2021-2022 Academic Year
Spring Semester

CS 342 PROJECT 2

Ahmet Hakan Yılmaz 21803399

Akın Kutlu 21803504

Introduction

We implemented multithreaded scheduling simulation using mutex and condition variables to simulate different scheduling algorithms. Here are the graphs of the results we get from our tests. Before starting tests we make sure that our program is working in minimum and maximum conditions. In our tests, we used < FCFS INF 50 100 uniform 50 10 100 0.5 0.3 0.2 0.5 10 30 1 > parameters as our basic parameters to get consistent data and to make comparisons more consistently. We selected our parameters like that as we thought they are close to optimal values for our simulation. We changed the algorithm and quantum time parameters of the input to get consistent results with algorithms. Although we made tests for every possible input we did not include them in our report as we thought the graphs we put is enough for algorithm comparison. For example, for each algorithm, we also tested with fixed and exponential distribution and with changing other parameters too.

The basic configuration:

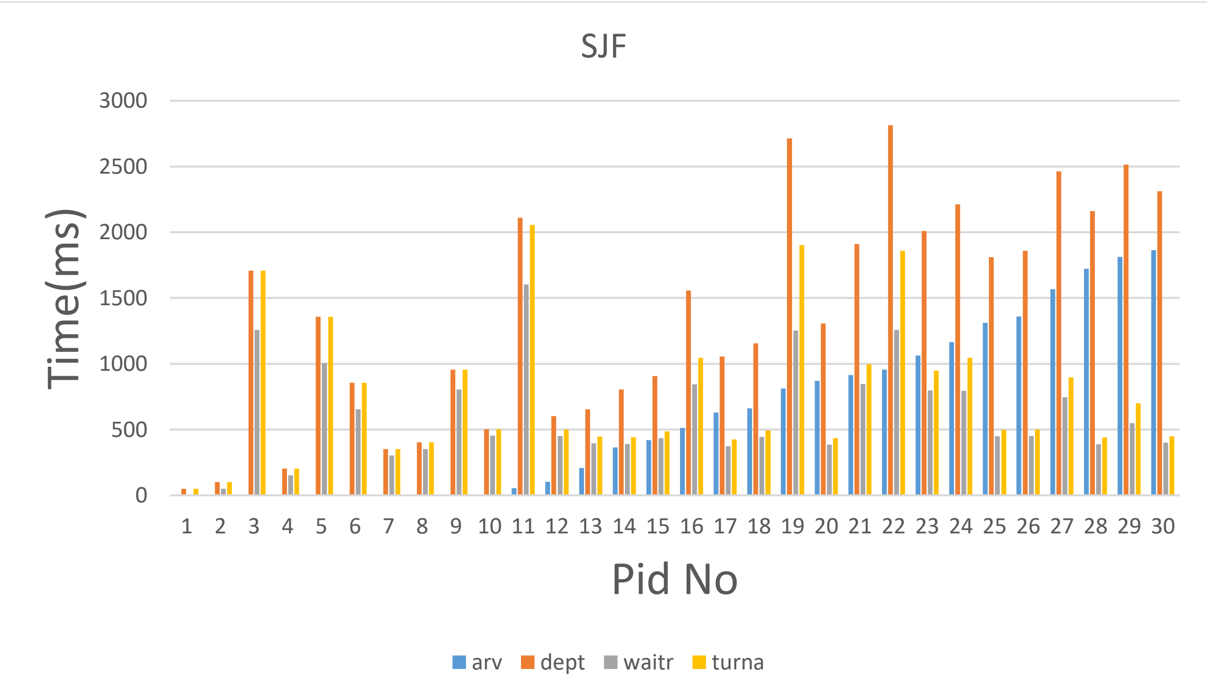
ALG:	FCFS	**Q:**	INF	**T1:**	50
T2:	100	**burst-dist:**	uniform	**Burstlen:**	50
min-burst:	10	**max-burst:**	100	**p0:**	0.5
p1:	0.3	**p2:**	0.2	**pg:**	0.5
MAXP:	10	**ALLP:**	30	**OUTMODE:**	1

Graphs for 30 processes according to different algorithms

First of all, we calculated, the arrival, dept, turnaround, and wait times for the first 30 processes terminated according to different algorithms and when the algorithm is RR according to different quantum times.

Shortest Job First

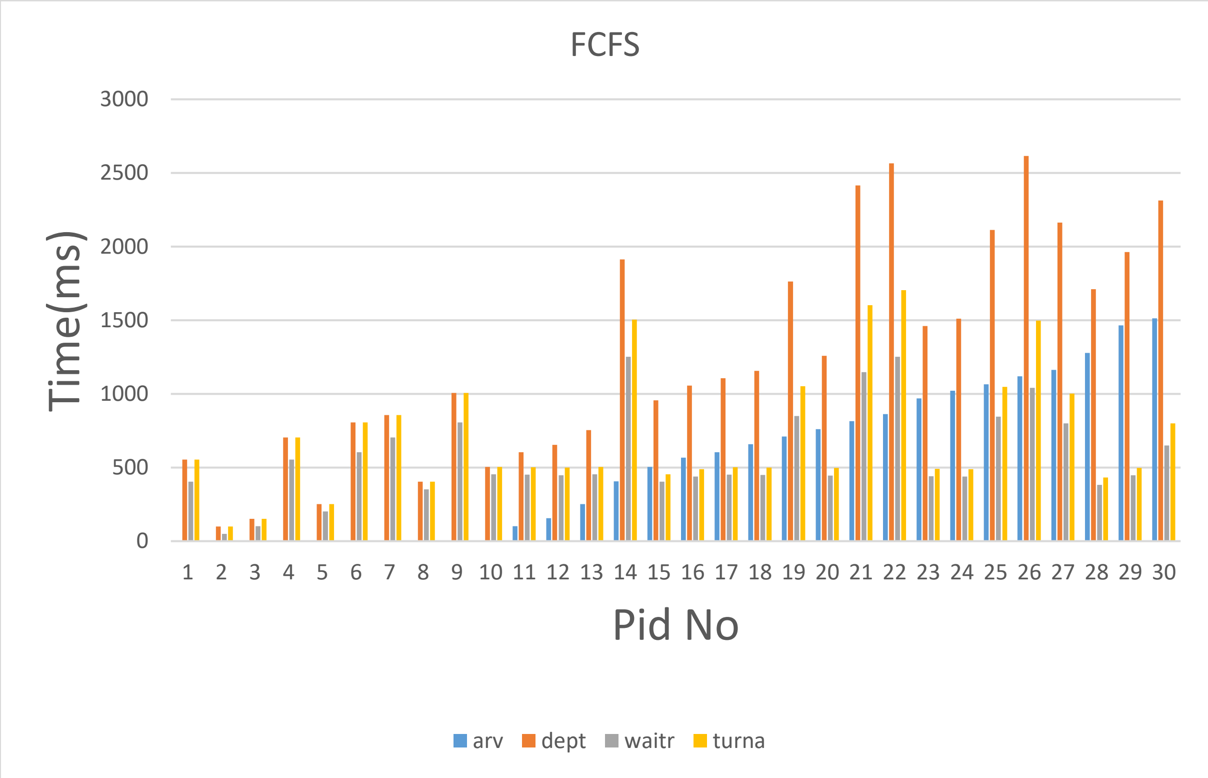
Parametres for that test: < SJF INF 50 100 uniform 50 10 100 0.5 0.3 0.2 0.5 10 30 1 >



(Figure 1)

First Come First Serve

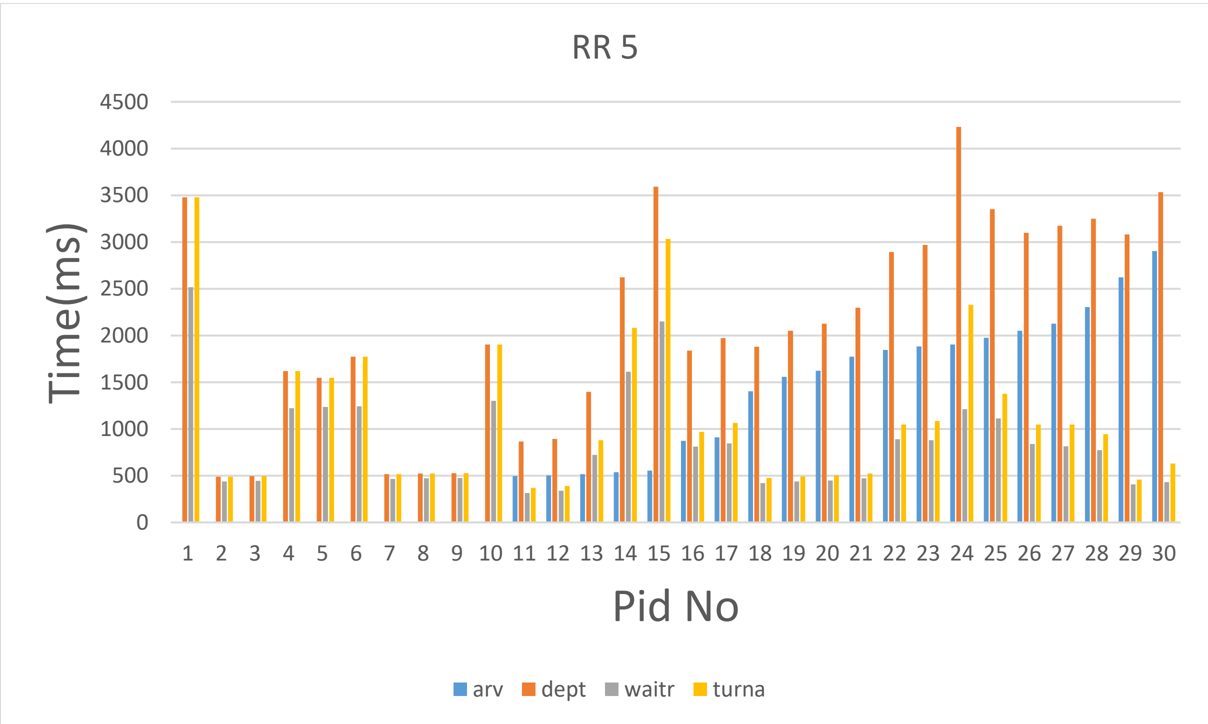
Parameters for that test: < FCFS INF 50 100 uniform 50 10 100 0.5 0.3 0.2 0.5 10 30 1 >



(Figure 2)

RR (Q = 5)

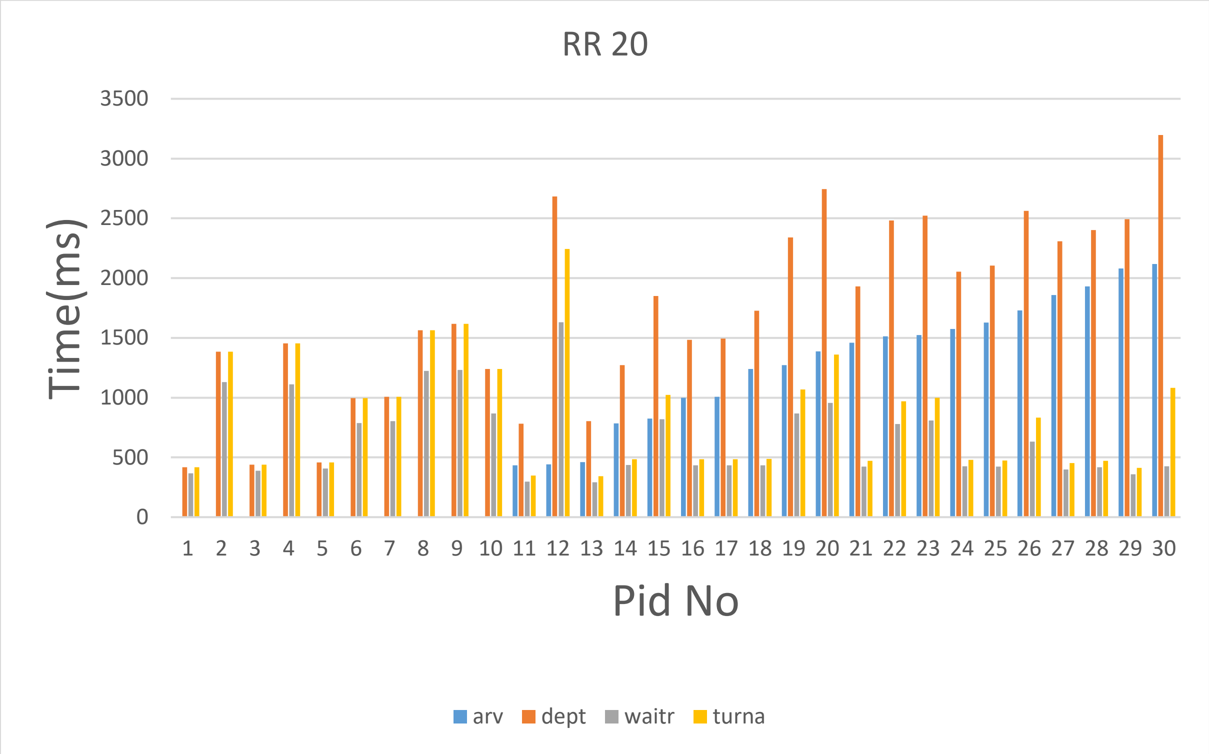
Parameters for that test: < RR 5 50 100 uniform 50 10 100 0.5 0.3 0.2 0.5 10 30 1 >



(Figure 3)

RR (Q = 20)

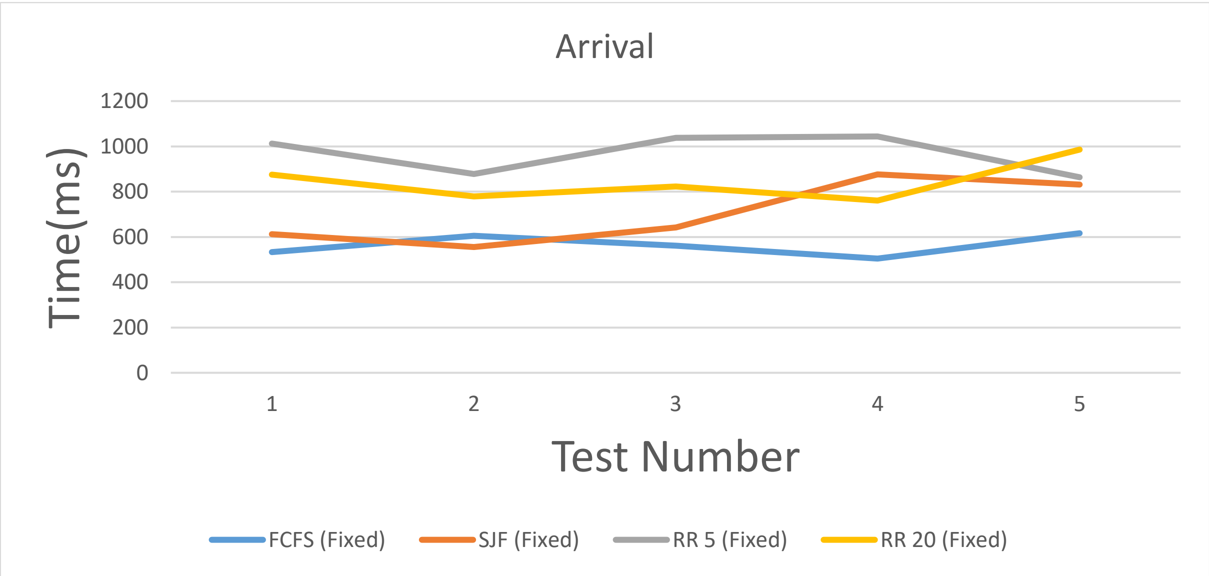
Parameters for that test: < RR 20 50 100 uniform 50 10 100 0.5 0.3 0.2 0.5 10 30 1 >



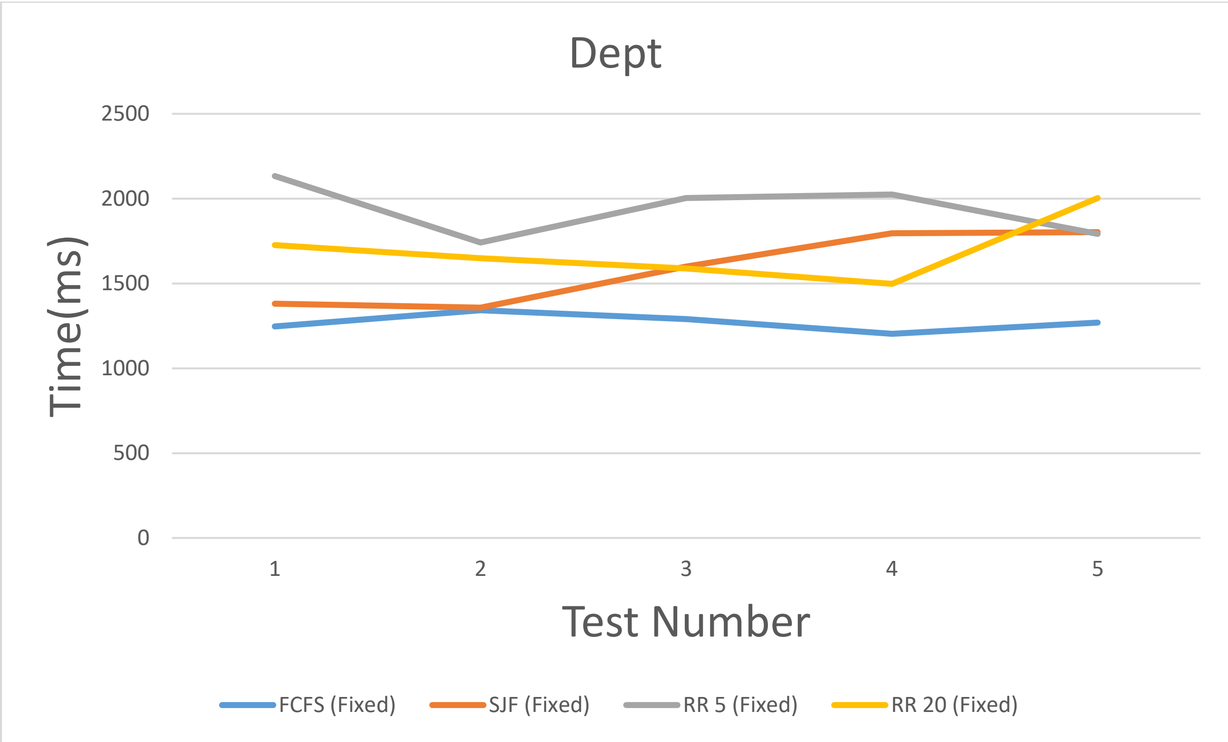
(Figure 4)

Graphs for Arrival, Dept, Turnaround and Wait times with 5 trials

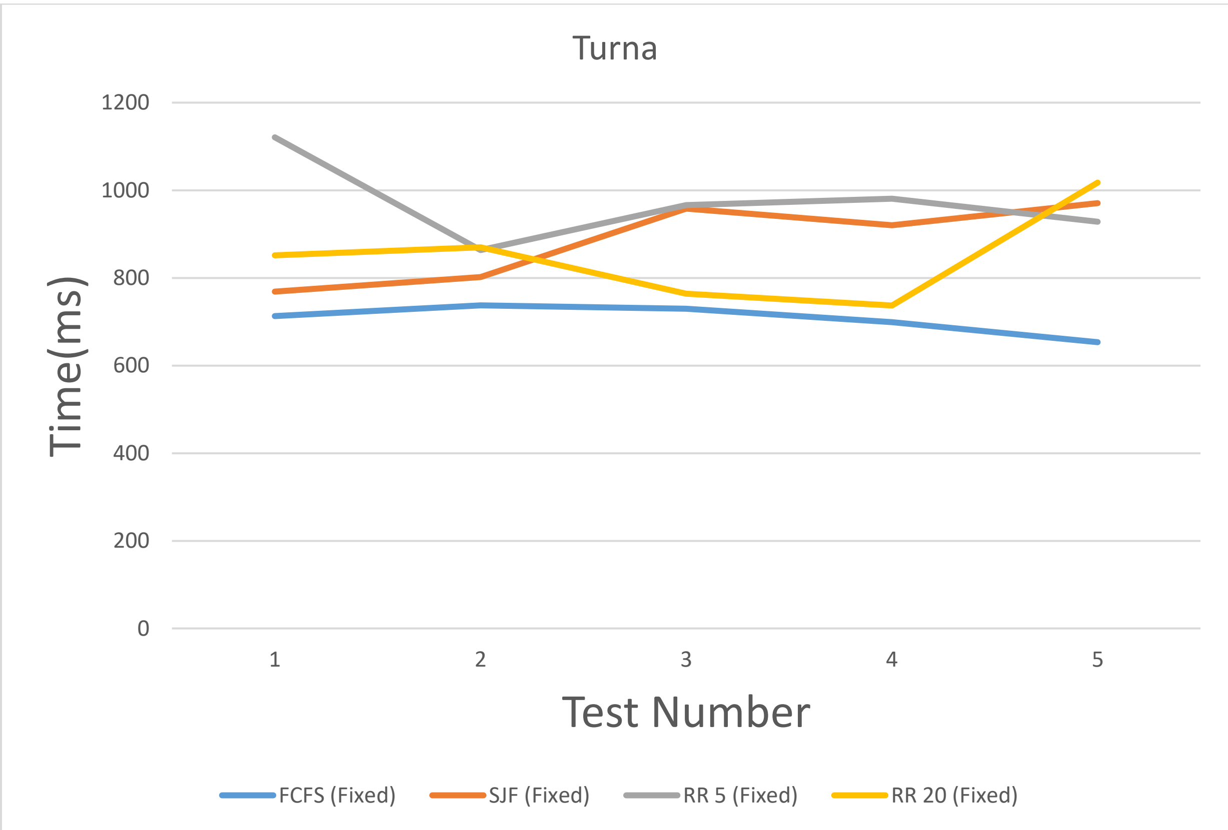
Here are the graphs for average arrival, dept, turnaround and wait times of 30 processes according to different algorithms, each algorithm is tested 5 times here



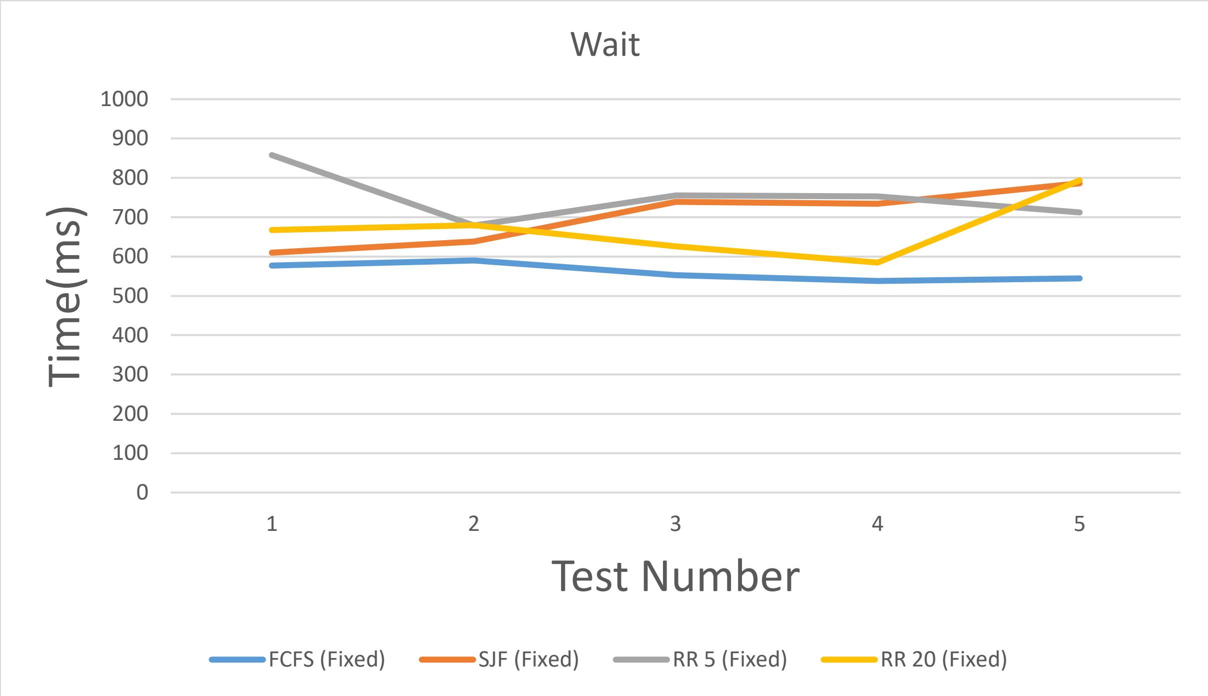
(Figure 5)



(Figure 6)



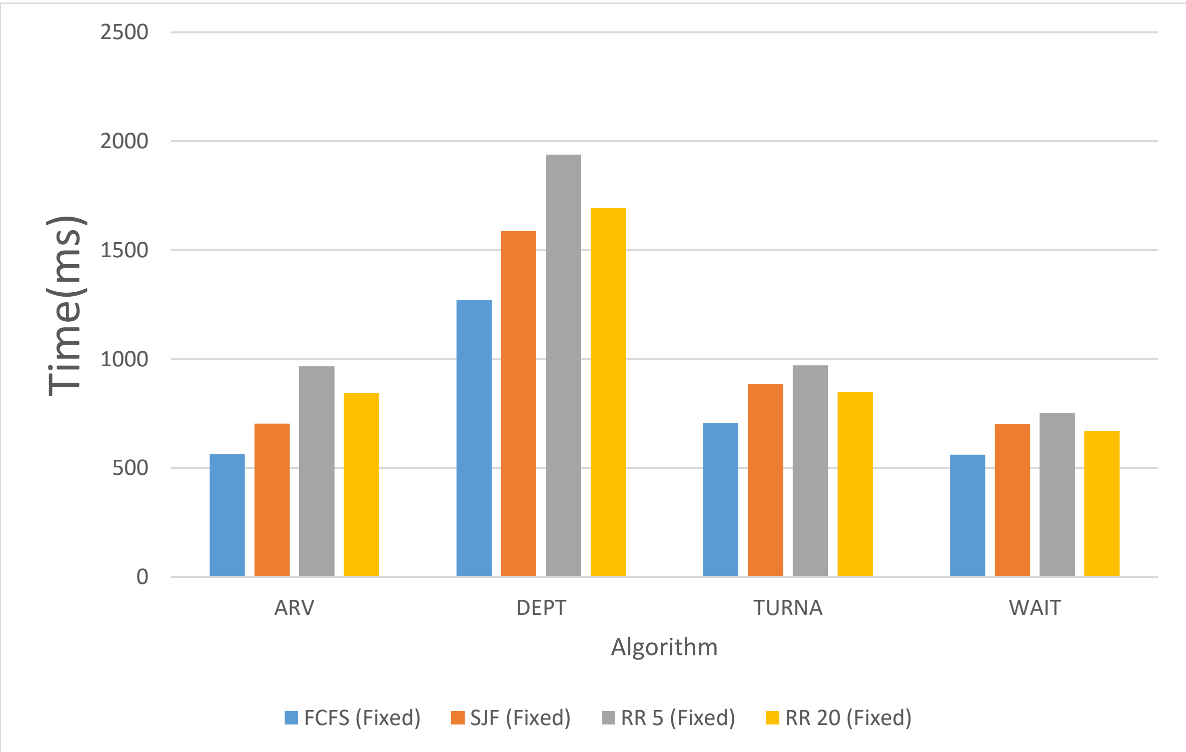
(Figure 7)



(Figure 8)

Graph for average arrival, dept, turnaround and wait times according to different algorithms

Here is the graph for average times of arrival, dept turnaround, and wait times according to different algorithms. We take the averages of Figure 5, Figure 6, Figure 7, Figure 8 and reflect the result here.



(Figure 9)

Interpretations:

It can be said that general summary of our graphs can be seen from Figure 9. If we look at Figure 9 we can say that average of process arrivals time ordering is RR(q = 5) > RR(q = 20) > SJF > FCFS the reason for RR algorithms to have the maximum average arrival times is that the process generator may wait for the number of threads in the system to be lower than the MAXP before creating another thread. We expected average time for FCFS would be slightly bigger than average arrival for SJF as we thought possibility of current thread number in FCFS reaching to MAXP is bigger than the possibility of current thread number in SJF reaching to MAXP. However, we saw in our tests, average arrival of SJF > average arrival of FCFS. We think that in our SJF algorithm complexity selecting the thread that is going to Cpu is O(n), even though we used small n the underlying cause may be this.

Average of process finishing time ordering is RR(q = 5) > RR(q = 20) > SJF > FCFS, we expected that RR(q=5) would be biggest as it uses very small time quantums, RR(q=20) is the second biggest as it uses time quantums with 20ms, and FCFS to be third and SJF to be fastest. We saw that SJF again is slower than FCFS we again interpret as like in first part our selecting thread algorithm may cause SJF to be bigger than FCFS algorithm.

Average of process turnaround time ordering is RR(q = 5) > SJF > RR(q = 20) > FCFS. That is the result of differences of average of finishing and average of arrival times which we interpreted above. We had also expected RR(q = 5) > RR(q = 20) > FCFS > SJF for average turnaround times as we expected SJF to be fastest.

Average of process waiting time ordering is RR(q = 5) > SJF > RR(q = 20) > FCFS. The ordering according to our expectation was RR(q = 5) > RR(q = 20) > FCFS > SJF. Most surprising result was that as SJF is slower than both RR(q=20) and FCFS. We had expected SJF to be fastest.

Although we got some unexpected results from our tests we believed as test number increase we will get close to theoretically true orderings and we also think complexity of SJF's selecting algorithm may have an affect on its slowness compared to others.