Embedded Operating System

Implementing Earliest Deadline First (EDF) with freeRTOS

Samuele Germiniani
samuele.germiniani@univr.it
Graziano Pravadelli
graziano.pravadelli@univr.it

OUTLINE

1. Introduction to EDF
2. Algorithm architecture
3. Implementation

1 - Introduction to EDF

1 - Introduction to EDF

Definition

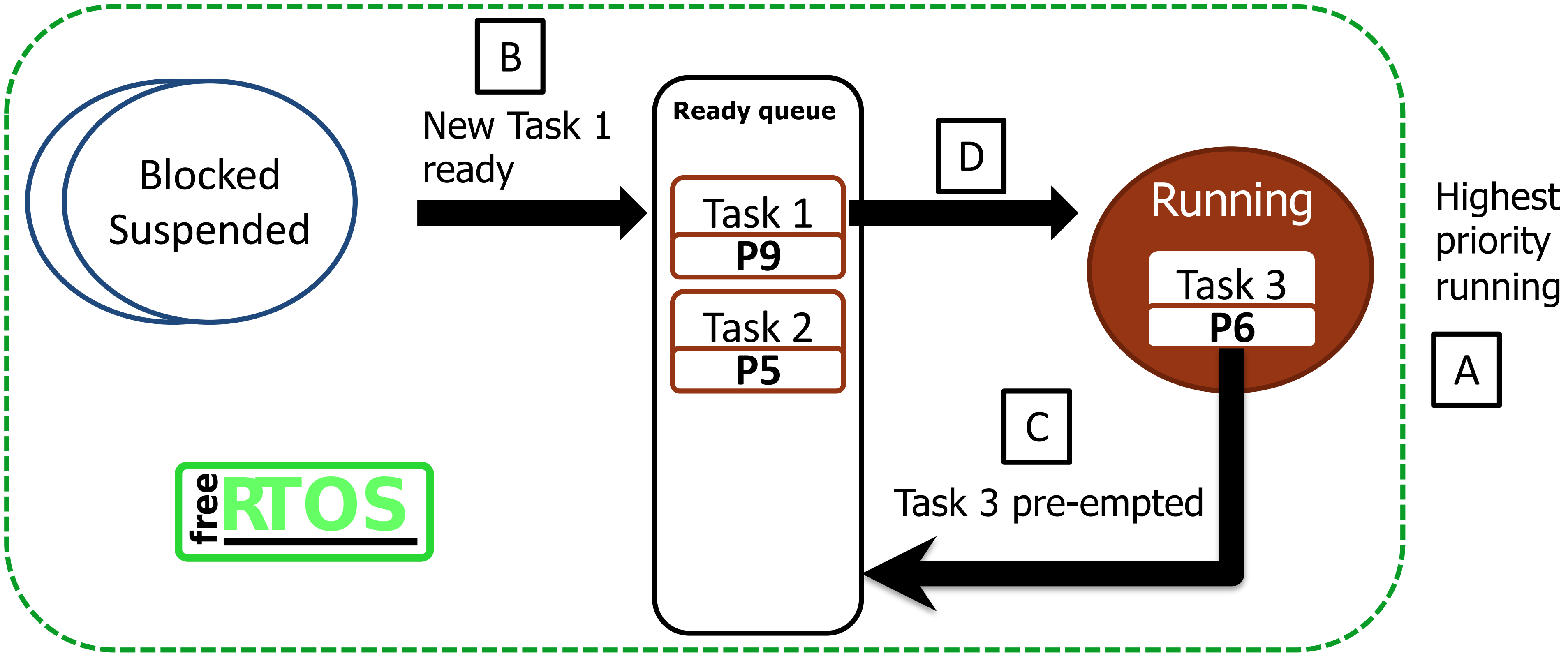
- Earliest deadline first (EDF) or least time to go is a dynamic priority scheduling algorithm used in real-time operating systems.

Scheduling Policy

❖ Whenever a scheduling event occurs (task finishes, new task released, etc.) the process closest to its deadline is the next to be scheduled for execution.

1 - Introduction to EDF

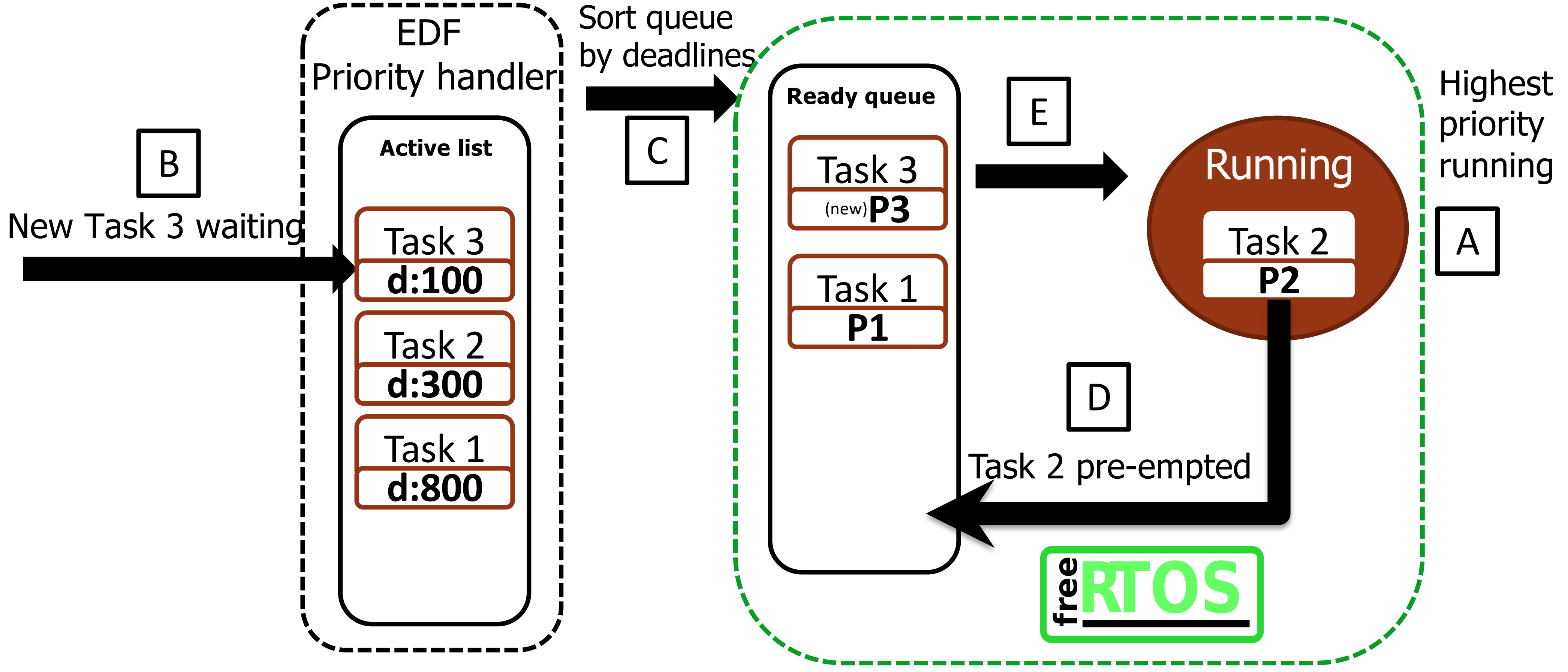
FreeRTOS scheduling policy



1 - Introduction to EDF

Can we modify the FreeRTOS scheduling policy?

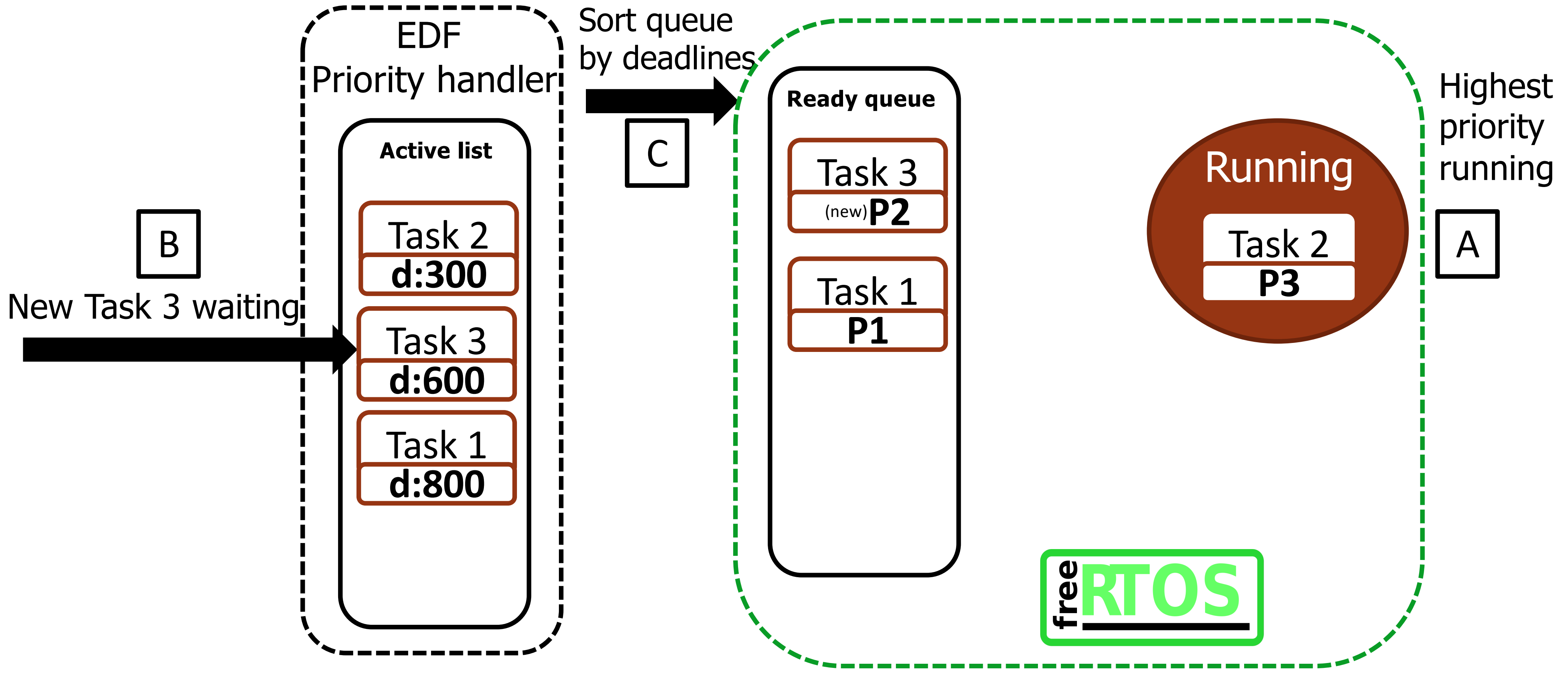
1) New Task with closest deadline



1 - Introduction to EDF

Can we modify the FreeRTOS scheduling policy?

2) New Task without closest deadline



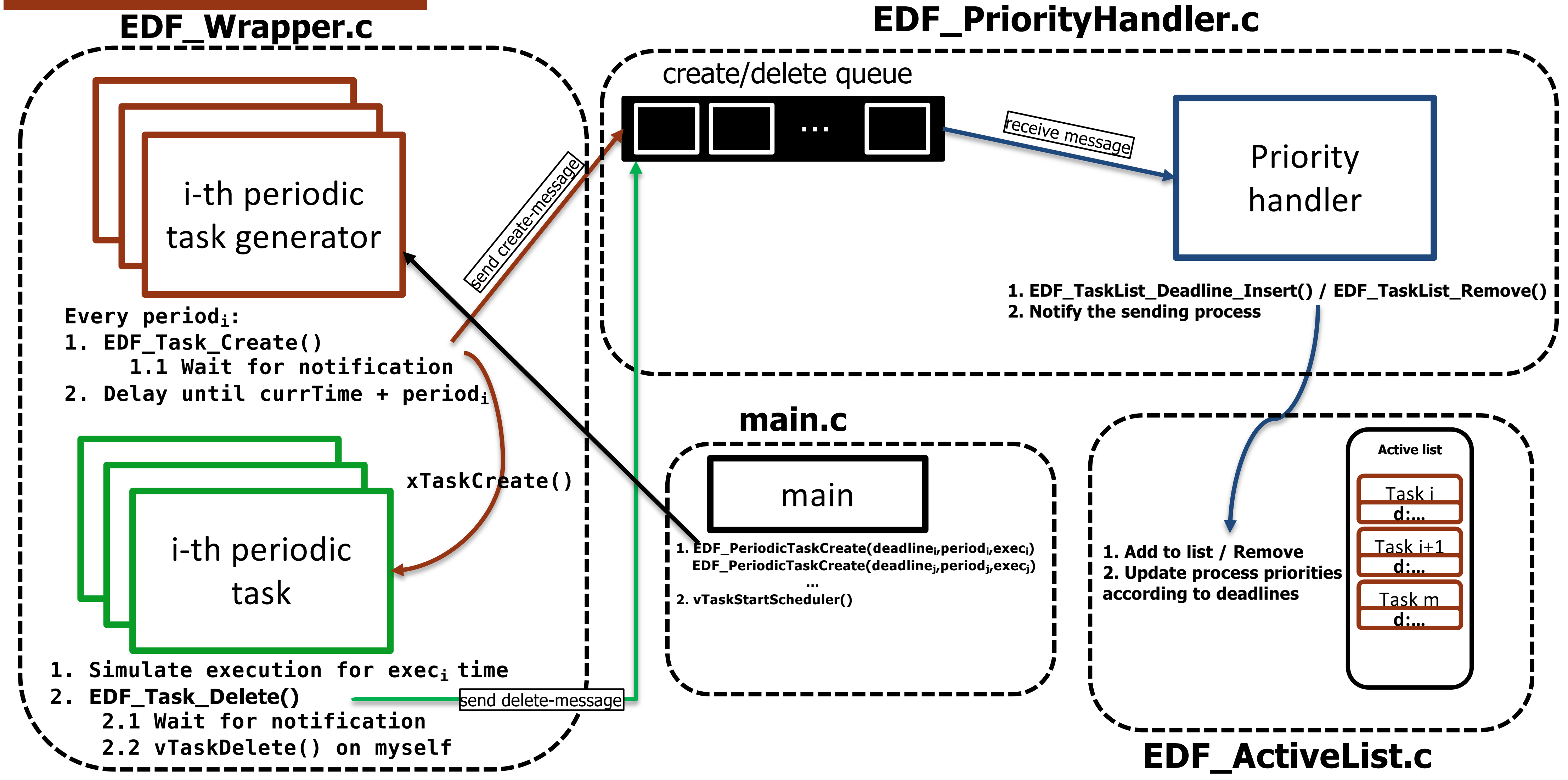
2 - Algorithm architecture

2 - Algorithm architecture

WARNING

The following algorithm is by no means the most efficient way of implementing EDF in FreeRTOS. However, it will exploit all FreeRTOS features described in the previous lecture.

High-level algorithm overview



```
Every period_i:
1. EDF_Task_Create()
     1.1 Wait for notification
2. Delay until currTime + period_i
```

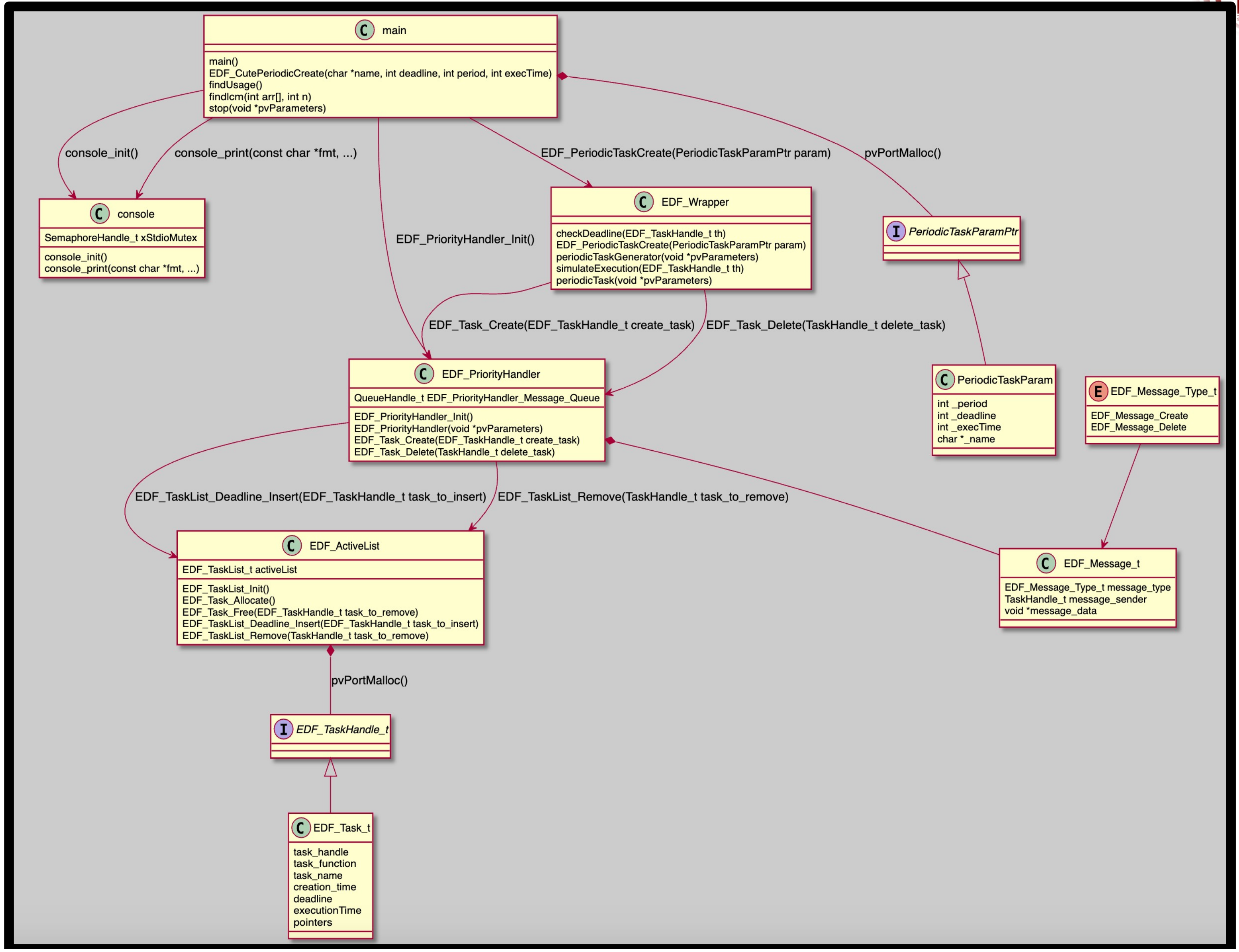
```
1. Simulate execution for exec_i time
2. EDF_Task_Delete()
     2.1 Wait for notification
     2.2 vTaskDelete() on myself
```

1. EDF_TaskList_Deadline_Insert() / EDF_TaskList_Remove()
2. Notify the sending process

```
1. EDF_PeriodicTaskCreate(deadline_i,period_i,exec_i)
   EDF_PeriodicTaskCreate(deadline_j,period_j,exec_j)
   ...
2. vTaskStartScheduler()
```

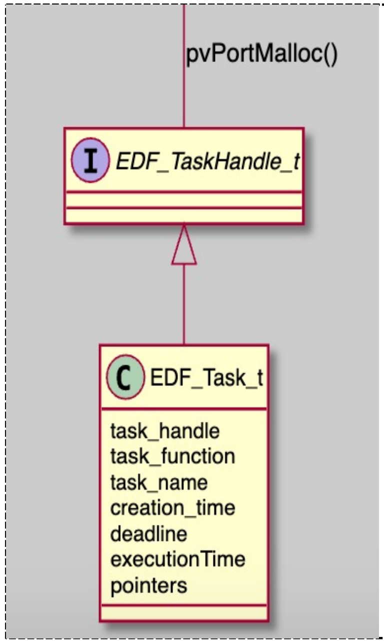
1. Add to list / Remove
2. Update process priorities according to deadlines

UML Diagram


main
main()
EDF_CutePeriodicCreate(char *name, int deadline, int period, int execTime)
findUsage()
findlcm(int arr[], int n)
stop(void *pvParameters)
console_init()
console_print(const char *fmt, ...)
EDF_PeriodicTaskCreate(PeriodicTaskParamPtr param)
pvPortMalloc()
console
SemaphoreHandle_t xStdioMutex
console_init()
console_print(const char *fmt, ...)
EDF_PriorityHandler_Init()
EDF_Wrapper
checkDeadline(EDF_TaskHandle_t th)
EDF_PeriodicTaskCreate(PeriodicTaskParamPtr param)
periodicTaskGenerator(void *pvParameters)
simulateExecution(EDF_TaskHandle_t th)
periodicTask(void *pvParameters)
PeriodicTaskParamPtr
EDF_Task_Create(EDF_TaskHandle_t create_task)
EDF_Task_Delete(TaskHandle_t delete_task)
EDF_PriorityHandler
QueueHandle_t EDF_PriorityHandler_Message_Queue
EDF_PriorityHandler_Init()
EDF_PriorityHandler(void *pvParameters)
EDF_Task_Create(EDF_TaskHandle_t create_task)
EDF_Task_Delete(TaskHandle_t delete_task)
PeriodicTaskParam
int _period
int _deadline
int _execTime
char *_name
EDF_Message_Type_t
EDF_Message_Create
EDF_Message_Delete
EDF_TaskList_Deadline_Insert(EDF_TaskHandle_t task_to_insert)
EDF_TaskList_Remove(TaskHandle_t task_to_remove)
EDF_ActiveList
EDF_TaskList_t activeList
EDF_TaskList_Init()
EDF_Task_Allocate()
EDF_Task_Free(EDF_TaskHandle_t task_to_remove)
EDF_TaskList_Deadline_Insert(EDF_TaskHandle_t task_to_insert)
EDF_TaskList_Remove(TaskHandle_t task_to_remove)
EDF_Message_t
EDF_Message_Type_t message_type
TaskHandle_t message_sender
void *message_data
pvPortMalloc()
EDF_TaskHandle_t
EDF_Task_t
task_handle
task_function
task_name
creation_time
deadline
executionTime
pointers

2 - Algorithm architecture

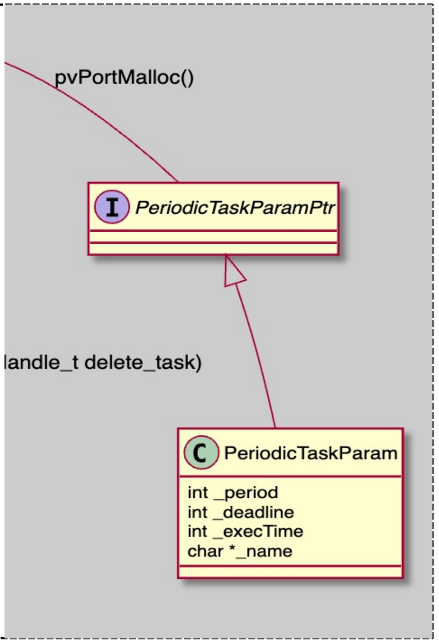
Understanding the data structures

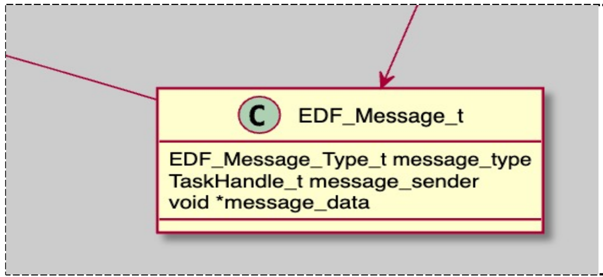


- **EDF_Task_t** is used for active tasks in the active task list
- **EDF_TaskHandle_t** is a pointer to **EDF_Task_t**
- To create a new item:

(EDF_TaskHandle_t)pvPortMalloc(sizeof(EDF_Task_t))

- **PeriodicTaskParam** is used to hold the parameters of a periodic task before creating a **EDF_Task_t**
- **PeriodicTaskParamPtr** is a pointer to **PeriodicTaskParam**





- **EDF_Message_t** is used in the queue to ask for creation or removal of tasks. The queue must be initialized with this data type.

3 - Implementation

3 - Implementation

Getting started

1. cd FreeRTOSv202104.00/FreeRTOS/Demo/
2. git clone https://gitlab.com/SamueleGerminiani/posix_eos2_start.git
3. cd posix_eos2_start

3 - Implementation

1. Understanding the data structures

Task

- Find the definition of the data structures listed in slide 12 and add the missing fields

You will need the following:

- Files: EDF_ActiveList.h, EDF_PriorityHandler.h
- Your eyes

3 - Implementation

2. Initialise the priority handler

Task

- Create the queue and the priority handler task. Follow the steps in function **EDF_PriorityHandler_Init.**

You will need the following:

- Files: EDF_PriorityHandler.c
- xQueueCreate()
- xTaskCreate()

3 - Implementation

3. Allocate and Free an active task

Task

- Complete functions **EDF_Task_Allocate** and **EDF_Task_Free** to allocate and free the memory of an active task.

You will need the following:

- Files: EDF_ActiveList.c
- pvPortMalloc()
- vPortFree()

N.B remember to cast to void* before freeing memory with vPortFree()

3 - Implementation

4. Implement the priority handler task function

Task

- Complete task function **EDF_PriorityHandler** to receive messages from the queue and to carry out the respective requests.

You will need the following:

- Files: EDF_PriorityHandler.c
- xQueueReceive()
- xTaskNotifyGive()
- EDF_TaskList_Deadline_Insert()
- EDF_TaskList_Remove()

3 - Implementation

5. Send creation requests

Task

- Complete function **EDF_Task_Create** to send create-messages to the queue and to create periodic tasks.

You will need the following:

- Files: EDF_PriorityHandler.c
- vTaskSuspend()
- vTaskResume()
- xTaskCreate()
- xQueueSend()
- ulTaskNotifyTake(pdTRUE, portMAX_DELAY);

3 - Implementation

6. Send deletion requests

Task

- Complete function **EDF_Task_Delete** to send delete-messages to the queue and to delete the periodic tasks.

You will need the following:

- Files: EDF_PriorityHandler.c
- vTaskDelete()
- xQueueSend()
- ulTaskNotifyTake(pdTRUE, portMAX_DELAY);

3 - Implementation

7. Insert a new active task in the active list

Task

- Implement function **EDF_TaskList_Deadline_Insert** to insert a new active task in the list

You will need the following:

- Files: EDF_ActiveList.c
- LIST_EMPTY(…);
- LIST_INSERT_HEAD(...,…,pointers);
- LIST_INSERT_BEFORE(…,…,pointers);
- LIST_INSERT_AFTER(…,…,pointers);
- LIST_NEXT(…,pointers);
- LIST_END(…);
- vTaskPrioritySet()
- **your brain**

At https://man.openbsd.org/queue.3 you can find the documentation for the **LIST** APIs.

3 - Implementation

8. Delete an active task from the active list

Task

- Implement function **EDF_TaskList_Remove** to remove an active task from the list

You will need the following:

- Files: EDF_ActiveList.c
- LIST_REMOVE(…,pointers)
- EDF_Task_Free()
- LIST_NEXT(…,pointers);

At https://man.openbsd.org/queue.3 you can find the documentation for the **LIST** APIs.

3 - Implementation

9. Implement the periodic task function

Task

- Implement task function **periodicTask.** An instance of this function will be created by the **periodic task generator** after each period.

You will need the following:

- Files: EDF_Wrapper.c
- console_print()
- xTaskGetTickCount()
- uxTaskPriorityGet()
- EDF_Task_Delete()

3 - Implementation

10. Implement the periodic task generator

Task

- Implement task function **periodicTaskGenerator**. It will create an instance of **periodicTask** after each period.

You will need the following:

- Files: EDF_Wrapper.c
- EDF_Task_Allocate()
- xTaskGetTickCount()
- EDF_Task_Create()
- vTaskDelayUntil()

3 - Implementation

11. Implement the periodic task generator API

Task

- Implement task function **EDF_PeriodicTaskCreate**. It will create an instance of **periodicTaskGenerator** with the give parameters.

You will need the following:

- Files: EDF_Wrapper.c
- xTaskCreate()

3 - Implementation

12. Complete main.c

Task

- Complete the main file. Follow the steps in the main function.

n.b. To complete this task, you will need to understand the "static" functions implemented in main.c

You will need the following:

- Files: main.c
- EDF_PriorityHandler_Init()
- findlcm()
- findUsage()
- console_print()
- xTaskCreate()