Background:

The goal of this project involved writing a DES from scratch using python which would support 3 programs of increasing complexity. The three programs implemented for project are ping, circling and phold. For all three programs, command line arguments and flags can be set to control the variables of the simulation. All three programs output the total number of trips or bounces between Logical processes (LPs).

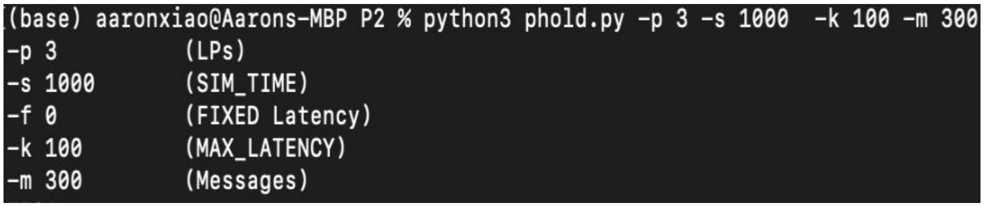
Methods:

This project was designed by using a C implementation of the ping program as reference. All three programs are equipped with the CLI implementation which reads in user chosen input values for the simulation. To do this, I used python argparse library to extract each input from the associated flag before storing them into global variables. The use of the python random library was incorporated into the three programs to make the interval time, the time it takes one LP to send a message to another LP, varied if the latency is not set to a fixed value.

Upon several attempt to generate matplotlib graphs by storing simulation data in a Pandas Data Frame, I was not able to succeed in generating the plot with the simulation data. As a result, the analysis and reflection done on the output of the programs are based on my own assumptions and the numeric data printed by the programs themselves.

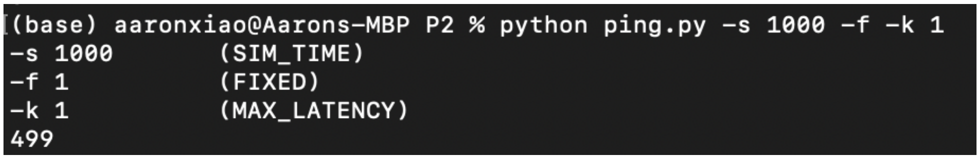
Results:

For all three programs the CLI flags used and the corresponding variables they represent are:



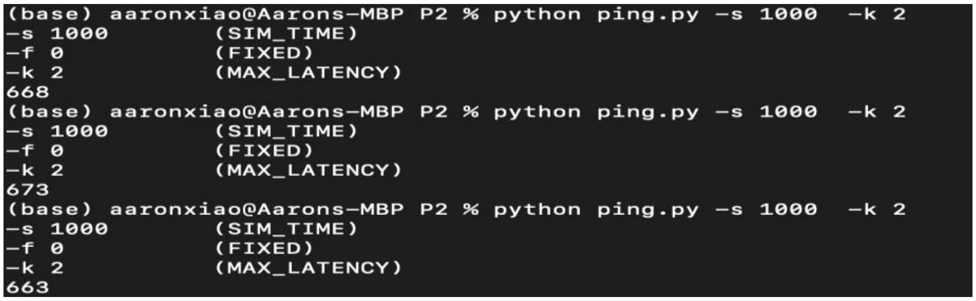
```
[(base) aaronxiao@Aarons-MBP P2 % python3 phold.py -p 3 -s 1000  -k 100 -m 300
-p 3          (LPs)
-s 1000       (SIM_TIME)
-f 0          (FIXED Latency)
-k 100        (MAX_LATENCY)
-m 300        (Messages)
```

Starting off with ping, which outputs the total trips between ping and pong. When the latency is fixed to 2 seconds with a Simulation time of 1000 the result number of trips equated to 499 due to the 500th trip finish at same time as the SIM_TIME causing it not to be counted.



```
[(base) aaronxiao@Aarons-MBP P2 % python ping.py -s 1000 -f -k 1
-s 1000       (SIM_TIME)
-f 1          (FIXED)
-k 1          (MAX_LATENCY)
499
```

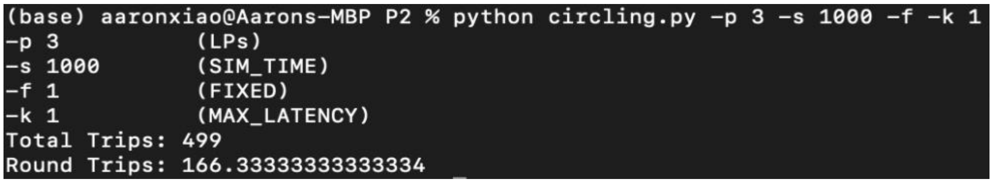
Upon removing the -f flag for fixed latency and setting the upper latency limit to 2, the results became varied with repeated runs due to an inconsistent latency.



```
(base) aaronxiao@Aarons-MBP P2 % python ping.py -s 1000  -k 2
-s 1000        (SIM_TIME)
-f 0           (FIXED)
-k 2           (MAX_LATENCY)
668
(base) aaronxiao@Aarons-MBP P2 % python ping.py -s 1000  -k 2
-s 1000        (SIM_TIME)
-f 0           (FIXED)
-k 2           (MAX_LATENCY)
673
(base) aaronxiao@Aarons-MBP P2 % python ping.py -s 1000  -k 2
-s 1000        (SIM_TIME)
-f 0           (FIXED)
-k 2           (MAX_LATENCY)
663
```

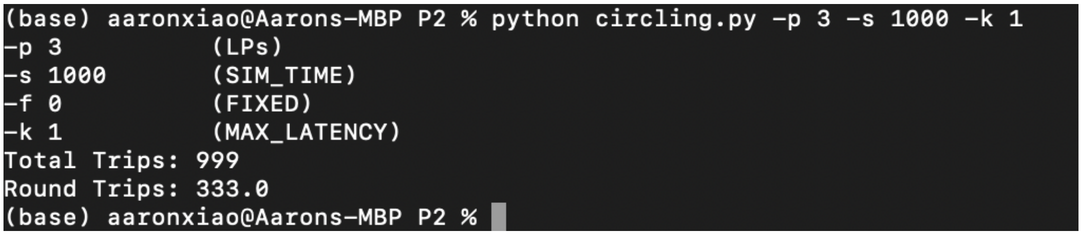
For the slightly more complicated circlying.py, the resulting total number of trips were pretty much equivalent to that of ping. The only difference in these programs is the sequential order at which the bounce message occurs in between the multiple LPs. For circlying.py the total round trips were also calculated and printed so as to differentiate from ping.

Result for circling.py with fixed latency of 2 and 3 LPs:



```
(base) aaronxiao@Aarons-MBP P2 % python circling.py -p 3 -s 1000 -f -k 1
-p 3          (LPs)
-s 1000       (SIM_TIME)
-f 1          (FIXED)
-k 1          (MAX_LATENCY)
Total Trips: 499
Round Trips: 166.33333333333334
```

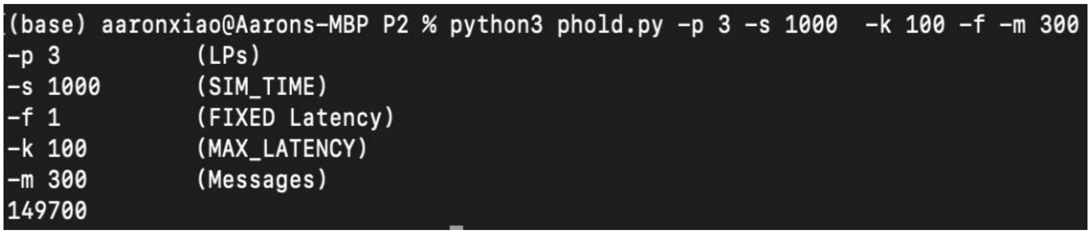
Same result for circling.py with random latency.



```
(base) aaronxiao@Aarons-MBP P2 % python circling.py -p 3 -s 1000 -k 1
-p 3          (LPs)
-s 1000       (SIM_TIME)
-f 0          (FIXED)
-k 1          (MAX_LATENCY)
Total Trips: 999
Round Trips: 333.0
(base) aaronxiao@Aarons-MBP P2 %
```

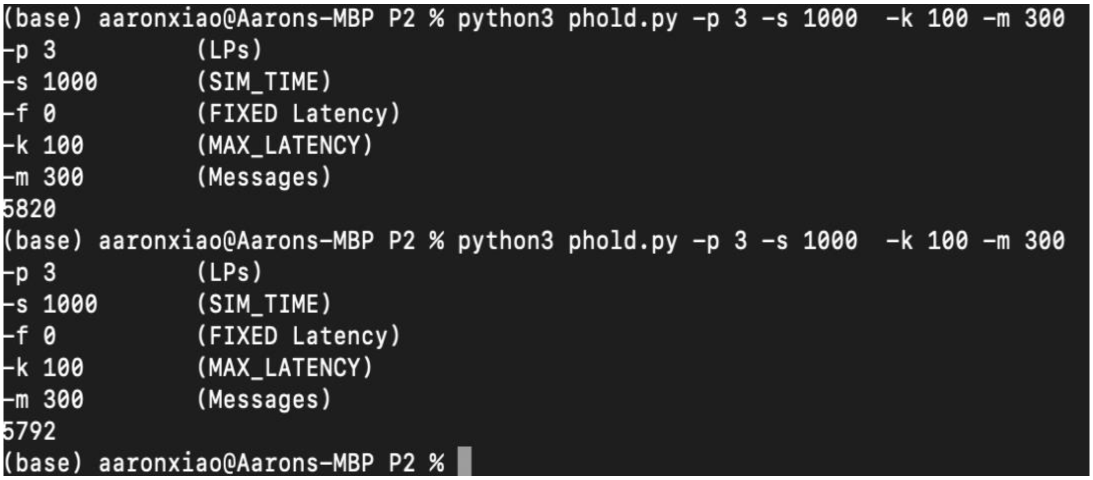
With the most complex simulation phold, the results see a substantial bump in quantity of trips.

Phold results with fixed latency at 2.



```
(base) aaronxiao@Aarons-MBP P2 % python3 phold.py -p 3 -s 1000  -k 100 -f -m 300
-p 3          (LPs)
-s 1000       (SIM_TIME)
-f 1          (FIXED Latency)
-k 100        (MAX_LATENCY)
-m 300        (Messages)
149700
```

Phold with random latency.



```
(base) aaronxiao@Aarons-MBP P2 % python3 phold.py -p 3 -s 1000  -k 100 -m 300
-p 3          (LPs)
-s 1000       (SIM_TIME)
-f 0          (FIXED Latency)
-k 100        (MAX_LATENCY)
-m 300        (Messages)
5820
(base) aaronxiao@Aarons-MBP P2 % python3 phold.py -p 3 -s 1000  -k 100 -m 300
-p 3          (LPs)
-s 1000       (SIM_TIME)
-f 0          (FIXED Latency)
-k 100        (MAX_LATENCY)
-m 300        (Messages)
5792
(base) aaronxiao@Aarons-MBP P2 %
```

Analysis:

By looking at the results of the two simpler simulations ping.py and circling it is evident that the latency value and the number of total trips have a negative correlation. When the latency for ping and circling was changed from being set to the fixed value of 2 and instead to a random value between 1 and 2, the total trips increased for both cases. This result makes sense with the data given since the less time it takes for a trip between two LPs then more trips can be made within that same given time period of the SIM_TIME thus the higher the resulting value for total trips.

Now looking at the results for phold, with the underlying assumption staying the same, the overall output from phold also supports this conclusion. The main difference that phold has with ping and circling is mainly the much larger scale at which trips between LPs are made. For phold N number of messages are dequeued and randomly sent between multiple LPs, each message will continue to travel between LPs until the end of the simulation. Observing the results for phold we see that with the fixed latency of 2 the 300 messages are passed around 149700 times within the span of the simulation. Now looking at the same phold simulation with a random latency value between 1 and 100, the resulting number of trips these messages were able to take were nearly cut to a third. Since the time it takes to travel between LPs increase the messages are not able to be scheduled and sent to as many LPs within the same time limit. Here we once again see a negative correlation between the total number of trips and latency period; the higher the latency the fewer the number of trips completed.