SAND2021-2273C

Next-Generation Relay Voting Scheme Design Leveraging Consensus Algorithms

Nicholas Jacobs*[1], Adam Summers[1], Shamina Hossain-McKenzie[1], Daniel Calzada[1],
Hanyue Li[2], Zeyu Mao[2], Chris Goes[1], Katherine Davis[2], Komal Shetye[2]
[1]Sandia National Laboratories
[2]Texas A&M University
*njacobs@sandia.gov

***Abstract*—Traditional protective relay voting schemes utilize simple logic to achieve confidence in relay trip actions. However, the smart grid is rapidly evolving and there are new needs for a next-generation relay voting scheme. Specifically, the ability to include inter-relay relationships and out-of-band data is needed; in this work, we explore the use of consensus algorithms and how they can be utilized to achieve consensus on both values and trip decisions. A proposed design is explored with a simple case study with two different scenarios, including simulation in PowerWorld Simulator, to demonstrate the consensus algorithm benefits; future directions are provided for comprehensive development.**

***Keywords*–cyber-physical system, protective relays, consensus algorithms, voting schemes**

I. Introduction

Protection schemes for the electric grid are designed to maximize system reliability, which involves upholding dependability and security. These protection schemes are comprised of protective relays, breakers, sophisticated fault detection, identification, and location algorithms, etc [1]. Their function is to mitigate disturbances such as faults and prevent cascading impact within the grid with quick isolation and mitigation. Depending on the type of system and its inertial characteristics, communication-assisted protections schemes are also being designed to deal with fast dynamics that require additional speed. Nonetheless, protection schemes play a critical role in maintaining grid reliability.

With protection scheme's integral role in grid reliability, it is important that when relays take action (e.g., opening/closing breakers), they do not misoperate. Voting schemes compare trip decisions from different relays, for the same measurements, and apply logic (e.g., two-out-of-three) for the final trip decision. In this manner, confidence in the trip action can be achieved and redundancy is obtained with the usage of multiple relays. The relays can be connected in series or parallel, depending on the logic used for the final tripping decision [2].

This type of relay implementation is most commonly seen in transmission systems, where redundancy is paramount. The placement, redundancy, and addition of advanced features (e.g., communications) can vary depending on deployment within transmission systems versus distribution systems versus microgrids. In this paper, we focus on transmission system protective relay for use within an adaptive, online real-time special protection scheme (SPS), discussed next.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525; this work is funded by Sandia National Laboratories LDRD project #222444. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

This work is a part of a laboratory directed research and development (LDRD) project titled "Harmonized Automatic Relay Mitigation of Nefarious Intentional Events (HARMONIE) Special Protection Scheme." The HARMONIE-SPS project is developing an adaptive and reactive SPS that learns system conditions, mitigates cyber-physical consequences, and preserves grid operations during both predictable and unpredictable events. A novel aspect of the HARMONIE-SPS is its usage of digital relay cyber-physical measurements (e.g., power system measurements and device configuration/settings)and deployment of proactive relay response.

However, with this increased reliance on relays, high confidence in the relay actions and measurements is necessary. In this paper, we aim to review traditional protective relay voting schemes and how the novel needs of the smart grid, including the need to mitigate nefarious, intentional events, necessitates next-generation voting schemes. Specifically, we explore the use of consensus algorithms and how they can be utilized to create a next-generation relay voting scheme.

II. Next-Generation Voting Scheme Needs

A. Traditional Voting Schemes

Voting schemes are used to achieve balance between security and dependability; they provide confidence in the resulting relay action, ensuring that there is redundancy in available relays and that the correct action is taken [1]. A typical voting scheme utilizes two-out-of-three trip logic and can involve two or more protection schemes protecting certain lines. Additionally, one-out-of-two and two-out-of-two logic can also be employed. In a paper by Altuve et al., the different logic implementations are discussed; two-out-of-three and two-out-of-two trip logic is focused on ensuring security while one-out-of-two trip logic focuses on dependability [2].

B. Novel Needs of the Smart Grid

As mentioned previously, the grid is becoming increasingly cyber-physical with the addition of new communication-enabled technologies, third-party software, internet connectivity, and remote interfaces. These advancements have greatly improved the operation of the grid, but have also broadened the attack landscape. Furthermore, extreme weather events and electromagnetic pulses (EMPs) are unpredictable events of concern. With these threats, it is important to adopt a

defense-in-depth approach for protecting the grid but also to improve the overall resilience of the interconnected system.

Disturbances in the smart grid can range from equipment failures and extreme weather events to more advanced events such as EMPs and cyber attacks. Cyber attacks are a growing concern due to the "smart" technologies being implemented and increased reliance on automated functions [3]–[6]. Furthermore, the unpredictable nature of these events, whether extreme weather or malicious cyber attacks, render planning and playbook response near impossible due to changing event trajectories.

Therefore, protective relays play a more essential role than ever by protecting the electric grid not only from standard power system faults but also preventing cascading failures stemming from unpredictable disturbances such as cyber attacks and EMPs. When the physical power system begins seeing signs of stress such as sustained limit violations, the protective relays can coordinate to isolate the impact and prevent further cascading physical impact. The importance of this safety system is also recognized by adversaries. In fact, in 2017, the Triton malware framework specifically targeted the Schneider Electric Triconex safety controller, which often serves as a critical defense mechanism against physical incidents. Although Triton was successfully detected by security researchers during network reconnaissance activity, the malware is difficult to detect on previously infected systems, as it employs the proprietary TriStation protocol and has the capability to overwrite data and return the controller to a running state to mask an attack [7].

It is critical to defend the operation of grid protective relays and to prevent relay compromise. The misoperation of relays must be prevented, whether due to measurement errors or relay compromise, to ensure the grid safety system can operate when needed and prevent cascading system impact.

C. Consensus Algorithms

Consensus algorithms are a class of problems in which a distributed set of agents need to reach agreement. This can fall into several forms, including problems like social decision making such as leader elections or opinion agreement, distributed averaging problems, or other related problems. As an example, PageRank can be considered to be a large-scale consensus problem for ordering the importance of nodes in a large directed graph (in this case, by using hyperlink connections to rank pages in the world wide web) [8]. Another prominent example are distributed ledgers, such as blockchain, which are a bit different but still seek to achieve consensus among distributed and possibly untrusted parties [9]. More information about distributed consensus algorithms can be found in [10], [11].

Consensus algorithms also have some interesting and useful properties for fault tolerance and security. For instance, moving an algorithm to a distributed version can often give benefits since a centralized algorithm results in a common point of failure. By taking that computing task and distributing it to multiple computing devices, the failure of one node might affect the result but will not necessarily cause the algorithm to fail completely. The security implications are similar, where we can look at the probability of cyber attacks successfully causing security failures.

However, moving to a distributed algorithm such as a consensus algorithm does introduce potential issues as well. One of these issues is based on the Byzantine General's problem, which describes a situation where a set of parties must agree on a strategy even though one or more of the parties are corrupt or unreliable. This can be related to distributed algorithms as Byzantine faults, which are arbitrary (malicious or inadvertent) failures of individual nodes in the network, and can be found discussed in a variety of settings, such as in [12]. One benefit of designing Byzantine Fault Tolerant (BFT) algorithms is that it makes it much easier to identify and isolate nodes that are misbehaving. Furthermore, when considering cybersecurity related scenarios and potential failures, we cannot assume that a node will fail in a specific way since a insecure relay could be made to either not respond (for instance, Denial-of-Service) or it could be modified to take incorrect actions. In this paper, we will introduce the basic formulation for a BFT inspired relay voting scheme but will not delve too deeply into the details of different variations of BFT algorithms and the implementation of them, such as cryptographic considerations for authentication. Further details and discussion beyond what we cover here on these considerations can be found in [13], [14].

III. Application of Consensus Algorithms to Relay Voting Schemes

There are a few ways that consensus algorithms could be utilized to augment existing techniques for secure and resilient power system protection using relays in the electric grid. Specifically, they can be utilized to detect when a relay is compromised or unintentional settings are applied; the remaining set of relays could help flag these issues and/or update to correct settings. Furthermore, consensus algorithms could be used to learn boundary protection zones between relays to minimize system interruptions. During an extreme interruption, consensus algorithms could be used to correctly island the system so each island is self-supporting.

A. Preliminaries

To analyze consensus algorithms for application in relay voting schemes, a few things are needed. First, to setup the problem we need to define the structure for how agents or nodes can communicate and how to relate the distributed equations calculated by each agent with the global behavior of the entire system.

1) Algebraic Graph Theory: A graph G has a set of vertices V and a set of edges $E := (v_i, v_j), \forall v_i, v_j \in V$. A path p_{ij} over graph G is a set of edges connecting vertex i to vertex j. G is strongly connected if $\exists p_{ij}$ from any node in G to any other node in G. If G has its edges replaced with undirected edges, and there exists a path through all vertices in G, then G is weakly connected.

To analyze graphs in a more algebraic setting, we can define some related matrices for the graph G and study properties of these matrices. One matrix of interest here is the adjacency matrix A, which has a non-zero element a_{ij} if there exists and edge between v_i and v_j, and is 0 if there is no such edge. These matrix properties have interesting connections to the behavior for convergence and consensus. For the distributed averaging system (1), if A is irreducible, then we will achieve consensus but it will not be the average of the initial states $x(0)$. If A is primitive, we will achieve consensus to the average of $x(0)$. This can be shown be

examining the eigenvalues and eigenvectors of A, and further discussion on this topic can be found in [10].

IV. Proposed Next-Generation Relay Voting Scheme Design

We now propose a new application of consensus algorithms to next-generation relay voting scheme designs for communication enabled relays. Here, we intend to rigorously analyze how distributed relays can split some calculations and check results across groups of relays for voting, while also address how each relay can adequately satisfy system protection requirements. We will start by considering simple averaging schemes for relays to agree on measurement values and system state, and then incorporate this into a proposed relay voting scheme that we apply here with under frequency load shedding for demonstration.

A. Consensus on Values

One way consensus can play a role in relay voting schemes is for relays to use consensus algorithms to reach agreement on the values and settings that are used for any required computations for the relay protection schemes. That is, reach consensus on variables such as settings, thresholds, and on measured variables. In this way, the relays can check values and average out differences or discrepancies. Note that this can be run as part of the BFT relay voting scheme proposed in the last section, as relays in a voting group will need to agree on the values for the measurements used in the replicated operations for the protection scheme.

If we take the distributed averaging system

$$x(k+1) = Ax(k) \tag{1}$$

where $x_i, i \in 1, ..., n$ is the value of some parameter or measurement that we want to reach consensus one at relay i, and n is the number of relays in the graph. x_i here could be voltage/current/power measurements or some other parameter setting. If we apply this equation repeatedly, we will see that

$$\lim_{k\to\infty} x(k+1) = A^k x(0) \tag{2}$$

From which we can see that convergence and consensus rely on the structure of A and the final result will depend on both the weights of A and the initial values of each node. Consider a simple two-bus system with two generation sources, two loads, and six relays (two generator relays, two line relays, and two load relays). We will consider several variations for the distributed consensus, one where the connections are: R1-R5, R2-R6, R3-R4, R1-R2-R3, and R4-R5-R6, which can be seen in Figure 2, and in the variation we will only connect relays connected to each bus, namely: R1-R2-R3 and R4-R5-R6, as shown in Figure 3.

Averaged consensus can be achieved using weighting schemes such as Metropolis-Hastings, such as is done in [15].

The weighted adjacency matrix for this sample system with Metropolis-Hastings weights is

$$A_1 = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 1/4 & 1/4 & 1/2 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 1/4 & 1/4 & 1/4 \\ 1/4 & 0 & 0 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 0 & 1/4 & 1/4 & 1/2 \end{bmatrix} \tag{3}$$

$$\begin{aligned} \lim_{k\to\infty} x_i(k+1) = 0.167x_1(0) + 0.167x_2(0) + 0.167x_3(0) \\ +0.167x_4(0) + 0.167x_5(0) + 0.167x_6(0) \end{aligned} \tag{4}$$

which is true because of (2). In other words, every relay will converge to the same value and achieve consensus, and that consensus will be the average of $x(0)$. Note that this weighting scheme incorporates information about the relative degrees of nodes to ensure that all get weighted equally.

See [10] for details and related background on when such an algorithm will achieve consensus, which is very much connected to properties of the graph and of the transition matrix A.

To fit this process into a relay voting scheme, one consideration is that any distributed averaging we perform will take multiple samples to converge, which will result in delays due to the communication overhead. Therefore, the convergence rate for the consensus algorithm will tell us how quickly new events affect the averaged values. See [16] for further discussion on these matters.

Furthermore, when considering the averaging of measurement signals, we can modify the process by incorporating new measurements into a rolling average for each term, as shown in (5).

$$x_i(k) = pq_i(k) + (1-p)x(k) \tag{5}$$

where q_i is a local measurement of a physical value in the power system, such as current or voltage. With some $p \in (0, 1)$ we can weight the importance of previous values and new measurements, similar to how a weighted moving average works. In this instance all previous is stored in the last value of x_i. Note that if $p = 0$, then only the initial values $x(0)$ will matter and we will reduce to our previous problem in (1), and conversely if $p = 1$ then only the latest measurements are used.

Before continuing, it is important to consider that in a power system there will be variations in system variables such as frequency and voltage due to the topology of the power system and because relays are located in different locations in the system. This will affect the choice for which relays go into the same voting group, as we will want relays that

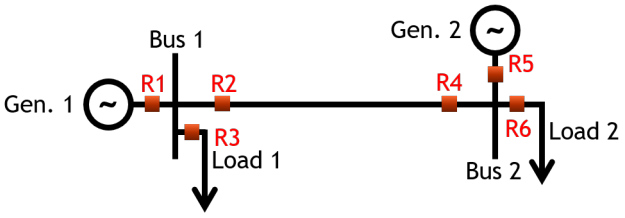

Fig. 1: Simple two-bus system with two generation sources, two loads, and six relays.

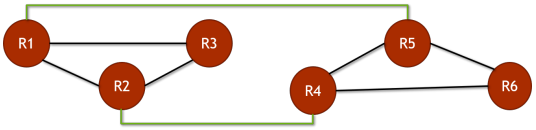

Fig. 2: Relay Connections with all six relays connected.

are close enough that they can all easily observe the local behavior in the circuit.

To demonstrate this, the simple 2-bus system shown in Figure 1 is used. In a scenario where generator 2 fails at 1.05 seconds, the voltages and averaged values using our scheme with $p = 0.05$ in the system are as shown in Figure 4. Note how all x_i are close to the average voltage of the entire system, and the variation between each x_i is due to the discrepancy between the latest voltage measurements between the 2 buses.

To further touch on this discussion, a second variation of this simple exemplar system can be seen in Fig. 3, with the adjacency matrix (6). In this system, we have broken the relays into 2 separate groups instead of only 1 group.

$$A_2 = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \end{bmatrix} \tag{6}$$

Here, we can note that we will not achieve global consensus among all relays but will instead have 2 connected subgraphs and all the nodes that are part of each subgraph will achieve consensus. In other words, we can analyze relays $\{1, 2, 3\}$ and $\{4, 5, 6\}$ separately and see that $\{x_1, x_2, x_3\}$ will converge to the same value and $\{x_4, x_5, x_6\}$ will converge to a value, but these two groups will not agree with each other in general. This is good, as we want to be able to specify groups of relays that need to be able to reach consensus among themselves but only consist of a subset of relays in the graph.

Furthermore, we can disconnect some of these edges and still achieve this result. For instance, if node 3 is disconnected due to being unresponsive the adjacency matrix A_2 would be modified to become $A_2^{'}$ as shown in (7).

$$A_2^{'} = \begin{bmatrix} 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \end{bmatrix} \tag{7}$$

To ensure that the value x_i does not die out and converge to 0, we need to add the requirement that for any node that any missed value is just dropped and ignored. This fits in well into the BFT relay voting scheme we will propose in the following section as any node that fails to update values for the averaging calculation will be ignored and flagged, but cannot impede the calculations of the other relays in the group from completing.

All this ties into the relay voting scheme proposed in Section IV-B, as it is important for the relays to be able to check that their values match. Additionally, a failure of

Fig. 3: Relay Connections with two disconnected subgroups of relays, one for each bus in two-bus system.

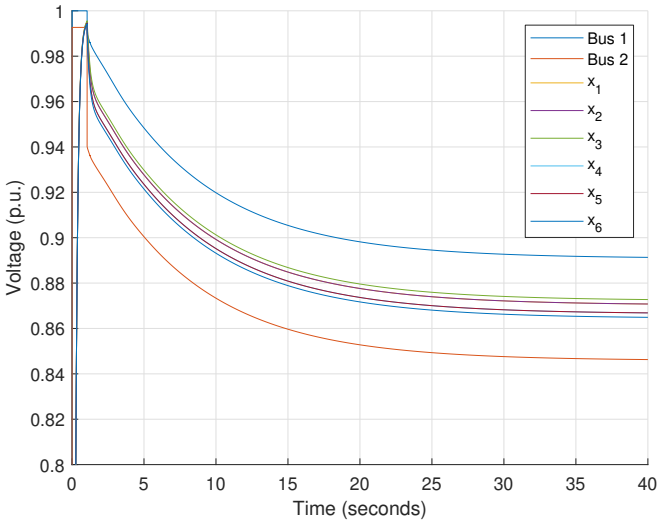

Fig. 4: Voltage measurements from Powerworld simulation of simple 2-bus system after generator 2 fails at 1.05 seconds

a relay to respond to the distributed averaging calculations or a continued discrepancy in results will help to show when a relay has either failed or is misbehaving. We will next go into how relays can vote on the correct steps to take to protect the system.

B. Consensus Algorithms Applied to Relay Voting

Consensus algorithms give us a way to characterize the relationships between relays, and to study the properties of any proposed voting procedure. By applying consensus algorithms we can more closely look at the security and resilience of the resulting system as we can examine the additional overhead that comes from additional required communications and how the algorithm reacts to nodes that have arbitrary failures.

One approach to developing a relay voting scheme is for each agent to reach a decision and vote on the desired protection action. This type of formulation ties in well with state machine replication approaches found in distributed computation and consensus, as seen in algorithms such as the the BFT algorithm. Here, we will apply ideas from BFT to develop a voting scheme for the relays to both achieve consensus and allow for the discovery of nodes that have are misbehaving without losing the ability for the relay voting group overall to protect the system and limit any potential damage from misbehaving relays. For this, we will draw from the Practical BFT algorithm (PBFT), developed in [13], and the Robust BFT (RBFT) algorithm as developed in [14]. While the PBFT variant provides the general structure for this scheme, the RBFT algorithm is a variant that includes duplicates of "primary" nodes to ensure that malicious "leaders" in the voting group cannot impede performance, which means that the ordering of requests is not limited to a single node. This is the approach we will take in our work, and we will also add in that in our relay voting scheme there is no client, but rather the relays themselves will create requests per local conditions and protection scheme requirements.

Since the BFT algorithm is a form of state machine replication, we can use it to create redundant copies of the execution

steps for each individual relay. In practice, this means that the calculation for load shedding, or other protection actions, is duplicated between all the relays in a voting group. When a round of voting is initiated, each relay will compute the desired action to take according to the replicated operation, and will broadcast its result to the other relays in the group. The relay that needs to act will wait for enough replies from the other relays to ensure the consensus action is taken. This entire process can be seen in Algorithm 1.

Algorithm 1 Relay voting with BFT

1) Relay i detects under frequency conditions
2) Relay i initiates request
3) Request for voting multicast to all other relays
4) All relays compute protection scheme calculations, determine load to shed
5) Each relay multicasts result to all other relays in group
6) Each relay waits for $f + 1$ replies, saves result.
7) Relay j that needs to shed load acts accordingly

In the implementation of this algorithm, there are several important considerations. First of all, when any relay in the group sees an event, such as under frequency conditions, the relay makes a request to all other relays in the voting group.

Each relay will wait until it gets $f + 1$ replies, where f is the max number of allowable node failures in order to get the correct result from a voting round. This is directly related to the number of nodes n in the voting group. For example, if there are 3 nodes that are voting, f will be 1 and this will be very similar to two-out-of-three voting. This ensures that the entire group of relays can reach the correct consensus. Note that if more than f nodes fail, then this scheme will not reach the correct result and the value for f is an important consideration in the implementation of these schemes.

In the case of arbitrary node failures, we need to account for instances for an individual relay does not respond to a request in a voting round, a relay does not create a new request when it should, and when a relay lies about its local measurements, conditions, or appropriate protection actions. We also need to consider the amount of communications overhead required and minimize this as much as possible. Discussion on this topic can be found in the background material in [13], [14] and we will address further in future work.

By using the setup from RBFT in [14] where we incorporate multiple primary nodes, we can ensure that the calculation of requests is not impeded when a relay does not respond to requests or send updates on its own values $x_i(k)$ in a timely manner. In dealing with relays that do not initiate requests when they should, we allow for any relay in the group to start a request. In protection schemes such as Under Frequency Load Shedding (UFLS), other relays besides the misbehaving relay will also see the under frequency conditions in the system and so the request will still get generated. If an individual relay lies about its measurements, it can be detected by the other relays by checking for large deviations between the consensus values for those measurements and the reported value, $||x_j - x_i|| > \epsilon \; \forall j \neq i$ for some $\epsilon > 0$. This is one place where the distributed averaging process we described in the last section comes in. The threshold for allowed deviations $\epsilon > 0$ needs to be designed to allow for normal variations due to relay location within the power system, as seen in the small normal variations seen in Figure 4. Note that x_j and x_i are the averaged values at nodes j and i respectively, and relay i is just checking if the values being reported by the other relays is agreeing with its own result.

Another potential issue that arises is a relay that is supposed to take protective action and does not. In this case, the other relays in the group will see the under frequency conditions persist and any expected improvement will not match the actual response. The other relays will see that the measurements in the system do not match what should be occuring if the protective action was taken and so in this case they can flag the relay that is not cooperating. At this point, a new round of voting could be initiated on secondary actions to take assuming that the relay that is not cooperating has failed. In this case, we repeat calculations to find a new set of actions to take that will minimize system impact.

When the relay voting group removes a misbehaving node, it will be ignored in future voting rounds until manually checked and re-verified. An alert may be initiated at this point for maintenance, including various steps like replacing failed equipment, reflashing firmware, and other maintenance activities. Each relay in the voting group will create this alert and send it to the system operator, who will wait for $f + 1$ alerts before proceeding, in order to eliminate spurious alerts from individual relays that may be incorrect.

C. Simple Case Study: Under Frequency Load Shedding

Here we dive deeper into the simple 2-bus system shown in Figure 1 and in Figure 5. The dynamic model is simulated in PowerWorld Simulator and analyzed using the transient stability tool. Here we will use UFLS so that if system frequency drops below the operational set point during major disturbance such as loss of generation, the system arrests declining frequency and assists recovery of frequency following under frequency events, such as described by FERC in [17].

Consider a contingency where Generator 2 drops off at 1.05 seconds and Generator 1 must keep supplying power to both Load 1 and Load 2 (Critical). Without a UFLS scheme, Generator 1 is isochronous and manages to stabilize the system frequency, however the frequency does drop to 59.86Hz. With a simple UFLS scheme, when the frequency is below 59.95Hz Load 1 is dropped, dramatically reducing the amount of frequency drop that is observed to only 59.986Hz. Note that while the overall impact in both these cases is rather limited, we are using this small system mainly to keep the analysis simple and illustrative for the relay voting.

The purpose of UFLS is to protect the Bulk Electric System (BES) against a major loss of generation. In the Western Electricity Coordinating Council (WECC), UFLS must drop sufficient load to keep the system frequency within the continuous operation range of the generation units (59.5 Hz and 60.5 Hz) [18]. Using a consensus algorithm ensures that the correct non-critical loads are removed first, and then critical loads can be removed in stepwise until the system frequency is stabilized. In this scenario, relay 3 in Figure 1 will need to disconnect load 1 to perform the UFLS scheme. If relay 3 does not respond and disconnect the load after the group of relays agrees on this action, the frequency in the system will follow the no load shedding curve in Figure 6. This is easily distinguishable by all relays checking their

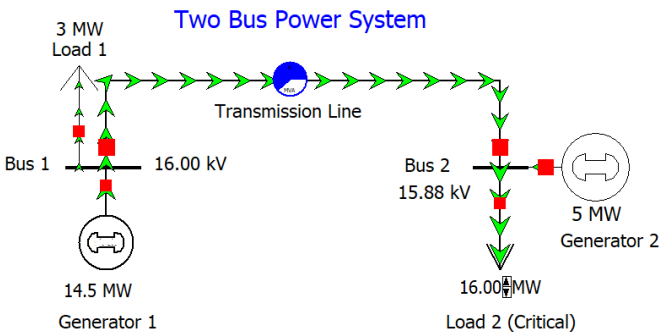

Fig. 5: Two-bus system modeled in PowerWorld Simulator.

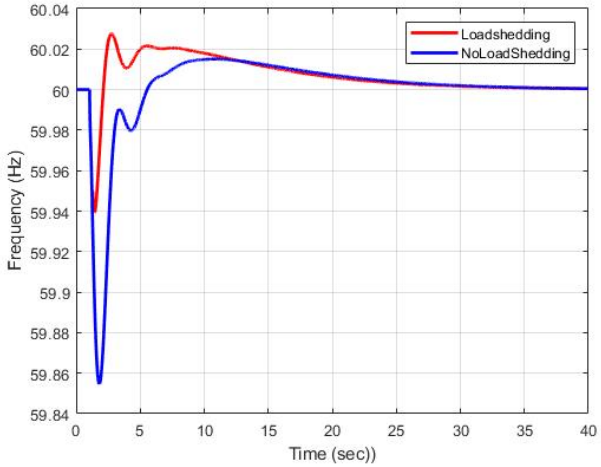

Fig. 6: Impact on frequency with and without load shedding in two-bus system modeled in PowerWorld Simulator.

value for frequency measurements x_i and seeing that it is not following the desired improvement in the system response. relay 3 would then be flagged by the other relays in this system and the system operator alerted by the other relays that it has failed.

In this 2 bus system there are no redundant connections so a disconnection by a nefarious relay would either remove a load, generator, or separate the 2 buses entirely. In practice, that is a failure that other relays cannot remedy if they are not also located at the same connection point, but they can quickly detect it has occurred and if able decide on actions to take to mitigate the damage to the system overall. Further discussion on that is left for future work where we will extend this procedure to larger systems.

V. Conclusions

Consensus algorithms can be utilized to develop next-generation relay voting schemes, providing ways to incorporate inter-relay relationships and out-of-band data to better protect the power system as a whole. Distributed calculation of system values, such as with distributed averaging, can be utilized to provide information that relays can check each others values and ensure that overall an entire group of relays is reaching agreement on the state of the power system. Furthermore, by incorporating knowledge about byzantine faults in distributed computing, we can develop new voting scheme algorithms for relays to use to agree on protective actions to take following various protection schemes, such as under frequency load shedding. By adding in this type of design into the system, relays are able to check and verify each others actions, and alert when other relays in the voting scheme fail to operate correctly. This allows for faster detection of mitigation of failures and potential security issues. In this paper, we have demonstrated this type of design on a simple 2-bus system that is meant to be illustrative and in follow on work are extending this to more realistic and larger systems, and will extend to other protection schemes besides load shedding and further examine how relay compromises are dealt with and mitigated.

References

[1] "IEEE Guide for Protective Relay Applications to Transmission Lines," *IEEE Std C37.113-2015 (Revision of IEEE Std C37.113-1999)*, pp. 1–141, 2016.

[2] H. J. Altuve, K. Zimmerman, and D. Tziouvaras, "Maximizing line protection reliability, speed, and sensitivity," in *2016 69th Annual Conference for Protective Relay Engineers (CPRE)*, 2016, pp. 1–28.

[3] C. Lai, N. Jacobs, S. Hossain-McKenzie, P. Cordeiro, O. Onunkwo, and J. Johnson, "Cyber Security Primer for DER Vendors, Aggregators, and Grid Operators," Sandia National Laboratories, Sandia Report SAND2017-13113, Dec. 2017.

[4] A. Chavez, C. Lai, N. Jacobs, S. Hossain-McKenzie, C. B. Jones, J. Johnson, and A. Summers, "Hybrid Intrusion Detection System Design for Distributed Energy Resource Systems," in *2019 IEEE CyberPELS (CyberPELS)*, 2019, pp. 1–6.

[5] S. Zonouz, K. M. Rogers, R. Berthier, R. B. Bobba, W. H. Sanders, and T. J. Overbye, "SCPSE: Security-Oriented Cyber-Physical State Estimation for Power Grid Critical Infrastructures," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1790–1799, 2012.

[6] Texas A&M University, "Deep Cyber Physical Situtational Awareness for Energy Systems: A Secure Foundation for Next-Generation Energy Management," 2020. [Online]. Available: https://cypres.engr.tamu.edu/

[7] M. K. D. S. N. B. C. G. Blake Johnson, Dan Caban, "Attackers Deploy New ICS Attack Framework TRITON and Cause Operational Disruption to Critical Infrastructure," *FIREEYE*, 2017. [Online]. Available: https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html

[8] H. Ishii and R. Tempo, *The PageRank Problem, Multi-Agent Consensus and Web Aggregation A Systems and Control Viewpoint*, 2013, _eprint: 1312.1904.

[9] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[10] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[11] A. Olshevsky and J. N. Tsitsiklis, "Convergence Speed in Distributed Consensus and Averaging," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 33–55, 2009, _eprint: https://doi.org/10.1137/060678324. [Online]. Available: https://doi.org/10.1137/060678324

[12] N. Gupta, S. Liu, and N. H. Vaidya, *Byzantine Fault-Tolerant Distributed Machine Learning Using Stochastic Gradient Descent (SGD) and Norm-Based Comparative Gradient Elimination (CGE)*, 2020, _eprint: 2008.04699.

[13] M. Castro and B. Loskov, "Practical Byzantine Fault Tolerance," in *Proceedings of the Third Symposiumon Operating Systems Design and Implementation*, New Orleans, USA, Feb. 1999.

[14] P. Aublin, S. B. Mokhtar, and V. Quma, "RBFT: Redundant Byzantine Fault Tolerance," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*, 2013, pp. 297–306.

[15] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33 – 46, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731506001808

[16] D. Spanos, R. Olfati-Saber, and R. Murray, "Dynamic Consensus for Mobile Networks," 2005.

[17] "Automatic Underfrequency Load Shedding and Load Shedding Plans Reliability Standards," FERC, Tech. Rep., Oct. 2011. [Online]. Available: https://www.ferc.gov/sites/default/files/2020-05/E-3_25.pdf

[18] "Underfrequency Load Shedding Program Assessment Report," Western Electricity Coordinating Council, Tech. Rep., Feb. 2018. [Online]. Available: https://www.wecc.org/Reliability/WECC%20UFLS%20Assessment%20Report%20-%202018_Approved.pdf