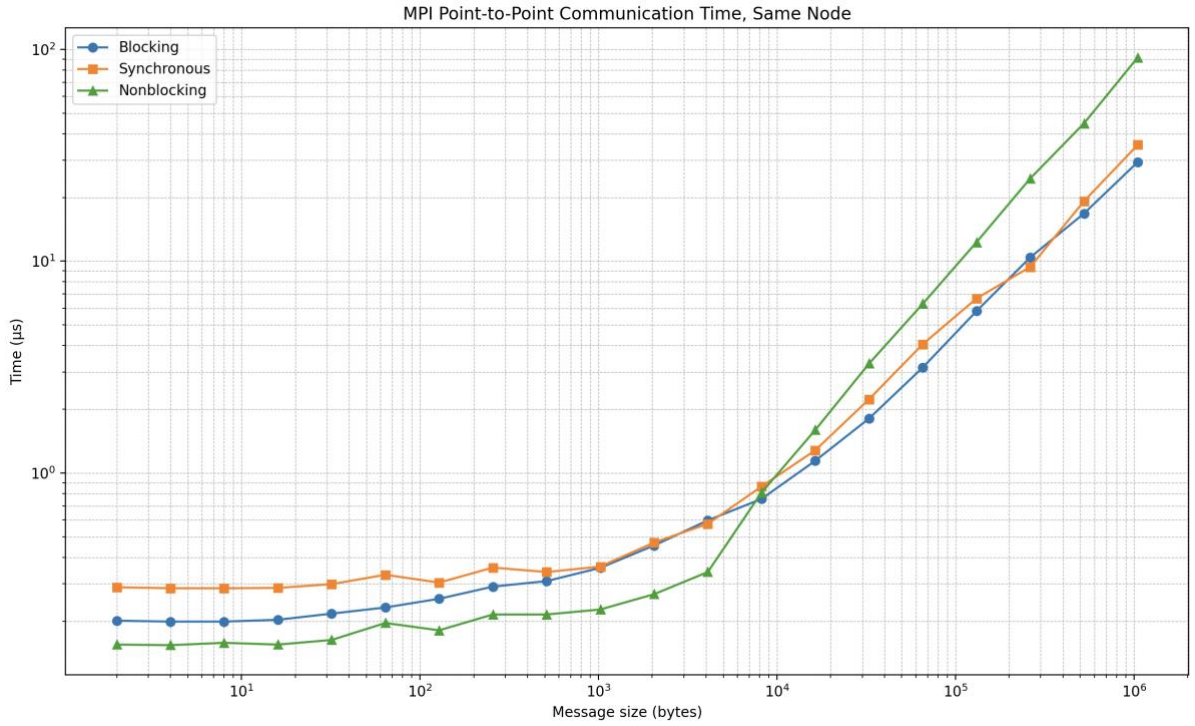
Alex Mecklin 907666

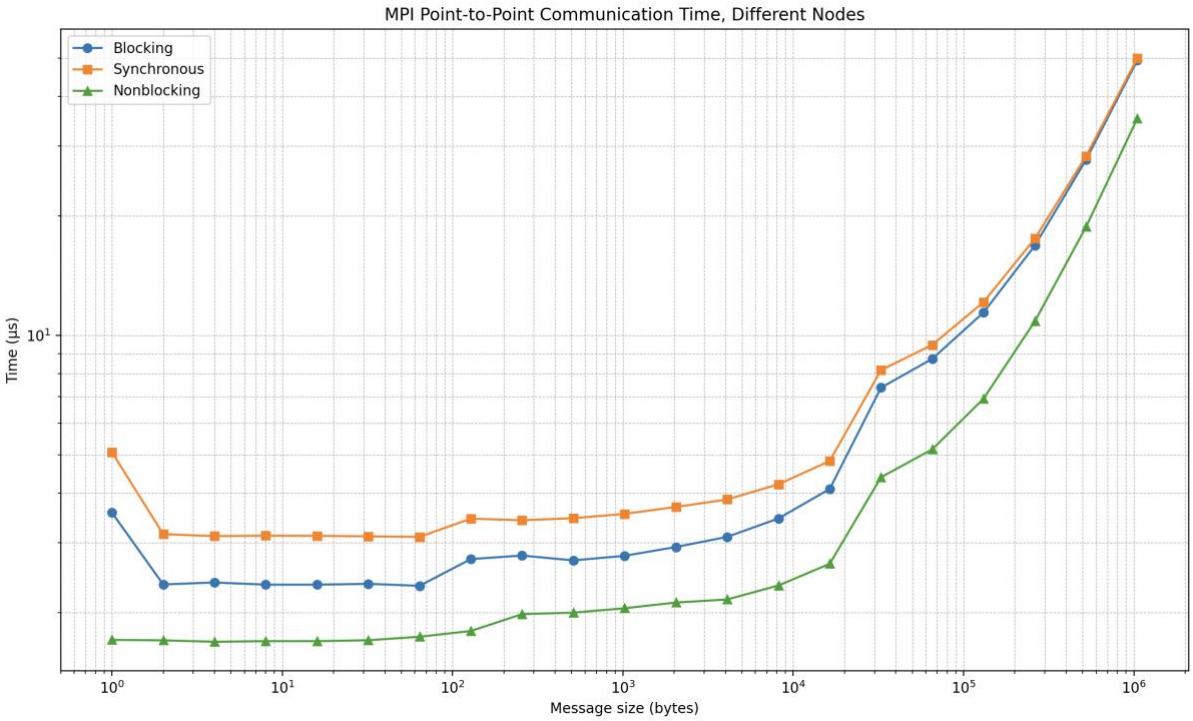
The program evaluates three types of communication: blocking, synchronous, and non-blocking. Blocking communication uses MPI_Send and MPI_Recv, where the sender waits until the data is safely handled by MPI, which may include internal buffering. Synchronous communication uses MPI_Ssend and MPI_Recv, ensuring that the sender waits until the receiver has started receiving the message. Non-blocking communication employs MPI_Isend and MPI_Irecv to initiate communication without waiting, and MPI_Waitall ensures that both operations complete before timing ends.

The program repeatedly measures the communication time for a series of message sizes, ranging from 1 byte up to 1 megabyte, increasing in powers of two. Each message size is sent and received 1000 times to reduce random variation in the results. Timing is measured using MPI_Wtime() and divided by two to estimate the one-way latency. At the end, the program prints a table showing the message size in bytes and the corresponding latency in microseconds for each communication type.

Graph for running 2 processes on the same node:



Graph for running 2 processes on different nodes:



At first, the results were as expected, with same-node communication being a bit faster for small message sizes. But surprisingly, for larger buffers the times on the same node became longer than on different nodes. One possible reason could be that MPI uses extra memory copies on the same node, which might slow things down when the messages get very big. Another idea is that the network hardware might actually handle large transfers more efficiently than shared memory.