IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 10, 2018, accepted December 19, 2018, date of publication January 17, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2893114

Defining a Reliable Network Topology in Software-Defined Power Substations

ALEXANDER LEAL AND JUAN FELIPE BOTERO
Department of Electronics and Telecommunications Engineering, University of Antioquia, Medellín 050010, Colombia
Corresponding author: Alexander Leal (erwin.leal@udea.edu.co)

This work was supported in part by the CODI Project under Grant 2015-7793, in part by CODI Sustainability Strategy of the University of Antioquia, from 2018 to 2019, and in part by the Colciencias Doctoral Fellowship Call 647 in 2014.

ABSTRACT In recent years, communication networks on modern power substations have grown both in size and complexity, demanding the highest levels of reliability. However, there is no unique criterion to define the structure of the topology in such networks, since in every substation the end user implements their own topology or the topology suggested by a vendor, according to IEC 61850 standard guidelines. This paper proposes a methodology, using integer linear programming, to solve the problem of generating a reliable network topology in a software-defined power substations context. The trustworthiness of the reached solution is evaluated using terminal reliability techniques, graph metrics, and end-to-end time delay performance. The obtained results confirm that the proposed network topology is highly reliable to be implemented in power substations, according to the network redundancy considerations proposed by the IEC 62439 standard, and the operation time requirements suggested by the on IEC 61850 standard. In addition, we present software defined networking-based solutions for loop-based topologies in the proposed network topology, which would be technically unfeasible using traditional network protocols. These solutions include algorithms to solve problems related to the broadcast traffic containment and the diffusion and reliability of the multicast traffic.

INDEX TERMS BFS, disjoint paths, GOOSE, graph metrics, graph theory, latency, loop-based topology, management, multicast, network reliability, power substation, SDN, spider web, SV.

I. INTRODUCTION

A power substation communications network is a mission-critical network and needs to be designed with redundancy principles to guarantee fault-tolerance. In this scenario, the network topology is one of the main components to provide reliability. Currently, different network topologies can be implemented for substation networks based on the IEC 61850 standard, for example: star, ring, multiple ring, or combinations of these [1]. However, there is no single network topology that provides better performance for all substation automation applications. Each topology has its strengths and weaknesses depending on the use, but it must always ensure fault-tolerance and low latency. This means, if one connection element fails, communication should still be possible through a backup connection ensuring an appropriate delay. Although there are other important factors associated with the choice of the topology (relative cost, administration issues and application suitability), this work is oriented to the reliability and latency aspects in a Software-defined power substations context, as explained in [2] and [3].

In previous work, we presented a reconceptualization of the power substations communications network architecture, taking the communications network as the central point and the SDN paradigm as a key element of its formulation [3]. There, we proposed a concentric model called S3N, Smart Solution for Substation Network (see Fig. 1).

S3N presents an alternative way to represent all interactions among the elements incorporated by the network model provided in the standard IEC 61850, a hierarchical architecture integrated by three levels identified as station, bay, and process; and interconnected via the process bus and the station bus (see Fig. 2) [4].

In S3N's architecture, the central axis corresponds to all interconnection devices, mainly ethernet switches and wiring. This component is called connectivity or S3N-CONNECT and provides communication, among all the components that are directly related to the operation of the communications network. S3N-CONNECT provides communication in a single physical network, in contrast to traditional approaches where two separated physical networks are used: one called

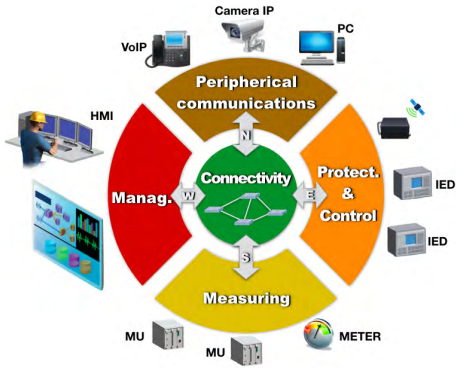


FIGURE 1. S3N Architecture, a reconceptualization of the power substations communications network architecture.

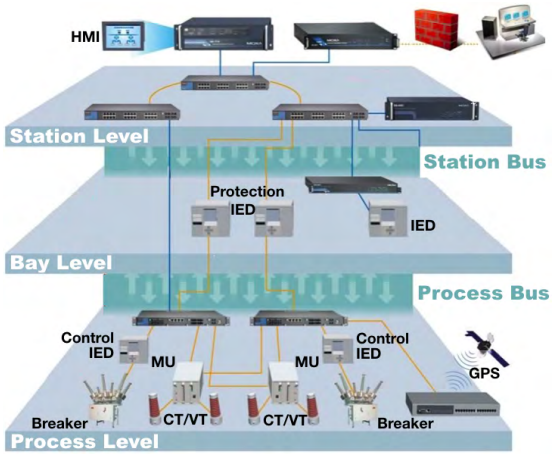


FIGURE 2. IEC 61850 network architecture.

the station bus (governed by a ring of partial mesh topologies) and the other one called the process bus (defined for the resiliency introduced by PRP - Parallel Redundancy Protocol and HSR - High-availability Seamless Redundancy) [5].

Fig. 1 shows the main four groups belonging to the S3N architecture: 1) protection and control devices (Intelligent Electronic Devices - IEDs, actuators), grouped in the component Protection and Control or S3N-PROTECT; 2) management devices (Events Recorder, Monitoring) which are grouped in the component Management or S3N-MANAGE; 3) measurement devices (Merging Units - MUs and meters), grouped in the component Measuring or S3N-MEASURE; and 4) peripherical communication devices (Cameras, VoIP phones) grouped in the component Peripherical communication or S3N-PERCOM.

In traditional power substation networks, PRP and HSR protocols are used to guarantee zero recovery time and fault-tolerance. But this feature is achieved at the expense of duplicating the communications network (PRP instance) and/or adding new network devices (PRP and HSR instances). Therefore, it is paramount to define a convenient topology that simplifies the system, eliminates existing inefficiencies, and ensures network resiliency and zero recovery time in a single network. With a suitable topology, SDN allows programming the transmission of traffic flows in the network avoiding the need of duplicating the number of devices in the network or incorporating other ones. Therefore, the main contributions on this paper are:

- The introduction of a novel methodology to specify and characterize a reliable network topology under an SDN environment, which ensures network resiliency and zero recovery time without duplicating networks or adding new network devices.
- A significant instrument to compare network topologies according to different criteria: terminal reliability, graph metrics, and End-To-End (ETE) time-delay.
- A set of algorithms to illustrate the SDN benefits to solve complex issues related to loop-based topologies such as broadcast traffic control, path redundancy, packet redundancy, and multicast traffic management.

The remainder of this paper is organized as follows. Section II presents the related work while Section III exposes the problem statement. Section IV briefly describes the optimization model used to build an undirected graph based on the aforementioned considerations. Next, in Section V, terminal reliability, graph metrics and ETE time-delay analysis are conducted to validate the proposed topology, and the obtained results are discussed. Section VI exposes several SDN use cases about how to address particular needs in loop-based topologies. Finally, conclusions are presented in Section VII.

II. RELATED WORK

In previous work [1], [5]–[9], different authors have discussed about how the network topology in a power substation environment should look like (see Fig. 3). However, many design factors like redundancy, efficiency, diagnostics, scalability, management, latency, cost, reliability, substation size, among others, may influence the decision to choose one topology over the other.

In [1], authors analyze several reference network topologies for the station bus, the process bus and the connection of station bus with the process bus(es). Here, they argue that there are no 'best' network topology and no 'best' redundancy protocol. They all have strengths and weaknesses and the correct choice for an application depends on many factors. For example, the ability to withstand points of failure, application suitability, protocol dependencies, relative cost, administration issues, redundancy, among others.

Similarly, in [5], authors provide a brief qualitative analysis about substation network topologies like star, double star, ring and multiple ring; exposing their pros and cons. They suggest that according to the different size, complexity and design criterion of the network; combination of ring and star topologies can be applied.

Likewise, Yadav and Kapadia [6] present a comparison between the dual redundant tree and ring topologies taking into account the following features: physical redundancy, latency, bandwidth efficiency, scalability, multicast and broadcast containment, maintenance, fairness,

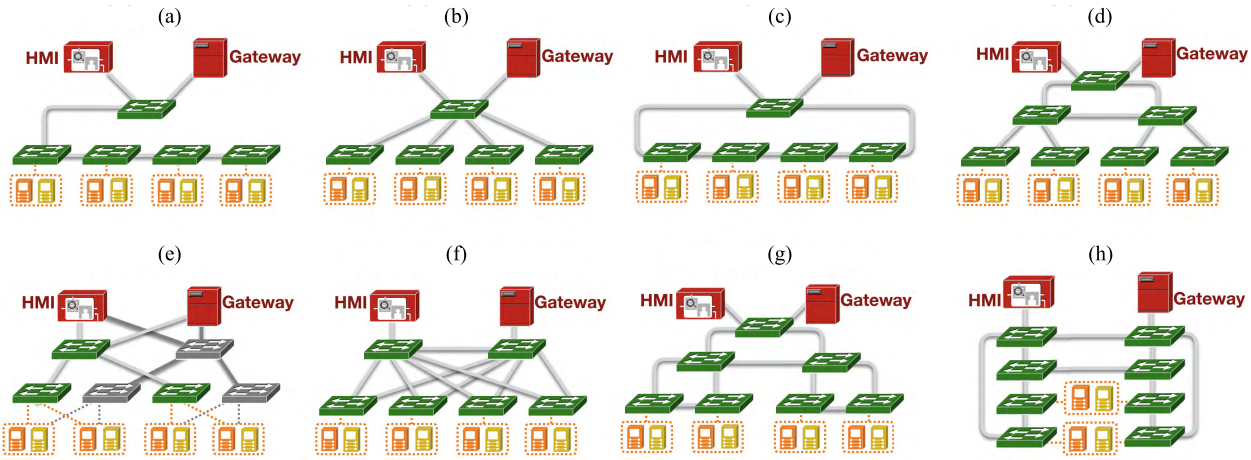


FIGURE 3. Common network topologies in power substations. (a) Cascade. (b) Star or tree. (c) Ring. (d) Star-ring. (e) Dual tree. (f) Dual redundant tree. (g) ring and subrings. (h) Redundant ring.

fast convergence, and cybersecurity. The results suggest that dual redundant tree topology is the best option.

Also, a comparative analysis of topologies was performed in [7] taking the reliability as reference. The reliability and availability calculations employ the Reliability Block Diagram (RBD) method along with the Mean Time To Failure (MTTF) and the Mean Time To Repair (MTTR) reliability parameters. The results achieved show that the redundant-ring provides the highest reliability as compared to other three studied topologies (Cascade, Ring, Star-ring).

On the other hand, in [8], researchers a novel process bus network topology based on cobweb in nature, including single and dual-cobweb architectures. The natural cobweb architecture has a spoke-ring structure, where each spoke intersects with the corresponding ring at a node and all spokes converged in a hub zone. In this study, the Fault Tree Analysis (FTA) is used to assess the reliability of the novel cobweb topology in comparison with traditional topologies (cascade, ring and star), showing better reliability. In addition, the ETE time-delay performance for the topology proposal conforms the IEC 61850 standard requirements ($\leq 3\ ms$).

Finally, Ali *et al.* [9] propose a novel network topology for the process bus where the bay Ethernet switches are connected forming a ring network, while each IED is connected to two different Ethernet switches (with its own bay Ethernet switch and to the adjacent bay Ethernet switch). In this work, the reliability and the end-to-end (ETE) time-delay performance of the network topology proposal are compared with traditional topologies such as cascade, star, ring and star-ring. The achieved results show that the proposed network topology is highly reliable, fast, and has a deterministic nature in comparison with the others.

Unlike the aforementioned proposals, this paper proposes a methodology, using Integer Linear Programming (ILP), to generate a reliable network topology in a Software-defined power substations context, according to the guidelines described in the S3N (Smart Solution for Substation Networks) architecture. An architecture to power substations communications network where the station and process buses are not separated and it is not necessary to duplicate networks or add new devices for guaranteeing zero recovery time with no packet loss, as is the case of the PRP and HSR protocols. In addition, we validate our approach using different strategies: k-terminal reliability techniques, graph metrics and ETE time delay; in comparison with the topologies: cascade, tree, ring, mesh, redundant ring and dual redundant tree.

To conclude this review, Table 1 summarizes the methodologies used to compare topologies on each work, and whether or not there is a novel network topology proposal.

III. PROBLEM STATEMENT

The concept of automated substation, guided by the IEC 61850 standard [4], allowed the transition from traditional wired connections to an Ethernet-based network with IP support, introducing the definition of new communication concepts in order to provide: reliability, interoperability and flexibility. For instance, new network protocols have appeared with the aim to offer redundancy (PRP and HSR), bring protection and control (GOOSE - Generic Object Oriented Substation Event), get measures (SV - Sample Measure Value) or allow management (MMS - Manufacturing Message Specification), among others. Also, regarding network topology, the convention of using two communications buses (station and process) to interconnect all devices within the power substation is nowadays commonly used. However, all these new features increase the network management complexity, as well as the CAPital EXpenditures (CAPEX) and OPerating EXpense (OPEX). Particularly, this paper is focused in proposing a reliable network topology in a Software-defined power substations context, because we consider that technological enablers such as SDN [10] facilitate the administration of complex networks, guaranteeing that

TABLE 1. Summarized overview of studies around network topologies in a power substation environment.

Work	Comparative topologies	Methodology	New topology approach
Communication networks and systems for power utility automation, Part 90-4: Network engineering guidelines [1]	Tree, dual tree, ring and multiple ring (parallel rings, ring of rings, ring and subrings)	Qualitative evaluation of the following criteria: simplicity of topology, traffic control, latency, redundancy, specificity of a device or protocol and limitations.	No
Hirschmann White Paper, Data Communication in Substation Automation System (SAS) [5]	Star, double star, ring and multiple ring	Qualitative analysis of redundancy, data throughput, diagnostics, expandability, cable connectivity, latency and network size.	No
IP and Ethernet communication technologies and topologies for IED networks [6]	Dual redundant tree and ring	Comparative table of latency, bandwidth, scalability, multicast and broadcast containment, maintenance, and fairness in terms of superior or inferior qualification	No
Investigating performance and reliability of process bus networks for digital protective relaying [7]	Cascade, ring, star-ring and redundant ring	Reliability Block Diagram (RBD) using the Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR) reliability parameters	No
A high-reliability and determinacy architecture for smart substation process-level network based on cobweb topology [8]	Cascade, ring, star and suggested topology (cobweb)	Reliability analysis based on Fault-Tree Analysis (FTA), using the Mean Time To Failure (MTTF) as reference parameter	Yes
IEC61850 substation communication network architecture for efficient energy system automation [9]	Cascade, ring, star-ring and proposed topology	Reliability Block Diagram (RBD) using the Mean Time To Failure (MTTF), and End-to-End (ETE) time-delay	Yes

network topology becomes agnostic of the operation scheme of any protocol.

The following subsections explain which conditions should be taken into account to define a reliable topology in a Software-defined power substations context.

A. S3N ARCHITECTURE LEGACY

In the S3N architecture, the connectivity component (S3N-CONNECT) is governed by a Software Defined Networking environment (SDN), integrating in a single network the station and process buses defined in the IEC 61850 standard [3]. For this reason, in the topology design problem, we are going to consider a single network topology. In the SDN environment, it is necessary to establish a communication channel between an element called SDN controller and the set of switches of the network topology, with the purpose of exchanging control messages. There are two ways to implement this control channel: *out-of-band* and *in-band* control. *Out-of-band* control uses separate Ethernet ports and links to connect the switches to the controller and exchanges control traffic in a dedicated way, acting as a separate physical network. On the other hand, *in-band* control uses available network links to send both types of traffic, data and control.

The S3N architecture is based on a concentric model (see Fig. 1). Therefore, at the beginning, to provide basic connectivity among all the edge components of the architecture (HMI, IEDs, MUs, Bay Controllers, Cameras, among others), the switches attached to those components will be interconnected by a star topology to the main switch (located at the center of the star). Hence, the communication between SDN controller and the others switches is carried out through the main switch, using *in-band* SDN management.

In addition, according to the Network engineering guidelines (IEC 61850-90-4), all devices belonging to a single bay such as IEDs, MUs and/or bay controller; should be connected to a single switch [1].

B. REDUNDANCY

It refers to the careful use of redundant connections between network devices or, the use of multiple network devices itself, to provide reliability on the network. This ensures that, if a link or node component fails, several communication connections will not be affected. Particularly, in our approach, the network topology is oriented to guarantee two link-disjoint paths between any source and destination switch of the topology, so that if one link in the main path fails, communication is still possible through a backup path.

This feature combined with the transmission of duplicated packets over each link-disjoint path guarantees zero recovery time with no packet loss, a paramount characteristic within power substations to allow the service continuity in case of network failure. This point is the most important concern in our design, reason why the S3N architecture [2] includes two modules to guarantee the simultaneous transmission of multicast GOOSE messages through two independent paths: *Path redundancy* and *Packet redundancy* (more details about these features are presented on Section VI, Use cases implementation with SDN, Subsection C, GOOSE multicast traffic management). In this way, the S3N architecture will provide similar behavior to the one provided by protocols such as PRP or HSR, without the need to use additional devices (redbox or HSR interface) or duplicated networks. Fig. 4 shows, on the same graph, the concept related to the expressions *two node-disjoint paths* and *two link-disjoint paths*, for a pair of vertices (u, v).

To guarantee the aforementioned operation, a *two-edge-connected* graph is required. First, a connected graph is a graph where there always exists a path between any pair of vertices. Therefore, a *two-edge-connected* graph is a graph where there are two paths to connect any pair of vertices of the topology (see Fig. 4); it means that the graph will always be connected, even if any edge of the graph is removed.

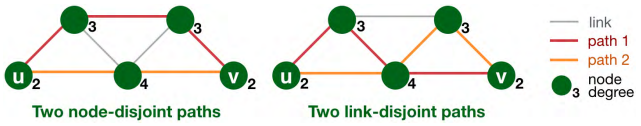


FIGURE 4. Node-disjoint paths and Link-disjoint paths concepts.

To determine what are the requirements in a *two-edge-connected* graph, we use one of the most important facts about connectivity in graph theory; the Menger's theorem [11], which characterizes the edge-connectivity of a graph in terms of the number of independent paths between vertices.

> *Menger's Theorem (edge version): Let G be a graph and u, v ∈ V(G). The maximum number of edge-disjoint u,v paths in G is equal to the minimum number of edges needed to be removed from G to disconnect u from v.*

Then, as a consequence of Menger's Theorem, the maximum edge connectivity of a given graph is the smallest degree of any node, since deleting these edges disconnects the graph [12]. In this way, according to the Menger's Theorem, Fig. 4, corresponds to *two-edge-connected* graph because the smallest degree of the nodes is two. In conclusion, to guarantee a *two-edge-connected* graph we have to ensure that each node in our topology is connected, at least, through two links.

IV. OPTIMIZATION MODEL

To solve the problem of generating a reliable network topology in a Software-defined power substations context, we propose an Integer Linear Programming (ILP) model.

A. FORMULATION

Before introducing the model, Table 2 details the notation. Also, the terms "switch" and "node" are used interchangeably throughout this paper.

TABLE 2. Model elements.

Parameter	n	Refers to the number of switches (nodes), in the S3N-Connect component.
Indexes	i	index associated to the source node of the link.
	j	index associated to the destination node of the link.
Variable	$x_{i,j}$	Binary variable to indicate the presence of a link between the i node and the j node.

B. OBJECTIVE FUNCTION

Suppose the network topology is represented by an undirected graph $G(N, L)$ composed of a set of nodes N and a set of links L. The links are assumed to be bidirectional, with the same characteristics (cost, bandwidth). The goal of the model is to formulate a network topology that guarantees the reliability considerations defined in Section II, given the n parameter, such that the number of required links is minimized.

$$minimize \sum_{i} \sum_{j} x_{i,j} \tag{1}$$

C. CONSTRAINTS

- *No loops:* A node cannot be source and destination of one link.

$$x_{i,j} = 0 \Rightarrow i = j \tag{2}$$

- *SDN Criteria:* All switches (nodes) are connected to the main switch to facilitate their connection to the SDN controller, since the main switch is directly attached to the SDN controller. For the sake of simplicity in the formulation, the main switch corresponds to the first node ($i = 1$).

$$\sum_{j=2}^{n} x_{i,j} = n - 1 \Rightarrow i = 1 \tag{3}$$

- *Link-disjoint paths:* Each node (switch) should be connected, at least, through of two links. In other words, the minimal degree for each node (switch) should be two.

$$\sum_{j=1}^{n} x_{i,j} \geq 2 \quad \forall\, i \tag{4}$$

To summarize, Fig. 5 illustrates the meaning of the aforementioned constraints on the network topology modeling.

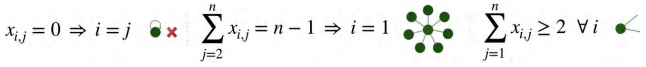


FIGURE 5. Graphical representation of the ILP constraints.

D. SOLUTION

The solver used to solve the ILP problem was the GNU Linear Programming Kit (GLPK) [13]. The obtained results for different n values are shown in Fig. 6. As noted, the restrictions were accomplished and the reader can check by direct observation that the obtained graph is *two-edge-connected*, even though one link fails, the connectivity in the network remains because there is a feasible path between any pair of nodes.

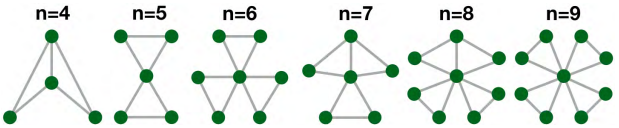


FIGURE 6. Topologies obtained from the proposed ILP.

E. IMPROVED SOLUTION

The resulting topologies show a single point of failure (the central node). It means that the topology addressed would not retain the connectivity (connected graph) in case of a failure in a single node. A graph is connected when there is a path between every pair of vertices and here, if the central node fails, the graph would not be connected. Accordingly, we propose to increment the degree of the nodes to 3 in the constraint defined by the Eq. (4), obtaining the results illustrated in Fig. 7. It is easy to note in Fig. 7 that changing the constraint raised in the Eq. (4), we achieve a 100% redundant solution (connected graph) in case a single node fails. This topology is known as an artificial spider web or cobweb, and could be regarded as the mixture of two topologies: star and ring. A spider web topology gives a highly optimized structure, efficient to rapidly transmit information between nodes that are located further away from the center of the topology and 100% tolerant to link failures [14]. Here, it is important to mention that in [8], this topology was analyzed as an architecture for the substation process bus with promising results in comparison with traditional topologies (cascade, ring and star).

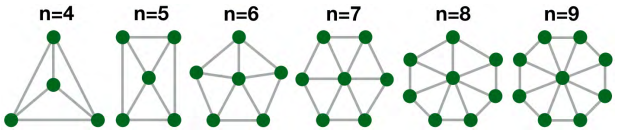


FIGURE 7. Topologies obtained from the proposed ILP with a node degree 3.

V. PERFORMANCE EVALUATION

To validate the reliability of the obtained solution, we choose to compare the performance of different network topologies (included the spider web topology), over a use-case. Particularly, we choose a small substation of 220/132 kV, with single bus, which is classified as T1-1 by IEC 61850-5 [15]. Fig. 8 shows the diagram line for such substation, along with a possible structure for the devices of control, protection and measuring, in each line bay.

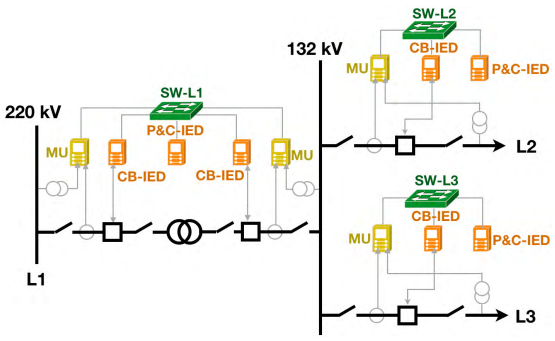


FIGURE 8. Substation T1-1 type.

According to [7], in this type of substation, there are 5 bays, three line bays, one transformer bay and one bus bay, where each bay has a specific scheme of measuring, protection and control. Furthermore, with regard to the S3N architecture we will have to interconnect at least 7 switches: SW_{MNG}, SW_{COM}, SW_{L1}, SW_{L2}, SW_{L3}, SW_{XFRM} and SW_{BUS} (see Fig. 9).

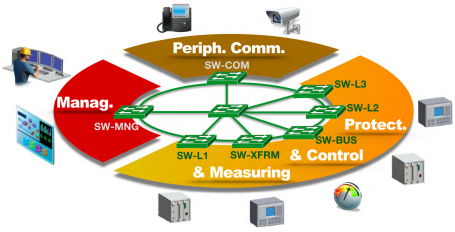


FIGURE 9. Spider web topology under T1-1 use-case in the S3N architecture.

A. METRICS

The evaluation process involves three different types of analysis: k-Terminal Reliability, Graph Metrics and ETE delay analysis; over seven different network topologies, according to the mentioned use-case (see Fig. 10). However, although the suggested methodology will be applied to one particular type of substation, this method can be applied to any power substation.

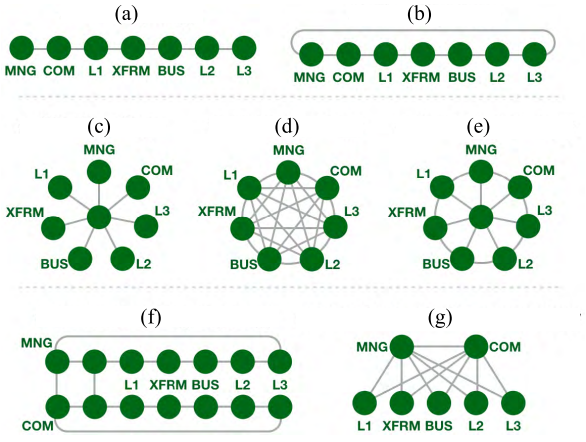


FIGURE 10. Network topologies for the evaluation of the proposed use-case. (a) Bus topology. (b) Ring topology. (c) Star topology. (d) Mesh topology. (e) Cobweb topology. (f) Redundant ring topology. (g) Dual redundant tree topology.

1) k-TERMINAL RELIABILITY

Terminal reliability (connectedness probability), is defined as the probability of maintaining nodes connected considering all possible paths between origin-destination pairs. In other words, this metric reflects the redundancy of a network in which alternative routes could be used when a link fails. That is the reason why we are going to use this metric to characterize the probability of network operation. Specifically, k-terminal reliability gives the probability that k specified nodes in a network are connected [16].

For the computation of the two-terminal reliability metric, we choose the algorithm SYREL [17], which incorporates the best features of both path and cutset methods, is accurate and has a relatively low computational complexity. At SYREL, the network topology is represented by a probabilistic graph $G(N, L)$, where N and L are the set of nodes (switches) and links. Also, each simple path between a given pair of nodes is denoted as P_i (with $i = 1, 2, 3...$). Finally, the algorithm assumes that the elements' failures are statistically independent and have a probability p_l of being available, or q_l of being in failed state, with $p_l = 1 - q_l$ and $l \in L$. The terminal reliability between a pair of nodes is given by Eq. (5), where E_i denotes the event where path P_i is up and m represents the number of paths between those two nodes.

$$R_{terminal} = Pr(\bigcup_{i=1}^{m} E_i) \tag{5}$$

This expression for terminal reliability can be computed by decomposing the set of paths in the graph into another set of mutually exclusive paths between two nodes. For example, if we have three possible paths between two nodes called P_1, P_2 and P_3 (see Fig. 11); the terminal reliability expression consists of three terms which correspond to three mutually exclusive events: the first event occurs when P_1 is up, the second event occurs when P_2 is up and P_1 is down, and the third event occurs when P_3 is up and both P_1 and P_2 are down. In other words, if $\bar{E}_i$ denotes that path i is not operational, $R_{terminal}$ can be decomposed into mutually exclusive events as $Pr(E_1) + Pr(E_2 \wedge \bar{E}_1) + Pr(E_3 \wedge \bar{E}_2 \wedge \bar{E}_1)$.

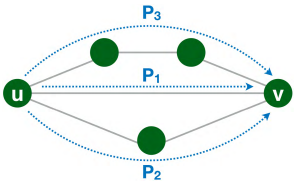


FIGURE 11. Possible paths between two nodes in a K-terminal context.

Now, to apply this mechanism it is necessary to establish what is the probability p_l that link $l \in L$ is available. For this purpose, we are going to use the MTBF concept [18]. MTBF, or Mean Time Between Failures, which is a basic measure of a system reliability, typically represented in units of time, where the higher MTBF the higher reliability of the product is. Now, once the MTBF of an element is calculated, the probability that element will be operational for a time given could be expressed with the Eq. (6), assuming a failure-rate constant.

$$R_{system} = e^{-(time/MTBF)} \tag{6}$$

Thus, the MTBF for passive copper cables is around 50 million hours, typically order of magnitude higher than industry data of fiber cables [19], and the MTBF for an Ethernet network interface is 1 million hours [20]. With this information, we can determine the probability that these elements operate without failure during a time using Eq. (6). For example, for a period of 10 years, the probabilities that a cooper cable and Ethernet network interface remain available are 0.99825 and 0.91613 respectively.

Now, we can define p_l, the probability that link $l \in L$ is available with Eq. (7), (see Fig. 11).

$$p_l = (R_{nicSrc} \cap R_{cable} \cap R_{nicDst}) = 0.83783_{(10\ years)} \tag{7}$$

2) CONNECTIVITY MEASURES

Graph theory gives several measures to analyze and compare network efficiency over different topologies [21], [22]. In this paper, we use the diameter as delay metric, beta index as a complexity indicator, the number of edge-disjoint paths as resilience/reliability measure and the average path length as a robustness attribute.

- *Diameter (d_{max}):* Indicates the largest distance between any two nodes in the network, through the shortest path. Diameter (d_{max}) could be defined by Eq. (8) where $d(i, j)$ denotes the length of the shortest path between node i and j. This feature is important in our comparison because it gives a perspective about the behavior of the latency, since the latency is directly proportional to the distance $d(i, j)$. High values of d_{max} imply an important factor to take into account in the latency operation.

$$d_{max} = max\ d(i, j) \tag{8}$$

- *Beta Index(β):* It measures the level of connectivity in a graph and is expressed by the relationship between the number of links (l) over the number of nodes (n). Beta index can take values between 0 and 3. Simple network topologies like tree or cascade have a β value of less than one. A connected network with one cycle has a value of 1. More complex networks have a value greater than 1.

$$\beta = l/n \tag{9}$$

- *Average shortest path length ($\langle d \rangle$):* It is defined as the average of the shortest paths between all nodes in the network. For a directed network of N nodes, $\langle d \rangle$ gives the expected distance between two connected nodes. The lower the result, the more efficient the network in providing ease of circulation.

$$\langle d \rangle = \frac{1}{n(n-1)} \sum_{i \neq j} d(i, j) \quad \forall\, i, j \in V \tag{10}$$

- *edge-disjoint paths ($P_{edge-disj}$):* Two paths are edge-disjoint (edge independent) if they don't share any edges.

3) END-TO-END DELAY

Latency is a critical factor in the operation of power substations communications networks and its performance is directly related to the number of hops (switches) per path. That is the reason why we consider its analysis in this paper. IEC 61850-5 [15], defines the maximum transmission times required for different services. For example, SV data values

and GOOSE trip commands are services with highest latency requirements (Transfer Time class TT6), which demand 3ms as transmission limit or End-To-End delay (ETE). The end-to-end delay is the elapsed time from the time a data packet is sent out from the source application layer, until it is completely received by the application layer in the destination node. However, it is very important to take into account that, according to IEC 61850-10 [23] and IEC 61850-90-4 [1], the ETE of 3ms is distributed as follows: 80 percent to the processing times in the IED stacks (2.4ms) and the remaining 20 percent (0.6ms) for the communication network.

In [1], [24], and [25], authors analyze quantitatively the latency caused by a single hop in a path evaluating the wireline propagation delay and the processing time of a packet at a switch (store and forward, switch fabric processing, and frame queuing processes).

- *Wireline Propagation Delay (T_{wl}:* Refers to the elapsed time taken by the packet to traverse the physical medium. It depends on the physical link length and propagation speed. Now, assuming that we will use Ethernet cable CAT-5E / CAT-6, which gives a propagation delay of 0.64C in a distance of 100mt, where C is the speed of light, we can define the T_{wl} as follows Eq. (11):

$$T_{wl} = \frac{distance}{prop.factor * C} = \frac{100mts}{0.64C} = 520ns \quad (11)$$

- *Store and Forward Delay (T_{sf}):* Refers to the elapsed time in the switch while the first packet is fully received and stored in memory, the packet is read back from memory, and the packet is forwarded to the output queue. This delay is proportional to the size of the frame forwarding and is inversely proportional to the bit rate. Now, assuming that we are working on a FastEthernet network (100Mbps) with compact Goose frames of 300bytes, the T_{sf} can be defined as follows Eq. (12):

$$T_{sf} = \frac{framesize}{bitrate} = \frac{300bytes}{100Mbps} = 24\mu s \quad (12)$$

- *Switch processing delay (T_{sw}):* Corresponds to a fixed value given by the vendor, which depends on the processing speed of switch chip. Generally, industrial Ethernet switches have a value around $8\mu s$.
- *Queuing delay (T_q):* It is the most difficult metric to determine because depends on the knowledge of all traffic patterns on a network (network load), at any moment. For this reason, some assumptions have to be made. Thus, on a loaded network, it is possible to assume that the likelihood of a frame already in the queue is proportional to the network load. Under this condition, the average queuing latency can then be estimated in Eq. (13), where $T_{sf(max)}$ is the store and forward latency of a full-size frame (1500 bytes). Now, a network with 50% load would have an average queuing latency of $60\mu s$.

$$T_q = \%_{load} \cdot T_{sf(max)} = (0.5)\frac{1500bytes}{100Mbps} = 60\mu s \quad (13)$$

Based on the analysis above, we propose the scenario of Fig. 12 to calculate the latency caused by links and interconnection devices (switches) on the communication network path. Particularly, the scenario is validated for network topologies with $d_{max} = 2$ in normal operation conditions (e.g. star, spider-web and dual-tree). However, the proposed methodology could be applied to the calculation of other distances or topologies.

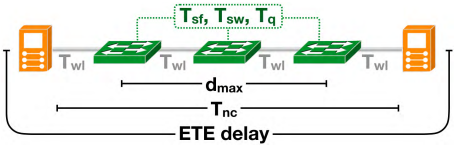


FIGURE 12. Elements involved in the estimate latency for End-To-End Delay.

According to Fig. 12, the latency related to the network communications T_{nc} will be given by Eq. (14),

$$T_{nc} = 4T_{wl} + 3(T_{sf} + T_{sw} + T_q) = 278\mu s \quad (14)$$

B. ANALYSIS OF RESULTS

The reliability of the obtained solution was evaluated using three different approaches: k-terminal reliability, graph metrics and ETE time-delay performance. Table 3 summarizes the results obtained for these metrics according to the network topologies defined for our use case (see Fig. 10).

1) TWO TERMINAL RELIABILITY

On each network topology (see Fig. 10), the two terminal reliability calculations were conducted for three different pairs of nodes $R_{(MNG,L3)}$, $R_{(L1,L3)}$, and $R_{(L2,L3)}$ to verify the behavior when nodes are located both, close or far away. The terminal reliability analysis showed that topologies with highest values of probability of maintaining nodes connected

TABLE 3. Two terminal reliability, connectivity and ETE analysis for different topologies.

TOPOLOGY	$R_{(MNG,L3)} \rightarrow P_i$	$R_{(L1,L3)} \rightarrow P_i$	$R_{(L2,L3)} \rightarrow P_i$	β	d_{max}	$\langle d \rangle$	$P_{edge-disj}$	$ETE\ (d_{max})$
Cascade	$0.346 \rightarrow 1$	$0.493 \rightarrow 1$	$0.837 \rightarrow 1$	0.86	6	2.66	1	$556\mu s$
Ring	$0.838 \rightarrow 2$	$0.791 \rightarrow 2$	$0.838 \rightarrow 2$	1.00	3	2.00	2	$370\mu s$
Star	$0.792 \rightarrow 1$	$0.792 \rightarrow 1$	$0.792 \rightarrow 1$	0.88	2	1.75	1	$278\mu s$
Mesh	$0.999 \rightarrow 326$	$0.999 \rightarrow 326$	$0.999 \rightarrow 326$	3.00	1	1.00	6	$186\mu s$
Redun. Ring	$0.900 \rightarrow 4$	$0.827 \rightarrow 4$	$0.900 \rightarrow 4$	1.14	7	3.04	2	$648\mu s$
Dual Redun. Tree	$0.973 \rightarrow 6$	$0.947 \rightarrow 10$	$0.947 \rightarrow 10$	1.56	2	1.47	2	$278\mu s$
Spider Web	$0.926 \rightarrow 29$	$0.953 \rightarrow 27$	$0.990 \rightarrow 23$	1.75	2	1.50	3	$278\mu s$

were: mesh (with a value of 0.999 on the three different pairs of nodes: $R_{(MNG,L3)}$, $R_{(L1,L3)}$, $R_{(L2,L3)}$); spider web (with values of 0.926, 0.953 and 0.990 for each pair of nodes) and dual redundant tree (with values of 0.973, 0.947 and 0.947 for each pair of nodes). However, although the mesh is the topology with greater redundancy in our study, its complexity, as shown by the beta index, is the worst in terms of network management; that is the reason why this topology is not implemented on real networks and it will be disregarded in our analysis. Hence, in terms of reliability, the spider web and the dual redundant tree are the network topologies offering the best performance.

Also, next to the two-terminal reliability value, the number of paths used to calculate P_i are indicated (see Table 3). There, if we compare the P_i values obtained by the spider web and the dual redundant tree network topologies, one can observe that the spider web topology gives two or even three times more paths to interconnect a pair of nodes than dual redundant tree topology (29 instead of 6, 27 instead of 10 and 23 instead of 10), according to the pair of nodes evaluated.

In this context, the network topology with best two terminal reliability is the spider web.

2) GRAPH METRICS

Let us begin this part of the analysis with the diameter d_{max} and the average shortest path length $\langle d \rangle$ metrics. These parameters give an idea about the proximity level of the network. Small values of these parameters indicate the best network performance because if, in average nodes are closer, the transmission path will have fewer elements, and that means a lower probability of system failure and shorter delays. According to Table 3, network topologies with smaller values for these metrics are dual redundant tree ($d_{max} = 2$, $\langle d \rangle = 1.47$) and spider web ($d_{max} = 2$, $\langle d \rangle = 1.50$), excluding the mesh topology for the previously presented reasons.

Another important parameter used in this analysis is the beta index (β). This index is a good instrument to address the complexity of the network. Values between 1 and 2 are appropriate, while that 3 it is the worst scenario (e.g. mesh topology). Topologies such as a ring ($\beta = 1$), dual ring ($\beta = 1.14$), dual redundant tree ($\beta = 1.56$) and spider web ($\beta = 1.75$) offer an appropriate complexity level. Nevertheless, ring and dual ring topologies did not get a good qualification under the first analysis criteria (terminal reliability), even when they offer two edge-disjoint paths. Consequently, dual redundant tree and spider web topologies continue displaying the best metrics.

Last graph metric to present in this discussion is the number of edge-disjoint paths between a pair of nodes ($P_{edge-disj}$). This graph parameter brings a determining factor in this evaluation in terms of reliability requirements. Table 3 shows that spider web topology with $P_{edge-disj} = 3$, offers the best option above other topologies, without taking into account the mesh topology. In this terms, although dual redundant tree and spider web topologies give similar values to graph meters d_{max}, $\langle d \rangle$ and β, the parameter $P_{edge-disj}$ is key to select the spider web topology.

3) ETE TIME-DELAY

As a third analysis criteria, the latency is an important indicator. The ETE time-delay analysis showed that dual redundant tree and spider web topologies obtained a value of $ETE(d_{max}) = 278\mu s$, assuming that all switches of the path had a network load of a 50% and worked with FastEthernet technology. This measure satisfies the communication network time defined for a IEC 61850-5 standard of $600\mu s$, equivalent to the 20% of the transmission time required for critical services (3ms). However, it is important to mention that: 1) the link delay contribution is negligible in comparison with the delay caused by the switch operation and, 2) the latency values calculated could be reduced implementing GigaEthernet technology, and with this, the margin of reliability of the network would be increased.

In summary, the results confirm that network resulting from the ILP optimization, spider web, is highly reliable. But, dual redundant tree showed being a good option too.

VI. USE CASES IMPLEMENTATION WITH SDN

As it was mentioned before, a substation communications network has to be designed with redundancy principles in order to guarantee fault-tolerance. However, managing the proposed topology in a traditional way, using common network mechanisms like Spanning Tree Protocol (STP) or Virtual Local Area Network (VLAN), would be technically unfeasible. On the one hand, in order to offer fault-tolerance the traffic must have the possibility to traverse different paths; however, on the other hand, STP will disable all redundant paths in the topology to avoid possible loops. This fact becomes a motivation to show the benefits of working in an SDN context.

Recently, SDN developments have also emerged around communication networks in power substations with the purpose of improving the management, availability and reliability inside this type of networks [26]. Likewise, SDN has also started to be applied into the entire smart grid with the purpose to increase the efficiency and resiliency of the smart grid systems [27]. Table 4 highlights the research proposals addressed to the study of the application of SDN in the management and operation of communication networks, within the power substations.

This section illustrates how with SDN we can develop applications to solve particular problematics within loop-based topologies such as SV and GOOSE multicast traffic management, and ARP broadcast traffic control; according to the chosen topology and the operational requirements of the power substation communications network.

All proofs of concept presented here were designed and implemented for the T1-1 power substation (see Fig. 8), using the spiderweb network topology (see Fig. 9) in accordance with the analysis exposed in the previous section

TABLE 4. Summarized overview of SDN solutions in power substation communication networks.

	Title (Country, Year)	Contribution
Proof of concept approaches	Software-defined energy communication networks (SDECN): From substation automation to future smart grids [28] (US, 2013)	SDECN. System to facilitate the management of power substation communication networks through a novel system of discovery and autoconfiguration of the network with SDN. Modules: self-discovery and monitoring.
	SMARTFlow: Uma Proposta para a Autoconfiguração de Redes de Subestação IEC 61850 Baseada em OpenFlow [29] (BR, 2014)	Self-configuring system for network management based on the IEC 61850 standard that reduces network load by up to 44 %, compared to traditional solutions. Modules: routing based on multicast trees, traffic prioritization, failure detection and loops (STP), and self-discovery.
	Using Software Defined Networking to manage and control IEC 61850 based systems [30] (ES, 2015)	Network architecture for power substations based on the IEC 61850 standard, supported by SDN operating principles. Modules: ShortestPath routing, traffic filtering, monitoring, quality of service, load balancing and security (firewall, anomaly detection and spoofing control).
	Software defined power substations: An architecture for network communications and its control plane [3] (CO, 2016)	A concentric model called S3N, Smart Solution for Substation Network, that proposes a reconceptualization of the power substations communications network architecture, taking the communications network as the central point and the SDN paradigm as a key element of its formulation. Modules: network topology discovery, network virtualization, monitoring (failure reporting and traffic statistic), distributed controller, security, traffic engineering, multicast and broadcast traffic management in loop-based topologies, path redundancy, packet redundancy and quality of service.
	Substation Security Mechanism based on SDN and MMS [31] (BR, 2018)	This work proposes IED authentication, authorization and control (AAA) mechanisms in a SDN context. Within the proposed architecture the following modules are identified: discover topology, spanning tree, best path, check MMS, authentication, rule release, and fault recovery.
Evaluation of technological requirements	Software-defined networking for Smart Grid communications: Applications, challenges and advantages [32] (DE, 2014).	It demonstrates the benefits of incorporating SDN to the SmartGrid communication networks, compared to the traditional mechanisms, providing a reliable communication network capable of handling complex failure scenarios. Use cases: link disruption and congestion management through bandwidth reservation and dedicated links.
	Evaluation of software defined networking for power systems [33] (AT, 2014)	SDN and OpenFlow offer important advantages. However, its use is not recommended in production environments according with the found results. It is expected that in the future SDN may be incorporated into the operation of the energy communication networks. Use cases: Link failure and traffic overload.
	Managing path diversity in Layer 2 critical networks by using OpenFlow [34] (ES, 2015).	SDN solution to use several routes simultaneously, increasing the performance of the network, through the combination of load balancing and multiple trajectory techniques. As a validation mechanism, robust network topologies were implemented that meet the operation and recovery requirements defined by the IEC 61850 standard.
	Reliable and flexible communications for power systems: Fault-tolerant multicast with SDN/OpenFlow [35] (AT-BR, 2015).	A robust multicast forwarding solution for substation environments using OpenFlow. This proposal uses the fast-failover groups feature of OpenFlow to provide one-link fault tolerance with little packet loss and can provide routes that use resources efficiently and are less likely to fail.
	SDN based dynamic and autonomous bandwidth allocation as ACSI services of IEC61850 communications in smart grid [36] (CN, 2016)	Novel scheme for bandwidth allocation based on SDN and IEC 61850 standar. It guarantees an adequate management of the bandwidth of MMS, GOOSE and SV traffic. Modules: Mapping of network services to ACSI services, traffic classifier, quality of service.
	Software Defined Networking enabled resilience for IEC 61850-based substation communication systems [37] (US, 2017)	SDN framework that incorporates security risk model, mitigation policies and end to end QoS to ensure resilience in a IEC 61850-based substation communication network.
	QoS Proposal for IEC 61850 traffic under an SDN environment [38] (CO-ES, 2018)	An architecture to implement QoS policies and guaranteeing the time-critical requirements of power substation protocols such as GOOSE, under an SDN environment.
Industry	Watchdog Project [39] and Software Defined Networking (SDN) Project [40] (US, 2018).	Alliance between a governmental entity (US Department of Energy, DoE) and a private entity (Schweitzer Engineering Laboratories, SEL), for developing solutions that improve the management of power substation communication networks with SDN.

and the principles of the S3N architecture [2]. The algorithms proposed were implemented under the Opendaylight SDN controller [41], while the topology was built using Mininet [42], a network emulator widely used in the SDN research networking field. The measuring (MUs), protection and control (IEDs) and supervising devices were emulated using *hosts-containers* executing applications from the libIEC61850 project [43]. libIEC61850 is an open-source framework with code written in C that provides an API for implementing MMS server and client, GOOSE publisher and subscribers and SV publisher and subscribers.

Finally, the application of these algorithms can be evidenced in [38], an SDN architecture for QoS provisioning in power substations communications network.

A. BROADCAST TRAFFIC CONTROL IN LOOP-BASED TOPOLOGIES

To mitigate the broadcast storm problem related to the broadcast protocols in loop-based topologies, networking engineers traditionally implement solutions like STP which introduces associated difficulties: redundant links will be disabled to all traffic types even knowing that it is only necessary to block the broadcast traffic. Instead, since the SDN controller is fully aware of network topology, we can take advantage of this knowledge to detect loops and use programmable techniques to handle the broadcast traffic.

In particular, we present an SDN solution to handle the Adress Resolution Protocol (ARP) broadcast traffic, without blocking other traffic types, building an exclusive graph tree

Algorithm 1 ARP Manager in an SDN Context

Input: Network Graph $G(N, L)$ where N are switches and L links
Output: OpenFlow rules set to install in the switches

1: Initialize variables:
   rootNode⟨N⟩, *vertex*⟨N⟩, *neighbor*⟨N⟩, *queueNodes*⟨*Queue of* N⟩,
   foundNodes⟨*List of* N⟩, *visitedNodes*⟨*List of* N⟩,
   linksBFSTree⟨*List of* L⟩, *linksToBlockArp*⟨*List of* L⟩
2: **for each** *switch* $\in G(N, L)$ **do**
3: **if** *switch* degree is greater than 3 **then**
4: *rootTree* ← *switch*
5: **end if**
6: **end for**
7: *queueNodes* ← enqueue *rootNode*
8: *foundNodes* ← *rootNode*
9: **while** *queueNodes* is not empty **do**
10: *vertex* ← front of *queueNodes*
11: **for each** (*neighbor* of *vertex*) $\in G(N, L)$ **do**
12: **if** *neighbor* is not in (*foundNodes* AND *visitedNodes* **then**
13: *linksBFSTree* ← *link*(*vertex*, *neighbor*)
14: *queueNodes* ← enqueue *neighbor*
15: *foundNodes* ← *neighbor*
16: **end if**
17: **end for**
18: *visitedNodes* ← *vertex*
19: *queueNodes* ← dequeue
20: **end while**
21: *linksToBlockArp* ← (all links $L \in G - linksBFSTree$)
22: **for each** *switch* in G **do**
23: Add flow priority = 5, dl_type = *arp_request*, actions = *flood*
24: **for each** *switchPort* in *switch* **do**
25: **if** *switchPort* belong to *linksToBlockArp* **then**
26: Add flow priority = 10, dl_type = *arp*, in_port = *switchPort*, actions = *drop*
27: **end if**
28: **end for**
29: **end for**

to manage the ARP traffic since the SDN controller has a centralized view of the communication network. However, this use case can be extended to handle other IPv4 broadcast protocols such as: Dynamic Host Configuration Protocol (DHCP), Server Message Block (SMB), etc.

This application uses the Breadth First Search (BFS) Algorithm [44], taking the node located in the center of a spider web topology as root. BFS is an algorithm for traversing or searching elements in a graph, often used on trees. It starts selecting a node as tree root and exploring all the neighbors of this node. Then for each neighbor discovered explores their respective adjacent neighbors, and so on until the whole tree is traveled. The proposed structure for this application is shown in Algorithm 1.

At the beginning, we build a network graph with the topology information contained in the SDN controller. Then, on lines 2 to 6, we determine the root node checking the connectivity degree of each switch (node) in the topology graph. Our interest is to start at the center of the topology, where this root node is the only one inside the topology with a connectivity degree greater than three. Here, it is important to mention that, according to the S3N architecture, we will always have a switch for the Management sector, another one for the Peripherical communications sector and the remaining ones for each bay. In this context, the root node will always have a connectivity degree greater than three.

Lines 7 to 20 implement the BFS algorithm to obtain the links belong to the BFS tree, *linksBFSTree*. With this information, on line 21, we obtain the links that do not belong to the BFS tree, *linksToBlockArp*, subtracting them from all links of spiderweb stored in *linksBFSTree*.

Finally, lines 22 to 29 are in charge of applying flows to the switches through some SDN southbound (OpenFlow, OVSDB). In short, when an ARP frame of type REQUEST arrives to any switch, the switch sends the frame out all interfaces, not including the incoming interface (FLOOD action). In contrast, when an ARP frame of any type arrives to a switch interface belonging to the source or destination link of *linksToBlockArp*, that frame will be dropped independently of the ARP type (REQUEST, REPLY, etc). Fig. 13 shows, step by step, the operation scheme of the Algorithm 1.

In addition, the Algorithm's behavior was evaluated in two failure conditions: 1) turning off one link belonging to the BFS tree and 2) disabling the central switch of the topology. In both scenarios the network topology connectivity was guaranteed, and the flows were installed properly according to the conditions defined in the algorithm (see Fig. 14).

B. SV MULTICAST TRAFFIC MANAGEMENT

SV communication service is a protocol used to transmit analog values (current and voltage) from the MUs to any IEDs in the power substation. This service uses a publisher/subscriber mechanism where the publisher (MU) sends information to one, or more subscribers simultaneously (IEDs), taking advantage of the multicast functionality provided by Ethernet. A publisher (MU) always is sending SV messages, which are available to any device that requires them (IEDs) by subscribing the corresponding publisher. Also, in the SV communication service there is not retransmission protocol, a lost sample is overwritten by the next successful one.

In traditional networks, a switch handles the multicast traffic in the same way that it deals with broadcast traffic. That means, the multicast frames flood all interfaces the switch, except for the interface where the frame arrived. But this behavior is unacceptable in a critical network. To mitigate the issue described, network administrators implement solutions based on Internet Group Management Protocol (IGMP) or VLAN technologies to handle any multicast traffic efficiently. However, the right set-up of multicast traffic is hard since the administrator has to configure the network devices one by one and a misconfiguration can cause erratic operations. In this context, there is a new opportunity to illustrate the benefits of operating under SDN environment developing customized solutions. Here, we get the most out of graph theory to develop an SV multicast traffic management module. This module takes advantage of the fact that the SDN controller has an entire view of the communication network and the knowledge of the publishers and subscribers along with their relationships (which IEDs are subscribed to a specific MU).

In particular, we present an SDN solution to calculate SV graph trees for the adequate propagation of the SV multicast traffic. In this way, it is guaranteed that this traffic does not flood the network and we can have as many SV trees as SV publisher/subscriber groups since each Publisher will have its own propagation tree (*svTree*). A svTree tree will include: a root node that corresponds to an SV traffic generator device or Publisher (MU); sheets that correspond to IEDs devices or Subscribers; intermediate nodes that correspond to

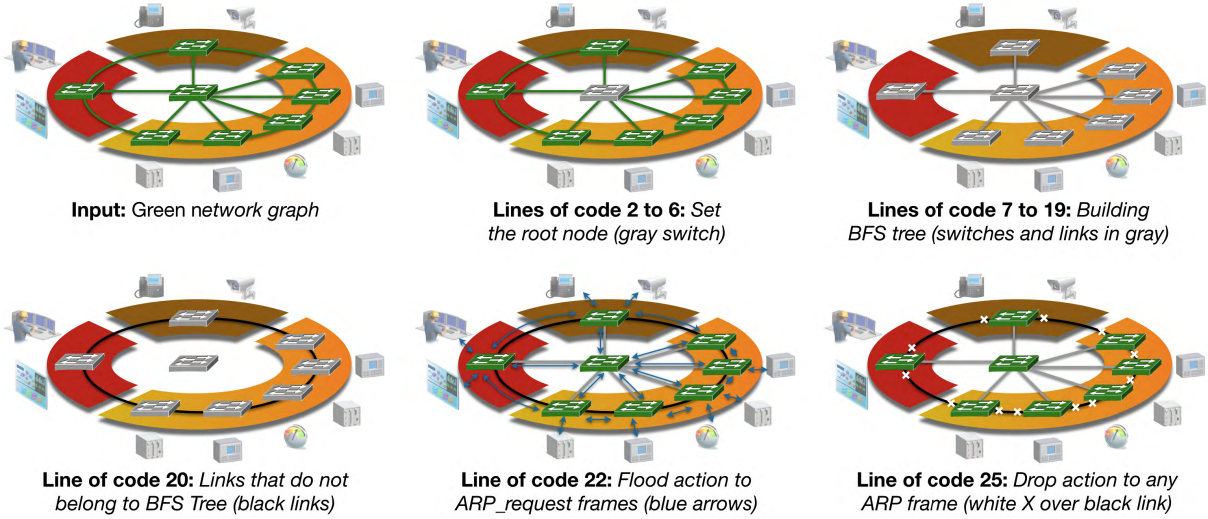


FIGURE 13. Operation scheme to manage ARP traffic.

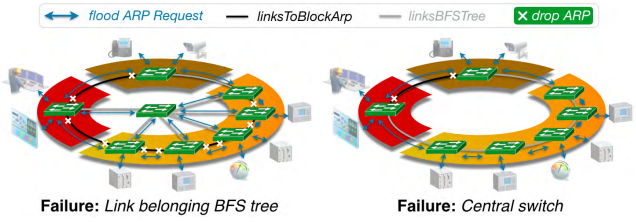


FIGURE 14. Algorithm behavior with node and link failure conditions.

Algorithm 2 SV Manager in an SDN Context

Input: Network Graph $G(N, L)$ where N are switches, MUs or IEDs, and L links
svPublisher and his corresponding *svSubscribers* list
sv_multicast_address
Output: Tree Graph *svTree(N,L)* where N are switches, MUs or IEDs, and L links
OpenFlow rules set to install in the switches

```
1: Initialize variables:
   svPublisher⟨N⟩, svSubscribers⟨List of N⟩, subscriber⟨N⟩,
   path⟨List of L⟩, branch⟨G(N, L)⟩, svTree⟨G(N, L)⟩, rootNode⟨N⟩,
   vertex⟨N⟩, neighbor⟨N⟩, queueNodes⟨Queue of N⟩,
   foundNodes⟨List of N⟩, visitedNodes⟨List of N⟩,
   link⟨L⟩, switchPorts⟨List of ports⟩
2: for each subscriber in svSubscribers do
3:     path ← findPath(svPublisher, subscriber)
4:     branch ← buildBranch(path)
5:     svTree ← addBranch(branch)
6: end for
7: rootNode ← findRootTree(svTree)
8: queueNodes ← enqueue rootNode
9: foundNodes ← rootNode
10: while queueNodes is not empty do
11:     vertex ← front of queueNodes
12:     for each (neighbor of vertex) ∈ svTree do
13:         if neighbor is not in (foundNodes AND visitedNodes) then
14:             if vertex ∈ switches then
15:                 link ← getLink(vertex, neighbor)
16:                 switchPorts ← addSwitchPort(vertex,
                                                link)
17:             end if
18:             queueNodes ← enqueue neighbor
19:             foundNodes ← neighbor
20:         end if
21:     end for
22:     visitedNodes ← vertex
23:     queueNodes ← dequeue
24:     if vertex ∈ switches then
25:         Add flow dl_src = svPublisher dl_dst
          = sv_mcast_address,
            actions = output:switchPorts
26:         switchPorts ← clear
27:     end if
28: end while
```

switches and branches that are only the links through which this traffic will be transmitted.

This module uses the BFS Algorithm [44] to explore the svTree calculated, taking the publisher node as root. Then for each intermediate node discovered in the tree (switches), flows rules are applied to guarantee the adequate forwarding of SV traffic. The proposed structure for this application is shown in Algorithm 2 and Fig. 15. This algorithm only illustrates the building of a single SV multicast group, it means one publisher and their corresponding subscribers.

Initially, the module takes as input the network graph $G(N, L)$ built with the topology information contained in the SDN controller along with information related to the SV multicast group (publisher, subscribers and SV multicast address).

Then, on lines 2 to 6, we build a tree graph called *svTree* sequentially. First, for each subscriber of the SV multicast group, we find the shortest *path* between the publisher and the respective subscriber. This *path* will be used to get a single tree branch. Later, each branch is added to the *svTree* structure. Here, it is important to mention that each *svPublisher* has associated a single *svTree*. This feature becomes important in the future in the presence of link failures in the topology because, when the failed link is identified, we can easily determine using graphs theory which SV trees were affected and re-calculate only these specific trees. Thus, the system response is optimized. In traditional networks, with the aforementioned behavior, this type of characteristic is unfeasible.

Lines 7 to 28 implement once again the BFS algorithm. But, in this case, we use the algorithm to explore the *svTree* calculated and not to build a specific tree (see lines 7 to 20 in Algorithm 1). In the tree discovery process, we identify the intermediate nodes (switches) on line 14 and we take note of the switch ports related to the *svTree* branches

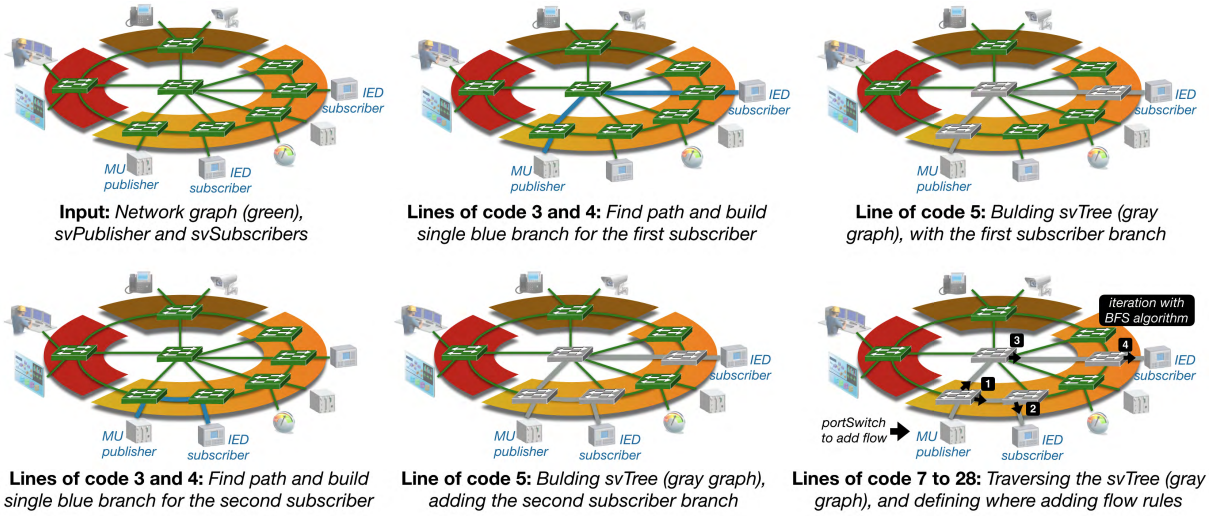


FIGURE 15. Operation scheme to build the *svTree* and handle the SV traffic.

(lines 15 and 16). This information is important because it will be used on line 25 to apply flow rules to the switches through some SDN southbound (OpenFlow, OVSDB). In this occasion, we uniquely identify the SV flows using only the Ethernet MAC addresses. Although it is possible to add another field such as EtherType or the SV message identifier svID, we prefer to make the match as simple as possible. Also, note that the flow action output allows sending the same SV message through different ports at the same time.

Last, although the Algorithm 2 only shows the procedure to build a single *svTree*, the SV multicast traffic management module reuses this algorithm as many *svTree* as required. All SV tree graphs calculated are stored to guarantee a fast recovery process as we mentioned before.

C. GOOSE MULTICAST TRAFFIC MANAGEMENT

GOOSE communication service is a protocol used with the purpose of distributing information triggered by events (commands, alarms, indications, trip messages), between all IEDs across the entire power substation communication network. Particularly, the fast and reliable transmission of GOOSE trip messages is an important feature within the power substation because these messages carry out critical information (for example, the instruction to trigger a breaker and protect the power substation in case of a transient in a high voltage power line). In this context, to guarantee a fast event-driven transmission, GOOSE messages are exchanged at layer 2 (link layer), optimizing the processing times in the sender (encoding) and the receiver (decoding). Like SV service, in GOOSE the information exchange is based on a publisher/subscriber mechanism using the multicast functionality provided by Ethernet. However, since GOOSE messages are multicast, they are not acknowledged by the destination. To overcome this multicast feature, a GOOSE trip message is retransmitted several times in a row to rectify possible frame losses.

As we mentioned before, GOOSE is a critical communication service with high requirements of reliability. To get this feature, traditional networks in power substation use PRP or HSR, two recovery protocols defined by the IEC 62439-3 standard that provide zero recovery time with no packet loss.

The basic concept behind PRP is that a device is connected to two independent networks. Any message this device publishes is mirrored to both networks. Subscribing devices, also connected to both networks, will accept the first version of the message received, and discard the second version. If one network link fails, the mirrored message will still go through on the second network. The two networks do not need to be identical, but they must not be connected to each other (see Fig. 16) [45].

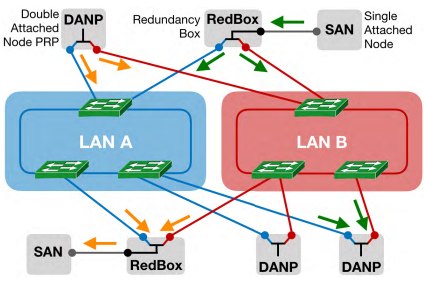


FIGURE 16. PRP concept. Redundancy based on duplicate LANs.

The basic concept behind HSR is that all devices are connected in a ring topology, without switches. Any message from the publishing device is duplicated, and sent both directions around the ring. A subscribing device accepts the first version of the message received, and discards the second version. If a network link fails, the version of the message traveling the other direction around the ring will be received and used (see Fig. 17) [45].

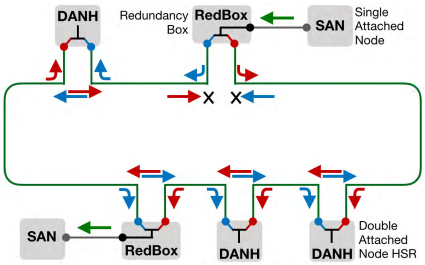


FIGURE 17. HSR concept. Redundancy based on ring topology.

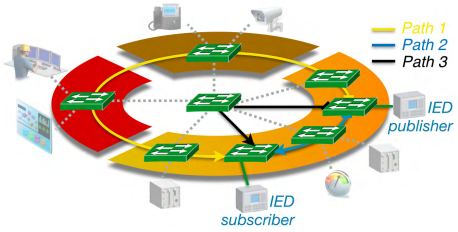


FIGURE 18. Three edge-disjoint paths for GOOSE traffic on spiderweb network topology.

Although PRP and HSR provide a high-availability network, they also have weaknesses. PRP requires two completely isolated networks for its operation. It means, a double capital investment, with the addition of the cost of the new devices (DANP and Redbox). This, without mentioning the technical aspects related to the configuration. For example, PRP operates over standard Ethernet switches, therefore it is also necessary to define how to handle the multicast traffic within each network: VLAN or IGMP. On the other hand, HSR requires specialized LAN nodes for its operation and HSR Ethernet frames are not compatible with standard Ethernet frames. So, there is no compatibility with other Ethernet networks. In addition, the number of nodes connected to an HSR ring is limited by the node port that has the least bandwidth. This represents a scalability issue.

In this context, there is a challenge about how to handle GOOSE traffic under an SDN environment with high requirements of reliability (zero recovery time with no packet loss). However, we already advanced in overcoming this matter since we define a reliable network topology (the spiderweb), considering criteria such as SDN environment and edge-disjoint paths. Here, we will explain the operating principles of the GOOSE multicast traffic management module under SDN perspective. This module emulates the functionalities of zero recovery time protocols in a single network topology, without the need to use additional devices (Redbox or HSR interface) and with Ethernet networks compatibility. In short, this module is in charge of guaranteeing path redundancy and packet redundancy.

- *Path redundancy.* This functionality allows the discovery of redundant paths between two nodes. In addition, when a path is defined to forward the GOOSE traffic, this implies that there is no indication of flooding process related to multicast traffic. The spiderweb network topology provides three edge-disjoint paths between two end-nodes located in different bays: one path passing through the central switch (path 3) and two paths traversing the ring in opposite directions (paths 1 and 2), as shown in Fig. 18.
- *Packet redundancy.* This feature allows sending duplicate packets through two edge-disjoint shortest paths.

In particular, we present an SDN solution to calculate GOOSE directed graphs for the adequate propagation of the GOOSE multicast traffic. Our approach computes two edge-disjoint shortest paths in the spiderweb topology and sends duplicate information through them for each publisher-subscriber pair. This is possible since the SDN controller has an entire view of the communication network and it also has information about the publishers and subscribers relationships (which IEDs are subscribed to a specific IED publisher).

This module uses a modified version from BFS algorithm to explore the calculated *directedGraph*, taking the publisher node as the first node to start the procedure. Then for each intermediate node discovered in the directed graph (switches), flows rules are applying to guarantee the adequate forwarding of GOOSE traffic. The structure proposed for this application is shown in Algorithm 3. This algorithm only illustrates the building of a single GOOSE multicast group, it means one publisher and their corresponding subscribers.

At the beginning, like SV multicast traffic module, the GOOSE module takes the network graph $G(N, L)$ along with information related to the GOOSE multicast group (publisher, subscribers and GOOSE multicast address).

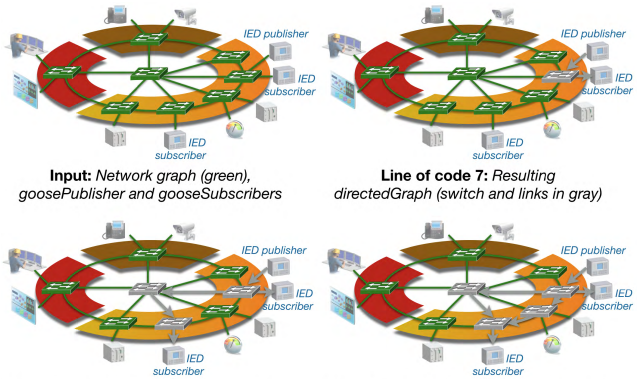
Then, on lines 2 to 16, we build a directed graph called *directedGraph* merging tree directed graphs according to several criteria. 1) If the publisher and the subscriber are connected to the same node (switch), the tree corresponds to the simple path between the publisher and the subscriber. 2) If the publisher and the subscriber are located on different bays we merge two tree graphs to the *directedGraph*, one tree passing through the central switch and another tree traversing the shortest path through the ring. In a similar way, just as the SV multicast traffic management, each *goosePublisher* has a directed graph associated. This feature contributes to optimize the system response since, in case of link failures, the broken link could be identified and, using graph theory, we can determine easily which GOOSE directed graphs were affected and re-calculate only these specific structures. Fig. 19 shows snapshots in the building of the GOOSE *directedGraph*.

Lines 17 to 37 implement a modified version on the BFS algorithm to traverse the *directedGraph* related with a GOOSE group (publisher and their corresponding subscribers). In this directed graph, *successorNode* is a reachable node from *predecessorNode* while *predecessorNode* is the node that discovers a *successorNode*. Like in Algorithm 2, we identify the intermediate nodes (switches) on line 22 and

Algorithm 3 GOOSE Manager in an SDN Context

Input: Network Graph $G(N, L)$ where N are switches, MUs or IEDs, and L links
goosePublisher and his corresponding *gooseSubscribers* list
goose_mcast_address
Output: Directed Graph *directedGraph*(N, L) where N are switches, MUs or IEDs, and L links; and OpenFlow rules set to install in the switches

1: Initialize variables:
publisher⟨N⟩, *gooseSubscribers*⟨*List of* N⟩, *subscriber*⟨N⟩,
path⟨*List of* L⟩, *centerPath*⟨*List of* L⟩, *ringPath*⟨*List of* L⟩,
branch⟨$G(N, L)$⟩, *centerBranch*⟨$G(N, L)$⟩, *ringBranch*⟨$G(N, L)$⟩,
directedGraph⟨$G(N, L)$⟩, *predecessorNode*⟨N⟩, *successorNode*⟨N⟩,
queueNodes⟨*Queue of* N⟩, *foundNodes*⟨*List of* N⟩, *link*⟨L⟩,
visitedNodes⟨*List of* N⟩, *switchPorts*⟨*List of ports*⟩
2: *publisher* ← *goosePublisher*
3: **for each** *subscriber* in *gooseSubscribers* **do**
4: **if** (*publisher* AND *subscriber*) are connected to the same switch **then**
5: *path* ← *findPath*(*publisher*, *subscriber*)
6: *branch* ← *buildBranch*(*path*)
7: *directedGraph* ← *mergeBranch*(*branch*)
8: **else**
9: *centerPath* ← *findPathViaCentralNode*(*publisher*, *subscriber*)
10: *centerBranch* ← *buildBranch*(*centerPath*)
11: *directedGraph* ← *mergeBranch*(*centerBranch*)
12: *ringPath* ← *findShortestPathViaRing*(*publisher*, *subscriber*)
13: *ringBranch* ← *buildBranch*(*ringPath*)
14: *directedGraph* ← *mergeBranch*(*ringBranch*)
15: **end if**
16: **end for**
17: *queueNodes* ← enqueue *publisher*
18: *foundNodes* ← *publisher*
19: **while** *queueNodes* is not empty **do**
20: *predecessorNode* ← front of *queueNodes*
21: **for each** (*successorNode* of *predecessorNode*) ∈ *directedGraph* **do**
22: **if** *predecessorNode* ∈ switches **then**
23: *link* ← *getDirectedLink* (*predecessorNode*, *successorNode*)
24: *switchPorts* ← *addSwitchPort*(*predecessorNode*, *link*)
25: **end if**
26: **if** *successorNode* is not in (*foundNodes* AND *visitedNodes*) **then**
27: *queueNodes* ← enqueue *successorNode*
28: *foundNodes* ← *successorNode*
29: **end if**
30: **end for**
31: *visitedNodes* ← *predecessorNode*
32: *queueNodes* ← dequeue
33: **if** *predecessorNode* ∈ switches **then**
34: Add flow dl_src=*publisher* dl_dst=*goose_mcast_address*, actions=output:*switchPorts*
35: *switchPorts* ← clear
36: **end if**
37: **end while**



Input: *Network graph (green), goosePublisher and gooseSubscribers*

Line of code 7: *Resulting directedGraph (switch and links in gray)*

Line of code 11: *Resulting directedGraph (switches and links in gray)*

Line of code 14: *Resulting directedGraph (switches and links in gray)*

FIGURE 19. **Snapshots in the building of the GOOSE *directedGraph*.**

store in *switchPorts* the ports used to reach the successor nodes (lines 23 and 24). This information will be used on line 34 to set flow rules to the switches through some SDN southbound interface (OpenFlow, OVSDB). Once again, we uniquely identify the GOOSE flows using only the Ethernet MAC addresses, though it is possible to add another field such as EtherType or the application identifier appID. Also, note that the flow action output allows sending the same GOOSE message through different ports at the same time.

Finally, it is important to mention that we are sending duplicate information through different paths with the purpose to guarantee zero recovery time with no packet loss. In other words, the GOOSE subscribers are receiving GOOSE message duplicates all the time. This behavior implies a reduction in the network efficiency at expense to provide high levels of availability and reliability. However, although our algorithm sends duplicated information just like the PRP and HSR media redundancy technologies, it reduces the number of equipment to be used and brings Ethernet compatibility. Here it is important to mention that power substations are considered critical infrastructures because, a failure in a substation, may involve the interruption of the operation of other systems (traffic lights, public lighting, telecommunication services, among others), which can seriously affect the physical and/or economic security of the people. In this context, power substation communications networks must have high reliability and fault tolerance, so they can guarantee the continuity of the operation in the face of sabotage, a natural disaster or network failures. For this reason, the redundancy is a desirable feature at the expense of the network efficiency. Ideally, in our proposal the switch that connects the GOOSE subscriber should eliminate the duplicate GOOSE messages. However, the SDN environment does not allow interacting with the kernel-switch to set a discard function. Therefore, it will be a task of the GOOSE subscriber to detect and delete the duplicated messages. So, to reach this functionality, we modified the code associated with the GOOSE subscriber from the libIEC61850 library [43].

VII. CONCLUSIONS

The selection of the requirements to define the right network topology changes according to the purpose for which a network is built, and they are the key to a good design. However, a right balance between the requirements is not easy to get. Reliability, efficiency, costs, real-time performance and management are features that can not be satisfied simultaneously.

In this paper, we propose an ILP model to solve the problem of reliability for a network topology in a Software-defined power substations context, taking into account requirements such as SDN environment (S3N architecture), edge-disjoint paths (redundancy) and IEC 61850-90-4 recommendations (backward compatibility). Our studio corroborated that the proposed solution, the spider web topology, is a reliable network topology that allows to improve the performance of the operation network, by using three different analytical approximations: terminal reliability, graph metrics and ETE time-delay performance; and its comparison against several practical Ethernet architectures. In addition, our solution emulates the behavior of traditional recovery protocols such as PRP or HSR, without the need to use additional devices (Redbox or HSR interface) or duplicated networks.

This article also shows how SDN applications can solve complex issues in loop-based topologies such as SV and GOOSE multicast traffic management and broadcast traffic control; which would be technically unfeasible using common network protocols. In particular, we develop applications

to define the traffic behavior in the proposed network topology. The achieved results, when the algorithms were applied over our testbed, were consequent with the expected operation scheme of the communication network. In addition, excepting the GOOSE algorithm which takes advantage of the particularities of the spider web topology, the others algorithms proved to work satisfactorily in a dual redundant tree topology.

REFERENCES

[1] *Communication Networks and Systems for Power Utility Automation, Part 90-4: Network Engineering Guidelines*, Standard IEC IEC/TR TC57 61850-90-4, 2013.
[2] E. A. Leal and J. F. Botero, "S3N-smart solution for substation networks, an architecture for the management of communication networks in power substations," in *Management and Security in the Age of Hyperconnectivity* (Lecture Notes in Computer Science), vol. 9701. Cham, Switzerland: Springer, 2016, pp. 52–56.
[3] E. A. Leal and J. F. Botero, "Software defined power substations: An architecture for network communications and its control plane," in *Proc. 8th IEEE Latin-Amer. Conf. Commun. (LATINCOM)*, Nov. 2016, pp. 1–6.
[4] *Communication Networks and Systems for Power Utility Automation*, Standard IEC 61850, 2010.
[5] Hirschmann Company. (2012). *Hirschmann White Paper, Data Communication in Substation Automation System (SAS). WP 1004HE—Part 3*. [Online]. Available: http://www.beldensolutions.com/de/Service/Downloadcenter/index.phtml
[6] N. Yadav and E. Kapadia, "IP and ethernet communication technologies and topologies for IED networks," Cisco, San Jose, CA, USA, Grid InterOp, Oct. 2010. [Online]. Available: https://www.gridwiseac.org/pdfs/forum_papers10/kapadia_gi10.pdf
[7] M. Kanabar, "Investigating performance and reliability of process bus networks for digital protective relaying," Ph.D. dissertation, Univ. Western, Ontario, London, ON, Canada, 2011.
[8] X. Liu, J. Pang, L. Zhang, and D. Xu, "A high-reliability and determinacy architecture for smart substation process-level network based on cobweb topology," *IEEE Trans. Power Del.*, vol. 29, no. 2, pp. 842–850, Apr. 2014.
[9] I. Ali, M. S. Thomas, S. Gupta, and S. S. M. Hussain, "IEC 61850 substation communication network architecture for efficient energy system automation," *Energy Technol. Policy*, vol. 2, no. 1, pp. 82–91, 2015.
[10] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[11] K. Menger, "Zur allgemeinen Kurventheorie," *Fundam. Math.*, vol. 10, no. 1, pp. 96–115, 1927.
[12] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory With Mathematica*. Reading, MA, USA: Addison-Wesley, 1990, p. 177.
[13] A. Makhorin. (2000). *Glpk (GNU Linear Programming Kit)*. [Online]. Available: http://www.gnu.org/software/glpk/
[14] Y. Aoyanagi and K. Okumura, "Simple model for the mechanics of spider Webs," *Phys. Rev. Lett.*, vol. 104, no. 3, p. 038102, 2010.
[15] *Communication networks and systems in substations—Part 5: Communication Requirements for Functions and Device Models*, Standard IEC 61850-5, 2013.
[16] F. Beichelt and P. Franken, *Reliability and maintenance: Networks and Systems*. Boca Raton, FL, USA: CRC Press, 2012.
[17] S. Hariri and C. S. Raghavendra, "SYREL: A symbolic reliability algorithm based on path and cutset methods," *IEEE Trans. Comput.*, vol. COMM-36, no. 10, pp. 1224–1232, Oct. 1987.
[18] W. Torell and V. Avelar. (2004). Mean Time Between Failure: Explanation and Standards. American Power Conversion. [Online]. Available: https://pdfs.semanticscholar.org/de97/955063b165d594e0aa0b55e6e550b51aa51a.pdf
[19] G. Shainer. (2015). *Copper Vs. Fiber: What's Best In The Next-Gen Data Center?* [Online]. Available: https://www.networkcomputing.com/data-centers/copper-vs-fiber-whats-best-next-gen-data-center/1410243954
[20] Cisco Systems. (2015). *Data Sheet, CISCO 1000BASE-T SFP*. [Online]. Available: https://www.future-x.de/images/datasheet/s101552.pdf
[21] K. Karl and P. Danscoine, "Measures of network structure," *Flux*, vol. 5, no. 1, pp. 89–121, 1989.
[22] A.-L. Barabási, *Network Science*. Cambridge, U.K.: Cambridge Univ. Press, 2016.
[23] *Communication Networks and Systems for Power Utility Automation—Part 10: Conformance Testing*, Standard IEC 61850-10, IEC, 2012.
[24] Siemens. (2014). *Latency on a Switched Ethernet Network*. [Online]. Available: https://support.industry.siemens.com/cs/attachments/94772587/RUGGEDCOM.pdf
[25] X. Xu and Y. Ni, "Analysis of networking mode caused by GOOSE delay of smart substation," in *Proc. 4th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, May 2013, pp. 503–506.
[26] E. A. Leal and J. F. Botero, "Transforming communication networks in power substations through SDN," *IEEE Latin Amer. Trans.*, vol. 14, no. 10, pp. 4409–4415, Oct. 2016.
[27] M. H. Rehmani, A. Davy, B. Jennings, and C. Assi. (2018). "Software defined networks based smart grid communication: A comprehensive survey." [Online]. Available: https://arxiv.org/abs/1801.04613
[28] A. Cahn, J. Hoyos, M. Hulse, and E. Keller, "Software-defined energy communication networks: From substation automation to future smart grids," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Oct. 2013, pp. 558–563.
[29] Y. Lopes, N. C. Fernandes, and C. A. Malcher, "Smartflow: Uma proposta para a autoconfiguração de redes de subestação IEC 61850 baseada em openflow," in *Proc. 29th Workshop Gerência Operação Redes Serviços (WGRS)*, 2014, pp. 31–44.
[30] E. Molina, E. Jacob, J. Matias, N. Moreira, and A. Astarloa, "Using software defined networking to manage and control IEC 61850-based systems," *Comput. Elect. Eng.*, vol. 43, pp. 142–154, Apr. 2015.
[31] A. C. Pigossi and Y. Lopes, "Substation security mechanism based on SDN and MMS," in *Proc. Simposio Brasileiro Sist. Eletricon. (SBSE)*, 2018, pp. 1–6.
[32] N. Dorsch, F. Kurtz, H. Georg, C. Hägerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Nov. 2014, pp. 422–427.
[33] T. Pfeiffenberger and J. L. Du, "Evaluation of software-defined networking for power systems," in *Proc. IEEE Int. Conf. Intell. Energy Power Syst. (IEPS)*, Jun. 2014, pp. 181–185.
[34] E. Molina, J. Matias, A. Astarloa, and E. Jacob, "Managing path diversity in layer 2 critical networks by using OpenFlow," in *Proc. 11th IEEE Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015, pp. 394–397.
[35] T. Pfeiffenberger, J. L. Du, P. B. Arruda, and A. Anzaloni, "Reliable and flexible communications for power systems: Fault-tolerant multicast with SDN/OpenFlow," in *Proc. 7th IEEE Int. Conf. New Technol., Mobility Secur. (NTMS)*, Jul. 2015, pp. 1–6.
[36] G. Li, J. Wu, L. Guo, J. Li, and H. Wang, "SDN based dynamic and autonomous bandwidth allocation as ACSI services of IEC61850 communications in smart grid," in *Proc. Smart Energy Grid Eng. (SEGE)*, Aug. 2016, pp. 342–346.
[37] H. Maziku and S. Shetty, "Software defined networking enabled resilience for iec 61850-based substation communication systems," in *Proc. IEEE Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2017, pp. 690–694.
[38] E. A. Leal, J. F. Botero, and E. Jacob, "QoS proposal for IEC 61850 traffic under an SDN environment," in *Proc. 10th IEEE Latin-Amer. Conf. Commun. (LATINCOM)*, Nov. 2018, pp. 1–6.
[39] Schweitzer Engineering Laboratories. (2018). *Watchdog Project*. [Online]. Available: https://www.selinc.com/workarea/downloadasset.aspx?id=104832
[40] Schweitzer Engineering Laboratories. (2018). *Software Defined Networking (SDN) Project*. [Online]. Available: https://www.selinc.com/WorkArea/DownloadAsset.aspx?id=109179
[41] Open Foundation. (2013). *OpenDaylight Controller*. [Online]. Available: https://www.opendaylight.org
[42] Mininet Team. (2012). *Mininet. An Instant Virtual Network on Your Laptop*. [Online]. Available: http://mininet.org
[43] M. Zillgith. (2016). *libIEC61850 Open Source Library for IEC 61850*. [Online]. Available: http://www.libiec61850.com/libiec61850/
[44] K. Zuse. (1948). *Der Plankalkül—Programming Language. Gesellschaft für Mathematik und Datenverarbeitung*. [Online]. Available: http://zuse.zib.de/item/gHI1cNsUuQweHB6
[45] R. Hunt and B. Popescu, "Comparison of PRP and HSR networks for protection and control applications," in *Proc. Western Protective Relay Conf.*, Spokane, WA, USA, 2015, pp. 1–44.

ALEXANDER LEAL received the B.S. degree in electronic engineering and the M.Sc. degree in engineering with major in telecommunications from the University of Antioquia, Medellín, Colombia, in 2001 and 2009, respectively, where he is currently pursuing the Ph.D. degree in electronic engineering. He was a full-time Faculty with the University of Santo Tomás, from 2001 to 2009. Since 2009, he has been an Assistant Professor with the Electronic Engineering Department, University of Antioquia, where he is currently with the Applied Telecommunications Research Group. His research interests include network security, SDN, and NFV.

JUAN FELIPE BOTERO received the B.S. degree in computer science from the University of Antioquia, Medellín, Colombia, in 2006, and the M.Sc. and Ph.D. degrees in telematics engineering from the Network Engineering Department, Technical University of Catalonia, Barcelona, Spain, in 2008 and 2013, respectively. Since 2013, he has been an Assistant Professor with the Telecommunications Engineering Department, University of Antioquia, where he is currently with the Applied Telecommunications Research Group. His current research interests include the Internet of the future, in particular network virtualization, mathematical programming, routing, energy efficiency and network flows, SDN, and NFV.