Applying New Technology to Improve the Performance of Teleprotection Communications

D. Dolezilek, C. Gordon, and D. Anderson
Schweitzer Engineering Laboratories, Inc.

Presented at the
13th International Conference on Developments in Power System Protection
Edinburgh, United Kingdom
March 7–10, 2016

Applying new technology to improve the performance of teleprotection communications

D. Dolezilek*, C. Gordon*, D. Anderson*

**Schweitzer Engineering Laboratories, Inc., 2350 NE Hopkins Court, Pullman, WA 99163 USA, dave_dolezilek@selinc.com*

Keywords: Ethernet, SDN, fast reconfiguration, redundancy.

Abstract

This paper explains how to use software-defined networking (SDN) for protection applications. A description of the process for building and designing teleprotection networks reveals the benefits of using SDN technology over traditional networking.

1 Introduction

Latency, reliability, and dependability are major metrics in the world of teleprotection communications. Though still served well by dedicated point-to-point serial communications links, many new interlock and teleprotection schemes exchange signals via IEC 61850 Generic Object-Oriented Substation Event (GOOSE) Ethernet messages. Due to the trend to perform all substation system Ethernet communications on a single shared network, protection signaling applications are now mixed among other Ethernet messages within complex, shared, queued-packet architectures. These communications network technologies were never meant to offer deterministic message delivery for critical systems. The performance of these applications for precise signal exchange via Ethernet packet methods is often unmeasured and usually inadequate.

Software-defined networking (SDN) is a compelling new technology that has been successfully deployed in global business applications in recent years. SDN allows Ethernet networks to be more responsive to dynamic communications network reconfiguration requirements by decoupling the processing of packet flow control algorithms from port-level data delivery instructions. This process emulates power system crosspoint switch technology with a centralized controller and distributed devices. The centralized control algorithm constantly monitors system parameters and calculates and delivers rule sets to the distributed crosspoint switches. The crosspoint switches react immediately based on pre-engineered designs without the need to communicate with one another. Similarly, once they receive a rule set from the controller, SDN Ethernet switches deterministically perform data packet decisions and delivery in microseconds. In contrast, traditional Ethernet methods are nondeterministic and take several milliseconds to make similar decisions.

Deterministic Ethernet packet delivery is required for the delivery of the command signals used to trigger remote relays to trip remote circuit breakers either directly (direct tripping) or only after the remote device enables the function (permissive tripping). Other protection schemes involve tripping prevention by the local protection device (blocking). The command signals must be received at the remote end in the shortest possible time, and interference on the communications channel must never cause unwanted operation of the protection. SDN provides much more secure and dependable Ethernet packet delivery.

IEC 62439, Industrial Communication Networks—High-Availability Automation Networks, Part 1, for automatically reconfigurable systems, references the Spanning Tree Algorithm (STA) and Rapid Spanning Tree Protocol (RSTP) to detect and isolate faults and then reconfigure Ethernet networks to mission-critical performance metrics. Part 3 references repairable systems that mask faults and rely on external methods to detect and repair failures. These duplication methods, including Parallel Redundancy Protocol (PRP) and High-Availability Seamless Redundancy (HSR), were designed for industrial systems where personnel are available to repair failed systems. This paper (an adaptation of [1]) describes the redundancy and duplication methods that are available in SDN and illustrates how these methods create lossless signal delivery (with the use of Ethernet packets) and address issues of teleprotection, interlocking, and automation applications based on IEC 61850 GOOSE messaging.

2 Timing requirements for digital message transport across an Ethernet network

In this paper, we assume a communications-assisted protection scheme, such as a transfer trip example requiring signal exchange via digital message transfer that takes no longer than 20 ms. From various international standards [2], we know that control blocking schemes require a 99.99 percent success rate and that direct control schemes require a 99.9999 percent success rate of digital message receipt, as per IEC 60834-1. IEC 61850-5 defines fast messages that meet the 3 ms transmission time as Type 1A, Performance Class P2/P3, which is further described in [2]. Failure to accomplish the 20 ms transfer time, which includes the subscribing intelligent electronic device (IED) processing the message, is defined as a delay in delivery greater than 18 ms when transferring a data signal to an IED with a 2 ms operating cycle. Therefore, the digital communications system must meet the 3 ms transmission time 99.99 percent of the time and have a delay no longer than 18 ms for the

remainder. Network reconfiguration around a path failure must be fast enough to satisfy the 18 ms maximum signal transfer during communications system failure and recovery. Subtracting the 3 ms subscribing IED packet processing time from this 18 ms maximum duration leaves 15 ms for the network to reconfigure after any failure. Should the network reconfigure within 15 ms, it can deliver the Type 1A, Performance Class P2/P3 message. This delivery takes as long as 3 ms, for a total of 18 ms. Add to this the maximum time to process the signal within protection logic of 2 ms, and we meet the maximum signal transfer time of 20 ms.

GOOSE applications actually publish a burst of several redundant GOOSE message packets after any change of state, including protection signals. The communications-assisted application is satisfied when one or more of these redundant GOOSE messages are delivered to the destination relay. Systems do not need to prevent packet loss but rather prevent signal loss. Signal loss is prevented as long as the network detects and isolates each fault and reconfigures data flow within 15 ms. This prevents signal loss by allowing one or more of the redundant GOOSE messages to be delivered. If the reconfiguration time of 15 ms cannot be satisfied with the chosen switch network, redundant networks and redundancy protocols are necessary.

Link failure detection, isolation, and reconfiguration are traditionally performed by an STA based on information received within RSTP messages published by network devices on the shared Ethernet network. When the STA reconfigures the Ethernet network failure faster than 15 ms, all or most of the redundant GOOSE packets are delivered for the satisfaction of protection signal applications. Network devices among IEDs (referred to as switches or middleboxes) with slower STAs cannot be relied on for mission-critical signaling. Some middlebox manufacturers offer proprietary solutions, but in systems including devices from multiple manufacturers, proprietary solutions are not useful because they are neither standardized nor interoperable. As a result, when middleboxes rely on slowly resolved STAs, manufacturers recommend purchasing and building two duplicate STA/RSTP Ethernet networks in the hopes that if one network fails to deliver packets, the duplicate network will not fail simultaneously. Simultaneous failure is statistically unlikely but still possible, so it is important to use network topologies that are more resilient than a simple ring network architecture [3].

3 Communications-assisted applications via digital messages

3.1 Signaling requirements for teleprotection, interlocking, and automation applications

The hard-wired exchange of protection information uses an analog circuit to provide a voltage value at the digital input contacts of the receiver to indicate the logical status of the signal from the sender. Typically, a voltage value of zero indicates a status value of zero, and the maximum voltage represents a status value of one. This method creates a constant signal value at the receiver that changes as the protection signal status changes. However, if the signal wire is cut or disconnected, the receiving device cannot distinguish between this failure and a legitimate zero analog value. With digital messages, the signal exchange is not constant. Each time a digital message is received, the signal status is confirmed or a change of status is recognized. The receiver assumes that the signal status remains unchanged during the time between messages. Also, the digital message exchange can be supervised. In this case, the receiver can detect when the communications link is lost. For example, MIRRORED BITS® communications publish digital messages every 2 ms over dedicated links, so signal confirmation of a change of state is detected within 2 ms at the receiver.

Non-change-of-state GOOSE messages typically occur once per second and act as an application heartbeat, with the time between signal confirmations at the receiver growing to once per second. A signal status change of state typically triggers an immediate GOOSE publication. The IEC 61850 standard does not specify how quickly an IED must process and act on the signal information within the GOOSE message. The application designer must understand and procure appropriately designed IEDs that streamline and prioritize processing of signal information receipt via GOOSE protocol. As with MIRRORED BITS communications, these subscriber IEDs receive and react to the change-of-state information within 2 ms, assuming the network performs correctly and delivers at least one of the GOOSE messages within the burst. To provide lossless signal transfer in the event of a network failure, IEC 61850 describes a method by which the source IED publishes several redundant change-of-state indications in a burst of redundant messages after the change of state occurs. When the failure is corrected before the burst is complete, lossless signal delivery is achieved.

Communications-assisted protection via GOOSE messaging relies on fast signal information message publication and subscription as well as fast network reconfiguration.

3.2 SDN-based redundancy methods

3.2.1 Introducing SDN

SDN essentially allows the management of networks as a single asset, giving network operators extremely granular levels of control over network functionality while simultaneously abstracting the complexity into a more traditional and functional programmatic interface [4]. The effects of the abstraction and granular control are the simplification of network operation, the ability for continuous monitoring in more detail, and the holistic centralized network control over the programming of individual middleboxes.

The fundamental shift in networking introduced by SDN is the decoupling of the systems that decide where the traffic is sent (i.e., the control plane) from the systems that perform the forwarding of the traffic in the network (i.e., the data plane). For traditional networks, packets must flow on links determined by the various distributed STAs in the

middleboxes supported by information published in RSTP packets. Middleboxes must contain additional traffic control methods, including IEEE 802.1Q virtual local-area networks (VLANs) and IEEE 802.1p Class of Service tags, to maximize reliability. In large networks, trying to match the STA-discovered path with an application-desired data path may involve changing configurations in hundreds of devices with a variety of features and configuration parameters. This complexity in management arises from the fact that each middlebox internally integrates a combination of control logic and data-forwarding logic. Fig. 1 illustrates the SDN building blocks, which provide logical separation of creating and executing the data flow rules [5].

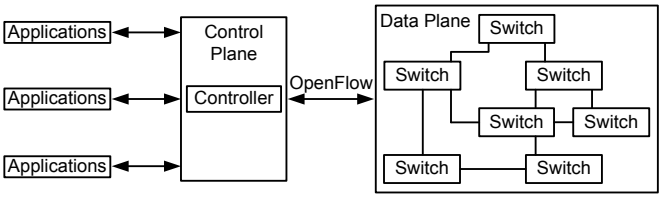

Fig. 1. SDN architecture overview.

The SDN controller calculates data flow through the entire network and sends rules to the SDN middleboxes, which are programmed simply to execute the flow rules as directed by the controller. SDN reduces or eliminates the need to use spare, inactive links to IEDs or other middleboxes. By defining the behavior of network paths, SDN allows otherwise inactive Ethernet paths to be actively used simultaneously. Therefore, all Ethernet connections to the IEDs and middleboxes can be used as designed and none are forced to hot standby mode. With SDN, the middleboxes no longer need to send RSTP messages or run STAs and can make fast and deterministic decisions about where to send packets based on logical paths through the network, as computed by the controller. Without STAs, the network can manage more than one active connection to each end device. When two cables are connected to an IED, they can function simultaneously and pass different or redundant messages on each link. As with STA, the relay will only identify a failover because of a failed connection to the primary port on the relay itself. However, using SDN, the network can be designed to deliver two different GOOSE messages from the source relay to the destination relay by using different paths in the same network. Using this method, the IED calculates message quality independently for both of the redundant GOOSE messages. This method provides redundant signals, packets, and messages and informs the protection logic if one or both of the two signal paths has failed. Also, SDN enables us to create redundant data paths in the same network.

3.2.2 Introducing OpenFlow™

SDN controls an entire network as a single operating environment and allows administrators to implement and operate the network in a way that is targeted directly at the data exchange needs of each application. SDN gives operators high-level control over individual message types by separating the control plane (or brain) that decides traffic flow from the data plane (or hardware) that pushes packets. OpenFlow is an increasingly popular protocol that acts as the interface between the SDN brain and the packet-pushing hardware. OpenFlow was developed by the Open Networking Foundation (ONF) to fulfill the need for a communications interface for SDN [6]. OpenFlow enables one or more OpenFlow controllers to define the path of packets through an Ethernet network by manipulating the packet flow tables of OpenFlow-enabled hardware. When using STA, each Ethernet middlebox is responsible for its own data flow decisions. This leads to complications during dynamic conditions because of all the simultaneous independent decision-making. SDN (when implemented with OpenFlow) solves this problem because the SDN controller precomputes all states of the network system, including those with host, link, and middlebox failures, and implementation occurs in OpenFlow-enabled middleboxes, eliminating the need for packet-pushing hardware to discover the state of the system from neighboring middleboxes.

OpenFlow provides three functions that help the controller make decisions on packets that enter an Ethernet middlebox: matches, actions, and counters (or statistics). Combinations of matches and actions form flow tables, while counters (collected by the controller via polls) detail information such as byte counts, flow table matches, and more.

- OpenFlow matches: When an Ethernet frame first enters the middlebox hardware, the OpenFlow-enabled middlebox examines the packet in search of match fields. Match fields can be physical and can include physical ingress ports and packet headers found in Open System Interconnection (OSI) Layers 2 through 4.
- OpenFlow actions: When a match is found for a particular flow table, actions can be performed. Actions can include traditional packet-handling methods, such as forwarding or dropping. OpenFlow also allows the copying of frames to multiple end ports, the application of metering or rate-limiting functions, and direct manipulation of packet headers.
- OpenFlow counters: OpenFlow-enabled middleboxes keep counters for every port, flow, flow table, and other logical and physical ports and actions performed. Examples of counter data can be the number of dropped frames, total byte count for a flow table, and so on.

OpenFlow can define the logical or physical data path for an Ethernet frame based on any matched packet field. OpenFlow can detect teleprotection messages by their Ethertype and treat them as critical traffic with unique data paths.

3.2.3 OpenFlow controller interaction

OpenFlow controllers interact with OpenFlow middleboxes in two ways. The first method of interaction is called reactive flow instantiation, where the OpenFlow middlebox forwards all frames that do not match any flow tables to the controller, which then reacts and decides how to process them. The controller updates flow entries in the middleboxes regularly as different flows cross the network. This method is inadequate for teleprotection networks because the time necessary to react to a failure is dependent on latency to and

from the controller, the processing time necessary for the controller to calculate a new network path, and the time it takes to instantiate a new flow rule into the OpenFlow middlebox. The second method of interaction between the controller and middlebox is called proactive flow instantiation, in which the OpenFlow controller has already precalculated and instantiated the flow rules that match all of the normal and failure possibilities that the middleboxes should experience. This is an ideal scenario for teleprotection communication because the OpenFlow middleboxes pass traffic much more quickly under dynamic circumstances and react to possible failure scenarios at or near line speed.

OpenFlow packets between controllers and middleboxes can be optionally encrypted with Transport Layer Security (TLS), with controller and middlebox identities verified using X.509 certificates. The enforcement of the confidentiality and integrity of OpenFlow communication represents a new shift toward stronger cybersecurity controls for control plane communication.

3.2.4 Examples of OpenFlow failure recovery

By programming an OpenFlow controller, users develop flow table logic, pre-engineer high-availability scenarios per middlebox and per link, and then update the flow tables in all OpenFlow middleboxes with this precalculated logic (proactive flow instantiation). Using a variety of methods, OpenFlow automatically calculates advanced failover scenarios that would effectively reroute traffic with only the loss of either the Ethernet frame currently being transmitted on the affected link or the frames currently in the output buffer for the particular port. With OpenFlow, the loss of a network link only affects the OpenFlow middleboxes that are directly connected to it. Link redundancy is easily performed by grouping ports together so that, on packet egress, the highest priority port presently available is used for the outgoing packet to the next hop or its final destination. By forwarding a packet to a group of ports, an OpenFlow-enabled middlebox will use only the first available port to forward the frame to its destination. If the port closest to the destination is unavailable, flow tables previously designed by a traffic engineer forward the packet to the next middlebox closest to the relay (see Fig. 2). By forwarding the packet to each port, if the first available port is not available, the next port is used immediately without the need to resend the packet.

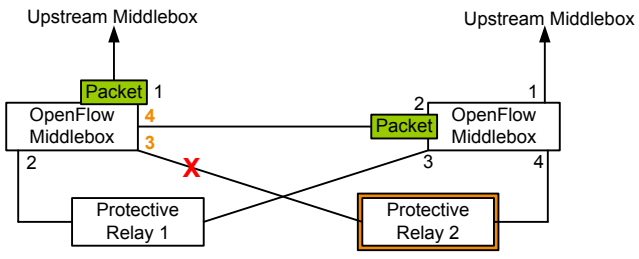

Fig. 2. OpenFlow group port failover.

In cases where ports are not grouped, logic can send a packet back out of the original port from which it ingressed (see Fig. 3). This practice is impossible with STA, PRP, or HSR but is possible with OpenFlow and redirects traffic back into a middlebox to resend it out of an alternate port. With OpenFlow(using granular enough rules), packets will always be delivered, except for scenarios in which a link fails while the packet is on the link itself or when packets are queued in the outgoing port of the middlebox hardware.

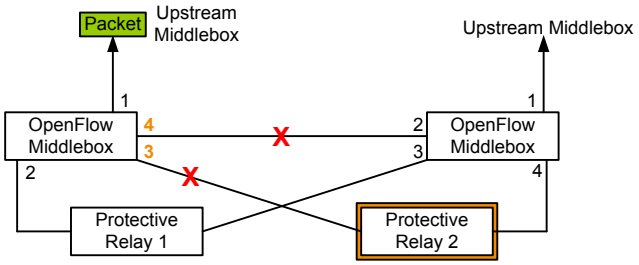

Fig. 3. OpenFlow outputs packet to original ingressing port.

OpenFlow can copy ingressing critical frames and forward out multiple ports, or it can dedicate physical links for specific flows to minimize possible queueing or latency. Because OpenFlow can operate on either a blacklisting or whitelisting premise, it can also effectively firewall all communications. OpenFlow neither consumes additional bandwidth nor disables backup links because only specific traffic flows need to be redundant, and OpenFlow hardware can use all links. OpenFlow rules can be initiated on a schedule to enable maintenance modes where OpenFlow flow tables redirect all traffic around affected portions of the network without loss of data.

4 Conclusion

To provide lossless signal transfer in the event of a network failure, IEC 61850 describes a method for an IED to send several redundant change-of-state indications in a burst via republications after a change of state occurs. If the failure is corrected before the burst is complete, lossless signal delivery is achieved. Protection application operations are also made redundant through dual primary trip devices in the same network consuming the GOOSE message or another network based on unique contact input information.

The most reliable application redundancy method is to provide two complete and separate systems, physically isolated from one another and acting as dual primary redundant applications. OpenFlow promises to be a future “best-in-class” choice for Ethernet networks within teleprotection systems because of its flexibility and because OpenFlow middleboxes make efficient use of existing links and simultaneously provide extremely granular control over passing Ethernet traffic. OpenFlow can support previously unsupported reliability scenarios, and it can even be programmed to emulate the primary benefits of protocols such as HSR, PRP, and others. Note that the number of flow rules required to support large or complex teleprotection networks can be a limiting factor. Because line speed failure recovery requires proactive flow instantiation, it is unknown whether fully redundant flow table rule sets can be efficiently reduced to fit within OpenFlow middlebox hardware

currently existing in the marketplace, which can currently support several thousand flow table entries. However, the creativity that OpenFlow inspires will enable further development that will increase the reliability of protection networks while decreasing cost and complexity.

References

[1] D. Dolezilek, C. Gordon, D. Anderson, and T. Tibbals, "Modern Ethernet Failure Recovery Methods for Teleprotection and High-Speed Automation," proceedings of the 5th International Scientific and Technical Conference, Sochi, Russia, June 2015.

[2] S. Chelluri, D. Dolezilek, J. Dearien, and A. Kalra, "Understanding and Validating Ethernet Networks for Mission-Critical Protection, Automation, and Control Applications," March 2014. Available: https://www.selinc.com.

[3] P. Franco, G. Rocha, and D. Dolezilek, "Case Study: Increasing Reliability, Dependability, and Security of Digital Signals Via Redundancy and Supervision," proceedings of the 5th International Scientific and Technical Conference, Sochi, Russia, June 2015.

[4] D. Dolezilek, C. Gordon, D. Anderson, S. McCreery, and W. Edwards, "Simplifying Teleprotection Communications With New Packet Transport Technology," proceedings of the 5th International Scientific and Technical Conference, Sochi, Russia, June 2015.

[5] R. Bobba, D. R. Borries, R. Hilburn, J. Sanders, M. Hadley, and R. Smith, "Software-Defined Networking Addresses Control System Requirements," April 2014. Available: https://www.selinc.com.

[6] Open Networking Foundation. Available: https://www.opennetworking.org.

20151117 • TP6738