DELAY ANALYSIS OF A REAL-TIME HARD RECONFIGURABLE SYNCHROPHASOR SYNCHRONIZATION GATEWAY

Prottay M. Adhikari
Electrical Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY, USA
email: mondap2@rpi.edu

Luigi Vanfretti
Electrical Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY, USA
email: luigi.vanfretti@gmail.com

ABSTRACT

Phasor Data Concentrators (PDC) receive and time-synchronize phasor data from multiple phasor measurement units (PMUs) to produce a real-time, time-aligned output data stream. PDCs are expected to handle large sets of data and may consume substantial hardware resources in terms of memory. This paper presents some preliminary results towards the development of a hard real-time storageless Synchrophasor Gateway based on National Instruments' Compact Reconfigurable Input-Output (cRIO) hardware platform. It utilizes the Khorjin library [1] which is able to receive and parse synchrophasor data from a PMU/PDC based on IEEE C37.118.2 protocol. A fully functional PDC is expected to store and publish PMU data. The proposed real-time hardware prototype, however does not store data, as its goal is to provide essential synchronization and aggregation functions to be used by protection and control devices. Hence, this prototype is described as a Synchrophasor Synchronization Gateway (SSG) instead of a PDC. In this paper, elementary tests related to timing, delay and reliability are performed on the SSG, and the results along with the observed issues are reported.

KEY WORDS

Phasor Data Concentrator (PDC), Phasor Measurement Unit (PMU), Synchrophasor Gateway, Wide Area Measurement System (WAMS), Wide Area Control System (WACS), Wide Area Monitoring Protection and Control (WAMPAC).

1 Introduction

1.1 Motivation

Phasor Data Concentrators (PDC) are integral part of modern synchrophasor measurement systems. PDCs are expected to receive, parse, allign, store and publish measurement data from field PMUs. They are expected to be compatible with synchrophasor transmission protocols such as IEEE C37.118.1/ 2 and IEC 61850-90-5. However, existing PDC hardware architectures, fail to comply with hard real-time control requirements. As reported in literature [2-4], most implementations are purely on the software level, and needs extensive testing in terms of real time performance. In [8] the authors have proposed a similar platform for WACS applications, for a single PMU stream. This paper reports implementation and testing of a synchrophasor synchronization gateway with multiple concurrent PMU/PDC streams, which makes it an attractive infrastructure to be used in WACS and WAMPAC applications.

This paper reports a prototype Synchrophasor Synchronization Gateway (SSG) implemented in both **real-time software** and **real-time hardware** which performs the most essential functionalities of an industrial PDC, except data-publishing and storage as the main goal is to provide synchronization and aggregation services for the real-time controls and protection.

The implementation is carried out based on the compact reconfigurable input-output (cRIO) devices. The underlying cRIO hardware is configurable by a graphical interface designed in the LabVIEW environment. This architecture is user friendly in terms of configuration, display and hardware management.

1.2 Related Works

Because, the proposed SSG performs the tasks traditionally performed by a PDC, the literature review mostly consists of past research that dealt with PDC implementations. The work in this paper extends the library presented in [1] for unwrapping PMU data in the proposed prototype SSG. The functions of this library are written in C and compiled into a dynamically linked library using National Instruments' CVI infrastructure. Concerning PDC design, [3] presented important proposals in terms of standardization of the measurement architecture. It also dealt with the problems of accurate time synchronization and management of data-loss scenarios. Even though this work was crucial in terms of standardization, it did not propose any feasible hardware implementation that can meet hard real-time control requirements at the sub-second level. These standardization efforts were further extended by the authors in [4]. The results discussed in [4] are used in this work to compare the work presented in this paper with the state of the art. Authors in [2] experimented with Raspberry Pi based hardware architecture for PDC implementation. Their work was successful in implementing a functional PDC, but the performance analysis in terms of time-synchronization, latency, reliability were not reported. [5] presented compar-

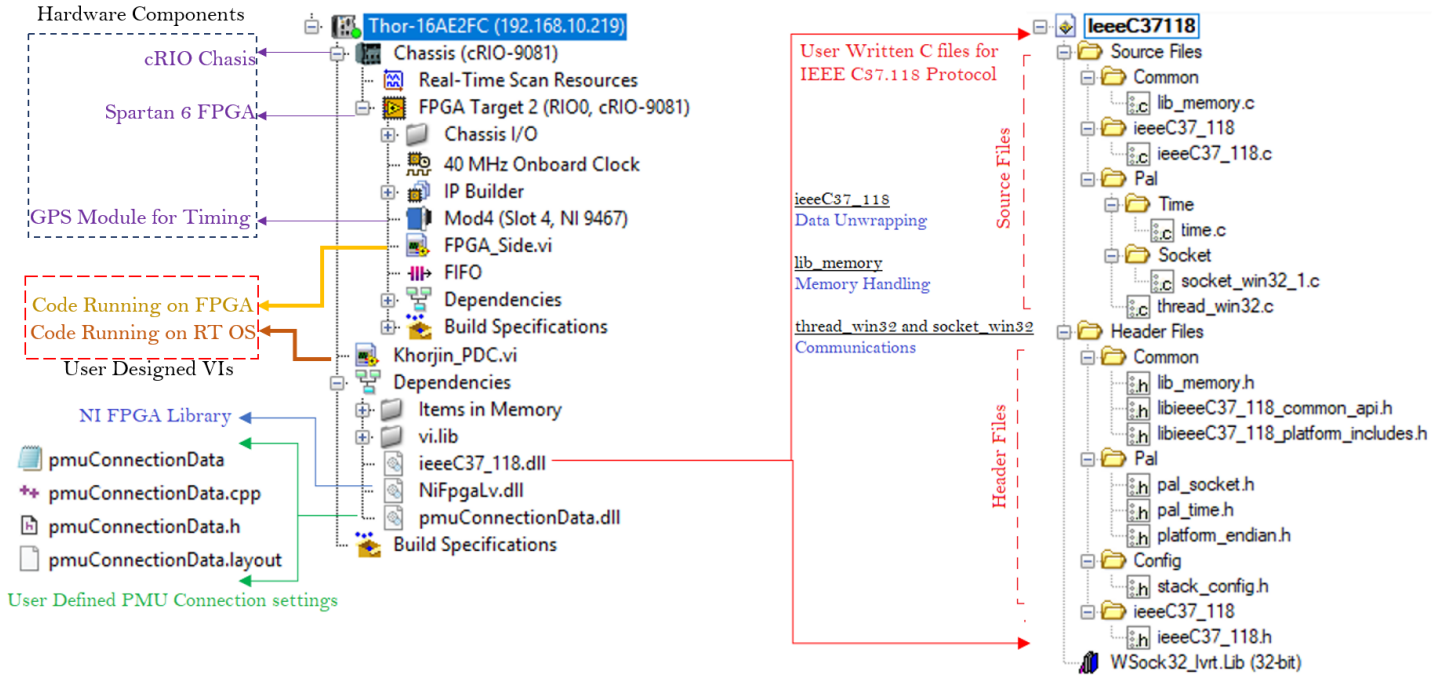

Figure 1: An Overview of the Hardware and Software Components for the PDC Implementation

isons between the existing open source PDC software systems. The authors in [6] proposed and compared PDC network architectures in a real life industrial scale networks in Switzerland and Netherlands.

1.3 Contributions of this Paper

- The application and extension of a C-based library to receive and parse synchrophasor data from a PMU/PDC to construct multiple receiver threads of the proposed prototype SSG.
- Implementation and testing of the proposed SSG prototype using multiple concurrent PMU streams, and characterizing its reliability and performance.
- Stress-testing of the SSG prototype by introducing network traffic and characterizing its performance under varying network conditions.

2 Proposed Real-time Hardware & Software Architecture

This section describes the components in the hardware and software layers of the proposed SSG prototype shown in Fig. 1. Additionally, the laboratory-setup and equipment used for testing the prototype are also introduced.

2.1 Software Components

The software architecture utilizes National Instruments' virtual instrumentation (VI) infrastructure. In general, there are two top level VIs. One of them runs on the real time processor, and the other one runs on a Spartan 6 FPGA inside the cRIO chassis. The VI that runs on the real time processor (Khorjin_PDC.vi) has all the functionalities for networking, parsing and processing. These functionalities are used to connect with different PMUs and to unwrap the PMU data being received. The functionalities are encoded in standard C, and were converted into a dynamic library using National Instruments Windows CVI tool-chain. The source code are also tested with Microsoft Visual Studio and Eclipse, however the use of CVI tool-chain is the most reliable and was ultimately recommended by NI. An additional cpp file was written to configure the connection settings for each of the PMUs that communicate with the gateway. This file, along with an appropriate header, were converted into dynamic libraries, which is used by the SSG VI running on the RT processor of the compactRIO hardware. The VI running on the FPGA communicates with the C series NI 9467 GPS module and provides the time-stamp which will enable the delay compuations in this paper.

2.2 Hardware Components

In addition to the software components described above, the SSG has appropriate physical hardware to enable the functionalities of those software components. The two major components of the hardware are (a) RT processor Intel

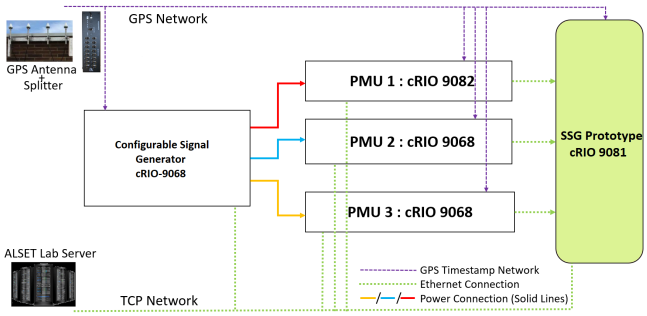

Figure 2: Block Diagram of the Laboratory Experimental Setup

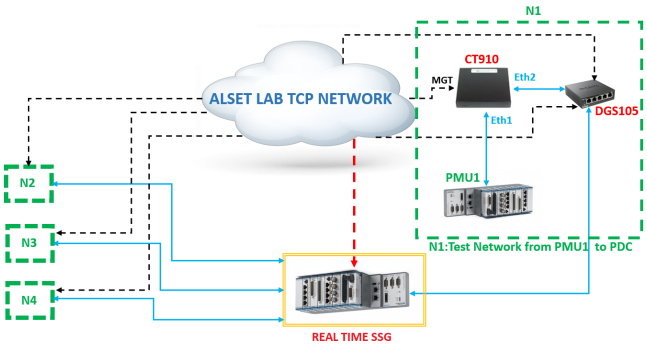

Figure 3: Networking Infrastructure incorporating the CT-910 Network Traffic Generator

Celeron U3405 of the cRIO 9081 chassis and (b) Xilinx Spartan 6 LX 75 FPGA. The RT processor runs Microsoft WES 7 Runtime OS. All the C37.118 functionalities along with the TCP communication interfaces run on this part of the hardware. On the other hand, the Spartan 6 FPGA interacts with the NI 9467 GPS module to acquire the latest time-stamp, and sends this time-stamp to the WES7 OS running on the real time processor for further use.

3 Laboratory Experimental Setup

To analyse the performance of any prototype PDC/SSG, more than one PMU is required. In this case, the PMUs used were implemented on three separate compact RIOs (one cRIO 9082 and two cRIO 9068). The basic design of these PMUs were obtained from National Instruments 'Advanced PMU Development System' [9] . Those designs were compiled and synthesized on NI's high performance computing server using Xilinx ISE synthesis toolchain. As seen from the diagram in Fig. 2, the TCP network is used to interface these PMUs run on the real time processors of the cRIO devices.

For delay analysis purposes, the SSG implementation requires to receive an accurate time-stamp from the individual PMUs which are part of the networked stream of each device; and most importantly from the GPS antenna through the NI 9467 GPS synchronizer. The SSG communication functions run on the real time processor of the cRIO 9081, while the Spartan 6 FPGA inside the cRIO runs the program to receive the GPS data. This part of the hardware also acquires accurate GPS time reference from the antennas as shown in the block diagram in Fig 2. GPS antennas also provide the 3 PMUs running on the cRIO 9082 and cRIO 9068 devices with their own time references via three NI 9467 GPS acquisition modules.

All the cRIOs are connected to the same TCP/IP network of the laboratory. The PMUs and the SSG running on the cRIO devices are expected to be discoverable by one another, since they are under the same subnet mask. With all of this setup connected, some disturbances (e.g. step changes) can be easily provided inside any of the programmable signal generators, for the sake of experimentation. It is important to note that, the cRIOs can talk to each other, via the physical TCP/IP network through their own ports. In our case, the SSG on the cRIO-9081 was interacting with the PMU1 on cRIO-9082 via port 4712, with PMU2 on cRIO-9068 via port 4713 and with PMU3 on cRIO-9068 via port 4714.

To test the proposed SSG prototype's reliability, a configurable network traffic generator CandelaTech CT-910 [10] was used. Using the CT-910 device, additional delays were introduced in between the PMU and the prototype SSG. In Figure 3, the insertion of the CT-910 network traffic generator within the existing network is graphically shown. The network traffic generator is capable of introducing a user-configurable delay between the two network. The CT-910 runs a standard Linux operating system, and is capable of modifying its own internal network architecture on the fly. This can be performed from a networked GUI, or by directly running shell-commands on the operating system of the network traffic generator.

The PMUs need to be fed with legitimate 3 phase signals in order for them to produce any meaningful output stream. To this end, three configurable time synchronized balanced 3 phase signals were designed on the existing cRIO 9081 device by the use of NI 9263 C series modules and configured using a LabVIEW GUI. An additional cRIO 9068 hardware was used to generate three programmable balanced three phase signals. A picture of part of the hardware used for the experiments is shown in Fig. 4.

4 Experimental Results and Analysis

4.1 Real-Time performance testing

The first test for validating the SSG prototype was to verify whether the communication is continuous, real-time and accurate. It needs to be noted, that the data observed in the SSG may suffer a lack of accuracy because of occasional data drops over the TCP network from PMU to SSG. It is also important to note that, a reduction in the transmission rate can lead to a more reliable communication, however, it

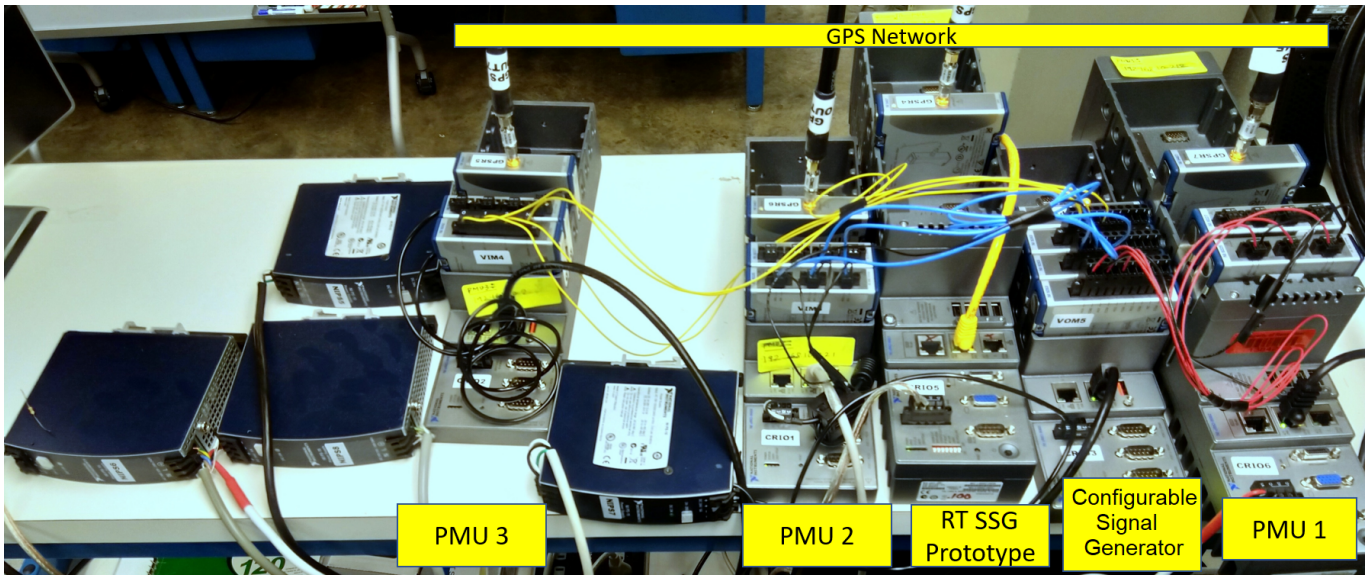

Figure 4: The Experimental Setup

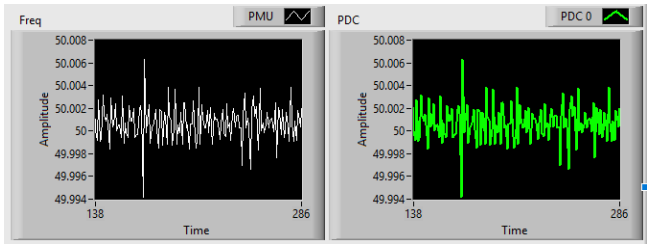

Figure 5: Frequency Measured in PMU (white) and the Measured Frequency received by the PDC(green)

will sacrifice the data rate offered to the SSG. To illustrate, in figure 5, the PMU (white) reports the frequency at a rate of 50 samples/second. However, the reporting rate is set to 20 samples/sec, thus the SSG reads only 20 samples every second.

For testing, the entire experiment was configured to run for several days, to make sure the TCP communication would not abruptly break or saturates. Upon rigorous testing, it was concluded that, the proposed architecture was stable, and can support multiple PMUs (up to 3 were) tested, connected over a network.

4.2 Formulation of delay from PMU to SSG

The reference timing diagram for the experimental setup is shown in Fig 3. PMU1 measures data at time t_1, whose phasors have a corresponding time-stamp T_{S1}. PMU2 measures data at time t_2, with corresponding time-stamp T_{S2}. PMU3 measures data at time t_3, with corresponding time-stamp T_{S3}. All these measurements are sent to the SSG.

The SSG will read any data at time t_P with corresponding time-stamp T_P. T_P will be *greater* than T_1, T_2 and T_3, because the SSG receives data that will be immi-

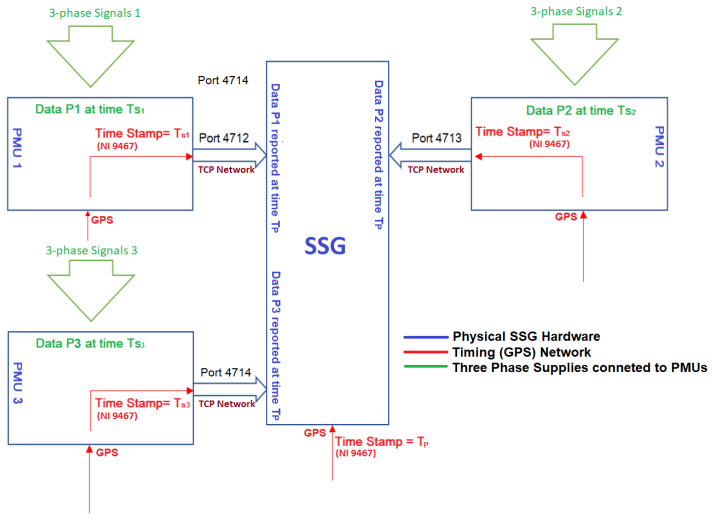

Figure 6: Timing Diagram of the Experimental Setup

nently delayed. For a given data snapshot, three new variables are defined as follows.

$$\Delta_1 = T_P - T_{S1}, \Delta_2 = T_P - T_{S2}, \Delta_3 = T_P - T_{S3}$$

In the sequel, Δ_1 Δ_2 and Δ_3 are analyzed to characterize the performance of the SSG setup. The PMU with the shortest delay can be found very easily. When more than three PMUs connected to the SSG, it would be necessary to sort the connected PMUs in terms of delay. In neutral test conditions Δ_1, Δ_2 and Δ_3 exhibited similar characteristics. NI's Network Published Shared Variable (NPSV) library was used to measure and compute these delays. With this test conditions, the three aforementioned delays were characterized by the statistic reported in Table

1. Afterwards, the experiments were performed without NPSV infrastructure as well.

Table 1 clearly shows that the network is largely uniform across all three communication links. Hence, the next experiments on delay were performed only for PMU 1, as all three PMU-SSG links will behave similarly way under the studied network conditions.

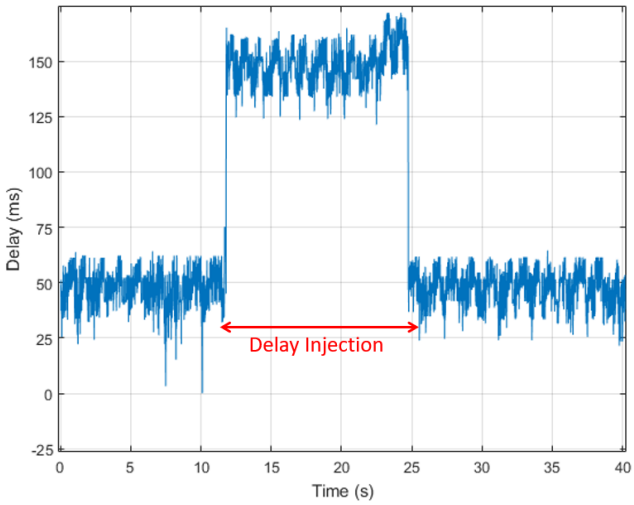

Figure 7: Example of 100 ms delay injection in the communication link

4.3 Stress-testing the SSG

In this particular test, artificial delays were introduced in between PMU1 and SSG prototype. To introduce artificial delays, CT910 Network Traffic Generator was used. The hardware configuration is shown in Fig 3. The CT910 has a configurable GUI which can tamper with the network between the two ports where PMU1 and SSG prototype were connected, respectively. In Fig. 8, the GUI is shown. It can be seen that the network between the two ends are closed (black) and the network is functional (45M-clean). It can incorporate user specified network traffic, as shown by the green lines (45M-random and 45M-impair). In Fig. 7, a sample network impairment (by a delay injection of 100 ms) for a duration of 15 seconds is shown. Figure 9 shows the plots with variable amounts of network delay injected in between the PMU1 and the SSG prototype. It is interesting to note that, with 50 ms delay injected (blue plot), the actual average PMU1 to SSG delay is close to 90 ms. This gives an indication that the network itself has an intrinsic

Table 1: Delay Statistics for all the PMUs (NI Network Published Shared Variable used for Measurements)

	Δ_1	Δ_2	Δ_3
Mean (s)	0.0481	0.04726	0.04824
Standard Deviation (s)	0.0199	0.02065	0.01945

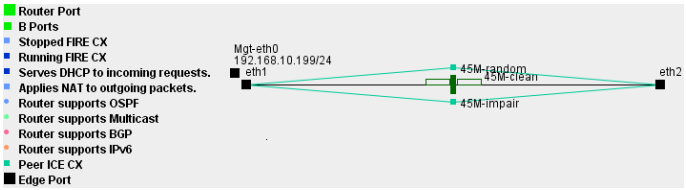

Figure 8: Tampering the Network Between PMU 1 and the SSG

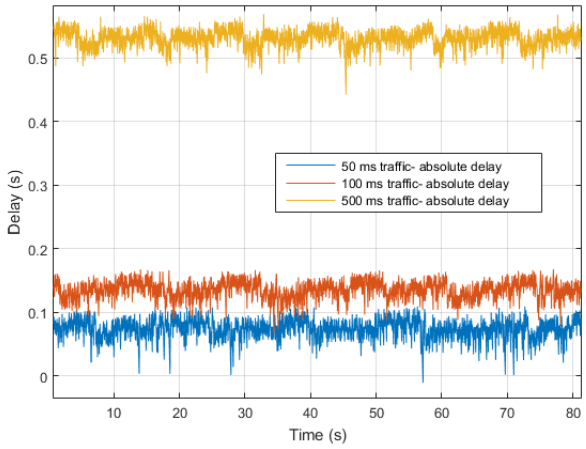

Figure 9: PMU1 to SSG delay after variable additional network traffic injection

delay of about 40ms. To validate this hypothesis, from all the delay plots of Fig 8, the injected delay is subtracted (i.e. if the injected delay is 500 ms, 500 ms is subtracted from all observations). The new parameter is described as the Estimated Pass-through (EPT) delay from here onward.

The EPT delays for three different values of fixed injected delay (50ms, 100ms, and 500ms) are plotted in fig

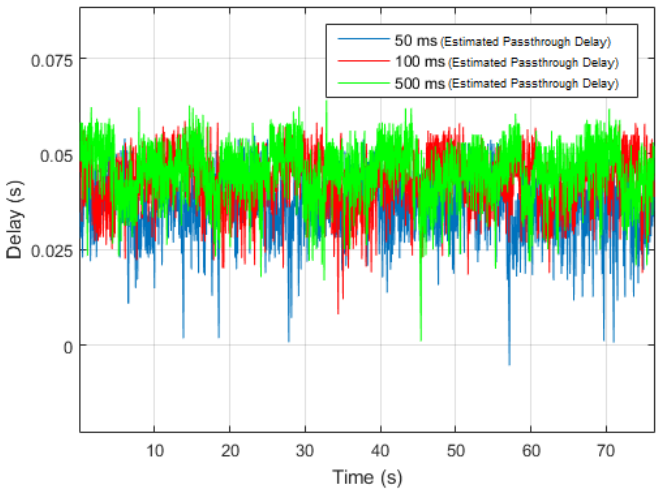

Figure 10: Estimated Pass-through (EPT) Delay for PMU1 to SSG delay after variable additional traffic injection

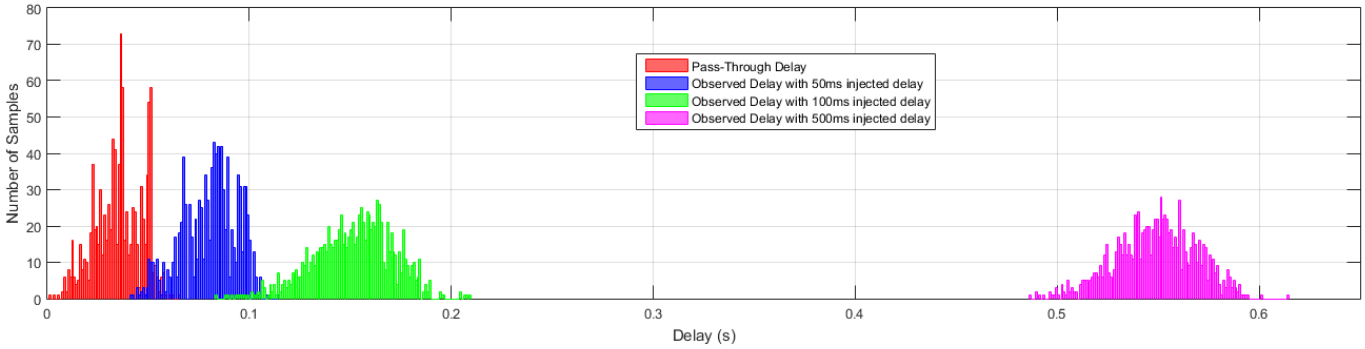

Figure 11: Histograms for Measured Delay in different network traffic injections

Table 2: Delay Reduction after remove NI Network Published Shared Variable Infrastructure

Test-setting	μ	σ
With NI Network Published Shared Variable	0.0481	0.0199
Without NI Network Published Shared Variable	0.0180	0.0215

10. It can be seen that, the intrinsic delay of the path is fairly similar for all three configurations. It was also observed, that even without any injected delay, the network exhibited an average PMU1-to-SSG delay of 47 ms over a run-time of 4 hours.

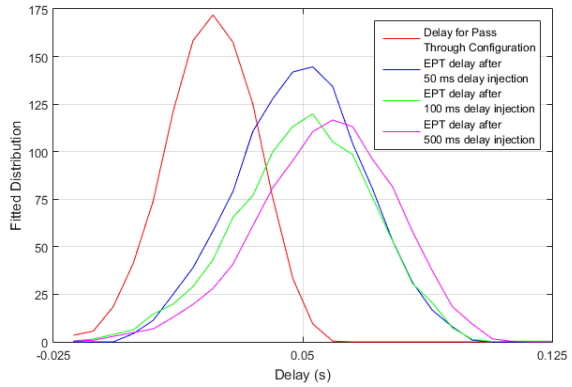

Figure 12: Fitted Distributions of the Delay-Histogram for different values of injected delays

To investigate further, histograms for all the delays with varying injected delays are shown in Fig 11, along with the delay of the system when the network is set in pass-through mode (i.e. no injected delay). It can be observed that, the histogram gets wider as the injected delay increases. Figure 12 shows the fitted distributions of the EPT delay-histograms for these observations.

In order to reduce this delay, two changes were made to the network (i) the PMU-SSG interconnect was reconnected with crossover cables, (ii) all instances of NI Net-

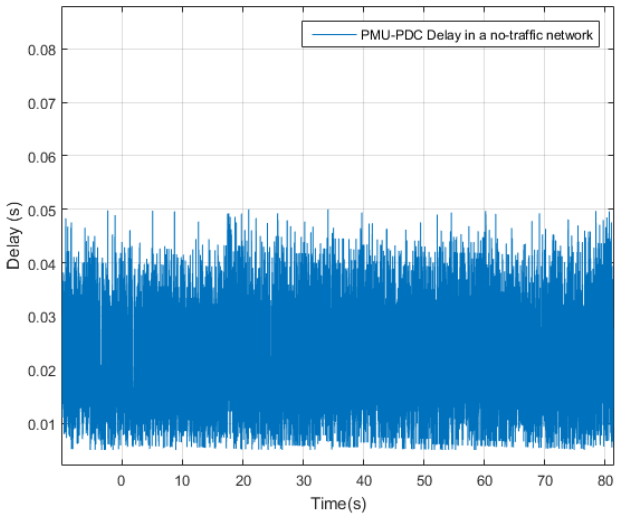

Figure 13: PMU1 to SSG delay without using NI Network Published Shared Variable

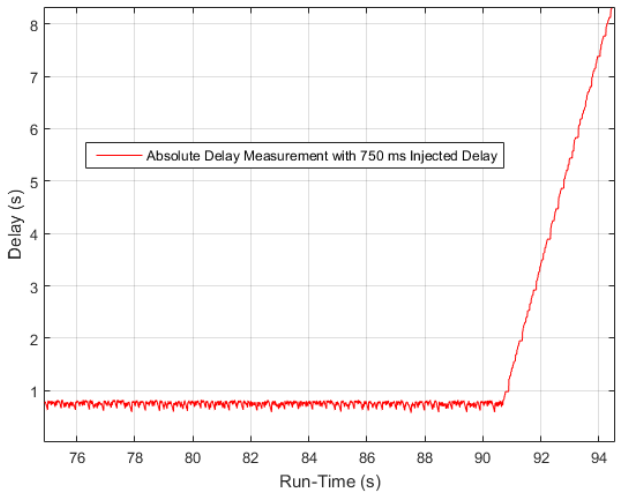

Figure 14: Absolute PMU1 to SSG delay with very high delay-injection (750 ms)

work Published Shared Variables were removed. Once, these changes were made, the actual time of execution of the SSG algorithm and data-transmission protocols should be the only component present in the total delay.

However, without the presence of shared variables, the measurement procedure becomes more difficult. To measure precise time-stamps in this new setting, time-stamps are converted to numbers and sent through the analog channels of the PMU which are part of the C37.118.2 protocol. In this new setting, the delay between the PMU1 and the SSG prototype is observed. A standard plot of the reduced delay is shown in Fig 13. Table 2 exhibits the improvements observed in terms of timing when compared to the results in Table 1.

In order to stress the system, the injected delay was increased from 500 ms to 750 ms. As shown in Fig. 14, it was observed that, with such a high injection of delay, the network becomes extremely unstable and breaks the communication link between the PMU and SSG after a few seconds. In fact, this phenomenon was observed, whenever the injected delay was more than 600 ms. It can thus be concluded that 500 ms is the maximum possible delay that can be injected into the network without compromising the reliability of the system.

5 Conclusion

This paper proposes a new architecture an implementation for a true real-time Synchrophasor Synchronization Gateway (SSG) architecture. The goal of this SSG is to replace a PDC in PMU-based real-time control applications. For testing, three physical PMUs were connected to the prototype SSG. The delay of measured data transmission from all the PMUs to the SSG was recorded and analyzed for assessing the reliability of the proposed real-time SSG architecture. With additional hardware, network traffic of varying amount is injected- to test the timing-performance and robustness of the proposed SSG prototype. By these experiments, the delay of the TCP network between PMU and SSG prototype was characterized, its limits were tested, and possible counter-measures to minimize this delay was proposed.

References

[1] S. R. Firouzi, L. Vanfretti, A. Ruiz-Alvarez, F. Mahmood, H. Hooshyar and I. Cairo, "An IEC 61850-90-5 gateway for IEEE C37.118.2 synchrophasor data transfer," 2016 IEEE Power and Energy Society General Meeting (PESGM), Boston, MA, 2016, pp. 1-5.

[2] P. Castello, C. Muscas, P. Attilio Pegoraro and S. Sulis, "Low-Cost Implementation of an Active Phasor Data Concentrator for Smart Grid," 2018 Workshop on Metrology for Industry 4.0 and IoT, Brescia, 2018, pp. 78-82.

[3] M. G. Adamiak, M. Kanabar, J. Rodriquez and M. D. Zadeh, "Design and implementation of a synchrophasor data concentrator," 2011 IEEE PES Conference on Innovative Smart Grid Technologies - Middle East, Jeddah, 2011, pp. 1-5.

[4] H. Retty, J. Delport and V. Centeno, "Development of tests and procedures for evaluating phasor data concentrators," 2013 IEEE Grenoble Conference, Grenoble, 2013, pp. 1-5.

[5] J. Pongnark and S. Tanachutiwat, "Performance and reliability benchmarking of phasor data concentrator software systems and preliminary designing of wide-area monitoring system," 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Phuket, 2017, pp. 326-329.

[6] A. Dervikadi, P. Romano, M. Pignati and M. Paolone, "Architecture and Experimental Validation of a Low-Latency Phasor Data Concentrator," in IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 2885-2893, July 2018.

[7] L. Vanfretti, I. A. Khatib and M. S. Almas, "Real-time data mediation for synchrophasor application development compliant with IEEE C37.118.2," 2015 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, 2015, pp. 1-5.

[8] L. Vanfretti, G.M.Jnsdttir, M.S.Almas, E.Rebello, S.R.Firouzi, M.Baudette, "AudurA platform for synchrophasor-based power system wide-area control system implementation" SoftwareX, Volume 7, Jan-Jun 2018. pp- 294-301.

[9] http://www.ni.com/en-us/innovations/energy/power-quality-analysis.html

[10] https://www.candelatech.com/ct910_product.php