



Schedulers: Real Time Systems

Richi Dubey • 24.01.2019



Overview

1. Schedulers

- Definition and validity
- Performance measures

2. Approach to schedule RTS

- Driven by clock
- Weighted round robin approach
- Based on priority



Scheduler

- Responsible for implementing the algorithm that allocates resources for a job and schedules it.

Allocates job to a processor




Or can say allocates processor to a job



Valid Schedules

Assumption: Jobs are not run in parallel on more than one processor to speed up their execution

- ❖ Every processor is assigned at most one job
- ❖ Consequently every job is assigned to at most one processor
- ❖ No job is scheduled before its release time



Scheduling a job means assignment of job on an available processor

Performance Measures



For Hard RTS: Feasibility

Feasible Schedule : Every job is completed by its deadline

An algorithm is said to be optimal if it always produces a feasible schedule for a group of jobs.

Conversely, if an optimal algorithm fails to find a feasible schedule, it can be concluded that the jobs can not feasibly be scheduled by any algorithm.

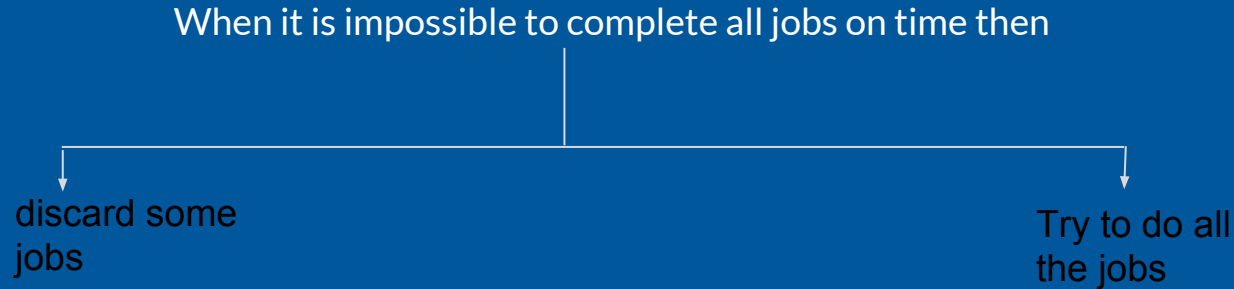
Average Lateness

In transmission of packets, packets are buffered at receiving end, then assembled together into a message. Early arrivals have to wait for late arrivals and are stored in buffer till then.

So average lateness is the best measure here.

For Soft RTS: Miss rate/Loss Rate

For soft real time systems, average response time is considered.(Lesser the response time, better is the algorithm)



Miss rate: % of jobs that are executed but completed too late

Loss rate: % of jobs that are discarded.

Contd ...

For some soft real time applications,
Sometimes it is better to

Complete
some jobs
late

Miss Rate 

Loss Rate 

Discard
some jobs

Miss Rate 

Loss Rate 

Invalid Rate

Measures sum of miss rate and loss rate and gives % of all jobs that do not give useful result

Therefore, Invalid rate should always be minimized.



Approaches



Clock Driven





Definition

All parameters of a hard real-time job are fixed and known

Decisions on what job to execute at what times are made at specific time instants which are chosen a priori before the execution begins.

The entire schedule is computed offline and stored for use at run time.

Scheduler schedules the jobs at each scheduling decision time.



All parameters of a hard real-time job are fixed and known

Decisions on what job to execute at what times are made at specific time instants which are chosen a priori before the execution begins.

The entire schedule is computed offline and stored for use at run time.

Scheduler schedules the jobs at each scheduling decision time.



Choosing scheduling decision time

Most used approach: Placing them at regularly spaced time intervals

Done by using hardware timer

This hardware timer expires periodically without intervention of scheduler. So scheduler overhead is removed.



So when system is initialized, scheduler selects and schedules the job(s) that will execute until the next scheduling decision time and then blocks itself waiting for the expiration of timer.

When timer expires, scheduler wakes up and repeats the process.



Priority Driven

Definition

Refers to class of algorithms that never leave a resource idle if any job is ready for execution. Aka greedy algorithms, as they make locally optimal choices.

So when a processor or resources is available and some job is there which can make use of it, this algorithm never makes the job wait.

How Does It Work?

Jobs which are ready for execution are put in a queue, this queue is sorted by priorities, and at any scheduling decision time, job with highest priority is scheduled and executed on available processor.

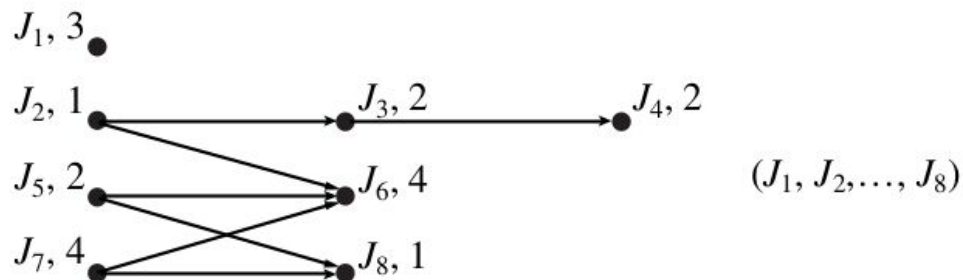
Upcoming Example:-

Jobs J1,J2..J8

Priority : J1>J2>J3>J4...

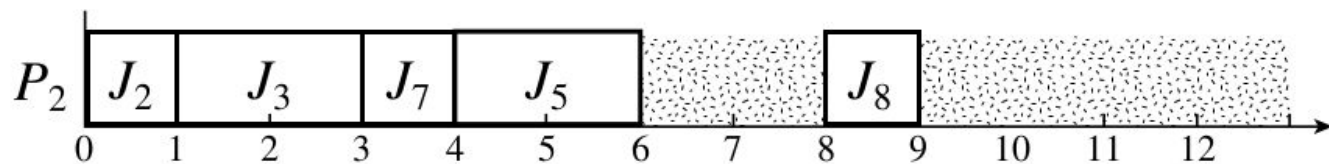
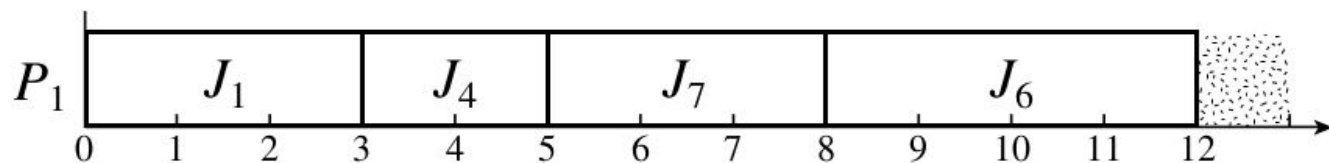
Release time : All except J5 at t=0,

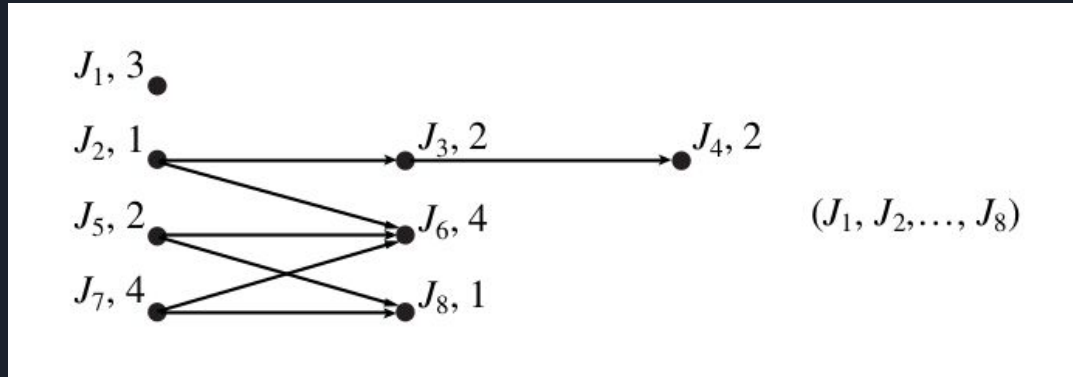
J5 at t=4;



Release time : All except
J5 at $t=0$,

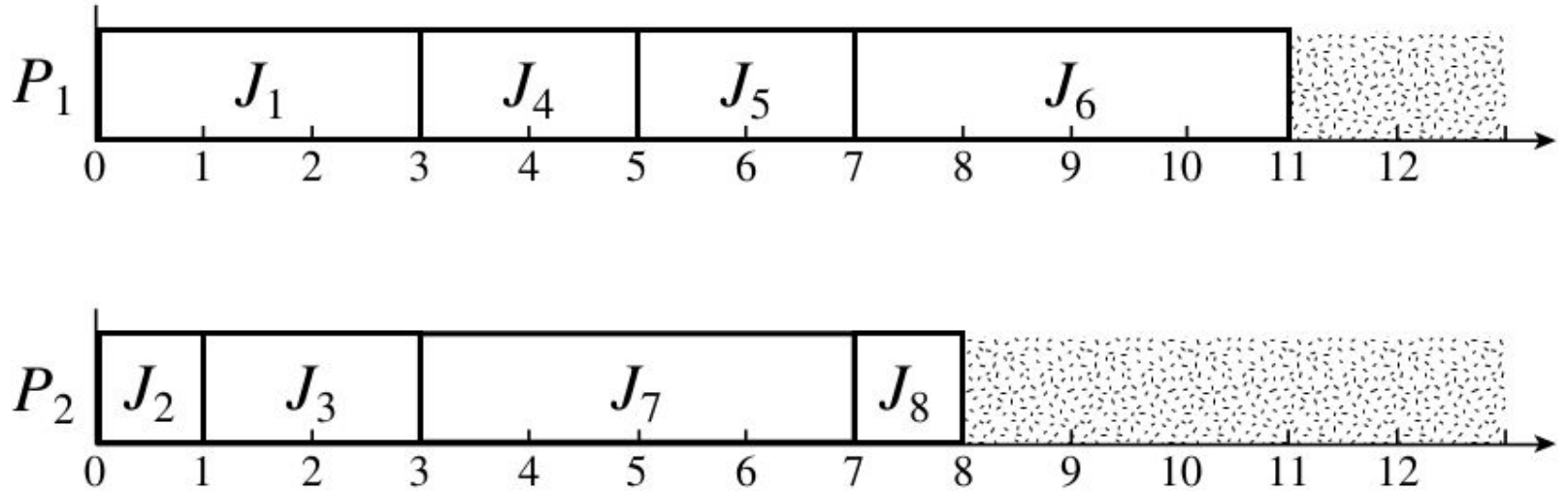
J5 at $t=4$;





Release time : All except
J5 at $t=0$,

J5 at $t=4$;



Which One Is Better?

Here non-preemptive approach was better but generally this is not the case. There is no way to better which approach would be better beforehand.

As a special case, if all jobs are released at the same time, preemptive approach would certainly be better if cost of preemption is ignored.

Minimum makespan of an optimal non preemptive algorithm is always more than preemptive algorithm. (Doubt: this case?)

For 2 processor CPUs, Coffman and Garey proved that min. makespan of non-preemptive algorithm would be at most $4/3$ times that of preemptive algorithm.



Weighted Round Robin

Round Robin

Every job joins the FIFO queue and each job executes for one time slice starting from the head of the queue.

If a job isn't completed within its 1 time slice, it is preempted and put back to the end of queue.

So, if there are n jobs ready for execution, each job gets $(1/n)$ th share of the processor.

Weighted Round Robin

Rather than giving all ready job equal share of processor, different jobs may be given different weights.

Different jobs may be given different time slices and using this we can control the progress of any job.

By giving each job a fraction of processor, this algorithm delays the completion time of every job.

Doubts after this in TBook page 75.

Lorem ipsum dolor sit
amet, consectetur
adipiscing elit

Lorem ipsum dolor sit
amet, consectetur
adipiscing elit

Lorem ipsum dolor sit
amet, consectetur
adipiscing elit



Lorem ipsum dolor sit
amet, consectetur
adipiscing elit

Lorem ipsum dolor sit
amet, consectetur
adipiscing elit



Next steps

Lorem ipsum dolor sit amet
Consectetur adipiscing elit, sed do eiusmod tempor

Lorem ipsum dolor sit amet

Consectetur adipiscing elit, sed do eiusmod tempor



Goals for next meeting

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit
2. Sed do eiusmod tempor incididunt ut labore
3. Ut enim ad minim veniam, quis nostrud exercitation



The End