2022 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) | 978-1-6654-3254-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/SMARTGRIDCOMM52983.2022.9961035

A Framework to Evaluate PMU Networks for Resiliency Under Network Failure Conditions

Reuben Samson Raj, Dong Jin
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, USA
{rs077, dongjin}@uark.edu

***Abstract*—Phasor Measurement Units (PMU), due to their capability for providing highly precise and time-synchronized measurements of synchrophasors, have now become indispensable in wide area monitoring of power-grid systems. Successful and reliable delivery of synchrophasor packets from the PMUs to the Phasor Data Concentrators (PDCs) and beyond, requires a backbone communication network that is robust and resilient to failures. These networks are vulnerable to a range of failures that include cyber-attacks, system or device level outages and link failures. In this paper, we present a framework to evaluate the resilience of a PMU network in the context of link failures. We model the PMU network as a connected graph and link failures as edges being removed from the graph. Our approach, inspired by model checking methods, involves exhaustively checking the reachability of PMU nodes to PDC nodes, for all possible combinations of link failures, given an expected number of links fail simultaneously. Using the IEEE 14-bus system, we illustrate the construction of the graph model and the solution design. Finally, a comparative evaluation on how adding redundant links to the network improves the Power System Observability, is performed on the IEEE 118 bus-system.**

I. Introduction

Phasor Measurement Units (PMU) are currently the most prevalent form of measuring devices deployed in Wide Area Monitoring Systems (WAMS). PMUs provide highly precise and GPS time-synchronized measurements of an electrical phasor quantity (e.g., voltage or current). The resulting measurements are called synchrophasors. PMUs enable measurement of buses located at widely dispersed locations across the grid. Each PMU takes measurements at a sampling rate between 30 to 240 Hz. Continuous time-synchronized measurements at such high sampling rates enable power system operators to detect frequency imbalances or any stress on the power network, thereby preventing potential power outages.

The synchrophasors from the PMUs are transmitted to Phasor Data Concentrators (PDC) in the form of data packets as defined in IEEE standard C37.118.2 [1], supported by an underlying TCP/IP communication network (hereon referred to as PMU network). PMU networks and devices are vulnerable to a range of failures that include cyber-attacks, system or device level outages and link failures. Since PMUs play a critical role in preventing power outages, successful and reliable delivery of synchrophasor packets is of paramount importance and thus it is imperative that PMU networks are resilient to such failures.

978-1-6654-3254-2/22/$31.00 ©2022 IEEE

Several works exist in the area of evaluating and enhancing resiliency of PMU networks. A PMU placement model under failure contingencies is proposed in [2]. In [3], the authors present a data-mining approach to provide security assessment in the context of missing PMU data due to failed PMUs. An SDN based approach to detect a compromised PDC and install new forwarding rules in switches to re-route data to alternate PDCs, was developed in [4]. In [5], the authors propose a risk-mitigation model to provide an optimal response under cyber-attacks. A self-healing mechanism to restore Power System Observability in the event of compromised PMUs/PDCs was done in [6].

Our approach differs from these works in that, we have attempted to evaluate resiliency by quantifying the impact on power system observability under failure conditions. More specifically, we have attempted to present a framework to evaluate the resiliency of PMU networks under link failure conditions. In contrast to some of the works listed above, our proposed solution is not an attempt to detect a failure and provide real-time mitigation, but rather, given a PMU network design, our solution provides an evaluation of the resiliency of the network.

The design of our proposed framework is inspired by model checking and formal verification methods for transition systems and networks. Typically in these methods, a certain desired property, expressed in Linear Temporal Logic (LTL) or Computational Tree Logic (CTL), is verified if it holds true in all states of that system. Model checking tools such as [7], [8] achieve this by exhaustively traversing all possible states of the system and output a violation (known as counter-example) if any found. Model checking approaches have proven to be extremely successful in uncovering bugs, misconfigurations, protocol implementation errors, security vulnerabilities and preventing failure states in networks that could arise due to non-deterministic effects [9]–[11].

Although inspired by model checking, our proposed approach fundamentally differs from traditional model on two aspects. Firstly, we do not intend to verify any property, although property verification is certainly relevant in the context of PMU network resiliency. For instance, given a threshold observability T and an expected number of link failures occurring simultaneously, an LTL property to be verified could be of the form:

$G(O > T)$, that is – "Always (or in all possible failure states of the system), the Power System Observability O is greater than the threshold T".

In contrast, our approach involves modeling the PMU network as a connected graph and exhaustively checking reachability between PMUs and PDCs for all link failure combinations. We then evaluate resiliency as a measure of number of failure combinations for which there is zero or minimal loss in Observability. The components of the network (e.g., PMUs, PDCs and switches) are abstracted as nodes of the graph. The interconnects between the nodes are abstracted as edges of the graph, and a link failure is abstracted as an edge being removed from the graph. Modeling the PMU network as a graph offers several advantages. With our method of abstraction, graphs provide a better topological view of the network. Also, the availability of a number of fast graph search algorithms enables us to perform reachability checks between nodes on an exhaustive scale.

Secondly, while traditional model checking uses only network specific characteristics, our approach integrates both network and power system characteristics. As we will see in Sec II, the calculation of Observability % for a particular link failure combination requires a Bus-PMU mapping, which is computed based on power system characteristics. Also, traditional model checking involves exploring an exponential state space to discover violations and is thus slower compared to our approach.

The key contributions in our work are summarized as follows:

- The framework seeks to provide a fast evaluation of a PMU network operating in conjunction with a bus system. Our framework uniquely integrates both communication network and power systems characteristics to evaluate resiliency.
- We introduce pre-computed data structures to store basic paths that effectively reduce the search space and provide quicker run times.
- The framework is able to perform fast evaluation on the IEEE 118-bus system with optimally placed PMUs, in 209 ms and is scalable to larger networks.

The remainder of the paper is organized as follows : In Sec. II, we present a detailed description of the framework along with an illustrative example. In Sec. III, we present the experimental results and a comparative evaluation on the 118-bus system, with and without redundancy. Finally in Sec. IV, we conclude with potential directions for future work.

II. Evaluation framework design

A. Design Assumptions

The proposed evaluation is performed on a given PMU-PDC network along with a Bus-PMU mapping, both of which are assumed to be provided from the Utility provider. The Bus-PMU mapping is stored as a data structure that maps each PMU to the list of buses covered by that PMU. In development of our proposed evaluation framework, we make the following assumptions about the power system and the PMU network:

1) The scope of our network is restricted to PMUs, PDCs and the switches that connect them.
2) The Bus-PMU mapping is pre-computed at the Utility provider's end and so the mapping may include any form of measurement including Direct-Bus, Adjacent-Bus, and Zero-Injection Bus (ZIB) measurements.
3) The Observability metric used in this paper is defined as the ratio of buses that are observable to the total number of buses in the power system. A bus is considered observable if (i) it is covered by at least 1 PMU (defined by the aforementioned Bus-PMU mapping) and (ii) the PMU is reachable to a PDC.

Assumption #1 has been adopted to maintain simplicity in the topology. We do not include the parts of a WAMS that are beyond a local PDC, although our framework allows scaling towards sufficiently larger networks with a higher number of nodes. In regards to Assumption #2, for the demonstration of our framework, we have formulated a Bus-PMU mapping based on only Direct-Bus and Adjacent-Bus measurements. The presence or absence of ZIB's is not particularly relevant to the demonstration of our framework. In regards to Assumption #3, although different methods of computing Power System Observability exist in literature such as Numerical, Algebraic and Topological Observability [12], we have simplified the computation to the aforementioned ratio to demonstrate the loss in bus monitoring coverage for each link or device failure.

B. Network Modeling and Abstraction

We adopt the following modeling and abstractions of the power system and PMU network. Also listed below are notations used in the remainder of the paper:

- We model the PMU-Network as an undirected connected graph $G = (V, E)$ where V is the set of vertices representing PMUs, PDCs and switches, and E is the set of edges representing links.
- We denote P, D and S as the set of all PMUs, PDCs and switches respectively in V.
- A link could be between PMU and Switch, Switch and Switch, or Switch and PDC. For any pair of adjacent nodes $u, v \in V$, we abstract link failure as the removal of the corresponding edge (u, v) from G.
- Given k links are expected to fail simultaneously, we create a list $linksDownComb$ of $\binom{|E|}{k}$ combinations, where each combination c_i is a possible k-link failure, and $|c_i| = k$.
- The $pmuBusMapping$ is a pre-computed list that maps each PMU to the set of buses covered by that PMU. A bus can be covered by more than 1 PMU despite an optimal PMU placement.
- The $masterPathList$ is a pre-computed list to store basic paths for each PMU to a PDC.

C. 3-Step Algorithm with Illustrative Example

We present a description of the 3-step algorithm, with each step followed by an illustration. For the purpose of illustration, we employ the IEEE 14-bus system, with the 14 buses covered through 6 PMUs. The bus network in conjunction with the PMU network is shown in Fig. 1. We use 2 additional PMUs instead of the traditional optimal placement that employs only 4 PMUs for

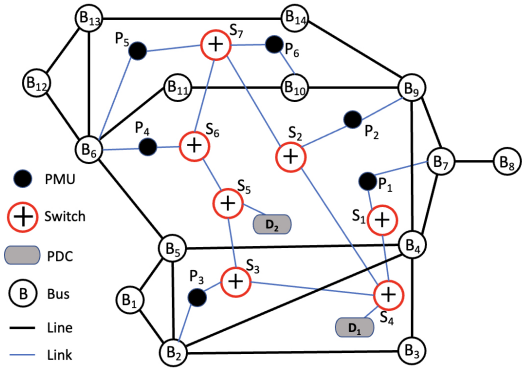

Fig. 1: IEEE 14-bus system along with PMU network

the 14-bus system so that a subset of the buses could be covered by more than 1 PMU. From Fig. 1, it can be seen that bus B_6 is covered by 2 PMUs (i.e., P_4 and P_5) through direct measurement and bus B_{10} is covered through both P_6 (via direct measurement) and P_2 (via adjacent-bus measurement).

In Sec. III, we demonstrate how including a minimal amount of redundant coverage (either in the form of additional PMUs or adding additional links) leads to a significant increase in overall observability of the system. The graph network G comprising of only PMUs, PDCs, and switches (excluding the buses) is depicted in Fig. 2.

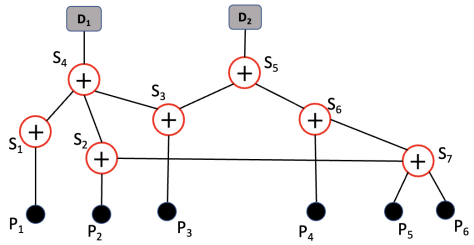

Fig. 2: Graph representation of PMU network excluding buses

Step-1: Generate masterPathList for each PMU. As a preliminary step, a list *masterPathList* comprising of basic paths, one for each PMU to a PDC, is computed. A path would be the list of nodes starting from that PMU to a PDC. The ordering of nodes in the list denotes the path. This list thus contains $|P|$ paths, each path indexed via its corresponding PMU.

The motivation to create this pre-computed list is to reduce the search space. Instead of having to search the entire graph, we are able to perform a quick preliminary check to obtain a list of potentially affected PMUs for a given link failure combination. This list is pre-computed using two guiding principles: (i) For a given PMU $p_i \in P$, the destination PDC $d_i \in D$ is selected such that d_i could be reached in minimal number of hops; (ii) There is an approximately even distribution of PDCs to PMUs. For a given network, we may certainly formulate more than one *masterPathList* and it is left to the framework user to create this list with a reasonable balance between the above two guiding principles.

Illustration. For the network in Fig. 2, a basic path for PMU P_1 would be the path $P_1 \rightarrow S_1 \rightarrow S_4 \rightarrow D_1$. Similarly, a basic path for PMU P_3 could be the path $P_3 \rightarrow S_3 \rightarrow S_5 \rightarrow D_2$, as shown in Fig. 3. While the path $P_3 \rightarrow S_3 \rightarrow S_4 \rightarrow D_1$ is an equally valid basic path, we avoid this in accordance with the principle (ii) mentioned in the preceding paragraph, which is to achieve an even distribution of PDCs to PMUs. Computing all such basic paths for each PMU in Fig. 2 results in the following *masterPathList* list:

$$
\begin{aligned}
\{\ & P_1 : [S_1, S_4, D_1], \\
& P_2 : [S_2, S_4, D_1], \\
& P_3 : [S_3, S_4, D_1], \\
& P_4 : [S_6, S_5, D_2], \\
& P_5 : [S_7, S_6, S_5, D_2], \\
& P_6 : [S_7, S_6, S_5, D_2] \\
\}&
\end{aligned}
$$

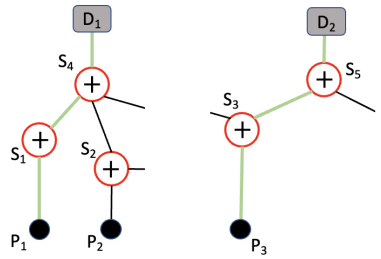

Fig. 3: Sub-graphs of the network showing basic paths for PMUs, P1 (left) and P3 (right). Highlighted green edges indicate path.

Step-2: Obtain list of affected PMUs. From the expected number of link failures k, all possible k-link combinations from E are generated and stored in the list *linksDownComb*. For each failure combination $c_i \in$ *linksDownComb*, the framework compares each link $l_i \in c_i$ against each PMU's basic path in *masterPathList*. If l_i is present in the basic path of a PMU, then the PMU is marked as 'potentially-affected' since the path to reach the corresponding PDC is now broken. PMUs whose basic paths do not contain any of the links from the combination remain unaffected. For each PMU in the potentially-affected list, the framework then checks if there is an alternate path to another PDC. If an alternate path is present, the PMU is removed from the affected PMU list (Lines 6-16 in Algorithm 1).

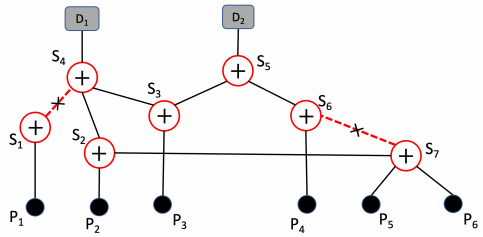

Fig. 4: Graph of PMU network showing failed links marked with dashed red lines

Illustration. We consider a failure scenario with $k = 2$, i.e., we expect that 2 links in the network could fail simultaneously. For the network in Fig. 2, we have $|E| = 15$. Total number of 2-link combinations is $\binom{15}{2} = 105$ combinations. Let us consider the 2-link failure combination [S_1-S_4, S_7-S_6], shown in Fig. 4.

Using the *masterPathList*, we obtain the list of potentially-affected PMUs due to this link failure combination as follows: link S_1-S_4 is present in the basic path for PMU P_1. Link S_7-S_6 is

present in the basic paths for PMUs P_5 and P_6. Thus the list of PMUs that are potentially-affected is : P_1, P_5, and P_6.

The framework then checks for alternate paths for each of the potentially affected PMUs P_1, P_5, and P_6, to a PDC. The computed alternate paths for these PMUs are as below:

$P_6 \rightarrow S_7 \rightarrow S_2 \rightarrow S_4 \rightarrow S_3 \rightarrow S_5 \rightarrow D_2$
$P_5 \rightarrow S_7 \rightarrow S_2 \rightarrow S_4 \rightarrow D_1$
P_1 : No alternate paths

From the potentially affected PMU list $[P_1, P_5, P_6]$, we remove P_5 and P_6 since they have alternate paths to reach a PDC. The final list of affected PMUs, which is $[P_1]$ in this case, is passed on to the next step of the algorithm to compute the number of impacted buses.

Step-3: Obtain list of unobservable buses. Using the final list of affected PMUs and the *pmuBusMapping* list, for each bus that is present in the coverage list of an affected PMU, the framework checks if the bus is covered through another unaffected PMU. If there is an alternate PMU covering the bus, the bus is marked Observable. Else, it is marked Unobservable. The Observability is then computed as the ratio: $\frac{\text{Total \# buses - \# unobservable buses}}{\text{Total \# buses}}$ (Lines 17-25 in Algorithm 1).

Steps 2 and 3 are repeated for each link failure combination in *linksDownComb*. Thus we obtain a final list of $\binom{|E|}{k}$ values of Observability, one for each link failure combination.

Illustration. For the network shown in Fig. 1, the Bus-PMU mapping, *pmuBusMapping*, would be as below, where we have considered direct-bus measurement and adjacent-bus measurement:

$\{\ P_1 : [B_7, B_8, B_9, B_4],$
$P_2 : [B_9, B_4, B_7, B_{10}, B_{14}],$
$P_3 : [B_2, B_1, B_3, B_4, B_5],$
$P_4 : [B_6, B_5, B_{11}, B_{12}, B_{13}],$
$P_5 : [B_6, B_5, B_{11}, B_{12}, B_{13}],$
$P_6 : [B_{10}, B_{11}, B_9]$
$\}$

As part of the final step, the framework checks if the buses mapped to P_1, that is $[B_7, B_8, B_9, B_4]$ are covered by any of the other unaffected PMUs. Through a simple search across the mapping, it could be computed that bus B_7 is covered by P_2, B_9 is covered by P_2 and P_6 and B_4 is covered by P_2 and P_3 (highlighted green in the mapping), while B_8 is not covered by any of the unaffected PMUs. With the total number of unobservable buses = 1, we compute the Observability % as $100 \times \frac{14-1}{14} = \mathbf{92.85}\%$

Note that this value of observability is computed for a single link failure combination. For all $\binom{|E|}{k} = 105$ combinations, a list of 105 values of observability is generated. A barplot of these value values and CDF of the Unobservability % is shown in Fig. 5. We define Unobservability % as 100 - Observability %.

It could be observed from Fig. 5 that in 45.71 % (48 out of 105) of failure combinations, the Observability is still retained at 100 %. This no loss in Observability could be attributed to one or more of the below 3 reasons:

1) The links of the failure combination are not part of any of the basic paths present in *masterPathList*, thereby resulting in no affected PMUs

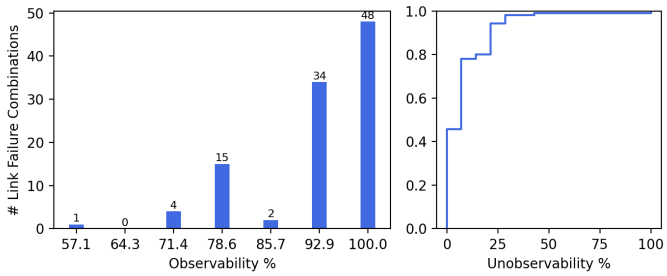

Fig. 5: Bar-plot of Observability % and CDF of Unobservability % for the 14-bus system

2) A potentially-affected PMU due to a link failure, is still able to reach a PDC through an alternate path.
3) For a PMU that is fully affected (that is, no alternate path to a PDC exists), the list of buses covered by that PMU is also covered through 1 or more un-affected PMUs.

Every link failure combination that resulted in zero loss of Observability would fall into either category 1, or a combination of categories 2 and/or 3. Also, the worst case Observability of 57.14 % occurs for the link failure combination [S_2-S_4, S_6-S_5] which resulted in $6(\frac{100\times(14-6)}{14} = 57.14\%)$ unobservable buses.

Note: Depending on the value of k, it is possible to have one or more failure combinations that result in 0 % Observability. For any design, the trivial case is when $k = |E|$. For designs where each PMU or PDC is served through only 1 switch, 0 % Observability would occur if $k \geq |P|$ or if $k \geq |D|$, that is, for the failure combinations involving all PMU-Switch links or all PDC-Switch links. The combination [S_4-D_1, S_5-D_2] in the above example, with $k = |D| = 2$, results in 0% Observability.

Algorithm 1 Evaluation Framework pseudocode

1: **Inputs**: PMU Network Graph G, k, *pmuBusMapping*
2: **Output**: Observability % values for each link combination
3: *observabilityList* = []
4: pre-compute *masterPathList* for each PMU ▷ Step-1
5: generate *linksDownComb* list ▷ Step-2
6: **for** each *linkCombination* in *linksDownComb* **do**
7: $affectedPMUs$ = []
8: *tempGraph* = G
9: **for** each *link* in *linkCombination* **do**
10: **if** *link* in *masterPathList*[*pmu*] **then**
11: add *pmu* to $affectedPMUs$ list
12: remove *link* edge from *tempGraph*
13: **for** each *pmu* in $affectedPMUs$ **do**
14: **if** alternate path exists from *pmu* to a PDC in *tempGraph* **then**
15: remove *pmu* from $affectedPMUs$
16: **for** each *pmu* in $affectedPMUs$ **do** ▷ Step-3
17: **for** each *bus* in list of buses covered by *pmu* **do**
18: **if** *bus* covered through another unaffected PMU **then**
19: mark *bus* as Observable
20: **else**
21: mark *bus* as Unobservable
22: $obsPercent = 100 \times \frac{\text{Total \# buses - \# buses marked Unobservable}}{\text{Total \# buses}}$
23: add (*linkCombination*, *obsPercent*) to *observabilityList*

III. Evaluation on 118-bus system

A. Experimental Setup

For our experimental evaluation with the 118-bus system, we designed the network topology (Fig. 6) as below.

- 118-buses mapped to 32 PMUs via the Optimal PMU Placement (OPP) [13]
- 6 PDCs to serve the 32 PMUs.
- 6 switches connected in a ring topology to serve as interconnect between the PMUs and PDCs. Each switch serves 1 PDC and around 5-6 PMUs.
- *masterPathList* of 32 basic paths, one for each PMU
- *pmuBusMapping* of 32 lists, one for each PMU
- Expected number of link failures $k = 2$

The PMU network graph, *pmuBusMapping*, *masterPathList* for the above setup is constructed in Topo-118Bus-withoutRedundancy.py at [14]. In Fig. 6, PMU subscript indices refer to the respective bus covered. For example, P_2 covers bus 2 of the 118-bus system. PMUs connected to only switch S_1 are shown for simplicity.

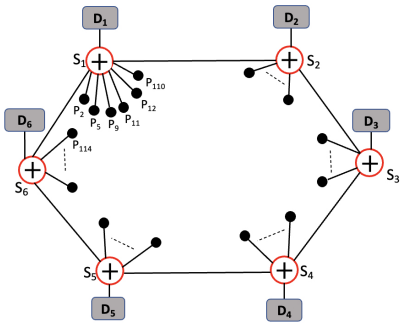

Fig. 6: PMU network of the 118-bus system.

B. Results for 118-bus without Redundancy

The experiments were performed on a MacBook Pro laptop computer, equipped with an M1 processor and 16 GB RAM. The Python implementation of the framework is available at [14]. The above topology has 44 links and thus 946 2-link combinations. On average, the framework took 209.14 ms to perform the evaluation. Fig. 7 shows the bar-plot for the 946 Observability values obtained (blue bars). Of the 946 combinations, 66 (6.98 %) resulted in 100 % Observability.

C. 118-bus with Redundancy

To demonstrate the effect of adding redundancy in the above topology, we first choose a threshold T_P ($T_P = 7$ in our 118-bus network) to select those PMUs in the *pmuBusMapping* list that cover T_P or more buses. For the *pmuBusMapping* list in our 118-bus, this resulted in identifying 5 PMUs that covered 7 or more buses. This method of adding redundancy is adopted, so that we may introduce additional links only for those PMUs that are relatively more critical in terms of number of buses covered. For these PMUs, in addition to their existing link to a switch, we add an additional link to a neighbouring switch. For example, PMU P_{12} covers 8 buses $[B_2, B_3, B_{11}, B_7, B_{117}, B_{14}, B_{12}, B_{16}]$ and is connected to switch S_1 in the non-redundant network. In the redundant network, we include an additional link from P_{12} to S_2, since $|P_{12}| > T_P$. The PMU network graph, *pmuBusMapping*, *masterPathList* for the above setup is constructed in Topo-118Bus-withRedundancy.py at [14].

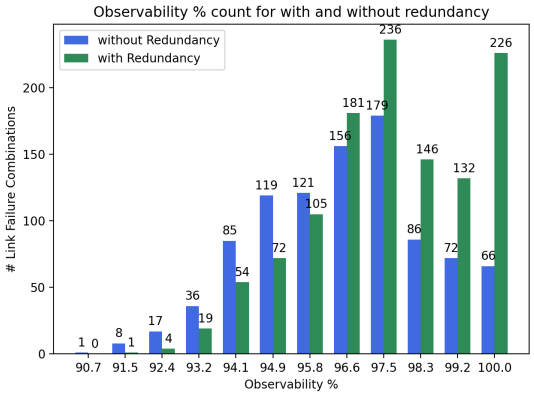

Fig. 7: # Link Failure combinations plotted against each Observability % value obtained after Step-3

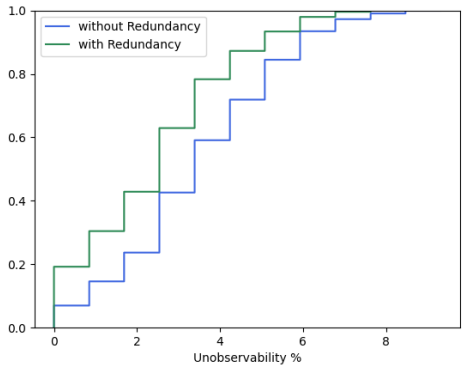

Fig. 8: CDF of Unobservability for with and without redundancy

D. Results of 118-bus with Redundancy

The introduction of 5 additional links resulted in 1176 2-link combinations, and thus 1176 Observability % values. On average, the framework took 256.2 ms to perform the evaluation. With redundancy, it can be seen from Fig. 7 that 226 combinations (or 19.21 % of 1176) resulted in 100 % Observability(green bars).

Consequently, the number of link combinations resulting in lower Observability % has also reduced. The addition of a very minimal number of redundant links resulted in a significant increase in higher Observability % counts and a significant decrease in lower Observability % counts. This is in direct contrast with the previous experiment. Fig. 8 shows the CDF for Unobservability % for with and without redundancy. The improvement in higher Observability % can be seen from the gap between the 2 CDF plots.

Alternatively, we may choose to evaluate our experimental results using a threshold Observability O_T. Consider a threshold $O_T = 97.5\%$. From Fig. 7 it could be observed that 224 out of 946 combinations (23.67 %) resulted in Observability greater than 97.5 %, for the case without redundancy. However, with redundancy we see that 504 out of 1176 combinations (42.86

%) resulted in Observability greater than 97.5 %, which is a significant increase from the previous case.

The choice of setting PMU criticality threshold value $T_P = 7$ in our case is rather arbitrary, and is intended only to demonstrate the effect of redundant links on the power system Observability. The choice of a threshold is left to the discretion of the framework user, after careful consideration of a trade-off between cost of additional links and increased Observability.

E. Run-time Comparison - IEEE 14-bus and 118-bus

As part of evaluating the run time, we executed the framework on another PMU network for the same 14-bus system. This network has a different topology with slightly fewer links (13 links) than the one used in the illustrative example in Sec. II. We refer to this network as 14-bus Design-II and the illustrative example as Design-I. Table I lists the framework execution times for the 14-bus Design II, 14-bus Design-I, and the two 118-bus systems with and without redundancy. From the table, it could be observed that the run-time increases linearly with increase in number of link combinations.

TABLE I: Computation time for 14-bus and 118-bus systems

	PMUs	Links	Link combinations*	Run time†	Std. dev
14-bus Design-II	6	13	78	6.64 ms	0.07 ms
14-bus Design-I	6	15	105	9.72 ms	0.24 ms
118-bus without redundancy	32	44	946	209.14 ms	1.52 ms
118-bus with redundancy	32	49	1176	256.2 ms	1.66 ms

* Number of combinations for expected link failures $k = 2$
† Average run time over 10 executions

F. Benefits and Applicable Use-cases

- Our framework enables a power utility provider to comparatively evaluate multiple network designs that have different topologies, yet subject to the same failure conditions.
- Our framework offers flexibility in selecting all or a subset of link types that could fail. For instance, the scenario where only Switch-Switch links are expected to fail, while PMU-Switch or PDC-Switch links are not expected to fail.
- Our framework offers flexibility in simulating failure conditions involving more than one type of network element. For instance, the scenario where a user might wish to evaluate resiliency given that at any given time, k links and l PMUs fail simultaneously.
- To check for presence of an alternate path, our framework allows the user to define their own search algorithms. Popular algorithms include Breadth-First-Search(BFS), Depth-First-Search(DFS), A*(A-star), Djikstra's and Bellman-Ford. Our current implementation provides the option to choose either BFS or DFS during run-time.

IV. Conclusion and Future Works

In this paper, we presented a framework to evaluate resiliency of PMU networks under link failures. Our experiments show that our framework is fast and scalable to larger networks. Our framework also allows flexibility in simulating a wide range of failure scenarios. In future, we plan to perform evaluation by adding weights to links. Assigning weights to links could help in simulating latency and consequently, the reachability algorithm would check for reachable paths under the constraints of latency. We also plan on optimizing the alternate path search. Another direction of interest is modelling a PMU device, as opposed to a network. This poses unique challenges due to the complexity of the synchrophasor data protocol and is likely to consume a higher execution time. Modeling PMU devices to simulate protocol behaviour would enable us to explore security vulnerabilities in the protocol.

Acknowledgement

The authors are grateful to the support by the NSF Center for Infrastructure Trustworthiness in Energy Systems (CITES) under Grant EEC-2113903 and the Maryland Procurement Office under Contract No. H98230-18-D-0007.

References

[1] IEEE standard for synchrophasor data transfer for power systems. *IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)*, pages 1–53, 2011.

[2] 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.

[3] 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.

[4] Y. Qu, G. Chen, X. Liu, J. Yan, B. Chen, and D. Jin. Cyber-resilience enhancement of PMU networks using software-defined networking. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7. IEEE, 2020.

[5] J. Valenzuela S. Mousavian and J. Wang. A probabilistic risk mitigation model for cyber-attacks to PMU networks. *IEEE Transactions on Power Systems*, 30(1):156–165, 2014.

[6] 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.

[7] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A.Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuXmv symbolic model checker. In *International Conference on Computer Aided Verification*, pages 334–342. Springer, 2014.

[8] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on software engineering*, 23(5):279–295, 1997.

[9] M. Musuvathi and D.R Engler. Model checking large network protocol implementations. In *Proceedings of the First Symposium on Networked Systems Design and Implementation*, volume 4, pages 12–12, 2004.

[10] B. Liu, A. Kheradmand, M. Caesar, and P.B. Godfrey. Towards verified self-driving infrastructure. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, pages 96–102, 2020.

[11] S. Prabhu, K.Y. Chou, A. Kheradmand, P.B. Godfrey, and M. Caesar. Plankton: Scalable network configuration verification through model checking. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 953–967, 2020.

[12] M. Shahraeini and M.H. Javidi. A survey on topological observability of power systems. In *2011 IEEE Power Engineering and Automation Conference*, volume 3, pages 373–376. IEEE, 2011.

[13] N.M. Manousakis and G.N. Korres. A weighted least squares algorithm for optimal PMU placement. *IEEE Transactions on Power Systems*, 28(3):3499–3500, 2013.

[14] Reuben Samson Raj. PMU evaluation framework. https://github.com/reuben89raj/PMU-Eval-Framework.git.