Franklins Non Blocking

Samuel Flinkfelt
Anne Poso

Summary

The topic that we chose was “Write a distributed program in MPI that simulates one of the leader election algorithms learnt in class for a ring topology using a fixed number of processors.” We ended up choosing Franklins Algorithm, ended up using C as our base code language, installed the openMPI Library into both of our IDEs VS Code and Xcode.

Since we implemented our program in C, we had to include the mpi.h header for the MPI library
The MPI commands we used were
MPI_Init to initialize MPI,
MPI_Comm_size to figure out the number of processors that are going to be used
MPI_Comm_rank to assign the processor’s rank

And we implemented Franklin’s algorithm using non-blocking communication, so the MPI commands we used were MPI_Isend, MPI_Irecv, and MPI_Wait

The program starts off by initializing MPI, and getting the size and rank.

Then it defines a boolean array called isActive to track which processors are active, and they’re all initialized as true

And there’s a boolean variable isElected that’s used in our while loop to track if a process has been elected leader.

Then we define MPI_Request and MPI_Status variables for our send and receive operations
And then we define an integer array called rankID that holds the IDs for our ranks, so process of rank 0 has ID 50, process of rank 1 has ID 102, rank 2 has ID 75, and rank 3 has ID 98 and we define these integers as our message buffers that will be used sending and receiving the node rank IDs.