Comparison of process planning algorithms in order to determine their effectiveness in an operating system through a web simulation

Nancy Yaneth Gelvez García [#1], Danilo López Sarmiento [#2], Nelson Vera Parra[#3]
[# 1] Full Time Professor at Universidad Distrital Francisco José de Caldas,
Faculty of Engineering, Bogotá (Colombia-South America)
nygelvezg@udistrital.edu.co
[#2] Full Time Professor at Universidad Distrital Francisco José de Caldas,
Faculty of Engineering, Bogotá (Colombia-South America)
dalopezs@udistrital.edu.co
[#3] Full Time Professor at Universidad Distrital Francisco José de Caldas,
Faculty of Engineering, Bogotá (Colombia-South America
neverap@udistrital.edu.co

Abstract — This article seeks to compare process planning algorithms in operating systems; for this purpose test cases are defined for each of them, setting different processes which have a lifetime, resource and priority in order to perform an analysis and comparison. Similarly, graphical tools are used to contrast performance and lifetime. Advantages and disadvantages of each algorithm are evidenced based on the metrics set for processing algorithms within a web simulation.

Keywords — Planning algorithm, Operating Systems, processes, performance, metrics, resource, priority, appropriative/preemptive, non- appropriative/non-preemptive, queues, gantt.

I. INTRODUCTION

Most computer systems require an operating system as interface to display the processes running and the hardware that they will use for their implementation during the set lifetime. Thus it can be stated that an operating system is a set of programs or software that manages system resources and serves as the interface for the user to perform any actions on the system. In short, the operating system together with the hardware form a computer system. [1, 2]

Today modern operating systems make use of different methods of planning processes in order to perform essential tasks for them, thus we can assume they are becoming increasingly complex, due to the ease of switching from a single-task environment to a multitask one [3]. That is, through its initial configuration the operating system proceeds to execute processes and assign them to the CPU, ensuring fairness and avoiding inanition at all costs, in order to allow a series of processes at the same time, maximize CPU utilization of the CPU and switch between different processes with such a high frequency that users can interact with each program while it executes [4].

With this background, and understanding that operating systems are a vital part of any computer system, it opens up to the large and complex world that encompasses everything related to planning processes along with planning algorithms. It can be said that what this process planner seeks is to meet these objectives, selecting a process available (within a set of multiple processes) for it to be executed in the CPU. It is important to note that in single processor systems, there will never be more than one process running: if more, it will have to wait for the CPU to become available and another process may be assigned to it [5].

Knowing that the operation of each of these algorithms is based on the concept of queues that host all processes of the computer system, we proceed to develop a study showing and clarifying the notions of the most used planning algorithms, in addition to their efficiency and effectiveness when addressing various processes in the system. This simulation aims to make a comparison through their implementation in a web environment.

II. THEORETICAL FRAMEWORK

A. Processes

It could be argued that the processes are the vital unit of the whole operating system, therefore they are the most important concept that is immersed within an operating system, because they are a small abstraction of a running program that has the ability to operate concurrently, even when there is only one CPU available [1].

Also noteworthy is that every process executed changes status as time passes and this is defined as the current activity of that process. Several types of statuses (Fig. 1) which are defined as:

- **Execution:** The process currently running.
- **Ready:** processes waiting to be executed.
- **Blocked:** processes that can not be run until a certain event ends, an I/O operation.
- **New:** a process that was just created.
- **Completed:** a process that has completed its function, was executed and completed.

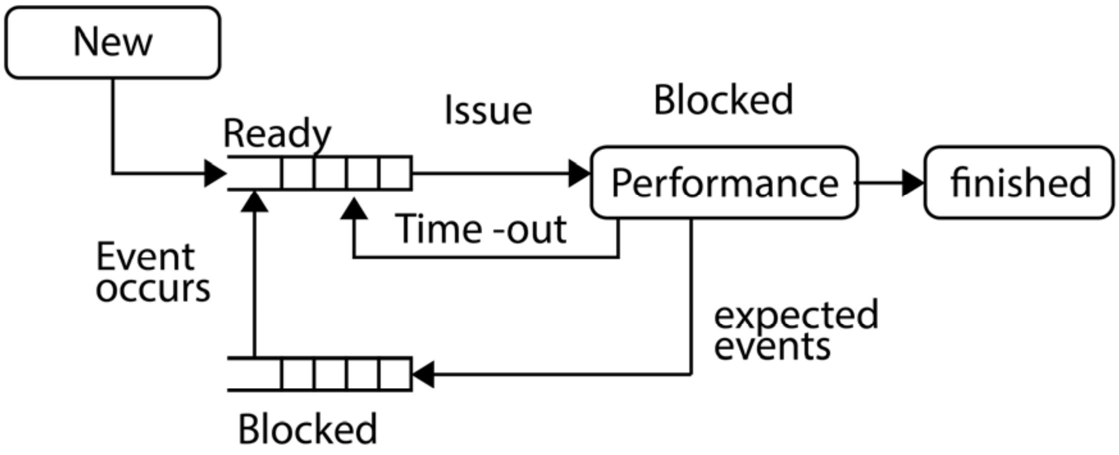


Fig. 1. Five-status model. [6]

Some examples of processes stand out, such as word processors, a web browser and an email program, seeking to achieve a result defined in terms of concrete actions to be performed within a given time [7]. Such processes are often associated with certain properties such as name, resources needed, runtimes or probable completion, priority, etc. [8].

B. Process planning

Each time the operating system is executed, the user may or may not make the decision to execute one or more processes simultaneously, then each process is assigned a ready state and arranged to be serviced by the CPU. This decision which is a part of the operating system is known as process planner and the algorithm used to service the processes that are in the ready queue is known as planning algorithm [1].

Not surprisingly, depending on the environment different planning algorithms are required based on the objectives that it wants to meet. Consequently, optimization within the planner is a subjective concept depending on the system that uses it [1].

Then, throughout the centuries a series of algorithms have been developed that according to their nature serve different purposes. They are governed by the precept of what each of them should do and what a good planning algorithm should have, according to the goal that must be accomplished they will be desirable or not, within the application environment [1, 5], below some of the basic criteria will be explained:

- **Utilization of the CPU:** desire to keep the CPU as busy as is possible.
- **Processing rate:** the number of processes that pass over to the completed queue per unit of time.
- **Runtime:** how long it takes to execute a process. Defined as the interval from the instant the process execution is ordered to the time it is completed.
- **Wait time:** the sum of the cost in time of some processes while waiting in the ready queue. Note that the CPU planning algorithm only affects the time that a process invests waiting in the ready queue.
- **Response time:** time elapsed since the request is submitted until the first response occurs.

Having clarified these concepts, it is said that the desired objective is to maximize CPU utilization and processing rate, minimizing the wait time, the execution time and response time. But we should also consider a very important feature in planning algorithms and that is the type of discipline applied, which are divided into [9]:

- Appropriative: One the CPU has been assigned a process, it cannot be withdrawn, i.e., it is non-ejecting.
- Non-appropriative: One the CPU has been assigned a process, it can be withdrawn, i.e., it ejects the process.

III. SIMULATION DEVELOPMENT

Having introduced the basic concepts of operating systems, we proceed to do a web simulation where planning algorithms are evaluated: Round Robin, Shortest Job First (SJF), Short Time Remaining First (SRTF), planning by appropriative and non-appropriative priority, multiple queues and multiple queues with feedback.

In order to evaluate the individual behavior of each of the planning algorithms, a metric is set based on the one proposed by "Tanenbaum" [1] with certain modifications. In Equation 1, we can see clearly how each of the criteria is defined for this particular exercise as it is important to note that the five-status model will not be used entirely, therefore it does not apply fully. However the same concepts previously explained for planning algorithms are used.

$$\begin{aligned}
&\textit{Response time (t)= time elapsed since the creation of the process until it is served on the CPU}\\
&\textit{Process time = lifetime}\\
&\textit{Wait time (TE)= Response time – Process time}\\
&\textit{TimeP= TE + t, where t equals performance}\\
&\textit{Penalty (P)= Response time /Process time}\\
&\textit{Response portion= 1/ Penalty}\\
&\textit{Total time = sum of total time of running processes}\\
&\textit{CPU utilization = (total time*100) / execution time of simulation}
\end{aligned} \tag{1}$$

For clarity, the statuses that are handled in this simulation are as follows:

- **Completed:** the process successfully completed its lifetime or quantum, as appropriate.
- **Ready:** the process is waiting to be executed.
- **Blocked:**the process cannot run and will remain in this status until the resource it needs to fulfill its function is released.
- **On hold:** the process was interrupted by an action that occurred within the system and will have to wait until the penalty time stipulated for this queue is met.

Additionally, to measure and calculate the effectiveness of each of the planning algorithms which is one of the pillars of this study, the following Equation is proposed:

$$\textit{Effectiveness = (Wait time – Time in CPU) / Total processes} \tag{2}$$

Applying these criteria will allow not only to give qualitative values on the performance of each of these planning algorithms, but also assign a quantitative value which will allow to conclude which one is the best time to avoid process inanition. Consequently, for the development of this simulation it is crucial that the same processes are used for all and each one of them, as shown in Table 1, in order for the simulations to be under the same conditions ensuring the accuracy and validity of the results.

TABLA I. Process description by processor

NAME	LIFETIME	PRIORITY	RESOURCE
Processor 1			
P0	34	1	Printer
P1	54	3	Speakers
P2	23	1	Keyboard
P3	65	2	Printer
P4	32	3	Printer
P5	76	1	Mouse
P6	54	2	Keyboard
P7	12	3	Mouse
P8	35	3	Speakers
P9	83	2	Keyboard
Processor 2			
P0	90	1	Speakers
P1	76	2	Printer
P2	35	1	Speakers
P3	89	2	Mouse
P4	70	3	Mouse
P5	47	2	Speakers
P6	85	2	Keyboard
P7	13	3	Screen
Processor 3			
P0	78	1	Speakers
P1	42	3	Keyboard
P2	67	2	Keyboard
P3	45	3	Mouse
P4	86	1	Printer
P5	94	2	Printer
P6	23	1	Printer
P7	13	2	Screen
P8	54	2	Screen
P9	87	3	Mouse
P10	32	2	Printer
P11	72	1	Keyboard
P12	49	3	Speakers

A. Round Robin

One of the oldest, most simple, impartial and widely used algorithms is Round Robin or circular shift. In which each process is assigned a small unit of time, known as quantum, which ensures that the CPU is shared equally among all processes and there is not a process that monopolizes. The Quatum decreases only when the process is in the CPU or critical zone, once this portion of time, if the process still has lifetime, this becomes suspended state and the quantum will be recalculated to go again to the ready queue. Otherwise, the process state passes over [10]. Also noteworthy is that the processes in the ready queue are governed by the concept of FIFO (First Input, First Output), first in, first out.

One of the most interesting questions of this algorithm is the length of time quantum and appropriation of the CPU until the quantum of time has expired. Despite being these innovative features in its infancy, this creates a major performance problem, because of the dependence you have a good estimate of the quantum; so, if it is small, all processes will be interrupted several times, affecting processing speed [11].

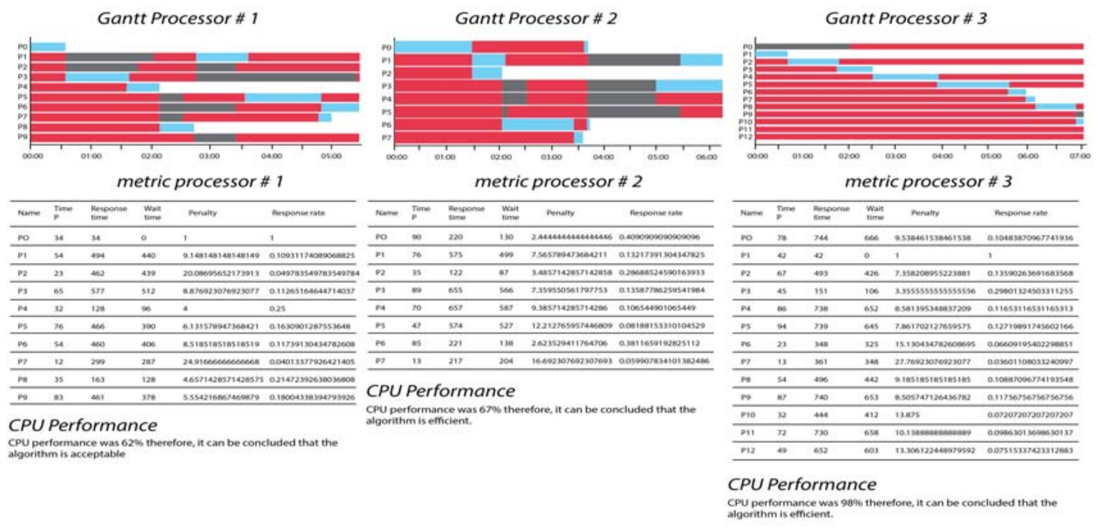
Then in Fig. 2, the results obtained from simulation and values that threw its metric is.

B. Shortest Job Firs (SJF)

This algorithm gives priority to processes that have a shorter lifetime within the ready queue. Every time SJF is executed, the queue is order by lifetime for the one with the shortest time to be the first to run. As this is a non-ejecting or appropriative algorithm, the process that is in critical zone (CPU) will not be interrupted until completion or until an interruption occurs in the system [11]. One feature to highlight is that shall there be a tie, this is settled by FIFO.

It is important to understand that in the absence of a quantum, the process will not enter the on hold status unless an interruption occurs in the system. Excluding the above it can be said that the behavior of the blocked and completed queue is the same as R.R.

Because of its nature, it is possible to say that SJF is optimal in the sense that the wait time for processes that have less time is minimal, but processes with too large times will present a very high wait time, therefore this mechanism is said to be useful for long-term planning.



metric processor # 1

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	34	34	0	1	1
P1	54	494	440	9.148148148148149	0.10931174089068825
P2	23	462	439	20.08695652173913	0.04978354978354784
P3	65	577	512	8.876923076923077	0.11265164644714037
P4	32	128	96	4	0.25
P5	76	466	390	6.131578947368421	0.1630901287553648
P6	54	460	406	8.518518518518519	0.11739130434782608
P7	12	299	287	24.916666666666668	0.04013377926421405
P8	35	163	128	4.6571428571428575	0.21472392638036808
P9	83	461	378	5.554216867469879	0.18004338394793926

CPU Performance

CPU performance was 62% therefore, it can be concluded that the algorithm is acceptable

metric processor # 2

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	90	220	130	2.4444444444444446	0.4090909090909096
P1	76	575	499	7.565789473684211	0.13217391304347825
P2	35	122	87	3.4857142857142858	0.28688524590163933
P3	89	655	566	7.359550561797753	0.13587786259541984
P4	70	657	587	9.385714285714286	0.106544901065449
P5	47	574	527	12.212765957446809	0.08188153310104529
P6	85	221	138	2.6235294117647 06	0.3811659192825112
P7	13	217	204	16.692307692307693	0.059907834101382486

CPU Performance

CPU performance was 67% therefore, it can be concluded that the algorithm is efficient.

metric processor # 3

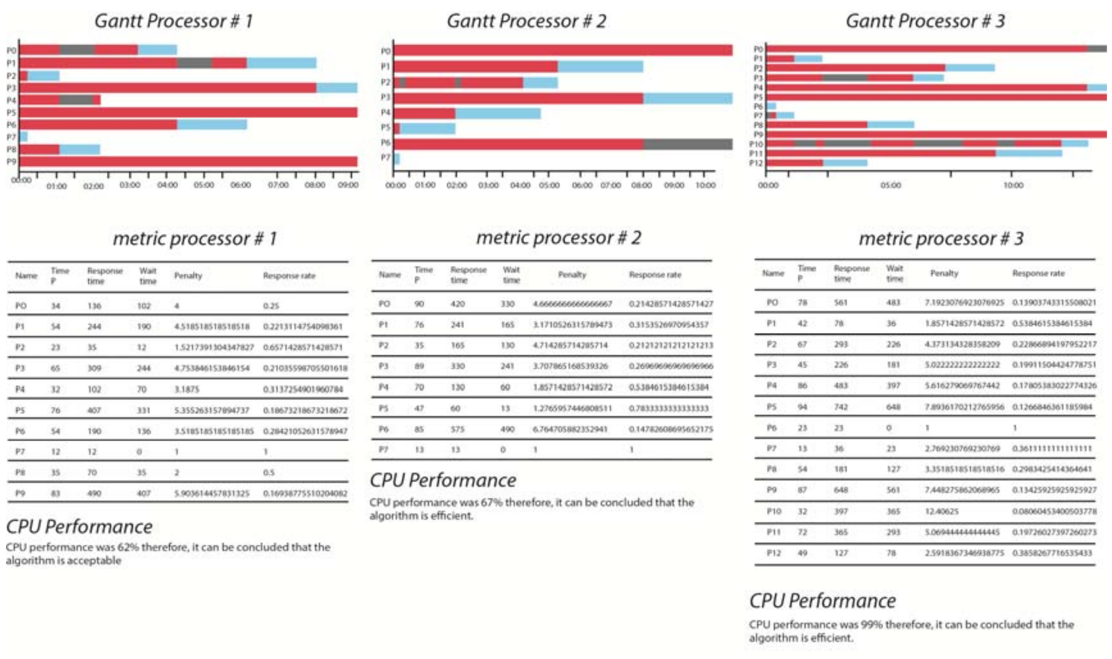
Name	Time P	Response time	Wait time	Penalty	Response rate
PO	78	744	666	9.538461538461538	0.10483870967741936
P1	42	42	0	1	1
P2	67	493	426	7.358208955223881	0.13590263691683568
P3	45	151	106	3.3555555555555556	0.29801324503311255
P4	86	738	652	8.581395348837209	0.11653116531165313
P5	94	739	645	7.861702127659575	0.12719891745602166
P6	23	348	325	15.130434782608695	0.06609195402298851
P7	13	361	348	27.76923076923077	0.036011080332409 97
P8	54	496	442	9.185185185185185	0.10887096774193548
P9	87	740	653	8.505747126436782	0.11756756756756756
P10	32	444	412	13.875	0.07207207207207207
P11	72	730	658	10.13888888888889	0.09863013698630137
P12	49	652	603	13.306122448979592	0.07515337423312883

CPU Performance

CPU performance was 98% therefore, it can be concluded that the algorithm is efficient.

Fig. 2. Round Robin planning algorithm

Fig. 3 shows the results obtained after applying the SJF algorithm to a set of previously established processes.



metric processor # 1

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	34	136	102	4	0.25
P1	54	244	190	4.518518518518518	0.2213114754098361
P2	23	35	12	1.5217391304347827	0.6571428571428571
P3	65	309	244	4.753846153846154	0.21035598705501618
P4	32	102	70	3.1875	0.3137254901960784
P5	76	407	331	5.355263157894737	0.18673218673218672
P6	54	190	136	3.5185185185185185	0.28421052631578947
P7	12	12	0	1	1
P8	35	70	35	2	0.5
P9	83	490	407	5.903614457831325	0.16938775510204082

CPU Performance

CPU performance was 62% therefore, it can be concluded that the algorithm is acceptable

metric processor # 2

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	90	420	330	4.666666666666667	0.21428571428571427
P1	76	241	165	3.1710526315789473	0.3153526970954357
P2	35	165	130	4.714285714285714	0.21212121212121213
P3	89	330	241	3.707865168539326	0.2696969696969696966
P4	70	130	60	1.8571428571428572	0.5384615384615384
P5	47	60	13	1.2765957446808511	0.7833333333333333
P6	85	575	490	6.764705882352941	0.14782608695652175
P7	13	13	0	1	1

CPU Performance

CPU performance was 67% therefore, it can be concluded that the algorithm is efficient.

metric processor # 3

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	78	561	483	7.1923076923076925	0.13903743315508021
P1	42	78	36	1.8571428571428572	0.5384615384615384
P2	67	293	226	4.373134328358209	0.22866894197952217
P3	45	226	181	5.022222222222222	0.19911504424778751
P4	86	483	397	5.616279069767442	0.17805383022774326
P5	94	742	648	7.8936170212765956	0.1266846361185984
P6	23	23	0	1	1
P7	13	36	23	2.769230769230769	0.3611111111111111
P8	54	181	127	3.3518518518518516	0.2983425414364641
P9	87	648	561	7.448275862068965	0.13425925925925927
P10	32	397	365	12.40625	0.08060453400503778
P11	72	365	293	5.069444444444445	0.19726027397260273
P12	49	127	78	2.5918367346938775	0.3858267716535433

CPU Performance

CPU performance was 99% therefore, it can be concluded that the algorithm is efficient.

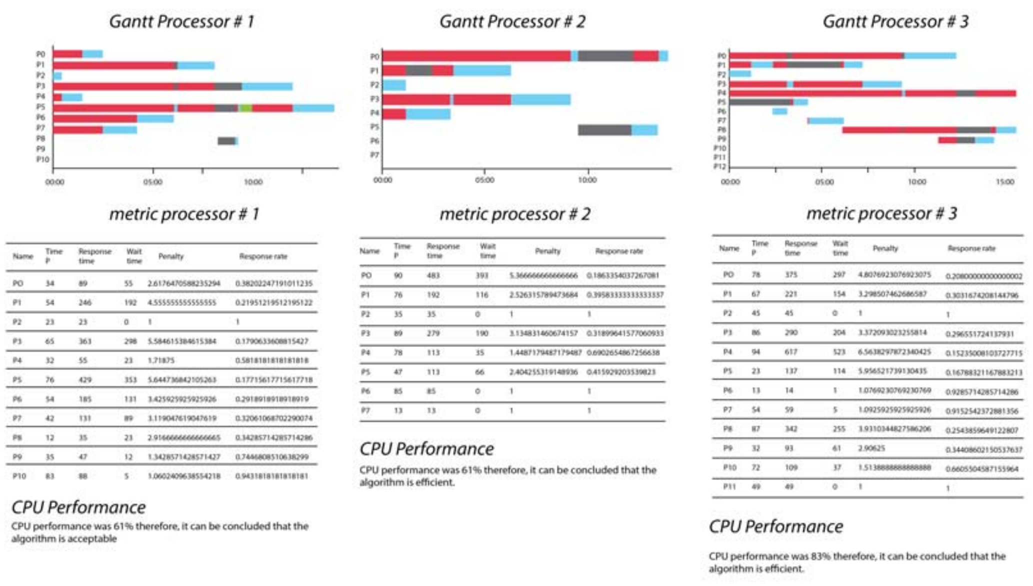
Fig. 3. SJF planning algorithm

C. Short Remaining Time First (SRTF)

It is an improved version of SJF and therefore, is very similar. The difference is that the design is based on a non-appropriative algorithm, i.e. it ejects the processes found in the critical zone depending on the condition to be established [11].

In this case the condition of removal is based on whether in the ready queue there is a process that has a shorter lifetime compared to that found in the CPU, then it will be interrupted and sent to the on hold queue until the penalty time finishes and it returns to the ready status with the remaining lifetime.

Fig. 4 shows the application of this algorithm in terms of the Gantt chart and its outcome in terms of the metric proposed.



metric processor # 1

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	34	89	55	2.6176470588235294	0.38202247191011235
P1	54	246	192	4.555555555555555	0.21951219512195122
P2	23	23	0	1	1
P3	65	363	298	5.584615384615384	0.17906336088815427
P4	32	55	23	1.71875	0.5818181818181818
P5	76	429	353	5.644736842105263	0.17715617715617718
P6	54	185	131	3.425925925925926	0.2918918918918919
P7	42	131	89	3.119047619047619	0.32061068702290074
P8	12	35	23	2.9166666666666665	0.34285714285714286
P9	35	47	12	1.3428571428571427	0.7446808510638299
P10	83	88	5	1.0602409638554218	0.9431818181818181

CPU Performance

CPU performance was 61% therefore, it can be concluded that the algorithm is acceptable

metric processor # 2

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	90	483	393	5.366666666666666	0.18633540372670881
P1	76	192	116	2.526315789473684	0.39583333333333337
P2	35	35	0	1	1
P3	89	279	190	3.134831460674157	0.31899641577060933
P4	78	113	35	1.4487179487179487	0.6902654867256638
P5	47	113	66	2.404255319148936	0.415929203539823
P6	85	85	0	1	1
P7	13	13	0	1	1

CPU Performance

CPU performance was 61% therefore, it can be concluded that the algorithm is efficient.

metric processor # 3

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	78	375	297	4.8076923076923075	0.2080000000000002
P1	67	221	154	3.298507462686587	0.3031674208144796
P2	45	45	0	1	1
P3	86	290	204	3.372093023255814	0.296551724137931
P4	94	617	523	6.5638297872340425	0.15235008103727715
P5	23	137	114	5.956521739130435	0.16788321167883213
P6	13	14	1	1.0769230769230769	0.9285714285714286
P7	54	59	5	1.0925925925925926	0.9152542372881356
P8	87	342	255	3.9310344827586206	0.2543859649122807
P9	32	93	61	2.90625	0.34408602150537637
P10	72	109	37	1.5138888888888888	0.6605504587155964
P11	49	49	0	1	1

CPU Performance

CPU performance was 83% therefore, it can be concluded that the algorithm is efficient.

Fig. 4. SRTF planning algorithm

D. *Appropriative priority algorithm*

This algorithm establishes an initial priority to all and each of the processes to be executed within the system, from a series of well-defined criteria:

- **High** – system process, also known as priority 1.
- **Medium –** processes performed by the user, also known as priority 2.
- **Low** – input and output processes, also known as priority 3.

Once this priority is established, processes will be served according to their importance in the processor, for this exercise, the top priority will be the system processes. Should there be a process running and in the ready queue there is a higher priority process, it cannot be served until the process in the critical area ends its lifetime [11]. Therefore, it can be said that this type of algorithm is similar to SJF, the only difference is their ordering criteria.A major problem with priority planning algorithms is the indefinite block or inanition, because the low-priority processes can stay waiting indefinitely to be served while there are still processes with higher priority [11].In Fig. 5 the graphical and numerical implementation of this algorithm results are shown.

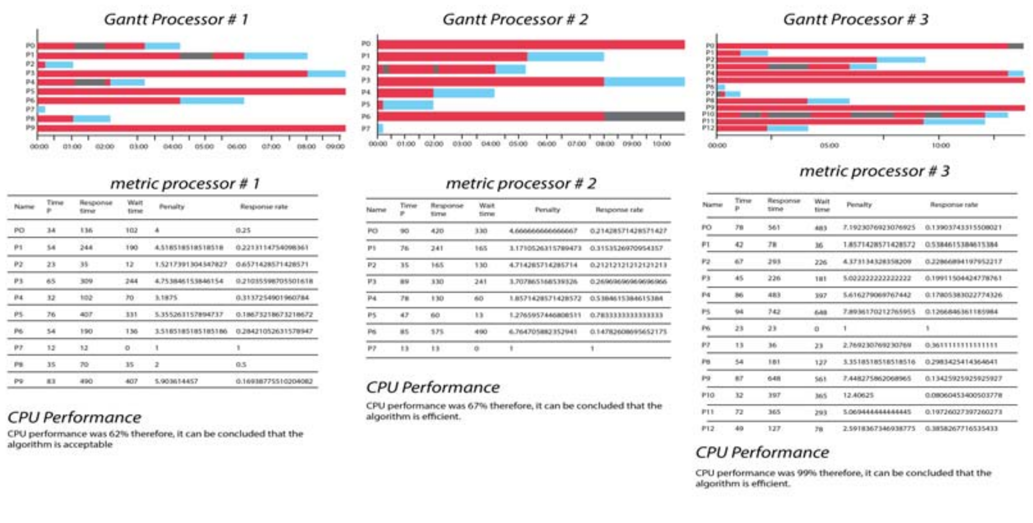
E. *Non- appropriative algorithm with Priority*

The non-appropriative algorithm with priority s a variation of the algorithm described in Section D, except this one has the ability to eject the process that is running in critical zone. If upon comparison with those found in the ready queue, there is a process with higher priority, this process will go into on hold status and upon ending its penalty time it will go back to the ready queue. In case there are multiple processes with the same priority, they will be served by FIFO.After running the algorithm, the results from the metric used were obtained, where inanition is less evident in the processes regarding the appropriative priority algorithm. For more clarity, see Fig. 6.

F. *Multiple queues*

This kind of planning algorithm is a solution developed to the problems that arise with the above algorithms. It is vital for operating systems where processes coexist with different needs. This algorithm is based on a predetermined scheme which gives special treatment to the processes according to the queue in which they are [11]. Here priorities are also handled, but this time the priority is in the queue, which means that there is a queue (ready, blocked and on hold) that has a higher priority over the others.

It is also worth emphasizing the appropriation of processes over the CPU and a defining characteristic of such algorithms, which specifies that until they find a queue (ready, blocked and on hold) that is completely free, they cannot move on to serve the following [11]. And this condition generates inanition processes in lower priority queues until they are fully served.



metric processor # 1

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	34	136	102	4	0.25
P1	54	244	190	4.518518518518518	0.2213114754098361
P2	23	35	12	1.5217391304347827	0.6571428571428571
P3	65	309	244	4.753846153846154	0.21035598705501618
P4	32	102	70	3.1875	0.3137254901960784
P5	76	407	331	5.355263157894737	0.18673218673218672
P6	54	190	136	3.5185185185185186	0.28421052631578947
P7	12	12	0	1	1
P8	35	70	35	2	0.5
P9	83	490	407	5.903614457	0.16938775510204082

CPU Performance

CPU performance was 62% therefore, it can be concluded that the algorithm is acceptable

metric processor # 2

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	90	420	330	4.666666666666667	0.21428571428571427
P1	76	241	165	3.1710526315789473	0.3153526970954357
P2	35	165	130	4.714285714285714	0.21212121212121213
P3	89	330	241	3.707865168539326	0.26969696969696966
P4	78	130	60	1.8571428571428572	0.5384615384615384
P5	47	60	13	1.2765957446808511	0.7833333333333333
P6	85	575	490	6.764705882352941	0.14782608695652175
P7	13	13	0	1	1

CPU Performance

CPU performance was 67% therefore, it can be concluded that the algorithm is efficient.

metric processor # 3

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	78	561	483	7.1923076923076925	0.13903743315508021
P1	42	78	36	1.8571428571428572	0.5384615384615384
P2	67	293	226	4.373134328358209	0.22866894197952217
P3	45	226	181	5.022222222222222	0.19911504424778761
P4	86	483	397	5.616279069767442	0.17805383022774326
P5	94	742	648	7.8936170212765955	0.1266846361185984
P6	23	23	0	1	1
P7	13	36	23	2.769230769230769	0.3611111111111111
P8	54	181	127	3.3518518518518516	0.2983425414364641
P9	87	648	561	7.448275862068965	0.13425925925925927
P10	32	397	365	12.40625	0.08060453400503778
P11	72	365	293	5.069444444444445	0.19726027397260273
P12	49	127	78	2.5918367346938775	0.3858267716535433

CPU Performance

CPU performance was 99% therefore, it can be concluded that the algorithm is efficient.

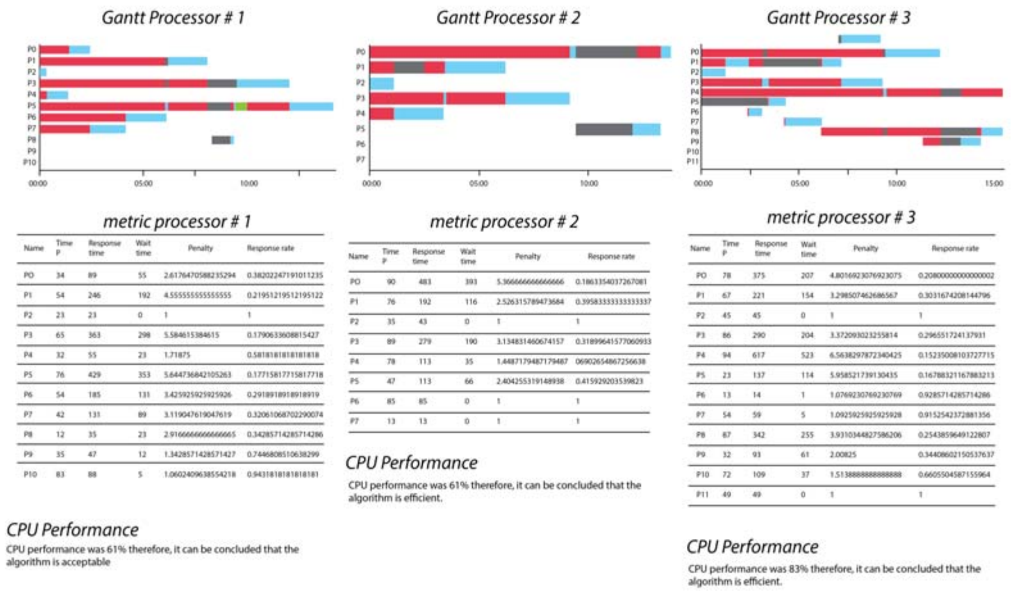
Fig. 5. Appropriative priority planning

For this exercise only three lines were used, each programmed with different Round Robin, SRTF and FIFO planning algorithms, assigning high, medium and low priorities respectively. After completion of the algorithm, the following results were obtained (see Fig. 7).

G. Multiple queues with feedback

This algorithm is governed under the same concept of multiple queues, which allows the movement of processes from one queue to another when a process takes too long in the CPU or when its aging time is over, but the difference is that the CPU is not being appropriated, if for example, there is a process in the critical area that has priority 3 and one with lower priority reaches the system it will be rejected and put on hold. This means that this algorithm performs a comparison between the CPU and the ready queues to determine which process should be executed first by priority [11].

Multiple queues with feedback as a solution to the major problems of inanition and indefinite blocking apply the aging mechanism, a technique to gradually increase the priority of the processes that have been waiting a long time in the system [11]. That is, a process that has been waiting too long in line 3, upon ending its aging time will jump to line 2 and so on.



metric processor # 1

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	34	89	55	2.6176470588235294	0.38202247191011235
P1	54	246	192	4.555555555555555	0.21951219512195122
P2	23	23	0	1	1
P3	65	363	298	5.584615384615	0.1790633608815427
P4	32	55	23	1.71875	0.5818181818181818
P5	76	429	353	5.644736842105263	0.17715817715817718
P6	54	185	131	3.425925925925926	0.2918918918918919
P7	42	131	89	3.119047619047619	0.32061068702290074
P8	12	35	23	2.9166666666666665	0.34285714285714286
P9	35	47	12	1.3428571428571427	0.7446808510638299
P10	83	88	5	1.0602409638554218	0.9431818181818181

CPU Performance

CPU performance was 61% therefore, it can be concluded that the algorithm is acceptable

metric processor # 2

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	90	483	393	5.366666666666666	0.18633540372670081
P1	76	192	116	2.526315789473684	0.39583333333333337
P2	35	43	0	1	1
P3	89	279	190	3.134831460674157	0.31899641577060933
P4	78	113	35	1.4487179487179487	06902654867256638
P5	47	113	66	2.404255319148938	0.415929203539823
P6	85	85	0	1	1
P7	13	13	0	1	1

CPU Performance

CPU performance was 61% therefore, it can be concluded that the algorithm is efficient.

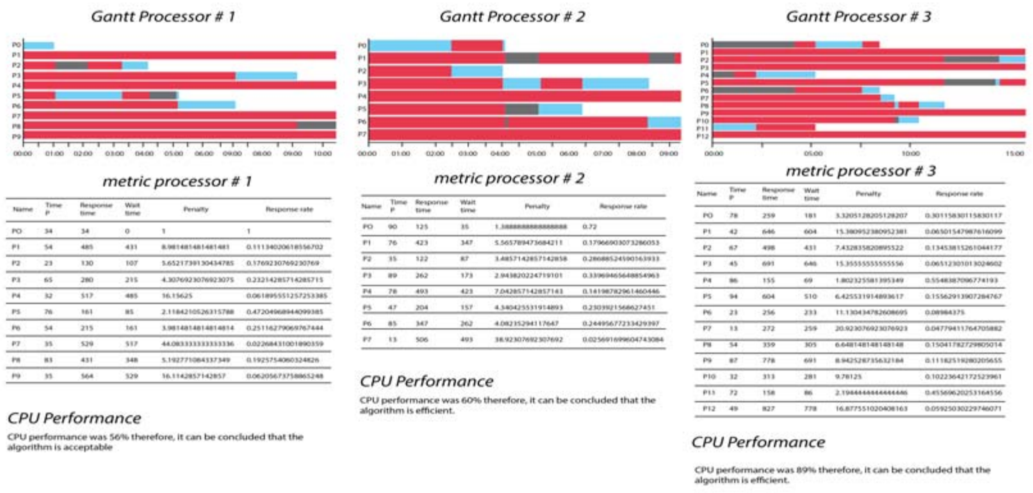
metric processor # 3

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	78	375	207	4.8016923076923075	0.20800000000000002
P1	67	221	154	3.298507462686567	0.3031674208144796
P2	45	45	0	1	1
P3	86	290	204	3.372093023255814	0.296551724137931
P4	94	617	523	6.5638297872340425	0.15235008103727715
P5	23	137	114	5.958521739130435	0.16788321167883213
P6	13	14	1	1.0769230769230769	0.9285714285714286
P7	54	59	5	1.0925925925925928	0.9152542372881356
P8	87	342	255	3.9310344827586206	0.2543859649122807
P9	32	93	61	2.00825	0.34408602150537637
P10	72	109	37	1.5138888888888888	0.6605504587155964
P11	49	49	0	1	1

CPU Performance

CPU performance was 83% therefore, it can be concluded that the algorithm is efficient.

Fig. 6. Non-appropriative planning algorithm with priority



metric processor # 1

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	34	34	0	1	1
P1	54	485	431	8.981481481481481	0.11134020618556702
P2	23	130	107	5.6521739130434785	0.1769230769230769
P3	65	280	215	4.3076923076923075	0.23214285714285715
P4	32	517	485	16.15625	0.061895551257253385
P5	76	161	85	2.1184210526315788	0.4720496894409385
P6	54	215	161	3.981481481481814	0.2511627906976744
P7	35	529	517	44.083333333333336	0.022684310018903590
P8	83	431	348	5.192771084337349	0.19257540603248226
P9	35	564	529	16.1142857142857	0.06205673758865248

CPU Performance

CPU performance was 56% therefore, it can be concluded that the algorithm is acceptable

metric processor # 2

Name	Time P	Response time	Wait time	Penalty	Response rate
PO	90	125	35	1.3888888888888888	0.72
P1	76	423	347	5.565789473684211	0.17966903073286053
P2	35	122	87	3.4857142857142858	0.28688524590163933
P3	89	262	173	2.943820224719101	0.33969465648854963
P4	78	493	423	7.042857142857143	0.14198782961460446
P5	47	204	157	4.340425531914893	0.2303921568627451
P6	85	347	262	4.08235294117647	0.24495677233429397
P7	13	506	493	38.92307692307692	0.025691699604743084

CPU Performance

CPU performance was 60% therefore, it can be concluded that the algorithm is efficient.

metric processor # 3

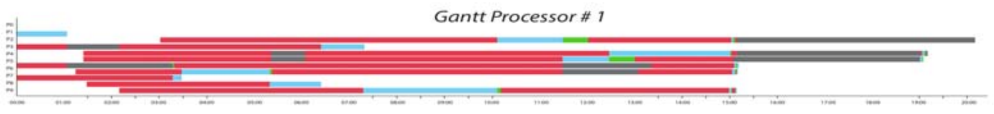
Name	Time P	Response time	Wait time	Penalty	Response rate
PO	78	259	181	3.3205128205128207	0.30115830115830117
P1	42	646	604	15.380952380952381	0.06501547987616099
P2	67	498	431	7.432835820895522	0.13453815261044177
P3	45	691	646	15.355555555555556	0.06512301013024602
P4	86	155	69	1.8023255813953490	0.5548387096774193
P5	94	604	510	6.425531914893617	0.15562913907284767
P6	23	256	233	11.130434782608695	0.08984375
P7	13	272	259	20.923076923076923	0.04779411764705882
P8	54	359	305	6.648148148148148	0.15041782729805014
P9	87	778	691	8.942528735632184	0.11182519280205655
P10	32	313	281	9.78125	0.10223642172523961
P11	72	158	86	2.1944444444444446	0.45569620253164556
P12	49	827	778	16.877551020408163	0.05925030229746071

CPU Performance

CPU performance was 89% therefore, it can be concluded that the algorithm is efficient.

Fig. 7. Multiple queue planning algorithm

Despite the advantages of this adaptation algorithm based on the system that is to be applied, a high complexity arises in defining the best set of planners required for a specific environment. In this case, the same configuration will be used for multiple queues and results will be evaluated as shown in Fig. 8.

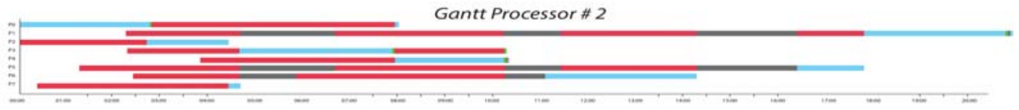


metric processor # 1

Name	Time P	Response time	Wait time	Penalty	Response rate
P0	34	34	0	1	1
P1	54	520	466	9.62962962962963	0.10384615384615384
P2	23	228	205	9.91304347826087	0.10087719298245615
P3	65	527	462	8.107692307692307	0.12333965844402278
P4	32	523	491	16.34375	0.06118546845124283
P5	76	461	385	6.065789473684211	0.16485900216919738
P6	54	415	361	7.685185185185185	0.13012048192771083
P7	35	119	107	9.916666666666666	0.10084033613445378
P8	83	146	111	4.171428571428572	0.23972602739726026
P9	35	389	306	4.686746987951807	0.21336760925449874

CPU Performance

CPU performance was 61% therefore, it can be concluded that the algorithm is acceptable

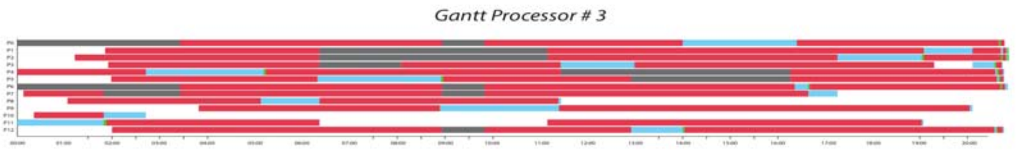


metric processor # 2

Name	Time P	Response time	Wait time	Penalty	Response rate
P0	90	224	134	2.488888888888889	0.4017857142857143
P1	76	515	439	6.776315789473684	0.14757281553398058
P2	35	122	87	3.4857142857142858	0.28688524590163933
P3	89	219	130	2.460674157303371	0.4063926940639269
P4	70	178	108	2.5428571428571427	0.39325842696629215
P5	47	468	411	9.74468085106383	0.10262008733624454
P6	86	322	237	3.7882352941176467	0.2639751552795031
P7	13	114	101	8.76923076923077	0.11403508771929824

CPU Performance

CPU performance was 66% therefore, it can be concluded that the algorithm is acceptable



metric processor # 3

Name	Time P	Response time	Wait time	Penalty	Response rate
P0	78	736	658	9.435897435897436	0.10597826086956522
P1	42	671	629	15.976190476190476	0.06259314456035768
P2	67	685	618	10.223880597014926	0.09781021897810219
P3	45	657	612	14.6	0.0684931506849315
P4	96	734	648	8.534883720930232	0.11716621253405995
P5	94	656	562	6.978723404255319	0.14329268292682926
P6	23	740	717	32.17391304347826	0.031081081081081083
P7	13	599	586	46.07692307692308	0.021702838063439065
P8	54	368	314	6.814814814814815	0.1467391304347826
P9	87	571	404	6.563218390804598	0.15226427320490367
P10	32	75	43	2.34375	0.4266666666666667
P11	72	676	604	9.38888888888889	0.10650887573964496
P12	49	650	601	13.26530612244898	0.07538461538461538

CPU Performance

CPU performance was 98% therefore, it can be concluded that the algorithm is acceptable

Fig. 8. Multiple queues with feedback planning algorithm

IV. ANALYSIS OF RESULTS

After execution of the planning algorithms, the simulation produced the following results that contrast performance versus runtime. It can be seen that time varies depending on the algorithm and in general processor 3 behavior tends toward better performance.

The Round Robin planning algorithm ran all its processes in 751 seconds and its performance on most processors tends to decrease over time (Fig. 9). The troughs show some kind of prolonged inactivity, therefore we can say that the algorithm is acceptable.

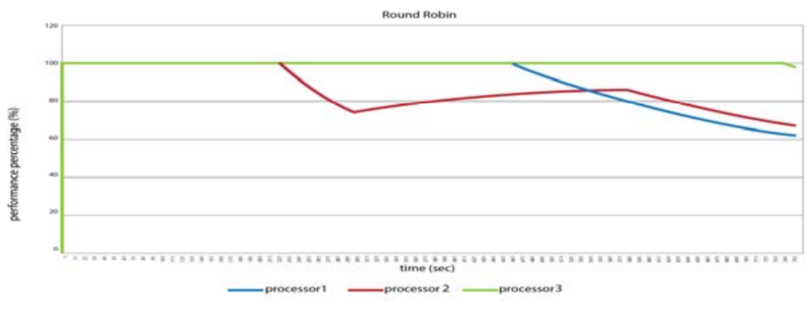


Fig. 9. Performance of Round Robin Planning Algorithm

On the other hand, the SJF algorithm and appropriative priority algorithm show better performance in CPU, since the peaks and troughs are not as pronounced and inactivity of the critical zone starts later with respect to the previous algorithm. Fig. 10 shows that the runtime for the proposed processes was 721.

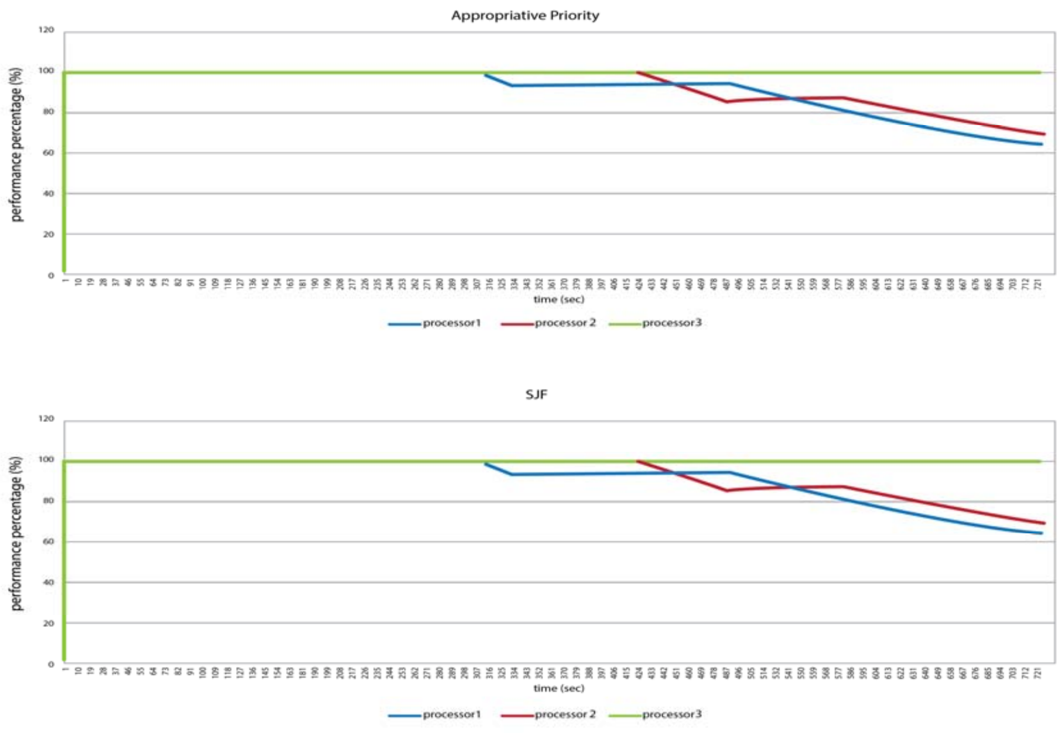


Fig. 10. Performance of SJF and appropriative priority planning algorithms

The SRTF and non-appropriative algorithm with priority, as shown in Fig. 11, show greater variety in performance of the critical zone of each of the processors, as process ejection generates downtimes and strong activity peaks.

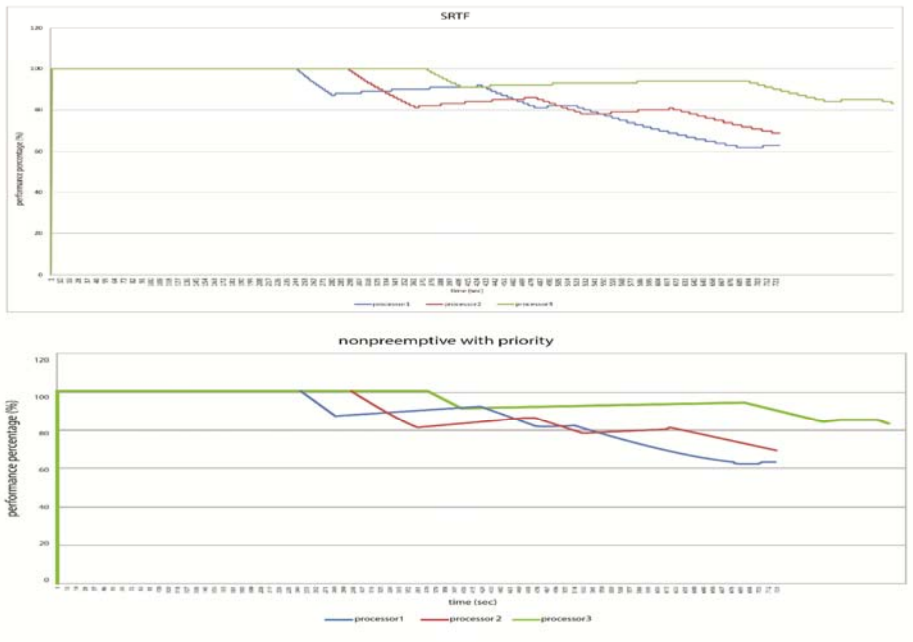


Fig. 11. Performance of SRTF and non-appropriative algorithm with priority

By nature, multiple queues have a greater runtime for process completion. As displayed in Fig. 12, this algorithm shows with more urgency a downtime in critical zones, but performance remains constant over time.

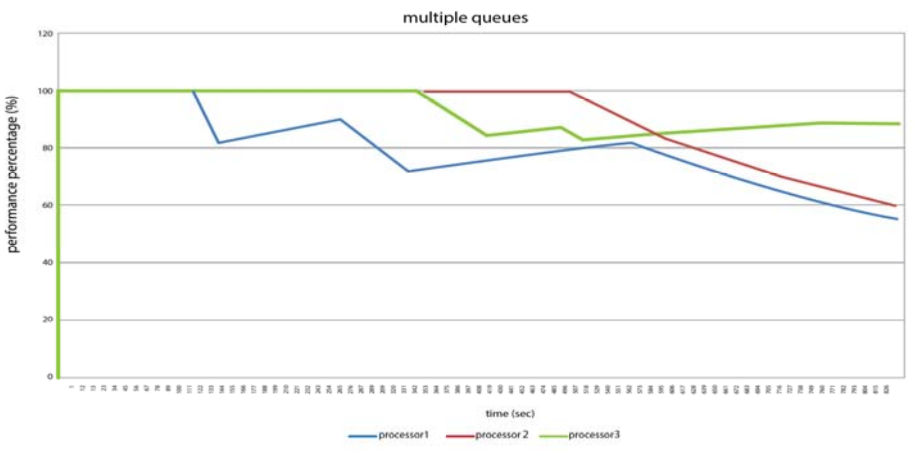


Fig. 12. Performance of multiple queues algorithm

Finally, multiple queues with Feedback (Fig.13) present a more uniform and consistent performance over time since having an aging time makes possible combating inanition processes easier.

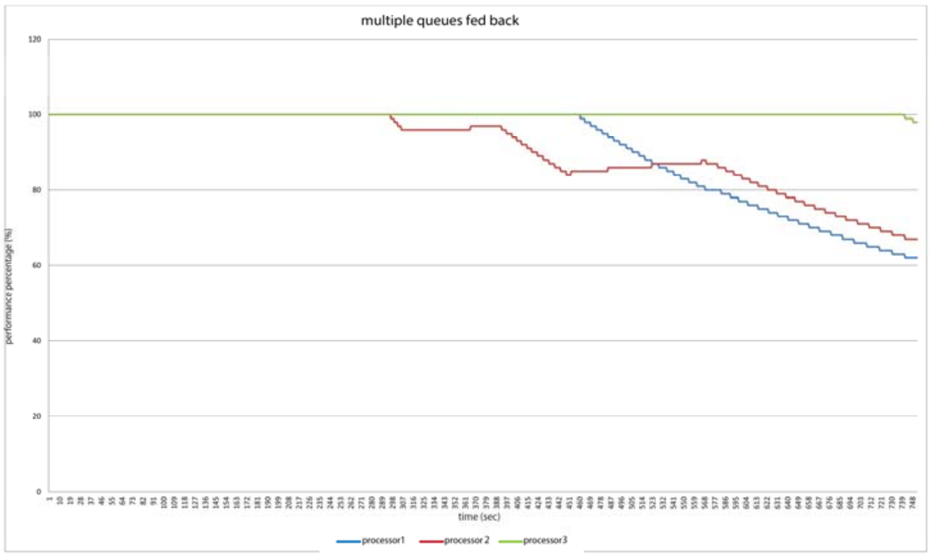


Fig. 13. Performance of multiple queues with feedback algorithm

By averaging the criteria that are involved in the metrics, it can be seen that some algorithms respond better due their nature as evidenced in Table 2, comparing their response time, wait time, penalty time and response rate. It shows that the SRTF algorithm has a better response in all items.

Given that the response time refers to the time it takes for the process to be served, this is one of the most important and conclusive metrics to determine the superiority of an algorithm. Having said that, SRTF and the non-appropriative with priority algorithms manage to optimize management of processor resources, minimizing in each of its processors the indefinite blocking and process inanition.

Another metric that can support the above statement is the "penalty" in which a more lenient process behavior is observed, so they do not waste time in queues designed to penalize but to concentrate more on ready queues.

Also in Table 2, it is clearly seen that is no coincidence given by a context pattern, the similarity in the results obtained by the metric, but instead is a behavior that persists between processors 1, 2 and 3, where despite the change of initial conditions, number of process, lifetime and priorities, they remain around the arithmetic mean.

TABLE II. Average criteria for evaluation of planning algorithms

Processor 1				
Name	**Response time**	**Wait time**	**Penalty**	**Reponses rate**
Round Robin	354,4	307,6	9,3	0,2
SJF	199,5	152,7	3,6	0,4
SRTF	153,7	107,4	3,0	0,5
Appropriative with priority	199,5	152,7	3,6	0,4
Non-Appropriative with priority	153,7	107,4	3,0	0,5
Multiple queues	334,6	287,8	10,8	0,3
Multiple queues with feedback	336,2	289,4	7,8	0,2
Processor 2				
Name	**Response time**	**Wait time**	**Penalty**	**Reponse rate**
Round Robin	405,4	342,3	7,7	0,2
SJF	241,8	178,6	3,4	0,4
SRTF	164,1	100,0	2,2	0,6
Appropriative with priority	241,8	178,6	3,4	0,4
Non-Appropriative with priority	164,1	100,0	2,2	0,6
Multiple queues	310,3	247,1	8,5	0,3
Multiple queues with feedback	269,0	205,9	5,0	0,3
Processor 3				
Name	**Response time**	**Wait time**	**Penalty**	**Response rate**
Round Robin	513,7	456,6	10,4	0,2
SJF	320,0	262,9	5,1	0,3
SRTF	195,9	137,6	3,0	0,5
Appropriative with priority	320,0	262,9	5,1	0,3
Non-Appropriative with priority	195,9	137,6	3,0	0,5
Multiple queues	447,4	390,3	9,7	0,2
Multiple queues with feedback	601,4	544,3	14,0	0,1

V. CONCLUSIONS

Appropriative planning algorithms such as Round Robin, significantly help improve processor performance, thanks to the fulfillment of the objectives of efficient planning, when partitioning process runtime. This addresses its shortcomings by merging with priority algorithms and aging strategies to reach the expected performance.

It is difficult to make a definitive comparison between all planning algorithms, because performance depends on several factors, including the probability distribution of process service times, planning efficiency and mechanisms of change of context.

It can be stated in regard to the simulation presented, that the calculation of quantum length directly affects the performance of the planning algorithm, for example, a short quantum with respect to process lifetime reduces CPU performance, and a long quantum negatively affects response times.

Processes that are assigned a low priority or a very long lifetime, show inanition in planning algorithms that are based on priorities and those sorted by lifetime.

REFERENCES

[1] A.S. Tanenbaum, Sistemas Operativos Modernos. Pearson Educación, México. 3 – 7. 2009.
[2] H.M. Deitel Introducción a los Sistemas Operativos. Addison-Wesley Iberoamericana, México. 1987.
[3] A. Singh,, P. Goyal, S. Batra. An Optimized Round Robin Scheduling Algorithm for CPU Scheduling. International Journal on Computer Science and Engineering Vol. 02. 2383-2385. 2010.
[4] A.Silberchatz, P. BGalvin, G.Gagne. Operating systems concepts. John Wiley & Sons, Inc, Estados Unidos. 2003.
[5] A. Silberchatz, P. BGalvin, G.Gagne. Fundamentos de los sistemas operativos. Mc Graw Hill, España. 2005.
[6] H. J. Ortiz, Sistemas operativos modernos. Sello editorial, Colombia. 2005.
[7] J. A. Jiménez,J. J. Medel,Planificación por prioridades para sistemas SISO a través de un controlador. Latin-American Journal of Physics Education. Vol. 5, Nº. 1. 2010.
[8] A. S. Tanenbaum, M. V Steen Sistemas Distribuidos. Pearson Educación, México. 2008.
[9] D. Martínez, Sistemas Operativos: diseño e implementación. Martin Iturbide, Argentina. 1998.
[10] A. Mclver, IFlynn, Sistemas Operativos. Cengage Learning Editores. México. 2010.
[11] P. Martínez, M. Cabello, J. C. Díaz, Sistemas operativos: teoría y práctica. Ediciones Díaz de Santos. España. 1996.