

Multi-vendor MRI Scanner State Monitoring and Experiment Orchestration

Vinai Roopchansingh^{1,3}, Catherine R. Walsh^{2,3}, J. Andrew Derbyshire^{1,3}, Peter A. Bandettini^{2,1,3}

1. Functional MRI Facility, 2. Section on Functional Imaging Methods, 3. National Institute of Mental Health, National Institutes of Health, Bethesda, MD

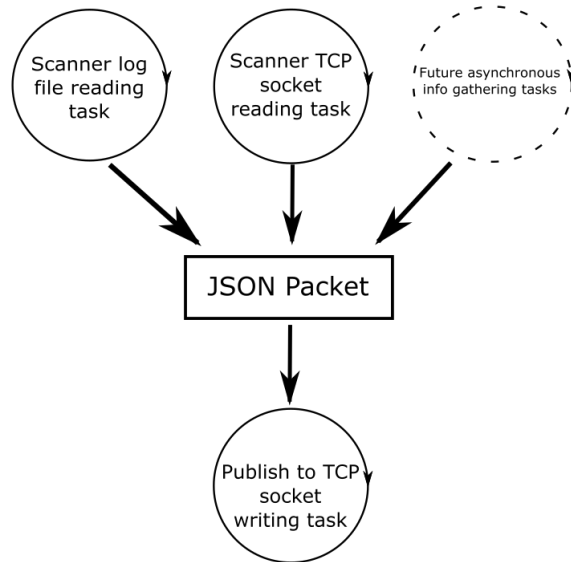
Introduction

Within research on MRI, fMRI, and fMRI with neurofeedback, a multitude of reasons exist for replicating studies, using similar, but not identical hardware and tools as the original study. These reasons include (but are not limited to): conducting multi-site studies, potentially utilizing multiple vendors' hardware platforms and software suites, and demonstrating that the results obtained are due to the effects being studied, and not an artifact of the equipment and tools used. As with our previous work, our goal is to provide tools and techniques to allow researchers to replicate fMRI and fMRI with neurofeedback experiments across multiple hardware platforms, and across different software suites. This will be accomplished by leveraging tools and utilities to extract information from multiple sources, and to translate that information into a format common across all platforms.

Methods

Our software, Dynamic Multi-Vendor MRI Scanner Watcher(DMV_MRI_S-Watch) is currently available from GitHub. This toolbox leverages Python's file reading, text parsing, networking, and (most importantly for this iteration of our application) asynchronous task scheduling extensions. The primary source of information fed to this toolbox is a MRI scanner console's log files. As these files are parsed, specific events happening on the scanner (inferred by the appearance of corresponding text flags in said log files), along with the time at which those events occurred, are extracted and used to determine the state which a MRI scanner is currently in. The software is currently able to do this for General Electric and Siemens MRI scanners. Previous iterations [1] of this tool published the scanner's state information to a Flask web application. However, relinquishing Flask for a more general TCP interface facilitated additional asynchronous capabilities, and allowed the publishing of a scanner's state information in a JSON (JavaScript Object Notation) packet to a network TCP socket. The information in the JSON packet then becomes available to any entity which can access this network socket, and can be leveraged to orchestrate experiments that need to be coordinated with the various state of MRI scanner operation (e.g. acquisition preparation, image acquisition, etc.). The states currently recognized and handled are: if a patient or object is registered to be scanned, the entity being deregistered and removed from the scanner, if the scanner is about to acquire data, and whether data are being acquired, and if data acquisition has stopped. Besides parsing a scanner console's log files, other parallelly executing, asynchronous tasks have been implemented, such as additional information gathering jobs, as well as publishing the scanner's current state. As of the submission of this work, one of these additional information gathering jobs reads data from another TCP socket to which the scanner's vendor publishes data. Binary file reading tasks are also being implemented, and in the long term, reading data from open, network-streamed data formats, such as ISMRMRD and NIFTI, and other tools, like LabJack and BIOPAC are being considered. A schematic of the various tasks, and how information flows through these is shown here:

Tasks and information flow



Discussion

The parallelized and asynchronous core of our scanner monitoring utility vastly expands the sources of information that can be used to determine the state a MRI scanner is in. Publishing this information to a network socket also makes it easier to leverage such information, and can even make hyper-scanning experiments (i.e. one involving more than one MRI scanner and subject) more accessible. Using the software described here, the state information from multiple instruments can be made available on a common, easily accessible interface.

References

1. Roopchansingh and Bandettini, rtFIN 2022, New Haven, CT, USA.