Real Time Operating Systems Design and Programming

Homework

OS Scheduling

Contents

1 Overview

You are expected to be able to work through the following questions once you have completed the lecture. Please go over the lecture slides and any other relevant references to ensure you grasp the basic ideas of scheduling and some of the introduced scheduling algorithms. These questions are designed as examples to help you better understand the concepts in more detail.

2 Definitions

The usage of some notation or terms may vary slightly across the computer science community; here we use the following definitions for this homework.

- $T_{Arrival}$(i) = Time when process **i** request for service arrives
- $T_{Response}$(i)= Delay between request for service and start of process **i**
- $T_{Turnaround}$(i) = Delay between request for service and completion of service for process **i**
- $T_{Execution}$(i) = Time needed to perform computation for the process **i**
- $T_{Overhead}$(i) = Time needed to dispatch or stop the process task by the scheduler, usually ignored

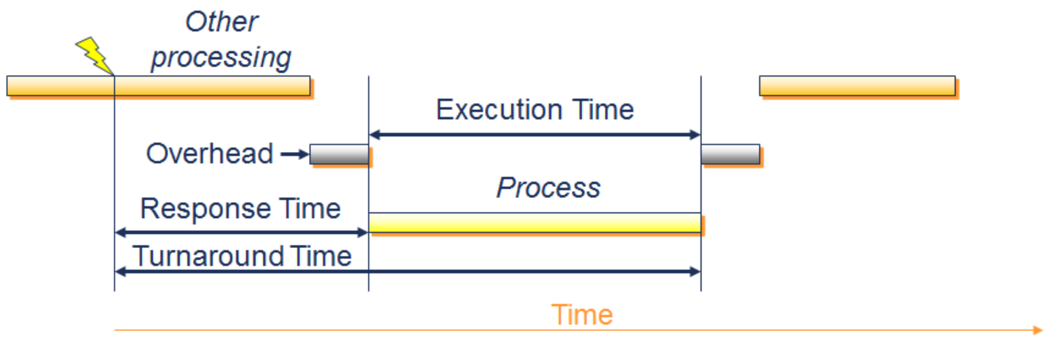


Figure 1 Definitions Related to Time

- Average waiting time = Sum of all non-execution time divided by the number of processes
- Throughput = Number of processes completed per unit of time
- Processor utilisation = Percentage of time that the processor is not idle

3 Questions

1. Recall that the seven states model below is an extension to the five states model:

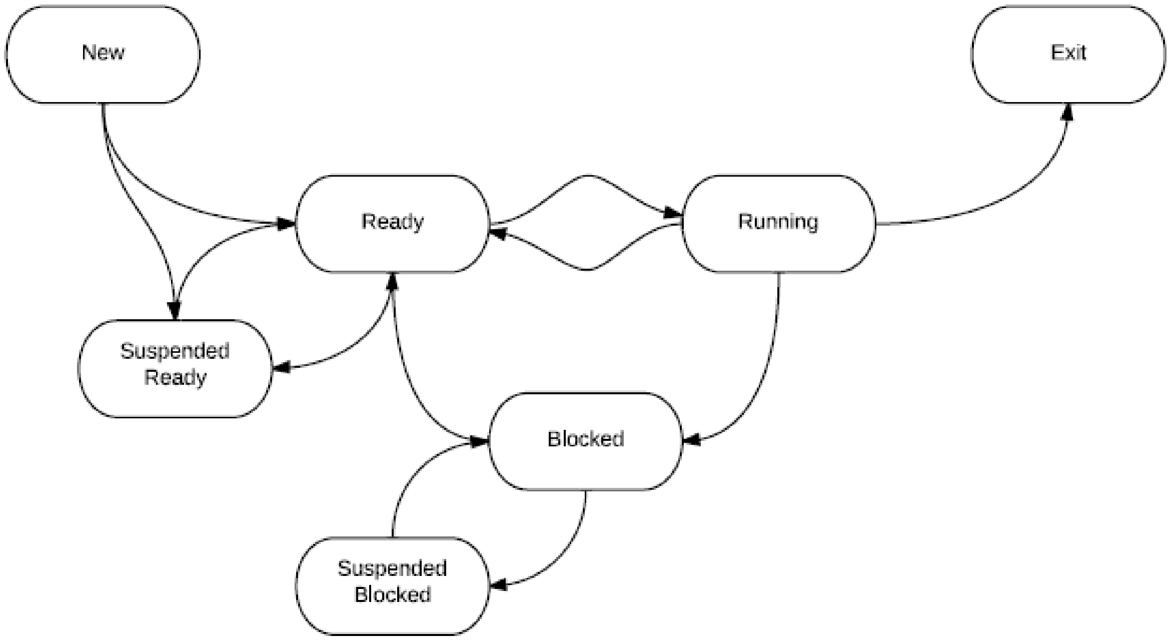


Figure 2 The Seven States Model

Further, recall that there are three levels of scheduling. List the level of scheduling of every applicable transition. For example, from New to Suspended Ready is the task of long-term Scheduling.

2. How to we evaluate and compare the performance of the scheduling algorithms/policies? You may choose a utility or (expected utility), usually a quantified evaluation of the system (such as the average waiting time) as the major target for your study.

   Now, work out the expression for Average Waiting Time given the $T_{Turnaround}(i)$, $T_{Execution}(i)$ of all processes and the number of processes.

3. Assuming 0 overhead for now and considering the general scheduling algorithms: FCFS (First Come First Served), SPN (Shortest Process Next), RR (Round Robin) with quantum = 3, SRT (Shortest Remaining Time First); for the following process set:

Process ID	A	B	C	D	E

$T_{Arrival}$	0	2	3	4	10
$T_{Execution}$	4	7	3	1	5

Draw a diagram that explicitly illustrates the process that is running at each time step. Work out the average waiting time (AWT) for each scheduling algorithm, and determine the best and worst algorithms for this particular case. You can assume the oldest process has the highest priority if necessary.

4. Notice that two algorithms perform exactly the same scheduling in question 3. Consider the following case for these two algorithms only, and repeat the analysis you have done for question 3 (still using the zero overhead assumption).

Process ID	A	B	C	D	E
$T_{Arrival}$	0	2	3	6	10
$T_{Execution}$	4	1	7	3	5

5. Now you should consider overheads for all context switching. Assume an overhead of one time step, which is an exaggeration of reality of course (but it serves our purposes). Repeat question 3 for RR only (FCFS is done for you below); also work out the AWT for both RR and FCFS. How does the quantum of RR affect its AWT?

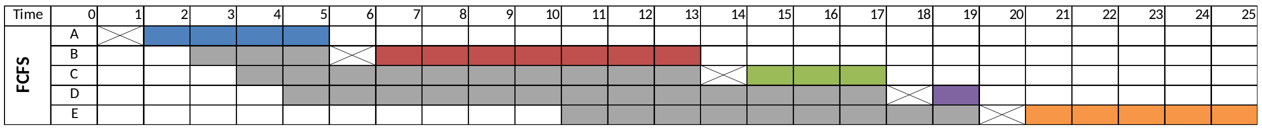


Figure 3 Example FCFS

6. To stress the problem of fairness, we define a new utility **U -** the sum of all squared consecutive waiting times:
For example, if a process starts at 0, is blocked at 2, resumes at 4 and finishes at 5 then the U for the process is $(4-2)^2$ =5. Explain why smaller **U** means better fairness; compare the FCFS and RR in question 1, which is a better algorithm in terms of having smaller **U**? Make sure to show the details of your work.

7. Think of another kind of measure or utility that can be used to evaluate the fairness. Is it possible to define a utility that estimates both the fairness and the throughput at the same time?

8. Now, repeat question 3 for the feedback algorithm. Use the following configuration:

    a. Two RR queues with quantum = 2
    b. New processes join the queue with highest priority
    c. Each time a process is pre-empted, it will be downgraded and join the end of the next lower priority queue (except the lowest queue)
    d. Scheduler always dispatches the first process in the highest queue that is not empty

    Also work out the AWT and U. Comment on this policy: is it a good, poor, or moderately balanced approach? How would you change the configuration to improve the performance?

9. (Optional) Read up the scheduling policy comparison table in your favourite OS textbooks.
10. (Optional) Case study on how your desktop OS schedules your processes.

4 Answers

1. See the following diagram:

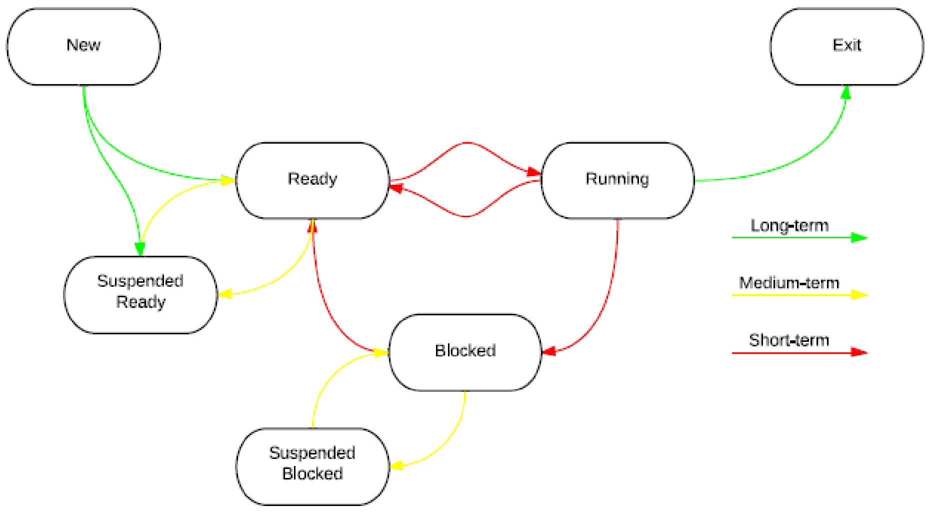


Figure 4 the Seven States Model

2. $$Average\ waiting\ time = \frac{\sum_i \left(T_{Turnaround}(i)\text{¿} - T_{Execution}(i)\right)}{Number\ of\ processes}\text{¿}$$

3. See the chart below for the scheduling:

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FCFS	A	🟦	🟦	🟦	🟦																
	B			▨	▨	🟥	🟥	🟥	🟥	🟥	🟥	🟥									
	C				▨	▨	▨	▨	▨	▨	▨	▨	🟩	🟩	🟩						
	D					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	🟪					
	E											▨	▨	▨	▨	▨	🟧	🟧	🟧	🟧	🟧
SPN	A	🟦	🟦	🟦	🟦																
	B			▨	▨	▨	▨	▨	▨	🟥	🟥	🟥	🟥	🟥	🟥	🟥					
	C				▨	▨	🟩	🟩	🟩												
	D					🟪															
	E											▨	▨	▨	▨	▨	🟧	🟧	🟧	🟧	🟧
RR	A	🟦	🟦	🟦	▨	▨	▨	🟦													
	B			▨	🟥	🟥	🟥	▨	▨	▨	▨	▨	🟥	🟥	🟥	▨	▨	▨	🟥		
	C				▨	▨	▨	▨	🟩	🟩	🟩										
	D					▨	▨	▨	▨	▨	▨	🟪									
	E											▨	▨	▨	▨	🟧	🟧	🟧	▨	🟧	🟧
SRT	A	🟦	🟦	🟦	🟦																
	B			▨	▨	▨	▨	▨	▨	🟥	🟥	🟥	🟥	🟥	🟥	🟥					
	C				▨	▨	🟩	🟩	🟩												
	D					🟪															

AWT: Count the grey boxes (waiting time) and divide the number of grey boxes by 5
FCFS: (0+2+8+10+5)/5=5
SPN: (0+6+2+0+5)/5=2.6 (Best)
RR: (3+9+4+6+5)/5=5.4 (Worst)
SRT: (0+6+2+0+5)/5=2.6 (Best)

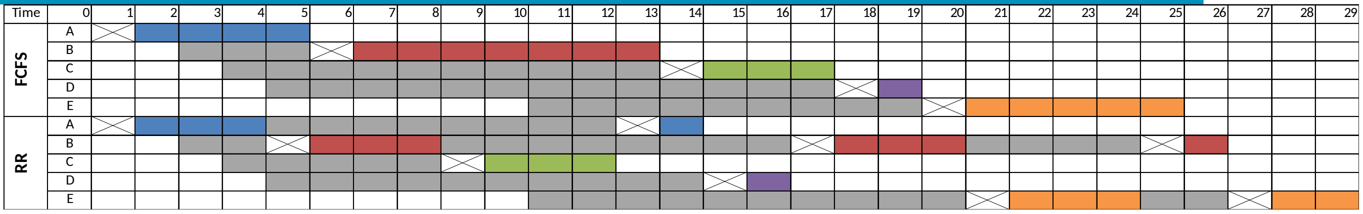
4. See the chart below for the scheduling:

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SPN	A	🟦	🟦	🟦	🟦																
	B			▨	▨	🟥															
	C				▨	▨	🟩	🟩	🟩	🟩	🟩	🟩	🟩								
	D							▨	▨	▨	▨	▨	▨	🟪	🟪	🟪					
	E											▨	▨	▨	▨	▨	🟧	🟧	🟧	🟧	🟧
SRT	A	🟦	🟦	▨	🟦	🟦															
	B			🟥																	
	C				▨	▨	🟩	▨	▨	▨	🟩	🟩	🟩	🟩	🟩	🟩					
	D							🟪	🟪	🟪											
	E											▨	▨	▨	▨	▨	🟧	🟧	🟧	🟧	🟧

AWT:
SPN:(0+2+2+6+5)/5=3
SRT:(1+0+5+0+5)/5=2.2(Better)

So that in some cases, SRT can react faster by switching to the shortest task through preemption.

5. See the chart below for the scheduling:



AWT: Also count the crossed boxes for overheads

FCFS:(1+4+11+14+10)/5=8

RR:(10+17+6+11+14)/5=11.6

Notice that the overhead not only prolongs the AWT, but may also change the way how processes are scheduled (but not in this case). Short quantum result in higher AWT but long quantum will be effectively the same as FCFS.

6. U can be considered as a penalty and it is the sum of the square of all consecutive waiting times, which means processes with longer waiting time suffer.

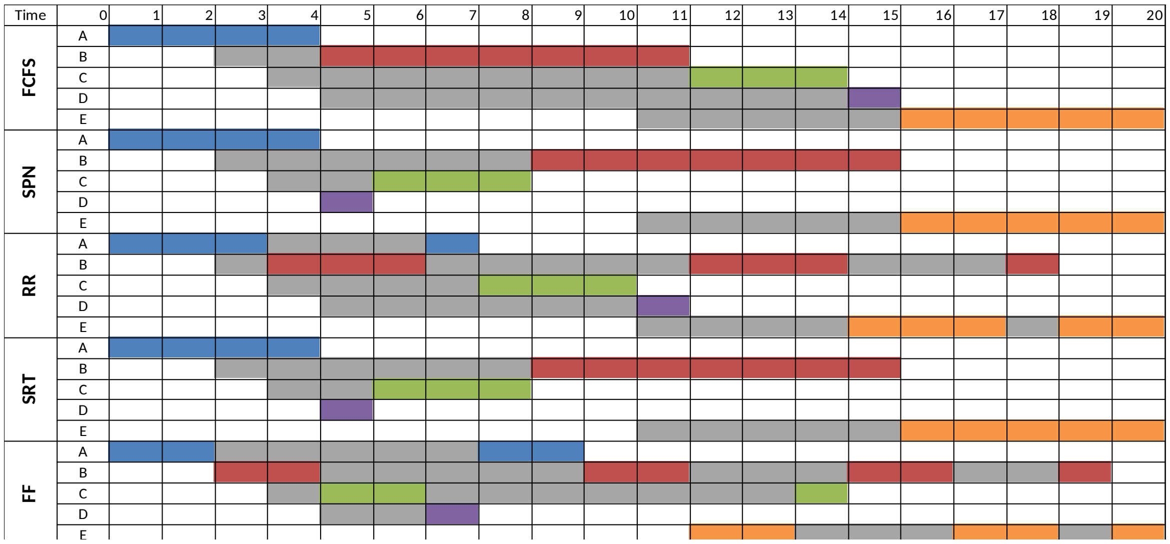
   U(FCFS):$0^2+2^2+8^2+10^2+5^2=193$

   U(RR):$3^2+1^2+5^2+3^2+4^2+6^2+4^2+1^2=113$

   Round robin is better in fairness, at the cost of having a higher AWT.

7. Anything similar to the utility (U) in question 6 that penalises longer consecutive waiting times addresses fairness. To address AWT, a possible measure could be an expression such as: a*AWT+b*U whereby a and b are constants that reflect the importance of each criteria. This will turn the problem into a mathematical optimisation one.

8. See the chart below for the scheduling:



AWT:(5+10+8+2+4)/5=5.8
U:$5^2+5^2+3^2+2^2+1^2+7^2+2^2+3^2+1^{2=}127$

This seems like a poor policy in this specific case. Increasing the quantum time may help. Other approaches are also acceptable, such as adjusting the policies in each queue.