von Tüllenburg et al. *Energy Informatics* (2019), 2(Suppl 1): 14
https://doi.org/s42162-019-0086-y

Energy Informatics

RESEARCH Open Access

Virtualising redundancy of power equipment controllers using software-defined networking

Ferdinand von Tüllenburg[1*], Peter Dorfinger[1], Armin Veichtlbauer[2], Ulrich Pache[2], Oliver Langthaler[2], Helmut Kapoun[3], Christian Bischof[3] and Friederich Kupzog[4]

From The 8th DACH+ Conference on Energy Informatics
Salzburg, Austria. 26–27 September 2019

*Correspondence:
ftuellen@salzburgresearch.at
[1]Advanced Networking Center, Salzburg Research Forschungsgesellschaft mbH, Salzburg, Austria
Full list of author information is available at the end of the article

Abstract

Power system automation is heavily dependent on the reliable interconnection of power system field equipment and centralised control applications. Particularly important to achieve reliability in automated power systems is the redundant connection of field equipment controllers. Today, the fundamental redundancy management and switch-over processes are handled by the power system control applications itself. This, however, increases the complexity of the control systems and leads to an intersection of concerns. Consequently, the design and the implementation of field equipment controller redundancy is time-consuming and cost-intensive.
In this paper, we propose the implementation of a redundancy virtualisation layer for power system controllers based on software-defined networking (SDN). The goal is to relieve the control applications from managing field level redundancy. Thus, our SDN approach allows to detect gateway failures and to perform transparent switch-overs. This significantly simplifies the configuration of the control application with redundant components and finally leads to more flexible and simplified redundancy design, deployment and operation. Arbitrary redundancy topologies, such as triple-controller-redundancy can be implemented without modifying the control applications.

Keywords: Smart grid, Redundancy, Software-defined networking, Failover

Introduction

Electrical power systems are becoming more and more automated in order to account for the increasing decentralisation of the energy supply. Power system automation is of particular importance in substations, where sensors, actuators and control systems are situated and interlinked with supervisory control and data acquisition (SCADA) systems in control centres. Thus, data communication for sensor values and control commands is required between the central SCADA system here, and the sensors and actuators there, which are deployed in remote power system equipment such as a substation or directly in the field.

Power system operators usually maintain dedicated communication networks for monitoring and control purposes. In most cases nowadays, these networks are operated as pure

layer 2 domains, where VLANs (virtual local area networks) are used for segmentation. Recently, operators have started to develop their networks towards multiprotocol label switching (MPLS) networks which promise improvements in protocol independence, network resource management and failure mitigation (Deng et al. 2012).

Due to the critical nature of the power supply, operational safety and reliability play a crucial role in the design and the implementation of power system automation. This is particularly valid for the communication subsystem. Traditionally, redundancy in its various aspects is an important building block in primary substations when creating safe and reliable systems. For instance, it is best practice to implement the power system monitoring and control networks in a ring-of-rings topology, providing two-way path redundancy for each connected node. Furthermore, redundancy exists for equipment at the control centre and at the field level.

The ring-of-rings topology along with the set of layer 2 network protocols such as rapid spanning tree (RSTP) or MPLS fast re-route (MPLS-FRR) provides path redundancy and failover functionality at the network infrastructure. However, when it comes to failures of ICT equipment at power system level, these protocols cannot help, as knowledge about redundancy is only available at the application layer, but missing at the network layer.

Approach

In this paper, we focus on failover processes regarding the redundancy of control and automation equipment and its connectivity. Exemplary for many other cases, we consider the redundant connection of a remote controllable substation.

In Fig. 1, we show three variants of redundancy implementation. The first variant shows a traditional approach without redundancy, the second variant the state of the art redundancy solution, and the third variant our envisioned approach with multiple software-defined redundancy.

In such scenarios, typically, two levels according to the Smart Grid Architecture Model (SGAM) (Smart Grid Coordination Group 2012) can be distinguished: The station level and the operation level. The operation level corresponds to the necessary infrastructure for connecting a facility such as a substation to the SCADA system via the monitoring and control network, while the station level corresponds to the facility's internal infrastructure necessary to connect local equipment. At each level, this infrastructure basically consists of a communication network and multiple controller devices, denoted as intelligent electronic devices (IEDs).

In all of the considered variants, the station level of our example infrastructure is structurally identical. On the lowest level, the electrical equipment such as a disconnector is equipped with a remote terminal unit (RTU). These RTUs allow access to the equipment.

For remote access from central SCADA to the RTUs, a gateway component plays a central role as a bridge between the station and operation level. Monitoring data and control signals are exchanged between the networks, which also includes protocol translation. For instance, IEC 60870-5-104 is used for operation level communication while IEC 61850 is used at the station level.

From here, different variants for connecting SCADA and station arise. In the traditional single gateway approach (in Fig. 1), the RTUs are redundantly connected to only one gateway device, forming a single point of failure. As the potential loss of monitoring data and control capabilities has become critical with the development of smart grids, the

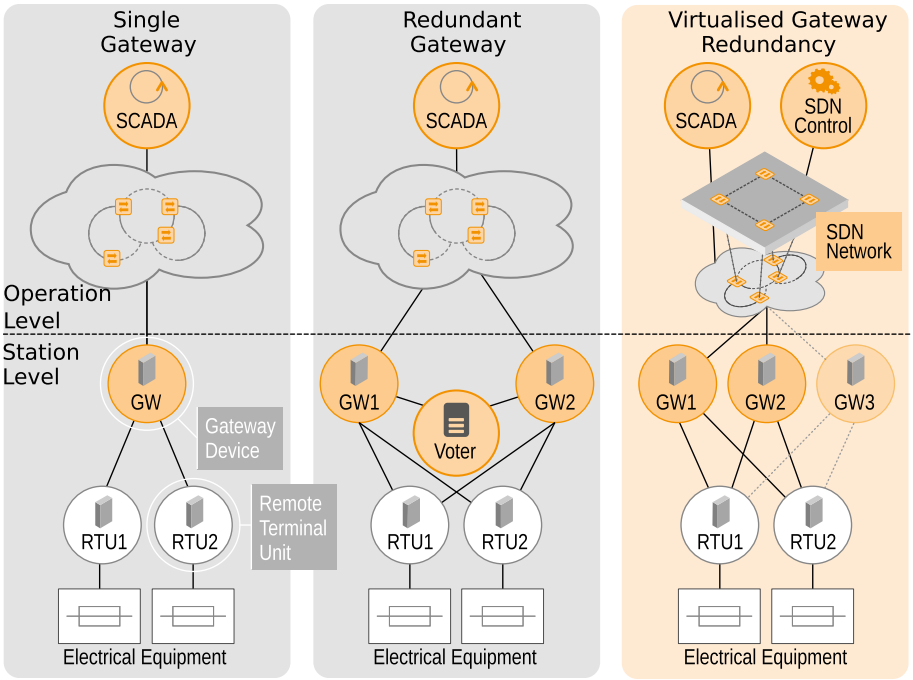

Fig. 1 Compared to traditional implementations (single and redundant gateway), virtualisation provides flexible redundancy solutions with low configuration efforts

reliability of the gateway devices has gained importance. If the gateway device fails, the whole station including all contained RTUs becomes inaccessable from central SCADA.

The second variant (redundant gateway) shows the state of the art approach for implementing gateway failover by using a voter component. The purpose of the voter is the selection of the most appropriate gateway as primary device, the active surveillance of its availability, and the initialisation of a switch-over in case of a failure of the primary device. This approach, however, has a number of shortcomings:

- The intersection of concerns due to the fact that the SCADA application (resp. the voter) must be aware of the gateways and their state, which are mainly communication devices, not EPS equipment.
- This also leads to dependencies between failover, the SCADA application, and employed communication protocols.
- The engineering of particular deployments is cost intensive due to required but scarcely available expert knowledge for configuration and particular protocols used in certain deployments.
- The voter-based approach only supports two-gateway-redundancy and other variants with three or more gateways would require the re-implementation of parts of the SCADA system.

Contribution

In this paper, we thus propose a third variant: The implementation of a redundancy virtualisation layer for power system controllers in general, and gateway devices in particular. The goal is to relieve the SCADA system from managing redundancy at the station level.

Our solution employs software-defined networking (SDN) to detect gateway failures and perform switch-overs transparently, without notifying the SCADA application. This remedies the aforementioned deficiencies, and finally leads to cost improvements with respect to design, deployment and operation of redundancy in SCADA infrastructures. To the best of our knowledge, our approach is the first one in that direction.

Based on our initial concept (Veichtlbauer et al. 2018), we develop our solution by first analysing the different gateway failure scenarios, before we examine which information of the network layer can be used to detect these failures. Next, we investigate how SDN can be utilised to collect the required information and to implement application and communication protocol independent gateway failover. We implement a demonstrator including all necessary hardware and software components and, finally, evaluate our approach.

Related work

The transformation of the electrical grid into a smart grid, capable of integrating a rising number of communicating nodes and providing sophisticated distributed control of electrical appliances, increases the requirements on the underlying ICT infrastructure dramatically (Kammerstetter et al. 2014). These requirements comprise scalability (due to the participating nodes), flexibility (due to rapidly changing regulatory conditions), security (as dedicated infrastructures are no longer isolated but connected to public ICT infrastructures), real-time capability (for the nature of control applications), and availability (as the grid is a critical infrastructure).

Regarding availability, regulatory bounds to outage times are given in some cases. For instance, the standard IEC 61508 (International Electrotechnical Commission (IEC) 2010) defines "Safety Integrity Levels" (SIL), which limit the timely share of outages (e.g., SIL 4 relates to an outage probability of below 10^{-8}.). This can only be reached by implementing fail-operational systems (Sommerville 2015), consisting of multiple, equally capable parallel systems (multiple redundancy). A crucial issue is the replacement of a dysfunctional system by a parallel functional system (failover), as the failover shall be performed in a minimum of time, and without negatively impacting any other running systems.

For smart grid ICT infrastructures, several approaches have been used to deploy multiple redundancy in an appropriate manner. Lopez et al. (2017) named Rapid Spanning Tree Protocol (RSTP), Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR), yet stating that they "do not scale" (Lopes et al. 2017), which is a criterion for exclusion for smart grid ICT infrastructures. Bakken et al. (2011) listed networking technologies like virtual local area networks (VLANs), IP multicast, Pragmatic General Multicast (PGM), and Multiprotocol Label Switching (MPLS); yet, still none of these technologies have been considered sufficient to fulfil the complex requirements of the smart grid domain. MPLS, for instance, lacks "facilities for combining flows across different ISPs, as would be required in a WAMS-DD" (Bakken et al. 2011), where a WAMS-DD is a "wide-area measurement system for data delivery".

Yaghmaee et al. (2013) propose to use MPLS with its traffic engineering extension MPLS-TE to provide sufficient quality of service (QoS) to several classes of smart grid traffic. MPLS-TE is likely the most common solution for traffic engineering (and network management in general) nowadays. With traffic engineering, QoS parameters like latency or loss shall be guaranteed by means of resource reservation (a.k.a. IntServ) or service prioritisation (a.k.a. DiffServ). Using service classes, Yaghmaee et al.

follow a DiffServ approach to ensure loss rates below a defined limit and thus network availability.

However, even MPLS-TE based traffic engineering comes to its limits when considering smart grid ICT requirements. As Akyildiz et al. (2014) argue, drawbacks of MPLS-TE result (among others) from the fact, that "many control protocols, ..., are built on top of the Internet protocol suite". This is especially true in the smart grid domain, where overlay architectures, gateways, and application layer routing draw the big picture. In the use case at hand, the availability of an intermediary gateway device, translating between application layer protocols IEC 61850 (at station level) and IEC 60870-5-104 (at operation level) has to be guaranteed (both protocols using an own TCP underlay to the gateway) - MPLS has no means to do that.

Another approach has been brought into discussion by Zaballos et al. (2011). They propose to use a middleware approach called "ubiquitous sensor network" (USN) architecture; however, using middleware is not a lightweight and easily deployable solution for legacy field components common in the electricity domain. Consequently, the use of software-defined networking (SDN) gained increasing importance in the smart grid domain (Kim et al. 2015). SDN offers various benefits to smart grid communication compared to MPLS networks (Sydney et al. 2013):

- Flexibility: The smart grid ICT environment is heterogeneous (Emmanuel and Rayudu 2016), which makes network management complex and susceptible to errors. SDN offers more flexibility to adopt to different communication scenarios (Cahn et al. 2013). It can be used to simplify network management and therefore simplify the extensive deployment of smart grids (Rinaldi et al. 2015).
- Programmability: Network management in the context of smart grids can not only be relieved, but also partly automated, as has been examined for instance in (Zhang et al. 2013). This makes it especially useful for industries operating in a widely distributed manner, as the electricity domain does.
- Monitorability: SDN also simplifies network monitoring because no additional monitoring hard- and software is required. The SDN controller has complete knowledge of all network devices and all network traffic (Rinaldi et al. 2015). This can be used to supervise devices and provide additional traffic management capabilities.

SDN has been utilised in several scenarios in the smart grid domain (as have been surveyed by Rehmani et al. (2019)), especially for mitigating cyber-attacks and for ensuring QoS, i.e., to meet security and performance requirements (Aydeger 2016). Security issues have for instance been discussed by Maziku et al., who researched the resilience of IEC 61580 traffic against denial of service (DoS) attacks (Maziku and Shetty 2017) and performed an appropriate risk assessment (Maziku et al. 2019). For ensuring, that SDN itself remains available, multiple SDN controllers have to be utilised to avoid single-point-of-failure scenarios (Pashkov et al. 2014). Availability can even be improved by appropriate placing strategies (Müller et al. 2014).

For ensuring QoS requirements, mainly the efficiency of routing strategies has been considered to allow for a timely and dependable delivery of control information for smart grid applications. Zhao et al. (2016) used QoS measurements for issuing re-routing whenever required, showing that this approach could outperform traditional link state (Dijkstra) routing due to SDN's superior dynamics. With SDN, failover is neither bound

to regularly performed routing updates, nor to a routing protocol's preferred metrics. Besides routing in the Internet, this approach can also be used in special environments (such as field infrastructures), which is an important feature for the smart grid. Alishahi et al. (2018) used SDN for virtualising the Routing Protocol for Low Power and Lossy Networks (RPL), proving efficiency gains.

Cokic and Seskar (2019) showed the applicability of SDN based network management to several smart grid scenarios like Automatic Generation Control (AGC) or Volt/Var Optimisation (VVO). Al Rubaye et al. (2019) applied SDN onto Industrial Internet of Things (IIoT) scenarios for collecting real-time measurement data to gain situational awareness in a local environment, and to allow for a fast recovery in case of outages.

SDN was also used as technology to provide redundancy to smart grid ICT infrastructures, e.g., by Dorsch et al. (2017). In Dorsch et al. (2016) they researched failover performance for smart grid communication networks. They showed an approach of combining fast failover mechanisms (which provide a very fast rerouting, but potentially non-optimal routes) with controller based failover (which provides better optimisation power, but slower failover); however, they concentrated on link failures, as the fast failover approach requires failure detection at the data plane (SDN switches). Unfortunately, this approach is not working for device failures (as are considered in our research work), since the switches are per definition not able to detect missing functionality at the application layer.

For device redundancy, laborious control plane calculations are thus inevitable, which limits the possibilities regarding failover times (Aydeger et al. 2015). Nevertheless, if failover times in the range of seconds are acceptable, SDN allows to easily configure multiple redundancy at station level, without affecting the SCADA system. Herewith, the generation of systems, which tolerate a pre-defined number of faults, is possible with SDN, hence enabling the intended fail-operational systems. This approach allows us to use SDN for providing a failure management system for legacy components, which extends the state of the art by applying comparable approaches to running legacy environments.

Software-defined networking for failover functionality

In this section, we present our motivation to employ SDN for the implementation of our virtualised redundancy solution for power equipment controllers. For this purpose, we precisely define typical failure scenarios, specify the required network functionality and show how SDN can be used to achieve the desired result.

The fault tree depicted in Fig. 2 shows the events that can lead to a permanently unreachable gateway device. First, a disconnection on the network layer can occur either due to a gateway failure or a network failure. Furthermore, gateway failures can occur either due to an internal software failure or an internal hardware failure, which also includes the loss of the power supply.

In general, cases of a failed gateway - either due to power loss, network disconnection, or software failure - require a switch-over to another gateway. That is, of course, if the failed gateway is the currently active instance - denoted as primary gateway. If a backup gateway fails, switching must not take place.

In order to properly replace the SCADA-based failover, our envisioned network infrastructure based solution needs to provide following functionalities:

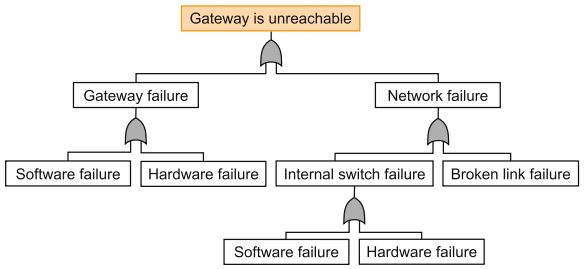

Fig. 2 This fault tree shows link disconnection errors, which need to be handled by the gateway redundancy system

- Network state monitoring: This refers to two aspects. First, the capability of continuously monitoring the network for link state changes in order to detect a physical network disconnection of a gateway at an early stage. Second, network traffic monitoring is required in order to detect non-responding gateways caused by software failures.
- Autonomous network reconfiguration: This denotes the required functionality to reconfigure forwarding paths within the network without manual intervention. As a means of gateway failover, network reconfiguration establishes a bidirectional end-to-end path between the backup gateway device and the SCADA system.
- End-to-end protocol handling: Depending on the network protocol stack employed by the SCADA system, it might not be sufficient to only reconfigure the forwarding behaviour in the network. In order to establish a valid data transmission between the SCADA system and the backup device with minimal effect on the SCADA system, additional actions such as on-the-fly network packet modifications may be required. These refer to changing network addresses in order to make packets receivable by the new destination, or pro-actively triggering the re-initialisation of an end-to-end connection.

The concepts of SDN for network management and network programming provides a promising basis for the realisation of these functional aspects. The main concept behind SDN is the separation of a network's control and data plane (Networking Foundation 2012). The control plane consolidates the network control logic at the (virtually) centralised SDN controller, while the data plane, which is represented by the network switches, is exclusively responsible for data forwarding. The controller is responsible for monitoring the network's state and changing the network's behaviour and forwarding service according to the current requirements. For these tasks, SDN makes use of a standardised interface - the southbound interface (SBI). OpenFlow[1] is a widely used representative of a vendor-independent SBI.

The network state consists of topology information including detailed information of network nodes and existing links and flows in the network, as well as traffic information such as forwarding rules at switches and bandwidth demands of flows. With the acquisition of this information, an always up-to-date and global view of the network can be created, which is indispensable for the required network state monitoring capabilities.

[1] OpenFlow Switch Specification, version 1.5.1

The separation of the control and forwarding plane also reduces the complexity of traditional network switches to pattern matchers with the task to forward and modify incoming packets depending on their content, based on forwarding rules received from the SDN controller. For instance, OpenFlow compatible switches allow matching and modifying arbitrary fields of Ethernet-based network packets. This simplification of forwarding devices together with standardised configuration and monitoring finally enables the implementation of the autonomous network control and also end-to-end protocol handling.

Another important building block provided by SDN is the possibility to implement custom network logic in form of so-called SDN apps. The information maintained by the control plane can be accessed by external applications via a controller-specific API denoted as northbound interface (NBI). SDN apps can enable a network to react in a specified way if certain situations or events occur. Possible reactions include redirection of network flows, modifications of network packets or even the injection of network packets created at the control plane.

However, the centralisation of network control logic at the SDN controller raises questions regarding the reliability and robustness:

The first concern lies in the single point of failure that the controller potentially represents. However, in real world environments, SDN controllers are usually operated as a cluster in order to increase availability. Even if the entire cluster fails, an OpenFlow compatible switch either retains the current state of the data plane or operates as legacy switch depending in implementation and configuration. Even though, the gateway redundancy is lost. Furthermore, a design goal of the envisioned failover functionality is to withstand re-initialisation after total failure of the controller infrastructure without interruption.

The second concern lies in the availability of the management network required for connection of data plane switches and the SDN controller. Similar to controller failures, this could lead to a loss of gateway redundancy. However, in critical infrastructure communication, it is common to operate redundant networks such as a ring-of-rings topology, which alleviates this risk. Complementary to a dedicated management network, SDN also allows the usage of the operational network for management tasks using so-called in-band control.

Failover implementation using ONOS and OpenFlow

This section describes the implementation details of our failover solution. First, the demonstrator used for implementation and evaluation ("Demonstrator setup" section) is introduced, followed by a description of the communication behaviour of the involved components. Finally, the implemented failover operation is shown in detail.

Demonstrator setup

Our demonstrator consists of IEDs with emulation capabilities for EPS equipment as well as a communication infrastructure. The setup is depicted in Fig. 3.

On the lowest layer of the setup, two IEDs emulate the sensing and controlling functions of the redundant RTUs. Each IED also contains the state emulation of a controlled electrical breaker (which is not physically present in the demonstrator).

A breaker can be in one of the states 'open', 'closed', 'actuating', or 'error'. The state 'open' corresponds to the situation when the electrical contactor of the breaker is in the end

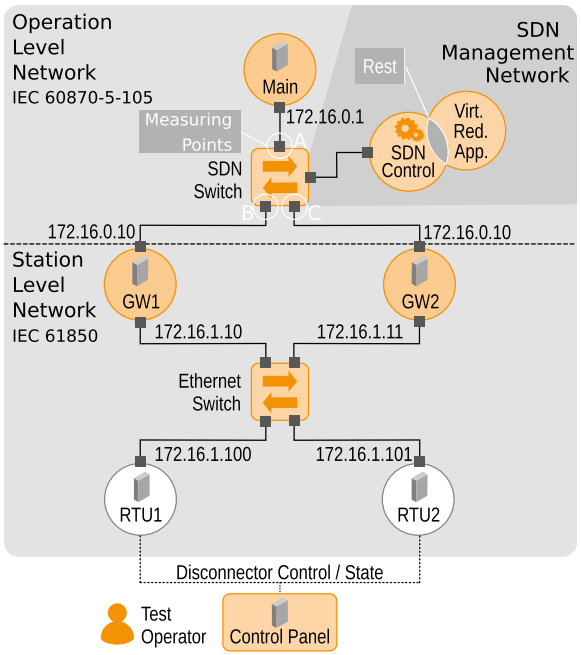

Fig. 3 In difference to real existing environments, the demonstrator is equipped with single switches instead of complete networks at operation and station level

position, where the power line is disconnected and the power flow is interrupted. The state 'closed' corresponds to the situation in which the electrical contactor is in the contrary end position, where the regarding power lines are connected and power flow is not interrupted. The state 'actuating' refers to the situation when the breaker mechanics are moving whereas the exact position of the contactor is unknown. The 'error' state refers to a situation in which the breaker is either non-operational or the state cannot be distinguished properly due to an hardware of software defect.

On the next layer, the setup consists of two IEDs emulating gateway devices. A 100 MBit Ethernet switch is used for the interconnection of lower layer RTUs and gateways. For information exchange, a commonly used and standardised IEC 61850 protocol stack is used.

On the top layer, the main component emulates the functionality of a SCADA system, which collects and processes information received from RTUs via the gateway devices. In the given setup, the main component is connected to the gateways via an OpenFlow-enabled SDN switch. On the application layer, IEC 60870-5-104 is used. In contrast to the state of the art, the SCADA application does not need information about the redundant gateways. It only knows about a single IPv4 address (172.16.0.10), which is identical for both gateway devices.

For controlling and monitoring by a human test operator, the setup is equipped with a control panel consisting of six DIP switches and six LEDs. The switches initiate the emulation of several commands such as opening the disconnector. To monitor the activity, LEDs are installed on the panel, indicating the state of the disconnector and presence of errors. For later evaluation, a measurement system has been attached which allows tracing

of network traffic at the measuring points A, B, and C (based on port mirroring and a PC with Wireshark installed).

Outsourcing of the failover procedure requires additional components. Besides the above mentioned OpenFlow enabled SDN Switch, an SDN controller and a failover management app, denoted as Virtual Redundancy App (VRApp), are used. As SDN controller, we use the Open Network Operating System (ONOS) in version 2.0. Failover management by the VRApp includes monitoring the availability of the gateways and according reconfiguration of the network. The VRApp is implemented as a Java application accessing ONOS's REST interface for requesting required information from other control plane apps, and allows initiation of flow control activities via the controller. Besides the VRApp, only four ONOS apps are enabled: Default Drivers (loaded automatically), Host Location Provider (collecting MAC address information required by the VRApp), OpenFlow Base Provider (needed for OpenFlow functionality), and Optical Network Model (required by OpenFlow Base Provider).

Communication behaviour of IEDs

As the envisioned SDN solution needs to cover both operational and failover behaviour, we analysed the communication behaviour of IEDs during their activation phase after switch-on and the operational phase, when the network is up and running. In the activation phase, each IED runs a startup sequence including self-tests and initialisation processes. Once the IED's Ethernet interface comes up, it sends a single gratuitous ARP (G-ARP) message. A G-ARP message is basically an ARP reply without prior ARP request, which is used to announce the IED's MAC and IP addresses and detect IP address collisions (Stevens 1993).

In traditional setups, the Main component continues with sending ARP requests in order to determine the MAC address of the gateway device to be used. The IP address required for sending the ARP Request is either preconfigured in the Main device (single gateway approach) or selected by the voter component (redundant gateway approach). As soon as Main receives a valid ARP reply onto the request, it initiates a TCP connection to a gateway device (three-way-handshake). In every case, Main continuously sends out ARP requests, as long as no TCP connection to a gateway has been established. Even in the case that a TCP connection breaks, Main starts sending out ARP requests at the end of a connection timeout. This timeout is configured in the Main component and usually preset to 10 s.

After the TCP session has been successfully established, the Main component and the gateway initiate an IEC 60870-5-104 (abbr. 104) session for data exchange. 104 sessions maintain sequence numbers for exchanged messages additionally to the sequence numbers of the TCP session (probably for compatibility to other transport layer protocols). Furthermore, in real-world deployments, 104 messages are end-to-end encrypted via TLS.

In our redundancy virtualisation layer, we hide the available gateway devices behind a single IP address and the responsibility to connect the Main to one of the gateways is outsourced to the network infrastructure. This means in particular, that the envisioned SDN solution needs to actively control how ARP packets and 104/TCP are forwarded between Main and the redundant gateway components. This requires awareness of the availability of gateway devices and the Main component in the SDN control plane. The

MAC addresses of the devices are obviously suitable for this purpose for two reasons: They enable unique device identification and can be used directly to define forwarding rules in OpenFlow-based SDNs.

In order assist the bootstrapping process, our SDN-based solution needs to support network address resolution regardless of the starting order of the involved components (Main, gateways, network, SDN controller including SDN apps). Furthermore, after bootstrapping, it must be ensured that the 104/TCP packets are forwarded within the network according to the relation of Main and the gateways. Proper bootstrapping and operation is done by configuring the SDN with the following set of forwarding rules and priorities:

(A) ARP messages from all hosts are forwarded to the SDN controller to inform the control plane about their respective MAC addresses (flow created by the ONOS App "Host Location Provider" with priority 40000).
(B) ARP requests of the Main device are forwarded to the controller and all available gateway devices (priority 40500, partially overruling the above flow).
(C) Traffic from Main and from both gateways is forwarded to the controller (fallback to enable the ONOS App "Host Location Provider" to learn MAC addresses even if no ARP messages are sent, priority 20000).
(D) Once the MAC addresses of the Main Device and a Gateway have been ascertained, packets are forwarded between Main and the currently active gateway (priority 41000). This means in particular, that ARP replies of the gateway are forwarded to Main (but not to other gateways) and, thus, allows the initiation of the 104/TCP connection.

The first rule is installed by the ONOS App "Host Location Provider" (using default priority 40000) and allows learning of all available MAC addresses in the system. The second and third rule are installed on indication of the VRApp automatically after activation of the VRApp. The VRApp adds the first flow rule in the list with default priority 40000. The second rule is installed with slightly higher priority 40500 in order to overrule the Host Location Provider and allow ARP requests sent by Main to reach the gateways. The third rule is used as backup in case an earlier TCP session has not yet timed out and thus Main and a previously active gateway are not sending ARP messages but TCP retransmissions.

The first three rules fill a global MAC table which is maintained by the Host Location Provider app. The information of this table is used by the VRApp in order to check availability of IEDs. As soon as Main and at least one gateway is available, the VRApp indicates the controller to install the fourth rule set stated above. This way, the ARP requests of the active gateway are forwarded to Main and the 104/TCP session can be initiated. If multiple gateways are known, it is preset in the VRApp by convention that the gateway connected to the lowest port number of the SDN switch is chosen as active gateway.

Regarding the influences of the device activation orders on proper system bootstrapping, it has to be ensured that in any case the VRapp is aware of the MAC addresses of the gateways and Main. When starting from a clean system, our investigations have shown that the whole set of different activation orders can be broken down into two bootstrapping classes: One in which the G-ARP messages (sent out by devices during activation) are used for learning device availability and configuring forwarding paths ('Boot-G'), and the other class, which uses ARP messages emitted by Main after activation ('Boot-A'). In cases where the SDN controller

and all required SDN apps are up and running before all of the IEDs, G-ARP is used.

In bootstrapping situations where the switch is up and running but ONOS and the apps are not, the OpenFlow switch has no forwarding rules installed and will discard every incoming packet by default. However, after the controller becomes available, the forwarding rules will be installed as stated above and, finally, the ARPs sent out each 500ms by Main can be utilised for bootstrapping the system.

Failover activities

As explained above, we focus on gateway failover caused by connection loss. This could either mean a disruption of the network connection on the physical layer ('Fail-NW'), or a unresponsive gateway due to a software failure at the gateway ('Fail-GW'). In either case, the failed gateway needs to be replaced by another. For replacement, the MAC addresses of the substitute devices are needed. As explained above, these addresses become known either due to G-ARP or ARP, but it is the responsibility of the control plane to keep that information ready in order to relieve the SCADA component from gateway failover procedures.

Two cases can be distinguished for gateway failover during bootstrapping: The gateway failed before it announced its MAC address using G-ARP, or the gateway loses its connection after sending G-ARPs. In the first case, no switch-over is necessary, as connection is immediately established to another available gateway (considering the port number convention, if necessary). The failed gateway, however, remains currently unknown. This can be remedied, e.g., by implementing a timeout and generating a warning message if a gateway fails to correctly initialise. The second case does not differ from situations, when a gateway device fails at some time after bootstrapping. Hence, a switch-over is required. In "Evaluation" section, an evaluation of different startup orders is examined.

Regardless of the failure type, during the operational phase of the network, the switch-over procedure remains the same. The only difference lies in the slightly longer time required for detecting a non-responding gateway ('Fail-GW'), which requires to examine the communication behaviour of the gateways. In both cases, the failover procedure reconfigures the network to connect the backup gateway to the Main component. However, although the backup gateway shares the same IP address, a seamless switch-over to the backup gateway is hindered by the following reasons:

- The second gateway receives packets addressed to another MAC address and will discard the packet,
- the TCP packets belong to an unknown session, or finally
- a corresponding 104 session does not exist or is encrypted.

Thus, in order to perform a proper switch over, besides rerouting the traffic, either a MAC address manipulation or notification of the Main component regarding the new MAC address is necessary. In addition to this, it is necessary to enforce the reinitialisation of the 104/TCP session between the Main component and the gateway device.

The VRApp utilises two procedures to detect failed gateways:

- In order to detect gateways disconnected at the physical network layer, the VRApp monitors the status of the switch port that is associated with a given device.

- For the sake of detecting non-responding gateways, the number of transmitted bytes between the main component and the gateway is examined. The communication analysis revealed that for each data packet sent from the Main component towards the gateway, at least one responding packet is sent by the gateway. If it becomes evident that the gateway does not send the expected amount of data for a predetermined timeout period, it is assumed that the gateway is in a non-responding state. This timeout is the reason, why the failover procedure for non-responding gateways takes longer compared to the network disconnection.

After a gateway failure has been detected, the VRApp performs the switch-over. In the first step, the VRApp instructs ONOS to redirect the data flows from Main towards the backup gateway. This includes the modification of the destination MAC addresses in order to make packets sent by Main receivable by the backup gateway, and vice versa.

In order to enforce the 104/TCP session re-initialisation between Main and the backup gateway, an automatism of the IEDs can be leveraged. The 104/TCP packets that now arrive at the backup gateway cannot be processed due to the mentioned session mismatches. Consequently, the backup gateway will send a TCP session reset to the address of Main. Due to network reconfiguration, the necessary data packets are correctly forwarded between Main and the backup gateway. The default behaviour of Main is to always reinitialise the 104 session after starting a new TCP session.

A special case that should additionally be considered is the hardware replacement of a faulty gateway device. When switching on the new device, it sends the G-ARP messages during its activation phase. These messages are forwarded to the control plane by the SDN Switch, and the global MAC table is updated accordingly. In the following, the VRApp is informed about the change but will not initiate a switch-over as this would unnecessarily interrupt the currently active connection between Main and the gateway.

SDN availability

Finally, besides the discussed gateway failures, the SDN infrastructure itself could fail as well. While this would not have a direct impact on the network (as the flow configuration is not affected by the controller failure), it would lead to a transient loss of failover capability and MAC address information on the SDN controller side. Thus, in real deployments, the SDN controller is running as redundant cluster. Should an outage still occur, upon re-connection, the VRApp queries the SDN controller for the current flow configuration and device status in order to recover the current relations between the Main component and the gateways so that unnecessary flow alterations can be avoided.

Upon re-connection, essentially two possible scenarios exist: In the first case, it may merely have been a temporary connection loss between the SDN Controller and the VRApp (caused e.g. by downtime caused by planned maintenance). This will be detected as such by the VRApp and no action will be taken. That is, of course, unless the active gateway has failed in the meantime, in which case a failover will immediately be triggered.

In the second case, the SDN controller may have been newly initialised and may thus not have stored any flows, leading to a short loss of SCADA/gateway connectivity, as the flows stored in the switches will automatically be deleted by the new SDN Controller. The required course of actions regarding the setting of flows for this case is basically identical to bootstrapping, which will also be detected as such by the VRApp. The only notable

difference is that in this case, the VRApp will activate the gateway whose MAC address becomes available first (rather than always gateway 1).

Evaluation

We evaluated our solution regarding correct behaviour during arbitrary startup orders, and in failover situations. Finally, we evaluated our approach using three gateways.

Evaluation of arbitrary boot orders

Regarding bootstrap evaluation we analysed the two fundamentally distinct cases: 'Boot-G' and 'Boot-A', as defined in "Communication behaviour of IEDs" section. In both cases, the components Main, GW1, and GW2 are each connected to a dedicated port of the SDN switch. The traffic at each of these ports was captured using Wireshark. In ex post analyses of the traffic captures, we examined which components exchange messages (ARP or 104/TCP) with each other.

Boot-G

For the 'Boot-G' scenario, we assume an order where the ONOS including all apps has been up and running before the links of the SDN switch are getting active. Regardless whether the link comes up because the IED is switched on after the SDN switch or vice versa, the process that the link gets active triggers the IEDs to send out a G-ARP message. In this exemplary case, we assume w.l.o.g. that GW1 to be powered on before GW2.

The startup order as well as the observed communication behaviour are depicted in the sequence diagram shown in Fig. 4. Switching-on of the involved components is done manually by the test operator without precise time recording using the power switches of the IEDs. This happens after the SDN infrastructure is up and running. Both gateways are activated in quick succession. By activating the two gateways within a period of some seconds we avoid that the algorithm assumes GW2 has failed and switches into the failover behaviour. In pretests, a activation period of around 3 seconds (roughly estimated by the test operator) has shown to be suitable to achieve the intended behaviour of the system.

First, the SDN controller installs the ARP forwarding rules as indicated by the involved SDN apps. After activation, the Main component sends the Gratuitous ARP message (G-ARP), which subsequently arrives at the SDN switch and is forwarded to ONOS. This finally informs the VRApp about the availability of the Main component. Now, the system is in a stable state waiting for MAC announcements of gateways.

After manual activation of GW1 (at an arbitrary point in time after registration of Main), it also emits a G-ARP message as intended, which is also forwarded to the SDN controller. As soon as the VRApp is informed about the activation of GW1, it is able to select GW1 as the primary gateway device and instructs ONOS to install the respective forwarding rules. Now, ARP replies emitted by GW1 can reach Main. Finally, Main is able to initiate the 104/TCP connection towards GW1.

Boot-A

For the 'Boot-A' scenario, we assume a startup order where Main and GW1 have been manually activated in the beginning, while ONOS and the apps are activated after the test operator recognised the G-ARP messages emitted by Main and GW1 using the live traffic captures from measuring points A and B as shown in Fig. 3. The SDN switch is already active at the startup time of the two IEDs. GW2 is activated after the 104/TCP session initiation between Main and GW1 is accomplished.

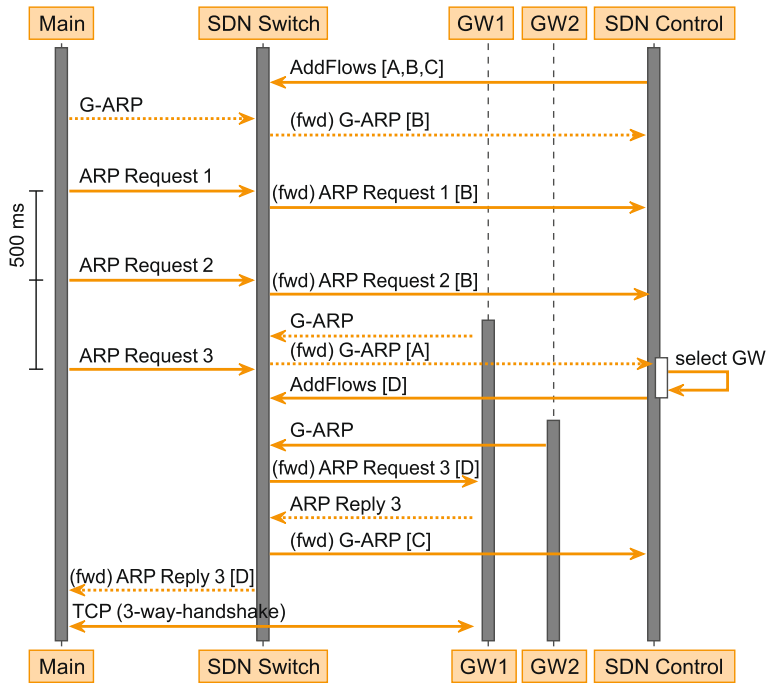

Fig. 4 The sequence diagram shows the first case using G-ARP messages for connection establishment. As intended, finally Main and GW1 establish the 104/TCP connection

The startup order and observed data exchange referring to this scenario is shown in the sequence diagram of Fig. 5. First, it becomes apparent from the figure, that the G-ARP messages emitted by Main and GW1 miss the mark, as no forwarding rule is installed. Subsequently, Main starts sending ARP requests in order to resolve the IP addresses of the gateways. Initially, the messages are also discarded by the switch until the SDN controller is activated and the SDN apps install the corresponding rules. Based on the ARP forwarding rule, the ARP reply of GW1 is forwarded to the controller, but not to Main. As soon as the ARP reply arrives at the SDN controller, it is able to select GW1 as primary gateway device and installs the M-GW1 flow accordingly. After the next ARP request is emitted by Main, GW1 again answers with a ARP reply, which this time is forwarded to Main. Subsequently, the TCP three-way-handshake and the 104 session can be initialised. Finally, the operator activates GW2 which uses its G-ARP messages to announce its availability.

Evaluation of the failover behaviour

For the evaluation of the failover behaviour, we limit the examinations to failover during the operational phase. This is valid, as a gateway failure during network startup would either lead to a situation as described in "Evaluation of arbitrary boot orders" section or to the same activities as in the operational phase.

Despite of this, we conduct two distinct scenarios referring to network disruption ('Fail-NW') and to an otherwise unresponsive gateway device ('Fail-GW'). These two scenarios cover the failover logic implemented in the VRApp. In both cases, we measure the switch-over time from occurrence of the error until the re-initialisation of the connection to the backup gateway has been completed.

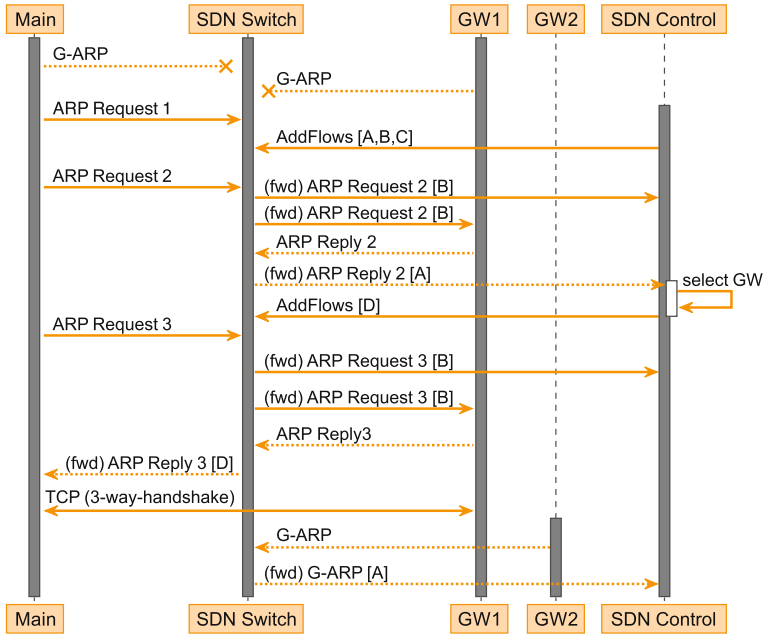

Fig. 5 The sequence diagram shows the second case using ARP messages for connection establishment. As intended, finally Main and GW1 establish the 104/TCP connection

In both cases, time measurement is done semi-manually. Utilising Wireshark measurements, the test operator performs an ex post analysis of the traffic trace captured at measurement point A, as depicted in Fig. 3. First, the test operator examines the traces for the last keep-alive packet emitted by Main before the failure occurred, which used the MAC address of the primary gateway. The failure is induced manually at an arbitrary point in time by switching off the primary gateway using its power switch. As Main sends keep-alive messages towards the gateway every second, which are usually answered by the gateway immediately, the time of the failure can be narrowed with a maximum error of one second. Next, the operator searches the PCAP traces for a special 104 packet ("STARTDT act"), sent by Main to the MAC address of the backup gateway. This packet represents the initiation of the general interrogation that Main performs by default after a valid 104/TCP connection has been established. The measurement time stamp of that second packet represents the second time stamp of the switch-over latency measurement.

For both scenarios, we benchmark our solution with a defined maximum failover target time of 5 s. This duration has been chosen because in practice, interruptions of up to 10 seconds are the norm for such devices in case of application-aware redundancy, with up to 20 seconds being considered acceptable. 5 seconds also do not constitute a relevant factor regarding system availability, which can be considered the relevant metric in this context. Furthermore, it ensures that operability is only marginally impeded when issuing commands (i.e., it is hardly noticeable for the operator of the SCADA system). Note that due to the general interrogation following a failover, commands which may have been sent to the failed gateway and therefore get lost prior to or during the failover process do not pose a serious issue, because a stable system state is maintained at all times and it will be obvious to the operator that the issued command has not been executed, so it

can subsequently be re-issued. It is worth mentioning that by convention commands are never re-issued automatically.

Furthermore, we are not expecting any significant scaling effects on experienced failover time as the actual failover happens at a single fixed location within the network. However, a software failure that is active on a redundant gateway at the same time as a detected initial failure can result in higher failover times in multi-redundancy (> 2) scenarios. The reason is that the VRApp switches to the gateway with the highest SDN port number and a working network link (i.e. no Fail-NW). The unresponsiveness of the redundant gateway (Fail-GW), however, only becomes apparent upon the attempt of a connection establishment. An additional delay of up to 5 seconds is possible.

The solid line in Fig. 6 shows the results of failover scenario 'Fail-NW' in form of an empirical cumulative density graph. The values shown are averaged over 20 measurements. The measurements reveal that in every case a switch-over time far below the envisioned 5 seconds could be achieved. The minimum latency value achieved was around 0.7 s, the maximum around 2 s, whereas the mean value was 1.1 s.

The results of scenario 'Fail-GW' are represented by the dashed line in Fig. 6. Again, 20 measurements have been executed. Also in this scenario, the switch-over time was within the allowed range, although the measured times are slightly higher due to the more elaborate failure detection mechanism. The minimum value has been around 3.6 s, the maximum value around 5 s. The mean time of the switch-over has been approximately 4.1 s. We see potential for further improvement for the second case by fine tuning of measurement intervals and system configuration.

In both scenarios, lost packets during connection establishment (TCP three-way-handshake, 104 "STARTDT act" packet) may increase the switch-over time. The increase depends on the actual value of the TCP retransmission timeout.

Evaluation of failover extensibility

For proving easy extensibility of redundancy, we connected a third gateway to the demonstrator. Without any further changes, we bootstrapped the network in the order of the 'Boot-A' scenario, yet with the third gateway activated shortly after the second.

Figure 7 shows the situation in direct connection to the 'Boot-A' scenario. After the system bootstrapping has been completed, we induced a network disruption at the currently active gateway device. After the functionality has been recovered, another link disruption

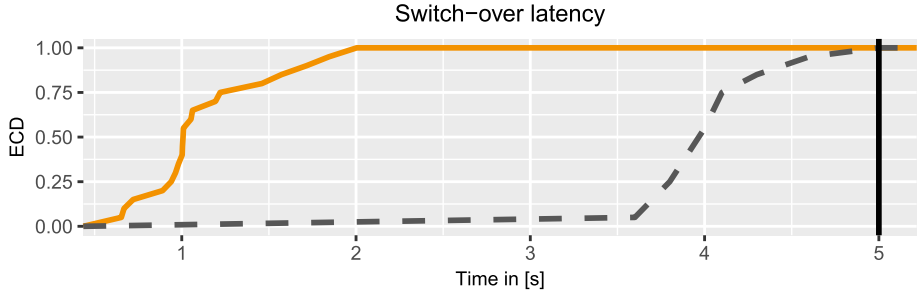

Fig. 6 The empirical cumulative density graph for the network disruption test case shows that all measured values in maximum reach 5 seconds, which is the upper limit

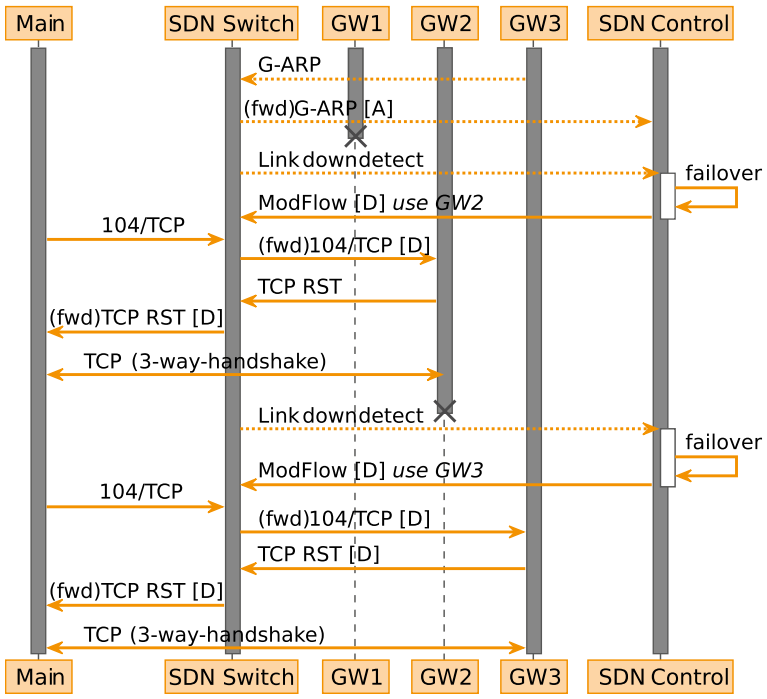

Fig. 7 The diagram shows the failover situation with 3 redundant gateways. No changes to the algorithm have been necessary

has been induced for the then active gateway, showing that the third gateway can take over functionality.

Conclusion and future work

In this paper we implemented a virtualisation layer for hiding redundancy of power system controllers from the application layer. Our solution outsources the management of redundancy and switch-over processes from the SCADA applications towards the network infrastructure. In this paper we showed, that this failover approach is capable of covering software and hardware failures of power system controllers and is able to perform a switch-over within 5 s from failure occurrence. Redundancy is added on infrastructure level without increasing the complexity of the control application, which makes redundancy engineering much easier to handle. This also allows to add redundancy to control systems using legacy communication protocols that do not have capabilities to handle redundant components.

In the current state, our demonstrator and evaluation only covers one SDN switch at the operation level. In the next step, we plan to extend our solution to also work in more complex networks such as commonly used ring-of-ring topologies. Both, pure SDN operation networks, and networks based on SDN as well as legacy technologies are in scope of our ongoing work. Furthermore, we plan to validate our solution with different protocols, in particular IEC 61850 or DNP3, in extension to the currently used IEC 60870-5-104.

Acknowledgements
Our thanks go to all of our colleagues who have provided us with expert advice and support. Our special thanks go to Daniela Gnad, who helped us with the fine drawing the most figures.

About this supplement
This article has been published as part of?*Energy Informatics*?Volume 2 Supplement 1, 2019: Proceedings of the 8th DACH+ Conference on Energy Informatics. The full contents of the supplement are available online at?https://energyinformatics.springeropen.com/articles/supplements/volume-2-supplement-1.

Authors' contributions
The work on this paper has been equally distributed between the authors with specific focuses: FvT: Evaluation and Measurements, main editor. PD: Redundancy algorithms, Measurements. AV: Related Work, Redundancy algorithms. OL: SDN Infrastructure development. UP: Redundancy algorithms. CB: Demonstrator implementation. HK: Demonstrator implementation. FK: Concepts and editing. All authors contributed to the final version of the manuscript according to their focuses. All authors read and approved the final manuscript.

Funding
The presented work is conducted in the research project VirtueGrid, which is funded by the Austrian Climate and Energy Fund (KLIEN) within the program e!MISSION (project number 858873). Publication of this supplement was funded by Austrian Federal Ministry for Transport, Innovation and Technology.

Availability of data and materials
No data and materials are available.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Advanced Networking Center, Salzburg Research Forschungsgesellschaft mbH, Salzburg, Austria. [2]Center for Secure Energy Informatics, Salzburg University of Applied Sciences, Puch/Salzburg, Austria. [3]Energy Management, Siemens AG Austria, Vienna, Austria. [4]Center for Energy, AIT Austrian Institute of Technology GmbH, Vienna, Austria.

Published: 23 September 2019

References
Al-Rubaye S, Kadhum E, Ni Q, Anpalagan A (2019) Industrial internet of things driven by sdn platform for smart grid resiliency. IEEE Internet Things J 6(1)
Alishahi M, Yaghmaee Moghaddam MH, Pourreza HR (2018) Multi-class routing protocol using virtualization and SDN-enabled architecture for smart grid. Peer-to-Peer Netw Appl 11(3):380–396
Akyildiz IF, Lee A, Wang P, Luo M, Chou W (2014) A roadmap for traffic engineering in software defined networks. Comput Netw 71:1–30. https://doi.org/10.1016/j.comnet.2014.06.002
Aydeger A (2016) Software defined networking for smart grid communications. Master's thesis
Aydeger A, Akkaya K, Uluagac AS (2015) SDN-based resilience for smart grid communications. In: 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN). IEEE, San Francisco. pp 31–33
Bakken DE, Bose A, Hauser CH, Whitehead DE, Zweigle GC (2011) Smart generation and transmission with coherent, real-time data. Proc IEEE 99(6):928–951
Cahn A, Hoyos J, Hulse M, Keller E (2013) Software-Defined Energy Communication Networks: From Substation Automation to Future Smart Grids. In: Proceedings of the 2013 IEEE International Conference on Smart Grid Communications (SmartGridComm 2013). pp 558–563. https://doi.org/10.1109/SmartGridComm.2013.6688017
Cokic M, Seskar I (2019) Software defined network management for dynamic smart grid traffic. Futur Gener Comput Syst 96:270–282
Deng Y, Lin H, Phadke AG, Shukla S, Thorp JS, Mili L (2012) Communication network modeling and simulation for Wide Area Measurement applications. In: 2012 IEEE PES Innovative Smart Grid Technologies (ISGT). IEEE, Washington, DC
Dorsch N, Kurtz F, Girke F, Wietfeld C (2016) Enhanced Fast Failover for Software-Defined Smart Grid Communication Networks. In: Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM). IEEE, Washington
Dorsch N, Kurtz F, Wietfeld C (2017) Communications in distributed smart grid control: Software-defined vs. legacy networks. In: Proceedings of the 2017 IEEE Conference on Energy Internet and Energy System Integration (EI2). IEEE, Beijing. pp 1–6
Emmanuel M, Rayudu R (2016) Communication technologies for smart grid applications: A survey. J Netw Comput Appl 74:133–148
International Electrotechnical Commission (IEC) (2010) Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. International Electrotechnical Commission, Geneva
Kammerstetter M, Langer L, Skopik F, Kupzog F, Kastner W (2014) Practical Risk Assessment Using a Cumulative Smart Grid Model. In: Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems (SMARTGREENS 2014). SciTePress, Barcelona
Kim J, Filali F, Ko Y-B (2015) Trends and Potentials of the Smart Grid Infrastructure: From ICT Sub-System to SDN-Enabled Smart Grid Architecture. Appl Sci 5(4):706–727
Lopes Y, Castro Fernandes N, Muchaluat-Saade DC, Obraczka K (2017) ARES: An Autonomic and Resilient Framework for Smart Grids. In: Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, Lisbon
Maziku H, Shetty S (2017) Software Defined Networking enabled resilience for IEC 61850-based substation communication systems. In: IEEE 2017 International Conference on Computing, Networking and Communications (ICNC). IEEE, Santa Clara. pp 690–694
Maziku H, Shetty S, Nicol DM (2019) Security risk assessment for sdn-enabled smart grids. Comput Commun 133:1–11

Müller LF, Oliveira RR, Luizelli MC, Gaspary LP, Barcellos MP (2014) Survivor: An enhanced controller placement strategy for improving SDN survivability. In: Proceedings of the 2014 IEEE Global Communications Conference. IEEE, Austin. pp 1909–1915

Networking Foundation O (2012) Software-defined networking: The new norm for networks. Open Networking Foundation, Palo Alto. https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

Pashkov V, Shalimov A, Smeliansky R (2014) Controller Failover for SDN Enterprise Networks. In: Proceedings of the 2014 International Science and Technology Conference (Modern Networking Technologies)(MoNeTeC). IEEE, Moscow. pp 1–6

Rehmani MH, Davy A, Jennings B, Assi C (2019) Software Defined Networks based Smart Grid Communication: A Comprehensive Survey. IEEE Commun Surv Tutor 2:1–34

Rinaldi S, Ferrari P, Brandao D, Sulis S (2015) Software defined networking applied to the heterogeneous infrastructure of Smart Grid. In: Proceedings of the 2015 IEEE World Conference on Factory Communication Systems (WFCS). IEEE, Palma de Mallorca

Smart Grid Coordination Group (2012) Smart Grid Reference Architecture. Technical report, CEN/Cenelec/ETSI Smart Grid Coordination Group

Sommerville I (2015) Software Engineering, 10 edn. Pearson Education, London

Stevens WR (1993) TCP/IP Illustrated (Vol. 1): The Protocols. Addison-Wesley Longman Publishing Co., Inc., Boston

Sydney A, Nutaro J, Scoglio C, Gruenbacher D, Schulz N (2013) Simulative Comparison of Multiprotocol Label Switching and OpenFlow Network Technologies for Transmission Operations. IEEE Trans Smart Grid 4(2):763–770

Veichtlbauer A, Pache U, Langthaler O, Kapoun H, Bischof C, von Tüllenburg F, Dorfinger P (2018) Enabling Application Independent Redundancy by Using Software Defined Networking. In: 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). IEEE, Moscow. pp 1–6. https://doi.org/10.1109/ICUMT.2018.8631238

Yaghmaee MH, Yousefi Z, Zabihi M, Alishahi S (2013) Quality of service guarantee in smart grid infrastructure communication using traffic classification. In: Proceedings of the 22nd International Conference on Electricity Distribution. IET, Stockholm

Zaballos A, Vallejo A, Selga JM (2011) Heterogeneous Communication Architecture for the Smart Grid. IEEE Netw 25(5):30–37

Zhao J, Hammad E, Farraj A, Kundur D (2016) Network-Aware QoS Routing for Smart Grids Using Software Defined Networks. In: Leon-Garcia A, Lenort R, Holman D, Staš D, Krutilova V, Wicher P, Cagáňová D, Špirková D, Golej J, Nguyen K (eds). Smart City 360. Springer, Cham Vol. 166. pp 384–394

Zhang J, Seet B-C, Lie T-T, Foh CH (2013) Opportunities for Software-Defined Networking in Smart Grid. In: Proceedings of the 2013 9th IEEE International Conference on Information, Communications & Signal Processing. IEEE, Tainan

Publisher's Note