Developing Process Variables Necessary to Operate Simulacrum: The LCLS Accelerator Simulator

Mirian G. Juan Estrella¹, William Colocho²

¹Department of Mathematics and Statistics, California State Polytechnic University, Pomona
²Normal Conducting Linac and Free Electron Laser Division, SLAC National Accelerator Laboratory

Motivation

Simulacrum is a system that will simulate the Linac Coherent Light Source (LCLS) and its control system. This will allow future operators to train on a simulated linear accelerator and provide research scientists the opportunity to test future experiments. We will study the network systems in Simulacrum to write process variables (PVs) that will update graphical user interfaces from LCLS.

Background

Network communications with LCLS begin when the user on an OPI machine that contains EPICS, a computer software, sends commands to Input/Output Controllers (IOCs). IOCs are computers that communicate with each other to provide information about a specific (PV) that measures the value of various parts of a device (See Figure 1.) They are computers that communicate with each other to provide information about a specific (PV) that measures the value of various parts of a device (See Figure 1.) Conversely, Simulacrum contains services containing PVs that communicate via a high-performance messaging library called ZeroMQ (See Figure 2.)

Methods

The creation of Simulacrum is a group effort. Therefore many Simulacrum environments can be created to build it and are distinguished by unique port numbers (See Figure 5).

Methods Continue

Next, to populate the PVs in the klystron and generic PV with valuable data we used the following methods:

- write MATLAB code to call the `caput('pvName_here')` command for all PVs
- write a Python script that imported sensible data from the archive or in production

Results

After running the model, klystron, and generic services and other MATLAB and Python scripts, we get can observe four graphical user interfaces (See LCLS Displays on Simulacrum). We encountered PVs that had to be populated with fake data, but data that still made sense. For instance, PVs can take in many values like integers, floats, vectors, strings, and bytes. Therefore, understanding the role of these PVs is important. A possible solution to addressing such unique PVs may be in creating a new service that houses them. This may make it possible for services that currently do have PVs with values to communicate efficiently.

Discussion

Our results lead to several questions left for discussion:

- What happens when a PV does not have a value in the archive or in production?
- Where will the values come from?

We encountered PVs that had to be populated with fake data, but data that still made sense. For instance, PVs can take in many values like integers, floats, vectors, strings, and bytes. Therefore, understanding the role of these PVs is important. A possible solution to addressing such unique PVs may be in creating a new service that houses them. This may make it possible for services that currently do have PVs with values to communicate efficiently.

Acknowledgements

I am very grateful for the invaluable direction and mentorship provided by William Colocho. Moreover, I am very thankful towards the 2019 STEM Teacher and Researcher Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funders.