Pacific Northwest
NATIONAL LABORATORY

PNNL-33317

End-to-end Analytics for Grid Arch Design & All-hazard Assessment

August 2022

Dexin Wang
Renke Huang
Jianming Lian
Henry Huang

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL-33317

End-to-end Analytics for Grid Arch Design & All-hazard Assessment

August 2022

Dexin Wang
Renke Huang
Jianming Lian
Henry Huang

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

Resiliency, reliability, and security of the next-generation smart grid depend upon leveraging advanced communication and computing technologies, integrating them with physical power systems, and developing real-time, fast, data-based applications to help in wide-area monitoring and control of the grid. Using a high sampling data rate from phasor measurement units (PMUs) to develop applications has opened the door to achieving the next-generation grid requirements. The North American Synchrophasor Initiative Network (NASPInet) was developed in 2007-09 to create a standard and guide for PMU data exchanges. With the advancement in both networking and grid requirements, it is necessary to evaluate the performance of different NASPInet versions and their impact on applications. Therefore, we need a cyber-power co-simulation framework that supports very large-scale co-simulation capable of running in parallel, high-performance computing platforms and capturing real-life network behavior. This work presents a cyber-physical co-simulation testbed using NS3 to model the communication network, GridPACK to model the power grid, and HELICS as a co-simulation engine. Comparative analysis of latency in synchrophasor networks and a performance evaluation of a power system stabilizer application based on PMU data in an Institute of Electrical and Electronics Engineers 39-bus test system is presented using this co-simulation testbed.

Acknowledgments

This research was supported by the Energy and Environmental Directorate (EED) Mission Seed, a Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). The computational work was performed using PNNL Institutional Computing at Pacific Northwest National Laboratory. Part of the research was performed using the Environmental Molecular Sciences Laboratory (EMSL), a national scientific user facility sponsored by the DOE's Office of Biological and Environmental Research and located at PNNL. PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830.

Contents

Figures

Tables

1.0 Introduction

POWER systems, which are evolving as smart grids, require many real-time applications to be developed and run to provide better insight about grid dynamics, with the help of better monitoring and fast automatic control capabilities. These applications need data inputs at a much higher rate, which are made possible by the increasing use of phasor measurement unit (PMU) in the transmission grid [1]. PMU can provide synchronized measurements at the rate of 30–120 samples per second, which is much higher than that of the traditional SCADA systems, thereby, creating the opportunity to develop many real-time applications and implement them into the smart grid [2].

One of the key factors to consider while running these PMU-based applications in a real power system is to ensure timely delivery of data [3]. Most of these applications will be effective in real-time scenarios if they meet their stringent latency requirement of a few milliseconds to several seconds [4]. To work with this massive amount of data exchange and timely data delivery, having a fast, reliable, and resilient underlying communication network is one of the vital things to consider while designing network architecture for synchrophasor data networks [5].

Given the importance of the communication network, in 2007, the North American Synchrophasor Initiative (NASPI) was formed to develop a sustainable framework for the design of synchrophasor data communication networks (NASPInet) [6]. Since then, the NASPInet framework is treated as the standardized communication network guideline for synchrophasor applications. NASPInet is generally formed with the integration of PMUs, phasor data concentrator (PDC), phasor gateways connected to a data bus, and a centralized phasor data concentrator called Super PDC [7].

Since the development of NASPInet, however, some things have changed significantly, such as increased data volume, network technology, protocol advancement, and more. Although most of the NASPInet design concepts are useful, initiative has been taken over the last several years revisiting the previous architecture to overcome the operational, maintenance, and implementation limitations, thus, preparing the framework for NASPInet 2 [6]. Previously, output of the PMU-based applications depended on precise synchronization and lower packet loss ratio. However, for current and future applications, low communication latency is equally important [6].

All these communication performance requirements make the synchrophasor network one of the most important components, considering the PMU-based applications are crucial for power system operation. Some previous work on the performance evaluation of wide-area synchrophasor networks using simulation appears in [8]–[11]. The use of PMU for real-time monitoring, control, and resiliency applications in the future is dependent on the re-evaluation of NASPInet architecture to NASPInet 2, with analysis and comparison of different network performance issues.

In order to evaluate the synchrophasor network and its impact on different applications before deploying in the real environment, we need to test and validate these networks in the simulated platform, which exhibits similar behavior to real systems. Therefore, researchers need to have a scalable, multi-featured co-simulation platform, and priority should be given to this area of research. In the past, there has been some work on developing a co-simulation testbed aimed to research and validate PMU networks and PMU-based applications, which can be found in [12]–[15]. From the literature of synchrophasor based co-simulations, it can be found that, for network simulations OMNET++, NS2, NS3, Mininet, and power system simulations OpenDSS,

PSLF, PSCAD, and MATLAB have been used primarily. For the co-simulation engine EPOCHS [16], ADEVS [17] has been used. None of the work developed synchrophasor network components in network simulator to emulate PMU-based traffic and analyze its impact on power system applications using co-simulation engine HELICS [18]. This paper presents a cyber-physical co-simulation testbed capable of analyzing the impact of communication network performance on various power system control applications and comparing different network performance issues for NASPInet 1 and NASPInet 2. We have developed different synchrophasor components in NS3 using C++ capable of exchanging data following Institute of Electrical and Electronics Engineers (IEEE) C37.118 protocol (used for PMU data exchange) [19] to model the communication network and used GridPACK, a software framework able to model the power grid and utilize high-performance computing. We have used HELICS, a powerful and scalable co-simulation engine for our testbed. IEEE 39-bus standard system is used to simulate and validate our test cases. The contributions of this paper can be summarized as follows:

- Developed PMU and PDC application following C37.118 frame structure in NS3, which can exchange data similar to the real PMUs
- Developed Synchrophasor sink application and Historian application in NS3 to simulate real-time synchrophasor network and perform end-to-end synchrophasor simulations with a provision of storing the logs
- To validate and demonstrate the testbed usage, powerful co-simulation Engine HELICS, a cyber-physical co-simulation testbed, was built where NS3 and GridPACK communicate via HELICS
- Developed a controllable graphical user interface from which users can create a different synchrophasor network and visualize its impact on different synchrophasor applications.

2.0 Co-Simulation Platform Architecture

Synchrophasors are time-synchronized electrical measurements consisting of voltage and current phasor, frequency, and rate of change of frequency. A lot of applications based on this fast timestamped measurement have already been developed. In the future, this trend will provide a better real-time picture of the grid. For these real-time applications, latency is a crucial factor that may significantly affect grid monitoring and control. The two main components of the synchrophasor network are PMUs and PDCs. A PMU is a device used to estimate the magnitude and phase angle of an electrical measurement, such as voltage or current, using a common time source, generally using GPS for synchronization. PDC receives data from multiple PMUs, which is combined and then sent to the applications as synchronized measurements [20]. This facilitates synchronized real-time, wide-area monitoring of the grid [21]. The developed cyber-physical co-simulation platform, as shown in Figure 1, is an architecture to evaluate the performance of synchrophasor networks and their impact on power system applications.

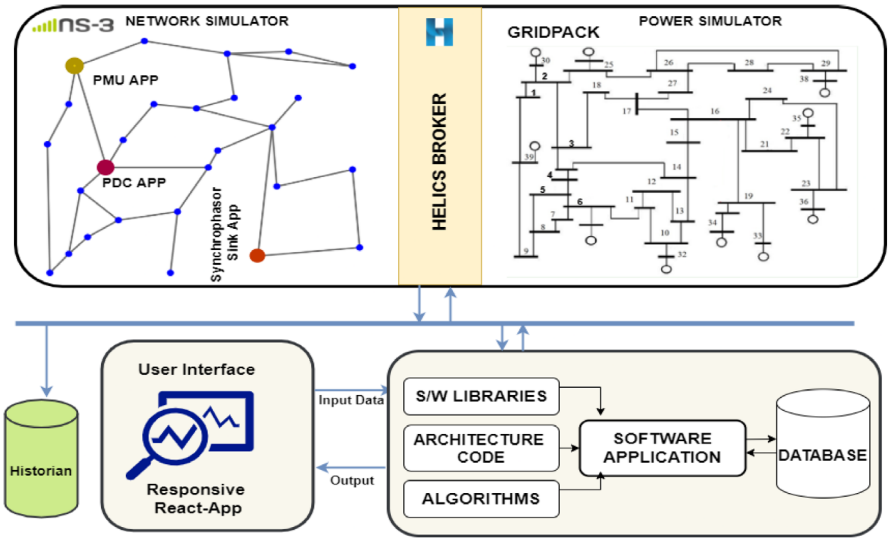

Figure 1. Cyber-Physical Co-simulation Architecture

The main components of this co-simulation platform include

- NS3: a discreet event communication network simulator
- HELICS: a scalable and powerful co-simulation engine
- GridPACK: a software framework for power grid modeling and data generation
- Graphical User Interface: a user interface to modify the synchrophasor network and visualize results.

2.1 NS-3

NS-3 is a discrete-event network simulator used widely by the scientific community for research and educational use. One of the promising features of NS3 is that we can develop and implement protocols within NS3 and install them in NS3 network nodes, which can exhibit the behavior of real-life communication NS3 [22]. Protocols that support smart grid communication, such as DNP3, C37.118, or Modbus, are not integrated with NS3, making it difficult for

researchers to test and validate applications based the data carried by those protocols. In this work, we have developed and implemented PMU and PDC functionality in NS3, which can be installed in NS3nodes to emulate an end-to-end synchrophasor network similar to real systems. The network functions developed include

- PMU App
- PDC App
- Historian App
- Synchrophasor Sink App.

2.1.1 PMU App Implementation in NS3

The PMU is implemented as an NS3 application in the C++ environment. Additionally, a helper file is implemented for easy installation of the application on the node containers. To behave like real-world PMUs, we gave the following attributes to each PMU application: (a) PmuID – an Integer ID given to the PMU, (b) PacketSize – the size of the data packets, (c) SamplingRate – a double value sampling rate of the device, (d) Remote – the IP address of the destination to which PMU sends the data packet, (e) Protocol – the type of protocol used by the application, and (f) MaxBytes – the total number of bytes to send.

The algorithm to explain the working of the PMU application is shown in Algorithm 1. The PMU application is installed on the NS3 node using the developed PMU application helper (developing helper files are a prerequisite for developing an NS3). The application helper sets the protocol and remote address for the PMU application. The PMU application has the same start time as the simulation start time. Once the simulation starts, the application creates a socket and schedules the time to send the packet based on the sampling rate. At the scheduled time, a packet is created along with the customized header. Once the packet is created, it is sent to the remote address, and the sending information is logged.

Algorithm 1: PMU Working Mechanism

Input : Sampling rate and YAML file
Output: PMU time-stamped data packet
1 Install PMU app in NS3 node
2 Set up protocol and remote address
3 Initiate application
4 Create socket and schedule sending time based on sampling rate
5 Create packet and send with timestamp

2.1.2 PDC App Implementation in NS3

The PDC application developed in this work has the following attributes: (a) PdcID – an integer ID given to the PDC node, (b) remote address of the destination to which the packet is to be sent, (c) Protocol – the typeId of the protocol to use for the Rx socket, and (d) PdcMap – the map type that holds the frames along with the PMU ID. The PDC application maintains a map (i.e., timestamp map) to store the received packets according to their timestamp. Each element of this timestamp map contains a frame buffer, along with the PMU IDs (i.e., phasor buffer) that contain stream ID and a vector containing the vector measurements with the same timestamp.

The PDC application opens a port to receive the packets sent from the PMU application and waits for a specified time (Δt) before scheduling a packet to be sent to Super PDC. As the packet is received at the PDC node, the PDC application follows the steps as described in Algorithm 2. The Super PDC node works precisely the same as the PDC. The same PDC application works as a Super PDC if no specific remote address is provided to the application. Therefore, the Super PDC node will only receive the data and create a timestamp vector and will not perform any further analysis. However, the functionality can be easily extended, per the requirements.

Algorithm 2: PDC Working Mechanism

```
   Input  : PMU data stream
   Output: Combined time synchronised measurement
1  while PDC App is running do
2      Create a socket if not already.
3      Read time-stamp from packet header.
4      if time-stamp Vector exists then
5          Iterate through the Map of the timeStampVector
6          Compare the TimeStamp of the new packet
             with TimeStamp of the old packet.
7          Iterate through the FrameBuffer.
8          if FrameBuffer has all expected PMU's then
9              Iterate through the time-stamp map element.
10             Create a new empty packet.
11             Get all the float frequencies from all the
                 packet headers and add them up.
12             Add PdcID, timeStamp and totalfreq to
                 header.
13             Attach the header to the packet and send
                 the packet to SuperPDC address.
14         else
15             Wait for the scheduled time for all the
                 expected PMU streams to arrive.
16         end
17     else
18         schedules a new Send with the next timeStamp
             in the Map.
19     end
20 end
```

2.1.3 Synchrophasor Sink App Implementation in NS3

The main aim of the Synchrophasor Sink App is to publish the readings of PMU measurements into the HELICS and the attributes so that historians and GridPACK can use them. The Synchrophasor Sink App instance has the following parameters: (a) HELICS publication pointer – a HELICS publication object retrieved using the publication key, (b) HELICS Publication Key – a string value that contains the publication key, (c) ExpectedStreamId – stores all the expected stream IDs, and (d) ExpectedPmuId – stores all the expected PMU IDs. The Synchrophasor Sink App opens and listens on a port to receive the packets sent from the PMU Application. As the packet is received, the SynchrophasorSink App follows Algorithm 3.

Algorithm 3: Synchrophasor Sink App Working Mechanism

Input : Time stamped data packet from all the PMU's
Output: Published PMU measurement into HELICS

```
Peeks into the Packet header and reads the TimeStamp.
if timeStamp Vector exists then
    it stores the timestamp value.
    Iterate through Expected Stream Id's and Compare
      them with the Id's of present frame.
    if All PMU Id's are present then
    It uses the HELICS publication key and publishes
      the measurements into the HELICS.
    else
        Creates a key to the PMU Id and them
          publishes the measurements using new key.
    end
end
```

2.1.4 Implementation of Historian App in NS3

The main aim of the Historian Application is to record the readings of PMU measurements and print them onto the log file for further references. The Historian App instance has the following attributes: (a) HELICS publication pointer – a HELICS publication object retrieved using the publication key and (b) HELICS Publication Key – a string value that contains the publication. The Historian App opens and listens on a port to receive the packets sent from the PMU Application via HELICS. As the packet is received, the Historian App works as described in Algorithm 4.

Algorithm 4: Historian Working Mechanism

Input : Time stamped data packet from all the PMU's
Output: Log file containing measurements

```
Peeks into the Packet header and reads the TimeStamp.
if timeStampVector exists then
    Stores the timestamp value.
    Iterate through Expected Stream Id's and Compare
      them with the Id's of present frame.
    if All expected Id's are found then
    It opens up the Log file.
    Writes the PMU measurements with timestamp.
    else
        Wait for the scheduled time for all the expected
          PMU stream to arrive.
    end
end
```

2.2 HELICS

HELICS is an open-source, large-scale infrastructure co-simulation engine. HELICS can support much larger-scale co-simulations compared to other co-simulation engines. It integrates an event-driven communication system with a dynamic power system based on time-series data.

The main reason behind using this engine is the feasibility of converging the power system at each time step [23]. It also runs cross-platform and provides different APIs for interacting with other simulators, as well as the networking capabilities to interact with other “federates” on different machines and platforms [18].

2.3 GridPACK

GridPACK is a software framework used for the development of the programs that model power systems. It consists of libraries and components to be used for creating power system topologies. One of the main features is that it runs in parallel and with high-performance computing platforms simplifying parallel grid application development. Application developers only need to be concerned with the physics rather than worrying about distributing data among processors. We can also use standard input files describing grid networks directly into GridPACK to generate models. In this work, the IEEE 39-bus system is modeled using this software framework.

2.4 Graphical User Interface

The web application is an interactive interface built using ReactJS and connects to the back-end (JAVA) and architecture files (C++). The end-user can add multiple projects and visualize the output of the model by running the simulation. Also, a network of nodes (i.e., PMUs and PDCs) is visualized on the NS3 tab. On clicking the node, the right sidebar gives the detailed information of connected edges, ns3::SynchrophasorSinkApp, ns3::PmuHELICSApp applications where the input fields, like sampling rate, HELICSPubNamePrefix, HELICSInputKey, and ExpectedStreamIds, can be tweaked to run the simulation and extract the results.

2.5 Extraction of Data from Yet Another Markup Language (YAML)

We introduce a YAML file that would contain configuration details for a successful run of the co-simulation. In the YAML file, we declare the architecture. Next, we define all the elements in the architecture to be nodes and set up the nodes’ functionalities in the YAML file. We then connect each of the nodes in the architecture using a point-to-point protocol. We then use helper classes, which, in turn, would communicate with the Application classes and start the simulation.

3.0 End-to-End Simulation for Synchrophasor Applications Using Cyber-Physical Co-Simulation

Figure 2 shows the end-to-end simulation of the data flow in the cyber-physical co-simulation platform. In this framework, the phasor data is directly generated using the GridPACK software. The different processes performed under the PMU App are phasor data collection, phasor data reporting, and creating the data packet in the C37.118 synchrophasor standard. Once the data packet is ready, each PMU App in the NS3 network starts reporting the phasor data to the PDC App. The PDC App collects all the streaming data from multiple PMUs, which are configured to the PDC. Once the data from all the configured PMUs is received, the PDC App creates a new data packet and sends it to either Super PDC or Historian, which can be further used by the synchrophasor application to perform analysis.

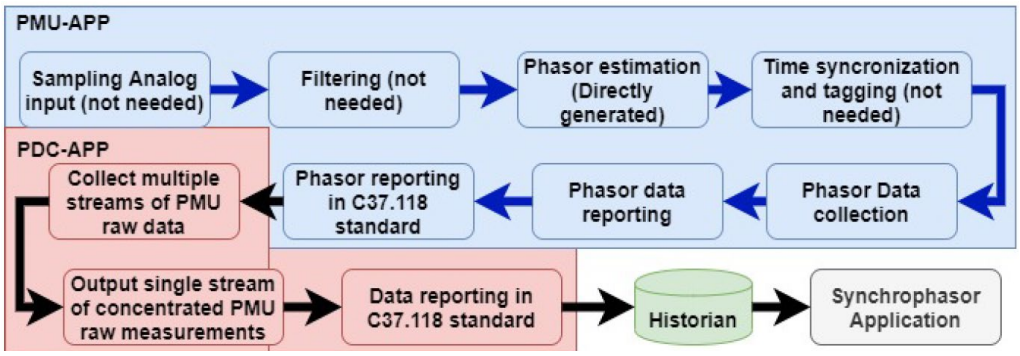

Figure 2. End-to-end Phasor Data Flow in Cyber-physical Co-simulation Platform

4.0 Case Studies

To validate and test the usability of our co-simulation testbed, we created a different NASPInet topology to analyze the network performance in terms of latency and implemented power system stabilizer application to analyze its impact on communication delay.

4.1 Latency Analysis on NASPInet 1 and NASPInet 2

In the first case study, we created a network topology for NASPInet 1 and corresponding modified network topology for NASPInet 2, shown in Figure 3. NASPInet 1.0 consists of PMU, phasor gateway/data bus, PDCs, and Super PDC. After Super PDC receives all the data and combines it according to the time stamp of the measurements, it forwards the data to synchrophasor applications. However, the concept of NASPInet 2 is to forward data directly to synchrophasor applications via generic network nodes and use PDC as a function, not a node. This modified concept of using NASPInet 2 results in reduced latency for end-to-end data exchanges, which are validated and tested in our testbed. Figure 3 represents a NASPInet 1 network with eight PMUs, where four PMUs are connected to one PDC, and the other four PMUs are connected to another PDC. Then, these PDCs are connected with Super PDC, which further forwards data to synchrophasor applications. PDC collects all the timestamped data from PMUs and waits for a specific time until all the expected data streams from all the PMUs are received. After that, it adds all the timestamped data into a PDC data packet and forwards that to the next PDC/Super PDC node to be fed into synchrophasor applications. This proposed PDC stacking architecture in NASPInet 1 needs to be re-evaluated for present-day low latency required applications. Due to PDC stacking and the PDC working mechanism, it adds some extra latency in every PDC point.

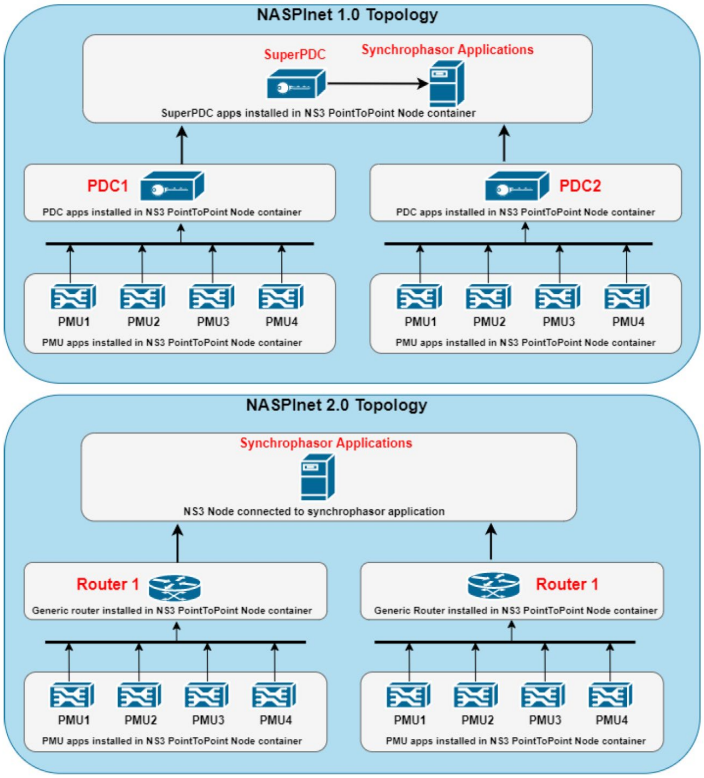

Figure 3. Different NASPInet Topology

In NASPInet 2, there has been significant research to form the network architecture without introducing PDC into the network as a node and forwarding data directly from PMUs to synchrophasor applications. In this case study, the same number of PMUs and similar network configurations (i.e., delay, bandwidth and topology) are used for both the network topology. We have used the NS3 Point-to-Point helper to create nodes. We assume the links to be optical fiber with a 1.2-ms delay and 100-Mbps data rate 25 km distance between the nodes. In NASPInet 2 network, we have used network nodes such as a router instead of PDCs to forward data packets directly to synchrophasor applications (omitting the Super PDC node). We have calculated end-to-end delay for both networks for different PMU sampling rates.

As seen in the results of Table 1, replacing PDC node by network router in synchrophasor network has a significant impact on latency. From this simple base network and single PDC stacking, we can see almost 2 ms of reduced delay for each sampling rate of PMU. This is happening as PDC waits for some time until it receives all the PMU packets and then combines them to create new packets before sending; whereas the router only reads the packet header to forward to the next destination based on the IP address. Our testbed can be used for further complex network models and see the latency differences in both NASPInet 1 and NASPInet 2.

Table 1. Performance Analysis on NASPInet 1 and NASPInet 2

Sampling Rate	Avg. Throughput from PMUs	Avg Latency in NASPInet 1	Avg Latency in NASPInet 2
30 Hz	25.10 Mbps	7.51 ms	5.92 ms
60 Hz	50.91 Mbps	7.67 ms	5.97 ms
90 Hz	70.34 Mbps	7.74 ms	6.16 ms
120 Hz	106.19 Mbps	7.96 ms	6.35 ms

4.2 Communication Delay on Wide-Area Power System Stabilizers (PSSs) in a 39-Bus System

Damping controllers have been used over the years to mitigate power system oscillations. PSSs are damping controllers used as a supplementary feedback control loop to the automatic voltage regulators in order to improve damping in power systems. While local PSSs are effective in damping the local-area oscillation modes, they lack global observation and are not effective against inter-area modes. It has been proven that an inter-area mode may be controllable from one area and be observable from another area. Therefore, a wide-area PSS offers great potential in complementing the limitation of conventional local PSS by exploiting wide-area PMU measurements. The wide-area PSS takes PMU measurements, such as the bus frequencies, at different locations as inputs. Its output is used as additional feedback to the excitation system of the generator the wide-area PSS is associated with.

Since the wide-area PSS rely on measurements from a different location, its performance can be affected by communication imperfections, such as latency, packet loss, congestion, and more. In this case study, we investigate the impact of communication delay caused by the PDCs on the performance of wide-area PSS. A modified IEEE 39-bus system is developed with a wide-area PSS located at Bus 34, and the PMUs at Buses 30 and 34, as shown in Figure 4. We assume a fault occurs at t = 1s on the line connecting Buses 16 and 17 and evaluate the performance of the wide-area PSS over NASPInet 1 and 2, respectively. The generator speed output of the generator at Bus 34 is shown in Figure 5. Due to communication latency in

NASPInet 1 architecture, the generator speed went uncontrolled, whereas the speed is damped to its nominal value for NASPInet 2 architecture after the fault.

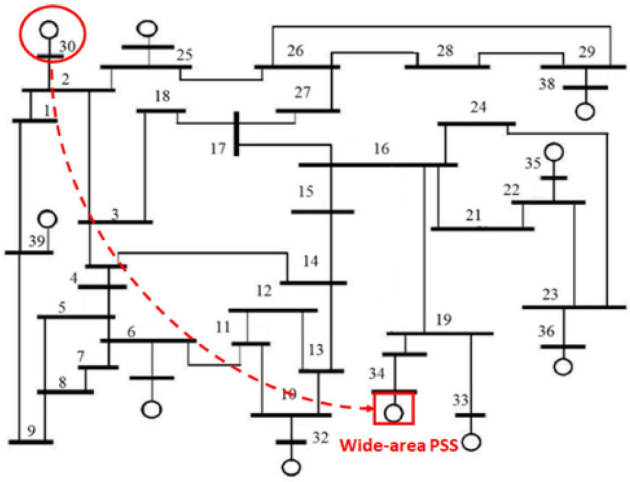

Figure 4. PSS on IEEE 39-bus

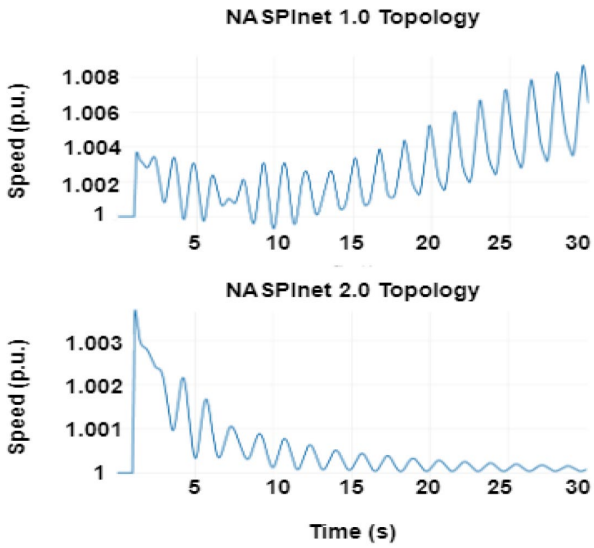

Figure 5. PSS Controller Output for NASPInet 1 and NASPInet 2 Topologies

5.0 Conclusions

This paper presents a cyber-physical, co-simulation testbed based on NS3, HELICS, GridPACK, and User Interface, specifically to research synchrophasor networks and show how adopting different network technologies and architecture impacts the real-time PMU-based applications. An integrated user interface makes it easier for other researchers to play with the network topology to see the implications and visualize the results. Developed PMU and PDC functionality in NS3 makes it possible to model more complex and extensive networks in NS3, while capturing real-time grid behavior in synchrophasor networks.

A comparative performance analysis based on latency in NASPInet 1 and NASPInet 2 architecture has been done, and applications like PSS have been implemented to visualize the impacts generated from synchrophasor networks' choice. In the future, we will integrate the DNP3 protocol in NS3 to capture the transactive market model and micro-PMUs in NS3 to model distribution networks in this testbed. We will open-source the codes and docker images to be used by the community.

6.0 References

[1] P. Kansal and A. Bose, “Bandwidth and latency requirements for smart transmission grid applications,” IEEE Transactions on Smart Grid, vol. 3, no. 3, pp. 1344–1352, 2012.

[2] F. Ye and A. Bose, “Multiple communication topologies for PMU-based applications: Introduction, analysis and simulation,” IEEE Transactions on Smart Grid, vol. 11, no. 6, pp. 5051–5061, 2020.

[3] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, “A survey on smart grid potential applications and communication requirements,” IEEE Transactions on industrial informatics, vol. 9, no. 1, pp. 28–42, 2012.

[4] Assessment of existing synchrophasor networks, April 2018. [Online]. Available: https://www.osti.gov/servlets/purl/1523382

[5] M. Chenine, K. Zhu, and L. Nordstrom, “Survey on priorities and communication requirements for PMU-based applications in the Nordic region,” in 2009 IEEE Bucharest PowerTech. IEEE, 2009, pp. 1–8.

[6] NASPInet 2.0 architecture guidance, version 1.19. [Online]. Available: https://gridarchitecture.pnnl.gov/media/NASPInet%202%20v1.19PNNL.pdf

[7] P. T. Myrda and K. Koellner, “NASPInet – the internet for synchrophasors,” in 2010 43rd Hawaii International Conference on System Sciences, 2010, pp. 1–6.

[8] M. Chenine, E. Karam, and L. Nordstrom, “Modeling and simulation of wide area monitoring and control systems in IP-based networks,” in 2009 IEEE Power Energy Society General Meeting, 2009, pp. 1–8.

[9] Y. Deng, H. Lin, A. G. Phadke, S. Shukla, J. S. Thorp, and L. Mili, “Communication network modeling and simulation for wide area measurement applications,” in 2012 IEEE PES Innovative Smart Grid Technologies (ISGT), 2012, pp. 1–6.

[10] M. Chenine and L. Nordstrom, “Modeling and simulation of wide-area communication for centralized PMU-based applications,” IEEE Transactions on Power Delivery, vol. 26, no. 3, pp. 1372–1380, 2011.

[11] M. Chenine and L. Nordstrom, “Investigation of communication delays and data incompleteness in multi-PMU wide area monitoring and control systems,” in 2009 International Conference on Electric Power and Energy Conversion Systems (EPECS), 2009, pp. 1–6.

[12] D. Bhor, K. Angappan, and K. M. Sivalingam, “A co-simulation framework for smart grid wide-area monitoring networks,” in 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS), 2014, pp. 1–8.

[13] D. R. Gurusinghe, S. Menike, A. I. Konara, A. D. Rajapakse, P. Yahampath, U. D. Annakkage, B. A. Archer, and T. Weekes, “Co-simulation of Power System and Synchrophasor Communication Network on a Single Simulation Platform,” Technology and Economics of Smart Grids and Sustainable Energy, vol. 1, no. 1, p. 6, Mar. 2016. [Online]. Available: https://doi.org/10.1007/s40866-015-0003-9

[14] D. Bhor, K. Angappan, and K. M. Sivalingam, “Network and power grid co-simulation framework for smart grid wide-area monitoring networks,” Journal of Network and Computer Applications, vol. 59, pp. 274–284, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804515001447

[15] H. A. Tokel, R. Al Halaseh, G. Alirezaei, and R. Mathar, “A co-simulation framework for integrated planning and analysis of wide area measurement and protection systems,” Journal of Communications Software and Systems, vol. 14, no. 1, pp. 40–50, 2018.

[16] J. S. Carson, “Proceedings of the 2003 winter simulation conference: Volume 1,” Winter Simulation Conference Proceedings, vol. 1, no. 1993, pp. 1656–1662, 2003.

[17] J. Nutaro, P. T. Kuruganti, L. Miller, S. Mullen, and M. Shankar, “Integrated hybrid-simulation of electric power and communications systems,” in 2007 IEEE Power Engineering Society General Meeting, 2007, pp. 1–8.

[18] HELICS. [Online]. Available: https://github.com/GMLC-TDC/HELICS

[19] IEEE standard for synchrophasor data transfer for power systems. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6111222

[20] Phasor data concentrator. [Online]. Available: https://openei.org/wiki/Definition:Phasor_Data_Concentrator_(PDC)

[21] Wikipedia. Phasor measurement unit. [Online]. Available: https://en.wikipedia.org/wiki/Phasor_measurement_unit

[22] Nsnam. Network simulator 3. [Online]. Available: https://www.nsnam.org/about/

[23] B. Palmintier, D. Krishnamurthy, P. Top, S. Smith, J. Daily, and J. Fuller, “Design of the HELICS high-performance transmission-distribution communication-market co-simulation framework,” in 2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2017, pp. 1–6.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354
1-888-375-PNNL (7665)

www.pnnl.gov