

LA-UR-20-25446

Approved for public release; distribution is unlimited.

Title: A Modern User Interface for the LANL Neutron Pulse Simulator (NPS)

Author(s): White, Thaddeus

Intended for: 2020 Virtual Student Symposium

Issued: 2020-07-23

Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

A Modern User Interface for the LANL Neutron Pulse Simulator (NPS)

Thaddeus “Peter” White

Neutron Pulse Simulator (NPS)

- Neutron Coincidence is a technique used to measure how neutrons are emitted from a radioactive material. This technique is able to capture the unique signature of nuclear material.
- The NPS is an instrument designed “for use as a training and testing tool for neutron coincidence and multiplicity counting systems” (LA-UR-06-4208).

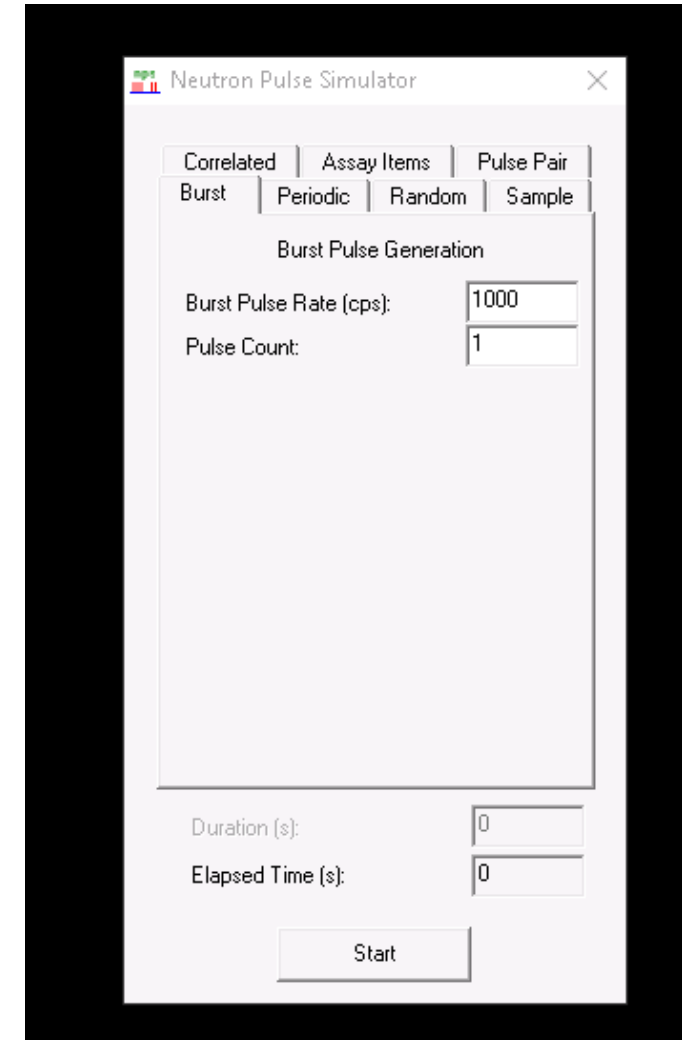


Importance of NPS

- The NPS aids development and testing of software and/or devices without actually using nuclear sources.
- The NPS is used for “measurement control, quality control, diagnostics, and training for neutron coincidence counting systems” (LA-UR-06-4208). In addition, it increases safety for nuclear material testing as it allows personnel to avoid contact with radioactive material.

Windows application

- Windows application was made to control the NPS' pulses.
- It gives the user 7 different modes to choose from to configure the pulses sent.
- Can be inconvenient to setup as it requires drivers to be installed.
- It also only works on a Windows operating system.



Modern Web Interface

- The issues with the Windows application inspired a newer more modern web user interface to be designed.
- The new application is much more robust, portable, and convenient.
- Because it is web based *any* device (including mobile) can control the NPS.
- Additionally there are no downloads or drivers required, simply a browser and the URL to the web application.

The screenshot displays a web application interface for controlling an NPS (Neural Network Processor). At the top, a pink status bar indicates 'NPS Stopped' and 'Last Command: PulsePair pulsePairRate: 1000 separation: 20'. Below this is a navigation bar with tabs: 'Burst' (selected), 'Periodic', 'Random', 'Sample', 'Correlated', 'AssayItems', and 'PulsePair'. The main control area includes three input fields: 'Pulse Rate' (set to 1000), 'Pulse Count' (set to 10), and 'Duration' (set to 0). An 'Elapsed Time' field shows 0. At the bottom of the control area are 'Start' and 'Default Values' buttons. A green status bar below the controls shows 'Connected. Stop Command Sent'. A light blue bar at the bottom indicates '1 User(s) connected'. An 'About' button is located at the very bottom.

NPS Stopped
Last Command: PulsePair pulsePairRate: 1000 separation: 20

Burst Periodic Random Sample Correlated AssayItems PulsePair

Pulse Rate
1000

Pulse Count
10

Duration
0

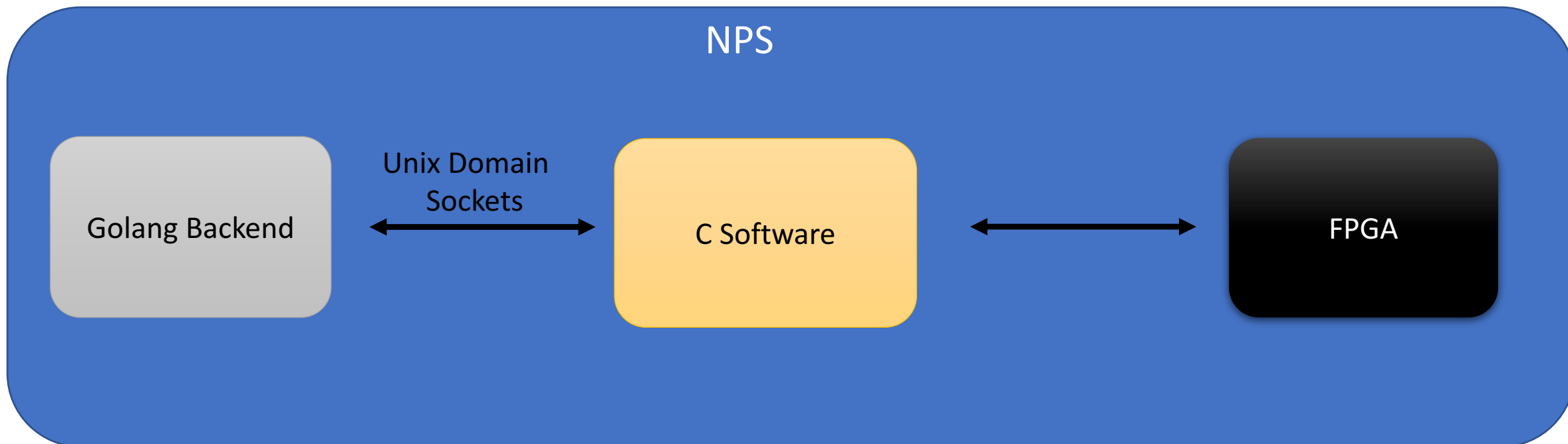
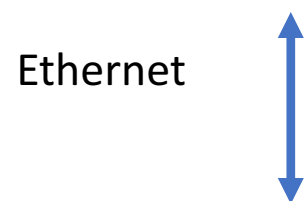
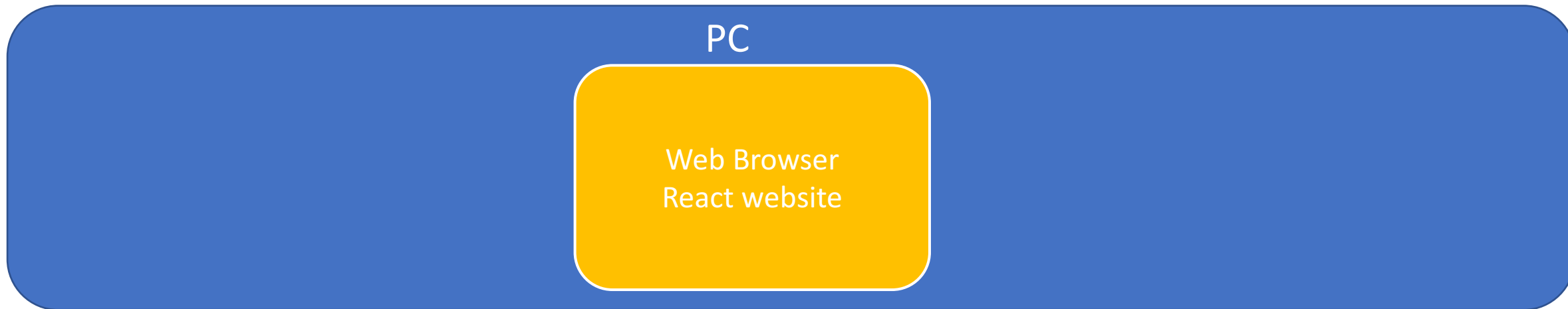
Elapsed Time:
0

Start Default Values

Connected. Stop Command Sent

1 User(s) connected

About



Challenges

- Having to learn modern JS and React, and starting with no knowledge of Go.
- Learning how to structure the programs, as well as how to use the communication protocols proved to be a bit more difficult than expected.
- In order to have smooth and efficient web socket communication, I had to rewrite the entire Go websocket handler to be concurrent. This allowed communication with the web interface, while simultaneously communicating with the NPS.
- Although there were many more challenges, the last I will mention was rewriting thousands of lines of code from the older Windows Application (written in C++) into Go.