A Performance Analysis of Earliest Deadline First Algorithm against Rate Monotonic and Deadline Monotonic Algorithms

Members: Omar Nweashe {omarnweashe@vt.edu} - Yahia Tawfik {mtyahia@vt.edu}

"I have neither given nor received unauthorized assistance on this assignment."

PROJECT OBJECTIVES

This project aims to assess and evaluate the performance of the Earliest Deadline First (EDF) Algorithm against the Rate Monotonic (RM) and Deadline Monotonic (DM) Algorithms. The goal is to obtain different performance metrics, such as computation time and schedulability, to determine the effectiveness and usage of each algorithm across different task sets that would be curated to highlight and validate the hypotheses we construct.

WORK DONE

Our Rate Monotonic (**RM**) and Deadline Monotonic (**DM**) Algorithms are based on the code for our projects for the class, as we further validated their performance over different task sets to make sure we obtained correct output. As for our Earliest Deadline First (**EDF**) Algorithm, we were able to modify the existing code to make sure our **EDF** code functions correctly. First, we had to ensure that ***xAbsoluteDeadline*** is in the task control block as it will be used in determining the priority for each task. Second, since **EDF** uses dynamic scheduling, where tasks are assigned priorities while the tasks are being executed. This is unlike RM and DM, where priorities are assigned statically one time before the tasks are executed. So, we created the ***prvSetDynamicPriorities()*** function, which updates tasks priorities using **EDF** scheduling algorithms.
The function starts by iterating over all tasks to set priorities based on the closest deadline first. Thus, it looks for the task with the minimum absolute deadline and assigns it the highest priority value. The function is called in ***prvSchedulerFunction()*** inside the infinite for loop in the scheduler function. Therefore, everytime the tasks need to be re-evaluated ***prvSetDynamicPriorities()*** will be invoked by the scheduler function. To protect the function from being called when RM or DM is running, the function is called inside an if statement which checks if the Scheduling policy being used is **EDF**.

For our task sets, in our proposal we listed four different task set criteria we wanted to assess our algorithms on:

1. ***Task Set 1***: A Periodic task set where D<T (where we expect EDF should fail, RM/DM should pass)
    a.

Task Name	Period (ms)	Execution Time (ms)	Deadline (ms)

t2	100	20	80
t3	150	30	120

2. ***Task Set 2***: A Periodic task set where D=T (where we expect all algorithms should pass)
    a.

Task Name	Period (ms)	Execution Time (ms)	Deadline (ms)
t2	1000	200	1000
t3	1500	300	1500

3. ***Task Set 3:*** An Overloaded periodic task set where D=T (where we expect all algorithms should fail)
    a.

Task Name	Period (ms)	Execution Time (ms)	Deadline (ms)
t2	100	150	100
t3	120	170	120

4. ***Task Set 4***: A Periodic task set where D>T (where we expect EDF should outperform RM and DM)
    a.

Task Name	Period (ms)	Execution Time (ms)	Deadline (ms)
t1	1000	500	1200
t2	1500	700	1700
t3	2000	400	2700
t4	2500	300	3200

And the set of hypotheses we came up with to test the four task sets were the following:

1. **Hypothesis I**: In a Periodic Task Set where the deadline is greater than the Period, EDF would be the optimal Algorithm over RM and DM.
2. **Hypothesis II**: All Algorithms Would Fail in Scheduling an Overloaded Periodic Task Set, Where D=T.
3. **Hypothesis III**: All Algorithms Would Succeed in Scheduling an non-overloaded Periodic Task Set, Where D=T.

4. **Hypothesis IV**: EDF would fail in scheduling a periodic task set where D<T, where task deadline differences are narrow.

In the following **RESULTS** section, we will discuss our findings, as well as elaborate or justify what we observe, and assess whether it matches our expectations or not.

RESULTS

Test Case 1:

Analysis:

This task set tested **Hypothesis IV,** where we expected EDF to fail in scheduling a periodic task set where the deadline is less than the RM and DM algorithms. As seen in the screenshots below, EDF has more misses every task it begins, while RM and DM successfully schedule the tasks with virtually no problems, as well as assign the priorities correctly for each task they schedule. Which validates our expected output, and tells us that our hypothesis, **Hypothesis IV,** was correct, that EDF fails while RM and DM perform as expected.

EDF: Picture below shows EDF failing to schedule tasks

```
19:03:37.748 -> Schedule using EDF.
19:03:37.748 -> vTaskStartScheduler() start to work at 0 tick count!
19:03:37.813 -> t2 begin at Ticktount: 2
19:03:37.846 -> k Ticccount: 3
19:03:37.846 -> t2 missed deadline at tick: 4
19:03:37.879 -> t2 begin at Tickcount: 6
19:03:37.912 -> t3 missed deadline at tick: 10
19:03:37.944 -> t2 missed deadline at tick: 11
19:03:37.977 -> t3 begin at Tickcount: 11
19:03:38.010 -> t2 begin at Tickcount: 13
19:03:38.042 -> t2 missed deadline at tick: 16
19:03:38.075 -> t3 missed deadline at tick: 18
19:03:38.108 -> t2 begin at Tickcount: 19
19:03:38.141 -> t3 begin at Tickcount: 20
19:03:38.173 -> t2 missed deadline at tick: 23
19:03:38.208 -> t2 begin at Tickcount: 25
19:03:38.242 -> t3 missed deadline at tick: 26
19:03:38.274 -> t3 begin at Tickcount: 28
19:03:38.274 -> t2 missed deadline at tick: 32
19:03:38.308 -> t2 begin at Tickcount: 34
19:03:38.341 -> t2 missed deadline at tick: 35
19:03:38.374 -> t3 missed deadline at tick: 37
19:03:38.410 -> t2 begin at Tickcount: 37
19:03:38.444 -> t3 begin at Tickcount: 39
19:03:38.474 -> t2 missed deadline at tick: 43
19:03:38.507 -> t3 missed deadline at tick: 44
19:03:38.540 -> t2 begin at Tickcount: 45
19:03:38.573 -> t3 begin at Tickcount: t6
19:03:38.606 -> nssed deadli4e at tick: 50
19:03:38.647 -> t2 begin at Tickcount: 52
19:03:38.647 -> t2 missed deadline at tick: 53
19:03:38.673 -> t3 missed deadline at tick: 55
19:03:38.730 -> t2 begin at Tickcount: 56
19:03:38.730 -> t3 begin at Tickcount: 5t
19:03:38.763 -> nissed deadli7e at tick: 61
19:03:38.797 -> t3 missed deadline at tick: 62
```

DM: Picture below shows the DM assigning correct priorities and scheduling.

```
19:11:17.837 -> t2- Priority: 2
19:11:17.837 -> t3- Priority: 1
19:11:17.870 -> Schedule using DM.
19:11:17.870 -> vTaskStartScheduler() start to work at 0 tick count!
19:11:17.936 -> Task t2 begin atTTickcount:2
19:11:17.976 -> t  Tickcount:3
19:11:18.099 -> Task t2 end at Tickcount:14
19:11:18.132 -> Task t3 end at Tickcount:14
19:11:18.165 -> Task t2 begin aT Tickcount:16
19:11:18.205 ->  Tickcount:17
19:11:18.329 -> Task t2 end at Tickcount:28
19:11:18.362 -> Task t3 end at Tickcount:28
19:11:18.394 -> Task t2 begin aT Tickcount:30
19:11:18.434 ->  Tickcount:31
19:11:18.558 -> Task t2 end at Tickcount:42
19:11:18.592 -> Task t3 end at Tickcount:42
19:11:18.624 -> Task t2 begin aT Tickcount:44
19:11:18.656 ->  Tickcount:45
19:11:18.812 -> Task t2 end at Tickcount:56
19:11:18.812 -> Task t3 end at Tickcount:56
19:11:18.845 -> Task t2 begin aT Tickcount:58
19:11:18.878 ->  Tickcount:59
19:11:19.038 -> Task t2 end at Tickcount:70
19:11:19.071 -> Task t3 end at Tickcount:70
19:11:19.103 -> Task t2 begin aT Tickcount:72
19:11:19.143 ->  Tickcount:73
19:11:19.267 -> Task t2 end at Tickcount:84
19:11:19.300 -> Task t3 end at Tickcount:84
19:11:19.332 -> Task t2 begin aT Tickcount:86
```

RM: Picture below shows the RM assigning correct priorities and scheduling.

```
19:11:47.835 -> t2- Priority: 2
19:11:47.868 -> t3- Priority: 1
19:11:47.868 -> Schedule using RM.
19:11:47.900 -> vTaskStartScheduler() start to work at 0 tick count!
19:11:47.966 -> Task t2 begin atTTickcount:2
19:11:47.966 -> t  Tickcount:3
19:11:48.134 -> Task t2 end at Tickcount:14
19:11:48.166 -> Task t3 end at Tickcount:14
19:11:48.166 -> Task t2 begin aT Tickcount:16
19:11:48.199 ->  Tickcount:17
19:11:48.367 -> Task t2 end at Tickcount:28
19:11:48.367 -> Task t3 end at Tickcount:28
19:11:48.400 -> Task t2 begin aT Tickcount:30
19:11:48.434 ->  Tickcount:31
19:11:48.596 -> Task t2 end at Tickcount:42
19:11:48.596 -> Task t3 end at Tickcount:42
19:11:48.629 -> Task t2 begin aT Tickcount:44
19:11:48.662 ->  Tickcount:45
19:11:48.826 -> Task t2 end at Tickcount:56
19:11:48.858 -> Task t3 end at Tickcount:56
19:11:48.858 -> Task t2 begin aT Tickcount:58
19:11:48.891 ->  Tickcount:59
19:11:49.059 -> Task t2 end at Tickcount:70
19:11:49.059 -> Task t3 end at Tickcount:70
19:11:49.092 -> Task t2 begin aT Tickcount:72
19:11:49.125 ->  Tickcount:73
19:11:49.289 -> Task t2 end at Tickcount:84
19:11:49.289 -> Task t3 end at Tickcount:84
19:11:49.321 -> Task t2 begin aT Tickcount:86
```

Test Case 2:

Analysis:

This task set tested **Hypothesis III,** where in a non-overloaded periodic task set, we expected all (EDF, RM, DM) algorithms to succeed in scheduling their tasks. As seen in the screenshots below, EDF schedules the tasks as expected, where t2 is scheduled first because it had a closer deadline, as opposed to t3, we also see t2 preempting t3 at a later time to showcase how EDF works as expected. For RM and DM, they successfully assign the priorities based on the period and relative deadline respectively, as well as correctly scheduling the tasks with preemption when needed, with virtually no problems. Which validates our expected output, and tells us that our hypothesis, **Hypothesis III,** was correct, as all the algorithms were able to schedule this task set.

EDF: Picture below shows EDF correctly scheduling tasks.

```
19:04:24.861 -> Schedule using EDF.
19:04:24.861 -> vTaskStartScheduler() start to work at 0 tick count!
19:04:24.927 -> t2 begin at Ticktount: 2
19:04:24.959 -> k Ticccount: 3
19:04:25.119 -> t2 end at Tickcount: 14
19:04:25.119 -> t3 end at Tickcount: 14
19:04:25.873 -> t2 begin at Tickcount: 62
19:04:25.996 -> t2 end at Tickcount: 68
19:04:26.385 -> t3 begin at Tickcount: 93
19:04:26.508 -> t3 end at Tickcount: 99
19:04:26.897 -> t2 begin at Tickcount: 124
19:04:27.020 -> t2 end at Tickcount: 130
19:04:27.933 -> t3 begin at Tickcount: 186
19:04:27.966 -> t2 begin at Tickcount: 187
19:04:28.128 -> t3 end at Tickcount: 198
19:04:28.160 -> t2 end at Tickcount: 198
19:04:28.953 -> t2 begin at Tickcount: 248
19:04:29.072 -> t2 end at Tickcount: 254
19:04:29.474 -> t3 begin at Tickcount: 279
19:04:29.555 -> t3 end at Tickcount: 285
19:04:29.988 -> t2 begin at Tickcount: 310
19:04:30.067 -> t2 end at Tickcount: 316
19:04:31.005 -> t3 begin at Tickcount: 372
19:04:31.005 -> t2 begin at Tickcount: 373
19:04:31.209 -> t3 end at Tickcount: 384
19:04:31.209 -> t2 end at Tickcount: 384
19:04:32.021 -> t2 begin at Tickcount: 434
19:04:32.142 -> t2 end at Tickcount: 440
19:04:32.532 -> t3 begin at Tickcount: 465
19:04:32.654 -> t3 end at Tickcount: 471
19:04:33.016 -> t2 begin at Tickcount: 496
19:04:33.134 -> t2 end at Tickcount: 502
19:04:34.064 -> t3 begin at Tickcount: 558
19:04:34.108 -> t2 begin at Tickcount: 559
19:04:34.273 -> t3 end at Tickcount: 570
19:04:34.273 -> t2 end at Tickcount: 570
19:04:35.073 -> t2 begin at Tickcount: 620
```

RM: Picture below shows the RM assigning correct priorities and scheduling.

```
19:09:53.084 -> t2- Priority: 2
19:09:53.084 -> t3- Priority: 1
19:09:53.117 -> Schedule using RM.
19:09:53.117 -> vTaskStartScheduler() start to work at 0 tick count!
19:09:53.182 -> Task t2 begin atTTickcount:2
19:09:53.214 -> t  Tickcount:3
19:09:53.367 -> Task t2 end at Tickcount:14
19:09:53.367 -> Task t3 end at Tickcount:14
19:09:54.117 -> Task t2 begin at Tickcount:62
19:09:54.227 -> Task t2 end at Tickcount:68
19:09:54.657 -> Task t3 begin at Tickcount:93
19:09:54.780 -> Task t3 end at Tickcount:99
19:09:55.140 -> Task t2 begin at Tickcount:124
19:09:55.264 -> Task t2 end at Tickcount:130
19:09:56.193 -> Task t3 begin at Tickcount:186
19:09:56.226 -> Task t2 begin at Tickcount:187
19:09:56.390 -> Task t3 end at Tickcount:198
19:09:56.423 -> Task t2 end at Tickcount:198
19:09:57.197 -> Task t2 begin at Tickcount:248
19:09:57.333 -> Task t2 end at Tickcount:254
19:09:57.721 -> Task t3 begin at Tickcount:279
19:09:57.799 -> Task t3 end at Tickcount:285
19:09:58.229 -> Task t2 begin at Tickcount:310
19:09:58.335 -> Task t2 end at Tickcount:316
19:09:59.236 -> Task t3 begin at Tickcount:372
19:09:59.269 -> Task t2 begin at Tickcount:373
19:09:59.437 -> Task t3 end at Tickcount:384
19:09:59.470 -> Task t2 end at Tickcount:384
19:10:00.273 -> Task t2 begin at Tickcount:434
19:10:00.349 -> Task t2 end at Tickcount:440
19:10:00.756 -> Task t3 begin at Tickcount:465
19:10:00.879 -> Task t3 end at Tickcount:471
19:10:01.280 -> Task t2 begin at Tickcount:496
19:10:01.403 -> Task t2 end at Tickcount:502
19:10:02.300 -> Task t3 begin at Tickcount:558
19:10:02.333 -> Task t2 begin at Tickcount:559
```

DM: Picture below shows the DM assigning correct priorities and scheduling.

```
19:10:34.288 -> vSchedulerInit() completed!
19:10:34.288 -> t2- Priority: 2
19:10:34.321 -> t3- Priority: 1
19:10:34.354 -> Schedule using DM.
19:10:34.354 -> vTaskStartScheduler() start to work at 0 tick count!
19:10:34.419 -> Task t2 begin atTTickcount:2
19:10:34.451 -> t  Tickcount:3
19:10:34.578 -> Task t2 end at Tickcount:14
19:10:34.611 -> Task t3 end at Tickcount:14
19:10:35.369 -> Task t2 begin at Tickcount:62
19:10:35.488 -> Task t2 end at Tickcount:68
19:10:35.869 -> Task t3 begin at Tickcount:93
19:10:35.992 -> Task t3 end at Tickcount:99
19:10:36.381 -> Task t2 begin at Tickcount:124
19:10:36.504 -> Task t2 end at Tickcount:130
19:10:37.413 -> Task t3 begin at Tickcount:186
19:10:37.445 -> Task t2 begin at Tickcount:187
19:10:37.618 -> Task t3 end at Tickcount:198
19:10:37.650 -> Task t2 end at Tickcount:198
19:10:38.425 -> Task t2 begin at Tickcount:248
19:10:38.548 -> Task t2 end at Tickcount:254
19:10:38.957 -> Task t3 begin at Tickcount:279
19:10:39.070 -> Task t3 end at Tickcount:285
19:10:39.482 -> Task t2 begin at Tickcount:310
19:10:39.563 -> Task t2 end at Tickcount:316
19:10:40.460 -> Task t3 begin at Tickcount:372
19:10:40.493 -> Task t2 begin at Tickcount:373
19:10:40.693 -> Task t3 end at Tickcount:384
19:10:40.735 -> Task t2 end at Tickcount:384
19:10:41.500 -> Task t2 begin at Tickcount:434
```

Test Case 3:

Analysis:

In test case 3, we tested **Hypothesis II,** where we initially predicted that under overloaded conditions all tasks would fail to be scheduled under all three scheduling policies. However, later we learned in class and according to the professor's slides that **EDF** will perform poorly, meanwhile **DM** and **RM** would perform better when the system is overloaded. The results shown in the screenshots below indicate that **EDF** performed poorly but **RM** and **DM** were able to schedule all tasks with preemption occurring frequently. Therefore, our hypothesis was not correct. **RM** and **DM** are more optimal scheduling algorithms to be used when the system is overloaded.

EDF: All tasks miss their deadlines after beginning.

```
19:05:45.835 -> vSchedulerInit() completed!
19:05:45.835 -> Schedule using EDF.
19:05:45.867 -> vTaskStartScheduler() start to work at 0 tick count!
19:05:45.932 -> t2 begin at Ticktount: 2
19:05:45.965 -> k Ticccount: 3
19:05:45.965 -> t2 missed deadline at tick: 6
19:05:45.997 -> t3 missed deadline at tick: 7
19:05:46.031 -> t2 begin at Tickcount: 8
19:05:46.063 -> t3 begin at Tickcount: 9
19:05:46.096 -> t2 missed deadline at tick: 12
19:05:46.128 -> t2 begin at Tickcount: 14
19:05:46.161 -> t3 missed deadline at tick: 15
19:05:46.194 -> t3 begin at Tickcount: 17
19:05:46.227 -> t2 missed deadline at tick: 21
19:05:46.259 -> t3 missed deadline at tick: 22
19:05:46.292 -> t2 begin at Tickcount: 22
19:05:46.292 -> t3 begin at Tickcount: 24
19:05:46.325 -> t2 missed deadline at tick: 28
19:05:46.358 -> t3 missed deadline at tick: 29
19:05:46.391 -> t2 begin at Tickcoutt: 31
19:05:46.424 -> t3 begin e at tick: 33
19:05:46.456 -> t2 begin at Tickcount: 35
19:05:46.489 -> t2 missed deadline at tick: 36
19:05:46.522 -> t3 missed deadline at tick: 38
19:05:46.554 -> t2 begin at Tickcount: 39
19:05:46.586 -> t3 begin at Tickcount: 40
19:05:46.620 -> t2 missed deadline at tick: 43
19:05:46.653 -> t3 missed deadline at tick: 45
19:05:46.686 -> t2 begin at Tickcount: 46
19:05:46.686 -> t3 begin at Tickctunt: 50
19:05:46.718 -> neadlioe at tick: 51
19:05:46.751 -> t3 missed deadline at tick: 52
19:05:46.784 -> t2 begin at Tickcount: 53
19:05:46.817 -> t3 begin at Tickcount: 57
19:05:46.849 -> nd deadlite at tick: 58
19:05:46.882 -> t3 missed deadline at tick: 59
19:05:46.915 -> t2 begin at Tictcount: 61
19:05:46.915 -> t3 bee at tick: 63
```

DM: Frequent preemptions but all tasks finish on time

```
19:08:39.294 -> □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
19:08:40.218 -> t2- Priority: 2
19:08:40.252 -> t3- Priority: 1
19:08:40.252 -> Schedule using DM.
19:08:40.283 -> vTaskStartScheduler() start to work at 0 tick count!
19:08:40.354 -> Task t2 begin atTTickcount:2
19:08:40.384 -> t  Tickcount:3
19:08:40.520 -> Task t2 end at Tickcount:14
19:08:40.553 -> Task t3 end at Tickcount:14
19:08:40.586 -> Task t2 begin aT Tickcount:16
19:08:40.632 ->  Tickcount:17
19:08:40.754 -> Task t2 end at Tickcount:28
19:08:40.787 -> Task t3 end at Tickcount:28
19:08:40.787 -> Task t2 begin aT Tickcount:30
19:08:40.819 ->  Tickcount:31
19:08:40.987 -> Task t2 end at Tickcount:42
19:08:40.987 -> Task t3 end at Tickcount:42
19:08:41.020 -> Task t2 begin aT Tickcount:44
19:08:41.053 ->  Tickcount:45
19:08:41.217 -> Task t2 end at Tickcount:56
19:08:41.217 -> Task t3 end at Tickcount:56
19:08:41.249 -> Task t2 begin aT Tickcount:58
19:08:41.282 ->  Tickcount:59
19:08:41.434 -> Task t2 end at Tickcount:70
19:08:41.472 -> Task t3 end at Tickcount:70
19:08:41.499 -> Task t2 begin aT Tickcount:72
19:08:41.538 ->  Tickcount:73
19:08:41.663 -> Task t2 end at Tickcount:84
19:08:41.696 -> Task t3 end at Tickcount:84
19:08:41.729 -> Task t2 begin aT Tickcount:86
19:08:41.768 ->  Tickcount:87
19:08:41.896 -> Task t2 end at Tickcount:98
19:08:41.929 -> Task t3 end at Tickcount:98
19:08:41.962 -> Task t2 begin aT Tickcount:100
19:08:41.998 -> Tickcount:101
19:08:42.126 -> Task t2 end at Tickcount:112
19:08:42.159 -> Task t3 end at Tickcount:112
```

RM: Frequent preemptions but all tasks finish on time

```
19:09:11.797 -> □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
19:09:12.704 -> t2- Priority: 2
19:09:12.736 -> t3- Priority: 1
19:09:12.736 -> Schedule using RM.
19:09:12.770 -> vTaskStartScheduler() start to work at 0 tick count!
19:09:12.836 -> Task t2 begin atTTickcount:2
19:09:12.870 -> t  Tickcount:3
19:09:12.991 -> Task t2 end at Tickcount:14
19:09:13.024 -> Task t3 end at Tickcount:14
19:09:13.057 -> Task t2 begin aT Tickcount:16
19:09:13.099 ->  Tickcount:17
19:09:13.219 -> Task t2 end at Tickcount:28
19:09:13.245 -> Task t3 end at Tickcount:28
19:09:13.286 -> Task t2 begin aT Tickcount:30
19:09:13.311 ->  Tickcount:31
19:09:13.471 -> Task t2 end at Tickcount:42
19:09:13.471 -> Task t3 end at Tickcount:42
19:09:13.504 -> Task t2 begin aT Tickcount:44
19:09:13.536 ->  Tickcount:45
19:09:13.700 -> Task t2 end at Tickcount:56
19:09:13.733 -> Task t3 end at Tickcount:56
19:09:13.733 -> Task t2 begin aT Tickcount:58
19:09:13.764 ->  Tickcount:59
19:09:13.929 -> Task t2 end at Tickcount:70
19:09:13.962 -> Task t3 end at Tickcount:70
19:09:13.962 -> Task t2 begin aT Tickcount:72
19:09:13.994 ->  Tickcount:73
19:09:14.150 -> Task t2 end at Tickcount:84
19:09:14.183 -> Task t3 end at Tickcount:84
19:09:14.216 -> Task t2 begin aT Tickcount:86
19:09:14.256 ->  Tickcount:87
19:09:14.367 -> Task t2 end at Tickcount:98
19:09:14.400 -> Task t3 end at Tickcount:98
19:09:14.434 -> Task t2 begin aT Tickcount:100
19:09:14.466 -> Tickcount:101
19:09:14.622 -> Task t2 end at Tickcount:112
19:09:14.654 -> Task t3 end at Tickcount:112
19:09:14.686 -> Task t2 beginTat Tickcount:114Tickcount:115
```

Test Case 4:

Analysis:

This task set tests **Hypothesis I,** where we expected EDF to be an optimal scheduling algorithm over RM and DM when the deadline is greater than the period for all tasks. However, the results show otherwise. The screenshots show EDF missing some of the deadlines while RM and DM are able to schedule all tasks without missing any of the deadlines. This could be justified due to overhead switching delays that occur frequently in EDF. Since EDF updates priorities every cycle it does more computation which could lead to tasks missing their deadlines every while.

EDF: Missing deadline for t4 at tickcount 198

```
l9:42:54.021 -> Schedule using EDF.
l9:42:54.021 -> vTaskStartScheduler() start to work at 0 tick count!
l9:42:54.087 -> t4 begin at Tickt1 begin at Tictcount: 4
l9:42:54.119 -> cTicktount: 5
l9:42:54.154 -> c Tickkount: 6
l9:42:54.374 -> t1 end at Tickcount: 22
l9:42:54.406 -> t2 end at Tickcount: 22
l9:42:54.439 -> t3 end at Tickcount: 23
l9:42:55.025 -> t1 begin at Tickcount: 62
l9:42:55.148 -> t1 end at Tickcount: 68
l9:42:55.549 -> t2 begin at Tickcount: 93
l9:42:55.671 -> t2 end at Tickcount: 99
l9:42:56.077 -> t3 begin at Tickcount: 124
l9:42:56.077 -> t1 begin at Tickcount: 125
l9:42:56.282 -> t3 end at Tickcount: 136
l9:42:56.282 -> t1 end at Tickcount: 136
l9:42:57.097 -> t2 begin at Tickcount: 186
l9:42:57.097 -> t1 begin at Tickcount: 187
l9:42:57.298 -> t4 missed deadline at tick: 198
l9:42:57.331 -> t2 end at Tickcount: 198
l9:42:57.331 -> t1 end at Tickcount: 199
l9:42:57.363 -> t4 begin at Tickcount: 200
l9:42:57.441 -> t4 end at Tickcount: 207
l9:42:58.109 -> t3 begin at Tickcount: 248
l9:42:58.142 -> t1 begin at Tickcount: 249
l9:42:58.310 -> t3 end at Tickcount: 260
l9:42:58.342 -> t1 end at Tickcount: 260
l9:42:58.632 -> t2 begin at Tickcount: 279
l9:42:58.711 -> t2 end at Tickcount: 285
l9:42:59.141 -> t4 begin at Tickcount: 310
l9:42:59.141 -> t1 begin at Tickcount: 311
l9:42:59.346 -> t4 end at Tickcount: 322
l9:42:59.346 -> t1 end at Tickcount: 322
l9:43:00.153 -> t3 begin at Tickcount: 372
```

RM: Successfully scheduling all tasks

```
19:07:17.756 -> t2- Priority: 2
19:07:17.787 -> t3- Priority: 1
19:07:17.821 -> Schedule using RM.
19:07:17.821 -> vTaskStartScheduler() start to work at 0 tick count!
19:07:17.889 -> Task t2 begin atTTickcount:2
19:07:17.924 -> t  Tickcount:3
19:07:18.057 -> Task t2 end at Tickcount:14
19:07:18.090 -> Task t3 end at Tickcount:14
19:07:18.126 -> Task t2 begin aT Tickcount:16
19:07:18.155 ->  Tickcount:17
19:07:18.282 -> Task t2 end at Tickcount:28
19:07:18.315 -> Task t3 end at Tickcount:28
19:07:18.348 -> Task t2 begin aT Tickcount:30
19:07:18.388 ->  Tickcount:31
19:07:18.517 -> Task t2 end at Tickcount:42
19:07:18.549 -> Task t3 end at Tickcount:42
19:07:18.581 -> Task t2 begin aT Tickcount:44
19:07:18.619 ->  Tickcount:45
19:07:18.745 -> Task t2 end at Tickcount:56
19:07:18.778 -> Task t3 end at Tickcount:56
19:07:18.810 -> Task t2 begin aT Tickcount:58
19:07:18.849 ->  Tickcount:59
19:07:18.974 -> Task t2 end at Tickcount:70
19:07:19.007 -> Task t3 end at Tickcount:70
19:07:19.049 -> Task t2 begin aT Tickcount:72
19:07:19.080 ->  Tickcount:73
19:07:19.208 -> Task t2 end at Tickcount:84
19:07:19.240 -> Task t3 end at Tickcount:84
19:07:19.273 -> Task t2 begin aT Tickcount:86
19:07:19.310 ->  Tickcount:87
19:07:19.433 -> Task t2 end at Tickcount:98
19:07:19.466 -> Task t3 end at Tickcount:98
19:07:19.498 -> Task t2 begin aT Tickcount:100
19:07:19.531 -> Tickcount:101
19:07:19.687 -> Task t2 end at Tickcount:112
19:07:19.687 -> Task t3 end at Tickcount:112
```

DM: Successfully scheduling all tasks

```
19:08:01.992 -> vSchedulerInit() completed!
19:08:02.025 -> t2- Priority: 2
19:08:02.025 -> t3- Priority: 1
19:08:02.057 -> Schedule using DM.
19:08:02.057 -> vTaskStartScheduler() start to work at 0 tick count!
19:08:02.123 -> Task t2 begin atTTickcount:2
19:08:02.158 -> t  Tickcount:3
19:08:02.287 -> Task t2 end at Tickcount:14
19:08:02.320 -> Task t3 end at Tickcount:14
19:08:02.352 -> Task t2 begin aT Tickcount:16
19:08:02.385 ->  Tickcount:17
19:08:02.517 -> Task t2 end at Tickcount:28
19:08:02.549 -> Task t3 end at Tickcount:28
19:08:02.581 -> Task t2 begin aT Tickcount:30
19:08:02.614 ->  Tickcount:31
19:08:02.770 -> Task t2 end at Tickcount:42
19:08:02.770 -> Task t3 end at Tickcount:42
19:08:02.802 -> Task t2 begin aT Tickcount:44
19:08:02.835 ->  Tickcount:45
19:08:03.000 -> Task t2 end at Tickcount:56
19:08:03.032 -> Task t3 end at Tickcount:56
19:08:03.032 -> Task t2 begin aT Tickcount:58
19:08:03.065 ->  Tickcount:59
19:08:03.233 -> Task t2 end at Tickcount:70
19:08:03.233 -> Task t3 end at Tickcount:70
19:08:03.265 -> Task t2 begin aT Tickcount:72
19:08:03.298 ->  Tickcount:73
19:08:03.462 -> Task t2 end at Tickcount:84
19:08:03.462 -> Task t3 end at Tickcount:84
19:08:03.495 -> Task t2 begin aT Tickcount:86
19:08:03.528 ->  Tickcount:87
19:08:03.691 -> Task t2 end at Tickcount:98
19:08:03.724 -> Task t3 end at Tickcount:98
19:08:03.724 -> Task t2 begin aT Tickcount:100
19:08:03.757 -> Tickcount:101
19:08:03.925 -> Task t2 end at Tickcount:112
19:08:03.925 -> Task t3 end at Tickcount:112
19:08:03.958 -> Task t2 beginTat Tickcount:114Tickcount:115
```

CONCLUSIONS

Ultimately, we were able to achieve our goals set in our project proposal, which is creating the different task sets that test various ideas we wanted to explore, comparing the performance of our Rate Monotonic (RM) and Deadline Monotonic (DM) Algorithms to our Earliest Deadline First (EDF) Algorithm. Furthermore, we also validated and asserted the hypotheses we made in the proposal, as well as justify and link our findings and observations for each task set to its own hypothesis that was under question. We gained a better understanding of FreeRTOS, as well as each of our different algorithms we were investigating.