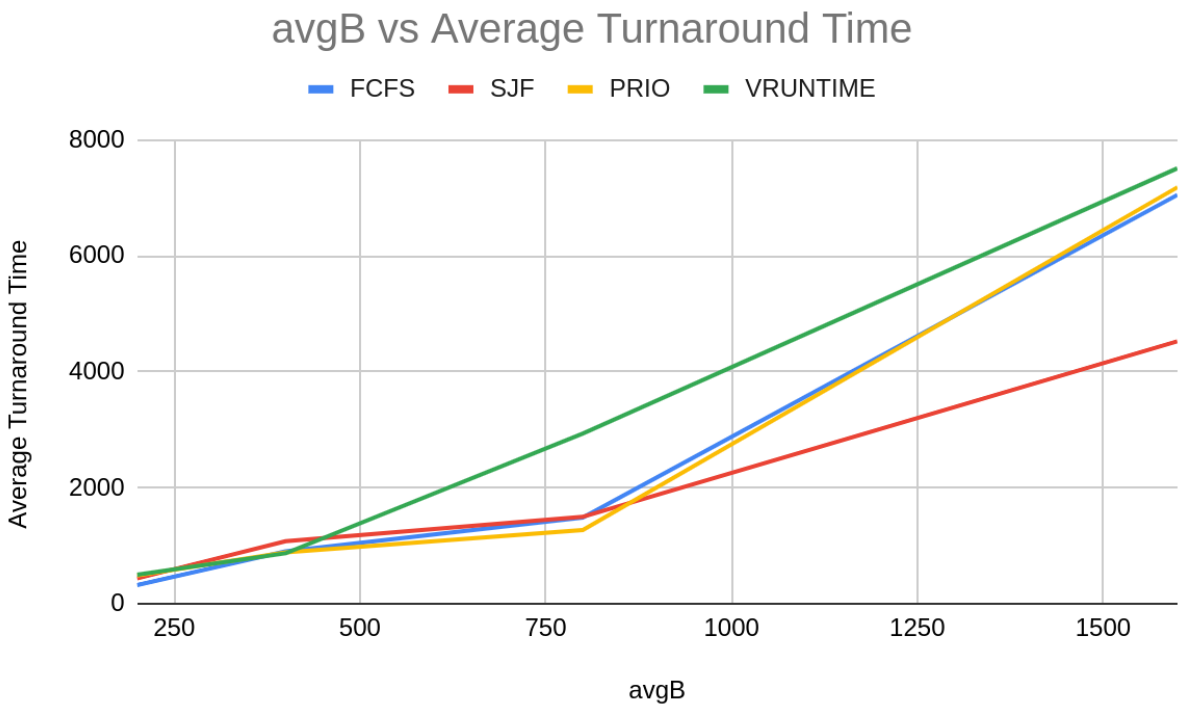
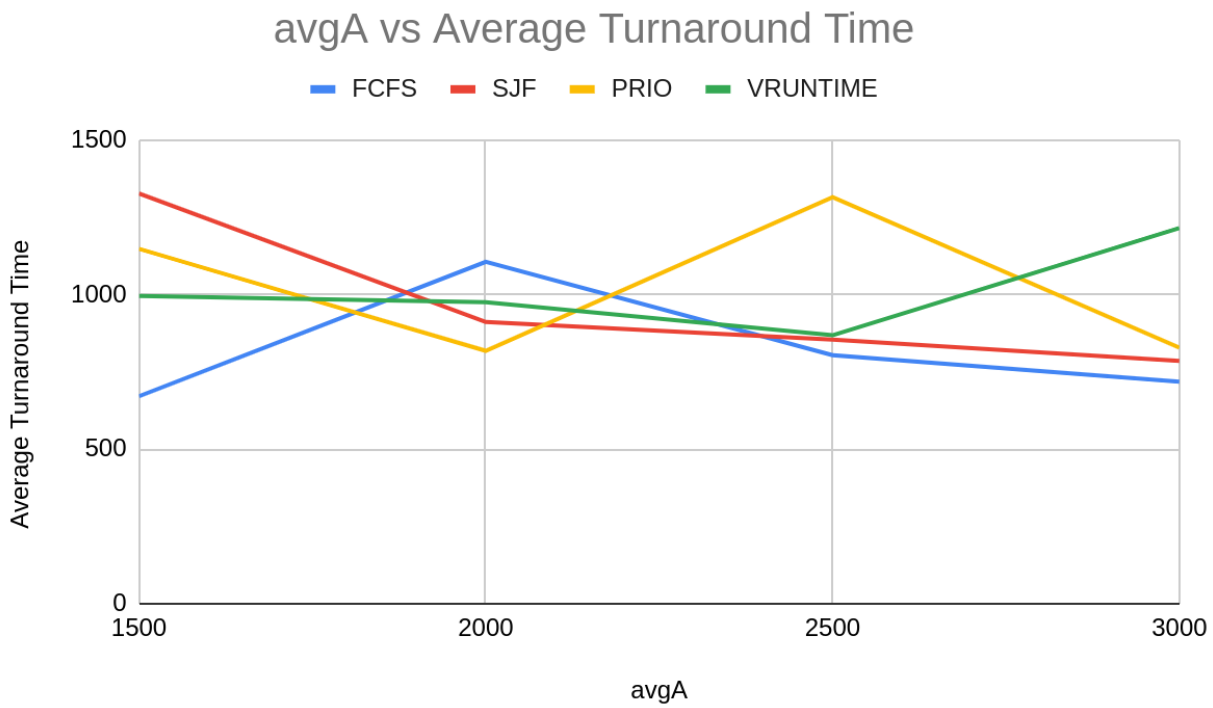
Osama Tanveer
21801147
Section 3

Experiment 1



The first experiment corresponds to increasing the average burst length and evaluating each algorithm with this increased burst length. The average turnaround time is used as an evaluation metric. As the average burst length increases, the average turnaround time increases for all algorithms, as indicated by the graph. Since all the algorithms are non-preemptive, once a burst is executing, the arrival of a new burst in the runqueue would not affect its execution. Therefore, as the average burst length is increased, it is highly likely that a burst with a very large length would take control of the cpu and would not let any other burst to execute. The increase of average burst length for the SJF algorithm, however, does not show the same degree of increase of average turnaround time compared to others. This is because the convoy effect is avoided in this case. The shortest burst is selected and executed. Therefore, the burst with the smallest burst length is not blocked by a burst with a larger burst length when the burst is selected from the runqueue to execute. The trend is greatest for VRUNTIME, FCFS, and PRIO algorithms.

Experiment 2

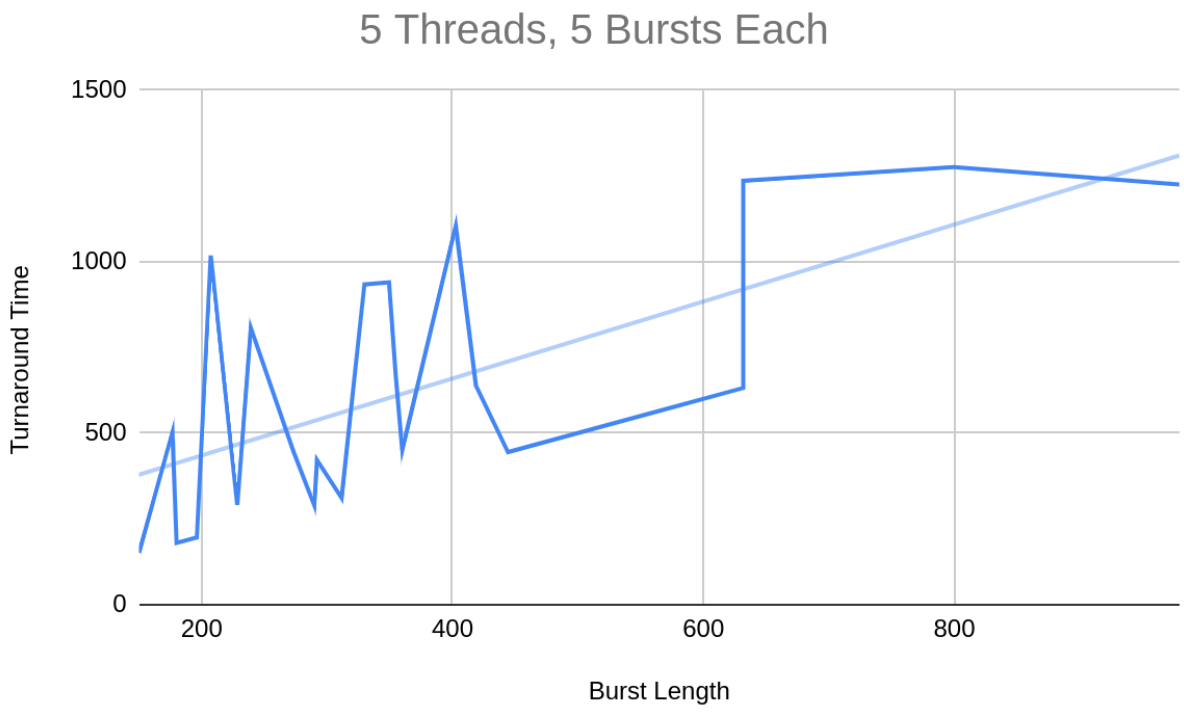


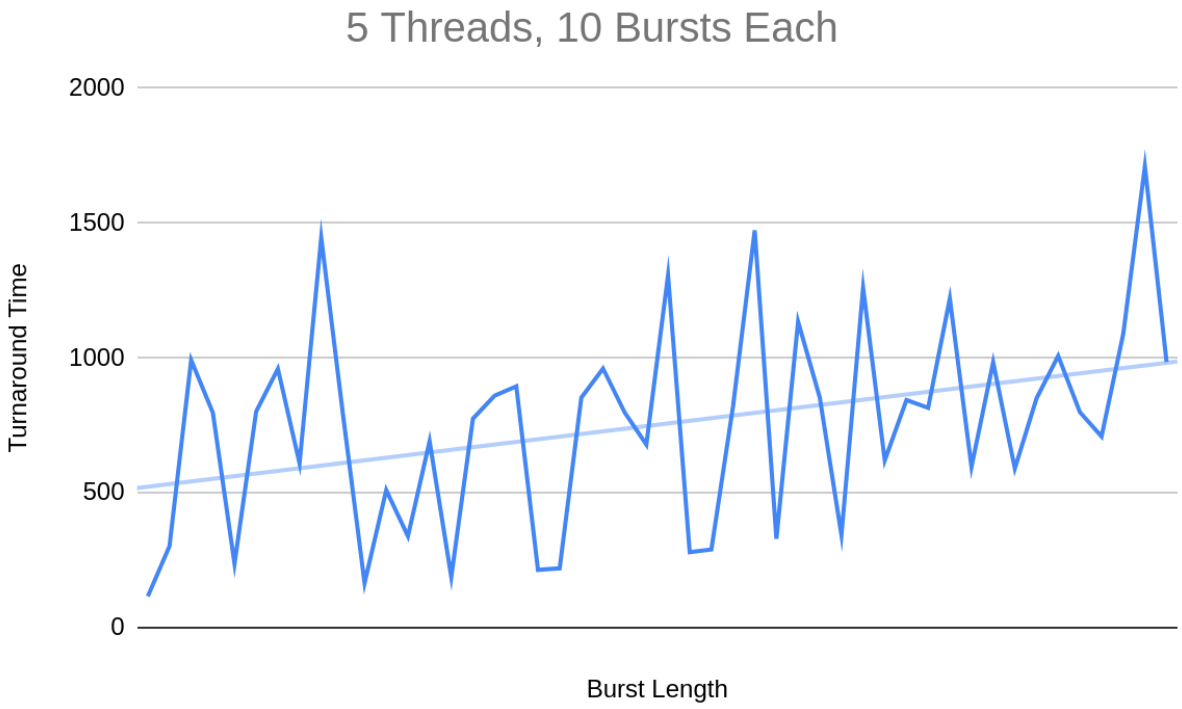
The second experiment corresponds to increasing avgA (the average arrival time of bursts) and plotting the value for the average turnaround time. The trend shown by algorithms is downwards. Since the arrival time of all bursts is delayed as the value of average arrival time is increased, the average number of bursts in the runqueue at any given instance of time is lower. Bursts do not have to wait a long time in runqueue before their turn. This gives rise to an overall decrease of average runtimes. There are some fluctuations and anomalous points in the graph whose presence can be explained by the random generation of arrival times. The general graph trend for all algorithms is downwards, indicating that as the average arrival time is increased, the average turnaround time decreases. This is possible because at any given instance of time, there are less bursts in the runqueue. The average burst length remains constant throughout the experiment.

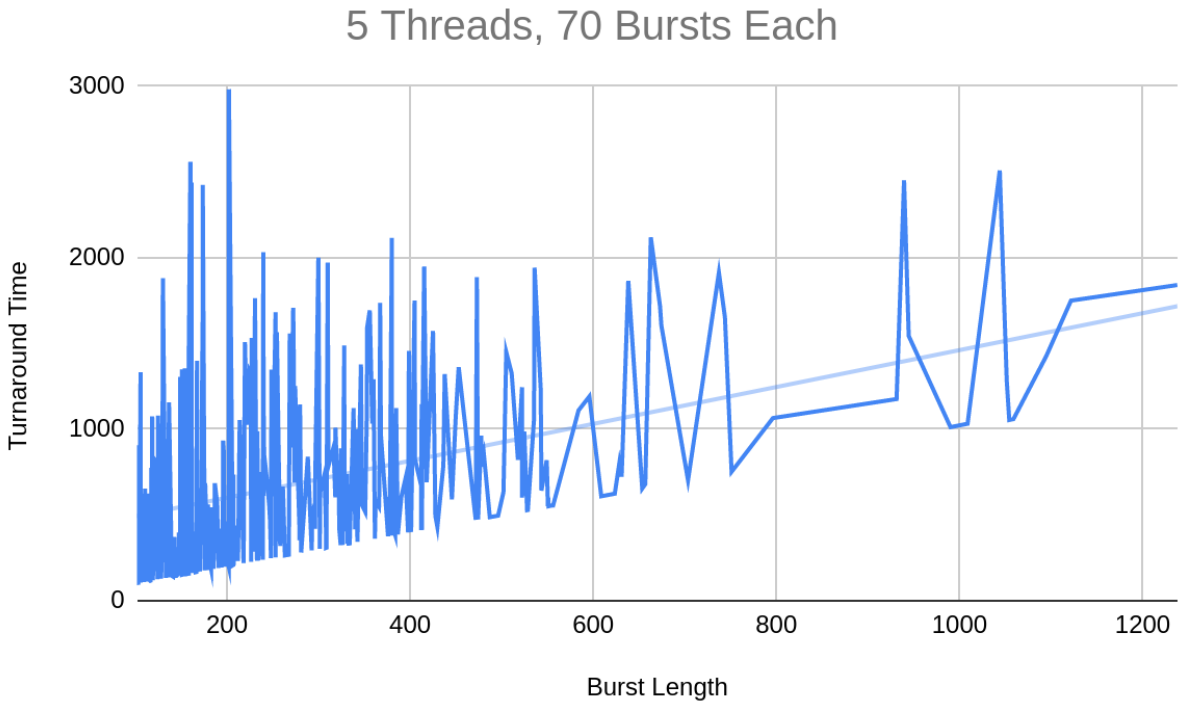
Experiment 3

This experiment involves increasing the burst length and for each increase plotting a graph of burst length against the turnaround time. The values for N, avgA, and avgB are kept constant. The value for the number of bursts is increased from 5 to 10 to 70.

FCFS



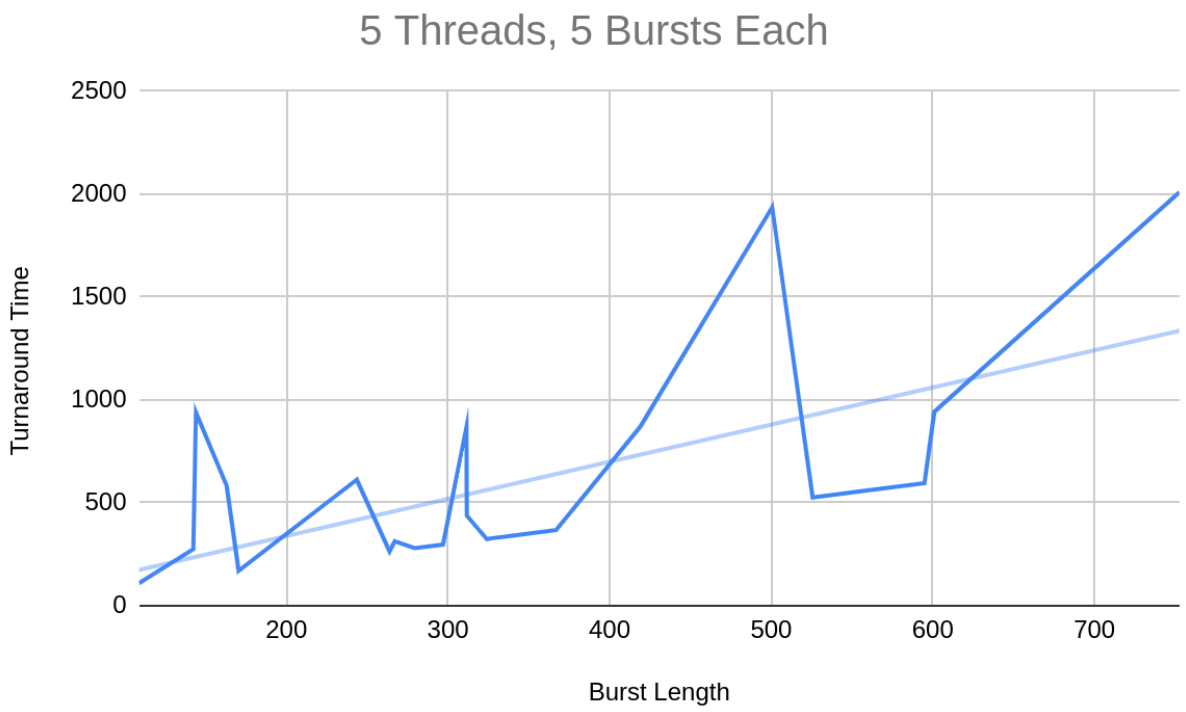


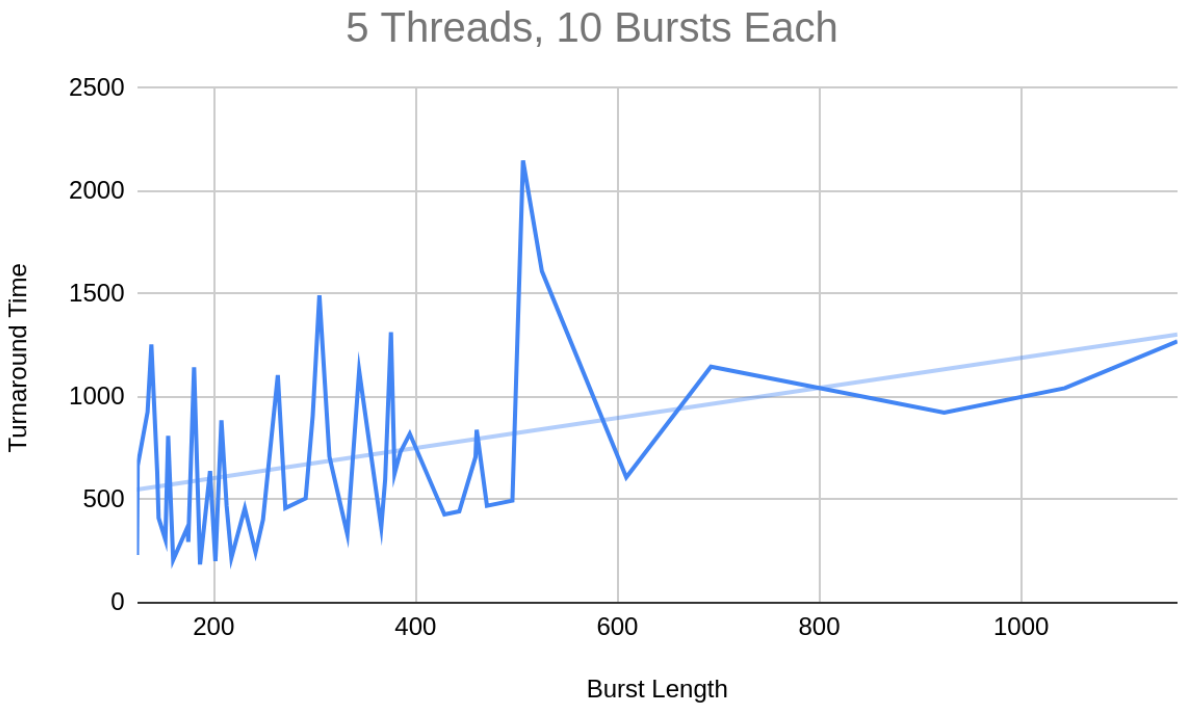


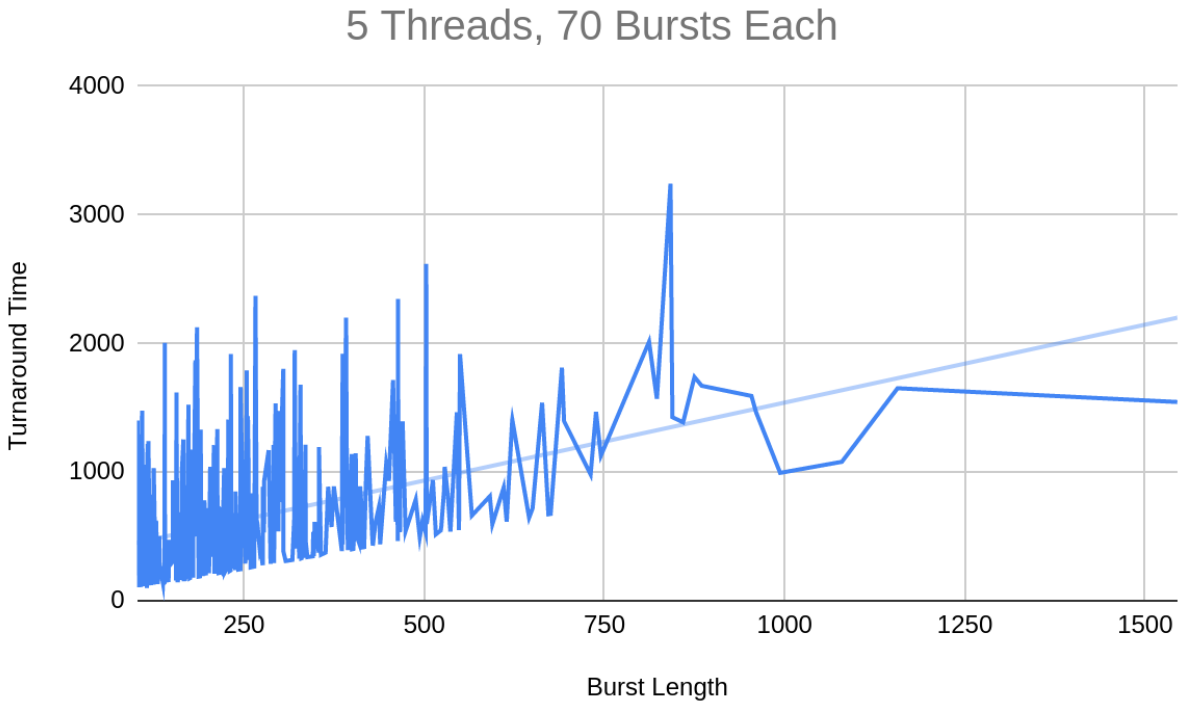
As it can be seen from the 3 graphs above, the average turnaround time shows an upward trend for each of the graphs. This is because of the convoy effect, the bursts with larger lengths block the execution of the bursts with the shorter lengths. The graph gets very rough and shows extreme fluctuations as the number of bursts is increased. This is because as the number of

bursts increases, it is highly likely that the bursts produced would have a length around the average value, which in this experiment was 200 ms. It can be seen that for these small bursts, the average turnaround time is extremely large. This is again because of the convoy effect. The larger length bursts block the execution of the smaller length bursts.

SJF



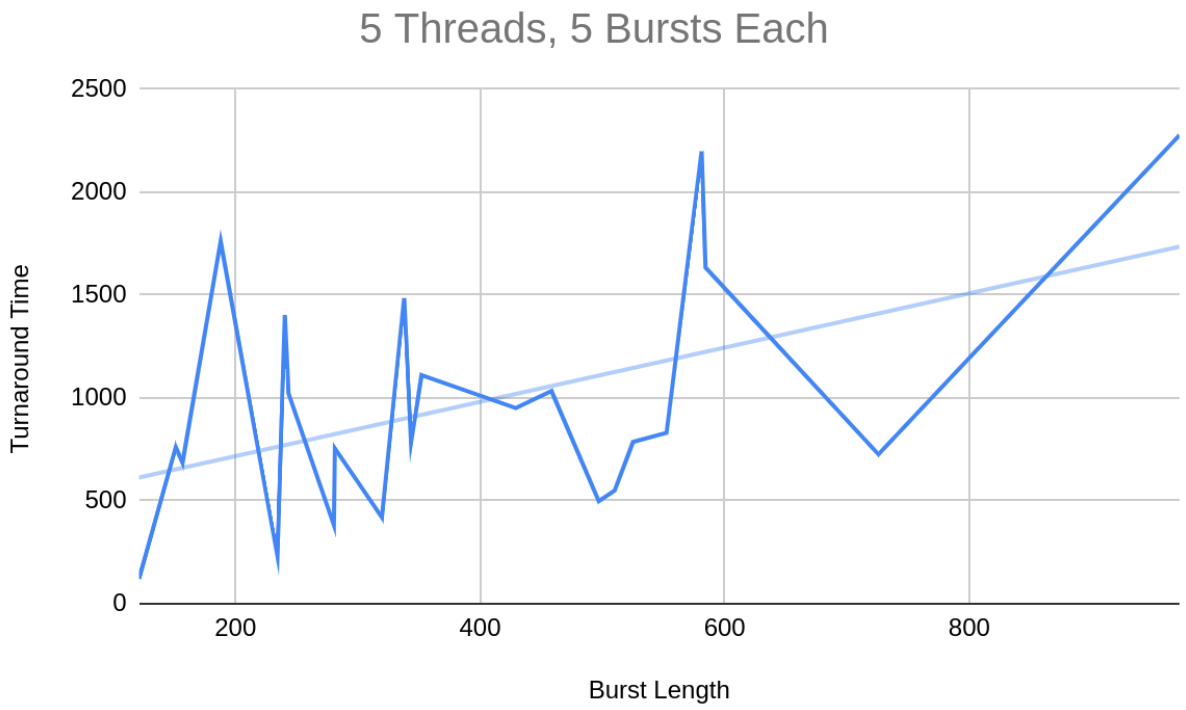


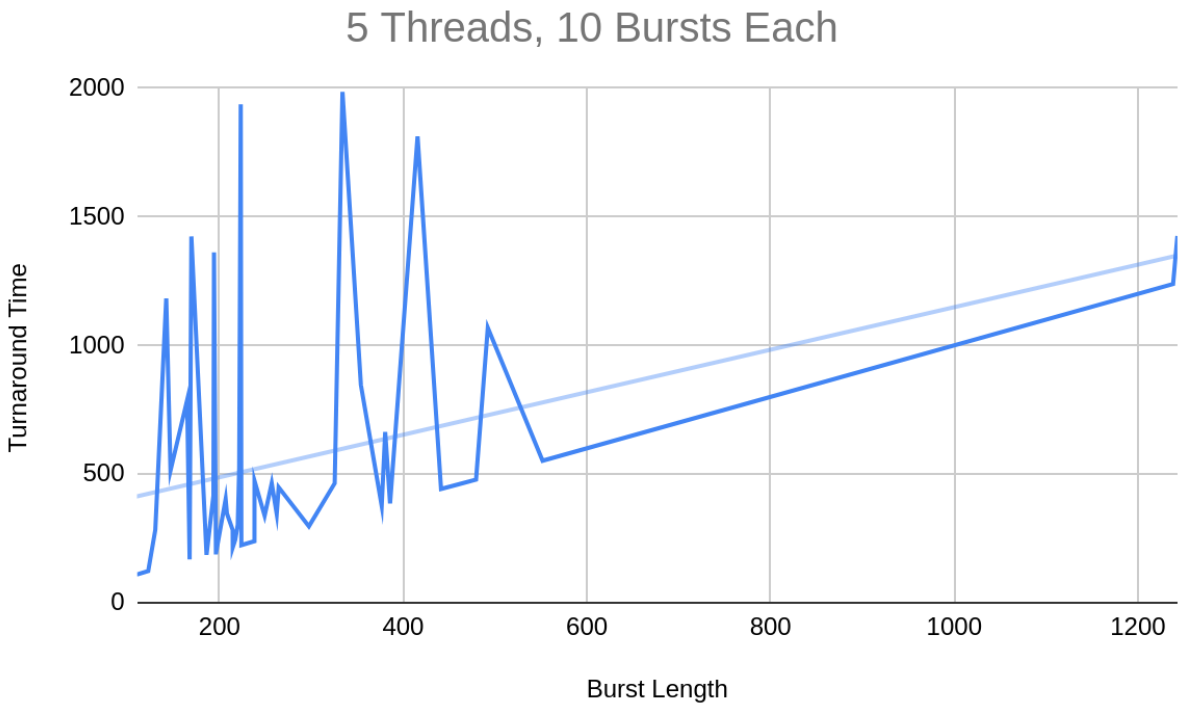


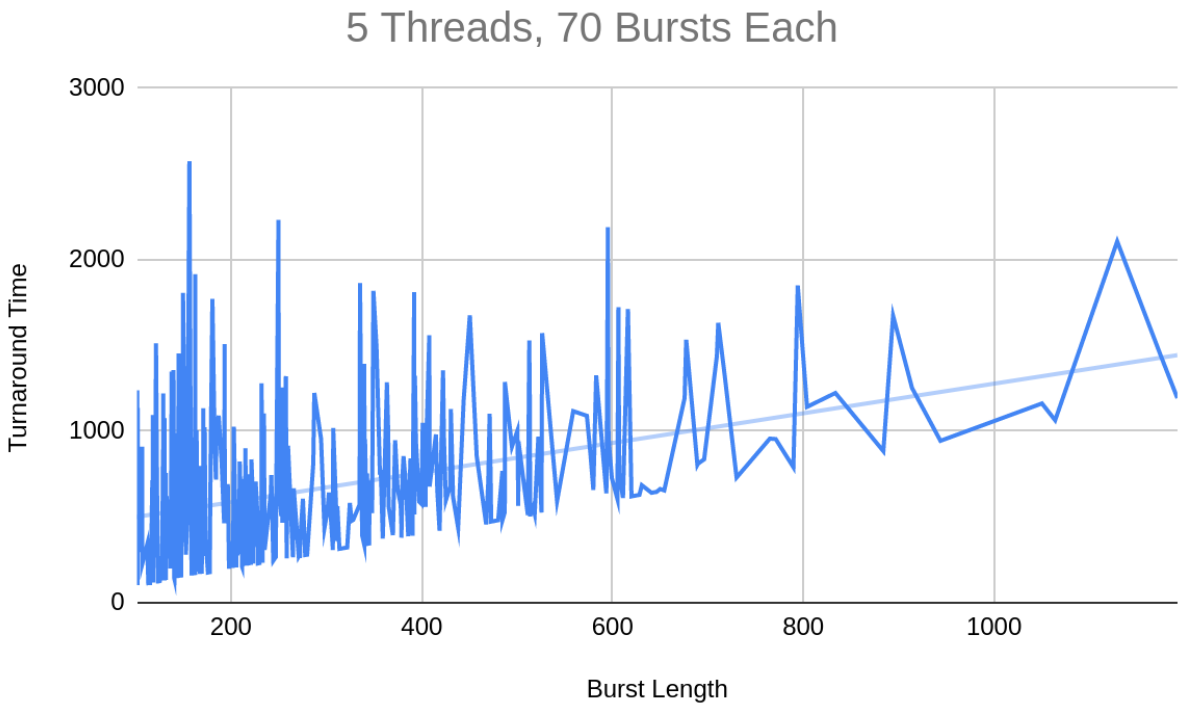
The average turnaround time again shows an upward trend. Even though SJF ensures that the shortest burst is selected first, the non-preemptive nature of the algorithm does not prevent the convoy effect. If a burst with a large length arrives and takes over the CPU, the shorter length bursts keep waiting in the runqueue. The spikes in this case are smaller than FCFS, as can be

seen from the graphs above. This indicates that the average turnaround time for shorter bursts are smaller compared to FCFS. This algorithm is definitely better than FCFS when we have a large number of bursts and their average burst lengths are small.

PRIO



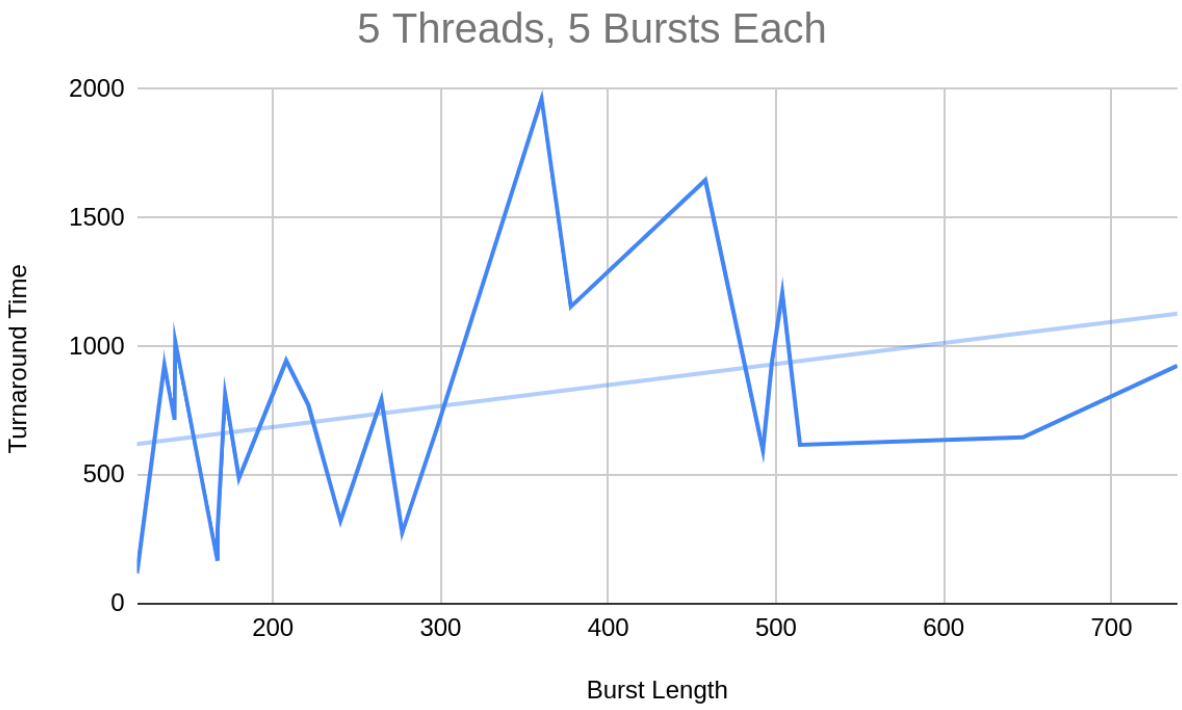


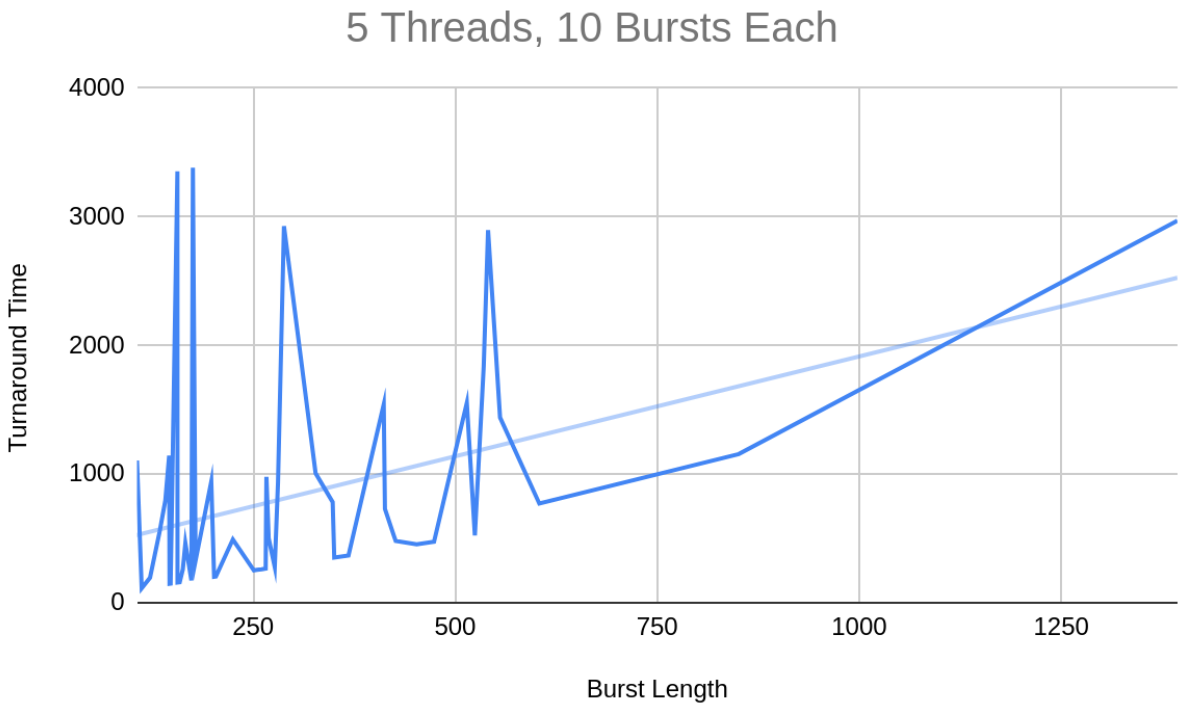


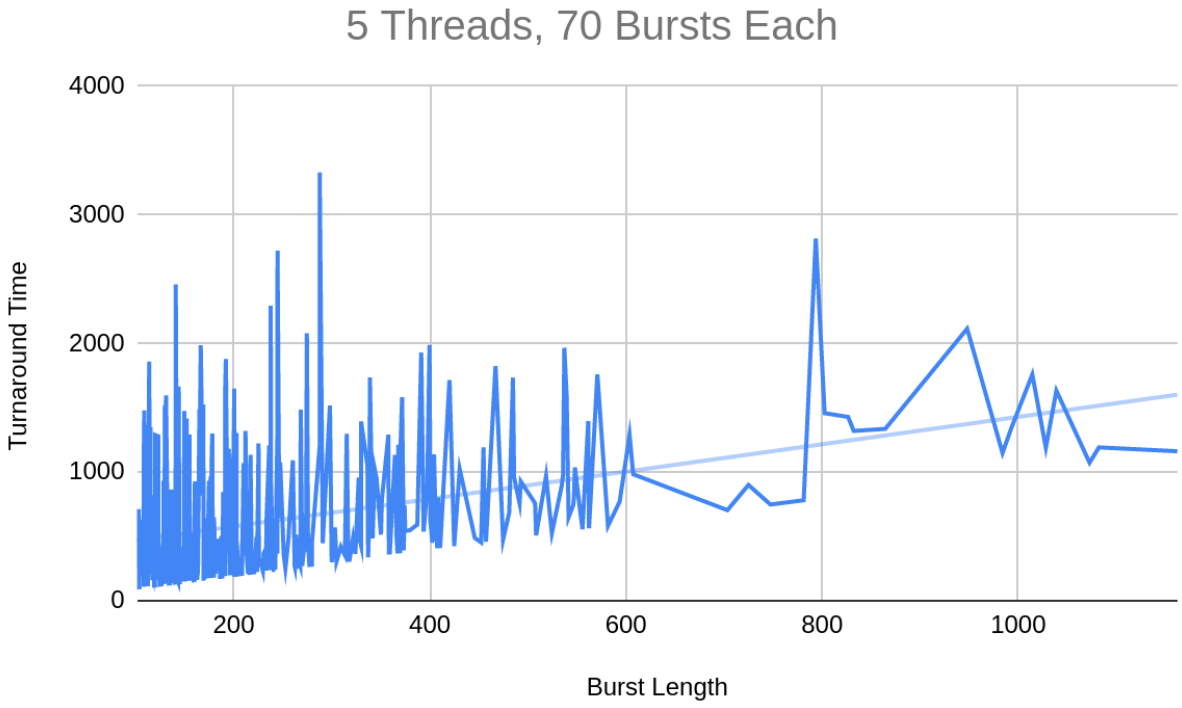
In the case of PRIO, decisions of which process to execute are solely based on the priority of the thread. Even though there is an upward trend, it is important to realize that the burst from thread i would always be executed before the burst from thread i+1. Therefore, no matter the

lengths of bursts produced from a high priority thread, they will always be executed before the bursts from a low priority thread. The initial spikes in graphs are definitely greater than SJF, but smaller than FCFS. An important application of these would be to assign a lower thread number to kernel threads and a higher thread number to unimportant user threads. Since kernel processes usually produce small bursts, we can expect a lower turnaround time for these bursts. The increasing trend can be explained by the fact that no matter the length of a high priority burst, it will always be executed before the lower priority burst. Therefore, it is better to use this algorithm to differentiate between the execution of kernel threads and user threads.

VRUNTIME







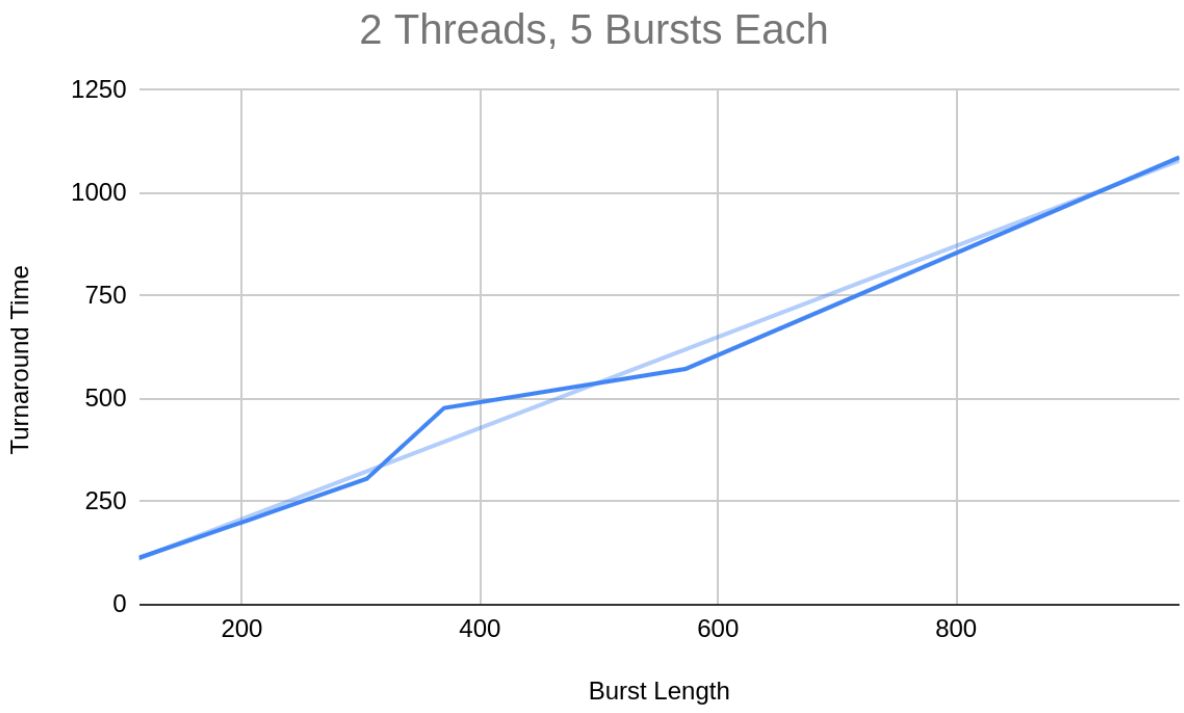
VRUNTIME algorithm is somewhat similar to the PRIO algorithm. This is because this algorithm assigns large values of vruntime to threads with larger thread number, and smaller vruntime values to threads with lower thread number. There is a general increasing trend. The spikes are

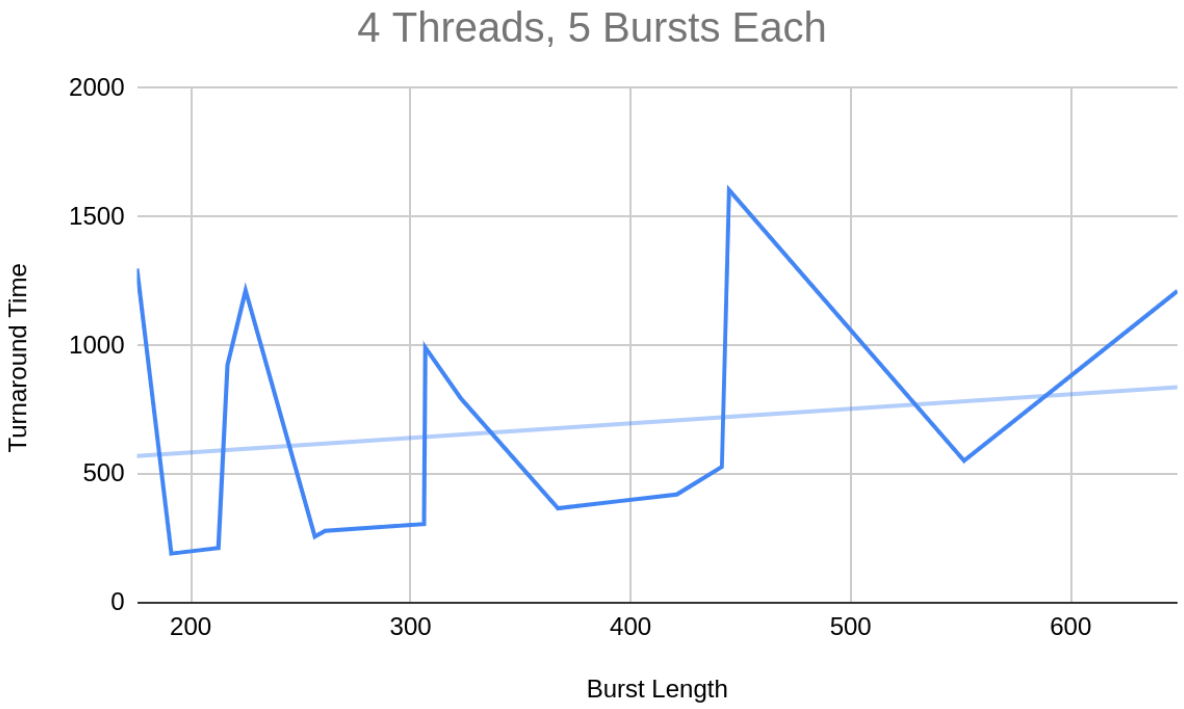
there again because if a thread with lower thread index takes over, no matter its burst length it is always executed first. This, again, can be used in applications where a differentiation of kernel threads and user threads is required.

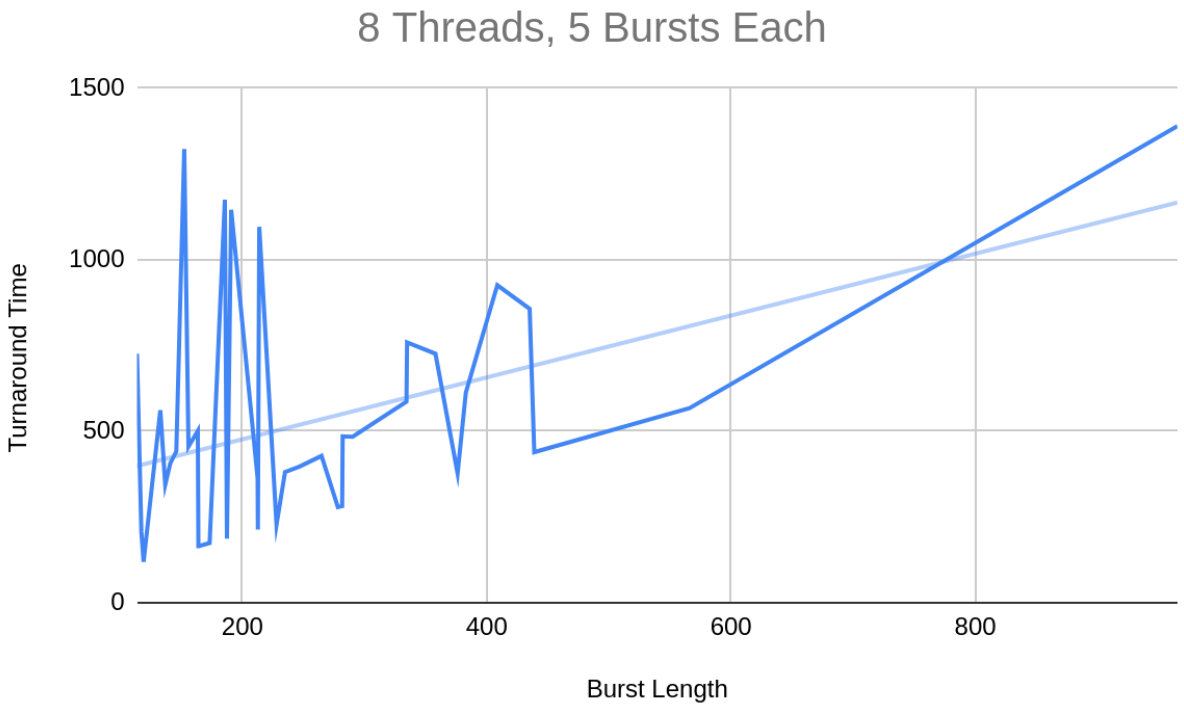
Experiment 4

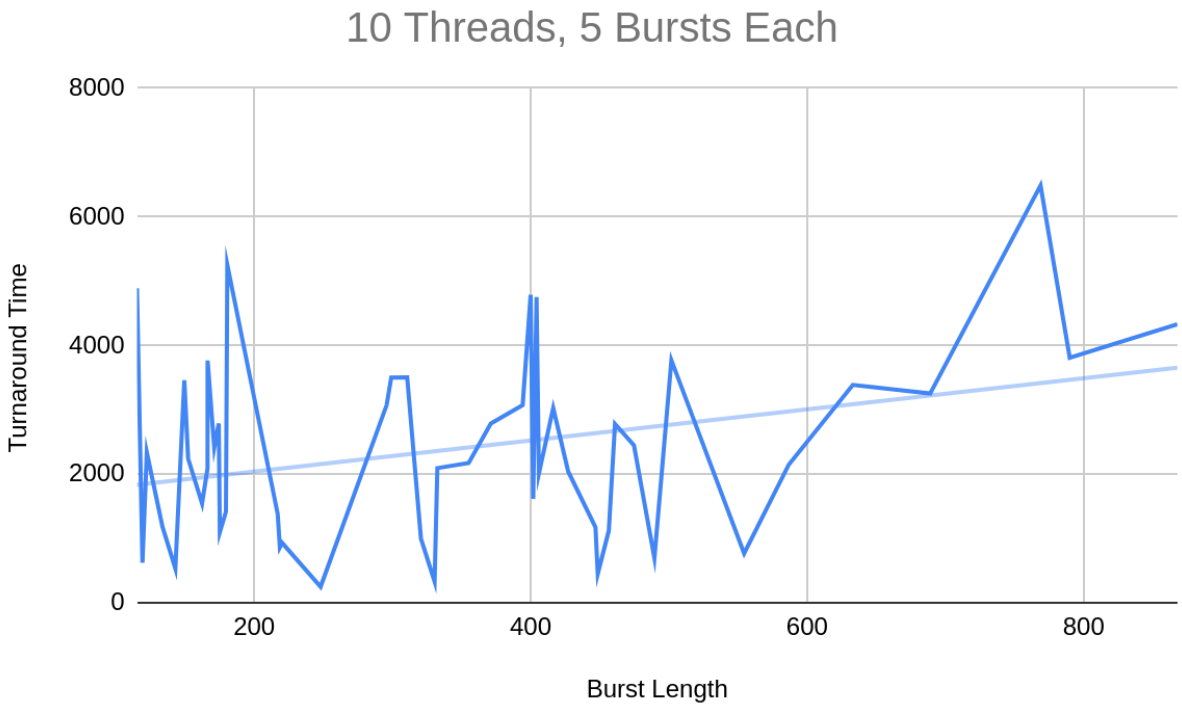
This experiment involves increasing the number of threads while keeping the burst count, average burst length and average arrival time constant.

FCFS



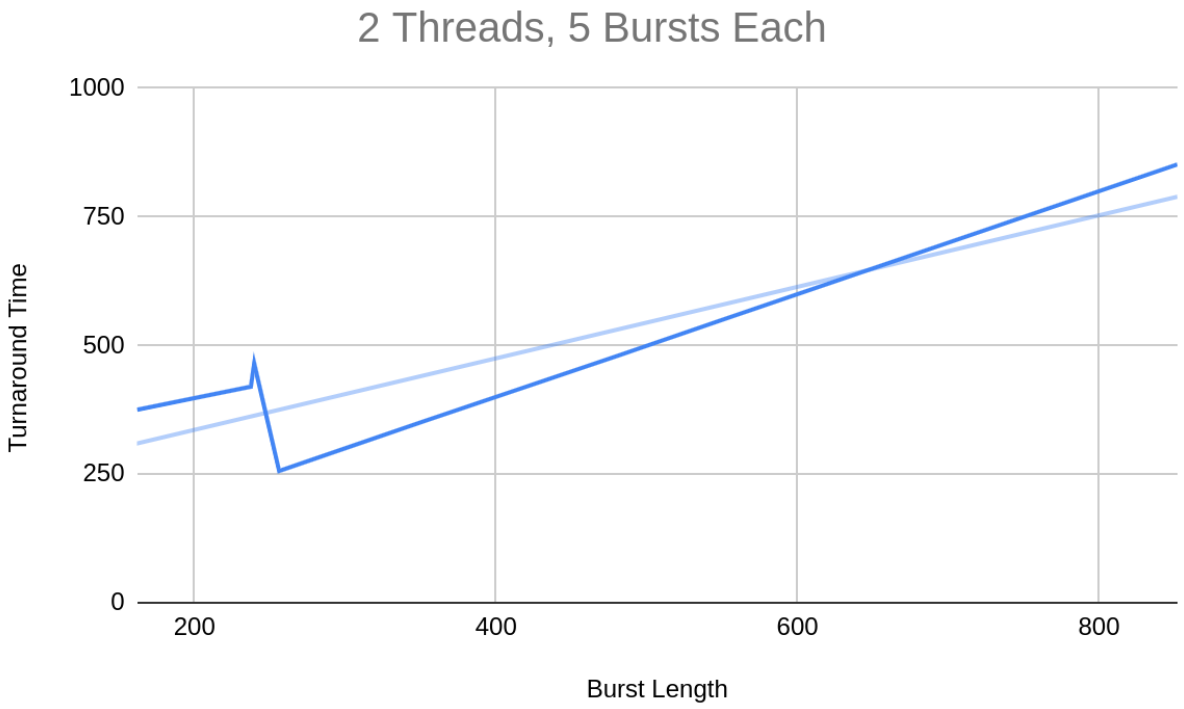

4 Threads, 5 Bursts Each
2000
1500
1000
500
0
Turnaround Time
200
300
400
500
600
Burst Length

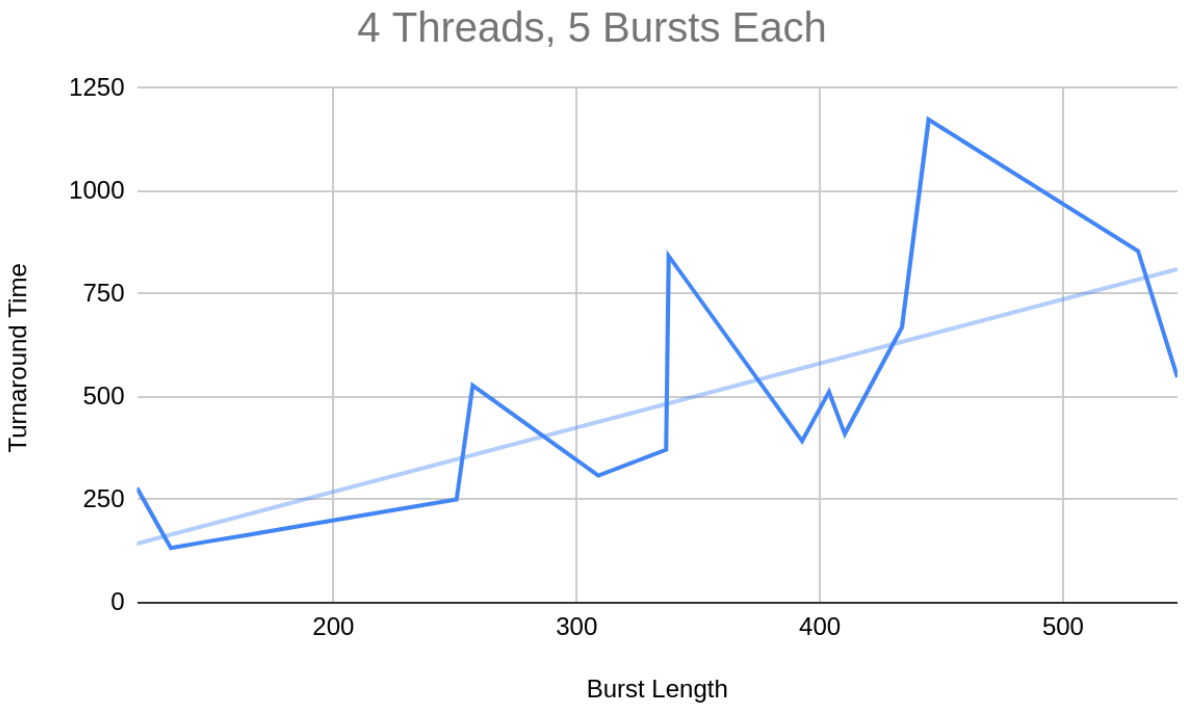

8 Threads, 5 Bursts Each
1500
1000
500
0
Turnaround Time
200
400
600
800
Burst Length

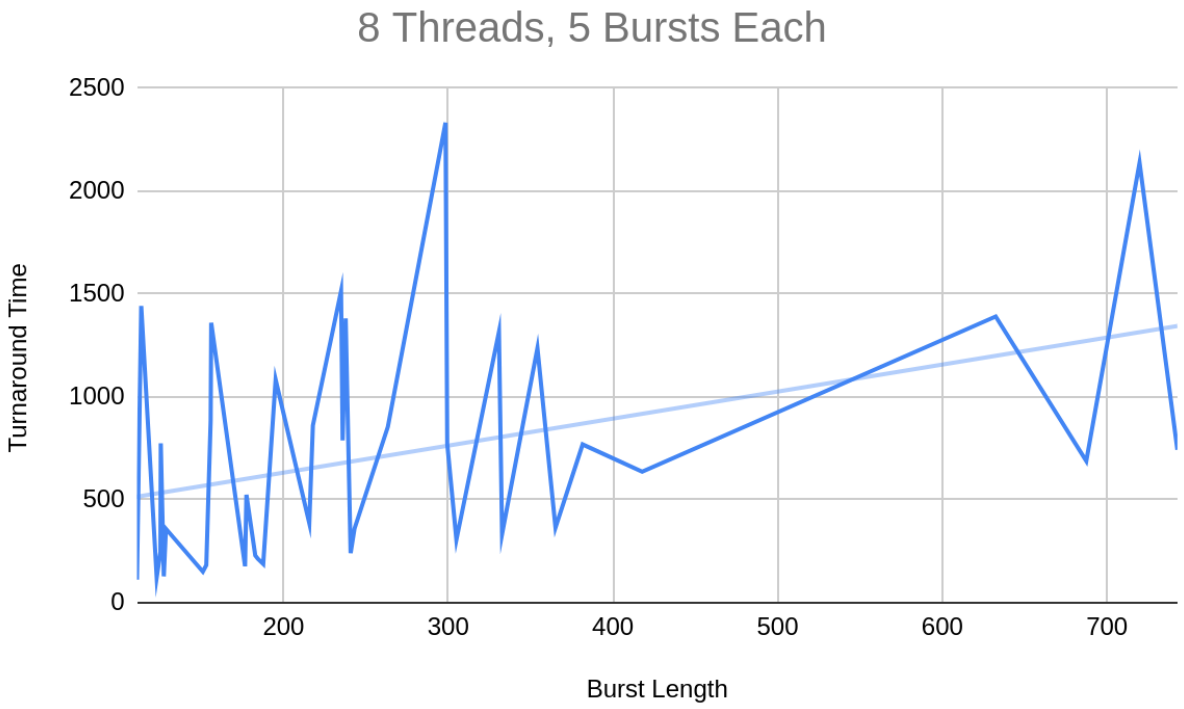


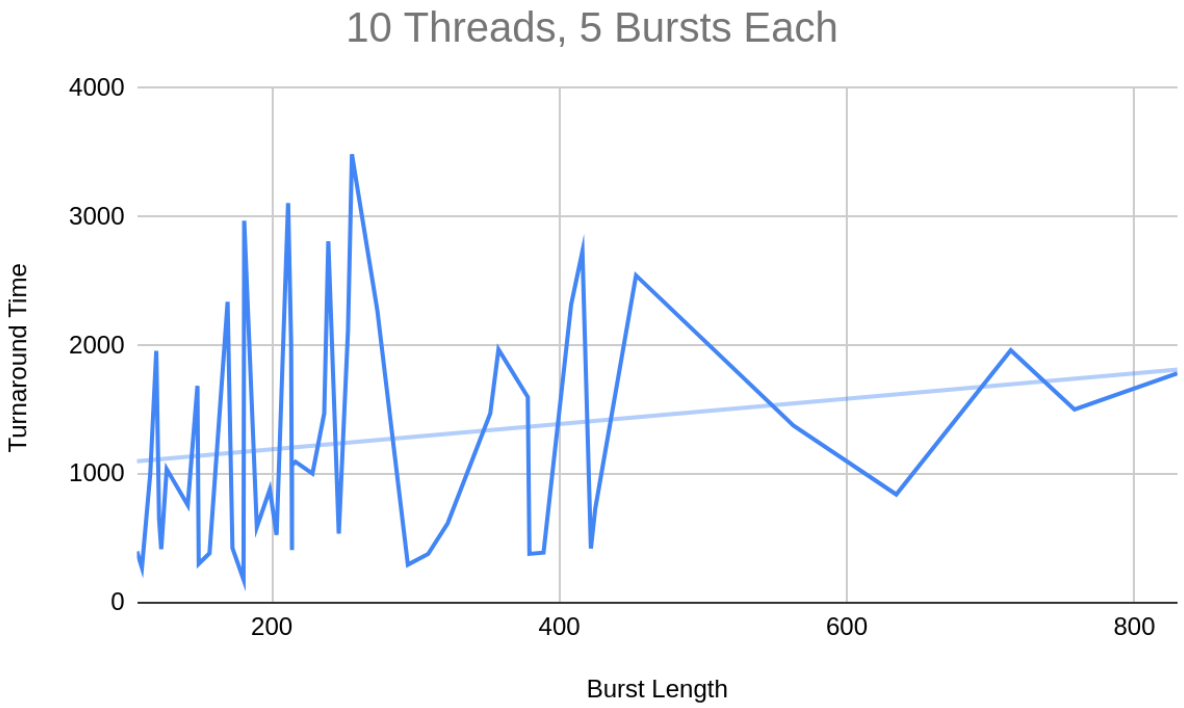
The trend is a general increase. As the number of threads increases, the total number of bursts produced increases as well. This results in more bursts waiting in the runqueue for them to be executed. Therefore, the turnaround time increases. The anomalies in the graphs can be again explained by the convoy effect, i.e. the bursts with larger length block the execution of bursts with smaller length.

SJF


2 Threads, 5 Bursts Each
1000
750
500
250
0
Turnaround Time
200
400
600
800
Burst Length


4 Threads, 5 Bursts Each
1250
1000
750
500
250
0
Turnaround Time
200
300
400
500
Burst Length

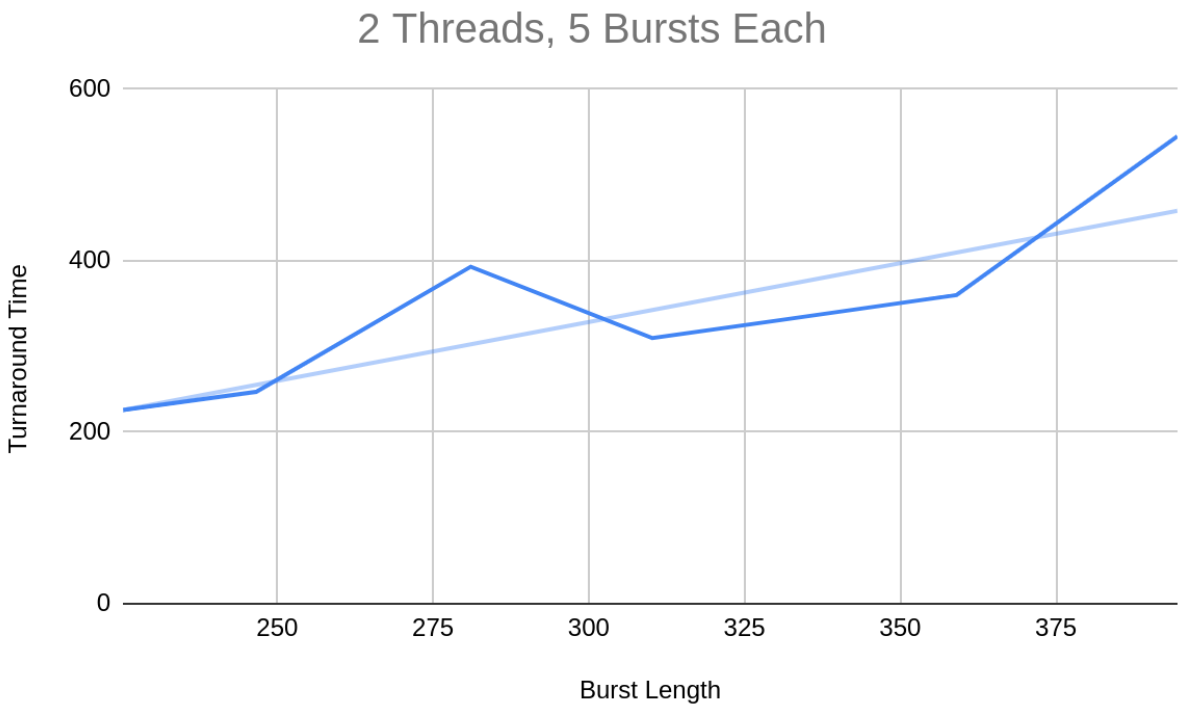


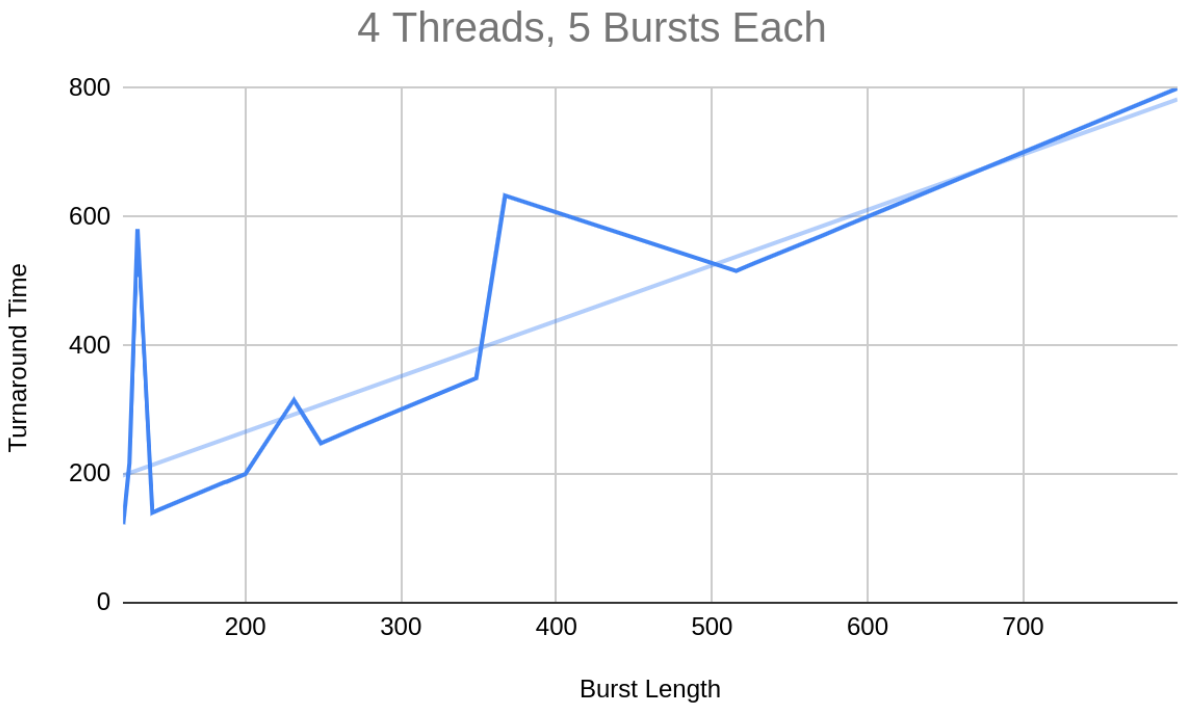


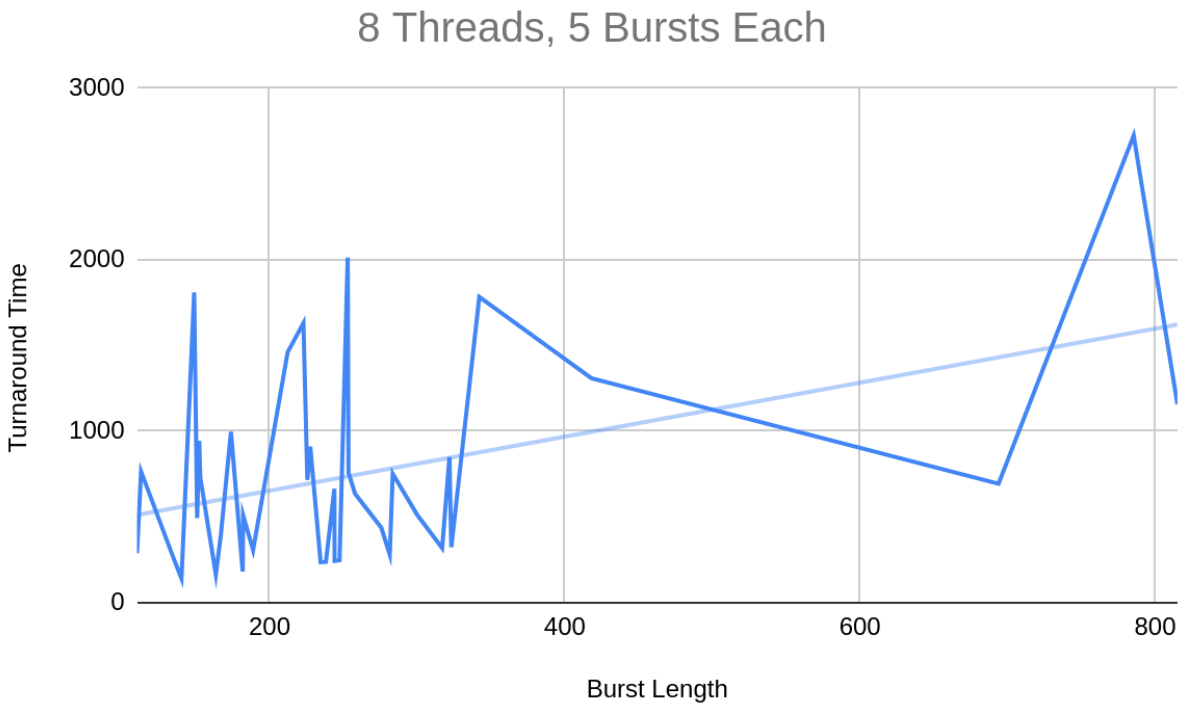
As the number of threads increase, the trendline shows that the average turnaround increases. This is partly because of the non-preemptive nature of the algorithm. Once a large length burst takes over the CPU, the smaller length burst has to wait for its completion. The trendline is

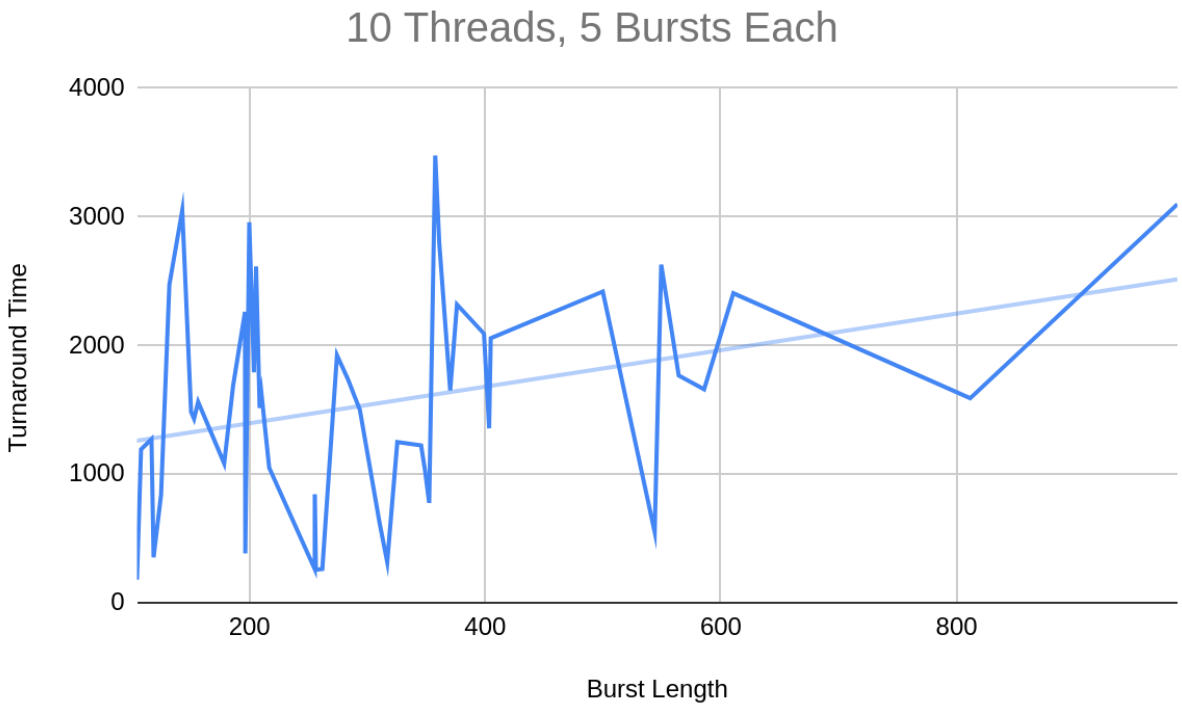
smaller than FCFS. The spikes are caused by the convoy effect. Moreover, it can be seen that the spikes for average turnaround time are smaller than FCFS. This algorithm proves to be better than FCFS.

PRIO



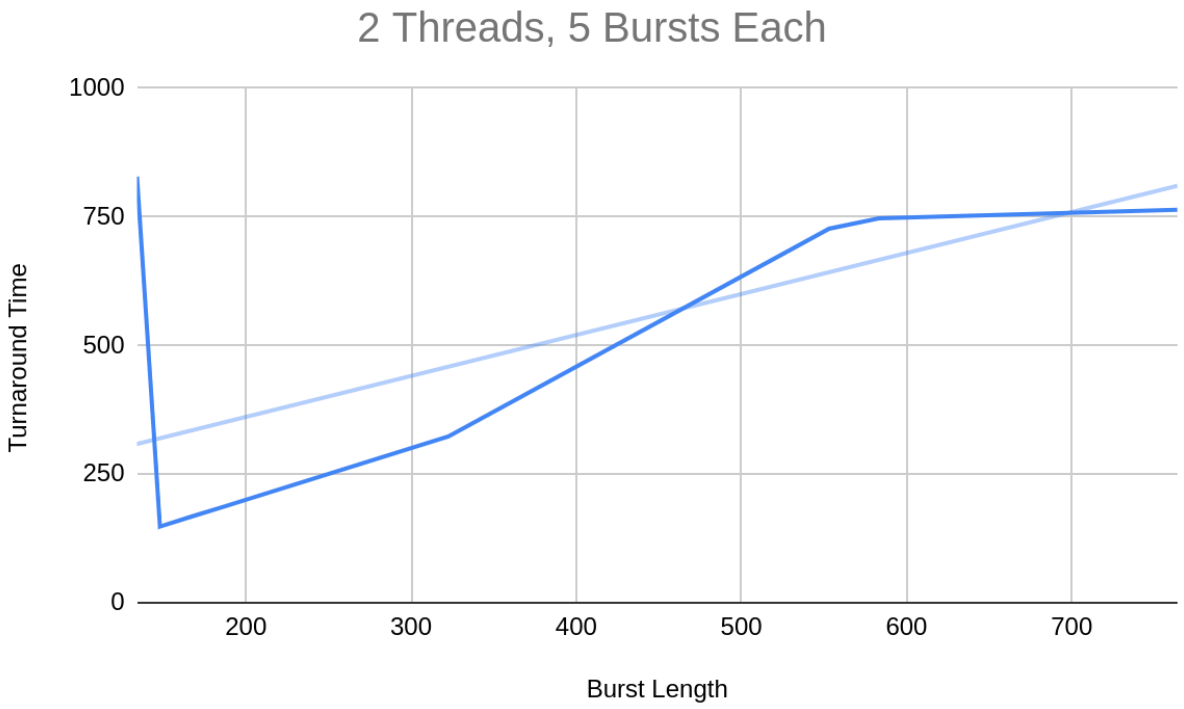

4 Threads, 5 Bursts Each
800
600
400
200
0
Turnaround Time
200
300
400
500
600
700
Burst Length

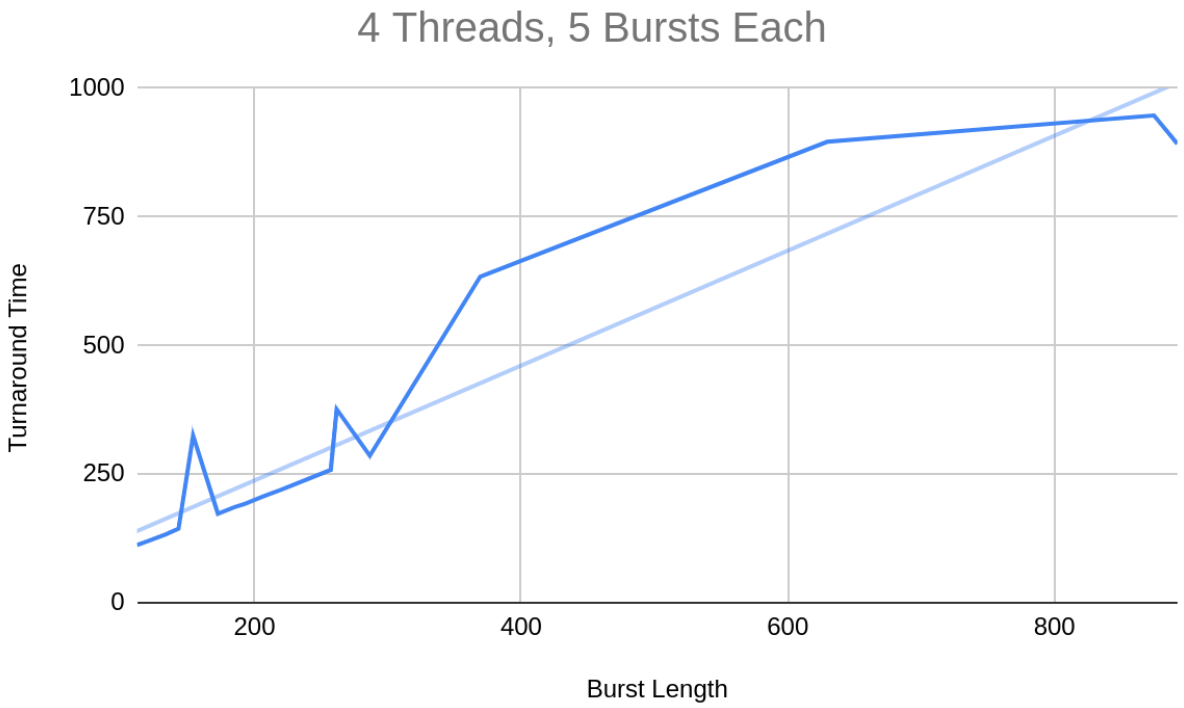

8 Threads, 5 Bursts Each
3000
2000
1000
0
Turnaround Time
200
400
600
800
Burst Length

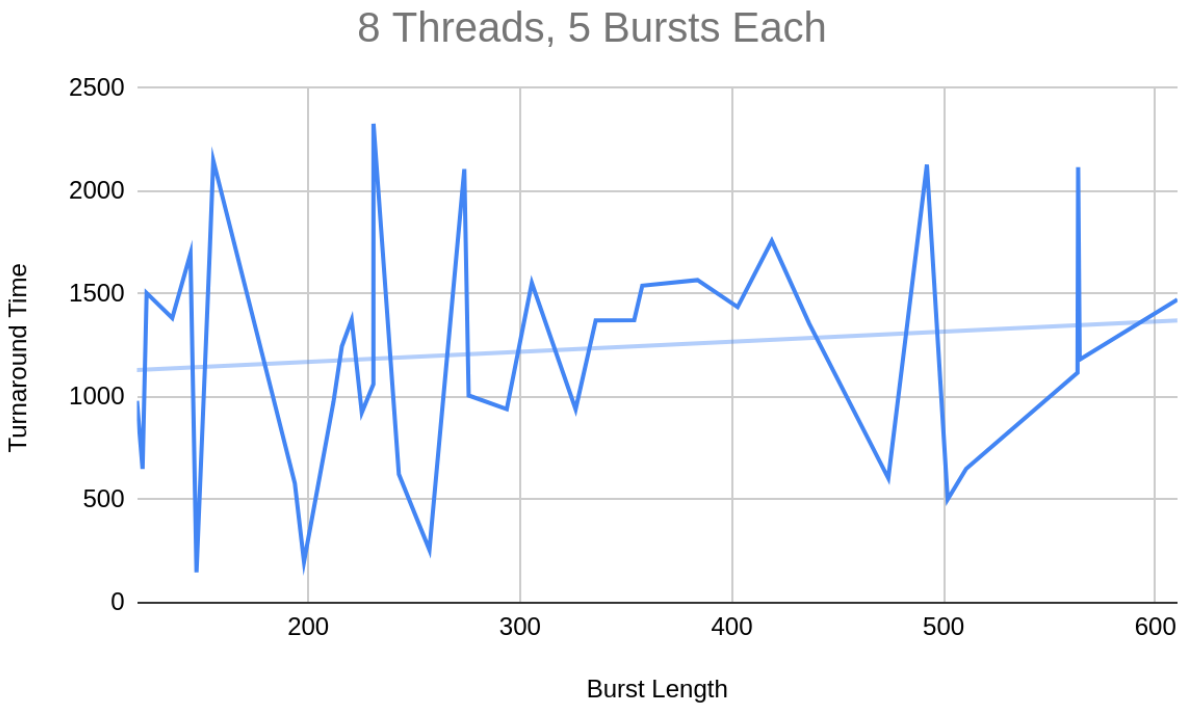


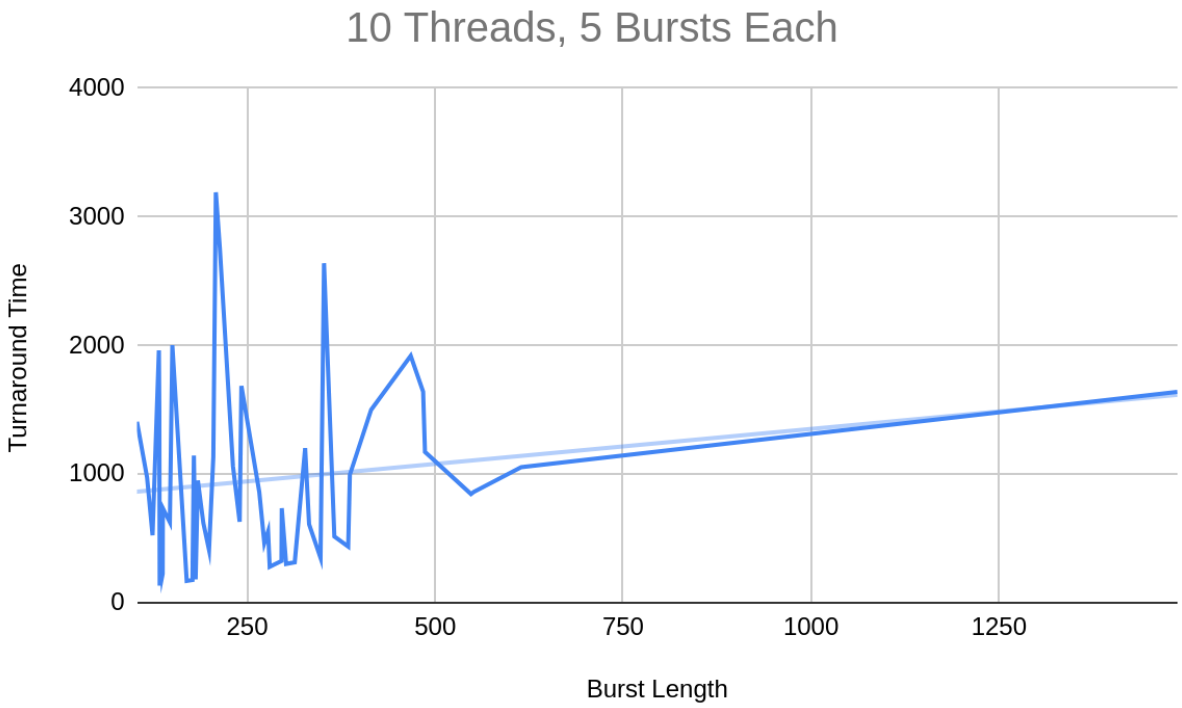
The explanation for PRIO is similar to what is mentioned before. A thread with a lower number will always be executed first. Therefore, these threads are likely to have a lower turnaround time compared to threads with a higher thread index. This algorithm is best suited to distinguish between kernel thread bursts and user thread bursts.

VRUNTIME


2 Threads, 5 Bursts Each
1000
750
500
250
0
Turnaround Time
200
300
400
500
600
700
Burst Length


4 Threads, 5 Bursts Each
1000
750
500
250
0
Turnaround Time
200
400
600
800
Burst Length





This algorithm tends to prefer threads with a lower thread index. The trendline shows a very slow increase. No matter the length of the bursts of smaller indexed threads, they are given a priority over the larger indexed threads. Initially, this algorithm is FCFS which can result in the

spikes as shown in the graph. It is likely that a large length burst takes over the CPU during this period. This algorithm is again suitable where a differentiation of kernel thread bursts and user thread burst is required.

Conclusion

It can be said that the SJF algorithm is better than FCFS where the total number of bursts is lower. Since all algorithms are non-preemptive, the convoy effect can not be avoided completely. SJF comes closest among these algorithms in minimising the convoy effect. For PRIO and VRUNTIME, it is better to use these threads in scenarios where kernel thread bursts have to be distinguished from the user thread bursts. Since kernel threads are likely to be small in length, these algorithms will be beneficial there. VRUNTIME algorithm requires the additional storage of vruntimes for each thread, which might pose a memory issue for large thread numbers. According to the test results, SJF is the best algorithm for general usage where kernel thread burst differentiation from user thread burst is not required. However, where the differentiation is required, PRIO is definitely a better algorithm than VRUNTIME since it does not have a storage overhead.