Institut Polytechnique des Sciences Avancées

FINAL ASSIGNMENT : Schedule search

JACQUART Rémi

Année universitaire 2024-2025

Table des matières

Ojective

This report demonstrates the schedulability of a periodic task set using the **Earliest Deadline First (EDF)** scheduling algorithm in its **non-preemptive** form. We include a response time analysis and verify that all deadlines are met. I use python in order to programm the EDF algorithm.

1st task : Schedulability and Hyperperiod

Firstly we present the task set for this project, we have :

Task	**Computation Time (C)**	**Period = Deadline (T_i)**
τ_1	2	10
τ_2	3	10
τ_3	2	20
τ_4	2	20
τ_5	2	40
τ_6	2	40
τ_7	3	80

We have to verify if the task set is schedulable or not, so we calculate :

$$U = \Sigma_{i=1}^{7} \frac{C_i}{T_i} = \frac{2}{10} + \frac{3}{10} + \frac{2}{20} + \frac{2}{20} + \frac{2}{40} + \frac{2}{40} + \frac{3}{80} = 0.8375$$

The task set is schedulable because $U < 1$
For the hyperperiod we have the LCM (lowest commun multiple) of the different period :

$$LCM(10, 20, 40, 80) = 80$$

So the hyperperiod is 80.

2nd task : Assumptions and Methodology

— **Non-preemptive EDF** : once a task starts executing, it runs to completion.
— **Implicit deadlines** : deadline equals the period of each task.
— Tasks are released periodically depending of their period.

3rd task : Complexity Analysis

At each simulation step (each time unit), the algorithm performs the following operations :

— Check for new task releases : $\mathcal{O}(n)$
— Insert released jobs into a min-heap by their deadlines : $\mathcal{O}(\log K)$ where K is the number of jobs currently in the queue.
— Select the task with earliest deadline (heap root) : $\mathcal{O}(1)$
— Execute task for C time units (non-preemptively) : constant time iteration.

In the worst case, this is repeated every unit of time for the hyperperiod :

$$\text{Worst-case complexity} = \mathcal{O}(H \cdot (\log N + n))$$

Where :
— H is the hyperperiod
— n is the number of tasks (here there are 7 tasks)
— N is the total number of jobs (during a hyperperiod)

4th task : Response Time Analysis

Task	Period (T)	Response times for all jobs	All met deadlines ?
τ_1	10	[2, 5, 3, 5, 4, 4, 4, 3]	Yes
τ_2	10	[5, 8, 5, 8, 7, 7, 7, 6]	Yes
τ_3	20	[8, 18, 11, 15]	Yes
τ_4	20	[10, 11, 15, 13]	Yes
τ_5	40	[12, 11]	Yes
τ_6	40	[13, 15]	Yes
τ_7	80	[21]	Yes

Each task instance finishes execution before its deadline. Therefore, the task set is schedulable using non-preemptive EDF.
With the code in Python we can see the schedule of the task set below :

```
Schedule   [(0, 'τ1'), (1, 'τ1'), (2, 'τ2'), (3, 'τ2'), (4, 'τ2'), (5, 'τ3'), (6, 'τ3'), (7, 'τ4'), (8,
'τ4'), (9, 'τ6'), (10, 'τ6'), (11, 'τ1'), (12, 'τ1'), (13, 'τ2'), (14, 'τ2'), (15, 'τ2'), (16, 'τ5'),
(17, 'τ5'), (18, 'τ7'), (19, 'τ7'), (20, 'τ7'), (21, 'τ1'), (22, 'τ1'), (23, 'τ2'), (24, 'τ2'), (25,
'τ2'), (26, 'τ4'), (27, 'τ4'), (28, 'τ3'), (29, 'τ3'), (30, 'τ1'), (31, 'τ1'), (32, 'τ2'), (33, 'τ2'),
(34, 'τ2'), (35, 'Idle'), (36, 'Idle'), (37, 'Idle'), (38, 'Idle'), (39, 'Idle'), (40, 'τ1'), (41,
'τ1'), (42, 'τ2'), (43, 'τ2'), (44, 'τ2'), (45, 'τ3'), (46, 'τ3'), (47, 'τ4'), (48, 'τ4'), (49, 'τ6'),
(50, 'τ6'), (51, 'τ1'), (52, 'τ1'), (53, 'τ2'), (54, 'τ2'), (55, 'τ2'), (56, 'τ5'), (57, 'τ5'), (58,
'Idle'), (59, 'Idle'), (60, 'τ1'), (61, 'τ1'), (62, 'τ2'), (63, 'τ2'), (64, 'τ2'), (65, 'τ4'), (66,
'τ4'), (67, 'τ3'), (68, 'τ3'), (69, 'Idle'), (70, 'τ1'), (71, 'τ1'), (72, 'τ2'), (73, 'τ2'), (74, 'τ2')
(75, 'Idle'), (76, 'Idle'), (77, 'Idle'), (78, 'Idle'), (79, 'Idle')]
```

With this algorithm we can conclude a lot of things like the idle time, the waiting time, the task starting time and the response time. Below we have the response time :

```
Response Times
 {'τ1': [2, 3, 3, 2, 2, 3, 2, 2], 'τ2': [5, 6, 6, 5, 5, 6, 5, 5], 'τ3': [7, 10, 7, 9], 'τ4': [9, 8, 9,
7], 'τ6': [11, 11], 'τ5': [18, 18], 'τ7': [21]}
```

With this figure we can calculate manually the waiting time, we have :

$$\tau_1 = 2 + 3 + 3 + 2 + 2 + 3 + 2 + 2 - 16 = 3$$

$$\tau_2 = 5 + 6 + 6 + 5 + 5 + 6 + 5 + 5 - 24 = 43 - 24 = 19$$

$$\tau_3 = 7 + 10 + 7 + 9 - 8 = 33 - 8 = 25$$

$$\tau_4 = 9 + 8 + 9 + 7 - 8 = 33 - 8 = 25$$

$$\tau_5 = 18 + 18 - 4 = 32$$

$$\tau_6 = 11 + 11 - 4 = 18$$

$$\tau_7 = 21 - 3 = 18$$

So if we sum all the τ we see that the waiting time is 140 and in Python we have :

```
Idle Time
13
Waiting Time
140
Tasks starting time:
{'τ1': [0, 11, 21, 30, 40, 51, 60, 70], 'τ2': [2, 13, 23, 32, 42, 53, 62, 72], 'τ3': [5, 28, 45, 67],
'τ4': [7, 26, 47, 65], 'τ6': [9, 49], 'τ5': [16, 56], 'τ7': [18]}
```

We have also the idle time (equal to 13) when the CPU is not tackling a task

5th task : Conclusion

This analysis confirms that the given periodic task set is schedulable under non-preemptive EDF scheduling. All task jobs meet their deadlines when scheduled over the hyperperiod. The algorithm is efficient for systems where task preemption is not permitted due to safety or implementation constraints.

Important : This result holds as long as processor utilization remains under 100% and task execution times do not vary.