A Practical Guide to Designing and Deploying OT SDN Networks

Robert Meine
Schweitzer Engineering Laboratories, Inc.

Presented at the
Power and Energy Automation Conference
Spokane, Washington
March 5–6, 2019

A Practical Guide to Designing and Deploying OT SDN Networks

Robert Meine, *Schweitzer Engineering Laboratories, Inc.*

***Abstract*—Operational technology software-defined networking (OT SDN) uses open standard SDN technology in specific ways to deliver unparalleled benefits for OT networks, such as security, determinism, traffic control, and failure recovery. OT SDN does this by using an application-focused, proactive, and predictable design based on a deny-by-default cybersecurity architecture.**

This paper briefly explores the philosophy behind OT SDN. It discusses what the benefits listed previously mean for OT networks and what OT SDN brings to OT networks. Other papers have gone into depth regarding the benefits of SDN to OT, but not with a focus on design and deployment. This paper fills that gap by describing an approach to designing and deploying an OT SDN network to incorporate these benefits. The paper discusses design processes (such as requirement gathering, topology design, and path planning) and engineering considerations (such as automation, validation, and testing).

I. Introduction

In the last decade, software-defined networking (SDN) has attracted an increasing interest. In the last five years, this interest has extended into operational technology (OT) networking, as shown by the growing body of literature examining the benefits of SDN to OT in general [1] [2] and to specific protocols [3], architectures [4], and aspects such as smart grids [5] [6] and IEC 61850 [7] [8].

Ethernet networks are now more common in OT networks such as in substations that have adopted IEC 61850 [9] and Transmission Control Protocol/Internet Protocol (TCP/IP) technologies [10] in end devices. These networks often use network equipment that follows IEEE 802.1 standards [11]; however, this type of traditional network often does not provide the necessary flexibility or security [2].

The introduction of SDN into OT networks provides many benefits, but these benefits are not automatic. As with any technology, design decisions can affect the benefits realized by the technology, and there is not yet a sufficient body of literature discussing specific OT SDN design methodologies. There is some discussion of information technology (IT) SDN networks [12] [13], but this does not translate very well to OT networks. There are many ways to deploy OT SDN, but the best solution for a particular system depends on the unique requirements of that system; the design process, therefore, should be tailored to the particular system to maximize the benefits.

This paper examines OT SDN design and engineering processes and uses specific decisions that focus on building an application-focused, proactive, deny-by-default network. This paper does not attempt to replace any current design processes or provide an in-depth example, but it is intended to provide insights into the process of SDN network deployment, based on the experience of the author. For example, the focus on automation is a result of the author's experience realizing the value of using automation to create tested, predictable networks.

This paper first provides a summary of SDN, focusing on the benefits it can bring to OT, and discusses how it can help meet the requirements of OT networks. Next, this paper discusses some of the significant decisions involved in building an OT SDN network and how a particular design influences the result. Finally, this paper briefly discusses some OT SDN deployment considerations.

II. SDN Overview

SDN is a network architecture that separates the control layer from the data layer. The data layer consists of SDN switches that forward packetized application data (flows). The control layer consists of one or more SDN controllers that instruct the data layer on how to forward those flows. They communicate through a southbound interface. Applications, such as operations, administration, and management (OAM) tools [14], drive control layer decisions (and through the control layer, the data layer) through a northbound interface. Fig. 1 gives a graphical depiction of SDN architecture.

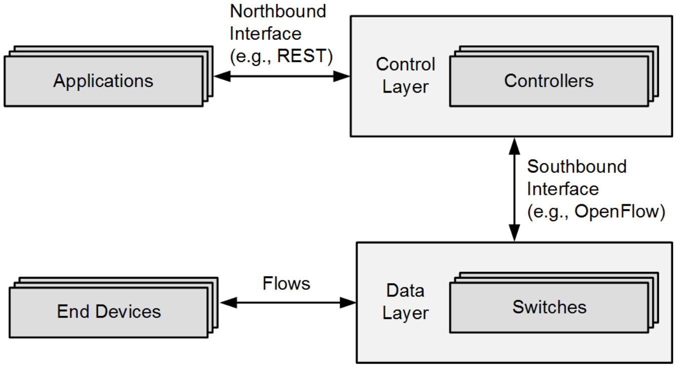
Northbound Interface (e.g., REST)
Applications
Control Layer
Controllers
Southbound Interface (e.g., OpenFlow)
End Devices
Flows
Data Layer
Switches

Fig. 1. A two-layer SDN architecture

A. A Network-Centric View

By centralizing the control layer, SDN provides a holistic view of the network. By making decisions through a network-centric view, operations such as path planning can be optimized across the network. Beyond the network-centric view, SDN controllers and switches tend to be implementation-specific, with many competing and overlapping implementation considerations, such as the southbound interface [15]. Some of these considerations are further discussed in this paper.

B. The Southbound Interface (OpenFlow)

The focus of SDN and OT SDN continues to be on the open standard protocol, OpenFlow [16], which is the southbound protocol maintained by the Open Network Foundation (ONF). OpenFlow has become the *de facto* standard for southbound communications. OpenFlow uses a match-action scheme to control the data layer: as packets enter the switch they are matched against a set of rules (i.e., match fields), and based on which rule matches the packet, the packet is either dropped or egressed from the switch, modified or unmodified (i.e., actions). A match-action pair is called a flow entry, and it is the basic unit of OpenFlow control.

Compared to traditional networking, OpenFlow provides much more advanced packet control capabilities that allow for control based on individual packet characteristics. Unlike traditional switches, OpenFlow switches support matching on and manipulating of not only Layer 2 (Ethernet) fields, but other fields such as Layer 3 (IP), Layer 4 (User Datagram Protocol [UDP]/TCP), and Address Resolution Protocol (ARP) fields. Therefore, an OpenFlow switch operates on multiple layers, not just on Layer 2.

An OpenFlow switch may or may not contain traditional switch logic. An OpenFlow switch programmed only through OpenFlow rule sets is called an OpenFlow-only switch, and an OpenFlow switch with separate non-OpenFlow logic to drive traditional control layer protocols (e.g., broadcasting ARP outside of OpenFlow rules or running Spanning Tree Protocol [STP]) is called a hybrid switch [16]. This paper only considers OpenFlow-only switches.

C. The Northbound Interface

The northbound interface is outside the OpenFlow standard and is not yet standardized [7]. It is the connection between OAM policies and tools and the control layer. The northbound interface of a controller commonly provides a Representational State Transfer (REST)-based application program interface (API) that, at a minimum, allows for monitoring and controlling packet forwarding policies [17]. Network control automation is possible through this API, either through high-level policy calls (e.g., allowing a particular protocol to pass through the network from one device to another) or through low-level calls that directly mimic OpenFlow functionality.

D. Comparison of SDN to Traditional Networking

Traditional networking devices operate according to standards (e.g., IEEE 802.1 for switches and Internet Engineering Task Force (IETF) standards such as [18] for IP routers). In SDN, Ethernet devices still follow Ethernet standards (e.g., IEEE 802.3 [19]) so the data layers are interoperable, it is control layer management that is different. In SDN, the control layer is centralized, and in a traditional network, it is distributed.

III. OT SDN Overview

OT SDN adapts SDN architecture to OT networks to improve performance and to better meet OT requirements. A growing body of literature explores OT network performance requirements and how OT SDN can help meet them [1] [7] [8] [20] [21]. This section focuses on requirements that are most relevant to design decisions and how OT SDN can better meet these requirements. This discussion drives the design decisions discussed in the next section.

A. Proactive Versus Reactive Approaches

OT network literature presents a variety of design considerations. For example, it compares proactive mode with reactive mode in programming the data layer [7] [22]. In a reactive approach, the control layer operates in reaction to data layer information such as new traffic flows. Topology, not design, influences how the control layer controls the data layer.

For example, a packet entering a switch may not match any packet-specific flow rule, so the switch sends the packet to the controller. The controller may then determine what action to take for the flow the packet represents, based on a network-centric view, and accordingly program the switch so that the flow reaches its destination. Fig. 2 shows this process.

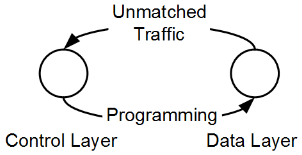

Fig. 2. A reactive model

The time between when the packet enters the switch and when the data layer is programmed to forward the flow is called the flow setup time. The flow setup time depends on several factors such as the distance from the flow controller and the number of new traffic flows entering the controller. The controller may also instruct the switch to delete the rules for the flow after a period of inactivity. If the flow appears again, the same process repeats.

Reactive modes have non-zero OpenFlow rule setup times, added system latency, and increased complexity. Reactive approaches are often applied to networks that have many flows that may not fit into the OpenFlow flow tables of a switch or that contain devices that are constantly connected to and removed from the network. In contrast, OT flows are more likely to be persistent and lower in number in comparison to IT networks, so a reactive system is less suitable when managing flows during operation in an OT network. The SDN controller must be available to prevent persistent out-of-service conditions in a reactive mode.

In a proactive approach, on the other hand, the network is pre-engineered to meet the requirements of the applications that will use it. With a proactive approach, the network does not react to traffic by sending it to the control layer for a decision. This minimizes dynamic processes and reduces flow setup time to zero for configured applications. The placement and number of the SDN controller are also less of a concern than in a reactive approach because metrics such as real-time flow setup are not critical.

A proactive approach, therefore, requires a predetermined network design that the controller or a northbound application can use to determine which devices are able to communicate with which other devices and which protocols each device can use. Fig. 3 shows one example of this process. Notice that the proactive model doesn't need flow information from the data layer in order to make control layer decisions. Instead, control layer decisions come from design. Although the switches may be sending diagnostics to the controller, this is not required for flows to operate.

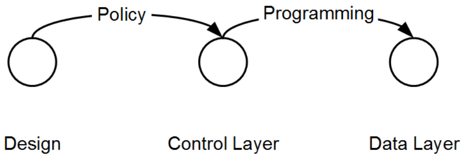

Fig. 3. A proactive model

Both proactive and reactive modes can run simultaneously, or a reactive process can be used by itself to initially learn the network. A reactive mode can manage infrequent flows such as engineering access, but these requirements can also be proactively engineered to remove any need for a reactive process in a persistent network. This paper focuses exclusively on a proactive design.

B. *Application Focused*

Network design includes a list of what devices are communicating and with what protocols, focusing on the device applications rather than just device-to-device communication.

OpenFlow can match many of the common packet fields, including ingress port and many Layer 2 through Layer 4 packet fields (e.g., Ethernet, IP, and UDP/TCP headers). This allows filtering and control of not only Ethernet header fields used traditionally in Layer 2 switches, but also IP and TCP/UDP headers used by routers and firewalls. For OT networks, OpenFlow provides the flexibility to tailor rules for a variety of purposes, such as:

- Using Layer 2 fields to control Ethernet protocols such as Generic Object-Oriented Substation Event (GOOSE) or to limit communications to between specific MAC addresses.
- Using Layer 3 IP address fields to limit communications to between two specific IP addresses.
- Using Layer 4 fields to control protocols at the TCP layer such as Multimedia Messaging Services (MMS).
- Using a combination of the previous purposes in the same rule to implement a stateless, distributed, multilayer, packet-filtering firewall [23].

The benefits of using traffic characterization to control latency during heavy link load are well-studied [24]. By using a proactive design and the per-flow matching capabilities of OpenFlow, applications can be individually classified and prioritized, even if there are no markings in the packets themselves. Each application can be assigned to a particular Class of Service (CoS) to prioritize traffic so that more critical applications are appropriately prioritized.

Removing STP provides additional benefits for some topologies because ports are not blocked. Redundant paths that would be blocked in an STP-enabled network are not blocked in an OpenFlow-only network and can be used for physical segregation of applications.

An OpenFlow switch allows control of flooding that is more flexible than a traditional IEEE 802.1 switch. For example, ARP is a common control layer protocol that is often broadcast. In a proactive design, the location of each device and IP address is known; therefore, it is not necessary to broadcast ARP to all locations. Instead, the ARP flow can be treated as a unicast flow from source to destination by using ARP match fields.

The same flexible behavior can also apply to multicast protocols such as IEC 61850 GOOSE. In traditional networking, virtual local-area network (VLAN) or media access control (MAC) filtering is used to segregate multicast traffic to prevent it from reaching more destinations than subscribed to the traffic. By using match fields instead of the user having to configure separate switch features (e.g., VLANs), the path planning itself restricts traffic destinations, which makes path planning and filtering consistent.

Unicast traffic may also be multicasted such as to the original destination and an intrusion detection system (IDS).

C. *Persistent and Predictable*

One of the major differences between OT and IT networks is that OT networks are persistent with relatively little change over time. This allows for persistent programming on switches as well as planned and scheduled changes.

Because we are only considering a proactive approach with OpenFlow switches that are not running dynamic network protocols, we can rely on OpenFlow logic instead of STP to determine primary and redundant path planning. When a network is pre-engineered, there is no need to use dynamic protocols to achieve failover because failover paths are already programmed. OpenFlow rules can be preprogrammed so applications use primary and backup paths, and because the state of the network configuration is persistent, the path of the flow in the network is predictable, even during a network fault.

A proactively engineered network also provides scalability. As switches or OpenFlow programming is added, modified, or removed, unaffected programming continues to operate. In addition, dynamic protocols are not present to cause disruptions to operating flows such as when new switches are added.

D. *Cybersecure, Deny-by-Default*

Cybersecurity is an important topic in all networks, not just OT networks. Many papers have explored the security benefits of SDN [25] [26] and of OT SDN in particular [20] [27] [28].

A proper network design defines policy so that the system takes appropriate actions in appropriate situations. A proactive design enhances cybersecurity by programming the network by design rather than relying on dynamic learning as with traditional networking.

An OpenFlow-only network operates based on a deny-by-default approach because rules must be programmed to forward traffic. Without a rule that matches and forwards traffic, the traffic is dropped. Therefore, an OpenFlow-only network operates based on whitelisting traffic, whereas traditional networks operate based on blacklisting traffic with additional optional features to provide whitelisting, such as firewalls.

Whitelisting not only prevents "bad" traffic but also prevents allowed traffic that enters the wrong port (which could be malicious) from accessing services. A proactive design uses prevention rather than remediation; that is, a proactive design attempts to prevent problems from occurring instead of responding when they do. A proactive design requires knowledge of all the communications on the network in enough detail to apply the necessary rules because communications that are not whitelisted are discarded in a deny-by-default architecture.

By using persistent, proactively engineered flow entries, the data layer is hardened against unintentional or malicious attacks from end devices. An OpenFlow-only switch does not use source MAC addresses to dynamically learn the location of an end device to control data layer behavior, which prevents ARP poisoning, and it does not use STP for path planning or loop mitigation, which prevents Bridge Protocol Data Unit (BPDU) spoofing. Some attacks based on IEEE 802.1Q VLAN [29] do not apply to OpenFlow-only switches because they do not operate on tags independent of the flow rules.

The flow control offered by OpenFlow allows missed traffic to be isolated and redirected to an IDS or to another device for deep packet inspection (DPI) [30] that does not need to be in-line with the traffic.

E. Available

Availability is the percentage of time that a service is available, and the minimizing of application data loss through path redundancy is a major part of OT communications availability management. If there is a network issue such as a link failure, a backup path must be used so the applications using that link still function.

In a proactive design, failover paths are built at the same time as the primary path, so no communications with the controller are required in order to achieve redundancy during a network event. Proactive designs that use fast failover groups can recover much faster than traditional [8] or reactive methods [31] while minimizing traffic loss, and they can reach recovery speeds of less than 100 µs in some circumstances [8]. To meet lossless protocol requirements for some IEC 61850 applications [32], mechanisms such as Parallel Redundancy Protocol (PRP) [33] can be used in an OT SDN network.

If active control layer participation is not required for the network to function, the controller can be removed. This improves long-term availability by removing a possible point of failure.

F. Performance

Besides reliability performance, OpenFlow packet matching provides additional benefits including enhanced quality of service (QoS) capabilities through physical segregation and reducing or eliminating multicast/broadcast domains. Removing broadcast traffic flooding and dynamic control layer traffic can reduce latency and jitter, in addition to providing cybersecurity benefits.

IV. Design Considerations

A good network design is tailored to the requirements of the system to maximize the desired benefits. Proactive network design directly defines allowed traffic and may also explicitly define the traffic that is not allowed. All undefined traffic is considered not allowed.

The design should contain not only what is being communicated and how but also traffic policy such as QoS and physical wiring. In a proactive, deny-by-default architecture, the network engineer uses this information to build the network. This section discusses design decisions to capture the benefits described in the previous section, starting with what must be gathered and then what OpenFlow considerations must be made.

A. Application Gathering

Before we can create a network engineered for device applications, we must first identify the applications and the protocols they use. On a simple level, each device is a set of "ins" and "outs." For example, a relay may be an MMS server that responds to MMS clients for supervisory control and data acquisition (SCADA), but it may also be a Network Time Protocol (NTP) client that requests time from an NTP server. If this information is not already available, the following questions regarding each device can help gather applications (example protocols are in parentheses):

- Does it require time services? (Precision Time Protocol [PTP], Simple Network Time Protocol [SNTP])
- Does it participate in protection? (GOOSE)
- Does it participate in SCADA? (GOOSE, MMS, Distributed Network Protocol [DNP3])
- Does it require engineering access? (File Transfer Protocol [FTP], Telnet)

Each "in" and "out" has a source and destination(s). Supporting control layer protocols such as ARP must also be considered if IPv4 unicast traffic is present. Another part of the design is gathering network device "ins" and "outs" such as management and monitoring protocols to and from the switches.

These data may also be obtained from other sources such as Wireshark captures; however, building the network from these sources is not recommended because the sources may not be derived from design and may contain unwanted traffic or not contain all the required traffic.

Documentation can take multiple forms such as spreadsheets and/or drawings: for example, data flow diagrams (DFD). Sources of data may also include derived sources such as IEC 61850 configuration files.

These inputs provide the information to develop OpenFlow programming. Documents can be used to directly program a network through consumption by a northbound application that uses the northbound interface to push a configuration to a controller.

B. *Designing to the Applications (Policy)*

After gathering device applications, we can create network policy to meet the requirements of those applications.

1) Path Planning, Topology Design, and QoS

For path planning, we select a primary path and, if designed, one or more backup paths based on metrics such as distance, port speed, and link utilization. If more than one application is running on a link, we may use QoS to prioritize applications. Each application should be classified by grouping together applications with similar requirements. Each class is then assigned a CoS value. OpenFlow provides an action for direct assignment of a CoS value to a packet, so packet markings are not necessary for prioritization because priority can be controlled proactively based on design.

Using proactive design with OpenFlow-only switches changes the benefits for some topology designs. For example, a ring may be expanded to two rings so that there are two links to each adjacent switch. Instead of relying only on CoS to control priority, higher-priority/lower-bandwidth traffic can use the one ring, while all other traffic can use the other ring, as Fig. 4 shows.

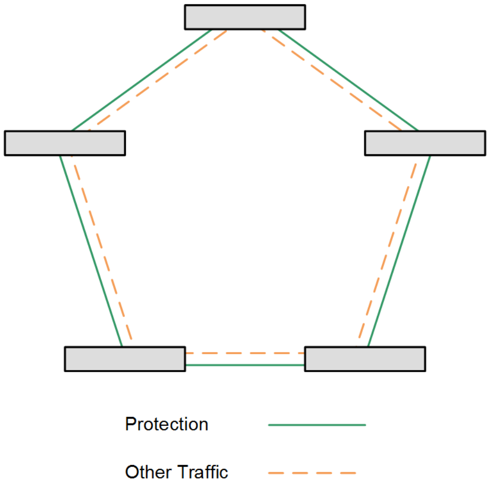

Fig. 4. Example of physical segregation in a ring

If one link is down, traffic from one path can be temporarily forwarded onto the other ring for just that one link. CoS then controls prioritization between the two classes of traffic on that link. This is possible because persistent OpenFlow rules determine the path.

For switches in close proximity, a mesh can be used, as Fig. 5 shows. If SCADA clients are isolated to one switch and high- and low-voltage protection devices are isolated to the others, each link could support specific applications that, during normal operations, do not interfere with each other. If a link fails, traffic is sent across one of the other three links to a backup switch and then forwarded to the destination switch.

These two topologies are examples of topologies that are more advantageous in an OT SDN network compared to a traditional network. Traditional topologies can also benefit from a proactive design, for example, through unicasting ARP messages and by applying flow-based CoS.

The connection from the end device to the switch can also participate in a redundancy scheme such as failover mode or PRP across two ports. In the former case, messages must be delivered to and from each port to provide immediate failover at the network level. In the latter case, the traffic from each local-area network (LAN) is forwarded to and from the port on the same LAN. Either redundancy protocol may be used in OT SDN to provide first- and last-hop redundancy.

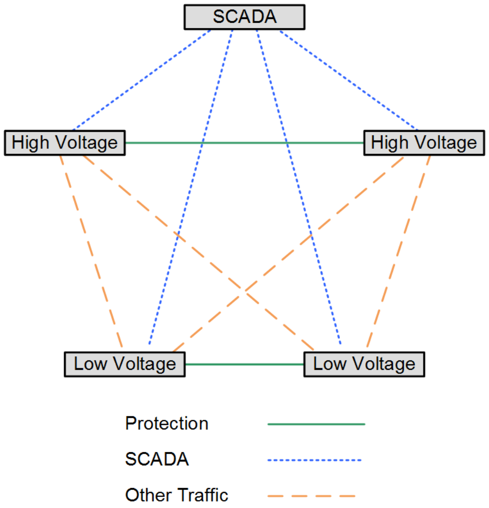

Fig. 5. Example of physical segregation in a mesh

2) SDN Controller

Another difference between traditional and SDN networks is the presence of the SDN controller. In a proactive design, the controller is optional during operation and is only required for change control or monitoring. Other monitoring techniques such as SNMP or Syslog may be used so the controller can be removed from the network to reduce the attack surface and prevent the controller from being a single point of failure or used as an attack vector into the network [25]. Even if the controller is present, however, if it only operates in monitoring mode (i.e., it does not attempt to change the network programming), attacks such as topology poisoning [34] are not effective in modifying current data layer programming.

The data layer should only be managed from the controller to maintain a consistent, network-centric view, so if the controller is removed, the network continues to operate according to its programming, regardless of dynamic changes.

The controller is the brains of the network and therefore must be cyber protected. To provide a confidential, authenticated, and secure channel, only encrypted communications should be allowed in the southbound and northbound interfaces and only using best practices, such as using strong ciphers. Allowing control layer communications that are not secure makes the control layer (and through the control layer, the data layer) vulnerable to attack.

3) Match Field and Action Selection

Match fields form the ingress control for OpenFlow programming. Match fields are required for differentiation of traffic, but they may also be used for filtering.

To match the packets of a device application, we resolve it into its possible match fields, which allows us to do the following:

- Distinguish the packets of one application from another.
- Control where the packets of an application egress

We can resolve an application such as time synchronization into its possible match fields by identifying the protocol used (e.g., NTP), then the transport mapping (unicast or multicast) of the protocol used, and then the message types (request and response) of the packets that are sent between the devices. The packet fields and their values in the message types determine the possible match fields and their values. Fig. 6 shows this process.

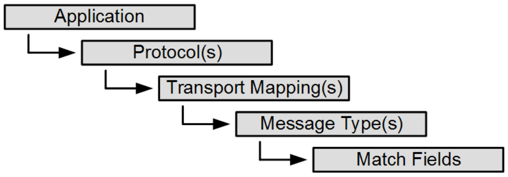

Fig. 6. Deriving match fields from applications

Matching on an ingress port limits the traffic attack surface to a single port (e.g., the port to which the device is attached in design). Restricting ingress port traffic to specific physical ports limits port access to only the services available to that port. It is recommended to use the ingress port in flow entries (as in Table I, Table II, and Table III) because the more you can control traffic at the edge of the network, the more you can control traffic on the entire network.

Including TCP or UDP ports such as NTP (UDP Port 123) further limits the services available to a device on the specified port. Dynamic processes such as selection of the TCP source port on the client are often determined at run time without user control, so that these logical ports cannot be matched; however, the destination port from client to server is often defined by standards or by settings.

Ignoring other rules on the switch, the match fields in Table I restrict communications on Port 1 to traffic from IP address 172.16.100.50 (NTP Client) to IP address 172.16.100.150 (NTP Server) for UDP destination Port 123 (NTP), which matches the NTP request. This rule does not restrict MAC addresses or the UDP source port.

MAC addresses may further restrict matching to a set of devices but would require flow entry changes for replacing devices. OpenFlow also supports matching on a range of values, so it is possible to match the vendor portion of a MAC address instead of the exact device MAC address to provide further matching restrictions.

ARP must also be added between two devices to satisfy host control layer requirements for unicast IPv4 traffic. Table II shows an example of match fields for an ARP flow entry. Restricting an ARP flow to only between the relevant two devices that must communicate reduces network utilization and increases security.

Multicast Layer 2 traffic such as IEC 61850 GOOSE is often tagged with a VLAN for traffic segregation. Table III shows one example.

A possible issue with proactive engineering is the amount of state information (e.g., in OpenFlow, the number of rules in each switch) required in the switches along with the limited number of stored match-action rules, but there are strategies that can handle this issue such as flow aggregation. Match fields can be strict (e.g., only applied to a specific flow) or broad (e.g., applied to trunk flows between switches or services between devices), depending on network policy. If strict matching is used at the edges of the network to control ingress and egress to devices, the core of the network can use broader rules to reduce the state requirements where many more flows pass through.

Actions form the egress control for OpenFlow programming. As discussed previously, failover groups provide a quick method of providing proactive failover calculations to the network. One major difference between traditional path-determining processes (e.g., STP) and OpenFlow controller processes is that in OpenFlow, path planning can be determined per flow. Because rules are made on a flow-by-flow basis, one set of flows can have redundancy or a different redundancy path than another set of flows, or a set of flows can be designed without redundancy at all.

TABLE I
MATCH FIELDS FOR AN EXAMPLE IP (NTP) FLOW ENTRY

Match Field Type	Ingress Port	EthType	IPv4 Source Address	IPv4 Destination Address	IP Protocol	UDP Destination Port
Match Field Value	1	0x800 (IPv4)	172.16.100.50	172.16.100.150	17 (UDP)	123 (NTP)

TABLE II
MATCH FIELDS FOR AN EXAMPLE ARP FLOW ENTRY

Match Field Type	Ingress Port	EthType	ARP SPA	ARP TPA
Match Field Value	1	0x806 (ARP)	172.16.100.50	172.16.100.150

TABLE III
MATCH FIELDS FOR AN EXAMPLE IEC 61850 GOOSE FLOW ENTRY

Match Field Type	Ingress Port	Ethernet Destination Address	VID	EthType
Match Field Value	1	01:0C:CD:01:00:01	901	0x88b8 (GOOSE)

Automation within the SDN controller or through applications on the northbound interface can generate the flow entries on behalf of a design but should be consistent with policy.

4) Network Monitoring

Monitoring can be divided into three parts: monitoring end devices, monitoring network devices, and monitoring network traffic. Directly monitoring devices is not only possible through SCADA, but also through common network diagnostics such as port states. Monitoring network devices can be performed through OpenFlow and its supported diagnostics if the SDN controller is online, or it can be performed through other common monitoring tools. Traffic monitoring can be performed through IDS or DPI equipment attached to the network.

C. Incorporating Automation

Automation allows for the creation of settings that, based on experience and testing, provide a predictable output with consistent inputs, which removes error-prone manual processes. SDN provides a more advanced set of automation capabilities than traditional networking through APIs that allow applications to control the control layer and, through the control layer, the data layer.

Automation may be made available by sending design information to the SDN controller (or a northbound application that communicates with the SDN controller), and then the SDN controller (or a northbound application that communicates with the SDN controller) creates the OpenFlow programming.

D. Validating the Design

The software aspect of SDN provides many capabilities for proactive design testing without the presence of physical devices. One popular testing and prototyping platform is Mininet [35]. A Mininet simulation can be set up to test whether flows operate as expected based on the programming and can include any failover testing. This is important in a deny-by-default architecture because the testing can validate that the programming operates correctly as a system. There are many other debugging, verification, and simulation tools available for SDN [36].

Using a simulation tool such as Mininet, testing can be performed before any physical devices are present and can be used to ask “what if” questions about the design. Care should be taken to consider the differences between Mininet and physical switches such as the number of flow entries per switch. The tested configuration may then be applied to set up the physical network at deployment.

Using simulation, some OAM activities can be performed before deployment, such as path verification, fault location, continuity check, connectivity verification, and resource checks (data layer verification). If the network to be deployed matches the simulation, issues can be corrected before they occur in the field, replacing traditional continuity check tools such as IP Ping and IP Traceroute [37], which may not work if these protocols are not whitelisted between the two points tested.

Validation focuses on problems in the data layer rather than just control layer configuration by validating the network at the data layer and verifying the output of the control layer rather than the inputs.

E. SDN Equipment Selection

The capabilities of an SDN network are intimately related to the capabilities of the equipment. This section summarizes the criteria for selecting OT SDN switches in the context of the requirements previously discussed. Criteria such as mean time between failures (MTBF) fall outside these requirements but should still be considered.

For SDN switches, hop speed and failover speed can vary depending on how much processing is performed in firmware versus hardware. As previously discussed, the variability in switch failover speed depends on factors such as hardware acceleration, media type, detection speeds, and underlying methods of link-loss detection.

Control layer traffic should use cybersecure communications. The SDN controller should support a proactive design capable of selecting match fields that promote a strict deny-by-default architecture.

An additional equipment feature to consider is the amount of automation available either directly in the controller or in the northbound applications. In some cases, the controller can generate the OpenFlow entries for proactive primary and redundancy path planning for device applications, and the controller can meet OT SDN requirements such as providing for OT redundancy schemes and protocols based on user-defined policy.

F. Network Documentation

In an OpenFlow-only switch operating in a completely proactive SDN network, the OpenFlow programming is the information necessary to examine the path of a packet. This includes flow entries, group entries, and port settings (if some ports are disabled). This programming information can be used to generate compliance information such as which devices have access to which devices and which protocols each device can use.

V. Deployment Considerations

Deployment of traditional and OT SDN networks are very similar. End devices and network devices are configured, tested (e.g., factory acceptance test), and then put into service. This section focuses on a few considerations specific to the deployment of OT SDN.

A. Network Programming

As in traditional networking devices, SDN switches can be programmed before placement onto the network. Depending on the match fields, some information such as MAC addresses may not be available until deployment, which would require configuration during deployment if they are required by policy to be used in flow programming. If the design is altered during deployment and new devices and applications are added, these changes can be simulated and tested first.

B. Deployment Testing

Testing ensures that for each contingency, the correct action is taken. Because a proactively designed network operates in a persistent, predictable manner based on flow programming and not based on dynamic processes, each test of a contingency produces consistent results. One contingency is a failure in the primary path of a flow. The primary path is completely determined by the programming, so the number of failure contingencies is limited to those links and switches on the programmed primary path.

Traditional networking devices operate based on dynamic learning protocols, such as dynamic MAC learning, so they do not need to be programmed based on the design to function. Deviations from an original design (such as incorrect device configurations or device placement) can therefore function or appear to function in the right circumstances (but not in others, such as during a failover event). Deviations from the design in an OT SDN system are easier to catch because communications may not be successful.

VI. Conclusion and Future Work

This paper examines specific design choices and the benefits they provide when applying SDN to OT systems. First, it reviews several requirements of OT networks and shows the benefits of SDN in meeting those requirements. It also explores application-centric, proactive policy-focused design, the benefit of validation through simulation, and how they help meet the requirements of OT SDN networks. Finally, it briefly discusses deploying an SDN network.

Future work includes a comparison of what SDN controllers and northbound applications are available and how they fit with the requirements for OT SDN. Another suggestion for future work is case studies of proactive OT SDN networks in operation and corresponding design choices. Additional research could also include investigating the relationship between proactive engineering and maintenance requirements.

VII. References

[1] N. Dorsch, F. Kurtz, and C. Wietfeld, “Communications in Distributed Smart Grid Control: Software-Defined vs. Legacy Networks,” proceedings of the 2017 IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, 2017, pp. 1–6.

[2] M. Hadley, D. Nicol, and R. Smith, “Software-Defined Networking Redefines Performance for Ethernet Control Systems,” proceedings of the Power and Energy Automation Conference, Spokane, WA, March 2017.

[3] A. Goodney, S. Kumar, A. Ravi, and Y. H. Cho, “Efficient PMU Networking With Software Defined Networks,” proceedings of the 2013 IEEE International Conference on Smart Grid Communications (SmartGridComm), Vancouver, BC, 2013, pp. 378–383.

[4] E. A. Leal and J. F. Botero, “Software Defined Power Substations: An Architecture for Network Communications and Its Control Plane,” proceedings of the 2016 8th IEEE Latin-American Conference on Communications (LATINCOM), Medellin, Colombia, 2016, pp. 1–6.

[5] N. Dorsch, F. Kurtz, S. Dalhues, L. Robitzky, U. Häger, and C. Wietfeld, “Intertwined: Software-Defined Communication Networks for Multi-Agent System-Based Smart Grid Control,” proceedings of the 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), Sydney, NSW, Australia, 2016, pp. 254–259.

[6] N. Dorsch, F. Kurtz, H. Georg, C. Hägerling, and C. Wietfeld, “Software-Defined Networking for Smart Grid Communications: Applications, Challenges and Advantages,” proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 2014, pp. 422–427.

[7] E. Molina, E. Jacob, J. Matias, N. Moreira, and A. Astarloa, “Using Software Defined Networking to Manage and Control IEC 61850-Based Systems,” *Computers & Electrical Engineering*, Vol. 43, 2014, pp. 142–154, ISSN 0045-7906.

[8] Q. Yang and R. Smith, “Improve Protection Communications Network Reliability Through Software-Defined Process Bus,” proceedings of the Grid of the Future Symposium, Reston, VA, October 2018.

[9] IEC 61850, Communications Networks and Systems in Substations.

[10] R. Braden (eds.), “Requirements for Internet Hosts – Communication Layers,” RFC 1122, October 1989.

[11] IEEE Standard 802.1, IEEE Standards for Local and Metropolitan Area Networks.

[12] P. Zanna, S. Hosseini, P. Radcliffe, and B. O’Neill, “The Challenges of Deploying a Software Defined Network,” proceedings of the 2014 Australasian Telecommunication Networks and Applications Conference (ATNAC), Melbourne, Australia, 2014, pp. 111–116.

[13] A. Lara and L. Quesada, “Priority-Based Routing in a Campus Network Using SDN,” proceedings of the 2017 IEEE 9th Latin-American Conference on Communications (LATINCOM), Guatemala City, Guatemala, 2017, pp. 1–6.

[14] E. Bellagamba, T. Mizrahi, N. Sprecher, and Y. Weingarten, “An Overview of Operations, Administration, and Maintenance (OAM) Tools,” RFC 7276, June 2014.

[15] W. Braun and M. Menth, “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices,” *Future Internet*, Vol. 6, Issue 2, 2014, pp. 202–336.

[16] Open Networking Foundation, “The OpenFlow Switch Specification, Version 1.3.4,” March 2014. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.4.pdf.

[17] N. Foster et al., “Languages for Software-Defined Networks,” *IEEE Communications Magazine*, Vol. 51, Issue 2, pp. 128–134, February 2013.

[18] F. Baker (ed.), “Requirements for IP Version 4 Routers,” RFC 1812, June 1995.

[19] IEEE Standard 802.3, IEEE Standard for Ethernet.

[20] J. O’Raw, D. M. Laverty, and D. J. Morrow, “Software Defined Networking as a Mitigation Strategy for Data Communications in Power Systems Critical Infrastructure,” proceedings of 2016 IEEE Power and Energy Society General Meeting (PESGM), Boston, MA, 2016, pp. 1–5.

[21] S. Dalhues, L. Robitzky, U. Häger, N. Dorsch, F. Kurtz, and C. Wietfeld, “Analysis of Real-Time Coordination of Distributed Power Flow Controllers Using Software-Defined Networking Communication,” proceedings of the 2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, 2018, pp. 1–5.

[22] M. P. Fernandez, “Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive,” proceedings of the 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), Barcelona, Spain, 2013, pp. 1009–1016.

[23] S. Kaur, K. Kaur, and V. Gupta, “Implementing OpenFlow Based Distributed Firewall,” proceedings of the 2016 International Conference on Information Technology (InCITe) – The Next Generation IT Summit on the Theme – Internet of Things: Connect Your Worlds, Noida, India, 2016, pp. 172–175.

[24] D. M. E. Ingram, P. Schaub, R. R. Taylor, and D. A. Campbell, “Network Interactions and Performance of a Multifunction IEC 61850 Process Bus,” *IEEE Transactions on Industrial Electronics*, Vol. 60, Issue 12, December 2012, pp. 5933–5942.

[25] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, “Security in Software Defined Networks: A Survey,” *IEEE Communications Surveys & Tutorials*, Vol. 17, Issue 4, fourth quarter 2015, pp. 2317–2346.

[26] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing Network Security Through Software Defined Networking (SDN)," proceedings of the 2016 25th International Conference on Computer Communication and Networks (ICCCN), Waikoloa, HI, 2016, pp. 1–9.

[27] C. Gray, "How SDN Can Improve Cybersecurity in OT Networks," proceedings of the 22nd Conference of the Electric Power Supply Industry, Kuala Lumpur, Malaysia, September 2018.

[28] R. Hill and R. Smith, "Purpose-Engineered, Active-Defense Cybersecurity for Industrial Control Systems," 2017. Available: https://selinc.com.

[29] IEEE Standard 802.1Q-2011, IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks.

[30] J. Dearien, "Implementing Packet Sniffing (IDS, DPI, Port Mirroring, etc.) in an SDN Network," SEL Application Guide (AG2017-18), 2017. Available: https://selinc.com.

[31] N. Dorsch, F. Kurtz, F. Girke, and C. Wietfeld, "Enhanced Fast Failover for Software-Defined Smart Grid Communication Networks," proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, 2016, pp. 1–6.

[32] IEC 61850-5, Communication Networks and Systems for Power Utility Automation – Part 5: Communication Requirements for Functions and Device Models, 2013.

[33] IEC 62439-3, Industrial Communication Networks – High Availability Automation Networks – Part 3: Parallel Redundancy Protocol (PRP) and High-Availability Seamless Redundancy (HSR), 2016.

[34] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," NDSS Symposium 2015, February 2015, pp. 8–11.

[35] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX), No. 9, 2010.

[36] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE,* Vol. 103, No. 1, January 2015, pp. 14–76.

[37] J. Postel, "Internet Control Message Protocol," RFC 792, June 1981.

VIII. BIOGRAPHY

Robert Meine is a graduate of the University of Idaho, where he received a B.S. degree in materials science and engineering, and a B.S. degree in computer science. Robert joined Schweitzer Engineering Laboratories, Inc. in 2013 and is currently an application engineer in the communications department of research and development, where he supports SDN-related products.