Cyber-Resilience Enhancement of PMU Networks Using Software-Defined Networking

Yanfeng Qu, Gong Chen, Xin Liu, Jiaqi Yan
Department of Computer Science
Illinois Institute of Technology
Chicago, USA
{yqu9, gchen31, xliu125, jyan31}@hawk.iit.edu

Bo Chen
Energy System Division
Argonne National Laboratory
Lemont, USA
bo.chen@anl.gov

Dong Jin
Department of Computer Science
Illinois Institute of Technology
Chicago, USA
dong.jin@iit.edu

***Abstract*—Phasor measurement unit (PMU) networks are increasingly deployed to offer timely and high-precision measurement of today's highly interconnected electric power systems. To enhance the cyber-resilience of PMU networks against malicious attacks and system errors, we develop an optimization-based network management scheme based on the software-defined networking (SDN) communication infrastructure to recovery PMU network connectivity and restore power system observability. The scheme enables fast network recovery by optimizing the path generation and installation process, and moreover, compressing the SDN rules to be installed on the switches. We develop a prototype system and perform system evaluation in terms of power system observability, recovery speed, and rule compression using the IEEE 30-bus system and IEEE 118-bus system.**

***Index Terms*—Phasor Measurement Unit, Grid Resilience, Cyber-Security, Software-Defined Networking**

I. Introduction

A reliable and sustainable power grid highly relies on the successful operations of the wide-area monitoring system (WAMS). Phasor measurement unit (PMU) networks are being rapidly deployed in the WAMS to provide GPS-based time-synchronized measurements of the power system against catastrophic events, system faults, and even cyber-attacks. A PMU device is essentially a vector-measurement digital recorder (e.g., voltage, current, and phase angles) with a sampling rate between 30 to 240 Hz. The timely detection of anomaly states also depends on a reliable underlying PMU communication network, which connects multiple PMUs to the phasor data concentrators (PDC) and ultimately, the control center for exchanging measurement data and control messages.

However, PMU networks today are suffering from the growing cyber-attacks both in numbers and sophistication, which negatively affect the situational awareness in a power system as well as the subsequent contingency analysis and emergency response [1] [2]. Therefore, it is critical to make the PMU networks attack-resilient, i.e., self-healing the network connection to restore power system observability, when facing compromised devices, broken links, and faulty measurements. The conventional mechanisms based on spanning tree or distributed routing algorithms can recovery a network from link and node failures [3]. However, they take place at the data link layer and network layer, which greatly restrict the ability

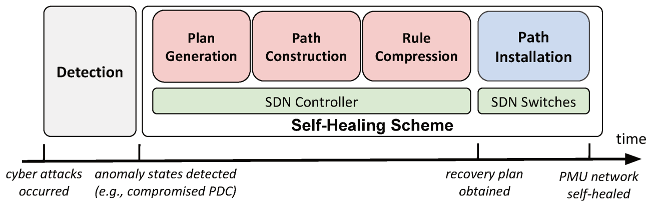


Fig. 1. Workflow of the PMU Network Self-Healing Scheme

of a self-healing scheme from reaching an optimal recovery solution by not considering characteristics specific to PMU networks and power systems.

Software-defined networking (SDN) is a promising solution to enhance PMU network resilience. Unlike conventional communication networks, SDN decouples the network control from the forwarding functions in network devices (e.g., switches and routers) and offloads the decision functions to a logically centralized SDN controller. SDN also offers direct programmability on the network devices for efficient network control and flow management. Prior works [4], [5] leverage those advantages to develop SDN-based applications to recovery PMU network connectivity and restore the power system observability against malicious cyber-attacks. In this work, we utilize the global visibility and direct programmability offered by SDN to develop an optimization-based network flow management scheme to quickly self-heal a PMU network against compromised or faulty devices. The contributions of our work are threefold compared with the existing works:

- The scheme aims to achieve fast recovery time by optimizing communication path installation;
- The scheme further reduces the recovery time with a novel flow-based rule compression module;
- We develop a working prototype system on a container-based network testbed and conduct an extensive evaluation on recovery time, system observability, and rule compression.

Figure 1 depicts the workflow of the PMU network self-healing scheme. Our scheme consists of four steps. The first three steps, plan generation, path construction, and rule compression, are executed in order on the SDN controller, and the final step, path installation, is issued by the SDN controller and executed on the SDN switches. Once a cyber-attack event is detected, we obtain the anomaly states of the PMU network as the inputs to our scheme (e.g., a list of compromised PDCs). Note that the detection of various cyber-attacks in PMU networks is important, but in this paper, we mainly focus on the self-healing scheme after a successful detection. The self-healing scheme now generates a list of disconnected yet working PMUs with the goal of reconnecting them to the appropriate PDCs and recovering the power system observability. In this step, we do not have to reconnect all the lost PMUs due to the measurement redundancy. The path generation module then creates a hop-by-hop communication path for each PMU-PDC connection subjecting to specific communication network and power system constraints. The paths are converted into per-switch SDN rules, and we then run the rule compression module to combine multiple overlapping rules into a single rule to further reduce the recovery time. Finally, the rules are installed on the SDN switches to realize the recovery plan and self-heal the PMU network.

We implement the self-healing scheme on an SDN controller and evaluate the prototype system using communication networks constructed from the IEEE 30-bus and 118-bus systems. Our scheme successfully recovers the power system observability for all test cases with fast recovery time. The total recovery time includes the model computational time and the recovery path installation time. With 10% of the PDCs compromised, the model computational time is less than 87.6 ms for the 30-bus cases and 318.5 ms for the 118-bus cases.

we also empirically demonstrate the efficiency of the rule compression module, which takes only 0.3 ms to 6 ms for computation and decreases the recovery path installation time by up to 45.5% for the 30-bus cases and up to 61.7% for the 118-bus cases. The number of rules to install is also reduced by up to 42.9% of for the 30-bus cases and up to 59.0% for the 118-bus cases.

The remainder of the paper is organized as follows. Section II introduces the related work. Section III shows an SDN-based resilient PMU network architecture with an illustrative example. Section IV describes an optimization-based self-healing scheme including plan generation, path construction, and rule compression. Section V presents the experimental setup and performance evaluation results. Section VI summarizes the paper with future works.

II. Related Work

Applying SDN technology to enhance power grid security and resilience is an emerging research topic [6]–[9]. Recent works include applications in substation automation [10], [11], substation risk assessment [12], reliability evaluation [13], quality-of-service optimization [14], fast failover mechanism [11], [12], [15], power bot detection [16], and dynamic resource allocation [17]. Researchers also construct several SDN-enabled testing platforms including a transmission-level co-simulation testbed [7] and hardware-in-the-loop testbeds integrating power system simulator, communication network emulator, and physical switches [8], [9], [18].

Existing works analyze the cyber resilience and security of PMU networks [19], [20], but do not focus on designing mitigation mechanisms by considering the constraints exclusive to PMU networks. A self-healing scheme [3] is designed using the conventional routing protocols to handle link failures but not compromised hosts. A recent work [4] focuses on using SDN to reconnect uncompromised PMU devices to restore power observability. Another work [5] realizes the PMU network self-healing through the centralized control over a distributed routing protocol. They both exploit global visibility and centralized control offered by SDN to optimize the self-healing scheme. However, minimizing recovery time is not the primary goal in their approaches. In this work, we optimize the path installation and explore the rule compression to further reduce the recovery time. This is important as many power system operations are time-critical. Each SDN rule contains information like source IP address, destination IP address, port number, and link-layer information. Researchers explore a general SDN rule compression technique with a compression ratio between 70% and 99% [21]. In the context of PMU networks, as a PMU typically connects to one PDC, we can further reduce the optimization complexity by only considering compression with destination IP addresses.

III. SDN-based PMU Network Architecture

We design an SDN-enable PMU network architecture, as shown in Figure 2, to enhance network resilience. Measurements of the underlying electrical system are captured by PMUs, and then aggregated at PDCs, and eventually transmitted to the control center through the communication layer. We integrate the SDN technology into the system by deploying a set of SDN-enabled switches with direct network programmability in the communication layer and incorporating an SDN controller into the existing control center facility. The SDN controller provides the global network visibility based on which we develop an optimization-based self-healing scheme. Upon detection of compromised/faulty devices, the scheme generates an optimal recovery plan including hop-by-hop communication paths to reconnect the selected PMUs and PDCs to restore the power system observability. The dynamic network flow management capability offered by SDN also allows us to quickly install the recovery plan to self-heal the network. Additionally, we explore the SDN rule compression technique to further reduce the recovery time.

Illustrative Example. We present a self-healing PMU network over the IEEE 14-bus system to illustrate step by step how the system can efficiently reconnect uncompromised but disconnected PMUs to restore the power system observability. Figure 3(a) depicts a communication network consisting of 6 switches, 4 PMUs, and 2 PDCs. PMU 2 and 6 send measurement data to PDC 5, and PMU 7 and 9 send measurement data to PDC 4. Assume an attacker compromises PDC 5, the PMU measurements at bus 2 and 6 cannot be transferred to this PDC, and thus the system observability is reduced at the control

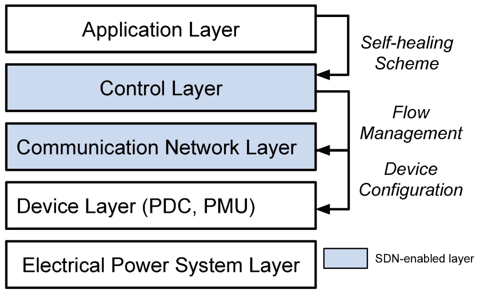


Fig. 2. An SDN-based self-healing PMU network architecture

center. Upon detection of the compromised PDC, our self-healing scheme in the SDN controller is triggered to recovery the PMU connection.

We first identify a set of disconnected PMUs and the associated buses (i.e., PDC 5 in this example). We then compute the set of PMUs to reconnect with the objective to restore the power system observability. In this example, PMU 2 and PMU 6 are selected. We then compute the destination PDC for each PMU yet to be connected (i.e., PMU 2 and PMU 6 both connect to PDC 4). By meeting all constraints, such as device capacity and communication bandwidth, we generate a communication path for each PMU-PDC pair and translate them into SDN rules for each SDN switch. To further shorten the recovery time, we develop a rule compression method to combine multiple SDN rules into a single one using appropriate wildcards. In this example, we only need one new rule $< *, 10.0.4.1, k_1 >$ to be installed on Switch 4. The rule has a wildcard in the source IP address field, PDC 4's IP in the destination IP address field, and k1 in the out-port number field. Finally, the SDN controller installs the updated rules, preferably in parallel, on the switches to realize the recovery plan as shown in Figure 3(b).

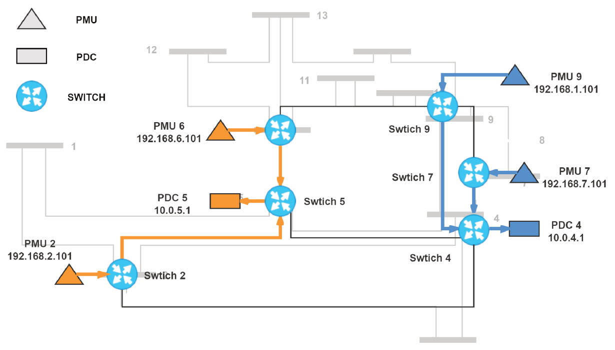
IV. System Modeling and Problem Formulation

Table I summarizes the key notions used in this section and the remainder of the paper. The power transmission network is represented by a graph $G_t = (\mathcal{B} \cup \mathcal{U}, T)$, where $\mathcal{B} = \{b_i\}$ is the set of buses, $\mathcal{U} = \{u_i\}$ is the set of PMUs, and T is a connectivity matrix concatenated vertically from a $|\mathcal{B}| \times |\mathcal{B}|$ matrix and a $|\mathcal{B}| \times |\mathcal{U}|$ matrix, and the element is defined by:

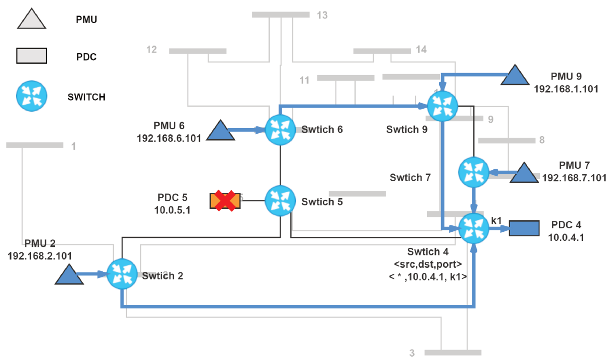
$$t_{ij} = \begin{cases} 1, & (b_i \text{ and } b_j) \text{ or } (b_i \text{ and } u_j) \text{ are connected} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The communication network is represented by another graph $G_C = (\mathcal{U} \cup \mathcal{D} \cup \mathcal{S}, L)$, where each PMU connects to a bus, $\mathcal{D} = \{d_i\}$ is the set of PDCs, and $\mathcal{S} = \{s_i\}$ is the set of OpenFlow switches. L is a connectivity matrix concatenated horizontally via the common columns $|\mathcal{S}|$ among a $|\mathcal{U}| \times |\mathcal{S}|$ matrix, a $|\mathcal{S}| \times |\mathcal{S}|$ matrix, and a $|\mathcal{D}| \times |\mathcal{S}|$ matrix.

$$l_{ij} = \begin{cases} 1, & u_i \text{ or } s_i \text{ or } d_i \text{ is connected with } s_j \\ 0, & \text{otherwise} \end{cases} \quad (2)$$



(a)



(b)

Fig. 3. (a) PMU network (b) The recovered PMU network assuming that PDC 5 was compromised

A. Recovery Plan Generation

A PMU measures the electrical waves of a bus at which it is placed and it can also estimate all the adjacent buses. We define x_{ij} to be a set of decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if PMU } u_i \text{ sends synchrophasor data to PDC } d_j \\ 0, & \text{otherwise} \end{cases}$$

where $\forall u_i \in \mathcal{U}_d$ and $\forall d_j \in \mathcal{D} \setminus \mathcal{D}_c$.

Bus b_i is observable if b_i connects to a PMU (see Equation 4) or one of its adjacent buses b_j connects a PMU (see Equation 5). A power system is observable if all the buses are observable, i.e., either directly or indirectly connected to PMUs.

$$O = \wedge_i^{|\mathcal{B}|} O_i^d \vee O_i^a \quad (3)$$

where $\forall i : |\mathcal{B}| + 1 \leq i \leq |\mathcal{B}| + |\mathcal{U}|$

$$O_i^d = \vee_j^{|\mathcal{U}|} t_{ij} \wedge (\vee_k^{|\mathcal{D}|} x_{jk}) \quad (4)$$

and $\forall i : d \leq i \leq |\mathcal{B}|$

$$O_i^a = \vee_j^{|\mathcal{B}|} t_{ij} \wedge O_j^d \quad (5)$$

A recovery plan includes a map between a set of disconnected PMUs to working PDCs in order to recover the power

TABLE I
SUMMARY OF NOTATIONS

Indices and Sets	
$\mathcal{B}$	Set of buses in the power system
$\mathcal{U}$	Set of PMUs in the network
$\mathcal{U}_d$	Set of disconnected yet uncompromised PMUs
$\mathcal{D}$	Set of PDCs in the network
$\mathcal{D}_c$	Set of compromised PDCs
$\mathcal{S}$	Set of SDN switches
G_t	Power transmission network
G_c	Communication network
L	PMU, switch and PDC connectivity matrix
T	Bus and PMU connectivity matrix
Parameters	
z	Maximum number of rules allowed on each switch
O	Power system observability
Functions	
$A : U \mapsto \mathcal{P}(U)$	Function that maps a PMU to its adjacent PMUs
$C : D \mapsto \mathbb{Z}^+$	Function that maps a PDC to its capacity to connect with PMU
$M : L \mapsto \mathbb{R}^+$	Function that maps a communication link to its bandwidth
$R : U \mapsto \mathbb{R}^+$	Function that maps a PMU to its required bandwidth
$W : R \mapsto \mathbb{Z}^+$	Function that maps a switch to its rule capacity
Decision Variables	
x_{ij}	Binary variable indication whether PMU i connects to PDC j or not
y_{ep}	Binary variable indication whether edge e belongs to path p or not

system observability. Our objective is to identify the minimum number of PMUs to reconnect to PDCs

$$\min \sum_{u_i \in \mathcal{U}_d} \sum_{d_j \in \mathcal{D} \setminus \mathcal{D}_c} x_{ij} \tag{6}$$

subjecting to the following constraints

$$\sum_{d_j \in \mathcal{D} \setminus \mathcal{D}_c} x_{ij} + x_{A(i)j} \geq 1, \, \forall u_i \in \mathcal{U}_d \tag{7}$$

$$\sum_{u_i \in \mathcal{U}_d} x_{ij} \leq C(j), \, \forall d_j \in \mathcal{D} \setminus \mathcal{D}_c \tag{8}$$

$$\sum_{d_j \in \mathcal{D} \setminus \mathcal{D}_c} x_{ij} \leq 1, \, \forall u_i \in \mathcal{U}_d \tag{9}$$

Assume that each bus is attached to one PMU, we revise Equation 3 and obtain Constraint 7, which ensures that each bus or one of its adjacent bus connects to a PMU. Constraint 8 guarantees that the reconnected PMUs do not exceed the connection capacity of a PDC. Constraint 9 ensures that each PMU transmits the measurement data to no more than one PDC.

B. Communication Path Construction

After solving Equation 6, we obtain the optimal recovery plan matrix X^*. We can view X^* as a set of path $\mathcal{P} = \{p_{ij}\}$ between PMU u_i and PDC d_j if $x^*_{ij} = 1$. These paths are still yet to be solved to generate the minimum number of SDN rules. For computing the communication paths of the recovery plan, we define another set of decision variables for $e \in L$ and $p_{ij} \in P$

$$y_{ep} = \begin{cases} 1, & \text{if edge } e \text{ belongs to the path of a recovery plan } p \\ 0, & \text{otherwise} \end{cases}$$

We also define two auxiliary functions, $I(p_{ij}, \bullet)$ and $O(p_{ij}, \bullet)$, which represent the in-degree and out-degree of device $k \in \mathcal{U} \cup \mathcal{D} \cup \mathcal{S}$ in the path p_{ij}.

$$I(p_{ij}, k) = \begin{cases} \sum\limits_{e \in \{l_{mk}=1 | \forall m \in \mathcal{U} \cup \mathcal{S}\}} y_{ep_{ij}}, & \forall k \in \mathcal{S} \\ \sum\limits_{e \in \{l_{mk}=1 | \forall m \in \mathcal{S}\}} y_{ep_{ij}}, & \forall k \in \mathcal{D} \end{cases}$$

$$O(p_{ij}, k) = \begin{cases} \sum\limits_{e \in \{l_{km}=1 | \forall m \in \mathcal{S}\}} y_{ep_{ij}}, & \forall k \in \mathcal{U} \\ \sum\limits_{e \in \{l_{km}=1 | \forall m \in \mathcal{D} \cup \mathcal{S}\}} y_{ep_{ij}}, & \forall k \in \mathcal{S} \end{cases}$$

We assume the SDN controller can distribute rules in parallel to the network and each switch sequentially installs its rules. The objective of the communication path construction formulation is to minimize the installation time of SDN rules, which can be formulated as the following min-max problem

$$\min_{\forall p_{ij} \in P} \max_{\forall s_k \in \mathcal{S}} \{I(p_{ij}, s_k)\} \tag{10}$$

subjecting to the following constraints

$$0 \leq I(p_{ij}, s_k), \, O(p_{ij}, s_k) \leq 1, \, \forall s_k \in S, \forall p_{ij} \in P \tag{11}$$

$$I(p_{ij}, d_j) = 1, \, \forall p_{ij} \in P \tag{12}$$

$$O(p_{ij}, u_i) = 1, \, \forall p_{ij} \in P \tag{13}$$

$$\sum_{d_j \in D} I(p_{ij}, d_j) = 1, \, \forall p_{ij} \in P \tag{14}$$

$$\sum_{u_i \in U} O(p_{ij}, u_i) = 1, \, \forall p_{ij} \in P \tag{15}$$

$$\sum_{\forall p_{ij} \in X} y_{ep_{ij}} \times R(u_i) \leq M(e), \forall e \in L \tag{16}$$

$$\sum_{\forall p_{ij} \in X} I(p_{ij}, s_k) \leq W(s_k), \, \forall s_k \in \mathcal{S} \tag{17}$$

We can introduce an auxiliary variable z and add the following constraint to solve the min-max problem:

$$z \geq \max_{\forall s_j \in \mathcal{S}} I(p_{ij}, j) \tag{18}$$

Constraints 11 to 15 ensure that a path exists between a PMU and a PDC. Constraint 11 ensures the path is loop-freedom. Moreover, Constraint 12 and 13 ensure that the PMU (source) and the PDC (destination) are included in the path. For each path, Constraint 14 means the out-degree of PMU is 1 and Constraint 15 means the in-degree of PDC is 1. Constraint 16 guarantees that the PMU data traffic does not exceed the link bandwidth. Finally, Constraint 17 ensures that the number of SDN rules to install do not exceed the switch capacity.

C. SDN Rule Compression

After solving Equation 10, we get the list of PMUs to reconnect with the detailed communication paths. The SDN controller now generates a set of OpenFlow rules based on these paths and then distributes them to the switches. Here, we explore an OpenFlow rule compression technique to reduce the rule installation time. Let a triplet (s,t,k) denotes a routing rule, where s, t, and k indicate the source IP, destination IP, and switch out-port of this flow. Given a default rule $r(s,t,k)$, we can use wildcards to merge those overlapping rules by source $g(s,*,k)$, by destination $g(*,t,k)$, or by both source and destination $g(*,*,k)$. The binary decision variables are $r(s,t,k)$, $g(s,*,k)$, $g(*,t,k)$, and $g(*,*,k)$. Each variable is 1 if selected and 0 otherwise. The objective function of the rule compression problem is defined as follows:

$$\min \sum_{\forall s,t,k} r(s,t,k) + g(*,t,k) + g(s,*,k) + g(*,*,k) \quad (19)$$

In this work, we do not need to consider the case of $g(s,*,k)$ (i.e,. compression by source) since each PMU reconnects to only one PDC. We also do not use $g(*,*,k)$ as it may unintentionally make the attack flows spread over the network. Therefore, we simplify the objective function and define the recovery rule compression problem as follows:

$$\min \sum_{\forall s,t,k} r(s,t,k) + g(*,t,k) \quad (20)$$

subjecting to the following constraints

$$\sum_{\forall k} g(*,t,k) \le 1 \quad (21)$$

$$r(s,t,k) + g(*,t,k) \ge 1,\ \forall (s,t,k) \quad (22)$$

$$r(s,t,k) \in \{0,1\},\ \forall (s,t,k) \quad (23)$$

$$g(*,t,k) \in \{0,1\},\ \forall (t,k) \quad (24)$$

The objective function in Equation 20 is to minimize the total number of rules. Constraint 21 ensures only one wildcard rule on each switch. Constraint 22 ensures to include every rule. The binary variables $r(s,t,k)$ and $g(*,t,k)$ are included in Constraint 23 and 24. Using rule compression optimizes the memory space utilization and saves the rule installation time. However, it may takes extra computational time. We thus propose a heuristic algorithm to compress rules as described in Algorithm 1 with time complexity of $\mathcal{O}(n)$. In order not to affect the original traffic flows, we always set the updated rules containing wildcards with a low priority.

Algorithm 1: Rule Compression

```
    Input: Set of rules $\mathcal{R}$
    Output: Compressed rules $\mathcal{R}_c$
initialize $\mathcal{R}_c$, Hashtable H $< (t,k), (s,t,k) >$
for $\forall (s,t,k) \in \mathcal{R}$ do
    if H.containsKey((t, k)) then
        $H.update((t,k),(*,t,k))$
    end
    else
        $H.insert((t,k),(s,t,k))$
    end
end
for $\forall (t,k) \in H.keys()$ do
    $(s,t,k) \leftarrow H.get((t,k))$
    add $(s,t,k)$ to $\mathcal{R}_c$
end
```

V. Evaluation

We develop a prototype system in Mininet [22], a container-based SDN emulator, and conduct extensive evaluation experiments in terms of system observability restoration, recovery time at different stages, and rule compression. The optimization model described in Section IV is developed in POX, a python-based SDN controller [23]. We use the GLPK package [24] to solve the ILP problem.

A. Experimental Setup

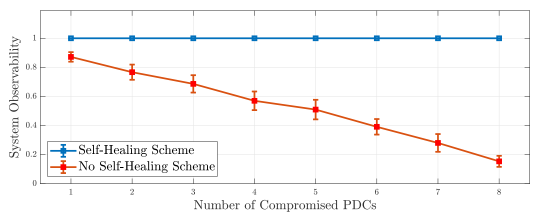
We evaluate the self-healing scheme using PMU networks constructed from the IEEE 30-bus and 118-bus systems. One PMU was attached to each bus in the transmission system. Each PMU had an adjacent PMU set, which was determined by the adjacent matrix of the transmission system. We applied the minimum set cover problem and output the least number of sets. PDCs and switches were then placed for each set. Finally, we connected switches using a ring topology so that each switch had a redundant link. We varied the number of compromised PDCs ranging from 10% to 80%, and repeated 20 runs for each experiment.

B. Performance Evaluation

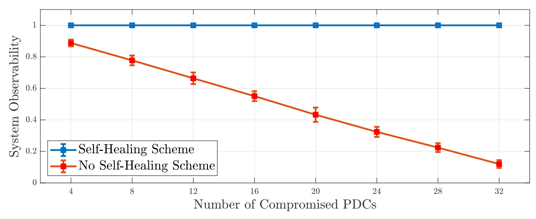
1) Plan Generation and Path Construction: We evaluate the effectiveness of the generated recovery plan in two aspects: (1) to what degree the power system observability is restored, and (2) the cost of model computational time.

Power System Observability is represented as the percentage of observable buses in a power system. Figure 4 shows the results collected on the IEEE 30-bus system and the IEEE 118-bus system. We observe that our self-healing scheme is able to recover the full power system observability for all cases. Without the self-healing scheme, the power system observability keeps decreasing as the number of compromised PDCs grows. The mean values of power system observability can drop to 15.3% in the IEEE 30-bus system and 12.0% in the IEEE 118-bus system with 80% compromised PDCs.

Model Computational Time is the time to solve the ILP model including the plan generation and path construction stages. Figure 5 shows the average computational times with standard deviations for both the IEEE 30-bus and 118-bus systems. In the 30-bus cases, the average computational time ranges from 24.4 ms to 26.5 ms to generate a recovery plan and ranges from 61.18 ms to 92.0 ms to construct the communication paths. In the 118-bus cases, the average computational time ranges from 52.5 ms to 106.1 ms to

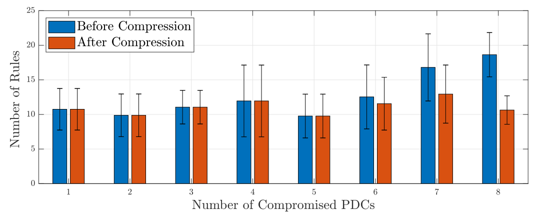


(a) IEEE 30-bus system

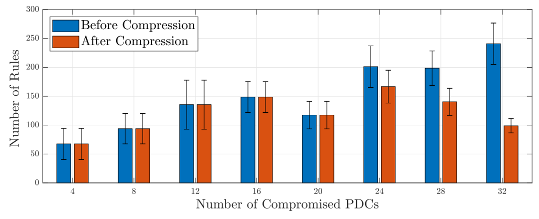


(b) IEEE 118-bus system

Fig. 4. Power system observability with/without the self-healing scheme

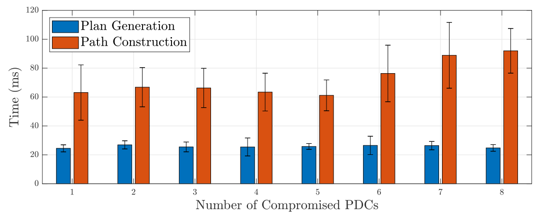


(a) IEEE 30-bus system

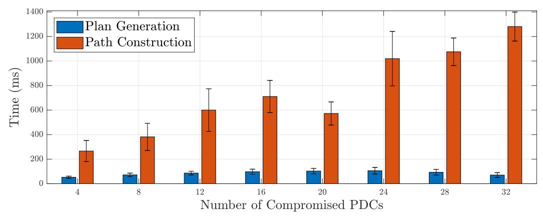


(b) IEEE 118-bus system

Fig. 6. Total numbers of rules installed on switches with/without compression



(a) IEEE 30-bus system



(b) IEEE 118-bus system

Fig. 5. Computational time for plan generation and path construction

generate a recovery plan and ranges from 266.0 ms to 1280.9 ms to construct paths. With 10% of the PDCs compromised, the total computational time is less than 87.6 ms for the 30-bus cases and 318.5 ms for the 118-bus cases. Even with 80% of the PDCs compromised, the total computational time is still less than 116.7 ms for the 30-bus cases and 1351.5 ms for the 118-bus cases.

The path construction time increases with the number of compromised PDCs because of the increasing number of communication paths to reconnect. However, the plan generation time is not greatly impacted by the number of compromised PDCs, because the redundancy of PMU placement (i.e., we only have to recover a subset of the PMUs to restore the full power system observability).

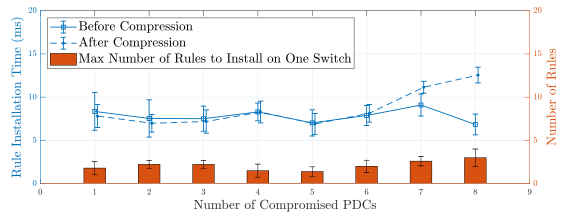
2) Rule Compression: We evaluate the performance of rule compression in terms of the total number of rules to install on the switches and the rule installation time.

Total Number of Rules to install on the SDN switches were measured before and after applying the compression technique, and the results are shown in Figure 6. We observe that the compression rate can reach up to 42.9% for the 30-bus cases and up to 59.0% for the 118-bus cases. As the number of compromised PDC grows, the total number of rules increases from 67 to 240 without compression, and from 67 to 98 with compression. The compression module greatly reduces the number of rules that the SDN controller needs to handle, especially when the number of compromised PDCs increases.

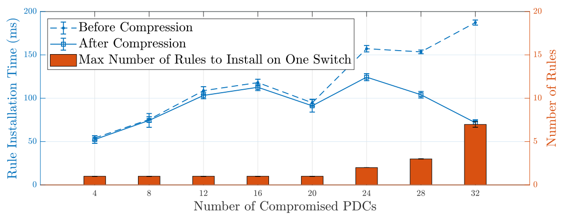
In our experiments, we set up a relatively large number of switches to form a ring topology, and therefore, the number of paths to install on the same switch is considered small. The rule compression technique can perform even better in the scenarios where switches handle more traffic flows from PMUs.

Rule Installation Time reflects how fast the self-healing scheme realizes the recovery paths. The total time consists of the time that the SDN controller generates and distributes the rules and the time that the switches install the rules onto the flow-entry tables. We repeated 20 experiments for every recovery path with the same compromised PDC. The results are plotted in Figure 7.

Without applying the rule compression, the rule installation completes in about 8.4 ms to 12.6 ms for the 30-bus cases and about 54.1 ms to 187.3 ms for the 118-bus cases, which is about one order of magnitude faster compared with the recovery plan generation and path construction time. The rule compression module further reduces the time, e.g., 6.8 ms for the 30-bus cases with 80% PDCs compromised. For the 118-bus cases, compression can save even more time, e.g., 71.8 ms with 80% PDCs compromised, which saves 61.7%



(a) IEEE 30-bus system



(b) IEEE 118-bus system

Fig. 7. Rule installation time with/without compression

of the installation time when compression is not used. The computational time (i.e., overhead) of the rule compression algorithm only takes up to 538.6 μs for 30-bus cases and up to 6.4 ms for 118-bus cases, thus it is beneficial to apply the rule compression for most cases.

In addition, we also plot the maximum number of rules to install on one switch, which is the objective function of the path construction model, in Figure 7. The results show a strong positive correlation to the rule installation time, which is the desired behavior as designed by the path construction ILP model.

VI. Conclusion and Future Works

We develop and evaluate a PMU network self-healing scheme built on top of an SDN-based network architecture.

In the future, we will consider the interdependencies between communication networks and power systems into the self-healing scheme. We also plan to extend the self-healing scheme to micro PMU networks for distribution systems and microgrids.

Acknowledgments. The authors are grateful to the support by the Air Force Office of Scientific Research (AFOSR) under Grant YIP FA9550-17-1-0240, the National Science Foundation (NSF) under Grant CNS-1730488, and the Maryland Procurement Office under Contract No. H98230-18-D-0007.

References

[1] X. Jiang, J. Zhang, B. J. Harding, J. J. Makela, and A. D. Domı´nguez-Garcı´a. Spoofing gps receiver clock offset of phasor measurement units. *IEEE Transactions on Power Systems*, 28(3):3253–3262, 2013.

[2] R. Deng, P. Zhuang, and H. Liang. Ccpa: Coordinated cyber-physical attacks and countermeasures in smart grid. *IEEE Transactions on Smart Grid*, 8(5):2420–2430, 2017.

[3] G. Rétvári, F.Németh, R.Chaparadza, and R.Szabó. OSPF for implementing self-adaptive routing in autonomic networks: A case study. In John C. Strassner and Yacine M. Ghamri-Doudane, editors, *Proceedings of Modelling Autonomic Communications Environments*, pages 72–85, Berlin, Heidelberg, 2009. Springer.

[4] H. Lin, C. Chen, J. Wang, J. Qi, D. Jin, Z. T. Kalbarczyk, and R. K. Iyer. Self-healing attack-resilient pmu network for power system operation. *IEEE Transactions on Smart Grid*, 9(3):1551–1565, 2018.

[5] Y. Qu, X. Liu, D. Jin, Y. Hong, and C. Chen. Enabling a resilient and self-healing PMU infrastructure using centralized network control. In *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, page 13–18, 2018.

[6] N. Dorsch, F. Kurtz, H. Georg, C. Hägerling, and C. Wietfeld. Software-defined networking for smart grid communications: Applications, challenges and advantages. In *Proceedings of the 2014 IEEE International Conference on Smart Grid communications (SmartGridComm)*, pages 422–427. IEEE, 2014.

[7] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk. Software-defined networking for smart grid resilience: Opportunities and challenges. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 61–68, 2015.

[8] L. Ren, Y. Qin, B. Wang, P. Zhang, P. B. Luh, and R. Jin. Enabling resilient microgrid through programmable network. *IEEE Transactions on Smart Grid*, 8(6):2826–2836, 2016.

[9] D. Jin, Z. Li, C. Hannon, C. Chen, J. Wang, M. Shahidehpour, and C. W. Lee. Toward a cyber resilient and secure microgrid using software-defined networking. *IEEE Transactions on Smart Grid*, 8(5):2494–2504, 2017.

[10] A. Cahn, J. Hoyos, M. Hulse, and E. Keller. 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)*, pages 558–563. IEEE, 2013.

[11] 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*, 43:142–154, 2015.

[12] H. Maziku and S. Shetty. Software defined networking enabled resilience for IEC 61850-based substation communication systems. In *Proceedings of the 2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 690–694. IEEE, 2017.

[13] T. Pfeiffenberger and J. Du. Evaluation of software-defined networking for power systems. In *Proceedings of the 2014 IEEE International Conference on Intelligent Energy and Power Systems (IEPS)*, pages 181–185. IEEE, 2014.

[14] J. Zhang, B. Seet, T. Lie, and C. Foh. Opportunities for software-defined networking in smart grid. In *Proceedings of the 2013 9th International Conference on Information, Communications & Signal Processing*, pages 1–5. IEEE, 2013.

[15] A. Sydney, D. S. Ochs, C. Scoglio, D. Gruenbacher, and R. Miller. Using GENI for experimental evaluation of software defined networking in smart grids. *Computer Networks*, 63:5–16, 2014.

[16] Y. Li, Y. Qin, P. Zhang, and A. Herzberg. SDN-enabled cyber-physical security in networked microgrids. *IEEE Transactions on Sustainable Energy*, 10(3):1613–1622, 2018.

[17] S. Al-Rubaye, E. Kadhum, Q. Ni, and A. Anpalagan. Industrial internet of things driven by SDN platform for smart grid resiliency. *IEEE Internet of Things Journal*, 6(1):267–277, 2017.

[18] C. Hannon, J. Yan, D. Jin, C. Chen, and J. Wang. Combining simulation and emulation systems for smart grid planning and evaluation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 28(4):1–23, 2018.

[19] R. L. Chen and J. Ruthruff. A scalable decomposition algorithm for PMU placement under multiple-failure contingencies. In *Proceedings of the 2014 IEEE PES General Meeting*, pages 1–5, 2014.

[20] M. He, V. Vittal, and J. Zhang. Online dynamic security assessment with missing PMU measurements: A data mining approach. *IEEE Transactions on Power Systems*, 28(2):1969–1977, 2013.

[21] M. Rifai, N. Huin, C. Caillouet, F. Giroire, D. Lopez-Pacheco, J. Moulierac, and G. Urvoy-Keller. Too many SDN rules? compress them with minnie. In *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, 2015.

[22] Mininet: An instant virtual network on your laptop (or other pc). http://www.mininet.org/.

[23] POX: a python-based sdn controller. https://github.com/noxrepo/pox.

[24] T. Finley. GLPK: GNU linear programming kit. http://tfinley.net/software/pyglpk/.