Cyber-Power Co-Simulation for End-to-End Synchrophasor Network Analysis and Applications

Hussain M. Mustafa, *Student Member, IEEE*, Dexin Wang, *Member, IEEE*, K. S. Sajan, *Member, IEEE*, Eshwar Nag Pilli, Renke Huang, *Member, IEEE*, Anurag K. Srivastava, *Senior Member, IEEE*, Jianming Lian, *Senior Member, IEEE*, Zhenyu Huang, *Fellow, IEEE*

***Abstract*—The resiliency, reliability and security of the next generation cyber-power smart grid depend upon efficiently leveraging advanced communication and computing technologies. Also, developing real-time data-driven applications is critical to enable wide-area monitoring and control of the cyber-power grid given high-resolution data from Phasor Measurement Units (PMUs). North American Synchrophasor Initiative Network (NASPInet) provides guidance for PMU data exchanges. With the advancement in networking and grid operation, it is necessary to evaluate the performance of different data flow architectures suggested by NASPInet and analyze the 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 an end-to-end automated and user-driven cyber-power co-simulation using NS3 to model communication networks, 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 utilizing PMU data in an IEEE 39 bus test system is presented using this co-simulation testbed.**

***Index Terms*—Phasor Measurement Units, Wide-area Monitoring, Co-Simulation, Cyber-power Systems.**

I. Introduction

POWER system, which is evolving as a smart grid, requires many real-time applications to be developed and run to provide better insights 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 PMUs 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, therefore, creating the opportunity to develop many real-time applications and implement them in the smart grid [2]. Most of these applications will be effective in real-time scenarios if they meet their stringent latency requirement of few milliseconds to several seconds [3]. 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 [4]. Given the importance of the communication network, in 2007 North American SynchroPhasor Initiative (NASPI) started to develop a sustainable framework for the design of synchrophasor data communication networks (NASPInet) [5]. Since then, the NASPInet framework is treated as the standardized communication network guideline for synchrophasor applications. NASPInet is generally formed with the integration of PMU's, phasor data concentrator (PDC), Phasor Gateways connected to a data bus, and a centralized phasor data concentrator called SuperPDC [6]. But since the development of NASPInet, several things have changed significantly, such as increased data volume, network technology, protocol advancement, etc. Although most of the NASPInet design concepts are useful, there has been a lot of initiative taken in the last several years revisiting the previous architecture to overcome the operational, maintenance, and implementation limitations to help preparing for the NASPInet 2 framework [5]. Previously, the output of the PMU based applications depended on precise synchronization and a lower packet loss ratio. However, for current and future applications, low communication latency is equally important [5]. Some previous work on the performance evaluation of wide-area synchrophasor networks using simulation appears in [7]–[9]. All these communication performance requirements make the synchrophasor network an essential component, considering the PMU based applications are crucial for power system operation. Therefore, the use of PMU in the future for different real-time monitoring and control applications or some resiliency applications such as CP-SAM [10] necessitates the re-evaluation of previous NASPInet architecture to a newer NASPInet 2 with analysis and comparison of different network performance issues. This re-evaluation of the synchrophasor network and its impact on various applications need to be tested and validated in the simulated platform, which exhibits similar behavior to real systems before deploying in the real environment. Therefore, researchers need to have a scalable multi-featured co-simulation platform, and emphasis should be given in this area of research.

In the past, there has been some work on developing co-simulation testbed aimed to research and validate PMU networks and PMU based applications [11], [12]. For co-simulation engine, EPOCHS [13], ADEVS [14] has been used. We focused on the main challenges in developing a co-simulation framework found from the works of literature, such as generating a communication network able to emulate real behavior, adopting industry standards, interfacing between cyber and power to synchronize time steps, and end to end synchrophasor data exchange capabilities. To overcome

We are grateful for the inspiration and guidance of Dr. Jeffrey Taft and James Ogle at Pacific Northwest National Laboratory. This work would not have been possible without their invaluable knowledge and insights on synchrophasor networking and related applications.

This work is supported by the United States Department of Energy Pacific Northwest National Laboratory's Laboratory Directed Research and Development Program through its joint Advanced Grid Institute with Washington State University. Authors would also like to acknowledge partial support from the UI-ASSIST project for this work. We would like to thank Anshuman and Nikhil for their technical support.

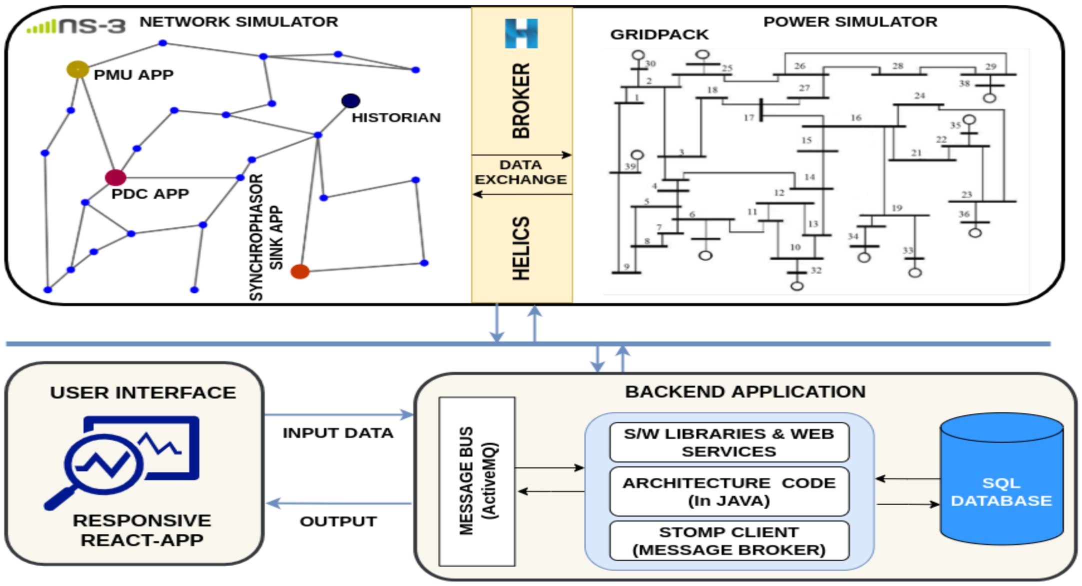


Fig. 1: Cyber-power co-simulation architecture

these challenges, with the objective of building a ***Customized User-driven web-based Cyber-Power Co-Simulation***, we have developed different synchrophasor components in NS3 using C++ capable of exchanging data following IEEE C37.118 protocol (used for PMU data exchange) [15] 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 [16], a powerful and scalable co-simulation engine for time-synchronized power-cyber interfacing. To visualize the results, we have integrated one web-based user interface built using reactJS. This testbed can analyze the impact of communication network performance on various power system control applications and compare different network performance issues for NASPInet 1 and NASPInet 2. IEEE 39 bus 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 app and Historian app 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-power co-simulation testbed, is built where NS3 and GridPACK communicate via HELICS.
- Developed a controllable graphical user interface (GUI) from which users can create different synchrophasor networks and visualize their impact on different synchrophasor applications.

II. 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 application based on this fast timestamp-based measurement has already been developed, and in future, this trend will go on to 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. PMU is a device used to estimate an electrical measurement's magnitude and phase angles 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. This facilitates synchronized real-time wide-area monitoring of the grid.

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. 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.
- **GUI:** An user interface to modify the synchrophasor network and visualize results.

A. *NS3*

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 [17]. 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 on 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 NS3 nodes to emulate end-to-end synchrophasor networks similar to the real system. The network functions developed include:

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

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 be able to behave like real-world PMU's we have given 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 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.

Algorithm 1: PMU Working Mechanism

```
Input  : Sampling rate and remote address
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) 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 destination to which the packet is to be sent, (c) Protocol - The type id of the protocol to use for the Rx socket, and (d) PdcMap - It is a 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. The PDC application opens a port to listen to receive the packets sent from the PMU application and waits for a specified time (Δt) before scheduling a packet to be sent to the SuperPDC. As the packet is received at the PDC node, the PDC application follows the steps as described in algorithm 2.

The superPDC node works precisely the same as PDC. The same PDC application works as a superPDC if no specific remote address is provided to the application. Therefore, the superPDC 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 as per the requirements.

3) SynchrophasorSink App Implementation in NS3: The main aim of the SynchrophasorSink Application is to publish the readings of PMU measurements into the HELICS and the attributes so that historians and GrisPACK can use them. The SynchrophasorSink App instance has the following parameters: (a) HELICS publication pointer - It is a Helics publication object retrieved using the publication key, (b) HELICS Publication Key - It is a string value that contains the publication

Algorithm 2: PDC Working Mechanism

```
Input  : PMU data stream
Output: Combined time synchronized 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             Combine data from all the PMUs.
12             Add PdcID, timeStamp and data 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
```

key, (c) ExpectedStreamId - Stores all the Expected stream Id's, and (d) ExpectedPmuId - Stores all the Expected PMU Id's. As the packet is received, the SynchrophasorSink App follows the 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
1  Peeks into the Packet header and reads the TimeStamp.
2  if timeStamp Vector exists then
3      it stores the timestamp value.
4      Iterate through Expected Stream Id's and Compare
         them with the Id's of present frame.
5      if All PMU Id's are present then
6      It uses the HELICS publication key and publishes
         the measurements into the HELICS.
7      else
8          Creates a key to the PMU Id and them
             publishes the measurements using new key.
9      end
10 end
```

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 - It is a Helics publication object retrieved using the publication key and (b) Helics Publication Key - It is a string value that contains the publication. As the packet is recieved, 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
1  Peeks into the Packet header and reads the TimeStamp.
2  if timeStampVector exists then
3      Stores the timestamp value.
4      Iterate through Expected Stream Id's and Compare
         them with the Id's of present frame.
5      if All expected Id's are found then
6      It opens up the Log file.
7      Writes the PMU measurements with timestamp.
8      else
9          Wait for the scheduled time for all the expected
             PMU stream to arrive.
10     end
11 end
```

B. HELICS

HELICS is an open-source large-scale infrastructure co-simulation Engine. HELICS can support much larger scale co-simulations comparing 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 is the feasibility of converging power system at each time step [18]. It also runs cross-platform, provides different APIs for interacting with other simulators, and the networking capabilities to interact with other federates on different machines and platforms. [16]

C. GridPACK

GridPACK is a software framework that is used for the development of the programs that model power systems. It consists of libraries and components to be used for creating power system topology. One of the main features is that it runs on parallel and high-performance computing platforms simplifying parallel gird application development [19]. 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, IEEE 39 bus system is modeled using this software framework.

D. Graphical user Interface

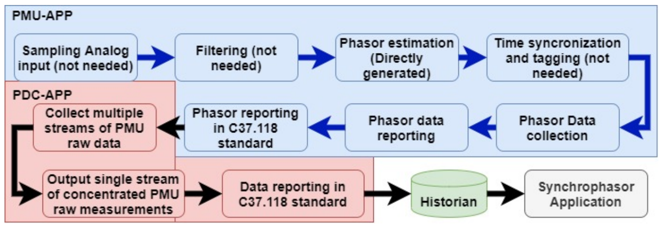
The web application is an interactive interface built using ReactJS. It connects to the back-end (JAVA) with the help of a STOMP client, which communicates with the messaging broker shown in figure 1. End-users can add multiple projects and visualize the output of the model through simulation. The network of nodes, i.e., PMUs and PDCs, is visualized on the NS3 tab. The right sidebar gives the detailed information of application nodes where the input field like sampling rate, HelicsPubNamePrefix, HelicsInputKey, and Expected-StreamIds can be tweaked to run the simulation, extract the results, and visualize it in the form of a graph.

E. Extraction of data from 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.

III. End-to-end simulation for Synchrophasor system using Cyber-Power co-simulation

Figure 2 shows the end-to-end simulation of the data flow in the cyber-power co-simulation platform. In this framework, the phasor data is directly generated using the IEEE 39 bus system modeled in GridPACK software. Once the data is generated, it enters the NS3 communication network using the HELICS co-simulation engine by synchronizing the time step. The PMU app functionality developed in NS3 receives the packet and creates a C37.118 packet structure to exchange data between PMU and PDC. Once the data packet 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 PDC. Once the data is received from all the configured PMUs, the PDC app creates a new data packet to send to SuperPDC. Finally, when SuperPDC processes the data in NS3, data is forwarded to synchrophasor sink app is developed in NS3 and it publishes the data using HELICS publication key and pointer to the power system stabilizer(PSS) application developed in GridPACK to perform analysis again synchronizing the time step, completing end-to-end data exchange through co-simulation. The Historian application in NS3 stores all the data values in the log file using the timestamp during the process. The whole co-simulation is made user-friendly and avoids the complexity of back-end programming by developing a user interface that has the options in the drop-down menu to select the co-simulation parameters and network topology and start the co-simulation to visualize results for a different combination of variables.



PMU-APP
Sampling Analog input (not needed)
Filtering (not needed)
Phasor estimation (Directly generated)
Time syncronization and tagging (not needed)
PDC-APP
Collect multiple streams of PMU raw data
Phasor reporting in C37.118 standard
Phasor data reporting
Phasor Data collection
Output single stream of concentrated PMU raw measurements
Data reporting in C37.118 standard
Historian
Synchrophasor Application

Fig. 2: End-to-End phasor data flow in cyber-power co-simulation platform

IV. Case Study

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

A. Latency analysis on NASPInet 1 and NASPInet 2

In case study 1, we have created a network topology for both NASPinet 1 and NASPInet 2, shown in fig. 3. NASPInet1.0 consists of PMU, phasor gateway/data bus, PDC's, and SuperPDC. In NASPInet 1 superPDC receives all the data and combines it according to the measurements' time-stamp, then forwards it to the synchrophasor applications.

TABLE I: Performance analysis on NASPInet 1 and NASPInet 2

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

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 reduces latency for end-to-end data exchanges, which is validated and tested in our testbed. Fig. 3 represents a NASPInet 1 network with 8 PMUs where 4 PMU's are connected to one PDC, and the other 4 PMUs are connected to another PDC. Then these PDC's are connected with SuperPDC, which further forwards data to synchrophasor applications. PDC collects all the time-stamped data from PMU's and waits for a specific time, Δt until it receives all the expected data streams from all the PMU's. After that, it adds all the time-stamped data into a PDC data packet and forwards that to the next PDC/SuperPDC 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 PDC working mechanism, it adds some extra latency in every PDC point.

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 PMU's and similar network configurations (i.e., delay, bandwidth, and topology) are used for both the network topology. We have used the NS3 PointToPoint helper to create nodes. We assume the links to be Band-b(G.652) optical fiber with 1310nm wavelength with 2000Mbps data rate of 1.2ms delay when they are 25km distance apart. 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 SuperPDC node). To see the problems related to PDC stacking, we have used different delays in some of the links marked in color shown in 3. As PDC waits for all the PMU data to be received and then proceed for processing, a delay in one PMU link causes an overall delay for the PDC to process the data for all PMUs. As routers don't wait for all the PMU data to arrive while forwarding it to synchrophasor applications, NASPInet 2 doesn't go through this extra wait time delay. In our case, we have calculated end-to-end delay for both networks for different PMU sampling rates and reported average delay for both networks under mentioned attributes above. It is seen from the results of table I replacing PDC node by network router in synchrophasor network has a significant impact on latency even in this simple and straightforward topology. 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 happens as PDC waits for some time until it receives all the PMU packets and then combines them to create new packets before sending wheres the router only reads the packet header to forward to the next destination based on the IP address. This extra latency can impact applications with really stringent latency requirements for data to receive. Our testbed can be used for further complex network models and see the latency differences in NASPInet 1 and NASPInet 2.

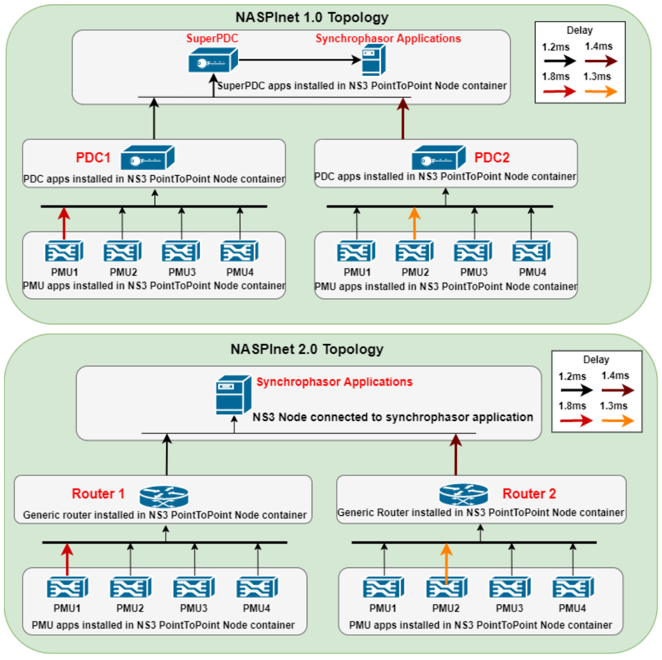

NASPInet 1.0 Topology
SuperPDC
Synchrophasor Applications
SuperPDC apps installed in NS3 PointToPoint Node container
Delay
1.2ms
1.4ms
1.8ms
1.3ms
PDC1
PDC apps installed in NS3 PointToPoint Node container
PDC2
PDC apps installed in NS3 PointToPoint Node container
PMU1
PMU2
PMU3
PMU4
PMU apps installed in NS3 PointToPoint Node container
PMU1
PMU2
PMU3
PMU4
PMU apps installed in NS3 PointToPoint Node container
NASPInet 2.0 Topology
Synchrophasor Applications
NS3 Node connected to synchrophasor application
Delay
1.2ms
1.4ms
1.8ms
1.3ms
Router 1
Generic router installed in NS3 PointToPoint Node container
Router 2
Generic Router installed in NS3 PointToPoint Node container
PMU1
PMU2
PMU3
PMU4
PMU apps installed in NS3 PointToPoint Node container
PMU1
PMU2
PMU3
PMU4
PMU apps installed in NS3 PointToPoint Node container

Fig. 3: Different NASPInet topology.

B. communication delay on wide-area-measurement-based PSS controls of a 39-bus system

Damping controllers have been used over the years to mitigate power system oscillations. Power system stabilizers (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, however, they lack global observation and are not effective against inter-area modes. It has been proved that an inter-area mode may be controllable from one area and be observable from another area. Therefore, 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, it's performance can be affected by communication imperfections, such as latency, packet loss, congestion etc. In this case study, we investigate the impact of communi-

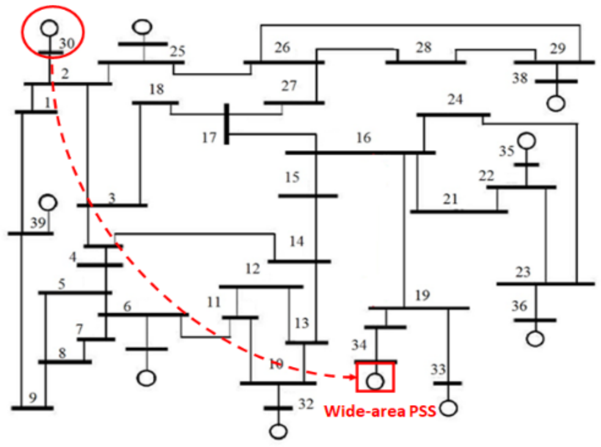


Fig. 4: PSS on IEEE39 bus

cation 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 Fig. 4. We assume a fault occurs at $t = 1$s on the line connecting Buses 16 and 17 and evaluate the performance of the wide-area PSS over NASPInet 1 and 2, respectively. The speed of the generator at Bus 34 is shown in figure 5. Due to communication latency in NASPInet 1 architecture, the generator speed oscillation is poorly damped and become unstable, whereas with NASPInet 2 architecture the same is well damped to its nominal value after the fault.

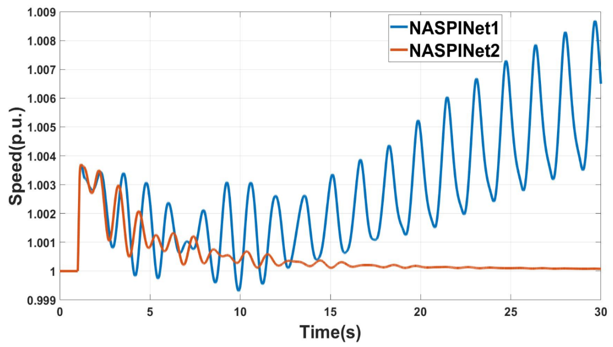


Fig. 5: Generator speed with wide-area PSS under NASPInet 1 and NASPInet 2

V. Conclusions

This paper presents a customized user-driven web-based cyber-power co-simulation based on NS3, HELICS, Grid-PACK, and a tailored user interface, specifically to research synchrophasor networks and shows how adopting different network technologies and architectures impacts the real-time PMU based applications. The integration of synchrophasor functionality in NS3 and using HELICS helps overcome the challenges related to network simulation with industry-standard adoption and cyber-power interfacing with time synchronization. An integrated user interface makes it easier for other researchers to play with the network topology to see the implications and visualize the results. The 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 1.0 and NASPInet 2.0 architecture has been done and used to demonstrate the functionality and validity of the developed cyber-power co-simulation platform. Applications like PSS have been implemented to visualize the impacts generated from synchrophasor networks' choices.
In the future, we will integrate the DNP3 protocol in NS3 to capture the transactive market model and Micro-PMUs in NS3 with the distribution network model in this testbed, with continuing efforts to automate the setup and configuration of these complex simulation tasks. We will open-source the codes and docker images to be used by the community.

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

[4] 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.

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

[6] P. T. Myrda and K. Koellner, "Naspinet - the internet for synchrophasors," in *2010 43rd Hawaii International Conference on System Sciences*, 2010, pp. 1–6.

[7] 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.

[8] 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.

[9] M. Chenine and L. Nordström, "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.

[10] P. S. Sarker, A. S. Saini, K. Sajan, and A. K. Srivastava, "Cp-sam: Cyber-power security assessment and resiliency analysis tool for distribution system," in *2020 Resilience Week (RWS)*. IEEE, 2020, pp. 188–193.

[11] 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.

[12] 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.

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

[14] 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.

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

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

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

[18] 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.

[19] Gridpack. [Online]. Available: https://www.gridpack.org/wiki/index.php/Main_Page