IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 29, 2021, accepted March 14, 2021, date of publication March 24, 2021, date of current version April 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068366

Enabling Emulation and Evaluation of IEC 61850 Networks With TITAN

ARTHUR ALBUQUERQUE ZOPELLARO SOARES[1], LEONARDO F. SOARES[1], DOUGLAS P. MATTOS[1], PAULO H. B. S. PINHEIRO[2], SILVIO E. QUINCOZES[1], VINICIUS C. FERREIRA[1], GUILHERME H. APOSTOLO[1], GABRIEL R. CARRARA[1], IGOR M. MORAES[1], CÉLIO ALBUQUERQUE[1], YONA LOPES[1,2], NATALIA C. FERNANDES[1], AND DÉBORA C. MUCHALUAT-SAADE[1]

[1]MídiaCom Lab, Fluminense Federal University, Niterói 24220-900, Brazil
[2]FRIENDS Lab, Fluminense Federal University, Niterói 24220-900, Brazil

Corresponding author: Arthur Albuquerque Zopellaro Soares (arthurazs@midiacom.uff.br)

This work was supported in part by the ANEEL's Research and Development Program under Grant PD-07130-0053/2018, in part by National Council for Scientific and Technological Development (CNPq), in part by Rio de Janeiro State Research Foundation (FAPERJ), in part by Coordination for the Improvement of Higher Education Personnel (CAPES), and in part by CAPES PRINT.

ABSTRACT Sensing and monitoring electrical signals and power device parameters within the smart grid network infrastructure plays a fundamental role in assessing the smart devices' proper functioning. Nevertheless, a key challenge for academic teaching and researching purposes is the elevated cost of real electronic devices, such as current and potential transformers, or even intelligent electronic devices. Therefore, traffic emulators are a valuable solution for the evaluation of new smart grid communication proposals. In this work, we propose TITAN, a tool to support the evaluation of automation systems' communication networks. TITAN enables the emulation of IEC 61850 communication, ranging from data sensing to data acquisition, thus supporting extensive research and development on this fundamental smart grid domain. This tool can interact and communicate with real elements, such as intelligent electronic devices. It enables the emulation of voltage and current data sensing communication, as well as the implementation of different data acquisition schemes by an emulated supervisory system. Our main contributions are: (i) TITAN, a tool with a distributed microservices architecture to execute communication traffic generation tasks; (ii) a user-friendly interface to integrate and manage all components; and (iii) a proof of concept testbed using TITAN and real teleprotection devices. Real case studies using TITAN reveal its feasibility in supporting low-cost testbeds for research, teaching, and testing purposes in sensing and acquisition for automation systems.

INDEX TERMS IEC 61850, IED, intelligent electronic device, SCADA, smart grid communication, supervisory system, traffic generator.

I. INTRODUCTION

With the advent of smart grids, modern information technology systems and communication protocols are integrated into the traditional power grid. This way, several novel features, such as energy load balancing and pricing, fault-tolerance, remote metering, and power quality analysis, are introduced through components called Intelligent Electronic Devices (IEDs). These devices have sensing, computing, and actuating capabilities and may be employed at multiple smart grid domains. Examples are smart meters at the advanced metering infrastructure domain; remote terminal units and programmable logic controllers at the Supervisory Control And Data Acquisitions (SCADAs) domain; and Digital Protection Relays at the substation domain [1].

This modern and complex digital infrastructure requires new protocols and standards to enable communication, monitoring, and control among the many IEDs as well as smart grid applications. The cooperation between industry and academia contributed significantly to promote innovation and standardized access to information. In particular, the IEC 61850 standards specify a number of automation services to communication protocols, named Manufacturing Message Specification (MMS) protocol (*i.e.,* for supervisory messages, reports and remote commands), Sampled Value

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Zhou.

(SV) protocol (*i.e.,* for digital transmission of analog signals such as voltage and current), and Generic Object Oriented Substation Event (GOOSE) protocol (*i.e.*, for fast horizontal communication between IEDs, at the bay level) [2].

Nevertheless, access to real devices are limited in some cases, especially in the digital substation domain, in which devices consist of expensive power equipment, such as three-phase transformers and series capacitor banks may cost above 3 million dollars [3]. Therefore, configuring and testing IEC 61850-compliant IEDs that interact with these devices, as well as training specialized professionals and supporting academic research, is challenging when real equipment is not available. An alternative solution is to create testbeds using virtual IEC 61850-compliant components, such as IEC 61850 traffic analysis and generation tools. However, most of the existing tools are licensed for commercial use only [4]–[10]. Besides, these tools have limited scalability, lack of control of traffic data and rate, and most are single platform.

In this work, we propose Traffic Generator for IEC-61850 Telecommunication Automation Networks (TITAN), a novel tool for enabling traffic emulation in IEC 61850-compliant networks to support the creation of testbeds, ranging from data sensing to data acquisition, thus supporting extensive research and development in the substation domain. In particular, we specify communication modules through a distributed microservice architecture. Each microservice is responsible for implementing a specific IEC 61850 protocol and operation mode. TITAN can interact with real IEC 61850-capable elements and has a user-friendly interface to integrate and manage all components. Moreover, TITAN is flexible, allowing the user to select transmission parameters, to define the number, type and rate of each traffic flow. Our tool also allows the definition of datasets and attribute values in messages, which is essential for fine-grained testing. Note that this work is an extended version of [11] and [12]. The prior version [11] focused on the evaluation of programming languages suitability for implementing traffic generators, whereas the first version of our GOOSE microservice was introduced in [12]. This article introduces our different microservices working in parallel, applications for TITAN, and presents a real case teleprotection communication scenario, as shown in the case study of Direct Transfer Trip (DTT), with 2 different ways of building a testbed for this scenario, replacing real equipment with TITAN.

The remainder of this article is organized as follows. Section II presents the background about IEC 61850 standards and teleprotection schemes. In Section III, we discuss the existing related tools that may be used to build a testbed. Section IV presents TITAN, our proposed IEC 61850 tool developed specifically to be used on testbeds, and its implementation details. In Section V, testbed experiments integrating TITAN and real IEDs are discussed. Finally, in Section VI, we present final remarks.

II. THE IEC 61850 STANDARD AND TELEPROTECTION

IEC 61850 [2], developed by the International Electrotechnical Commission's (IEC) Technical Committee 57, standardizes the communication networks for power utility automation, no matter how heterogeneous the network environment might be. With the evolution of several devices, such as current transformers becoming optical and protective relays becoming micro-processed, IEC 61850 enables their communication through a data network and grants new functionalities to these devices, now called IEDs. The IEDs may be connected to sensors, receiving current values or communication events; and to actuators, acting upon received commands from a supervisory system or from sensed data from another IED.

In order to allow the integration of these devices, IEC 61850 is based on several aspects, mainly: *data modeling* for classes definition and naming conventions for the data to be communicated; and *communication protocols* for information and service model mapping.

A. DATA MODELING

The IEC 61850 standard describes a physical device as a hardware that connects to the network with a specific address and has a set of classes that characterize its behavior [13]. Each physical device consists of at least one Logical Device (LD), which specifies a group with the same characteristics, for example: protection, control, measurement, among others. Then, each logical device consists of a set of Logical Nodes (LNs). Each logical node contains Data Objects (DOs), which have Data Attributes (DAs). Figure 1 shows part of a generic data model for an IED.

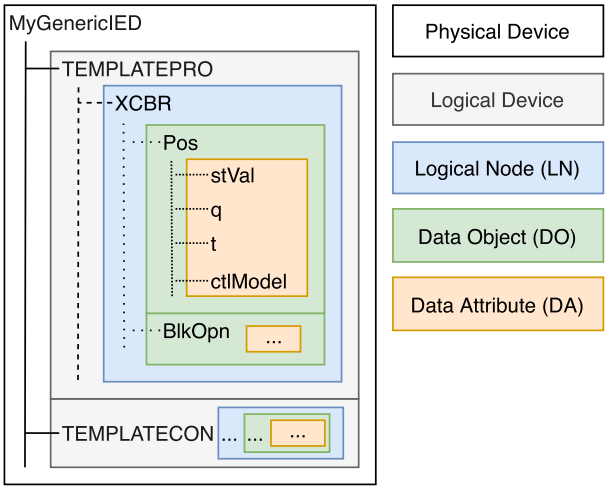

FIGURE 1. Part of a generic Data Model for an IED.

The standard provides a hierarchical view for classifying the functions performed by each device in the network, starting with the physical device until reaching the data attribute as shown in Figure 1. This template model presents part of the hierarchical structure and exemplifies the modeling for

TABLE 1. Example of IEC 61850 message types. Adapted from [14].

Type	Definition	Time Constraint P2/P3 Class	Protocol
1A	Fast messages – Trip	3 ms	GOOSE
1B	Fast messages – Others	20 ms	GOOSE
2	Medium speed messages	100 ms	MMS
3	Low speed messages	500 ms	MMS
4	Raw data messages	3 ms	SV

a circuit breaker that is part of a protection function. This example shows the *TEMPLATEPRO* logical device, which gathers several LNs used for the protection scheme. The *XCBR* logical node gathers several DOs, one of them being *Pos*, which comprises all the positional information (data attributes) for the circuit breaker. The *stVal* DA contains the current status of this circuit breaker (intermediate state, off, on, or bad state), whilst the *ctlModel* DA describes how this circuit breaker should be controlled.[1]

Finally, data attributes from different logical nodes can be grouped into a single *dataset*, in order to be sent as a single message through the network, composing the same control block, *e.g.*, one may group the position of a circuit breaker — *XCBRPosstVal* — and the readings of a current transformer — *TCTR$AmpSv$instMag* — and send both data in one single message.

B. COMMUNICATION PROTOCOLS

The IEC 61850 [2] standards specify three communication protocols: Generic Object Oriented Substation Event (GOOSE), Sampled Value (SV) and Manufacturing Message Specification (MMS). These protocols will be discussed in detail in the following subsections.

Any communication module implementing one of these protocols must guarantee the temporal communication requirements to be met, as established by the IEC 61850 Part 5 [14]. These time requirements are defined according to the type of message and are described in Table 1, which summarizes some of these types. Note that the time constraints shown in the table may vary depending on the scenario it will be used in.

1) GOOSE

The Generic Object Oriented Substation Event (GOOSE) is a communication protocol designed to send events through the network. These messages, or frames, are sent periodically in short periods of time or whenever a new electric event is triggered. GOOSE messages are exchanged between IEDs informing the state of variables so that each IED can take the appropriate action. The GOOSE protocol works asynchronously and is event-oriented.

The publish-subscribe model is used for sending GOOSE frames. Thus, the publisher IED sends frames using a multicast MAC address [15] to a subscriber's group. In order to meet the time requirements established in the IEC 61850 standard, the GOOSE protocol works directly over the Ethernet layer. The structure of the Ethernet frame contains a field titled *ethertype* to indicate the type of protocol. For GOOSE, the value in hexadecimal is 88B8. The standard also defines the GOOSE multicast address range between 01-0C-CD-01-00-00 and 01-0C-CD-01-01-FF. Note that the first four octets are fixed and the last two uniquely identify the multicast groups.

Because the GOOSE protocol works directly over the Ethernet layer, there is no reliability in message delivery. To deal with this lack of reliability, the IEC 61850 standard defines a transmission mechanism to reduce the impact of frame losses. This mechanism sends the same GOOSE message with progressively longer time intervals until a new event happens or it reaches the retransmission limit. The IED calculates the time intervals based on an equation, such as a geometric progression. This method is defined by the equipment supplier, and the minimal value in which it starts depending on the IED's function. Whenever a new event occurs, the time intervals are set to the minimal value, and it is gradually increased over time following the chosen method. Figure 2 shows an example of the behavior for the GOOSE message retransmission intervals on a timeline. In this figure, T0 is the retransmission limit under stable conditions. During (T0) an event occurs. After that, T1, T2 and T3 are the time intervals being increased until reaching T0 again.

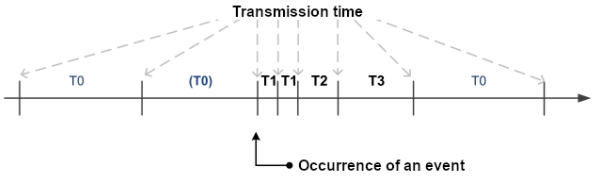

FIGURE 2. GOOSE message transmission intervals, adapted from [16].

Each GOOSE message contains a parameter called *timeAllowedToLive*, informing to the destination the maximum time until the next retransmission. If the next retransmission is not received until the maximum time ends, the destination assumes that there was a problem in the communication.

The IEC 61850 standard also defines a mechanism to provide priority to GOOSE messages based on IEEE 802.1Q. This mechanism enables the use of Virtual Local Area Network (VLAN) between the IEDs with different associated priorities.

Finally, IEC 61850 Part 90-5 defines Routable GOOSE (R-GOOSE), allowing GOOSE communication over the transport layer, enabling to route these messages to other networks, such as facilitating communication between substations.

2) SV

The purpose of SV messages is to send digitized current and voltage samples to IEDs. The SV communication is unilateral and can be done in two ways: the first one is done through

[1] Refer to IEC 61850 standard Part 7-3, "CtlModels definition" for details.

merging units, which are analog to digital signal converters and also merging devices; or through optical current and potential transformers that already send the digitized SV samples to IEDs.

The SV protocol has critical time constraints for generating and sending messages. For this reason, the communication is done via Ethernet. However, the SV protocol does not have a retransmission mechanism such as GOOSE, that is, in case of communication error, the message will not be received by the recipient.

In addition to standardizing the methods for communicating voltage and current values, the SV protocol specifies different sampling rates for protection and measurement applications. Each Ethernet SV frame contains four voltage samples and four current samples. For protection applications, each sample must reach its destination quickly. Therefore, 80 samples are generated and packed in 80 messages per cycle (16,6 ms in a network using 60Hz cycles). For measurement applications, 256 samples are accomplished per cycle. However, the samples are grouped into 8 sets and sent at a rate of 32 messages per cycle. This rate is lower because the transmission time is less important for measurement applications.

The IEC 61850 standard defines the structure for the MAC address used by the SV protocol in a similar way to GOOSE. However, the value of *ethertype* field is 88BA in hexadecimal and the range of the multicast addresses is between 01-0C-CD-04-00-00 and 01-0C-CD-04-01-FF. Like GOOSE, the first four octets are fixed. The IEC 61850 SV standard also supports multiple priorities based on IEEE 802.1Q.

IEC 61850 Part 90-5 also defines Routable SV (R-SV), allowing SV communication over the transport layer, enabling to route these messages to other networks, such as facilitating communication between substations.

3) MMS

The Manufacturing Message Specification (MMS) is a protocol projected for communication between programmable devices in an environment of Computer Integrated Manufacturing [17]. This protocol is standardized by ISO 9506 [18], developed and maintained by Technical Committee ISO 184. MMS protocol was incorporated into the IEC 61850 standard in order to carry out the control and supervision functions of the automation devices of the electrical system. Thus, not only the supervision of the devices is done through the MMS protocol, but also the commands are carried out from the supervisory (local and remote).

MMS protocol may be deployed over Transmission Control Protocol (TCP) or over the Open System Interconnection (OSI) transport standards depending on the application. Usually, MMS serves the Supervisory Control And Data Acquisition (SCADA) system. This protocol is connection-oriented and the exchange of information between devices follows the client-server model and with object oriented modeling.

Usually, control centers and monitoring systems like SCADA are defined as MMS clients. MMS servers run on substation devices, such as IEDs and provide the objects that an MMS client can access. Note that the devices involved in communication can perform the client and server functions simultaneously. Examples are local supervisors (gateways). These supervisors establish a connection to the control center and request information from several devices in the substation at the same time.

The MMS operations carried out by the entities are defined as services. The devices do not need to support all the mechanisms defined by the IEC 61850 standard but there is a subset of required services. The most used MMS services are [15]:

- Initiate: to initiate the communication between client and server.
- Identify: to inform the device identification including the model, vendor, and firmware.
- Read: to read IEC 61850 objects, including Datasets or the data itself.
- Write: to change the attribute values.
- Get: to get a map of objects that there are on the device. This process is called Self-Description.
- Information Report: to send a report on changing variable values.
- Conclude: to end the connection.

C. TELEPROTECTION SCHEMES

Improvements in power system protection might consider communication assisted protection functions, also called teleprotection. In such schemes, a communication network is used to compare the system conditions at the terminals of the protected equipment, providing selective high-speed fault clearance [19].

A guide for protection functions and teleprotection schemes that must be used for each type of protected equipment is presented in [20]. For instance, for transmission line protection, some protection functions that must be used include distance protection for phase and ground elements (21/21N), ground directional overcurrent (67N), loss of potential, switch onto fault, breaker failure (50BF), among others.

Teleprotection schemes are Direct Underreaching Transfer Trip (DUTT), Permissive Underreaching Transfer Trip (PUTT), Permissive Overreaching Transfer Trip (POTT), Directional Comparison Blocking (DCB), and Directional Comparison Unblocking (DCUB) [21]. These schemes establish a communication channel between IEDs, allowing trip schemes to be interconnected through the exchange of information on the IEDs' logical states, enabling the comparison of their responses and determination of the correct sense of the fault. This results in a considerable reduction in the fault extinction time [22].

Electric utilities better perform on transmission line protection implementing teleprotection schemes such as Direct Transfer Trip (DTT) and POTT, thus providing safety and reliability to the power grid. The DTT technique consists of sending/receiving trip signals via a communication link

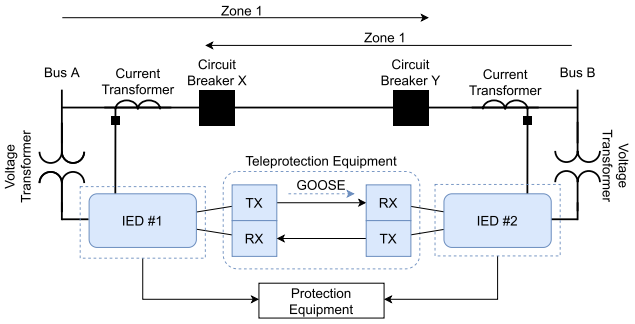

FIGURE 3. Direct Transfer Trip Scheme. TX means transmit, and RX means receive.

between substations [23], which can be achieved using the GOOSE protocol. When a fault is detected by the local IED, it sends a trip signal to its respective circuit-breaker and to the circuit-breaker located at the remote bus, as shown in Figure 3. As it is not a permissive scheme when the remote circuit-breaker receives the trip signal via a communication channel, it opens instantaneously. The most common protection functions associated with DTT are distance protection using zone 1, overcurrent protection, breaker failure, and series capacitor bank protection.

III. RELATED WORK

This section discusses the existing academic and commercial tools, that aim to enable the performance testing and communication monitoring of IEC 61850-based networks. Specifically, we present a brief overview and compare the related tools that support GOOSE, SV, and MMS protocols. Note that while some discussed tools support traffic generation, others are designed for data capturing and analysis purposes only. While traffic analysis is essential for evaluating the IEC 61850-based testbed behavior, tools focused on traffic generation may be used in place of real equipment to reduce deployment costs.

First, we survey the related work and then focus on the tools capable of generating IEC 61850-compliant traffic. As a metric for comparing these tools, we use the following criteria: (1) it allows changing the message payload, *e.g.*, sending messages with a field value *True*, then sending them with the value *False*; (2) it allows the remote management of different hosts, *e.g.*, starting a GOOSE traffic at host 1 and stopping an SV traffic at host 2; (3) it allows test automation, *e.g.*, sending messages with a value *True* for 10 seconds, then changing this value to *False*; and (4) it allows emulating different hosts in a single host, *e.g.*, starting two GOOSE traffic flows with different source addresses and diverse datasets.

A. ACADEMIC TOOLS

The GEESE 2.0 [24] packet generator consists of a desktop implementation based on a Python monolithic application designed for the Linux operating system. It was developed for academic purposes and has an intuitive graphical interface, allowing a quick configuration of a set of parameters (*e.g.*, GOOSE *DataSets*, Physical Device, output network interface) to support the GOOSE traffic generation. However, this tool does not allow setting important network information such as the source and destination MAC addresses, as well as the flow priority. The generation of SV and MMS traffic is not supported by GEESE 2.0.

IEDExplorer [25] is an monolithic open-source tool written in C# programming language. It was developed for academic purposes, aiming at supporting communication tests involving IEDs based on the IEC 61850 standard. Currently, at version 0.79k, this tool is only supported by the Windows operating system. Besides the GOOSE traffic generation and capturing features, the IEDExplorer also supports the emulation of an MMS client that is able to communicate through MMS services with IEDs running an MMS server. Among the IEDExplorer functionalities, an interesting feature is the capability of Substation Configuration Language (SCL) import, which is a file that describes the communication traffics of a smart grid. This allows the creation of GOOSE messages according to the real substation configuration. The manual creation of GOOSE messages for injection into the network is also possible, allowing it to generate several traffic flows emulating different sources.

libIEC61850[2] is an open-source library written in C programming language. Its goal is to provide a portable solution for implementing MMS, GOOSE, and SV protocols into embedded systems and micro-controllers, supporting Windows, MacOS and Linux operating system. Each protocol can be implemented and compiled as a standalone client or server, but there is no native solution for remotely managing each traffic generation instance. It is licensed under GPLv3, meaning it might be used freely for academic purposes but has to be bought for commercial use. As libIEC61850 is a library, the traffic generation must be coded and compiled to meet the user's goal. Also, a Graphical User Interface (GUI) is out of scope for the libIEC61850.

B. COMMERCIAL TOOLS

SimulGOOSE [10] is a commercial tool designed by a Brazillian company called Conprove Engenharia to allow publishing (injection) and subscribing (capture) of GOOSE messages in the network in order to test equipment compatible with IEC 61850. In addition to viewing the message data on the computer screen, it is also possible to generate GOOSE traffic to send *DataObjects*, which had their attributes extracted from previously captured GOOSE messages. It allows generating several traffic flows emulating different sources. The creation of GOOSE messages can be done either manually, by specifying their *DataSets*, or by importing SCD files. SimulGOOSE is a monolithic tool and it runs over the Windows operating system.

IEDScout [9] is a commercial tool designed by an Austrian company called OMICRON to allow capturing MMS and GOOSE messages, importing SCL files, and visualizing message data captured between IEDs. The tool is intended

[2]By MZ Automation, available at https://libiec61850.com/libiec61850.

TABLE 2. IEC 61850-compliant traffic generation (Gen.) and analysis (Ana.) tools comparison.

Tool	Platform	License of Use	Architecture	Traffic		Supported Protocols		
				Gen.	Ana.	MMS	GOOSE	SV
GEESE 2.0 [24]	Linux	Academic	Monolithic	Yes	No	No	Yes	No
IEDExplorer [25]	Windows	Academic	Monolithic	Yes	Yes	Yes	Yes	No
libIEC61850	Multi-plat.	Academic and Commercial	Monolithic	Yes	No	Yes	Yes	Yes
SimulGOOSE [10]	Windows	Commercial	Monolithic	Yes	Yes	No	Yes	No
IEDScout [9]	Windows	Commercial	Monolithic	Yes	Yes	Yes	Yes	No
SVScout [8]	Windows	Commercial	Monolithic	No	Yes	No	No	Yes
ITT600 SA Explorer [7]	Windows	Commercial	Monolithic	No	Yes	Yes	Yes	Yes
61850 TesT Software [6]	Windows	Commercial	Monolithic	Yes	Yes	Yes	Yes	Yes
Triangle Test Suite Pro [5]	Multi-plat.	Commercial	Monolithic	Yes	Yes	Yes	Yes	Yes
SmartGridware [4]	Multi-plat.	Commercial	Monolithic	Yes	No	Yes	Yes	Yes
TITAN	**Multi-plat.**	**Academic and Commercial**	**Microservices**	**Yes**	**No**	**Yes**	**Yes**	**Yes**

for testing and commissioning and allows the emulation of an IED according to the information based on the imported file. In addition, it allows changing the attributes of *DataObjects*. With a friendly graphical interface, it is possible to observe electrical measurements, including phasors visualization. IEDScout is also a monolithic tool and it runs over the Windows operating system.

SVScout [8] is a commercial tool also designed by OMICRON to allow capturing and analysis of SV messages. It is possible to observe through its graphical interface waveforms obtained from the samples sent in the payload of SV messages. The voltage and current measurements can be visualized through a phasor diagram with the effective value of each phase, or in the form of a table. SVScout is a monolithic tool that runs on the Windows operating system.

ITT600 SA Explorer [7] is a commercial tool designed by a Swedish company called ABB for easy diagnosis and troubleshooting of IEC 61850 compliant substation automation systems and applications. It is capable of importing IED configurations from SCL files and showing this information in a friendly way. It also supports capturing GOOSE, MMS, and SV messages. Similar to SVScout, the ITT600 has features related to SV messages for visualizing phasor and waveform data from the captured data. ITT600 is a monolithic tool that runs on the Windows operating system and is focused on data analysis, not supporting traffic generation.

61850 TesT Software [6] is a tool designed by an American company called DOBLE Engineering to simulate, monitor, and test IEC 61850 compliant devices. The tool emulates the Client/Server communication to exchange MMS messages. In addition, it allows capturing GOOSE and SV messages as well as importing SCL files. IED data is shown in a tree structure and in a table view. Through scripts, 61850 TesT Software allows test automation for more complex testing scenarios, for example, generating GOOSE traffic according to what the host is reading from another IEDs. It is also a monolithic commercial tool that runs on the Windows operating system.

Triangle Test Suite Pro [5] is a commercial tool designed by an the American company called Triangle MicroWorks for testing IEDs in the lab, performing troubleshooting, and assisting in the commissioning process. It supports SCL file import, GOOSE publishing and subscribing operations, SV tracking, and data visualization, as well full capabilities of MMS (Client/Server). It facilitates test automation through scripts or setting changes periodically and also allows emulating several traffic flows with different sources. Triangle Test Suite Pro is a monolithic commercial tool that runs on multiple operating systems. Specifically, there is an ANSI-C Library designed for development of Windows based applications, a C++ Library designed for creating IEC 61850 applications on multiple platforms, and an ANSI-C Library designed for high performance embedded IEDs or clients that are built on Real Time OS, Linux or Windows.

SmartGridware IED Simulator [4] is a commercial tool designed by an American company called Monfox to allow the simulation of IEDs. It provides support for MMS client/server as well as GOOSE profile and SV. It comprises a monolithic multi-platform application developed in Java that runs as a web server on the local system and provides a browser based graphical user interface for the simulation of IEC 61850 Server instances. By running as a web server, the host becomes accessible remotely, facilitating the management of several traffic generators from a single host.

C. TOOL COMPARISON AND DISCUSSION

As shown in Table 2, most existing tools are designed with commercial purposes and support only the Windows operating system. Some academic tools do not fully support IEC 61850 capabilities (*i.e.,* supporting MMS, SV, and GOOSE protocols simultaneously). In particular, SV protocol is only supported by libIEC61850 and other commercial tools. Most of the surveyed tools are single-platform. Even though support for an open-source operating system, such as Linux, may be desirable because it potentially decreases deployment costs, the ability to run on several different platforms is more important [26].

Most IEC 61850 compliant tools use monolithic architecture. Even though a monolithic tool may be easier to install and manage, the user is required to install all the

TABLE 3. IEC 61850-compliant traffic generation tools functionalities.

Tool	Payload Modification	Remote Management	Test Automation	Different Sources
GEESE 2.0 [24]	No	No	No	No
IEDExplorer [25]	Yes	No	No	Yes
libIEC61850	Yes	No	Yes	Yes
SimulGOOSE [10]	Yes	No	No	Yes
IEDScout [9]	Yes	No	No	No
61850 TesT [6]	Yes	No	Yes	No
Triangle Test Suite Pro [5]	Yes	No	Yes	Yes
SmartGridware [4]	Yes	Yes	No	Yes
TITAN	**Yes**	**Yes**	**Yes**	**Yes**

functionalities as a single piece of software instead of installing/running only the necessary features for each device in the testbed. That implies excessive disk and sometimes unnecessary CPU usage.

Table 3 compiles the features of the tools capable of generating IEC 61850 traffic. Modifying the messages' payload is essential to adjust the traffic generator to better fit the desired scenario.

With the increasing number of people working from home, full remote access to the testbed and its software is desired. Only SmartGridware and TITAN allow remote management of the traffic generators. Test automation is another desired feature. As the testbed scales, automation facilitates test reproducibility. Finally, the capability of emulating several traffic sources using a single host is desired as it reduces the testbed deployment costs.

TITAN stands out by being multi-platform and supporting the generation and transmission of MMS, GOOSE, and SV traffic. libIEC61850 might be the solution that closer achieves TITAN's goal but still misses two important aspects: a GUI for users with no programming knowledge; and remote management for several spread instances of traffic generation. Our proposal takes advantage of the microservices architecture to ensure a better scalability in traffic generation and also facilitating test automation. More details are provided in Section IV.

IV. TITAN - TRAFFIC GENERATOR FOR IEC-61850 524 TELECOMMUNICATION AUTOMATION NETWORKS

As described in Section II, IEC 61850 comprises three very specific communication protocols. In that sense, there is no reason for a communication tool compliant with the IEC 61850 to be a single piece of software. On the other hand, there is no much coherence in developing 3 different tools from scratch with the same objective in mind: emulating a IEC 61850-based communication testbed.

Previous traffic generators are either built as a protocol-specific or general-purpose piece of software. Either way, it is quite complex to build a communication testbed using those software. When dealing with protocol-specific tools, there is the need to manage different software, each for emulating a specific protocol as needed. Then again, for general-purpose tools, there is the need to embedded all protocols in each device, which might not be advisable for low-end equipment.

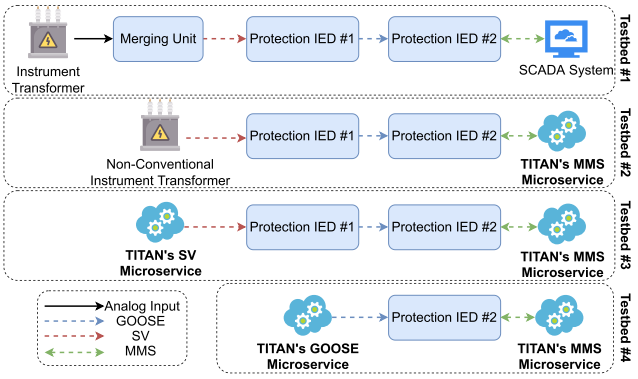

FIGURE 4. The same conceptual testbed but with different deployments of TITAN.

A. APPLICATIONS

A tool for emulating the communication on an IEC 61850 testbed may be used in several different scenarios, such as for academic researching, teaching, and testing purposes. TITAN can be used for: teaching the IEC 61850 standard and concepts, researching novel teleprotection philosophies; testing the communication network; testing real equipment; and researching novel smart grid proposals based on IEC 61850 altogether. Figure 4 shows one conceptual testbed with real equipment being replaced by TITAN.

Preparing IEDs for a particular scenario may not be simple. Each IED supplier has its own proprietary software for configuring it, meaning the user has to have some experience with each proprietary software. As the traffic generated by each IED is virtually the same,[3] by using TITAN, the user would need experience with only one software. So researching, teaching, and testing becomes straightforward.

By using TITAN, it is effortless to test novel teleprotection philosophies, while legacy IED is not capable of interacting with novel teleprotection solutions, TITAN may emulate any necessary traffic to research new scenarios.

With TITAN, testing the network becomes an easy task. Instead of forcing several IEDs to avalanche the network,[4]

[3]Based on IEC 61850.

[4]Forcing several new events on each IED.

TITAN facilitates this process by allowing the user to create a considerable number of messages using a single device. This is very important when a network is being designed for teleprotection with GOOSE messages being time restricted to 3 milliseconds. As TITAN generates real messages, it can be integrated with a ns-3 simulation using TAP,[5] which creates a bridge between real hosts and the simulated network. The same is true for OMNeT++ using OverSim.[6] Finally, TITAN may be used directly with Mininet, a network emulator.

Remote access to the testbed and its devices are worthwhile, even more now with the increasing number people working from home. In spite of IEDs having means to remotely manage their behavior, it is usually done through proprietary software and may not allow managing more than one at the same time. Another issue arises when dealing with IEDs from different suppliers, where the user has to use several software, one for each IED supplier. With TITAN, the user can easily manage all devices running its microservices from a single GUI.

Note that using TITAN in place of real IEDs (sensors or actuators) lower the testbed's deployment cost — saving thousands of dollars — for the purposes previously listed, although it does not mean that TITAN should replace real equipment under real power grid scenarios.

B. ARCHITECTURE

Taking those matters into account, we propose the Traffic Generator for IEC-61850 Telecommunication Automation Networks (TITAN), a microservice-oriented tool as seen in Figure 5.

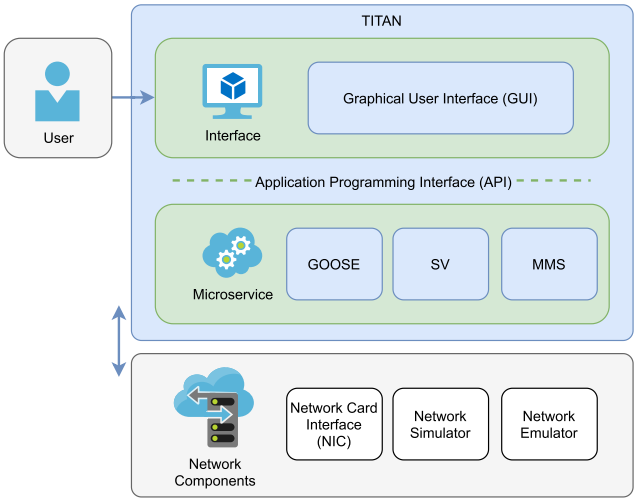

FIGURE 5. TITAN's Architecture.

Even though a monolithic tool may be easier to install and manage, the user is required to install all the functionalities as a single piece of software instead of installing/running only the necessary features for each device in the testbed. That implies excessive disk and sometimes unnecessary CPU usage. A microservice-oriented tool enables the user to install only the essential service in each device of a testbed while also allowing the remote control of these services by a unified GUI through an Application Programming Interface (API).

Our proposal takes advantage of the microservices architecture to ensure a better scalability in traffic generation and also facilitating test automation. Our tool consists of three microservices related to IEC 61850 communication protocols, and a centralized GUI. These microservices are independent, meaning they should be able to run alone in a single device, but might also communicate with the GUI through an API.

The user may manipulate the data model and traffic generation through a user-friendly interface. The GUI enables the user to model logical devices and datasets, but also to manage TITAN's microservices. Each microservice receives a pre-processed data and is told what to do with it. The microservice parses the data according to the protocol and sends the emulated traffic directly to the network components – e.g., a network emulator or even a Network Interface Card (NIC) connected to a testbed.

The SV microservice is used to emulate the communication of sensed data, such as voltage and current values. It receives a data structure followed by their values and necessary information to emulate the data frame, such as source and destination addresses.

The GOOSE microservice is used to emulate the communication of sensed events, such as teleprotection data. It receives a data structure followed by their values and necessary information to emulate the data frame, such as source and destination addresses.

Finally, the MMS microservice is used to emulate the data acquisition process, such as reading data from IEDs. It receives information of which type of data to acquire – e.g., the identification of an IED, its data structure, or one specific attribute.

C. IMPLEMENTATION

In this section, we present the implementation aspects of TITAN, as well as its components. Although GOOSE and SV protocols are designed to meet different goals, they have similar implementation requirements and restrictions. Therefore, we describe the traffic generation for both protocols together. Then, we introduce our MMS traffic generator. Finally, we present the implementation details regarding the graphical interface.

1) TITAN's GOOSE AND SV MICROSERVICES

The IEC 61850 standard establishes hard temporal constraints to GOOSE and SV protocols. Therefore, a suitable programming language should be used to achieve high performance. In our previous work [11], we performed an assessment considering both Python and C languages to traffic generation. As expected, the C programming language outperformed Python. Thus, based on this analysis, C was

[5] Avaliable at https://www.nsnam.org/docs/models/html/tap.html
[6] Available at https://omnetpp.org/download-items/OverSim.html

TABLE 4. GOOSE and SV endpoints.

Functionality	Method	Input	Output
Create GOOSE/SV flow	Post	Create parameters (see Table 5)	Flow identifier
Start/Stop GOOSE/SV flow	Post	Flow identifier	Flow status

chosen to be used in the GOOSE and SV microservices. Besides, the frames are built following the Abstract Syntax Notation One (ASN.1) [27] standard (*i.e.,* GOOSE and SV data structures are defined and compiled by an ASN.1 compiler). This design decision allows the message structures to be defined regardless of the programming language used. Moreover, other encodings to the same definition may be supported in future releases through ASN.1 libraries. The implemented GOOSE and SV microservices communicate to the GUI through a Representational State Transfer (REST) API. In particular, each microservice has a *create* and *start/stop* endpoint.

According to the Table 4, the *create* functionalities require *create parameters* as input for building GOOSE/SV flows. These parameters are listed in Table 5. There are common parameters in both protocols, such as source and destination addresses, application identifier, and frame identifier. Besides, each protocol has its own specific parameters. In SV protocol, there are also optional parameters such as *frame identifier* and *sending the reference time* flag. A *flow identifier* is returned as output to *create* requests. Once created, flows can be started or stopped.

The *start* and *stop* functionalities require the *flow identifier* as input. When the traffic generation is started, the output interface and port must be also informed. Those information are used together with the parameters of Table 5 to generate the traffic. As return, the *flow status* is given. Whenever a flow is started, the respective microservice triggers the traffic generation procedure. This generates GOOSE/SV traffic and transmits it through the network interface.

2) TITAN's MMS MICROSERVICE

In contrast to GOOSE and SV, MMS messages are not time-critical. Moreover, unlike GOOSE and SV, the MMS protocol is implemented over TCP. Therefore, Python 3.7 was chosen as the programming language to implement the MMS generation because there are native Python libraries that simplify TCP communication. Both client and server are implemented in our tool. The client-side implementation aims at generating requests to communicate with a real IED. The server-side includes the report service, as well as the response for the initiate and identify services.

Similarly to the GOOSE and SV microservices, the MMS microservice also communicates to the GUI through the implemented REST API. However, due to its higher complexity, MMS protocol has more endpoints implemented. The Flask library [28] was used to develop the API for the MMS microservice. The client-side endpoints functionalities, methods, input and output parameters are shown in Table 6.

To initialize an MMS connection, the *IP address* is required as input. A connection *identifier* and *connection status* are provided as output. To require equipment identification through MMS, the *identifier* is required. The output data are the IED's manufacture, model, and revision. The description of the data model configured on the MMS equipment requires the connection *identifier*. As output, the whole data structure of the IED is given (similar to Figure 1). To perform reading operations, the connection *identifier* and *path* of the attribute to be read must be provided. The output for this endpoint is the value of the read attribute. To perform writing operations, the connection *identifier*, *path* of the attribute to be written (or structure), and the *data* to be written must be provided. The output for this endpoint is the sending status. Finally, to terminate an MMS connection, the connection *identifier* must be provided. The connection *status* is given as output.

The server-side endpoints are shown in Table 7. To initialize the server, the *IP address* and the *port number* are required as input. The *server status* is returned as output. The report sending requires the *port identification* and the *data* to be sent (*i.e.,* the structure of reported values and types) as input. Buffer (*i.e.,* waiting time before sending a report) and interval (*i.e.,* interval between reports) are optional parameters. The output data is the report status. All aforementioned methods are post-based. For Get methods (*list report*, *verify server status*, and *stop the server*), no inputs are required. The output of *List Report* is a list containing all client identifiers. In order to list the reports, this endpoint receives a request using the Get method. The output data is a list of identifiers. The *verify server status* and *stop the server* endpoints output the server status.

3) TITAN's GRAPHICAL USER INTERFACE

The GUI modelling is illustrated in Figure 6. It is a Model-View-Controller architecture implemented with JavaFX. Thus, it can be supported by different platforms. The GUI communicates with GOOSE, SV, and MMS microservices via REST API. All graphical components are implemented at the View layer. It is composed by the Main Pane, Common, GOOSE Pane, MMS Pane, SV Pane, Dataset Pane, LogicalDevice Pane, and IED Pane.

The Main Pane component is the base for the other components of the View layer. It implements the menu functionalities and the loading window. Two important functionalities also implemented on Main Pane are the Save Project and Open Project buttons. These features enable the user to persist and recover protocol flows in TITAN. The Common component consists of functions of the graphical interface that have the same implementation for the different components of the View layer (*e.g.,* the interface language and dialog implementation).

GOOSE, MMS, and SV panes implement the graphical components and data structures for flow manipulating and showing. For each protocol, the generated traffic flows

TABLE 5. Parameters received from the REST API to generate GOOSE and SV traffic.

Attribute	Description	GOOSE protocol	SV protocol
MACsrc	Source MAC address	Yes	Yes
MACdst	Destination MAC address	Yes	Yes
APPID	Application Identifier	Yes	Yes
ASDU	List of ASDU's in the frame	Not applicable	Yes
gocbRef	Reference object path	Yes	Not applicable
timeAllowedtoLive	Maximum waiting time	Yes	Not applicable
datSet	Dataset name	Yes	Yes
goID/svID	Application identifier	Yes	Optional
stNum	Event number	Yes	Not applicable
test	Test message	Yes	Not applicable
confRev	Configuration version number	Yes	Yes
ndsCom	Commissioning required	Yes	Not applicable
RefrTmEnable	Enable the sending time of reference	Not applicable	Optional
smpSynch	Defines sync signal: non-existent, local or global	Not applicable	Yes
smpRate	Sampling rate	Not applicable	Yes
smpMod	Sets the mode of operation of the sample rate	Not applicable	Yes
allData	Content of selected dataset	Yes	Not applicable

TABLE 6. MMS client endpoints.

Functionality	Method	Input	Output
Initialize connection	Post	IP adress	Connection status, identifier
Equipment identification	Post	Connection identifier	Manufacture, model, revision
Self-description	Post	Connection identifier	The IED's data structure
Read data	Post	Connection identifier, path	Value of the read data
Write data	Post	Connection identifier, path, data	Sending status
Finalize the connection	Post	Connection identifier	Connection status

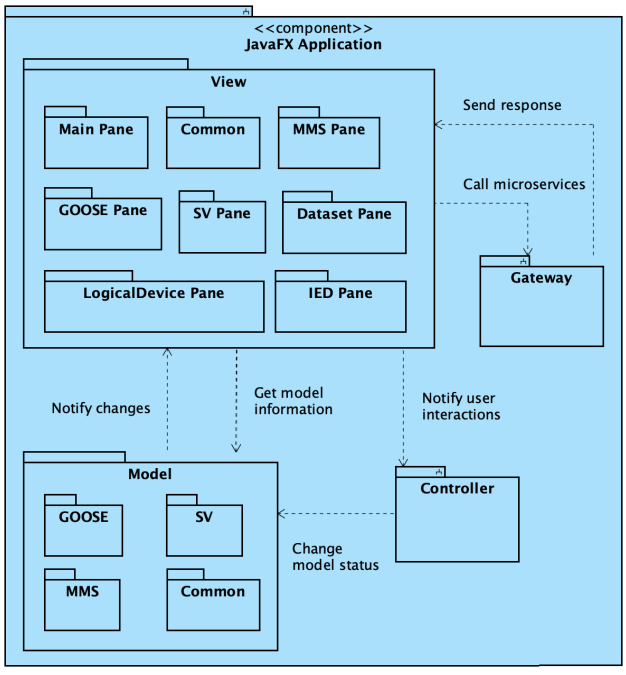

FIGURE 6. Implementation of the JavaFx Application.

are displayed in tables generated by their respective pane. Figure 7 shows the screen of MMS flows as an example.[7] The MMS Pane has more functions implemented than the GOOSE and SV Panes, including the connection establishment, and the Identify, Read, Self-Description, and Write services. The Dataset Pane enables the user to define and associate the DataSets to the IEDs.

[7]Please, refer to http://bit.ly/TITAN_SneakPeek for more screen captures.

TABLE 7. MMS server endpoints.

Functionality	Method	Input	Output
Initialize the server	Post	IP address, port	Server status
Send a report	Post	Report ID, data	Report status
List report	Get	-	List of identifiers
Verify server status	Get	-	Server status
Stop the server	Get	-	Server status

The View layer notifies the user interactions to the Controller component. From notifications of user changes in the View layer, the Controller updates the Model. Similarly, when the Model is changed, the View layer is notified to keep its consistency. Another important component in Figure 6 is the Gateway. This component contains the list of microservices information, including IP address and port. Therefore, it works as an interface between the microservices and the client application.

V. PRACTICAL EXPERIMENTS

In order to evaluate TITAN, first we assess the performance of both SV and GOOSE microservices, and then we assess the

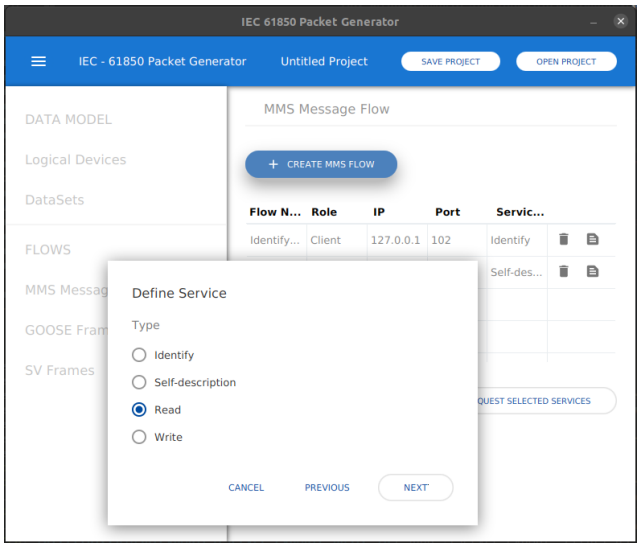

FIGURE 7. MMS flows interface.

usage of all microservices in two different DTT teleprotection scenarios. The goal of the following tests is to show the feasibility of building a testbed with TITAN.

A. PERFORMANCE

As a first experiment, we evaluated the performance of both GOOSE and SV microservices, as they are the most time sensitive protocols as seen in Table 1. In this experiment, we used a computer with both the GOOSE and SV microservices and with the following configuration: Linux Ubuntu operating system version 18.04.1, i7-7700 CPU @ 3.60GHz (4 Cores), 16 GiB of RAM and NIC NetXtreme BCM5719 Gigabit Ethernet (1 Gbit/s).

1) GOOSE MICROSERVICE

As explained in Section II-B1 and shown in Figure 2, once a new event happens, GOOSE frames are sent respecting a progression formula, starting at T1 and increasing after every new frame until it reaches T0, which is the highest time interval between frames. Thus, we ran two different scenarios: firstly we evaluated the minimum frame interval (T1); then we evaluated the interval progression once a new event occurs.

In the first experiment, we assessed the minimum interval between GOOSE frames (T1), sending each frame as fast as possible. The GOOSE microservice was left running for four different durations: 250 milliseconds; 500 milliseconds; 1 second; and 2 seconds. For all durations, as seen in Figure 8, the GOOSE frame median interval was always 4 microseconds, and the highest interval was below 30 microseconds.

In the second experiment, we assessed the interval progression by changing the value of a boolean variable, triggering a new status number (a new event). Figure 9 shows the sequence number progression through time, which is restarted once a new event occurs.

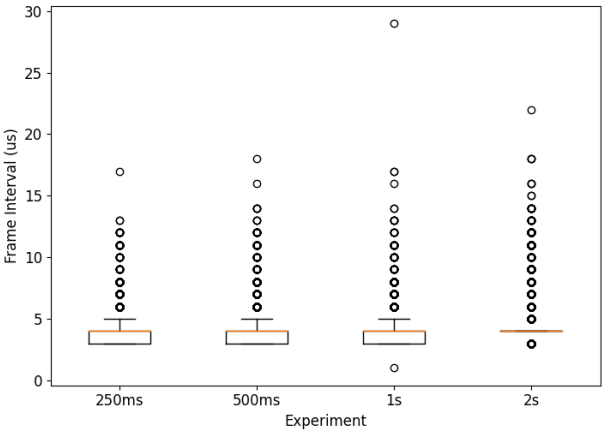

FIGURE 8. GOOSE microservice minimum frame interval.

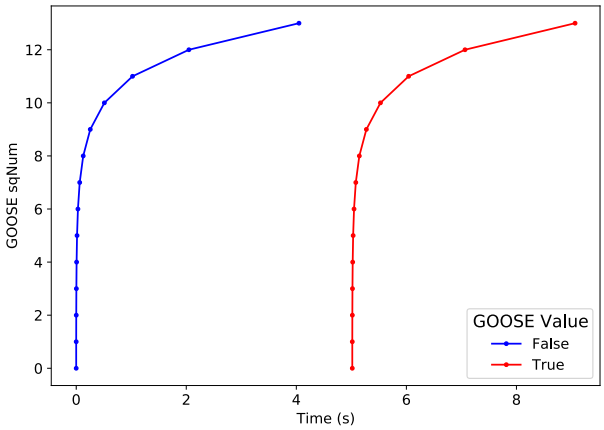

FIGURE 9. GOOSE microservice frame interval progression.

The boolean value starts as *False*. From second 0 to second 1, the frame interval starts small and is duplicated after each frame. The frame interval from second 1 to 2 is one second, and from second 2 to 4 it reaches T0 which is 2 seconds.

The next frame would be sent on second 6, but an event was triggered (the boolean value was changed to *True*) on second 5, restarting the interval progression as seen in Figure 9.

2) SV MICROSERVICE

The SV microservice works in both 50 Hz (European utility frequency) and 60 Hz (American utility frequency). This means that the traffic generator has to be able to send 4,000 frames per second under 50 Hz frequency, or 4,800 frames per second under 60 Hz frequency. That is, 250 microseconds between frames (50 Hz), or 208.333 microseconds between frames (60 Hz).

The SV microservice has to be able to maintain the defined frequency to work properly, thus we stressed the computer by instantiating several SV microservices in parallel working under 60 Hz (Figure 10) and assessed the frequency of frames being sent.

As shown in Figure 10, the frame interval starts to produce outliers after 7 instances, *e.g.*, with 7 instances we see the frame interval increase up to 1 millisecond second (around

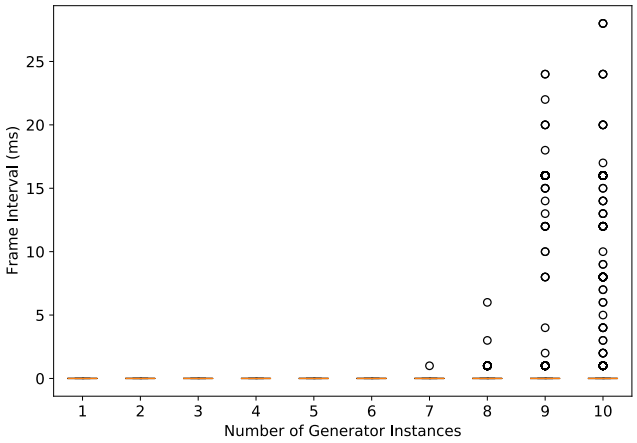

FIGURE 10. 10 instances of the SV microservice running in parallel (60 Hz).

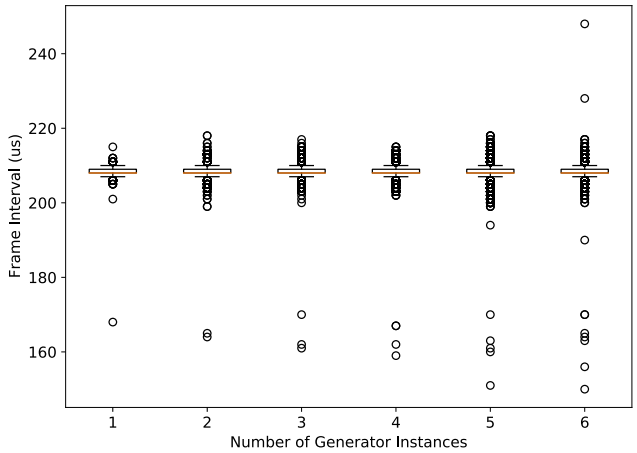

FIGURE 11. 6 instances of the SV microservice running in parallel (60 Hz).

5 times slower than expected) and with 10 instances the frame interval peaks at 27 milliseconds (around 130 times slower than expected). In other words, with 7 SV microservices instances, the IED would be expecting to receive at least 4 frames under 1 millisecond, but it would receive only 1 in that time window.

Figure 11 shows the same experiment, but with 6 SV microservices instances only. The frame interval does not exceed 220 microseconds up to 5 instances, meaning the IED will receive the frames in the expected time window. With 6 instances, the microservice might work for most of the time, but sometimes it might take too long which would trigger the IED's alarm warning the SV stream was momentarily lost. The outliers below 208 microseconds represent the microservice adapting to the CPU's concurrency, indicating that if it took too much time to send previous frames, the next one shall be sent faster.

3) DTT WITH THE GOOSE MICROSERVICE

This experiment's scenario is based on the testbed shown in Figure 12, created using real IEC 61850 equipment and TITAN.

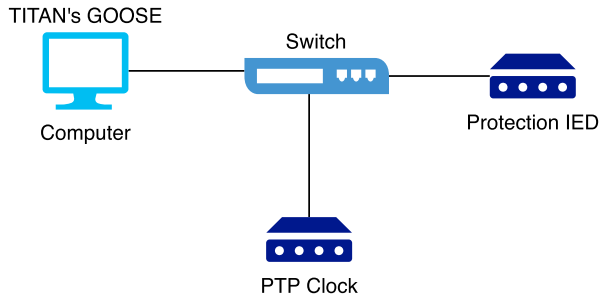

FIGURE 12. DTT with the GOOSE microservice network topology.

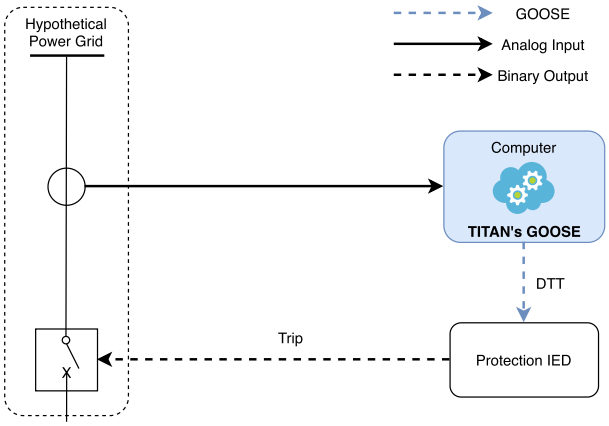

FIGURE 13. DTT scheme using TITAN's GOOSE microservice.

The computer replaces the need for a real protection IED. This computer uses the GOOSE microservice to operate as a sensor and has the following configuration: Linux Ubuntu operating system version 18.04.1, i7-7700 CPU @ 3.60GHz (4 Cores), 16 GiB of RAM and NIC NetXtreme BCM5719 Gigabit Ethernet (1 Gbit/s). The protection IEDs is a SEL 421-7 revision R406 with GOOSE publication and subscription capabilities. The PTP Clock is a SEL 2488 revision R102-V0 used to locally synchronize all the devices (IED and computer) using the Precision Time Protocol (PTP). Finally, the switch is a traditional gigabit Ethernet switch.

Figure 13 shows the traffic flow of this experiment. Basically, the GOOSE microservice is used to emulate a DTT with a real IED. The GOOSE microservice sends a boolean value representing the trip signal, true for trip, false for the opposite. The IED was configured to subscribe to TITAN's GOOSE frames and assign the received value (trip or no trip) to *VB001*. If the *VB001* is set to true, the IED trips.

We use the AcSELerator QuickSet version 6.9.0.2, a proprietary software to directly acquire information from SEL's IEDs. Before starting the experiment, the IED reports the error code *TTL EXPIRED*, which means that it is not receiving the GOOSE frames it is expecting, as seen in Figure 14a. After starting the GOOSE traffic, the IED reports no error code and the status and sequence number increases continually, as seen in Figure 14b.

Goose Report (a)

```
GOOSE Transmit Status

 MultiCastAddr    Ptag:Vlan AppID  StNum      SqNum      TTL     Code
-------------------------------------------------------------------------------
No GOOSE publications configured.


GOOSE Receive Status

 MultiCastAddr    Ptag:Vlan AppID  StNum      SqNum      TTL     Code
-------------------------------------------------------------------------------
GE_BreakerCFG/LLN0$GO$DTT
 01-0C-CD-01-00-03 4:3     3      0          0          0       TTL EXPIRED
 Data Set: GE_BreakerCFG/LLN0$Trip


IEC 61850 Mode/Behavior: On
```

Goose Report (b)

```
GOOSE Transmit Status

 MultiCastAddr    Ptag:Vlan AppID  StNum      SqNum      TTL     Code
-------------------------------------------------------------------------------
No GOOSE publications configured.


GOOSE Receive Status

 MultiCastAddr    Ptag:Vlan AppID  StNum      SqNum      TTL     Code
-------------------------------------------------------------------------------
GE_BreakerCFG/LLN0$GO$DTT
 01-0C-CD-01-00-03 4:3     3      1          20         2000
 Data Set: GE_BreakerCFG/LLN0$Trip


IEC 61850 Mode/Behavior: On
```

FIGURE 14. IED's GOOSE Report. (a) The traffic is off. (b) The traffic is on.

FIGURE 15. IED's front panel.

Sequential Events Recorder

```
Relay 1                                  Date: 09/30/2020  Time: 19:41:17.680
Station A                                Serial Number: 1200870465

FID=SEL-421-7-R406-V1-Z104002-D20191210

#      DATE        TIME          ELEMENT          STATE
1    09/30/2020  19:41:05.3498   VB001            DEASSERTED
2    09/30/2020  19:41:05.3498   TRIP             DEASSERTED
3    09/30/2020  19:41:00.3322   VB001            ASSERTED
4    09/30/2020  19:41:00.3322   TRIP             ASSERTED
```

FIGURE 16. IED's sequential events recorder.

The GOOSE traffic is started sending frames with trip set to false. After 5 seconds, a fault is emulated by setting the trip value to true. Finally, after 5 seconds it is changed back to false. During the 5 seconds of fault, the IED's trip led lights up as seen in Figure 15.

It is also possible to validate the expected behavior by checking the sequence of events of the IED. Once the microservice starts sending GOOSE frames with the boolean value set to true (trip), the IED's *VB001* is asserted, consequently the IED trips as seen in Figure 16. After 5 seconds, the GOOSE microservice changes the value back to false, which can also be seen in Figure 16.

Thus we conclude that the testbed built using TITAN's GOOSE microservice is working as intended, and could be used for teaching, testing or even researching.

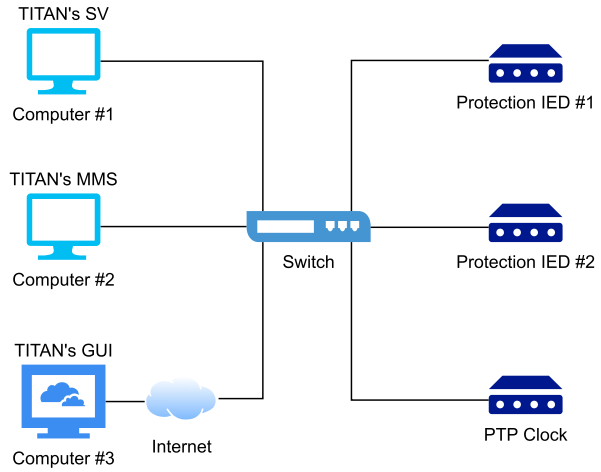

FIGURE 17. DTT with the SV and MMS microservices network topology.

4) DTT WITH THE SV AND MMS MICROSERVICES

This experiment's scenario is based on the testbed shown in Figure 17, created using real IEC 61850 equipment and TITAN.

Computers #1 and #2 have the same configuration: Linux Ubuntu operating system version 18.04.1, i7-7700 CPU @ 3.60GHz (4 Cores), 16 GiB of RAM and NIC NetXtreme BCM5719 Gigabit Ethernet (1 Gbit/s). Computer #3 is connected to the testbed through the Internet and has the following configuration: Linux Ubuntu operating system version 20.04.1, i5-9300H CPU @ 2.40GHz (4 Cores) and 8 GiB of RAM.

Computer #1 has the SV microservice installed to operate as a sensor, while Computer #2 has the MMS microservice installed to operate as a SCADA system. Computer #3 has the TITAN's GUI installed to manage the microservices as needed, note that Computer #3 is connected to the network through the Internet, enabling remote control over Computers #1 and #2 through TITAN's GUI. Computer #1 replaces the need for a real IED, while Computer #2 replaces the need for a real SCADA system.

IEDs #1 and #2 are a SEL 421-7 revision R406, the only difference between them is that IED #1 has the SV subscription capability. Both IEDs have GOOSE publication

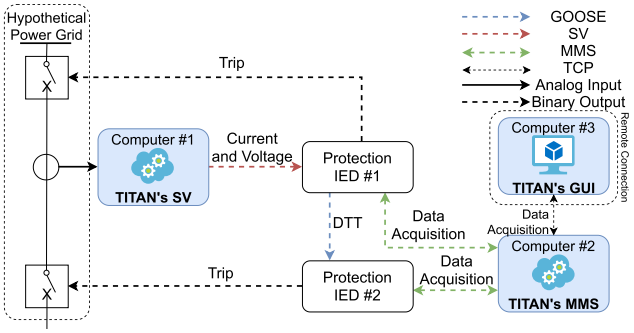

FIGURE 18. DTT scheme using the SV and MMS microservices.

and subscription capabilities, and are also MMS servers. The PTP Clock is a SEL 2488 revision R102-V0 used to locally synchronize all the devices (IEDs and computers) using PTP. Finally, the switch is a traditional gigabit Ethernet switch.

Computer #1 replaces a real non-conventional instrument transformer[8] and Computer #2 replaces a real local SCADA system. Both IEDs are set to work at 50 Hz. The SV microservice emulates the current sensing and sends the sampled values as SV frames to IED #1. This IED is configured with a protection mechanism (instantaneous overcurrent) to trip if the received current is over a defined threshold. This same IED sends GOOSE frames (DTT) to IED #2. The IED #2 was configured to subscribe to the IED #1's GOOSE frames and assign the received value (trip or no trip) to *VB001*. If the *VB001* is set to true, the IED #2 trips as well.

In this scenario, the MMS microservice is used as a SCADA to acquire data from both IEDs. The IEC 61850 traffic takes place in a local network. However, the GUI connects to the MMS microservice using its API over the Internet. Figure 18 shows the traffic flow of this experiment.

We use the AcSELerator QuickSet version 6.9.0.2, a proprietary software to directly acquire information from SEL's IEDs. Before starting the experiment, IED #1 reports the error code *SV STREAM LOST*, which means that it is not receiving the SV frames it is expecting, as seen in Figure 19a. After starting the SV traffic, the IED reports no error code and the *smpSynch* changes to 1, meaning the SV stream is locally synchronized, as seen in Figure 19b.

The SV microservice is configured to send three-phase current samples with 1,800 amplitude. Each phase is set at 120 degree angle of each other. It is possible to validate the expected behavior by checking the phase components and fundamental metering of IED #1. Figure 20a shows that the IED phase C is sensing around 1,272 amperes (Root mean square (RMS) current) and is set at 120 degree angle as expected. Figure 20b shows virtually the same readings, thus it shows the correct behavior of the SV microservice.

[8]Similar to an instrument transformer, but with an internal analog-to-digital conversor and embedded network card IEC 61850 ready.

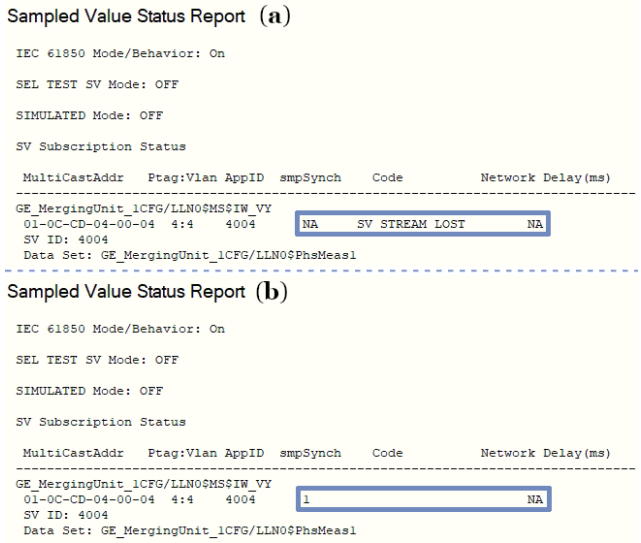

FIGURE 19. IED #1's SV Report. (a) The traffic is off. (b) The traffic is on.

In order to assess the correct behavior of the MMS microservice, the SV microservice is kept running and the MMS microservice is used to read the values of the *METMMXU1* logical node, which gathers information about the sensed current and voltage. Figure 21 shows IED #1's response with the data for the requested logical node. Phase C stores around 1,272 amperes (RMS current) and 120 degree angle as expected, virtually the same readings as in Figure 20. Thus, it shows the correct behavior of the MMS microservice.

The small difference in the data shown in Figures 20 and 21 is due to the data not being captured at the same time as there was no way to automate the data collection process through the IED's proprietary software.

The SV microservice was configured to send 2 different sampling profiles: (1) *pre-fault*, which represents the expected behavior of the hypothetical grid; and (2) *fault*, which represents the overcurrent scenario of the hypothetical grid. During the *pre-fault*, the microservice sends samples representing 1,800 amperes, and 38,000 amperes during the *fault*.

Figure 22a shows the moment IED #1's overcurrent protection was activated. After 1 second the traffic was turned off, thus the protection was no longer detecting a fault. Figure 22b shows that the DTT was carried to IED #2 through the configured GOOSE.

Note that the information shown in Figure 22 was acquired using the IED's proprietary software. This data might also be acquired using the MMS standard as shown in Figure 23. By using the MMS microservice, it is possible to check the value of the received GOOSE, by reading the *VBGGIO1* logical node; and check whether the IED tripped or not, reading the *TRIPPTRC1* logical node.

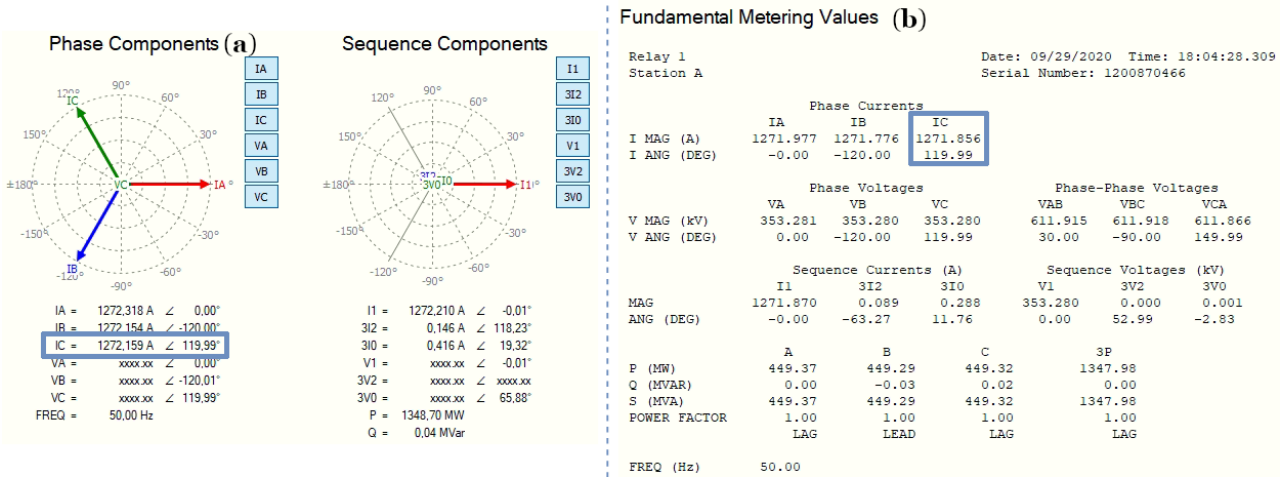

FIGURE 20. IED #1's Current Report. (a) Phase Components. (b) Fundamental Metering.

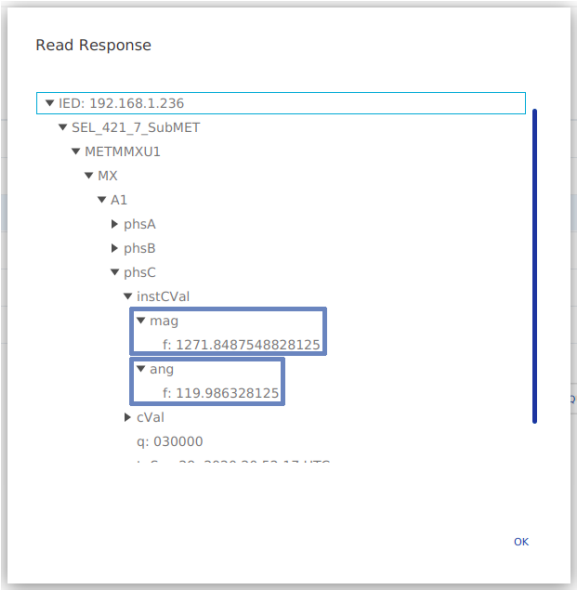

FIGURE 21. IED #1's MMS readings of the logical node *METMMXU1* using TITAN's GUI.

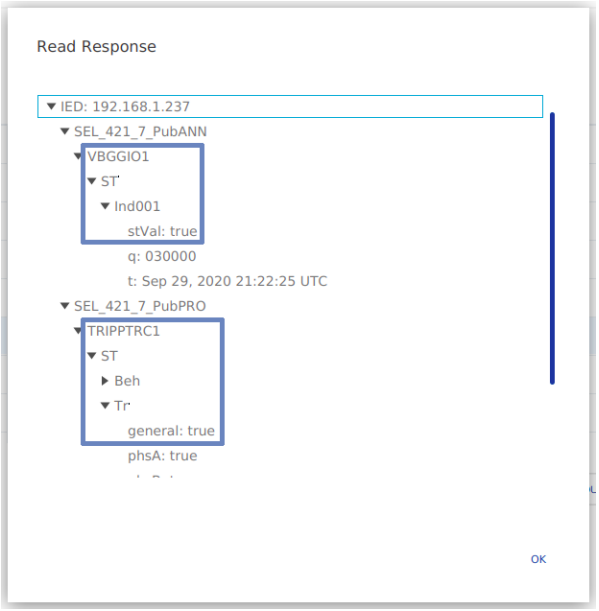

FIGURE 23. IED #2's MMS readings of the logical nodes *VBGGIO1* and *TRIPPTRC1* using TITAN's GUI.

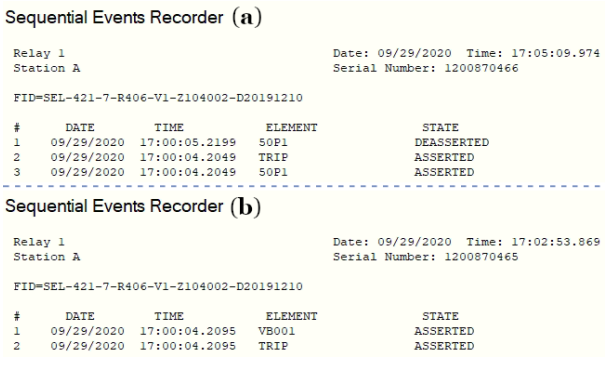

FIGURE 22. Sequential Events Recorder. (a) Trip recorded in IED #1. (b) Trip recorded in IED #2.

Finally, Figures 24 and 25 show, respectively, the oscillography registered in IED #1, containing the current in amperes and whether the overcurrent protection (50P1) and trip was activated; and the SV and GOOSE frames captured in the network, containing the current in amperes (SV frames) and the DTT value (GOOSE frames).

The oscillography was downloaded using the AcSELerator QuickSet and plotted using SEL SynchroWAVE Event version 1.6.0.523. The SV and GOOSE frames were captured during the experiment using tcpdump version 4.9.3 and plotted using Python version 3.8.5. The current wave registered in IED #1, as shown in Figure 24, is equivalent to the generated wave shown in Figure 25, which validates the expected behavior of the microservice.

Thus we conclude that the testbed built using TITAN's SV and MMS microservices is working as intended, and could be used for teaching, testing or even researching.

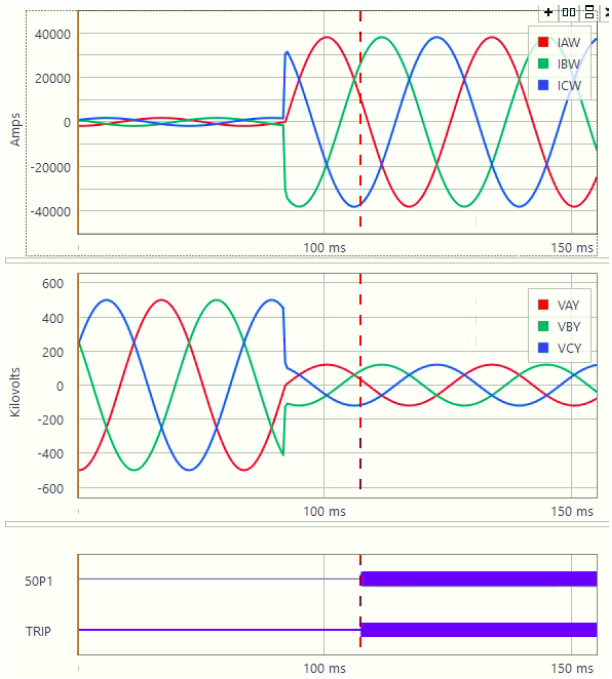

FIGURE 24. IED #1's oscillography.

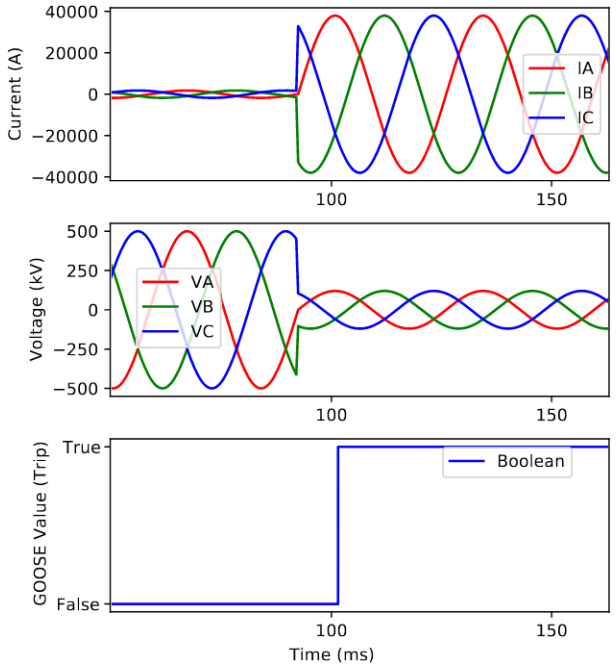

FIGURE 25. SV and GOOSE frames captured in the network.

VI. CONCLUSION

A key challenge for academic researching purposes is the elevated cost of real electronic devices, such as current and potential transformers, or even Intelligent Electronic Devices (IEDs). In this article, we presented Traffic Generator for IEC-61850 Telecommunication Automation Networks (TITAN), a novel tool to support the evaluation of automation systems' communication network. TITAN enables the emulation of IEC 61850 communication, ranging from data sensing (GOOSE and SV) to data acquisition (MMS), thus supporting extensive research and development on this particular smart grid domain. This tool can interact with virtual or real elements, such as IEDs.

Most existing tools are designed with commercial purposes and support only the Windows operating system. In contrast, TITAN is multi-platform and supports traffic generation and transmission of MMS, GOOSE, and SV traffic. Another key difference is that our proposed tool is microservice-oriented, thus supporting multiple instances running in both the same machine or multiple ones. This architecture facilitates the scalability for traffic generation.

This article introduced TITAN's microservice architecture, applications, and presented a real case teleprotection communication scenario with two different ways of building a testbed for this scenario replacing real equipment with TITAN. As stated previously, our main contributions were: (i) a novel IEC 61850-compliant power system automation and communication emulation scheme; (ii) TITAN, a tool with distributed microservices architecture to execute communication traffic generation tasks; (iii) a user-friendly interface to integrate and manage all components; and (iv) a proof of concept testbed using TITAN and real teleprotection devices. Real case studies using TITAN revealed its feasibility in supporting low-cost testbeds for research, teaching, and testing purposes in sensing and acquisition for automation systems.

Our experiment results showed that the GOOSE microservice was capable of sending frames under a 30 microseconds interval, whilst respecting the progression of the time interval between frames. The SV microservice was capable of sending 5 different samples of current and voltage in parallel, for systems working under both 50 Hz and 60 Hz. Finally, TITAN was successfully used in a testbed emulating a Direct Transfer Trip (DTT) scheme, interacting with real IEC 61850 devices.

As future work we intend to keep on improving TITAN features and adding new functionalities, such as adding support for routable GOOSE and SV, providing temporal synchronization, developing microservices for SV and GOOSE subscribing, and a microservice for an MMS server. This would allow building a fully emulated testbed, with no need for real IEDs, that could be used for teaching and researching. These new microservices may allow new possibilities for mixed-testbeds as well – only partially emulated. We also intend to add support for importing SCL files, which would automatically create the necessary traffic for a testbed, making it easier to build new testbeds or even changing old ones.

Finally, we intend to explore and research novel solutions for different smart grid domains using TITAN, such as: Advanced Metering Infrastructure (AMI); and smart charging for electric vehicles. It is quite complex to build a reliable testbed for both domains, while the first domain requires several smart devices as hosts, the second one needs mobile hosts. TITAN seems like a direct solution for AMI testbeds, while the mobility issue for smart charging can be researched

by embedding some TITAN microservices into a raspberry and using wireless communication.

ACKNOWLEDGMENT

The authors would like to thank Matheus Felipe Ayello Leite, Ms. Mayara Helena Moreira Nogueira dos Santos, Yago de Rezende do Santos, Franklin Jordan Ventura Quico, Prof. Dr. Luiz Claudio Schara Magalhaes, and Prof. Dr. Marcio Zamboti Fortes for all of their support and guidance provided during the research and development of this work.

REFERENCES

[1] A. A. Khan, M. H. Rehmani, and M. Reisslein, "Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 860–898, 1st Quart., 2016.

[2] *Communication Networks and Systems in Substations, 2002-2013*, Standard IEC 61850, International Electrotechnical Commission, 2002.

[3] L. Uchoa, S. Quincozes, J. L. Vieira, D. Passos, C. Albuquerque, and D. Mosse, "Analysis of smart grid fault recovery protocols," in *Proc. NOMS-IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–8.

[4] SmartGridware. *SmartGriware IED Simulator*. Accessed: Mar. 2019. [Online]. Available: http://www.smartgridware.com/index.html

[5] Triangle MicroWorks. *61850 Test Suite Pro*. Accessed: 2019. [Online]. Available: http://61850solutions.com/61850-test-suite-pro/

[6] Doble. *61850 TesT Software*. Accessed: 2019. [Online]. Available: https://www.doble.com/product/software-61850-test/

[7] ABB. *ITT600—Integrated Testing Tool*. Dec. 2020. [Online]. Available: https://new.abb.com/products/ITT600-20-M10/integrated-testing-tool

[8] *Svscout—Software Tool for Visualizing IEC 61850 Sampled Values*. Accessed: Jan. 2019. [Online]. Available: https://www.omicronenergy.com/en/products/svscout/description/

[9] *IEDScout*. Accessed: Jan. 2019. [Online]. Available: https://www.omicronenergy.com/en/products/iedscout/

[10] Conprove. *SimulGOOSE*. Accessed: Jan. 2019. [Online]. Available: http://www.conprove.com.br/pub/produtos/goose_simulator.html

[11] D. P. D. Mattos, L. C. S. Magalhaes, D. C. Muchaluat-Saade, S. Arthur A. Z., L. F. Soares, A. Delfino, L. Uchoa, N. C. Fetnandes, Y. Lopes, I. Moares, and C. Albuquerque, "IEC 61850 packet generator for testing substation communication," in *Proc. IEEE PES Asia–Pacific Power Energy Eng. Conf. (APPEEC)*, Dec. 2019, pp. 1–5.

[12] A. D. S. Delfino, N. C. Fernandes, R. Zanghi, and D. C. Muchaluat-Saade, "Development of electrical and network test tool for IEC 61850 based power system protection," in *Proc. IEEE PES Asia–Pacific Power Energy Eng. Conf. (APPEEC)*, Dec. 2019, pp. 1–5.

[13] *Communication Network and Systems for Power Utility Automation—Part 7-1: Basic Communication Structure—Principles and Models*, Standard IEC 61850-7-1, International Electrotechnical Commission, 2011.

[14] *Communication Networks and Systems in Substations—Part 5: Communication Requirements for Functions and Device Models*, Standard IEC 61850-5, International Electrotechnical Commission, 2003.

[15] *Specific Communication Service Mapping—Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3*, Standard IEC 61850-8-1, International Electrotechnical Commission, 2011.

[16] *Communication Networks and Systems for Power Utility Automation—Part 8-1: Specific Communication Service Mapping (SCSM)—Mappings to MMS (ISO 9506-1 and ISO 9506-2)*, Standard IEC 61850-8-1, International Electrotechnical Commission, 2011.

[17] *Industrial Automation Systems—Manufacturing Message Specification (MMS)—Part 2: Protocol Specification*, Standard ISO 9506-2, International Organization for Standardization, 2003.

[18] *Industrial Automation Systems—Manufacturing Message Specification (MMS):Part 1: Service Definition*, Standard ISO 9506-1, International Organization for Standardization, 2003.

[19] J. C. Das, *Power System Protective Relaying*, 1st ed. Boca Raton, FL, USA: CRC Press, 2017.

[20] *IEEE Guide for Protective Relay Applications to Transmission Lines*, IEEE Standard C37.113-2015 (Revision of IEEE Std C37.113-1999), Institute of Electrical and Electronics Engineers, Jun. 2016, pp. 1–141.

[21] H. J. A. Ferr and E. O. Schweitzer, *Modern Solutions for Protection, Control, and Monitoring of Electric Power Systems*, 1st ed. Pullman, WA, USA: Schweitzer Engineering Laboratories, 2010.

[22] R. N. Meira, R. L. A. Pereira, J. P. Nascimento, N. S. D. Brito, H. Silva, and B. A. Souza, "Analysis of interoperability of relays via teleprotection," in *Proc. Simposio Brasileiro de Sistemas Eletricos (SBSE)*, May 2018, pp. 1–6.

[23] C. Naradon, C.-I. Chai, M. Leelajindakrairerk, and C.-I. Chow, "A case study on the interoperability of the direct transfer trip (DTT) technique with carrier signal protection schemes (PTT and DEF) and SCADA system between two utilities in Thailand," in *Proc. IEEE Int. Conf. Environ. Electr. Eng. IEEE Ind. Commercial Power Syst. Eur. (EEEIC / I CPS Europe)*, Jun. 2017, pp. 1–7.

[24] Y. Lopes, D. C. Muchaluat-Saade, N. C. Fernandes, and M. Z. Fortes, "Geese: A traffic generator for performance and security evaluation of IEC 61850 networks," in *Proc. IEEE 24th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2015, pp. 687–692.

[25] IED. *Explorer IEC61850*. Accessed: Jan. 2019. [Online]. Available: https://sourceforge.net/projects/iedexplorer/

[26] A. A. Z. Soares, J. L. Vieira, S. E. Quincozes, V. C. Ferreira, L. M. Ucha, Y. Lopes, D. Passos, N. C. Fernandes, I. M. Moraes, D. Muchaluat-Saade, and C. Albuquerque, "SDN-based teleprotection and control power systems: A study of available controllers and their suitability," *Int. J. Netw. Manage.*, p. e2112, May 2020.

[27] O. Dubuisson, *ASN. 1 Communication Between Heterogeneous Systems*. San Mateo, CA, USA: Morgan Kaufmann, 2000.

[28] P. Vogel, T. Klooster, V. Andrikopoulos, and M. Lungu, "A low-effort analytics platform for visualizing evolving flask-based Python Web services," in *Proc. IEEE Work. Conf. Softw. Visualizat. (VISSOFT)*, Sep. 2017, pp. 109–113.

ARTHUR ALBUQUERQUE ZOPELLARO SOARES received the B.Sc. degree in information systems from the Instituto Federal Fluminense, in 2016, with a sandwich period in computer science from De Montfort University, in 2014, and the M.Sc. degree in computing from Universidade Federal Fluminense (UFF), in 2019, where he is currently pursuing the Ph.D. degree. His research interests include smart grid, IEC 61850, and software-defined networks.

LEONARDO F. SOARES received the B.Sc. degree in computer science from Universidade Federal Fluminense, Niterói, in 2018, where he is currently pursuing the M.Sc. degree. His research interests include computer networks, edge computing, and network security.

DOUGLAS P. MATTOS graduated the degree from Fluminense Federal University, in 2012. He received the master's degree in computer science from Fluminense Federal University in the area of multimedia systems, creating new technologies to enhance the authoring of hypermedia applications for digital TV and Web, in 2016. He is currently pursuing the Ph.D. degree with the Computer Science Institute, Fluminense Federal University, in the area of immersive multimedia systems. He has acted as a Visiting Researcher at Brunel University London, in 2019. He worked as a Researcher in Erasmus VR4STEM Project with the University Politehnica of Bucharest, in 2017. He is also a Professor with Infnet and CEDERJ. Also, he worked with Visagio as a TI Consultant with Java Technologies.

PAULO H. B. S. PINHEIRO received the bachelor's degree in electrical engineering from the University Centre of Volta Redonda, in 2018. He is currently pursuing the master's degree with Fluminense Federal University. His research interests include modeling and analysis of electrical power systems, power electronics, power system protection, and substation automation systems.

SILVIO E. QUINCOZES received the bachelor's degree in software engineering from the Federal University of Pampa (UNIPAMPA), in 2011, and the master's degree in computer science from the Federal University of Santa Maria (UFSM), in 2015. He is currently pursuing the Ph.D. degree in computing with Fluminense Federal University (UFF). He is interested in researches topics related to information security, computer networks, intrusion detection systems, smart grids, digital substation systems, the Internet of Things, data mining, and software engineering. He has authored the best paper at the IX Latin American Network Operations and Management Symposium (LANOMS), in 2019.

VINICIUS C. FERREIRA received the bachelor's degree in telecommunications engineering and the master's degree in electrical and telecommunications engineering from Universidade Federal Fluminense (UFF), in 2013 and 2016, respectively, where he is currently pursuing the Ph.D. degree. His research interests include computer networks, the Internet of Things, and e-Health.

GUILHERME H. APOSTOLO was born in Rio de Janeiro, Brazil, in 1993. He received the B.S. degree in telecommunication engineering from Universidade Federal Fluminense, Niterói, Brazil, where he is currently pursuing the M.Sc. degree in computing with the Institute of Computing. His current research interests include green networking, wireless communications, machine learning, and biomedical engineering.

GABRIEL R. CARRARA received the B.Sc. and M.Sc. degrees in computer science from Universidade Federal Fluminense (UFF), Brazil, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree. His research interests include blockchain technology, the Internet of Things, and software defined networks.

IGOR M. MORAES received the degree *(cum laude)* in electronic engineering, in 2003, and the M.Sc. and D.Sc. degrees in electrical engineering from the Universidade Federal do Rio de Janeiro (UFRJ), in 2006 and 2009, respectively. In 2018, he was an Invited Professor with the LIP6 Laboratory, Sorbonne Université, Paris, France. He is currently an Associate Professor with the Instituto de Instituto de Computação (IC), Universidade Federal Fluminense (UFF). His research interests include cache networks, opportunistic networks, wireless networks, and network security. He is also a member of the Brazilian Computer Society (SBC).

CÉLIO ALBUQUERQUE received the B.S. and M.S. degrees in electrical and electronics engineering from the Universidade Federal do Rio de Janeiro, Brazil, in 1993 and 1995, respectively, and the M.S. and Ph.D. degrees in information and computer science from the University of California at Irvine, in 1997 and 2000, respectively. From 2000 to 2003, he has served as the Networking Architect for Magis Networks, designing high-speed wireless medium access control. Since 2004, he has been an Associate Professor with the Computer Science Department, Universidade Federal Fluminense, Brazil. His research interests include wireless networks, network security, smart grid communications, the Internet architectures and protocols, and multicast and multimedia services.

YONA LOPES received the degree in telecommunication/electrical engineering, in 2010, and the M.Sc. degree in telecommunication engineering and the D.Sc. degree in computer science from Fluminense Federal University (UFF), Niterói, in 2013 and 2019, respectively. She is currently a Professor with the Electrical Engineering Department, UFF. Her research interests include IEC 61850, digital solutions for electrical energy, substation automation, and software-defined networks. She is also a CIGRE-BR Member on the Telecommunication and Information Technology Committee and has more than ten years of experience with substation automation.

NATALIA C. FERNANDES received the degree in electronics and computer engineering, the M.Sc. degree, and the D.Sc. degree in electrical engineering from the Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, in 2006, 2008, and 2011, respectively. She is currently an Associate Professor with the Telecommunications Engineering Department, Universidade Federal Fluminense (UFF), Niterói. Her research interests include network security, the future Internet, and wireless networks.

DÉBORA C. MUCHALUAT-SAADE received the bachelor's degree in computer engineering and the M.Sc. degree in computer science, in 1992 and 1996, respectively, and the Ph.D. degree in computer science from the Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), in 2003. She is currently a Full Professor with the Computer Science Department, Fluminense Federal University (UFF). In 2003, she founded the MídiaCom Research Lab, UFF. She received several research grants from the Brazilian National Council for Scientific and Technological Development (CNPq), the Rio de Janeiro State's Research Support Foundation (FAPERJ), the Coordination for the Improvement of Higher Education Personnel (CAPES), and other funding agencies. Her research interests include computer networks, wireless networks, smart grids, multimedia systems, and digital healthcare.