Cpt S 411 Assignment Cover Sheet

Assignment #1

For team projects:

List of all students (Last, First): (Valdez, Paul), (Hellwig, Benjamin)

List of collaborative personnel (excluding team participants):
Ananth Kalyanaraman

NOTE:
Reused parts of the example code given, specifically mpi_send_recv_test.c code.

I[1] certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received or given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment. I also certify that I have not referred to online solutions that may be available on the web or sought the help of other students outside the class, in preparing my solution. I attest that the solution is my own and if evidence is found to the contrary, I understand that I will be subject to the academic dishonesty policy as outlined in the course syllabus.

Please print your names.

Assignment Project Participant(s):
Paul Valdez
Benjamin Hellwig

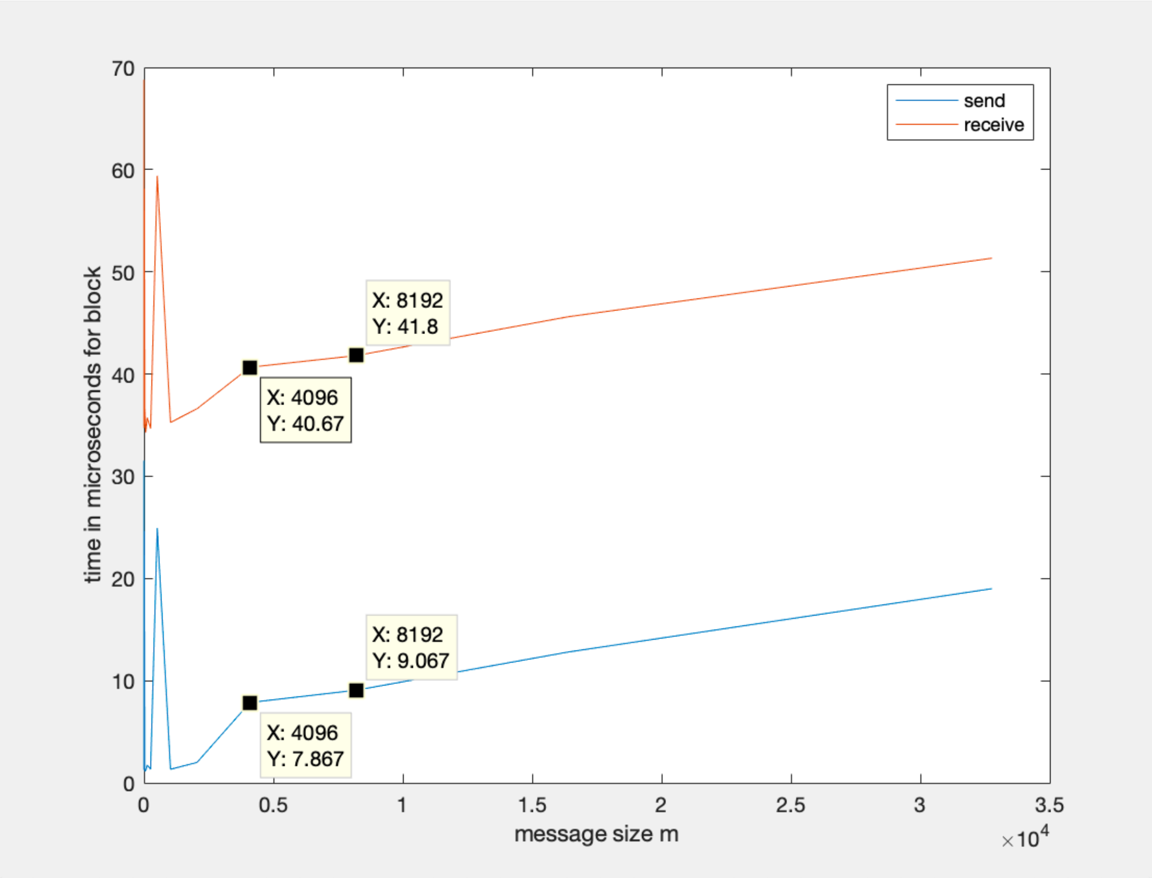
Today's Date:
September 16th, 2019

[1] If you worked as a team, then the word "I" includes yourself and your team members.

BLOCKING Test:

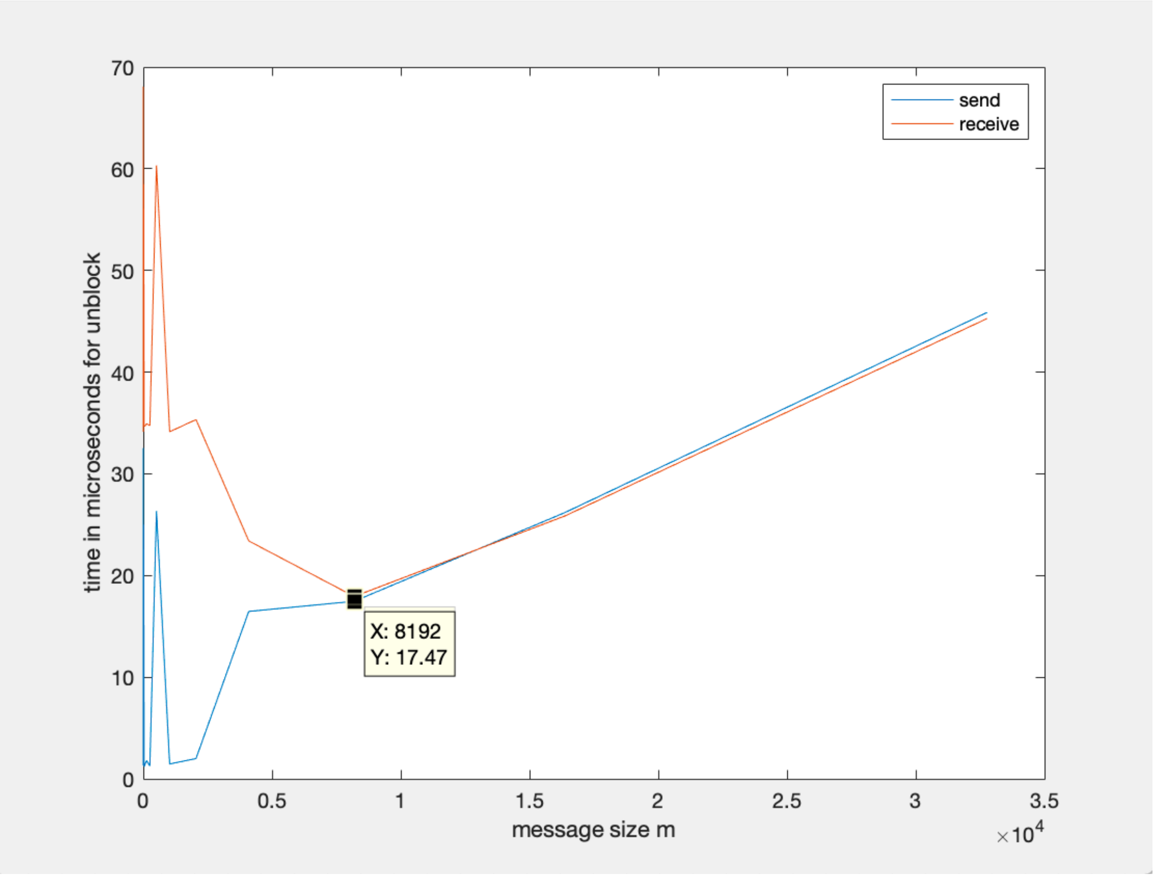
After running the code with message sizes m = 1,2,4,8,…,2^{26} bytes with 15 iterations for each given message size, we averaged the communication time for both MPI_Send and MPI_Recv, respectively, and used this data to create the following plots in MATLAB.

Figure 1.



Based off of figure 1 alone, it looks like the network buffer size could possibly be 4096 bytes since the graphs of communication time versus message size begin to increase linearly after that point; however, upon taking into account the non-blocking MPI implementation, as shown in figure 2, we can tell that the network buffer size is likely 8192 bytes, because at that point the times of the non-blocking MPI send and receive calls meet and begin to increase linearly as a function of message size.

Figure 2.



From figure 1, knowing it is likely that the network buffer size is 8192 bytes, we get that the latency for MPI_Send is approximately 9.067 microseconds. Likewise, we see the latency for MPI_Recv is approximately 41.8 microseconds.

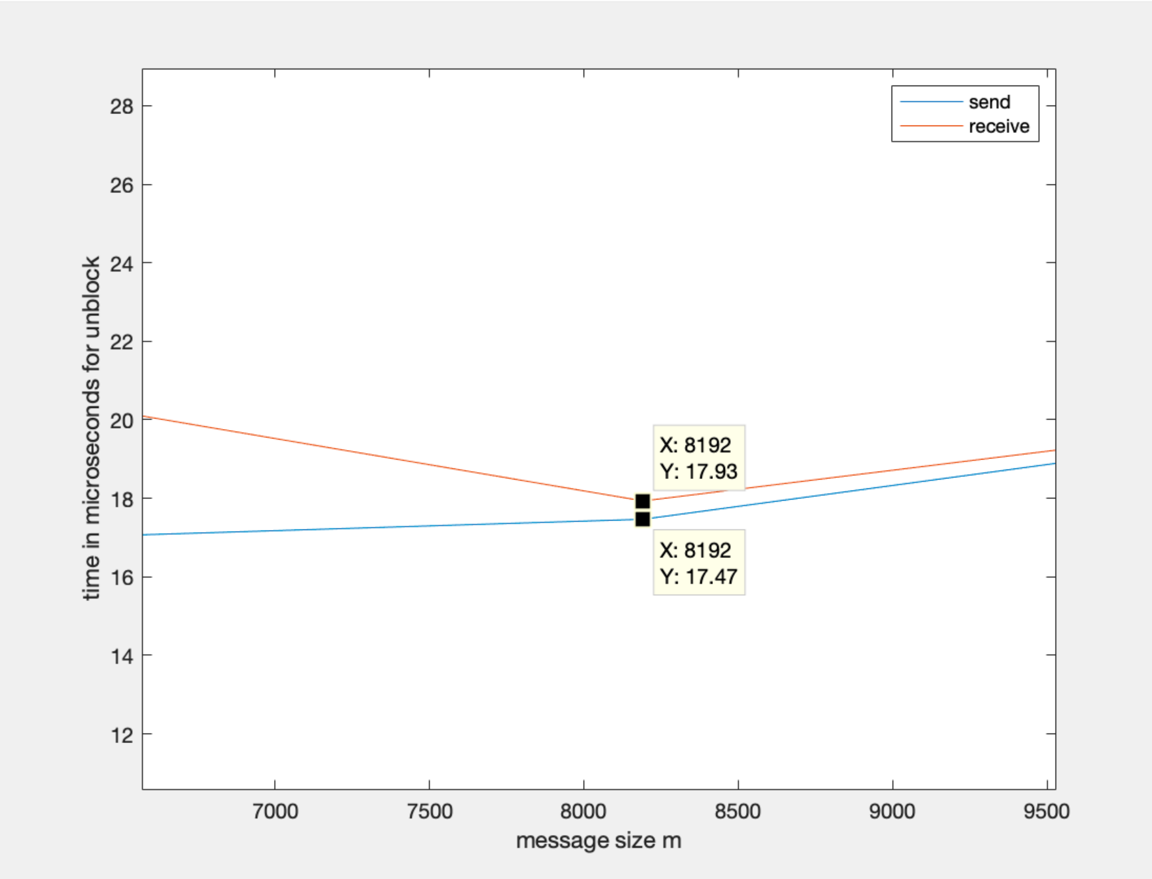
Using the built-in MATLAB linear regression tools for data points 14-27 (m = 2^{13} to 2^{26} bytes), we found that the bandwidth for MPI_Send in the blocking implementation is approximately 1047.71 bytes-per-microsecond. Likewise, we found that the bandwidth for MPI_Recv in the blocking implementation is approximately 1049.38 bytes-per-microsecond.

NON-BLOCKING Test:

After running the code with message sizes m = 1,2,4,8,…,2^{26} bytes with 15 iterations for each given message size, we averaged the communication time for both MPI_Send and MPI_Irecv plus MPI_Wait, respectively, and used this data to create figure 2, shown above, in MATLAB.

We zoom in on figure 2 to create figure 3. Knowing it is likely that the network buffer size is 8192 bytes, we get that the latency for MPI_Send is approximately 17.47 microseconds. Likewise, we see the latency for MPI_Irecv with MPI_Wait is approximately 17.93 microseconds.

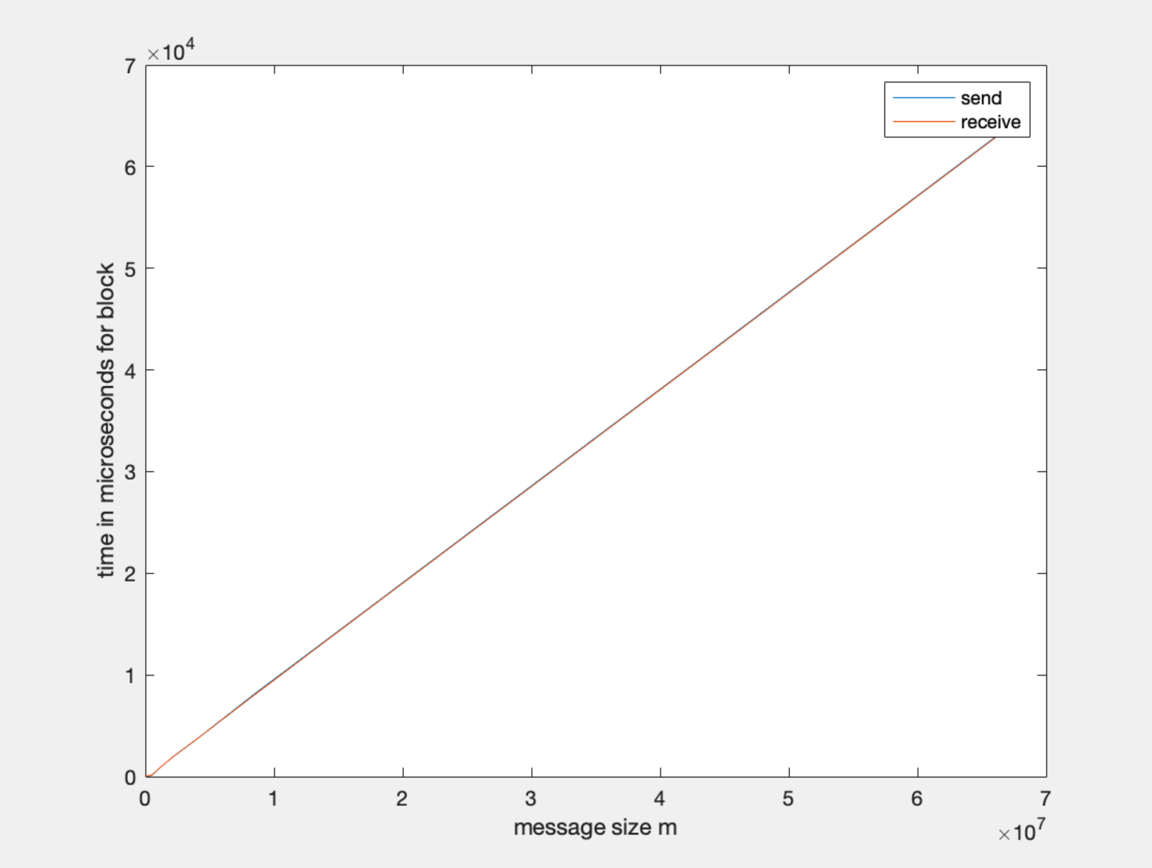
Figure 3.



Using the built-in MATLAB linear regression tools for data points 14-27 (m = 2^{13} to 2^{26} bytes), we found that the bandwidth for MPI_Send in the non-blocking implementation is approximately 1106.39 bytes-per-microsecond. Likewise, we found that the bandwidth for MPI_Irecv with MPI_Wait in the non-blocking implementation is approximately 1107.78 bytes-per-microsecond.

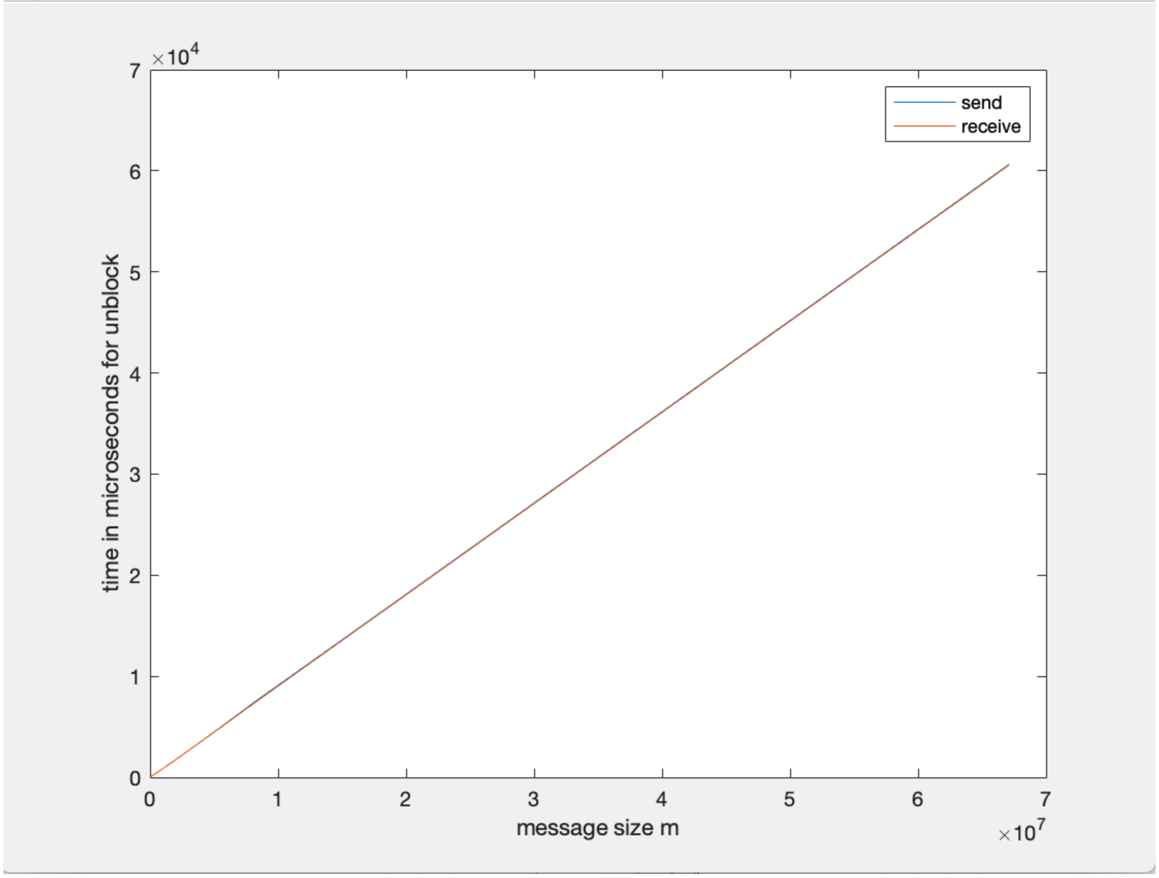
The complete graph for the blocking implementation is shown below in figure 4.

Figure 4.



The complete graph for the non-blocking implementation is shown below in figure 5.

Figure 5.


×10^4
time in microseconds for unblock
message size m
×10^7
send
receive