I was unable to get this OSP project completely working as of yet. I was able to at least mostly implement all of the functions, including: do\_create, do\_destroy, do\_send, and do\_recieve.

In the do\_create method, our primary goal is to assign a port to our current task. We must first ensure that we have a current task, and that the current tasks port count doesn’t exceed the max. We then set the task and status of the port, and update this port’s buffer size.

In the do\_destroy method, we destroy whatever port we call the function/method on. We set its status to PortDestroyed, notify all other threads that we destroyed a port, remove this port from the current task, and nullify the current task.

In the do\_send method, we first ensure that the message is not null, and that its length does not exceed our buffer size. Assuming these checks pass, we create a systemevent for the log, and pull the thread of the current task. We then continually check the buffer needed for the current thread, cycling through threads and suspending them until we can find one that will fit in the buffer, then finally appending that message. We notify the other threads of the success.

In the do\_recieve method, we obtain the current thread, and ensure the task isn’t null. We then iterate over the threads, suspending them if they are empty, and ensuring the status of the thread and port are what we expect. Once we finally satisfy conditions, we return the message.

My largest problems have come with the suspending of threads, and keeping track of the buffer. I have been through many iterations of my code, and none of my proposed solutions have worked fully. The error in the version I am submitting is that we are unable to enqueue a thread that has already been enqueued to this event. I am unable to troubleshoot this error fully. I believe that my implementation of the remaining code is at least in the correct direction.