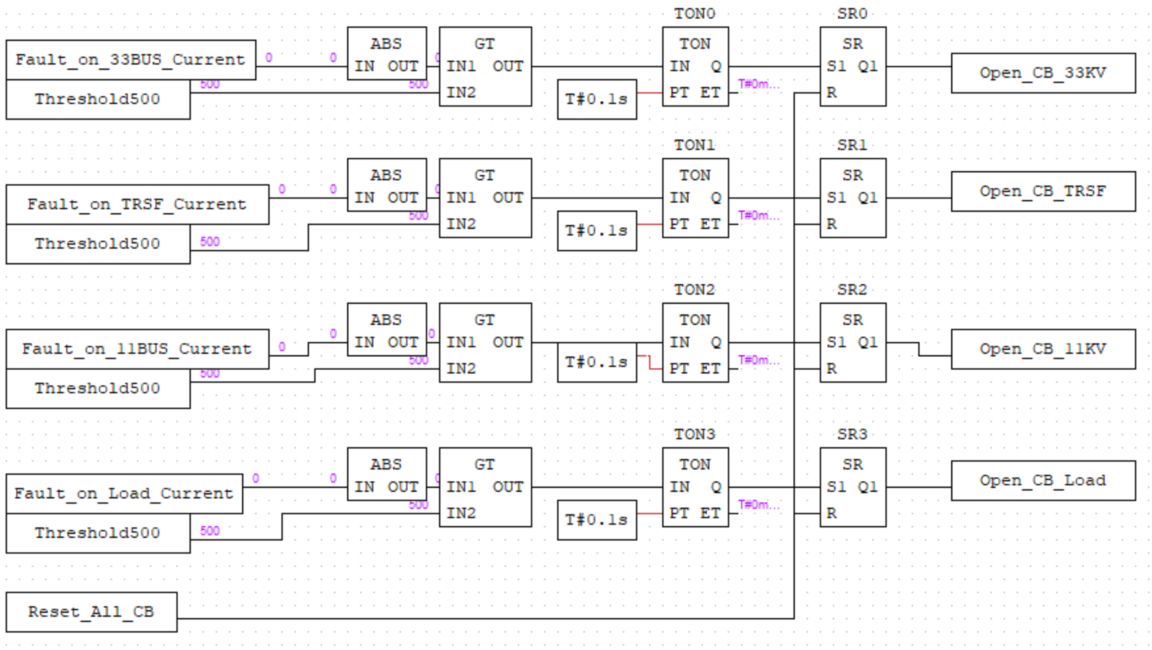

Figure 4.5: OpenPLC Editor FBD

Next we focus on OS-Interface, as per the system architecture 4.2, we will be writing the OpenPLC-runtime Trip commands data and storing it in a CSV file. To achieve this, we modified both simlink.cpp and the interface.cfg (copied and renamed to simlink_VM2.cpp and interface_VM2.cfg). OS-Interface simlink_VM2.cpp is modified to write data to CSV (CBPRTripfromPLC.csv), while OS-Interface interface_VM2.cfg is modified to define OpenPLC-Runtime UDP read ports parameters. Refer to Appendix C for detailed changes.

Final step is to setup the lib61850 GOOSE Publisher model to read trip data from the csv file and publish them using lib61850 GOOSE Publisher model. We navigate to /research/libiec61850/examples/goose_publisher.c (copied and rename to goose_publisher_VM2.c) and modify it to read trip commands data from to CBPRTripfromPLC.csv file. Refer to Appendix C for detailed changes.

Now that VM is setup and ready, the following files need to be put in the /research directory in preparation to start up the whole model; goose_publisher_VM2.c, interface_VM2.cfg, simlink_VM2.cpp, CBPRTripfromPLC.csv. Then run.sh need to be run using ”sudo ./run_VM2.sh” to compile and execute the code. Refer to Appendix D for run.sh content.

Testbed Configuration Summary and Operational Workflow of the Testbed

Communication between the MATLAB/Simulink simulation and the IED devices simulated in the OpenPLC Runtime is established using the OS-Interface. Data is transmitted using the UDP protocol, with predefined ports in the MATLAB /Simulink application using the UDP Send and Receive blocks. We then specify the predefined ports for each IED parameter, IED digital inputs (Port Prefix 101XX), IED digital outputs (Port Prefix 201XX), IED analogue inputs (Port Prefix 301XX) and IED analogue outputs (Port Prefix 401XX, to link them with the OpenPLC structure in a text-based configuration file named interface.cfg. The OS-Interface Simlink script utilizes the interface.cfg parameters to capture UDP communication and pass the data to the OpenPLC Runtime, thus completing the communication from MATLAB/Simulink to the OpenPLC Runtime representing the protection IED devices. In addition, the OS-Interface Simlink script writes the OpenPLC Runtime IED device-issued commands to a CSV file, serving as the input repository for the lib61850 Publisher library.

Thereafter, the lib61850 publisher library reads the CSV file contents and makes them available in the GOOSE protocol to simulate and enable communication between protection IED publishers hosted on VM#2 and the circuit breaker IED subscribers hosted on VM#1. Once the lib61850 subscribers library emulating the circuit breaker IED subscribers receive the GOOSE messages, the message gets written to a CSV file to record the communication. It is then re-transmitted using the UDP protocol to be captured by the MATLAB/Simulink application and reflected in our simulated smart grid scheme.

The protection relays (PR) IED devices are simulated using OpenPLC Project. The OpenPLC Editor facilitates the implementation and configuration of an IED device and its protection mechanism. For instance, in this study, if the current exceeds 500, a fault condition is triggered and a circuit breaker trip is initiated to protect the equipment and isolate the line where the fault has occurred, as illustrated in Figure 4.5. This configuration is then imported into the OpenPLC Runtime, which simulates the configuration and activates the IED. The web server page provides access to the import and enablement of the IED device in simulation mode, as shown in Figure 4.6.

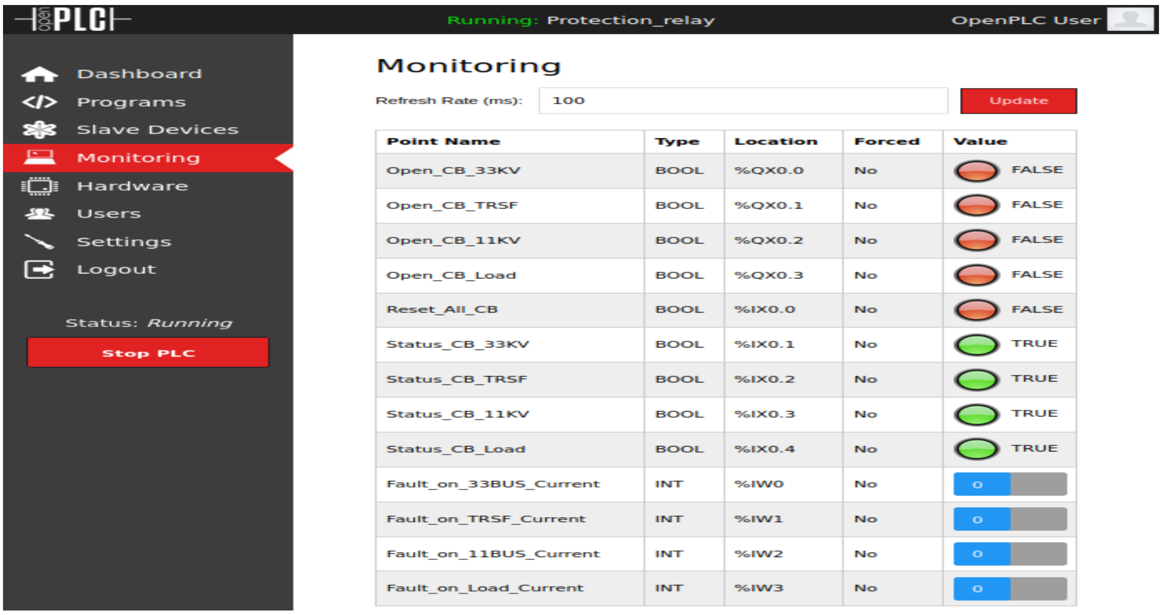

Figure 4.6: OpenPLC Runtime PR IED in Simulation Mode