Secured Fault Tolerant Wide Area Monitoring Architecture (WAMS) in the Power Grid

Anamitra Datta
Department of Electrical and Computer Engineering
University of Massachusetts Amherst
Amherst, USA
anamitradatt@umass.edu

Brendon Burke
Department of Electrical and Computer Engineering
University of Massachusetts Amherst
Amherst, USA
brendonburke@umass.edu

***Abstract*— A measurement system that incorporates PMUs (Phasor Measurement Unit) deployed over large portions of the power system is known as the Wide Area Monitoring System (WAMS). With the advent of smart power grids, PMUs are being used everywhere to locate faults instantly through GPS-synchronized timestamps. However, PMUs, being a networked device are prone to attacks. This paper proposes a strategic solution to mitigate such outside attacks on the power grid by deploying the Byzantine Fault Tolerant Consensus Building approach in power grids to stop the propagation of erroneous signals generated by any compromised PMUs or PDCs (Phasor Data Concentrator) outside of the local substations.**

I. INTRODUCTION

After the infamous 2003 cascading blackout of the Northeastern Grid, there was a renewed interest towards modernizing the power distribution system in order to isolate faults before they cascade to the entire network. Hence, Phasor Measurement Units (PMU) were introduced to improve the current Supervisory Control and Data Acquisition (SCADA) system. PMUs are power system devices that provide synchronized measurements of real-time phasors of voltages and currents. In this approach, data generated by the PMUs is transmitted to the local Phasor Data Concentrator (PDC) who in turn relays the data to the higher-level PDC. The higher level PDCs gather such information from all over the power grids and compare them for a particular instance by using the timestamps provided by the synchronized PMU data. For example, by measuring any difference in voltage level between two adjacent nodes, a fault in a bus connection can easily be detected. This is how WAMS (Wide Area Monitoring System) in smart power grids works, which can detect a fault in the power line within nanoseconds and prevent the propagation of such fault by sending appropriate signals back to the control centers of the appropriate substation. Unfortunately, PMUs and PDCs are sophisticated electronic devices connected to the network which are prone to outside attacks or failure. Once the PMU data is committed to the database, there is no way to roll it back. So, there is a critical need to detect bad data transmitted by a malfunctioning or hacked PMU and relay this information to the PDC in order to prevent faulty analysis. In this paper, we are proposing an architecture where PMU data is communicated to multiple regional PDCs. The PDC network can then use a majority consensus method by employing the Byzantine Fault Tolerance (BFT) algorithm before submitting the data for fault analysis. Figure 1 shows the proposed architecture.

II. PROPOSED SYSTEM

A. Anatomy of a Fault

As we are increasing the complexity of power systems by the deployment of the smart grid and additional interconnected devices, we are also increasing the probability of an attack on these devices from the outside world. So, failure can occur at any level. It has been observed that the faults are increasing in geometric progression. About 2/3rd of the faults in the power line happen in transmission lines. There are many reasons for these faults such as contact between two or more lines due to a natural calamity. It is impossible to physically monitor every section of power lines. So, the only indicator of such faults are the voltage, power and current measurements at the end nodes of these transmission lines. The problem is even worse if the nodes at both ends of these transmission lines are part of two different substations. The only way to get a time synchronized measurement of such a scenario is to run analytics on data gathered by a super PDC which receives PDC data from other geographical regions. So, it is of paramount importance to get and validate the PMU data before it goes out of the substation PDC network. Any PDC or PMU compromised by an outside attack, if unchecked, can send wrong data which can then propagate through the system and go into the central repository, thereby rendering the fault detection mechanism useless.

B. Byzantine Fault Tolerance Implementation Architecture

BFT is a Byzantine Fault Tolerant state machine replication algorithm that is safe in asynchronous systems. In the context of a power distribution system, PDCs provide replication services for a request originating from a particular PMU. The request typically is a measurement of a parameter like voltage, phase angle, etc. at any particular node at a particular instance under a specific substation. These replicas are sent point-to-point. They can use a cryptographic hash function to compute the message digest. Additionally, they can use a MAC (Message Authentication Code) to authenticate all replicas. BFT assumes very little from the nodes and the network. So a faulty node, in this case a PMU or a PDC, can behave arbitrarily by sending wrong data packets. These are called the traitor nodes.

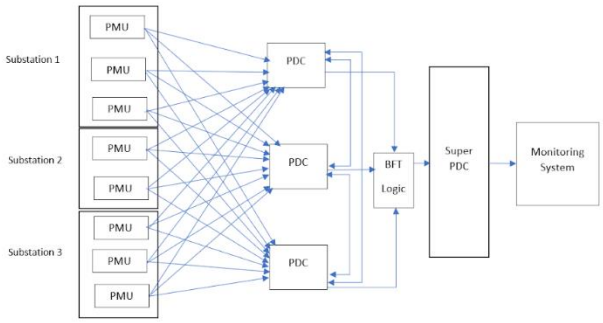

Fig. 1. Proposed Wide Area Monitoring System (WAMS) Architecture

The network can fail to deliver messages to its recipient in many ways. They can send stale, incorrect, delayed or duplicated data. The reasons for such behavior can vary as well. The nodes may be compromised due to hacking. They could also malfunction. It is also possible that they are impersonated by an outside hostile agent.

C. Byzantine Fault Tolerance Algorithm

Whatever be the case, BFT assumes nothing about the characteristics of such faults. It rather runs a consensus building algorithm with the participating nodes, where after exchanging numerous messages, they all individually come to a decision about the validity of a particular request. It can be proved mathematically that if we have n number of participating nodes in consensus building where n = 3f+1, then if there are at most f nodes which are malfunctioning, after the algorithm runs successfully, we are guaranteed to generate 2f+1 number of commit messages with the correct value. Figure 2 shows the BFT algorithm in the context of the smart power grid WAMS architecture.

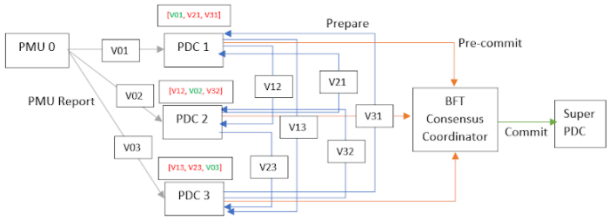

Fig. 2. Byzantine Fault Tolerance algorithm and consensus building process in Wide Area Monitoring System (WAMS)

For simplicity, Figure 2 shows the replication process of a request generated by a single PMU to three PDCs. By our assumption, any one of these nodes could be faulty. There are three distinct phases of this operation after the initial request has been made: prepare, pre-commit, and commit. In the prepare phase, every PDC builds its own data vector with an element of the same data with the same timestamp from the same PDC. It makes no judgement of the validity of any data based on any external criteria. If it doesn't receive the expected data due to delay, it simply fills it with an unknown value. At the end of the process, all the functioning PDCs take the majority value from its own vector and pre-commits it to the BFT module. If it can't come to a single majority value, it refrains from any action. In the pre-commit phase, the BFT module simply waits for at least 2f+1 certified messages from valid MACs. If it receives 2f+1 certified messages from 2f+1 different PDCs, it achieves quorum and proceeds to commit the request to the higher-level PDC.

This paper aims to describe such fault tolerant mechanisms in power grids with real life examples and the flow of data through the system. We hope to generate data from multiple PMUs (some of them may be compromised) and flow them through the process described above by applying the BFT consensus building before submitting them to the monitoring process at the super PDC. This will prevent the propagation of faulty data and cascading failures across substations located in different geographical areas.

III. Implementation

The simulation environment has been implemented through Python objects. For simplicity, we have precluded the super PDC from our environment. We also have provided a system where all PMUs and PDCs are part of one substation. But this limitation doesn't prevent us from experimenting and simulating signals from various combination of PDCs and PMUs, of which some can be compromised by an outside attack. This will be our traitor nodes in the WAMS. As it will be evident, the BFT mechanism will suppress any data from a compromised PMU or PDC without trying to identify the nature of the signal. The simulation has been tested in many environments, but for this paper, we have illustrated the results of a WAMS architecture comprising of two PMUs and six PDCs, in five different scenarios. The architecture of our WAMS demonstration network is shown in Figure 3.

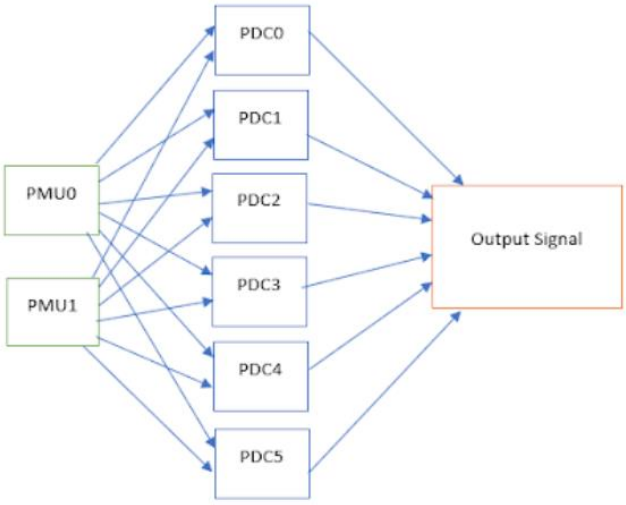

Fig. 3. WAMS System Architecture for simulation

The following are five different scenarios that have been tested on the given WAMS system.

1. Working PMU and PDCs
2. Attacked PMU and Working PDCs
3. Working PMU and one attacked PDC

4. Working PMU and two attacked PDCs
5. Working PMU and three attacked PDCs

A. *API Implementation*

We have three primary objects.

1. Substation
2. PMU
3. PDC

Where a substation is a singleton object, but it can instantiate multiple PMUs and PDCs and add them into the network. It is possible to set and reset a PDC as a traitor node for the purpose of staging an attack on a compromised device in the network. Each device can transmit a custom signal defined by the signal generator function, which can be customized as well.

The following is an example of a simulation code of a WAMS network.

```
st = Substation(0)
substation_list.append(st)

# Creating PMU 1 & 2
pmu0 = st.add_pmu(signal_generator,1) #faulty
pmu1 = st.add_pmu(signal_generator)

# Creating PDC 1 to 6
#all PDCs are not faulty initially
pdc0 = st.add_pdc()
pdc1 = st.add_pdc()
pdc2 = st.add_pdc()
pdc3 = st.add_pdc()
pdc4 = st.add_pdc()
pdc5 = st.add_pdc()
```

IV. RESULTS

We ran a simulation on the proposed network described in Figure 3. We simulated five different environments explained in the beginning of section II. The following is the simulation code for the designed WAMS network in Python.

```
# Simulation scenario 1
pmu1.start_transmit(st,10)
#consensus reached

# Simulation scenario 2
pmu0.start_transmit(st,10)
#no consensus reached because PMU itself is faulty

# Simulation scenario 3
pdc0.set_faulty(1) #set PDC 0 to faulty
pmu1.start_transmit(st,10)
#consensus reached from 2/3rds voting

# Simulation scenario 4
pdc3.set_faulty(1) #set PDC 3 to faulty
pmu1.start_transmit(st,10)
#consensus reached from 2/3rds voting

# Simulation scenario 5
pdc4.set_faulty(1)  #set PDC 4 to faulty
pmu1.start_transmit(st,10)
#no consensus reached because 3 PDCs out of 6, which is less than 2/3rds
```

1. Working PMU and PDCs. Consensus reached.

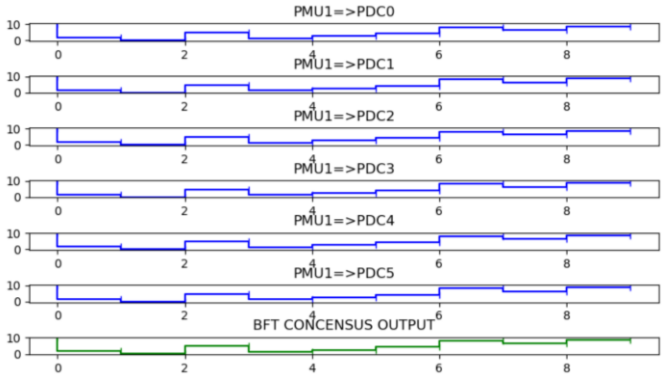

Figure 4. Result of scenario 1. Consensus reached due to agreement on signal vectors over time.

2. Attacked PMU and working PDCs. No consensus reached.

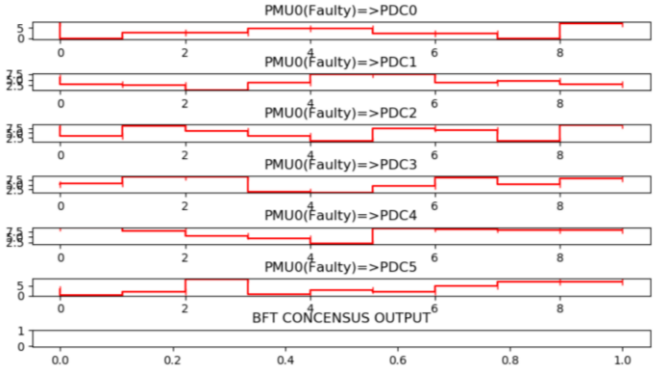

Figure 5. Result of scenario 2. Consensus not reached due to total disagreement on signal vectors over time.

3. Working PMU and one attacked PDC. Consensus reached.

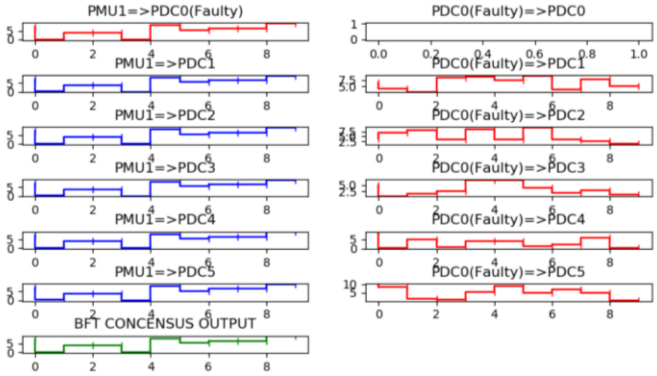

Figure 6. Result of scenario 3. Consensus reached due to 2/3rds majority agreement on signal vectors over time. The attacked PDC value/input is essentially ignored to prevent a fault or wrong value to be propagated into the entire system and network

4. Working PMU and two attacked PDCs. Consensus reached.

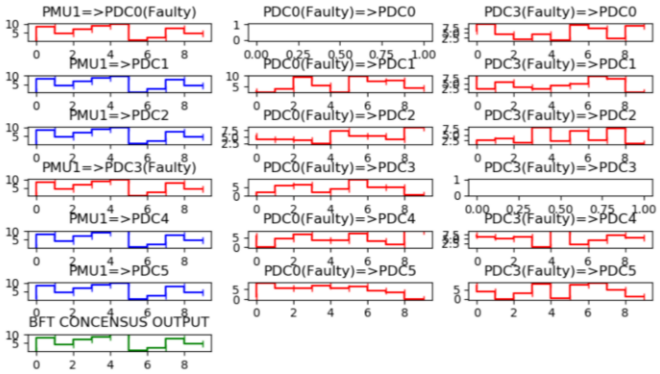

Figure 7. Result of scenario 4. Consensus reached due to 2/3rds majority agreement on signal vectors over time. The attacked PDCs values/inputs are essentially ignored to prevent a fault or wrong value to be propagated into the entire system and network

5. Working PMU and three attacked PDCs. No consensus reached.

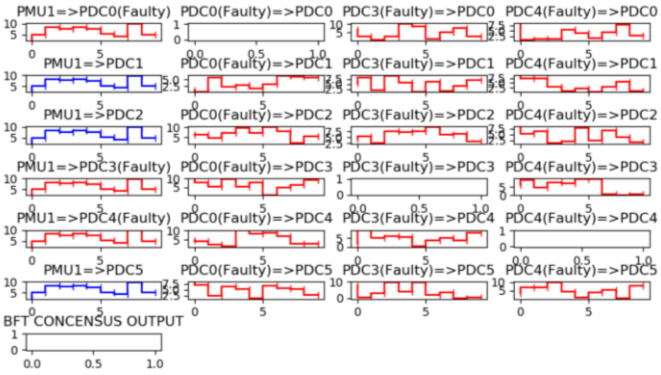

Figure 8. Result of scenario 5. Consensus not reached due to no 2/3rds majority agreement on signal vectors over time. The attacked PDCs values/inputs are blocking the correct input/value from being sent/recorded by the super PDC/monitoring system, so the compromised nodes do not affect the output of the system

V. Conclusion

In this paper, we have studied the feasibility of the Byzantine Fault Tolerant techniques to monitor and detect the overall health of a smart power grid system. BFT not only prevents propagation of wrong data by a malfunctioning device, it also suppresses any data emitted by device which is suspected to be compromised by some outside party. The framework doesn't assume any particular technology. It rather suggests an algorithmic approach to find a pattern of data flow and employs a prevention mechanism for any case where it is suspected that intrusion has happened. Our Byzantine fault tolerance framework is adapted from the highly efficient BFT algorithm developed by Castro and Liskov. This is a distributed system and relies on no particular device to work perfectly. Hence there is no underlying assumption that there is a central node who is responsible for the action. This is also the strength of the approach because the attacker cannot target any particular device in the network. The algorithm doesn't primarily focus on detection of the traitor node. It rather tries to mitigate the damage done to the traitor nodes. However, one can always run analytics and find such pattern to detect the source of any disruption and detect such traitors with some accuracy.

This process also ensures that whatever the condition, the network will never stop working and guarantee liveness without compromising safety. However, this paper doesn't take into account of out-of-sync packets or delayed packets inside the network. Future work can be done with variant of BFT which allows these conditions without inordinate delay in the network.

VI. Contributions

Anamitra Datta designed the system and implemented this architecture through a Python program. Brendon Burke was responsible for testing the system and making cases for different scenarios of the given architecture.

VII. Acknowledgements

We would like to thank Professor Ganz for the support in this project and for her suggestions, advice, and questions and for laying out the problem definitions and specifications

References

[1] W. Zhao and F. E. Villaseca, "Byzantine Fault Tolerance for Electric Power Grid Monitoring and Control," 2008 International Conference on Embedded Software and Systems , Sichuan, 2008, pp. 129-135. doi: 10.1109/ICESS.2008.13

[2] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," ACM Transactions on Computer Systems, vol. 20, no. 4, pp. 398–461, Jan. 2002.

[3] B. Singh, N. Sharma, A. Tiwari, K. Verma, and S. Singh, "Applications of phasor measurement units (PMUs) in electric power system networks incorporated with FACTS controllers," International Journal of Engineering, Science and Technology , vol. 3, no. 3, 2011.

[4] S. Kumar, M. Soni, and D. K. Jain, "Monitoring of Wide Area Power System Network with Phasor Data Concentrator (PDC)," *International Journal of Information Engineering and Electronic Business*, vol. 7, no. 5, pp. 20–26, Aug. 2015.