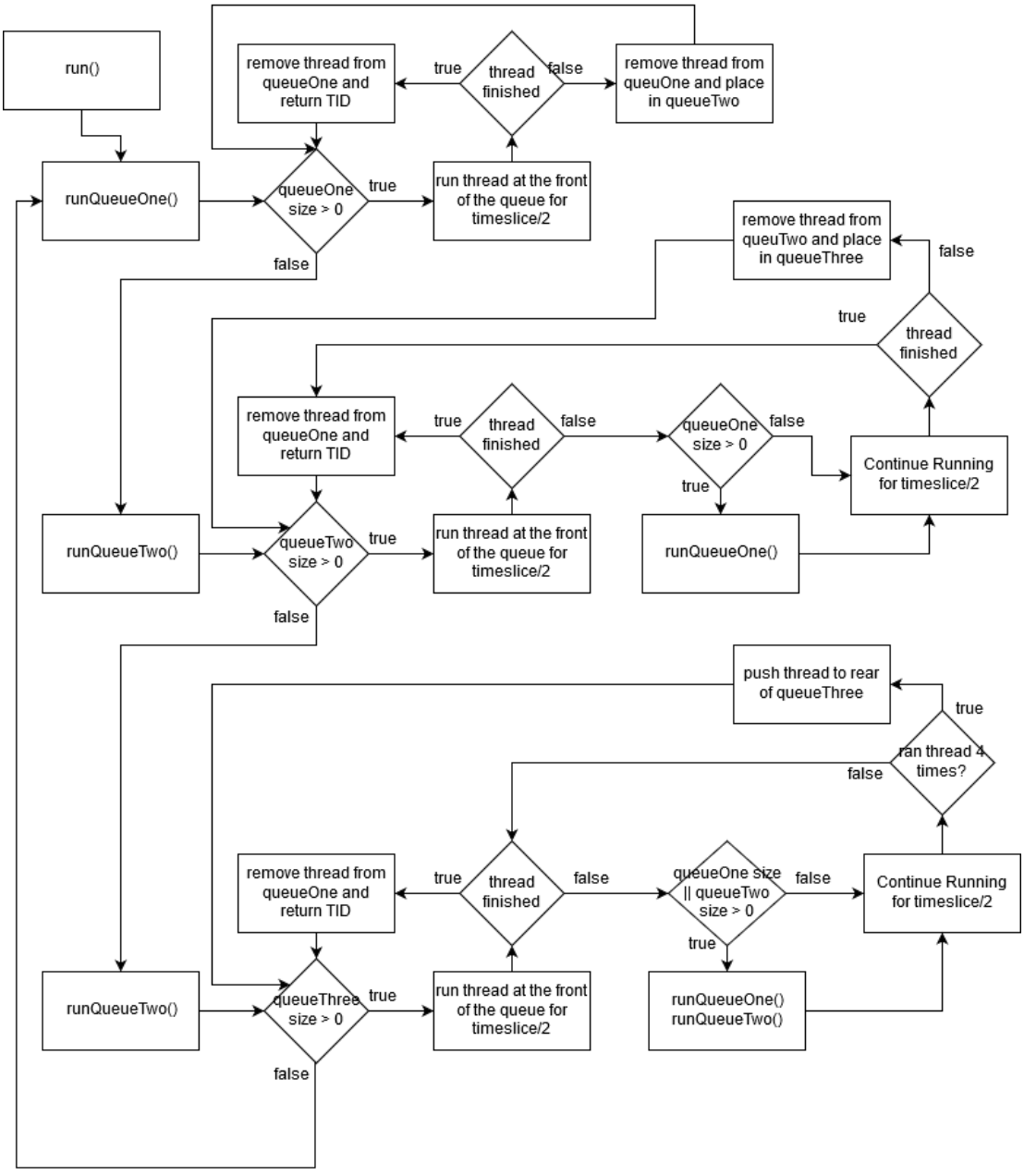
Joshua Landron
CSS430
Program 2

Contents

Algorithm design 1
Comparison of Results 2
Considering First Come First Serve Queue 3
Compiling and testing my implementation 4

Algorithm design

Figure 1: Algorithm flowchart



Comparison of Results

The multilevel feedback queue design had some improvements over the pure round robin queue in my tests. I modified test2 to give 10 results across a larger spectrum of CPU burst times. In my testing over the multilevel feedback queue had much better response times across the board. Turnaround and execution times constantly increased with the round robin queue as CPU burst time increased, while with the multilevel feedback queue the turnaround and execution times had local plateaus where performance did not increase much before increasing quickly after certain thresholds. Figures 2, 3, and 4 show these times revealing the large differences in times.

Figure 2: Round Robin Queue Results

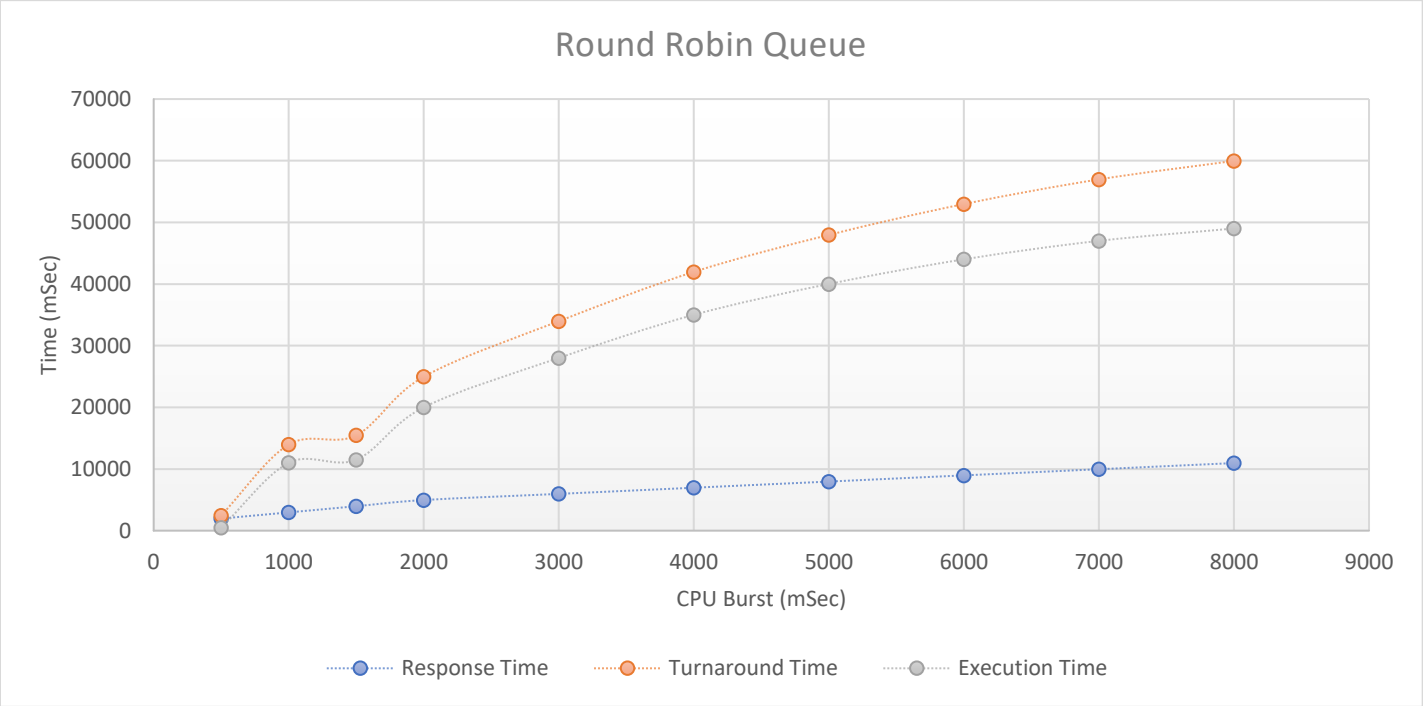
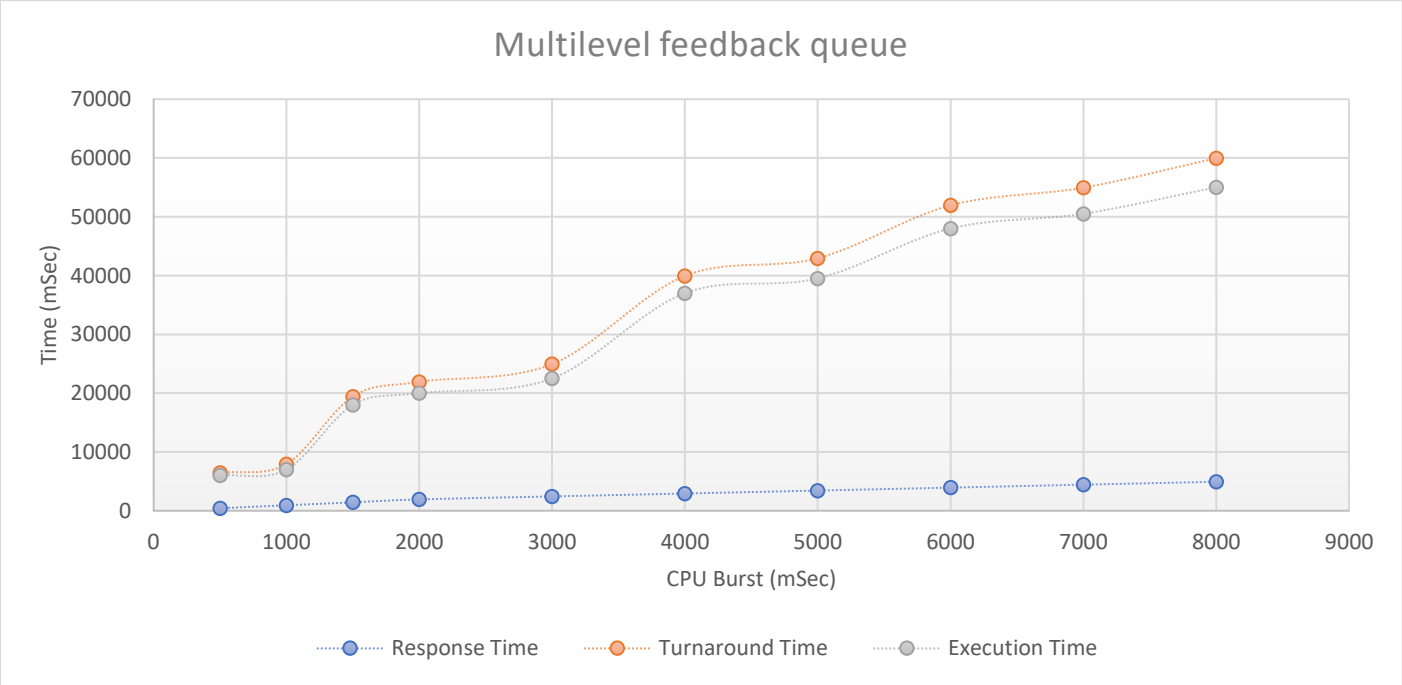


Figure 3: Multilevel Feedback Queue Results



These plateaus and rapid increases in the Turnaround and Execution times in the multilevel feedback queue show that short CPU burst times (<3000 mSec) tended to have more consistent times. This is due to the algorithm consistently

checking the top level queues. However this algorithm can be improved upon by checking if the processes are finished after every timeslice/2. The results of this change are shown in Table 3.

Figure 4: Optimized Multilevel Feedback Queue

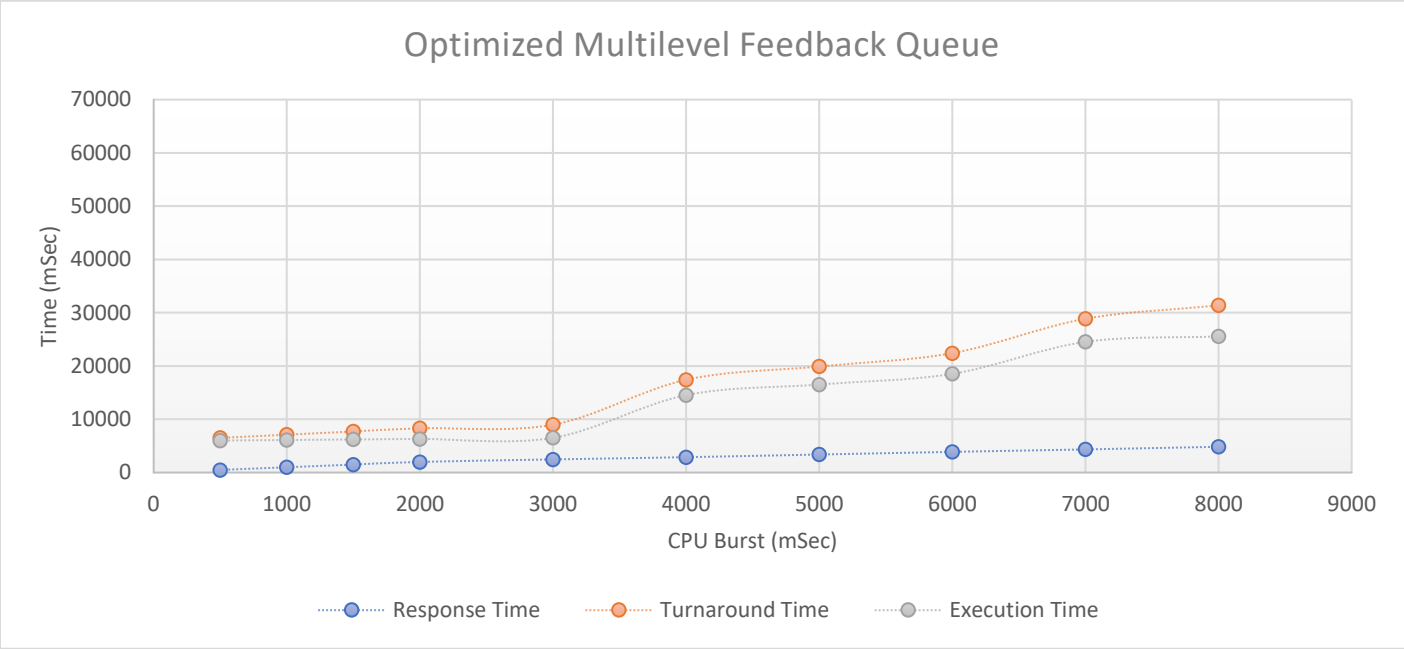


Table 3 shows that checking for process completion every time the scheduler checks the higher-level queues and before pushing the process into the queue to wait again substantially improves performance over both the previous iteration of the MLFQ and the Round-Robin style single queue.

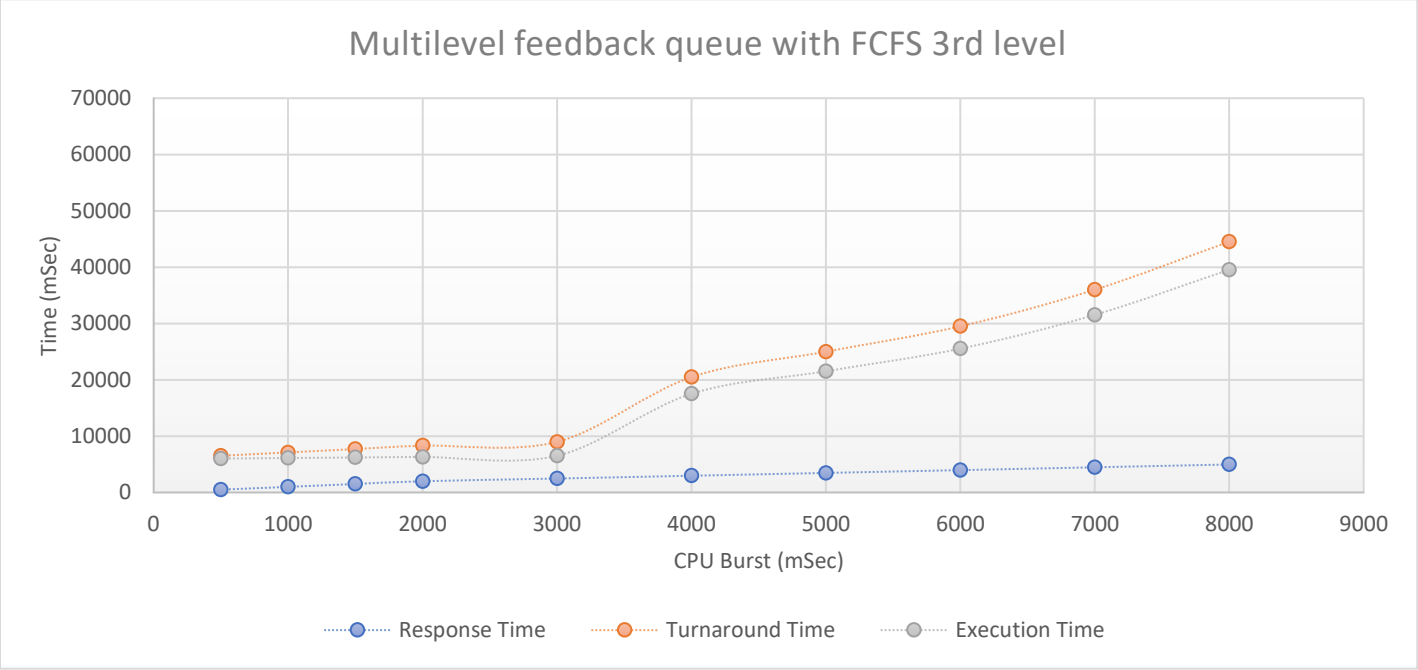
Considering First Come First Serve Queue

If the Multilevel feedback queue implemented a first come first serve queue third level queue the overall execution times would be lower at first for threads that last long enough to get into the third level queue. As threads that have longer CPU burst times start to build up in the third level queue the turnaround time for each of them will start to increase much faster than with either round robin or Multilevel feedback queue.

To test my theory, I implemented a first come first serve third queue and the results are presented in Table 3. While my first come first serve technically has a small round robin for the first two elements in the queue, it behaves much as I would have expected.

I had to implement the first come first serve with swapping first element and the second element due to the fact that I was running into threads that did not ever terminate. I believe this is the kernel thread and therefore just put inserted the working thread in the third queue into the second position in line and kept swapping this position as long as there was more than one thread in the queue. This simple implementation is prone to bugs such as if I attempt to launch programs from the P0 Shell the queue can get stuck waiting on two base programs to finish that rely on other threads.

Figure 5: Multilevel Feedback Queue with FCFS results



Compiling and testing my implementation

My .zip file contains 4 files.. These are the separate Scheduler.java files for both parts of this assignment and an example of each of the implantations output in a .txt file.

To compile my implementation simple open the Linux terminal, navigate to the folder containing the ThreadOS binaries that are currently executable (test this before compiling new Scheduler). Run the base ThreadOS with Test2 to check for ThreadOS errors before introducing new code.

Next replace the Scheduler.java file with the Scheduler.java file located in folder P2_part1. This is the pure round robin queue implementation. Compile this code with:

javac -deprecation Scheduler.java

Once this is compiled Boot ThreadOS with:

java Boot

and run test2 in ThreadOS with:

l Test2

Once this test is complete type q and press enter to quit threadOS.tre

Finally replace the Scheduler.java file with the Scheduler.java file located in folder P2_part2. This is the multilevel feedback queue implementation. Compile this code with:

javac -deprecation Scheduler.java

Once this is compiled Boot ThreadOS with:

java Boot

and run test2 in ThreadOS with:

l Test2

Once this test is complete type q and press enter to quit threadOS.