Project Report

Matthew Williams

**Average Turnaround:** There are two things that stand out here. First is the massive jump that RR 0.01 has over everything else. The second thing is the early jump that RR 0.2 has. The second one can be attributed to some error in the code, I believe since it happens every time I run lambda of 4. The first one though must be attributed to the fact that the quantum is so small while the ready queue fills up meaning it will take longer and longer for a process to finish, even with a small service time. Could also be an error on my side since we know RR 0.2 has one. The rest of the schedulers generally increase in turnaround except for STRF. I think that makes sense given that the smaller jobs will go first which might skew the data towards a smaller turnaround.

**Total Throughput:** I think this graph is most interesting one I made. This one more clearly shows the differences between the schedulers. It appears that having a higher quantum decreases the efficiency very early one, relative to the service time. The lower quantum makes it much further than the higher one before it bogs down. This goes along with an increase in queue size. Surprisingly FCFS and HRRN output more processes. This makes me wonder if something went wrong in the code (other than the fact that they are the exact same). Finally STRF has a roughly linear growth which appears to be about the same as its ready queue growth.

**CPU Utilization:** This one is pretty cut and dry. All the processes have to same utilization, but that is no surprise given that we calculated lambda \* service time for them all and they all had the same values of lambda and service time.

**Average Ready Queue Size:** This is the last graph has mainly one thing sticking out: The size of the ready queue for RR 0.2. This value starts increasing very fast early on. In fact I didn’t even include the last four values of lambda for it since it increased so much. I guess this makes since given that it is basically like FCFS with the quantum size, but the newest process will go first. RR 0.01 shows much slower growth, but it is still the second highest. STRF still shows small increases in its growth. FCFS and HRRN again show the same behavior for some reason and have the lowest average queue size. The data could be skewed due to how I process new arrivals.

**Conclusion:** Based on my data (which I am not feeling so good about now) both FCFS and HRRN seem to be the best schedulers. Maybe it has to do with the service time of the data or how I handle arrivals in my code or even possibly how I collect the data. While playing around with the data I have noticed that the higher the service time the more the schedulers begin the swap their rankings, but that does not rule out errors.