OpenPLC and lib61850 Smart Grid Testbed: Performance Evaluation and Analysis of GOOSE Communication

Ahmed Elmasry, Abdullatif Albaseer, Mohamed Abdallah
Division of Information and Computing Technology, College of Science and Engineering,
Hamad Bin Khalifa University, Doha, Qatar
{ahel51798, aalbaseer, moabdallah}@hbku.edu.qa

2023 International Symposium on Networks, Computers and Communications (ISNCC) | 979-8-3503-3559-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISNCC58260.2023.10323659

***Abstract*—Datasets are essential for training machine learning algorithms and ensuring accurate classifications. However, obtaining such datasets, especially those relating to IEC61850 GOOSE messages in smart grids, poses a significant challenge for researchers. Privacy regulations and potential misuse of industrial data have compounded this problem, necessitating alternative methods for generating datasets. In this paper, we detail the design and implementation of a simulation testbed that uses MATLAB/Simulink, OpenPLC and the lib61850 library. This testbed simulates a smart grid system and generates IEC61850 GOOSE messages for dataset creation. We assess the testbed performance using a simplified smart grid model with multiple protection relays, also known as Intelligent Electronic Devices (IEDs), enabling us to measure the response time of the protection system under various fault conditions. The testbed is flexible, allowing for design alterations and expansions, which aids researchers in generating diverse datasets to represent various smart grid networks. This flexibility will increase the availability of valuable datasets and enable the broader application of machine-learning techniques in this domain.**

***Index Terms*—Smart Grid, Testbed, IEDs, IEC61850 GOOSE Protocol, MATLAB/Simulink, OpenPLC, lib61850 Library, datasets.**

I. Introduction

The rapid evolution of smart grid systems has led to significant transformations in global power grid infrastructures. These developments have increased energy production and distribution efficiency, reliability and sustainability. Nevertheless, incorporating complex communication networks and IEDs also introduces several cybersecurity challenges. The IEC 61850 standard, which prescribes the communication protocols and data models for electrical substations, has been identified as a critical domain for cybersecurity research [1]. Among its associated communication protocols, GOOSE communication is integral to real-time information exchange and control command transmission between IEDs.

In recent years, various testbed environments have been proposed to facilitate cybersecurity research in smart grid systems. The OpenPLC61850 [2] [3] project has emerged as a promising platform for simulating Programmable Logic Controllers (PLCs) and IEDs, supporting the IEC 61850 standard. However, current OpenPLC61850 implementations primarily focus on the Manufacturing Message Specification (MMS) communication protocol, with limited attention to the testbed system's performance evaluation, particularly for GOOSE communication. Likewise, in [4], the authors concentrated on anomaly detection accuracy, with no emphasis on the time taken by the testbed for GOOSE message generation or the testbed's response rate against real operational scenarios. Furthermore, they did not provide an in-depth explanation of the detailed setup process for the proposed testbed.

In this paper, we aim to fill the gap in the current research by not only introducing a reliable and efficient model for a GOOSE simulation testbed, but also providing a comprehensive, step-by-step guide for its setup process. We aim to enhance the ease of replicating and extending the testbed, thereby fostering wider adoption in the research community. We perform a comprehensive performance evaluation of the OpenPLC and libiec61850 platform, with a particular focus on GOOSE communication. We detail the design and implementation of the testbed environment, which utilizes OpenPLC, the open-source libiec61850 library and other relevant tools to simulate a realistic smart grid network. We also propose a set of performance metrics to evaluate the system's efficiency, reliability and robustness under various fault conditions.

II. Related Work

IEC 61850 establishes a standardized method for connecting various communication profiles. A publisher-subscriber communication model is implemented with the GOOSE and Sampled Value (SV) protocols [5]. Figure 1 shows SV and GOOSE directly interact with Ethernet layers to meet the requirements of a transfer interval of less than 3 ms for critical and rapid responses [6]. This method bypasses the extra time and processing of higher network levels. However, due to the demand for high-speed performance and the limited processing power in IEDs, security measures like publisher authentication and message encryption are often not included [1].

One of the main aspects of the GOOSE Communication mentioned by Apotolov [7] is that GOOSE is not command-based; it operates by reacting to an event indication captured by the publisher and subsequently acted upon by the subscriber, ensuring reliable, high-speed communication, superseding traditional hard-wired signal communication. Figure 2

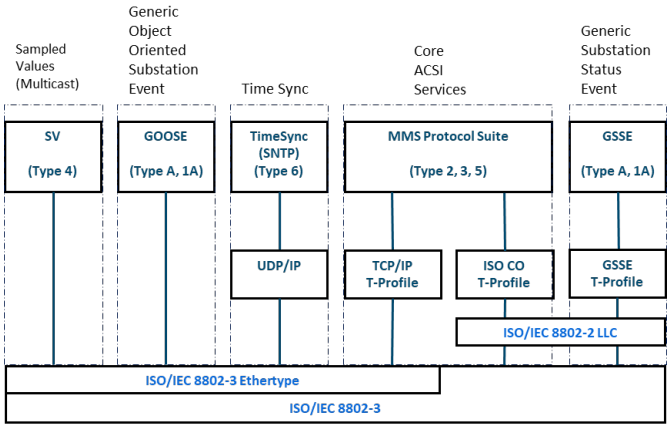

Fig. 1. IEC 61850 Functionality and Associated Communication Profiles [6]

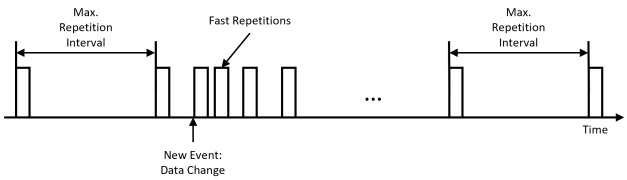

Fig. 2. GOOSE repetition mechanism [7]

illustrates the GOOSE retransmission mechanism, guaranteeing that a single lost message will not impact the system's functionality. Additionally, the mechanism serves as a heartbeat, enabling IED devices to monitor their neighbouring IED devices continuously.

To understand GOOSE messages' structure and functionality, we rely on the detailed structure analysis by C. Kriger et al. [8]. They divide the GOOSE message into three sections: fixed-length section, variable section and user-defined dataset section. Figure 3 depicts the GOOSE message sections as presented in the standard, alongside a packet generated using IEDscout software [9] and captured in Wireshark.

The fixed-length section consists of the Preamble, Start of Frame, Destination Address, Source Address, Tag Protocol Identifier (TPID), Tag Control Information (TCI), Ethernet Type, Application Identifier (APPID), Length, Reserved1 and Reserved2. The Preamble and the Start of the Frame are executed at the Physical layer, while the remaining components occur at the Data Link layer. The variable section is the crux of the GOOSE message, as it contains the data or payload called the Application Protocol Data Unit (APDU). This payload serves as the entry point for adversaries to launch malicious attacks. The PDU begins with a Tag identifier, indicating the class or type of the tag, followed by all parameters and ending with numdaset Entries. The final section, the user-defined dataset section, can encompasses three different data fields: Boolean, integer and bit-string with padding.

The OpenPLC61850 papers by M. M. Roomi et al. [3] [2], discussed the significance of PLCs and IEDs that comply with the IEC 61850 standard, emphasizing that these components

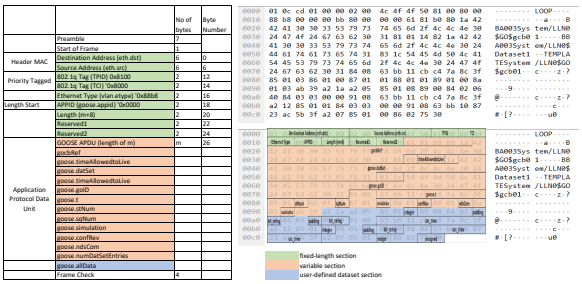

Fig. 3. Sections of the GOOSE message structure frame [8]

are expected to remain at the core of power grid systems for at least another decade. They also touch on the cybersecurity aspects of these devices, noting that security breaches can have detrimental consequences for smart grid networks and their supply. This highlights the necessity for creating testbed environments that allow researchers to investigate cybersecurity breaches and develop solutions without interfering with operational systems and disrupting the power network.

Although the paper acknowledges previous work on OpenPLC61850, it emphasizes that such work has not addressed the performance evaluation of the system. Consequently, the authors primarily focus on presenting the design, implementation and comprehensive evaluation of OpenPLC61850. The testbed was created using OpenPLC61850, the open-source libiec61850 library and a mapping tool. The system's physical topology was simulated using Pandapower, while the communication network topology was implemented through Mininet software. These software tools operated on a single Linux platform. The authors pinpointed a limitation in OpenPLC61850's support, as it only accommodates the primary protocol (IEC 61850 MMS) and not GOOSE or SV. This shortcoming became a motivation for the current research, emphasizing a vital area that necessitates further exploration.

While X. Wang et al. in [4], provided a valuable model for GOOSE simulating substation automation systems, they did not provide a detailed setup process for the testbed. This omission can present a significant hurdle for researchers aiming to replicate or extend their testbed to generate their own data, especially for those new to this field. This gap in detailed setup guidance served as another motive for our research.

By addressing these gaps, our work seeks to empower researchers by lowering the barrier to entry and promoting wider use of the testbed for generating diverse datasets, thereby enriching the broader research community by offering valuable insights into smart grid testbed implementation, discussing its potential as a robust and effective research tool for smart grid system simulations.

III. Simulation Testbed Design

Our simulation testbed design is based on the works of [4]. We utilize MATLAB/Simulink software [10] for the smart grid model simulation. A crucial component in our setup is the OpenPLC_Simulink-Interface (OS-Interface), a Simulink

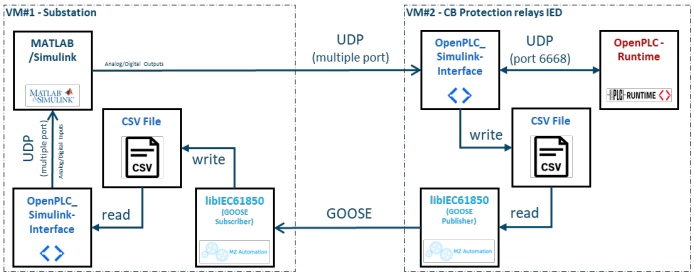

Fig. 4. Simulation System Architecture

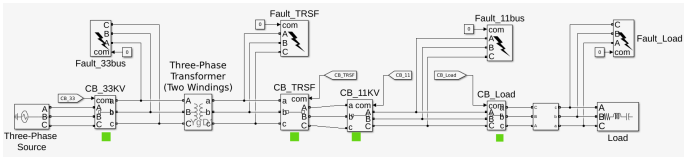

Fig. 5. MATLAB/Simulink Substation Scheme

C language script provided by Thiago Alves, one of the OpenPLC project developers [11]. This interface facilitates communication between the MATLAB /Simulink application and the OpenPLC Runtime environment, which we employ to simulate the IED devices.

For generating GOOSE messages based on the commands issued by the IED devices within the OpenPLC Runtime environment, we use the lib61850 library [12]. This library processes data passed to it and makes them available as GOOSE protocol messages, thereby simulating communication between publishers and subscribers in our model.

A. Testbed System Architecture and Setup

Figure 4 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.

B. Integration and Communication Setup

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

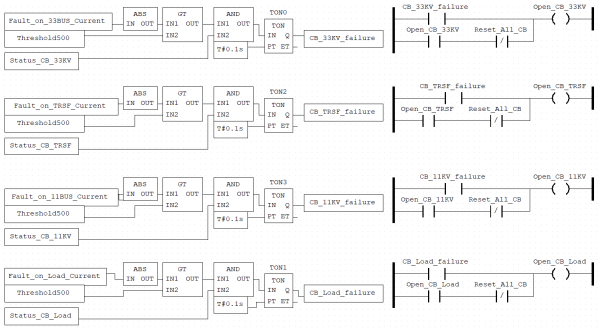

Fig. 6. OpenPLC Editor PR IED configuration

(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.

C. MATLAB/Simulink Smart Grid Model

We simulate a simplified smart grid containing multiple circuit breaker IEDs to safeguard the network from potential faults. The smart grid scheme, as depicted in Figure 5, features a typical 33kV substation generator connected to a transformer (TRFS) through a circuit breaker (CB_33KV). The transformer steps down the voltage to 11kV, which is then connected to the load through another protection circuit breaker (CB_11KV). We have added fault simulation devices at each section of the substation components to simulate faults at various locations within the network and monitor the behaviour of the protection circuit breakers.

D. OpenPLC IED Devices

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

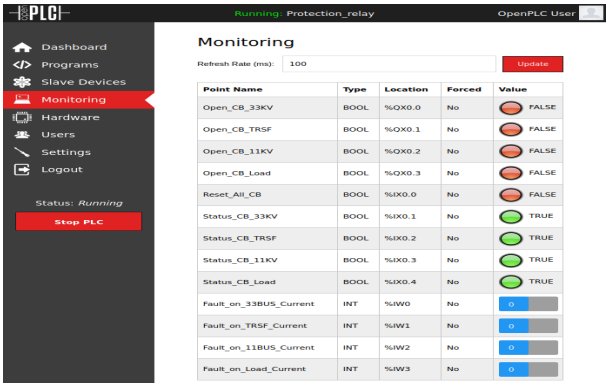

Fig. 7. OpenPLC Runtime PR IED in Simulation Mode

where the fault has occurred, as illustrated in Figure 6. 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 7

IV. Performance Evaluation Metrics

In this section, we detail the evaluation metrics used to assess the performance of the simulation testbed. We measure the effectiveness of the system based on communication latency and reliability.

A. Communication Latency

Communication latency plays the main role in the performance evaluation of the simulation testbed. Particularly, the communication between the IED devices and the MATLAB/Simulink application and the transfer of GOOSE messages between publishers and subscribers. The communication latency was measured from the moment a command is issued by the lib61850 publisher library on VM#2 until it is captured by the lib61850 subscriber library on VM#1. Another critical latency measured was the time taken from when a fault has been simulated in the MATLAB/Simulink application until a circuit breaker trip command has been received back by the MATLAB/Simulink application. The latency was observed and recorded under different fault scenarios. It is influenced by several factors, including script and library processing times, network conditions and VM resource allocation.

B. Reliability

We define reliability as the degree to which the GOOSE messages generated in the simulation environment match the ones produced by an IED device. This includes the consistency and correctness of communication between the MATLAB/Simulink application, the OpenPLC Runtime environment and the lib61850 library. Two aspects of reliability were considered; First, the reliability of the OS-Interface in accurately transmitting data packets from the MATLAB/Simulink application to the OpenPLC Runtime environment. Second, the reliability of the lib61850 library in generating GOOSE

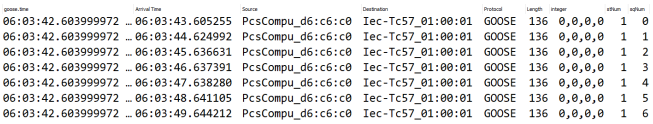

Fig. 8. PR IED GOOSE in Healthy Condition

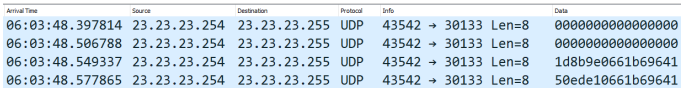

Fig. 9. MATLAB/Simulink UDP Message with Fault Current Value

Point Name	Type	Location	Forced	Value
Open_CB_33KV	BOOL	%QX0.0	No	TRUE
Reset_All_CB	BOOL	%IX0.0	No	FALSE
Status_CB_33KV	BOOL	%IX0.1	No	FALSE
Fault_on_33BUS_Current	INT	%IW0	No	17600

Fig. 10. OpenPLC Runtime initiating Trip Command

messages that accurately mirror the messages produced by IED devices, both in structure and content with no discrepancies.

V. Experiment Setup and Methodology

We first prepared the two VMs and simulated a normal condition with our fully functioning substation model in both MATLAB/Simulink and OpenPLC, where all circuit breakers (CB) are in a closed position and the protection relays in healthy condition monitoring the current as shown in Figures 5 and 7. For demonstration, we have also captured the protection relay IED GOOSE messages in healthy status, Figure 8 showing 0 values in the integer field prior to proceeding with our testing scenarios.

A. Test Scenarios

We simulate the fault scenarios for the 33KV bus, the transformer fault, the 11KV bus fault and the load fault. Then the Wireshark captures are analyzed to check the functionality of our simulation model as well as check the response time from the moment we simulate the faults in the MATLAB/Simulink simulation model to the final element where the circuit breaks receive the open/trip command in the MATLAB/Simulink simulation model. We focus on the 33KV bus fault scenario in this section and detail how it works, then we apply the same method for the transformer fault, the 11KV bus fault and the load fault and discuss their results in the next section V-B

1) 33KV Bus Fault: We started by simulating a fault condition at the 33KV bus using the ”Fault 33_bus” block in the simulation model resulting in extra current being drawn and leaked from the 33KV bus. The Wireshark capture shows the message being sent from the MATLAB/Simulink VM on port 30133, where the changes in the current value of the UDP

goose.time	Arrival Time	Source	Destination	Protocol	Length	Integer	stNum	sqNum
06:03:42.603999972 …	06:03:49.644212	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,0	1	6
06:03:49.659999966 …	06:03:49.669093	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	0
06:03:49.659999966 …	06:03:49.683190	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	1
06:03:49.659999966 …	06:03:49.699650	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	2
06:03:49.659999966 …	06:03:49.736739	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	3
06:03:49.659999966 …	06:03:49.801748	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	4
06:03:49.659999966 …	06:03:49.931423	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	5
06:03:49.659999966 …	06:03:50.199570	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	6
06:03:49.659999966 …	06:03:50.714673	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	7
06:03:49.659999966 …	06:03:51.716010	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	8
06:03:49.659999966 …	06:03:52.716445	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	1,0,0,0	2	9

Fig. 11. GOOSE message initiating 33KV Trip Command

Arrival Time	Source	Destination	Protocol	Info	Data
06:03:49.605191	127.0.0.1	127.0.0.1	UDP	56362 → 20133 Len=2	0000
06:03:49.706146	127.0.0.1	127.0.0.1	UDP	56362 → 20133 Len=2	0000
06:03:49.808287	127.0.0.1	127.0.0.1	UDP	56362 → 20133 Len=2	0100
06:03:49.908884	127.0.0.1	127.0.0.1	UDP	56362 → 20133 Len=2	0100

Fig. 12. OS-Interface UDP message Trip Command to MATLAB/Simulink

message being transmitted to the network as shown in Figure 9.

The OpenPLC Runtime simulator on VM#2 receives the fault value message. Where the OS-Interface gets it and passes it throught port 6668 to the OpenPLC Runtime simulator at location %IW0 as seen in Figure 10 giving an out of range value (-ve value) and the ”open_CB_33KV” command is initiated at location %QX0.0 to trip CB_33KV to isolate the faulty section of the substation simulation model to protect the equipment and the line.

We read the OpenPLC trip command using the OS-Interface, store it in a local csv file and re-transmit it using the lib61850 publisher library. The Wireshark packet of Figure 11 shows the StNum increments demonstrating a change in the GOOSE data as it becomes 2 while the SqNum restarts from 0 again. We can also observe the GOOSE repetition mechanism II, where the protection relay IED broadcasts the GOOSE messages generated with the last change at fast intervals for the first 8 SqNum confirming fast repetitions, then followed by one every second with the SqNum incremental.

The MATLAB/Simulink on VM#1 receives the broadcast GOOSE message from the publisher and processes it by the lib61850 subscriber library. The figure also shows the message received by the subscriber with the time stamp it is received (06:03:49.669093), which is 9 milliseconds after it was published (06:03:49.659999).

Finally, the GOOSE message is written by the lib61850 library to a local CSV file, allowing it to be read by the OS-Interface and re-transmitted as a UDP message over port 20133 to be read by the MATLAB/Simulink application. Figure 12 shows the Wireshark capture of the UDP message where the data field has changed to 0100 carrying the trip command of CB_33KV, marking a period of 1.25 seconds from the time we began simulating the fault in MATLAB/Simulink until the moment we received the trip command back in MATLAB/Simulink.

Figure 13 shows the final element where the MATLAB /Simulink simulation receives the Trip command of CB_33KV and opens the CB_33KV circuit break with the red indication showing its status.

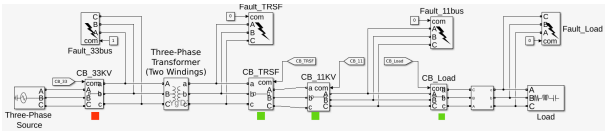

Fig. 13. MATLAB/Simulink CB_33KV Circuit breaker Open

Arrival Time	Source	Destination	Protocol	Info	Data
06:03:58.498560	23.23.23.254	23.23.23.255	UDP	43542 → 30122 Len=8	0000000000000000
06:03:58.534892	23.23.23.254	23.23.23.255	UDP	43542 → 30122 Len=8	9ee63a758d42a43e
06:04:05.468931	23.23.23.254	23.23.23.255	UDP	43542 → 30122 Len=8	6123935b4b77403e
06:04:05.470068	23.23.23.254	23.23.23.255	UDP	43542 → 30122 Len=8	0000000000000000
06:04:09.894531	23.23.23.254	23.23.23.255	UDP	43542 → 30111 Len=8	0000000000000000
06:04:09.960892	23.23.23.254	23.23.23.255	UDP	43542 → 30111 Len=8	3e2c0c70bd20fa3c
06:04:18.401253	23.23.23.254	23.23.23.255	UDP	43542 → 30111 Len=8	12b6f1012d003a3e
06:04:18.405045	23.23.23.254	23.23.23.255	UDP	43542 → 30111 Len=8	0000000000000000
06:04:22.147907	23.23.23.254	23.23.23.255	UDP	43542 → 30100 Len=8	0000000000000000
06:04:22.152112	23.23.23.254	23.23.23.255	UDP	43542 → 30100 Len=8	4e36e9b941a1e33c
06:04:26.066102	23.23.23.254	23.23.23.255	UDP	43542 → 30100 Len=8	de93418a01ae403e
06:04:26.076376	23.23.23.254	23.23.23.255	UDP	43542 → 30100 Len=8	0000000000000000

Fig. 14. MATLAB/Simulink UDP Message with Fault Current Values

goose.time	Arrival Time	Source	Destination	Protocol	Length	Integer	stNum	sqNum
06:03:52.715999960 …	06:04:01.760344	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,0	3	8
06:04:01.759999990 …	06:04:01.765036	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,1,0,0	4	0
06:04:01.759999990 …	06:04:04.864014	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,1,0,0	4	9
06:04:04.863999962 …	06:04:05.874254	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,0	5	0
06:04:04.863999962 …	06:04:12.905770	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,0	5	7
06:04:12.912999987 …	06:04:12.926198	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,1,0	6	0
06:04:12.912999987 …	06:04:15.987914	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,1,0	6	9
06:04:15.987999975 …	06:04:16.990620	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,0	7	0
06:04:15.987999975 …	06:04:25.054200	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,0	7	8
06:04:25.056999981 …	06:04:25.065575	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,1	8	0
06:04:25.056999981 …	06:04:29.101017	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,1	8	10
06:04:29.100999951 …	06:04:30.112155	PcsCompu_d6:c6:c0	Iec-Tc57_01:00:01	GOOSE	136	0,0,0,0	9	0

Fig. 15. GOOSE messages initiating Trip Commands

B. Data Collection

Similarly to the 33KV bus fault scenario V-A1, we simulate the fault scenarios for the transformer fault, the 11KV bus fault and the load fault.

The Wireshark capture shows the message being sent from the MATLAB/Simulink VM on ports 30122,30111 and 30100, where the changes in the current values of the UDP message being transmitted to the network as shown in Figure 14 at different intervals of time when we have simulated each fault scenario.

Figure 15 shows the messages received by the subscribers with the time stamp it is received, which is 5 milliseconds for CB_TRSF, 13 milliseconds for CB_11KV and 8 milliseconds for CB_load after the messages were published. It also shows the StNum increments demonstrating a change in the GOOSE data every time the value changes while the SqNum restarts from 0 every time.

The GOOSE messages get re-transmitted as a UDP message over ports 20122, 20111 and 20100 to be read by the MATLAB/Simulink application. Figure 16 shows the Wireshark capture of the UDP message where the data field changes with the trip commands, marking a period of 3.85 seconds for TRSF fault, 3.7 sec for 11KV Bus fault and 3.2 for the load fault, for the time taken from when we began simulating the fault in MATLAB/Simulink until the moment we received the trip command back in MATLAB/Simulink.

VI. Results and Discussion

This section presents an analysis of our experimental results, the simulation testbed’s performance and the implications for the research community.

Arrival Time	Source	Destination	Protocol	Info	Data
06:04:01.712572	127.0.0.1	127.0.0.1	UDP	43624 → 20122 Len=2	0000
06:04:01.817522	127.0.0.1	127.0.0.1	UDP	43624 → 20122 Len=2	0100
06:04:05.884976	127.0.0.1	127.0.0.1	UDP	43624 → 20122 Len=2	0100
06:04:06.011821	127.0.0.1	127.0.0.1	UDP	43624 → 20122 Len=2	0000
06:04:12.934015	127.0.0.1	127.0.0.1	UDP	37122 → 20111 Len=2	0000
06:04:13.039543	127.0.0.1	127.0.0.1	UDP	37122 → 20111 Len=2	0100
06:04:17.014388	127.0.0.1	127.0.0.1	UDP	37122 → 20111 Len=2	0100
06:04:17.117036	127.0.0.1	127.0.0.1	UDP	37122 → 20111 Len=2	0000
06:04:25.131642	127.0.0.1	127.0.0.1	UDP	51675 → 20100 Len=2	0000
06:04:25.237918	127.0.0.1	127.0.0.1	UDP	51675 → 20100 Len=2	0100
06:04:30.136944	127.0.0.1	127.0.0.1	UDP	51675 → 20100 Len=2	0100
06:04:30.237868	127.0.0.1	127.0.0.1	UDP	51675 → 20100 Len=2	0000

Fig. 16. OS-Interface UDP message Trip Command to MATLAB/Simulink

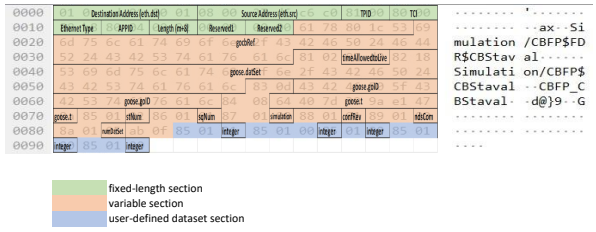

Fig. 17. Sections of the testbed GOOSE message structure frame

A. Communication Latency Results

The GOOSE communication messages are crucial in fault detection and a fast response process. The Wireshark captures revealed that the average time taken for the GOOSE messages to be transmitted and received by the respective subscribers was around 9 milliseconds, based on various conducted fault scenarios. This efficient communication time highlights the effectiveness of the GOOSE protocol in substation automation systems, where rapid fault detection and response are critical to ensure system stability and prevent equipment damage. We also measured the average time from the moment we began simulating the fault in MATLAB/Simulink until the moment we received the trip command back in MATLAB/Simulink. Our measurements indicated an average response time of approximately 3.5 seconds. This latency remains within acceptable limits, demonstrating that the simulation testbed can reliably generate datasets to aid in training machine learning modules for the detection and prevention of adversarial attacks on the GOOSE protocol.

B. Reliability Results

Our preliminary results indicate that both the OS-Interface and the lib61850 library show high reliability, effectively replicating the real-life conditions of GOOSE message generation and transmission. This is evidenced by comparing the message structure shown in Figure 17 with that discussed earlier in section II (Figure3). The fixed-length sections were identical and the variable section contained the same fields but with different sizes and lengths, which is expected. The user-defined dataset section contained the defined data type in our case that was integers only. Moreover, the testbed functioned as expected across all tested scenarios, capturing all necessary data and responding appropriately without any anomalies or deviations from the norm.

VII. Conclusion and Future Work

The simulation testbed provides valuable resources for research into substation automation and GOOSE protocol. It allows cost-effective scenario simulation, facilitates advanced studies and improves system security through machine learning. The testbed is adaptable to various substation sizes, aiding in comprehensive data collection for machine learning training. It also demonstrates easy integration of MATLAB/Simulink and OpenPLC, benefiting sectors such as energy and technology. The testbed's versatility underlines its importance for studying and enhancing substation automation systems. Furthermore, its benefit for the research community extends to generating datasets for studying adversarial attacks.

Acknowledgment

This publication was made possible by NPRP Cluster project (NPRP-C) Twelve (12th) Cycle grant # NPRP12C-33905-SP-67 from the Qatar National Research Fund (a member of Qatar Foundation). The findings herein reflect the work and are solely the responsibility, of the authors.

References

[1] M. G. da Silveira and P. H. Franco, "Iec 61850 network cybersecurity: Mitigating goose message vulnerabilities," in *Proc. 6th Annual PAC World Americas Conf*, 2019, pp. 1–9.

[2] M. M. Roomi, W. S. Ong, D. Mashima, and S. S. Hussain, "Openplc61850: An iec 61850 compatible openplc for smart grid research," 2021.

[3] M. M. Roomi, W. S. Ong, S. S. Hussain, and D. Mashima, "Iec 61850 compatible openplc for cyber attack case studies on smart substation systems," *IEEE Access*, vol. 10, pp. 9164–9173, 2022.

[4] X. Wang, C. Fidge, G. Nourbakhsh, E. Foo, Z. Jadidi, and C. Li, "Anomaly detection for insider attacks from untrusted intelligent electronic devices in substation automation systems," *IEEE Access*, vol. 10, pp. 6629–6649, 2022.

[5] I. E. Commission *et al.*, "Iec 61850: Communication networks and systems in substations," *vol. IEC*, vol. 61850, no. 2, 2013.

[6] R. E. Mackiewicz, "Overview of iec 61850 and benefits," in *2006 IEEE Power Engineering Society General Meeting*. IEEE, 2006, pp. 8–pp.

[7] A. Apostolov, "To goose or not to goose?-that is the question," in *2015 68th Annual Conference for Protective Relay Engineers*. IEEE, 2015, pp. 583–596.

[8] C. Kriger, S. Behardien, and J.-C. Retonda-Modiya, "A detailed analysis of the goose message structure in an iec 61850 standard-based substation automation system," *International Journal of Computers Communications & Control*, vol. 8, no. 5, pp. 708–721, 2013.

[9] OMICRON, "Iedscout software," https://www.omicronenergy.com/, accessed: 2023-03-21.

[10] T. M. Inc., "Matlab version: 9.13.0 (r2022b)," https://www.mathworks.com, Natick, Massachusetts, United States, 2023.

[11] T. Alves, "Openplc simulink interface," https://github.com/thiagoralves/OpenPLC_Simulink-Interface, 2020.

[12] mz automation, "libiec61850," https://github.com/mz-automation/libiec61850, 2018.