Student: Jose Gutierrez
Class: Operating Systems
Professor: Mina S. Guirguis
Date: 10/30/2016

Discrete-Time Event Simulator

Running the simulator:

Inside the sim folder there is a makefile that will build and compile the simulator. The executable that it will create is sim. To remove executable files after execution of the program a clean option is included in the makefile.

Sample run:

```
Enter the number of the arrivals (per second)range you want to start the test with:
1
Enter the number of the arrivals (per second)range you want to end the test with:
30
Enter the average arrivalrate (per second) for the test:
.06
Select Scheduler to Run
1 - First Come First Serve
2 - Shortest Job First:
3 - HRRN
4 - Round Robin:
5 - Exit
1
Run Again?
1: YES
2: NO
2
Program ended with exit code: 0
```

(Ran and tested on eros.cs.txstate.edu)
The following commands will run the program for 30 times starting with 1 arrival per second and ending 30 arrivals per second with an average service time of .06 seconds using the scheduler 1 (FCFS).

Project scope:

The simulation of process-scheduling for tasks which have been admitted into a ready queue using different algorithms in an event-driven system.

Algorithms Tested:

- First Come First Serve (FCFS).
- Shortest Remaining Time First (SRTF).
- Highest Response Ratio Next (HRRN).
- Round Robin (RR).

Data Structures Used:

- Linked List.

Basic Components

Simulation Clock: The clock is the key variable in the simulation. Since arrivals, departures, and time slice events all have a corresponding time attached to them and they are able to notify the system of the current state of the simulation. The clock is only updated at the time of event occurrences.

Event Queue: The event queue class constructs a linked list and implements operations to maintain it. This class holds a nested class called event. The event class is used to represent the individual events that represent the nodes in the event queue.

Ready Queue: The ready queue class constructs a linked list and implements operations to maintain it. It models the processes that have been admitted into the queue and are eligible for selection. This class holds a nested class called process. The process class creates the processes that represent the nodes in the ready queue.

Simulator: The simulator class allows the user to run the tests for a number of times. It coordinates the event creation and admission into the ready queue. In addition, it records the statistics that will be helpful when analyzing the behavior of the algorithms.

Driver: Used to display a menu and parse the arguments the user has selected. The arguments selected by the user are the number of arrivals (lambda) the user wants to run the test for and the average service time, which will follow an exponential distribution.

Experimental Method

Arrivals are scheduled following a poisson distribution. New arrivals are scheduled when an old arrival event is done being acknowledged by the CPU.

First Come First Serve: FCFS is a service policy whereby the requests of processes are attended to in the order that they arrived. If the cpu is busy the process will be placed in the ready queue. Otherwise, the departure of the event will simply be scheduled since the departure time will be known.
Placement in the ready queue: All process admitted into the ready queue will be placed at the tail of the queue. A tail pointer is kept to avoid traversal of the linked list.
Departure from the ready queue: The process in front of the queue will be the next process to leave the queue. Prior to deletion of the the process a new event departure is scheduled.

Shortest Remaining Time First: SRTF is a service policy in which the process with the least amount of burst time remaining are selected to run next. Since this is a preemptive algorithm, all

processes will be placed on the ready queue without any exemption. A process is not guaranteed to be done executing if it is selected to run and therefore it is needed to remain in the ready queue until finished. This is the difference between nonpreemptive and preemptive algorithms, hence the reason a departure of a process can be scheduled as soon as we process an arrival if the cpu is not busy with the FCFS AND HRRN algorithms.

Placement in the ready queue: The processes are placed arbitrarily at the tail of the queue. Since any arrival of a process can change the priority of events, it is best to simply avoid assigning a priority to a process at the time of insertion, and instead simply traverse through the list one time when a process is selected to be run.

Departure from the ready queue: Knowing the interarrival times is particularly helpful for this algorithm. Once traversing the list is completed a pointer “highest” points to the highest node in the list. In this case, the node with largest remaining time. There is two options: If the interarrival time is higher than the current highest service time, a departure is scheduled, the interarrival time is reduced by the highest service time and repeat the process. If the interarrival time is less than the highest service time, the highest service time is reduced by the interarrival time, and we process the next event arrival or departure. Note: that preemption occurs on every arrival, hence it is not needed to create a preemption event.

Highest Response Ratio Next: HRRN is a non preemptive scheduling algorithm in which the priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Similarly to FCFS departures can be scheduled without placement in the ready queue if the cpu is currently off. The only difference between the implementation of these policies in the simulator is the departure from the ready queue.

Placement in the ready queue: Place at the tail of the queue to avoid traversal.

Departure from the ready queue: Traverse through all the nodes one time and find the highest ratio, once the highest ratio is found, a pointer to this node is placed. The traversal continues until the end of the list. The departure is scheduled.

Round Robin: RR. During this algorithm time slices are assigned to each process in equal portions and in circular order. Every time slice represents a new event in the simulation. On any arrival or a time slice event the ready queue is checked and it allocates every process a quantum of time. This is stored in a variable called timeLeft, which is initialized to the original service time. On any time slice event the quantum is subtracted from the time left until the quantum is greater than the time left. At this time a departure will be scheduled. During every execution of a time slice event the head pointer is maintained (as described above) and immediately placed at the back of the queue, allowing for a circular motion.

Placement in the ready queue: All processes are inserted at the back of the queue. Since the algorithm behaves on a FCFS on new arrivals to the queue.

Departure from the ready queue: Described above.

Performance Metrics

- The average turnaround time.
- The total throughput (number of processes done per unit time).
- The CPU utilization .
- The average number of processes in the ready queue.

Testing Metrics

- Lambda 1 - 30.
- 10000 Departure events.
- Average Service Time: .06

Results

FCFS:

Scheduler	Quantum	Lambda	Utilization	Throughput	Average Number of processes in the Queue	Average Waiting Time in the Queue	Turn Around Time
1	0	1	0.0604856	1.0001	0	0	0.0604797
1	0	2	0.121471	2.0002	0.0003	1.51367E-05	0.0607447
1	0	3	0.177242	3.0002	0.0047	0.000308978	0.0593854
1	0	4	0.242646	4.00036	0.0185	0.00114467	0.061801
1	0	5	0.30018	5.00024	0.0406	0.00238534	0.0624192
1	0	6	0.361432	5.99985	0.0862	0.00486484	0.065105
1	0	7	0.409079	7.0003	0.1278	0.00728718	0.0657245
1	0	8	0.476347	8.0004	0.2196	0.0133792	0.0729197
1	0	9	0.54493	9.00012	0.3335	0.0199094	0.0804559
1	0	10	0.595913	9.99984	0.4827	0.0291282	0.0887203
1	0	11	0.653293	10.9972	0.697291	0.0423784	0.101784
1	0	12	0.725446	12.0006	1.0484	0.0638567	0.124308
1	0	13	0.781628	12.9922	1.40885	0.0837346	0.143896
1	0	14	0.8345	13.9997	2.1151	0.12491	0.184519
1	0	15	0.898438	15	4.06	0.242982	0.302879
1	0	16	0.936334	15.9756	6.30561	0.365946	0.424556
1	0	17	0.999796	16.3855	155.449	9.04338	9.10441
1	0	18	0.997209	16.8557	322.838	17.8425	17.9017
1	0	19	0.999834	16.6799	687.674	36.1058	36.1658
1	0	20	0.999936	16.5475	1024.46	50.9952	51.0556
1	0	21	0.999874	16.5192	1334.3	63.2231	63.2835
1	0	22	0.999899	16.8719	1546.51	70.6339	70.6931
1	0	23	1	16.4703	1930.7	83.0167	83.0775
1	0	24	0.99997	16.8933	2119.58	88.5753	88.6344
1	0	25	1	16.7548	2440.67	97.1909	97.2506
1	0	26	0.999819	17.0015	2633.14	100.984	101.042
1	0	27	1	16.697	3120.93	116.37	116.43
1	0	28	0.9999	16.9875	3256.23	116.62	116.679
1	0	29	0.999906	16.5443	3727.05	127.54	127.601
1	0	30	0.999998	17.0834	3743.34	123.844	123.903

SRTF:

Scheduler	Quantum	Lambda	Utilization	Throughput	Average Number of processes in the Queue	Average Waiting Time in the Queue	Turn Around Time
2	0	1	0.0589322	1.00008	0	-1.15669E-08	0.0589275
2	0	2	0.122665	2.00019	0	-6.21787E-07	0.0613267
2	0	3	0.180157	3.00019	0.0028	7.07189E-05	0.0601192
2	0	4	0.240385	4.0003	0.0154	0.000463203	0.0605545
2	0	5	0.30323	5.00015	0.0446	0.00133748	0.0619809
2	0	6	0.367982	5.99979	0.079592	0.00238996	0.063709
2	0	7	0.423969	7.00006	0.1309	0.00415627	0.064723
2	0	8	0.477386	8.00078	0.191	0.00625053	0.0659219
2	0	9	0.534237	8.99987	0.255474	0.00827108	0.067641
2	0	10	0.597157	9.99844	0.345631	0.0116102	0.0713397
2	0	11	0.659168	10.9997	0.463954	0.0163535	0.0762922
2	0	12	0.711572	11.9972	0.604819	0.0224898	0.0818164
2	0	13	0.762269	13.0005	0.7202	0.0265839	0.0852517
2	0	14	0.811617	13.9998	0.9567	0.0377241	0.0957197
2	0	15	0.876788	14.9985	1.38286	0.0589734	0.117463
2	0	16	0.946383	15.9947	2.67383	0.127702	0.18689
2	0	17	0.991925	16.8929	30.1405	0.451024	0.509747
2	0	18	1	17.721	79.636	0.50988	0.566307
2	0	19	0.99921	18.5867	101.266	0.494281	0.548039
2	0	20	0.99971	19.1325	216.56	0.483483	0.535728
2	0	21	1	19.8339	294.324	0.752771	0.803179
2	0	22	1	20.4992	386.042	0.81257	0.861346
2	0	23	0.999997	21.2748	416.883	0.672475	0.719474
2	0	24	0.999846	21.9539	466.146	0.395483	0.441021
2	0	25	1	22.6287	536.203	0.505204	0.549394
2	0	26	0.999938	23.0362	636.166	0.508168	0.551573
2	0	27	1	23.6497	702.947	0.485432	0.527712
2	0	28	1	24.2018	774.158	0.474872	0.516187
2	0	29	0.999923	24.7514	861.907	0.551796	0.592193
2	0	30	1	25.4487	904.196	0.45978	0.499073

HRRN:

Scheduler	Quantum	Lambda	Utilization	Throughput	Average Number of processes in the Queue	Average Waiting Time in the Queue	Turn Around Time
3	0	1	0.0596395	1.0001	0	0	0.0596385
3	0	2	0.121938	2.00015	0.0001	5.51758E-06	0.0609693
3	0	3	0.179582	3.00003	0.0044	0.000275652	0.060135
3	0	4	0.23823	4.00033	0.0152	0.000932544	0.0604856
3	0	5	0.298632	5.00028	0.043	0.0025822	0.0623052
3	0	6	0.360833	5.99989	0.0835	0.00471248	0.0648524
3	0	7	0.422082	7.00033	0.1399	0.00802865	0.0683234
3	0	8	0.48282	8.00071	0.2217	0.0129868	0.0733342
3	0	9	0.542467	9.0002	0.3147	0.0178556	0.0781282
3	0	10	0.595572	10.0005	0.4382	0.0249831	0.0845375
3	0	11	0.675543	10.9984	0.671066	0.0386033	0.100025
3	0	12	0.711076	12.0007	0.8398	0.0472866	0.10654
3	0	13	0.782008	13	1.1827	0.0658492	0.126003
3	0	14	0.842012	13.997	1.78434	0.100702	0.160858
3	0	15	0.916117	14.9993	3.51315	0.205476	0.266553
3	0	16	0.969609	15.9821	6.70216	0.388977	0.449646
3	0	17	0.999328	16.6826	119.393	6.90888	6.9688
3	0	18	0.999671	17.2349	239.057	12.7692	12.8271
3	0	19	0.998846	17.7693	345.221	16.9228	16.979
3	0	20	0.999819	18.4571	429.91	19.7083	19.7624
3	0	21	1	18.7964	581.118	24.5291	24.5823
3	0	22	0.999852	19.2935	702.868	27.4459	27.4977
3	0	23	0.999872	19.8114	799.785	29.189	29.2394
3	0	24	0.999952	20.3024	898.866	30.4452	30.4945
3	0	25	0.999822	20.9587	945.237	30.3703	30.4179
3	0	26	1	21.2643	1085.79	33.0776	33.1246
3	0	27	0.999936	21.8481	1180.26	33.5216	33.5674
3	0	28	0.999938	22.2724	1263.19	34.8062	34.8511
3	0	29	1	22.764	1354.16	34.5319	34.5759
3	0	30	0.999985	23.2615	1445.36	35.1229	35.1659

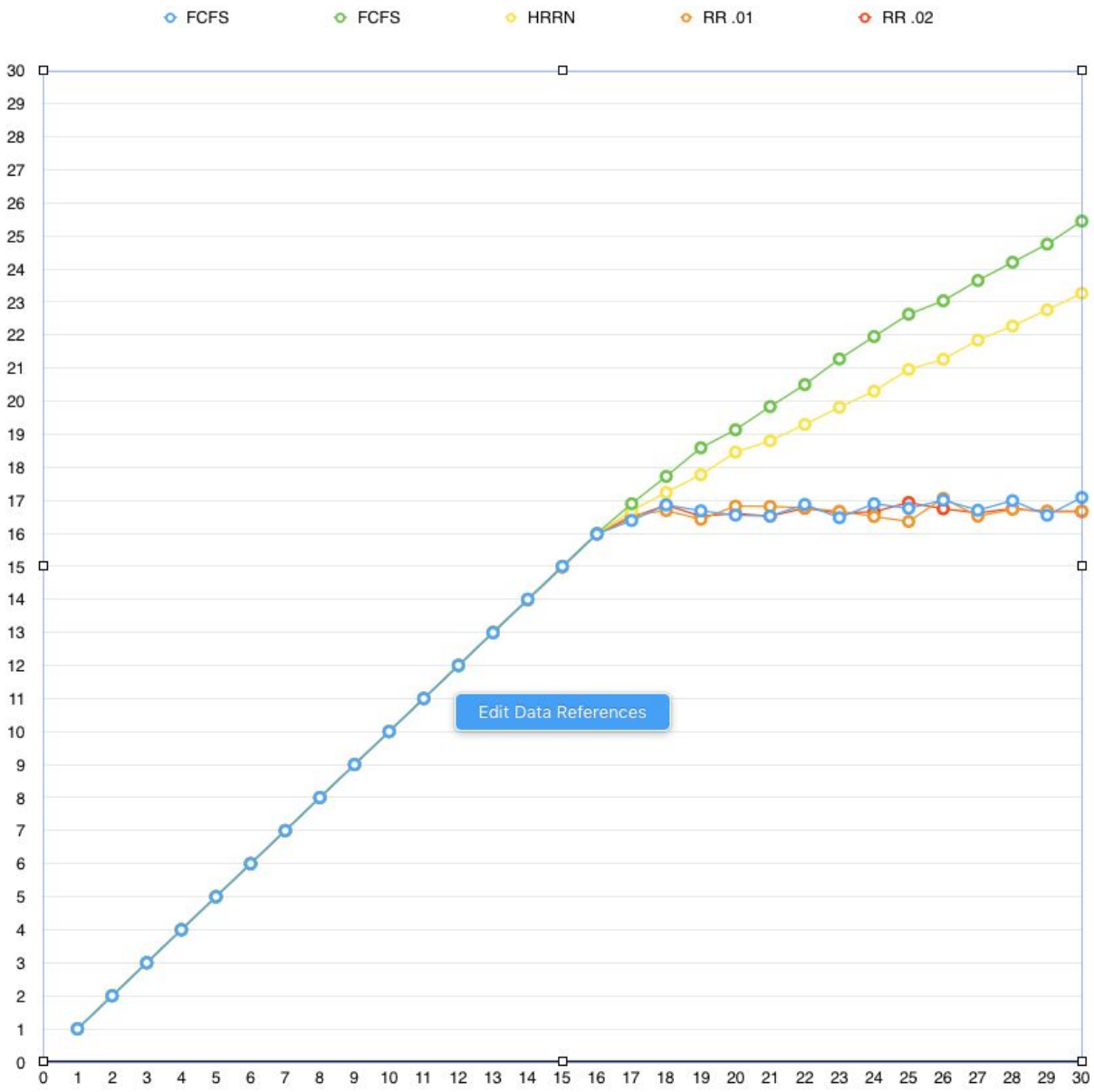
RR (.01):

Scheduler	Quantum	Lambda	Utilization	Throughput	Average Number of processes in the Queue	Average Waiting Time in the Queue	Turn Around Time
4	0.01	1	0.0593763	1.0001	0.8482	3.19628E-05	0.0601515
4	0.01	2	0.120115	2.00019	0.8479	5.70041E-05	0.0603055
4	0.01	3	0.179339	3.00019	0.8464	0.000266014	0.0601948
4	0.01	4	0.236831	4.00028	0.8619	0.00103359	0.0602904
4	0.01	5	0.295788	5.00021	0.8925	0.0023306	0.0619342
4	0.01	6	0.352682	6.00037	0.9343	0.0050186	0.0656139
4	0.01	7	0.394454	7.00037	0.9925	0.00920914	0.0685941
4	0.01	8	0.443538	8.00075	1.0761	0.0132508	0.0732506
4	0.01	9	0.478553	8.99959	1.15748	0.0179582	0.077017
4	0.01	10	0.568176	9.999	1.35926	0.0290612	0.0888662
4	0.01	11	0.584089	11.0004	1.5429	0.0403749	0.100868
4	0.01	12	0.639143	11.9997	1.7994	0.054328	0.11413
4	0.01	13	0.687382	12.9861	2.44721	0.0953989	0.154826
4	0.01	14	0.714906	13.9751	3.21831	0.139043	0.199179
4	0.01	15	0.752613	14.9797	4.93978	0.24059	0.300503
4	0.01	16	0.71328	15.9835	9.86984	0.531878	0.591394
4	0.01	17	0.995043	16.5436	134.541	7.62803	7.68678
4	0.01	18	0.999384	16.6863	406.989	21.0979	21.1536
4	0.01	19	0.998483	16.4215	768.977	35.9733	36.0265
4	0.01	20	0.999966	16.8231	947.644	42.108	42.1591
4	0.01	21	0.999969	16.8133	1258.17	51.8823	51.931
4	0.01	22	1	16.7669	1505.44	56.7998	56.8464
4	0.01	23	1	16.666	1929.34	69.5658	69.6104
4	0.01	24	0.999822	16.5049	2207.04	72.7372	72.7804
4	0.01	25	0.99991	16.3557	2642.12	82.9786	83.0204
4	0.01	26	1	17.0583	2615.52	79.5689	79.6093
4	0.01	27	1	16.5165	3121.81	87.7782	87.8171
4	0.01	28	1	16.7232	3346.53	90.6138	90.6511
4	0.01	29	1	16.674	3688.3	96.1634	96.2001
4	0.01	30	1	16.6739	3998.83	99.5643	99.5996

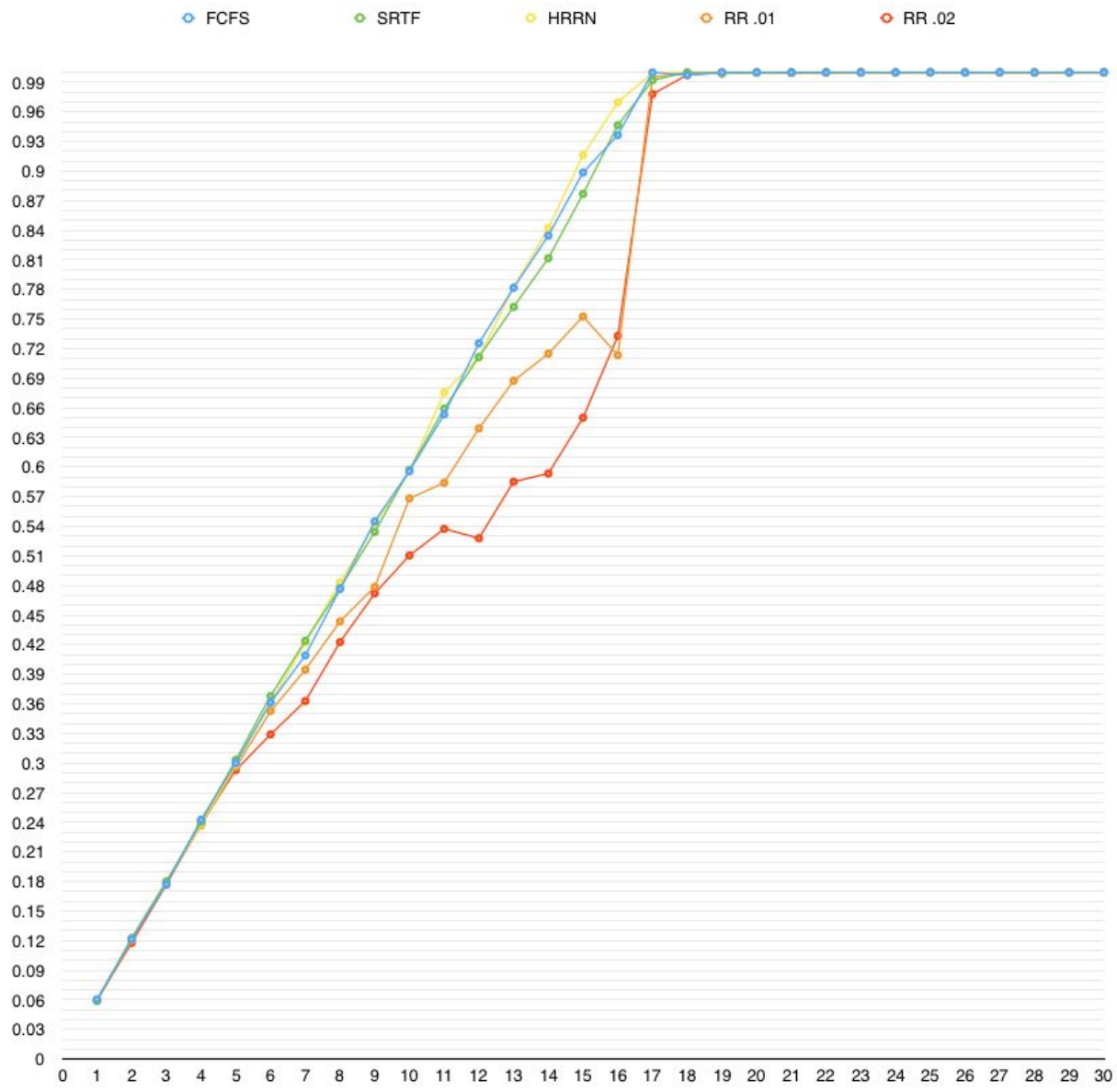
RR(.02):

Scheduler	Quantum	Lambda	Utilization	Throughput	Average Number of processes in the Queue	Average Waiting Time in the Queue	Turn Around Time
4	0.02	1	0.059837	1.00009	0.7161	6.54091E-05	0.0600793
4	0.02	2	0.11778	2.00016	0.7104	7.33108E-05	0.0589177
4	0.02	3	0.177074	3.00008	0.7247	0.00028366	0.0597888
4	0.02	4	0.238932	4.00034	0.7426	0.00109299	0.0622571
4	0.02	5	0.292928	5.00025	0.7706	0.0024045	0.061878
4	0.02	6	0.329052	6.0004	0.7993	0.00467433	0.0641358
4	0.02	7	0.362809	7.00001	0.8465	0.00774975	0.0675226
4	0.02	8	0.422572	7.99958	0.951805	0.0130575	0.0741129
4	0.02	9	0.471897	9.00061	1.065	0.0190317	0.0791043
4	0.02	10	0.510385	10.0004	1.2041	0.0258709	0.0856777
4	0.02	11	0.537371	10.9992	1.44416	0.041528	0.101538
4	0.02	12	0.527812	12.0005	1.6475	0.0541514	0.112777
4	0.02	13	0.585121	13.0008	2.38	0.0970575	0.157137
4	0.02	14	0.593446	13.9931	3.87916	0.194476	0.254543
4	0.02	15	0.650013	14.9786	4.84302	0.240653	0.300419
4	0.02	16	0.732858	15.9779	9.0027	0.483362	0.542548
4	0.02	17	0.977923	16.4789	136.742	7.7361	7.7951
4	0.02	18	0.997304	16.8505	368.166	19.6131	19.6693
4	0.02	19	0.999945	16.517	738.541	35.3888	35.4426
4	0.02	20	0.999993	16.5956	1030.81	46.0051	46.0565
4	0.02	21	0.999411	16.5193	1290.44	52.5897	52.6391
4	0.02	22	0.999604	16.7593	1584.37	62.2928	62.3405
4	0.02	23	1	16.6107	1944.32	72.0628	72.109
4	0.02	24	0.999945	16.6488	2249.76	79.3795	79.4242
4	0.02	25	0.999943	16.9309	2407.04	80.3464	80.3897
4	0.02	26	0.999872	16.7403	2740.6	85.8413	85.883
4	0.02	27	1	16.6167	3095.59	91.5056	91.5458
4	0.02	28	0.99971	16.7466	3361.02	97.2896	97.3291
4	0.02	29	1	16.6529	3703.21	102.051	102.09
4	0.02	30	0.999959	16.6571	3994.36	105.486	105.524

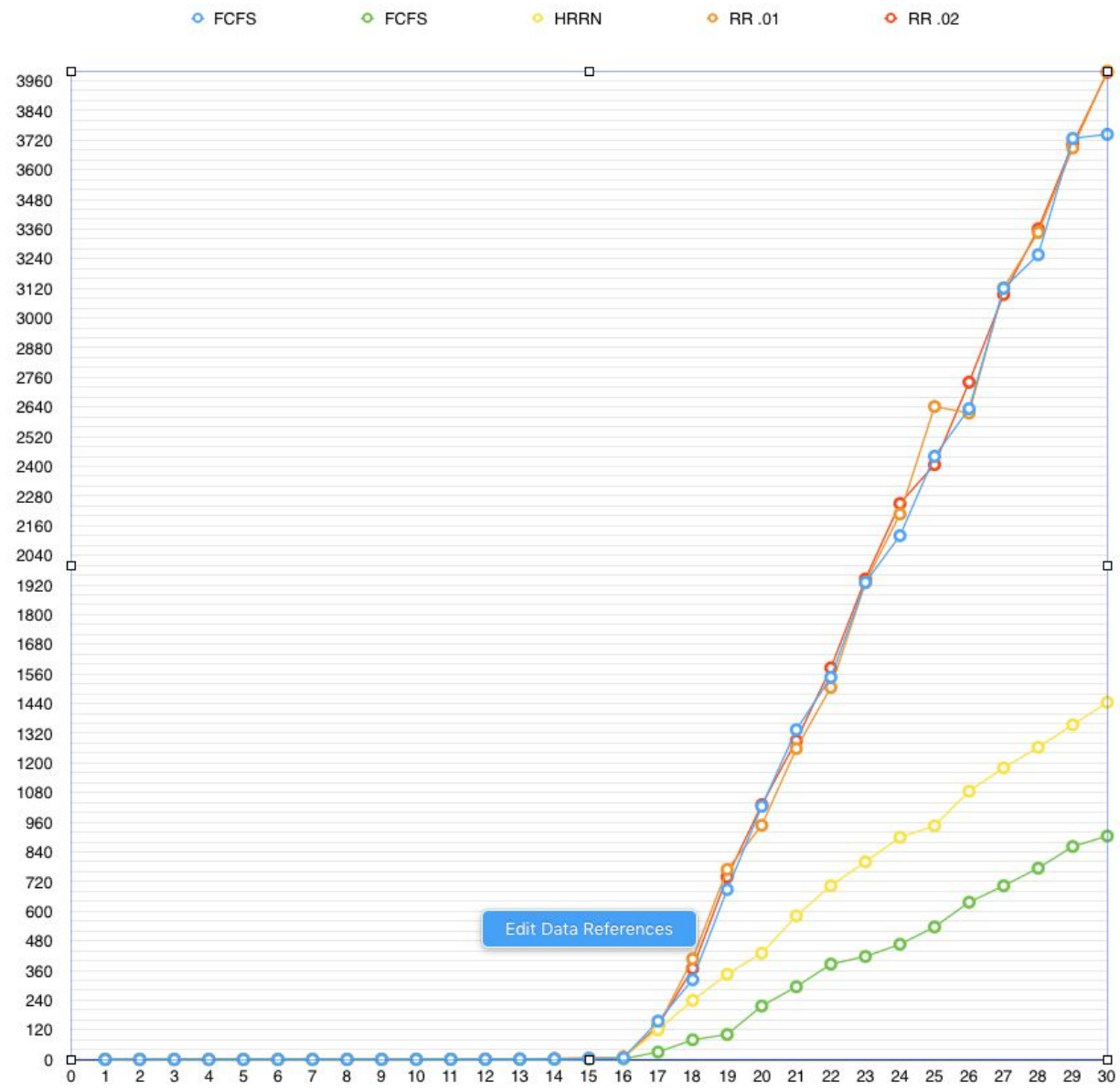
Throughput:



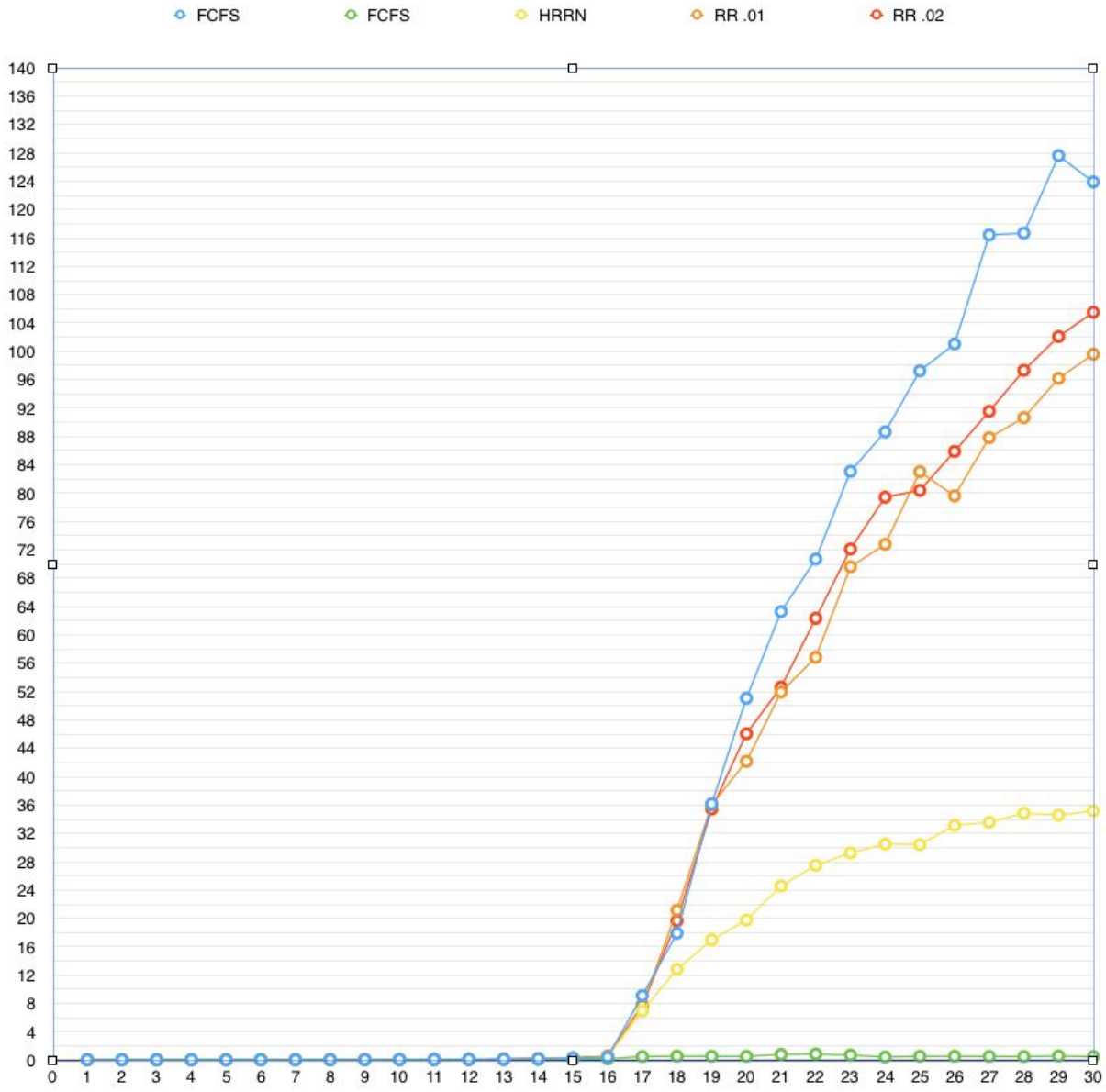
CPU Utilization:



Average Number of Processes Waiting for Execution in the Ready Queue:



Turn Around Time:



Short Observations and Findings

FCFS: Very easy to implement and understand. Poor in performance, however. The periods of waiting time are long and it causes starvation for those processes arriving last.

SRTF: It appears to be the most optimal. Since this algorithm favorites processes with short burst times, it is able to accomplish more tasks per second. **Note, however that:** The average waiting times seem to be very low, but this is in part to the fact that we are picking processes with short burst times, but there are processes waiting in the queue that have been waiting for a very long time. This algorithm is optimal for running short processes first, but will cause starvation significantly.

HRRN: *Does not cause starvation. Performs the best if starvation is a concern.*

RR: The longer the quantum times, the more it starts behaving like FCFS. In this case the closer the quantum time is to .06 it will behave more like FCFS. RR also keeps the most number of processes in the ready queue waiting.