EDF APIs

The user Application Program interface (API) to use the Earliest Deadline First (EDF) scheduler implmented in user space consists of invoking the C functions in the folllowing order:

- $EdfScheduleInit(PeriodicTaskArray, AperiodicTaskArray)$
- $EdfScheduleStart()$
- $EdfDeleteAllTasks()$

PeriodicTaskArray and AperiodicTaskArray are arrays of the public Task Control Block (TCB) structure $pubTCB\_t$, which along with the above function prototypes are defined in $edf\_scheduler.h$

Each of the struct elements coresopnding to the following parameters must initilized by the user in PeriodicTaskArray :

- Period (P_i)
- Worst case execution time (c_i)
- Phase (ϕ_i)
- Relative deadline (D_i)
- Task number (i)
- Task function (τ_i)

and in AperiodicTaskArray :

- Relative arrival time (a_i)
- Worst case execution time (c_i)
- Task number (i)
- Task function (J_i)
- Aperiodicity flag

EDF Option 1 using trace APIs and the inbuilt priority based scheduler of $FreeRTOS$ is implemented here based on [1]. Periodic tasks priorities are updated just prior to task switching via the $traceTASK\_SWITCHED\_OUT()$ macro [2]. Here, the TCBs stored in the doubly linked lists are sorted implictly by using the APIs from $list.h$

Aperiodic tasks are also scheduled by EDF by using a Total Bandwidth Server (TBS) [3], which assigns fictitios deadlines to them based on the server Utilitzation factor U_s, and arrival times a_i. The aperiodic tasks are deleted after execution of it's first instace.

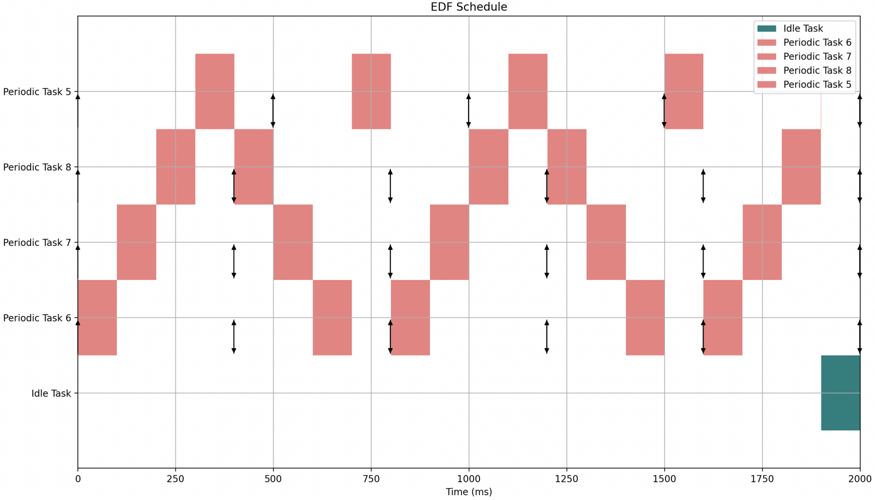
Task deadline overflows are monitored every defined tick via the $traceTASK\_INCREMENT\_TICK()$ macro and marked for deletion in the privately maintained TCB $exTCB\_t$. These are later deleted during the following priority update stage mentioned above.

EDF results: 4 periodic tasks, 95% utilization factor

	τ_5	τ_6	τ_7	τ_8
ϕ_i	0	0	0	0
a_i	0	0	0	0
c_i	100	100	100	100
P_i	500	400	400	400
D_i	500	400	400	400

Utilization factor

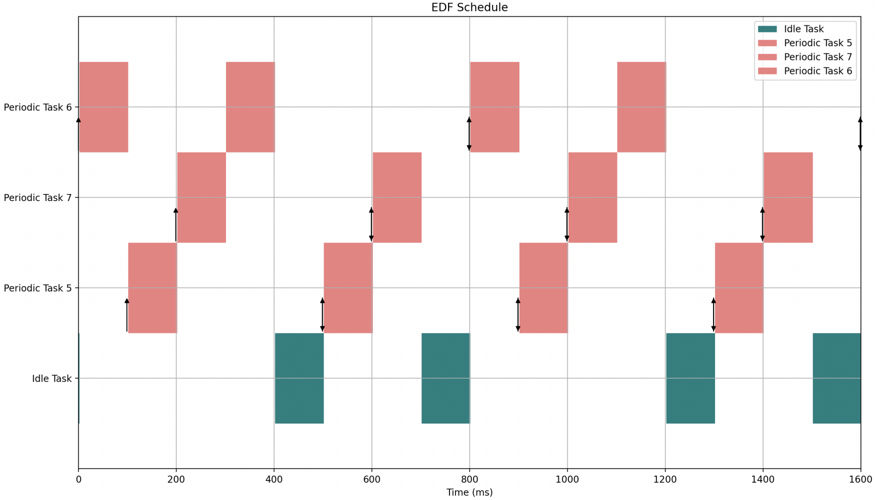
$$\begin{aligned} U_p &= \sum \frac{C_i}{P_i} \\ &= \frac{100}{500} + \frac{100}{400} + \frac{100}{400} + \frac{100}{400} = 0.95 \end{aligned}$$


EDF Schedule
Idle Task
Periodic Task 6
Periodic Task 7
Periodic Task 8
Periodic Task 5
Periodic Task 5
Periodic Task 8
Periodic Task 7
Periodic Task 6
Idle Task
0
250
500
750
1000
1250
1500
1750
2000
Time (ms)

	τ_5	τ_6	τ_7
ϕ_i	100	0	200
a_i	0	0	0
c_i	100	200	100
P_i	400	800	400
D_i	400	800	400

Utilization factor

$$U_p = \sum \frac{C_i}{P_i}$$
$$= \frac{100}{400} + \frac{200}{800} + \frac{100}{400} = 0.75$$

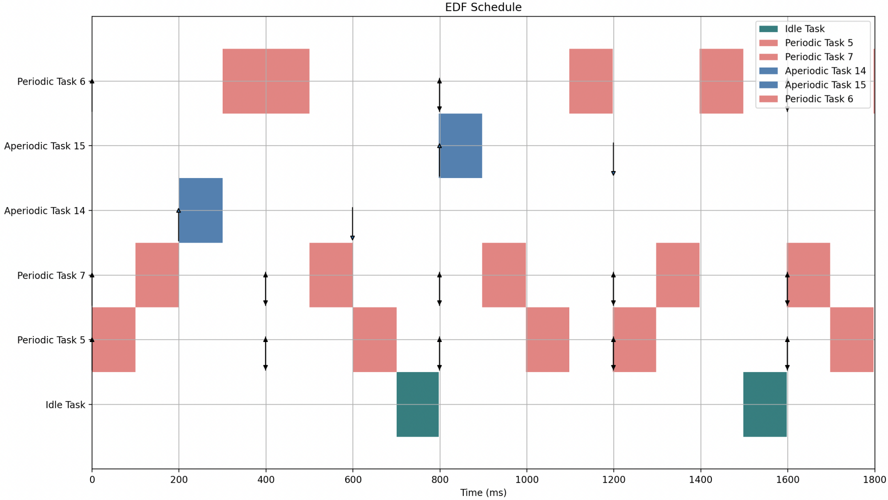

EDF Schedule
Idle Task
Periodic Task 5
Periodic Task 7
Periodic Task 6
Periodic Task 6
Periodic Task 7
Periodic Task 5
Idle Task
0
200
400
600
800
1000
1200
1400
1600
Time (ms)

EDF results: 3 periodic, 2 aperiodic tasks

	τ_5	τ_6	τ_7	J_{14}	J_{15}
ϕ_i	0	0	0		
a_i	0	0	0	200	800
c_i	100	200	100	100	100
P_i	400	800	400		
D_i	400	800	400		

Utilization factor and aperiodic deadlines for TBS

$$U_p = \frac{100}{400} + \frac{200}{800} + \frac{100}{400} = 0.75$$
$$U_s = 0.25$$
$$D_i = max(a_i + d_{i-1}) + \frac{C_i}{U_s}$$
$$\begin{aligned} D_{14} &= 200 + \frac{100}{0.25} \\ &= 600 \\ D_{15} &= 800 + \frac{100}{0.25} \\ &= 1200 \end{aligned}$$


EDF Schedule
Periodic Task 6
Aperiodic Task 15
Aperiodic Task 14
Periodic Task 7
Periodic Task 5
Idle Task
0
200
400
600
800
1000
1200
1400
1600
1800
Time (ms)
Idle Task
Periodic Task 5
Periodic Task 7
Aperiodic Task 14
Aperiodic Task 15
Periodic Task 6

EDF results: 3 periodic, 2 aperiodic tasks, with deadline overflows

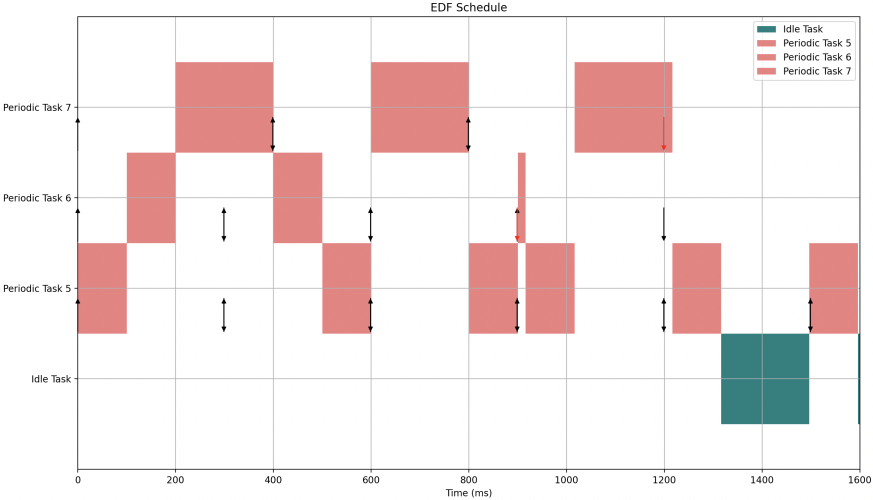
	τ_5	τ_6	τ_7	J_{14}	J_{15}
ϕ_i	0	0	0		
a_i	0	0	0	200	800
c_i	100	100	200	100	100
P_i	300	300	400		
D_i	300	300	400		

Utilization factor and aperiodic deadlines for TBS

$$U_p = \frac{100}{300} + \frac{100}{300} + \frac{200}{400} = 1.167$$

$U_s = 0$ (Aperiodic tasks not scheduled)

Expected overflow at 900ms.
Additionally detected at 1200ms, due to overhead in deleting τ_5.


EDF Schedule
Idle Task
Periodic Task 5
Periodic Task 6
Periodic Task 7
Periodic Task 7
Periodic Task 6
Periodic Task 5
Idle Task
0
200
400
600
800
1000
1200
1400
1600
Time (ms)

- [1] R. Kase, Efficient Scheduling library for FreeRTOS (Master Thesis), KTH Stockholm, 2016.
- [2] ESP-FreeRTOS APIs, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html
- [3] C. Scholl, Real Time Operating Systems and Worst Case Execution Times (Lecture Notes), University of Freiburg, 2022.