TDT4200 Problem Set 3

How can non-blocking communication be useful?

Non-blocking communication allows the sender (or receiver, but not both) to continue working without having to wait for the receiver. This could be combined with something like a queue to allow dependents to be slower than their dependencies.

Can you think of any limitations of the Hockney/pingpong model for modeling communication performance?

Latency and available bandwith vary wildly in modern systems and depend on the state of the system. Results will also be significantly different between local CPUs and remote CPUs. This makes the metric not very useful for measuring overall performance.

Find the execution time of the main loop when running with 1 processes. Find the execution time of the main loop when running with 2, 4 and 8 processes and document the speedup you get compared to running with a single process. Is the result as you would expect? Why/why not?

Tests done on a CPU with 8 execution units.

Ranks	Time	Comment
1	62.7 s	Pretty slow as expected.
2	Segfault	Something in `domain_init` is going horribly wrong. My guess would be a speedup of ~1.5x.
4	18.6 s	A 3x speedup is very much expected for this kind of workload. The result is still incorrect, though.
8	18.5 s	Surprisingly no faster at all. The console output definetly *looked* smoother. I imagine that the Linux scheduler simply won't let MPI use up all 8 execution units in order to not lock up my system. Also, this segfaults when freeing in `domain_finalize`.

The results seem to be predictable, considering the limitations of the system (and the quality of code). There is definetly something wrong with my solution but after hours of debugging I have to throw in the towel. I'll definetly go the lab this week to get some assistance as the same thing happened last time.