Latency Mitigation

Based on the previous graphs when we first implemented latency, we determined that excluding natural latency across servers, we have a relatively stable graph with execution time on the x-axis and estimated latency(in milliseconds) on the y-axis that follows our expectations of how the latency should be reflected on a timed interval.

Due to how we structured our latency buffer, we decided that it would be more fitting that we use Time Delay Model instead of Time Wrap model. How we created latency was that we created a queue on the server side that only updates game state at every designated server ticks through the periodic handler. After we implemented the mitigation mechanism, the periodic handler will now take a boolean parameter that determines whether the server updates game state and how long should the server delays itself to update the game state. On the client side, we implemented prediction by updating the board immediately after the client sends out information to the server.

After we’ve implemented the latency mitigation mechanism, we started the testing process by connecting to the server from two more different machines. We then observe the game states on the clients using the same latency patterns that we used for the previous milestone. The result shows that despite ping values remain the same, the game runs more smoothly. We then conclude that most of the latency was successfully mitigated.