CS537—Introduction to Operating Systems, Fall 2016

Graph Generation for Project P2B—Timeline of Process Scheduling using MLFQ scheduling.

Authors: Pradeep Kashyap Ramaswamy | pradeepkashyap@cs.wisc.edu | 907 567 9325
Karan Talreja | karan@cs.wisc.edu | 907 567 8186

How to interpret the Graph?

1. The X-axis represents the time in tics. In xv6, each tic is of 10ms.
2. The Y-Axis represents the priority queue. Highest Priority being 0 and lowest being -3. Note here that lower priorities are being represented using **negative number**. This is to make the higher to lower priority, flow from top to bottom. (if we represent the same using positive numbers, the flow will be from bottom to top). So, the **number 0 (zero)** represents the **highest priority** and **-3** represents the **lowest priority**.
3. The title of the graph is self-explanatory; Indicates what the graph is representing. For all graphs, explanations are also given.
4. *Our Processes will always start from **number 3*** (P3) because 1 and 2 are occupied by userinit and sh, which we do not want to show (they exist by default for all test cases, thus showing them would be redundant and they are not of our interest).
5. Each point in the graph has a label which is of format **$P'N'_{suffix}$(T)**. Here 'N' is the process's PID and suffix are of two types, ***e*** & ***s,*** *'e'* represents *end* and *'s'* represents *'start'*. 'T' represents the time in tics, the event happened. For e.g., **$P3_s$ (4)** means, process 3 started (entered the CPU) at tic 4 and **$P3_e$ (19)** means, process 3 ended (relinquished/removed from CPU) at tic 19.
6. For Better viewing, please zoom into the image in any PDF viewer. These Images are of high quality (1080p—1920x1080).
7. Also **note** that start and end labels of processes are **not** in the **horizontally aligned**; The start is closer to the point where as end is a somewhat at a distance above the point. This arrangement has been done to show two events at the same point in time. i.e. start of one process and end of another at the same time. For e.g. in Figure 2, process p3 ends at 15 and at the same time process p4 starts. This way of displaying things will make sure that texts of two events are not overlapping (thus giving a clean view of the timeline).

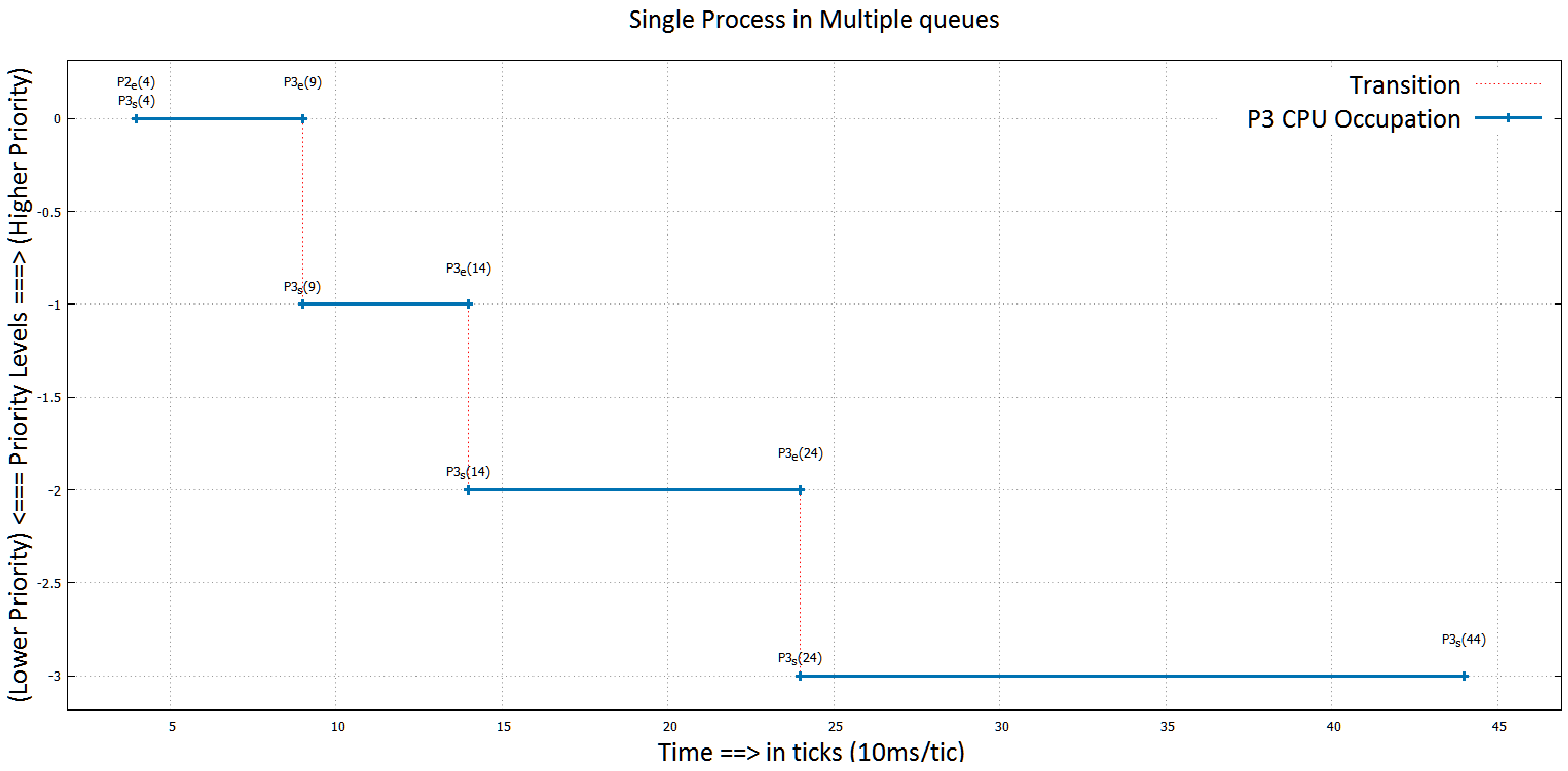

Figure 1

The process p3 is a CPU intensive process. Thus, it stays in level 0 for some 5 tics of CPU and at 9, the process is demoted to Q1 where it again runs for 5 tics. Thus it keeps getting demoted till Q3 where it completes its work and terminates. This is a **perfect example** of how a process being **CPU intensive**, is demoted to lower Queues.

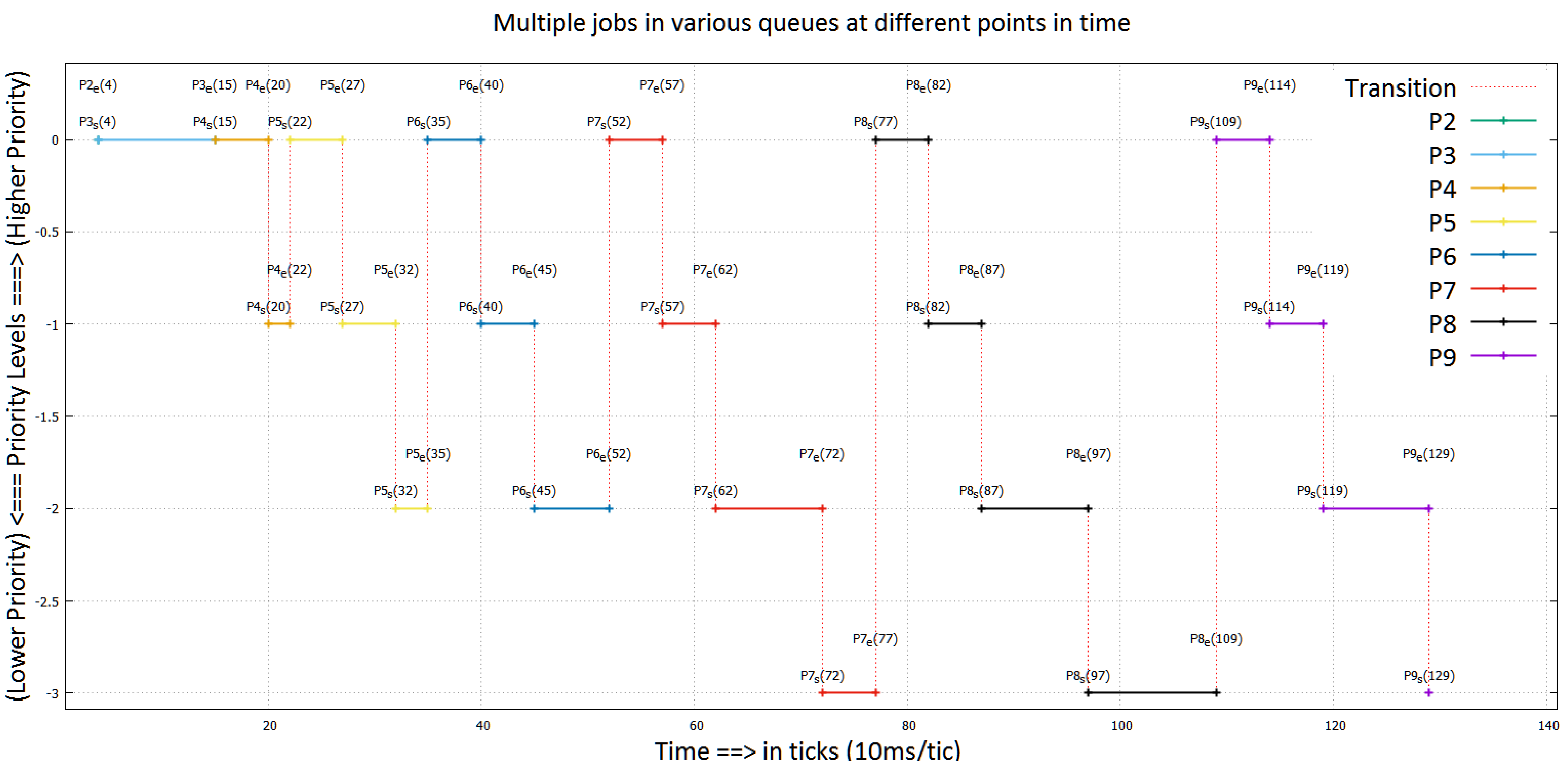

Figure 2

This diagram clearly explains the working of MLFQ. Process P7, P8 and P9 are all long running jobs, thus have been pushed from Highest queue to the lowest queues. The time slice of each queues can also be clearly noted. Processes in Q0 and Q1 run for maximum of 5 tics, Q2 and Q3 are 10 and 20 respectively. As the higher priorities kept coming, none of the process in Q3 were able to run for full 10 tics. This shows that higher jobs are being honored by preempting lower priority jobs. Also note that there was no boosting happened, because none of the processes starved.

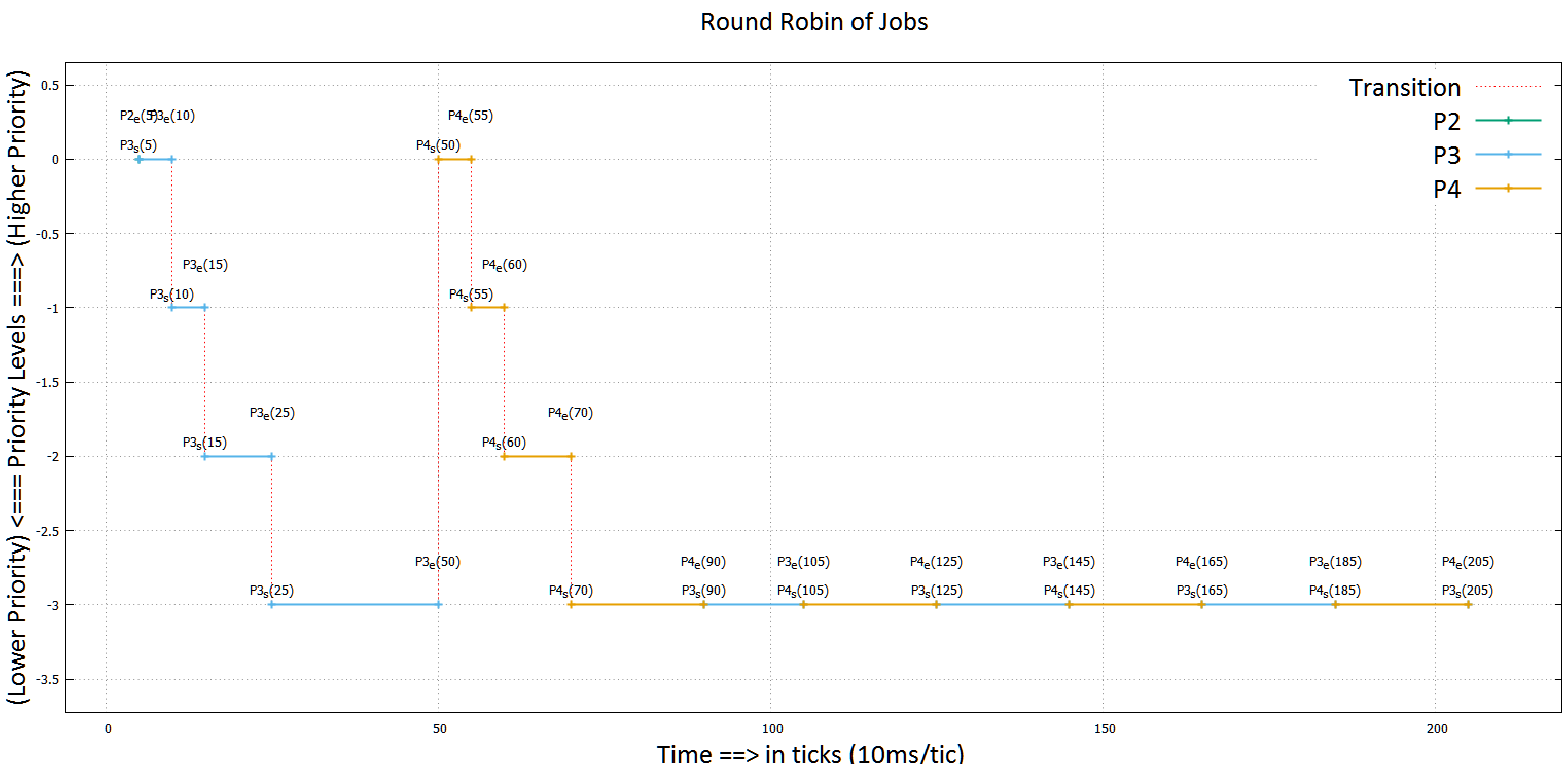

Figure 3

This graph clearly shows that two long running processes, eventually are moved to Q3 and there, they are scheduled in round-robin fashion.

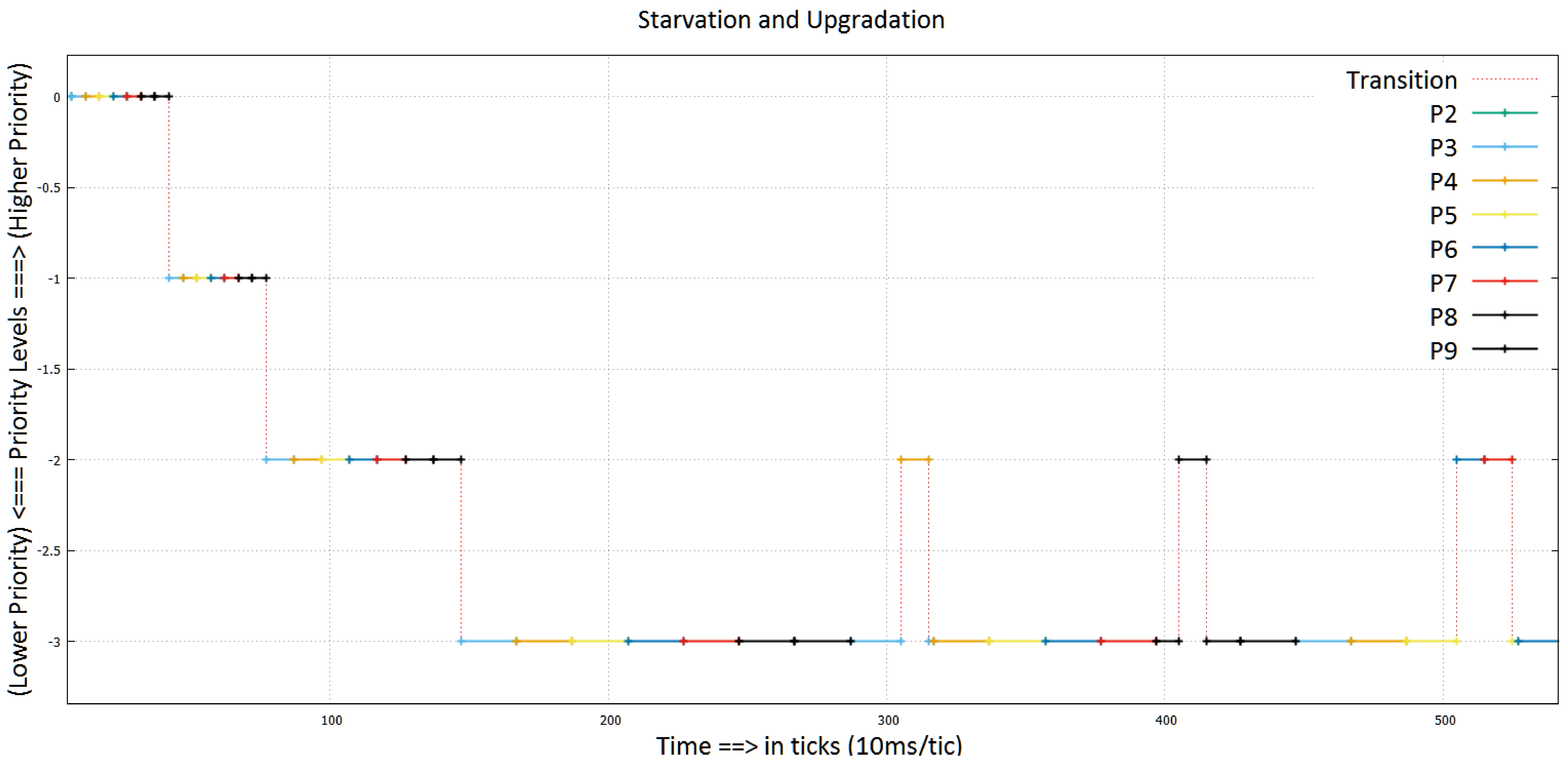

Figure 4 (a)

This graph is an example of how starvations are handled within the system. Note that spikes coming from level 3 to level 2 are of the processes whose priorities were boosted. In this graph p4, p9, p6 & p7 were the processes who were given boost. Note that p4 was boosted at ~305 because the last time it ran was at ~180 (i.e., it has been more than 100 tics since it ran, so it was given boost). Same applies for other processes as well. The part b of this figure will show things with timing information of each point in the graph which will be easier to evaluate things.

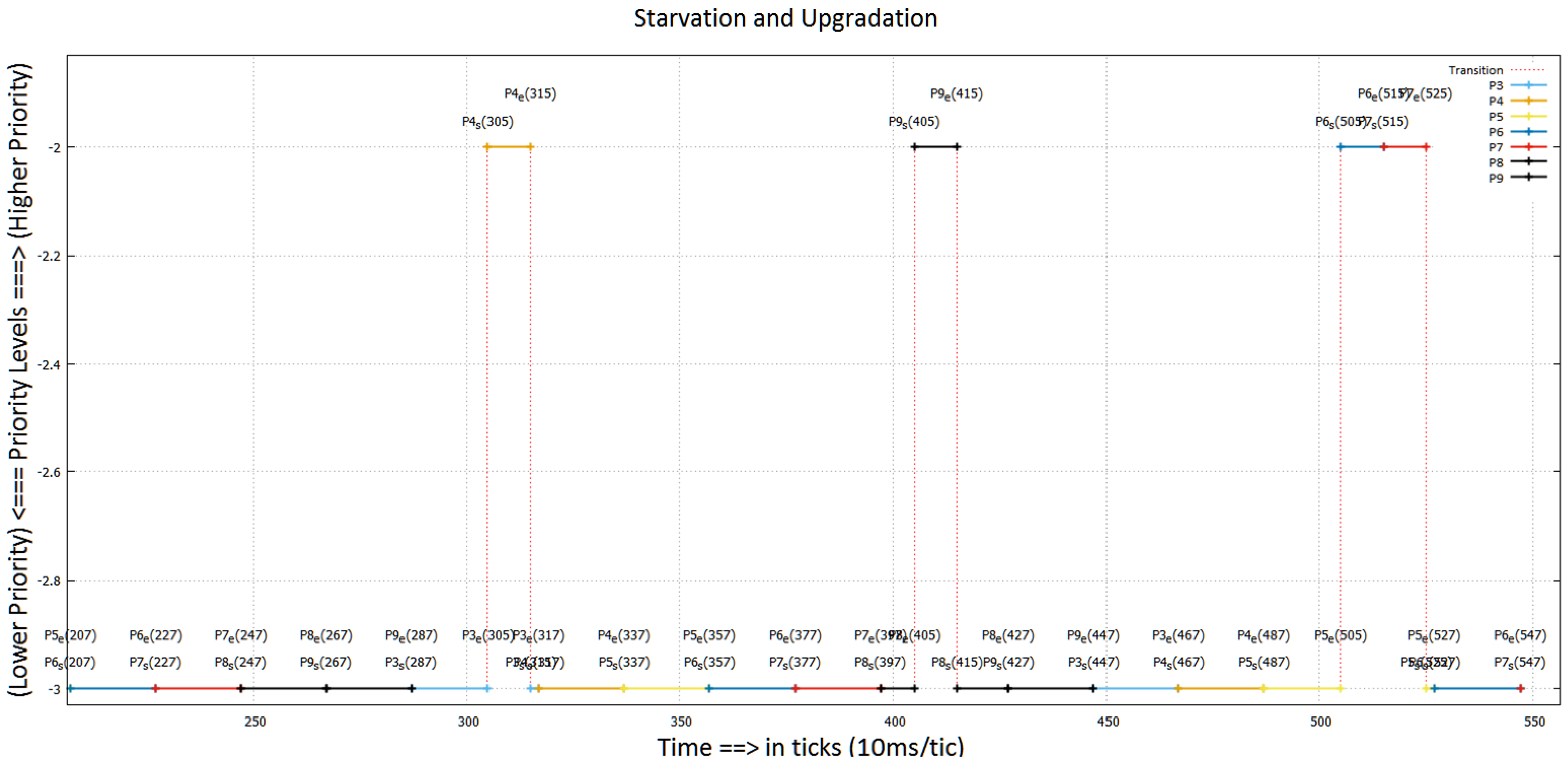

Figure 4 (b)

In this graph, all the timing values are given to understand and evaluate the MLFQ priority boosting better. The boosting decision is made for every 100 tics and this can be noted in the graph; The points 305, 405, 505 is where boosting is decided and any process which was not run in 100 tics would be given higher priority. Note that P9 was moved to higher level 2 queue form 3, because the last time it was run was at 287. So at 405 p9 was not run for more than 100 tics, thus will be boosted. The same applies for p6 & p7 (note two processes were boosted here not just one). At 505, when boosting decision is being made, both p6 & p7 were not executed for last 100 tics (p6's last tic was at 377 and p7's was at 397), thus they both are boosted.