Assignment 3 Technical Document

The primary goal of this assignment is to implement a custom barrier synchronization mechanism among a group of MPI processes. Barrier synchronization ensures that all processes in a group reach a certain point in the code before any of them can proceed further. This implementation aims to achieve synchronization with a maximum of $4 \log_2 P$ steps of (possibly parallel) send-receive operations, where P is the total number of processes.

Initialization

- The MPI environment is initialized, and each process obtains its unique rank and the total number of processes in the group.

Barrier Synchronization

- The custom barrier function, `mybarrier(MPI_Comm comm)`, is designed to synchronize all processes using only `MPI_Send()` and `MPI_Recv()` operations.
- The synchronization is achieved in two phases, each requiring $\log_2 P$ steps, thus ensuring that the total number of steps does not exceed $4 \log_2 P$.

First Phase (Sending Messages):

- Each process sends a message to its partner in each step. The partner's rank is determined by XORing the current rank with 2^i, where i is the step number (ranging from 0 to $\log_2 P - 1$).
- After sending the message, the process waits for a message from its partner, ensuring synchronization at this step.

Second Phase (Sending Confirmation Messages):

- In this phase, each process sends a confirmation message back to its partners in reverse order of the first phase.
- The process sends a message and then waits for a confirmation message from its partner, ensuring that both processes have reached this point in the barrier.

Finalization

- After completing the barrier synchronization, the MPI environment is finalized.

Conclusion

- By implementing a custom barrier using only send and receive operations, this assignment demonstrates an efficient way to achieve synchronization among MPI processes. The design ensures that the number of synchronization steps is kept to a minimum, enhancing the overall performance of parallel applications.