# RTOS

# Types of RTOS

## Hard Real Time systems

a) A hard real-time system considers timelines as a deadline, and it should not be omitted in any circumstances.
b) Hard real-time Systems do not use any permanent memory, so their processes must be complete properly in the first time itself.
c) Hard Real-Time System must generate accurate responses to the events within the specified time.
d) A hard real-time system is a purely deterministic and time constraint system.

## Examples of Hard Real Time Systems

a) Missile Guidance Systems
b) Medical System
c) Railway signaling system
d) Nuclear reactor control systems

# Types of RTOS

## Sort Real Time systems

a) In a soft real-time system, the meeting of deadline is not compulsory for every task, but the process should get processed and give the result.

b) Even the soft real-time systems cannot miss the deadline for every task or process according to the priority it should meet the deadline or miss the deadline.

## Examples of Soft Real Time Systems

- Personal computer
- Audio and video systems
- Set-top boxes
- DVD Players
- Electronic games
- Web browsing

# SCHEDULABILITY ANALYSIS

## SCHEDULING CRITERIA

1. **CPU utilization:** CPU should be working most of the time (Ideally 100% all the time)
2. **Throughput:** total number of processes completed per unit time(10 tasks/second)
3. **Turnaround time (TAT):** amount of time taken to execute a particular process, $TAT_i = CT_i - AT_i$ (Where $CT_i \rightarrow$ Completion Time, $AT_i \rightarrow$ Arrival Time)
4. **Waiting time(WT):** time periods spent waiting in the ready queue by a process to acquire get control on the CPU, $WT_i = TAT_i - BT_i$ (Where $BT_i \rightarrow$ CPU burst time)
5. **Load average:** average number of processes residing in the ready queue waiting for their turn to get into the CPU
6. **Response time:** Amount of time it takes from when a request was submitted until the first response is produced

**SCHEDULING OBJECTIVE:** **Max →CPU utilization, Throughput**
**Min → Turnaround time, Waiting time, Load average, Response time**

# SCHEDULABILITY ANALYSIS

## TASK MODEL

- A task = (C, P)
  - C: worst case execution time/computing time (C<=P!)
  - P: time period (Deadline = time Period)
  - C/P is CPU utilization of a task

U>1 (overload): some task will fail to meet its deadline
U<=1 : it will depend on the scheduling algorithms
U=1 : CPU is kept busy, all deadlines will be met

- A task set: (Ci,Pi)
  - All tasks are independent
  - The periods of tasks start at 0 simultaneously
  - $U=\Sigma(Ci/Pi)$ is CPU utilization of a task set

- CPU utilization is a measure on how busy the processor.

# SCHEDULABILITY ANALYSIS

## SCHEDULABILITY TEST

- Schedulability test determine whether a given task set is feasible to schedule?
- Necessary test:
  - If test is passed, tasks may be schedulable but not necessarily
  - If test is not passed, tasks are definitely not schedulable
- Sufficient test:
  - If test is passed, then task are definitely schedulable
  - If test is not passed, tasks may be schedulable but not necessarily
- Exact test: (Necessary test + Sufficient test)
  - The task set is schedulable if and only if it passes the test

# SCHEDULABILITY ANALYSIS

## SCHEDULABILITY TEST

- Necessary test

$$U = \sum_{i=1}^{N} \left( \frac{C_i}{P_i} \right) \leq 1$$

- Sufficient test (Utilization Bound test)

$$U = \sum_{i=1}^{N} \left( \frac{C_i}{PT_i} \right) \leq N(2^{\frac{1}{N}} - 1)$$

| N | B(N) |
|---|---|
| 1 | 1.0 |
| 2 | 0.828 |
| 3 | 0.779 |
| 4 | 0.756 |
| 5 | 0.743 |
| 6 | 0.734 |
| ∞ | 0.693 |

# REAL TIME SCHEDULING

# REAL TIME SCHEDULING

- Scheduling refers to the way processes are assigned to run on the available CPUs, since there are typically many more processes running on the system.
- Real-time scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU in RTOS.
- By switching the CPU among processes, the operating system can make the embedded system is more productive.
- Scheduling algorithm is the method by which threads, processes or data flows are given access to system resources
- The need for a scheduling algorithm arises from requirement for most modern systems to perform multitasking.

# REAL TIME SCHEDULING

## REAL-TIME SCHEDULING ALGORITHM

- The purpose of a real-time scheduling algorithm is to ensure that critical timing constraints, such as deadlines and response time, are met.
- Real-time systems use preemptive multitasking with priorities are assigned to tasks, and the RTOS always executes the task with highest priority.
- Most algorithms are classified as
  - Fixed-priority: Algorithm assigns priorities at design time, and those priorities remain constant for the lifetime of the task. Ex.: Rate-Monotonic Scheduling (RMS)
  - Dynamic-priority: Assigns priorities dynamically at runtime, based on execution parameters of tasks, such as upcoming deadlines. Ex.: Earliest Deadline First (EDF)
  - Mixed-priority: This algorithm has both static and dynamic components.

# REAL TIME SCHEDULING

## TYPES OF SCHEDULING ALGORITHM

- Non pre-emptive Algorithm:
  - First come First served (FCFS)
  - Priority(Non-pre-emptive)
  - Shortest Job First(SJF)

- Pre-emptive Algorithm:
  - Shortest remaining Time First(SRTF)
  - Round Robin(RR)
  - Priority(Pre-emptive)
  - Rate Monotonic Scheduling (RMS)
  - Earliest Deadline First (EDF)

# REAL TIME SCHEDULING

## RATE MONOTONIC SCHEDULING(RMS)

- **Concept:** Task with the shortest period executes with the highest priority.
- **Working :**
  - Rate-monotonic is a fixed priority based scheduling.
  - The scheduling scheme is pre-emptive; it ensures that a task is pre-empted if another task with a shorter period is expected to run.
  - Used in embedded systems where the nature of the scheduling is deterministic.
  - Consider three tasks with a period Task-1(10ms), Task-2(15ms), Task-3 (20ms), then as per RMS priority of the tasks can be assigned as:

priority (task1) > priority (task2) > priority (task3)

# REAL TIME SCHEDULING

## RATE MONOTONIC SCHEDULING(RMS) – EXAMPLE-1

- Consider set of task running on a Automotive control system as follows
  - Speed measurement Task (T1): C=20ms, P=100ms, D=100ms
  - ABS control Task (T2): C=40ms, P=150ms, D=150ms
  - Fuel injection Task (T3): C=100ms, P=350ms, D=350ms
  - Other software with soft deadlines e.g. audio, air condition etc.,

### Necessary Test

$$U = \sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq 1$$

U=(20/100)+(40/150)+(100/350)
= 0.2 + 0.2666 + 0.2857 = 0.7517 ≤ 1

Necessary Test is passed hence given task set must be tested under sufficient test to conclude the Schedulability. But, if necessary test is failed we may conclude given task set is definitely not schedulable by any scheduling algorithm.

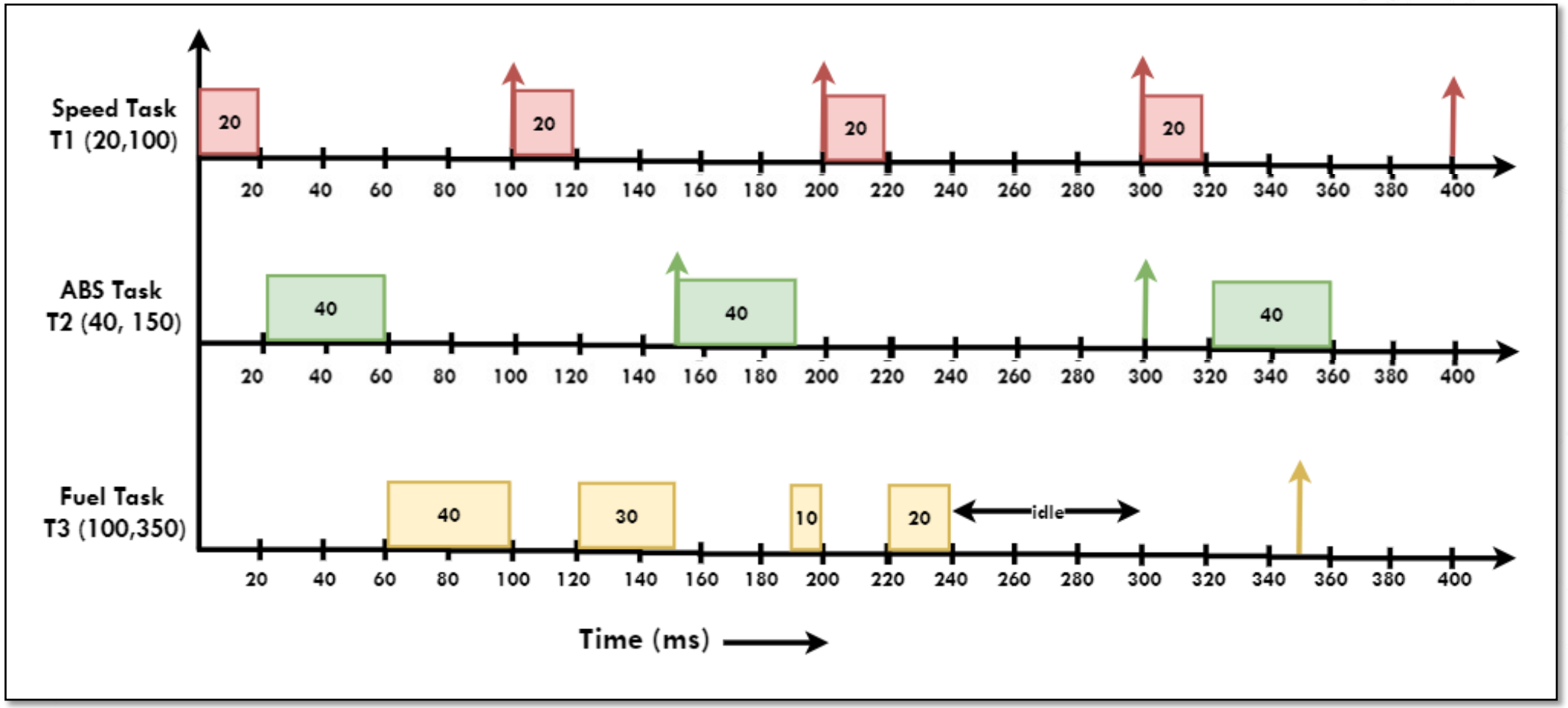
### Sufficient Test

$$U = \sum_{i=1}^{N} \left(\frac{C_i}{PT_i}\right) \leq N(2^{\frac{1}{N}} - 1)$$

B(3)=0.779
U=0.7517 ≤ B(3) ≤ 1

Since given task set passes Sufficient test we may conclude it is definitely schedulable under RMS

# REAL TIME SCHEDULING




This is only for the first periods. But this is enough to conclude that the task set is schedulable

# REAL TIME SCHEDULING

## RATE MONOTONIC SCHEDULING(RMS) – EXAMPLE-2

- Consider set of four task (Ci, Pi) running on embedded system as follows, T1= (1, 3) , T2= (1, 5) , T3= (1, 6) , T4= (2, 10)

### Necessary Test

$$U = \sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq 1$$

U=(1/3)+(1/5)+(1/6)+(2/10)=0.899 ≤ 1

Necessary Test is passed hence given task set must be tested under sufficient test to conclude the Schedulability. But, if necessary test is failed we may conclude given task set is definitely not schedulable by any scheduling algorithm.

### Sufficient Test

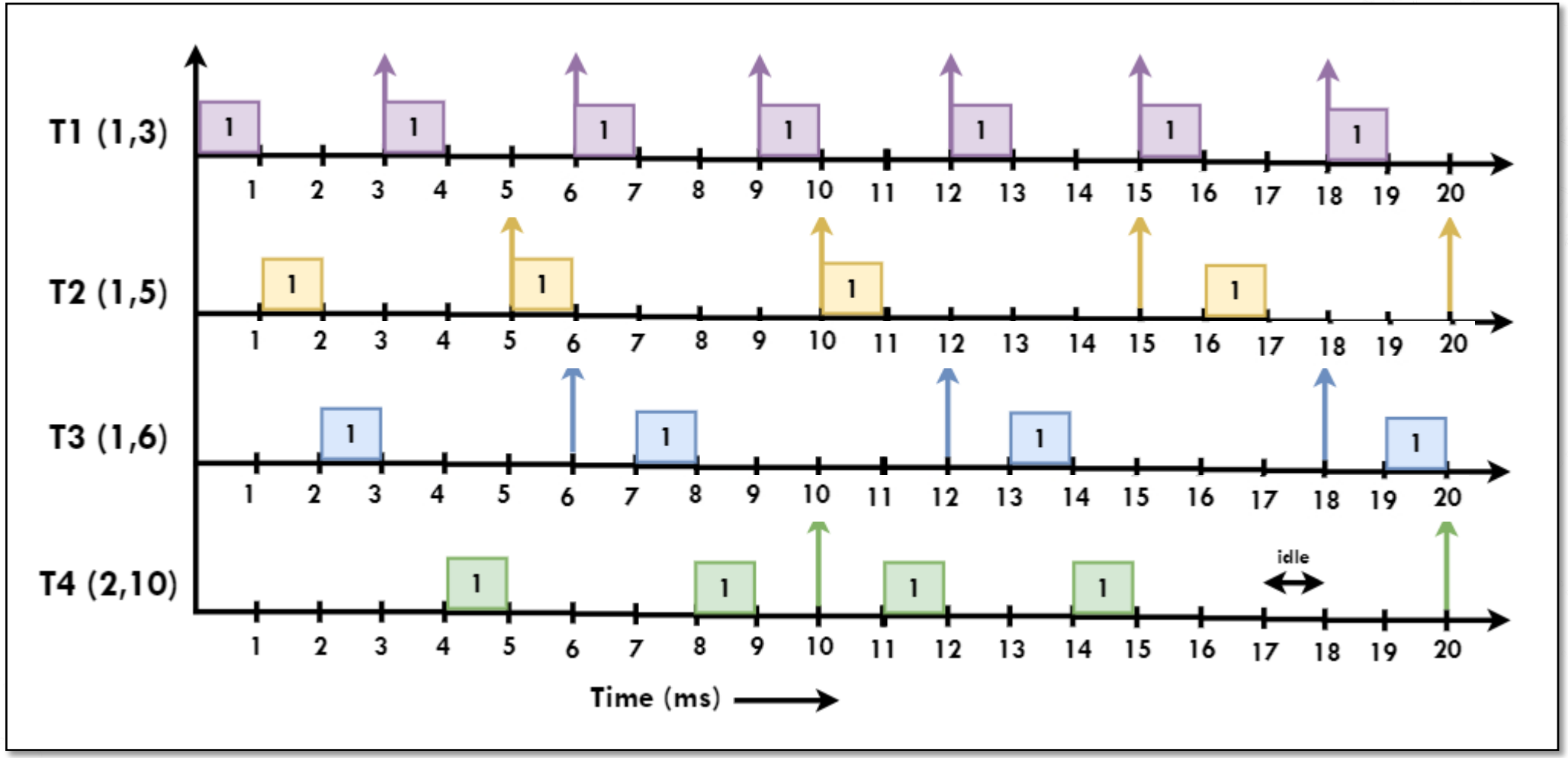
$$U = \sum_{i=1}^{N} \left(\frac{C_i}{PT_i}\right) \leq N(2^{\frac{1}{N}} - 1)$$

B(4)=0.756

The given task set fails in the Sufficient test due to U=0.899 > B(4) we can't conclude precisely whether given task set is schedulable or not under RMS

# REAL TIME SCHEDULING

## RATE MONOTONIC SCHEDULING(RMS) – EXAMPLE-2



# REAL TIME SCHEDULING

## RATE MONOTONIC SCHEDULING(RMS) – EXAMPLE-3

- Consider set of three task (Ci, Pi) running on embedded system as follows, T1= (10, 30) , T2= (15, 40) , T3= (5, 50)

### Necessary Test

$$U = \sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq 1$$

U=(10/30)+(15/40)+(5/50)=0.808 ≤ 1

Necessary Test is passed hence given task set must be tested under sufficient test to conclude the Schedulability. But, if necessary test is failed we may conclude given task set is definitely not schedulable by any scheduling algorithm.

### Sufficient Test

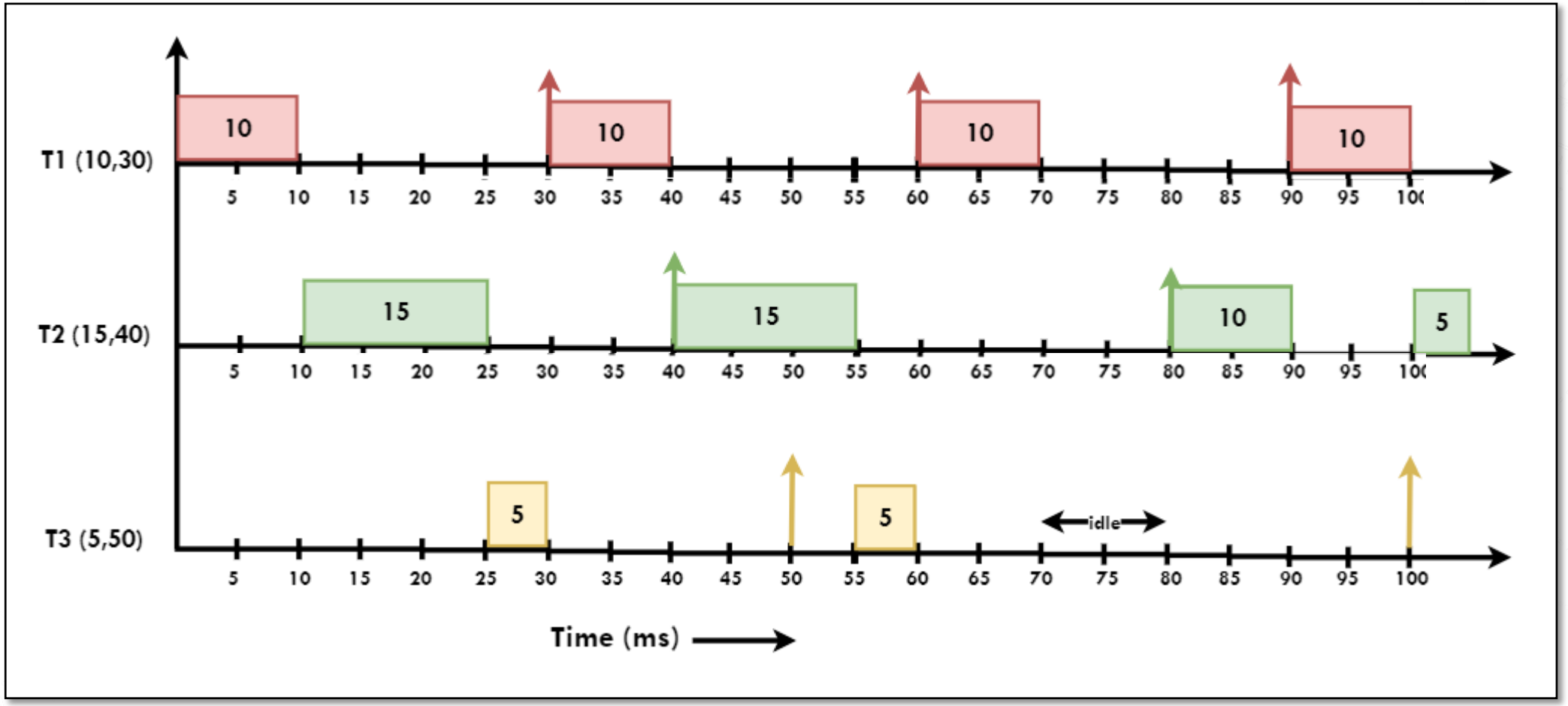
$$U = \sum_{i=1}^{N} \left(\frac{C_i}{PT_i}\right) \leq N(2^{\frac{1}{N}} - 1)$$

B(3)=0.779

The given task set fails in the Sufficient test due to U=0.808 > B(3) we can't conclude precisely whether given task set is schedulable or not under RMS

# REAL TIME SCHEDULING

## RATE MONOTONIC SCHEDULING(RMS) – EXAMPLE-3



# REAL TIME SCHEDULING

## RATE MONOTONIC SCHEDULING(RMS) - PROs & CONs

- PROs:
  - Simple to understand
  - Easy to implement (static/fixed priority assignment)
  - Stable: though some of the lower priority tasks fail to meet deadlines, others may meet deadlines
- CONs:
  - Lower CPU utilization
  - Requires D=T
  - Only deal with independent tasks
  - Non-precise Schedulability analysis

# REAL TIME SCHEDULING

## EARLIEST DEADLINE FIRST (EDF)

- Concept: Task closest to the end of its period assigned the highest priority
- Working :
  - Earliest deadline first is a dynamic priority based scheduling.
  - The scheduling scheme is pre-emptive; it ensures that a task is pre-empted if another task having a nearest deadline is expected to run.
  - EDF is optimal scheduling i.e it can schedule the task set if any other scheduling algorithm can schedule
  - If two task have the same deadlines, need to chose one if the two at random but in most of the case it proceed with the same task which is currently executing on the CPU to avoid context switching overhead.

# REAL TIME SCHEDULING

## EARLIEST DEADLINE FIRST (EDF) – EXAMPLE-1

- Consider set of task running on a weather monitoring station as follows
  - Temperature measurement Task (T1): C=1ms, P=4ms
  - Humidity measurement Task (T2): C=2ms, P=5ms
  - Co2 measurement Task (T3): C=2ms, P=7ms

### Necessary Test

$$U = \sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq 1$$

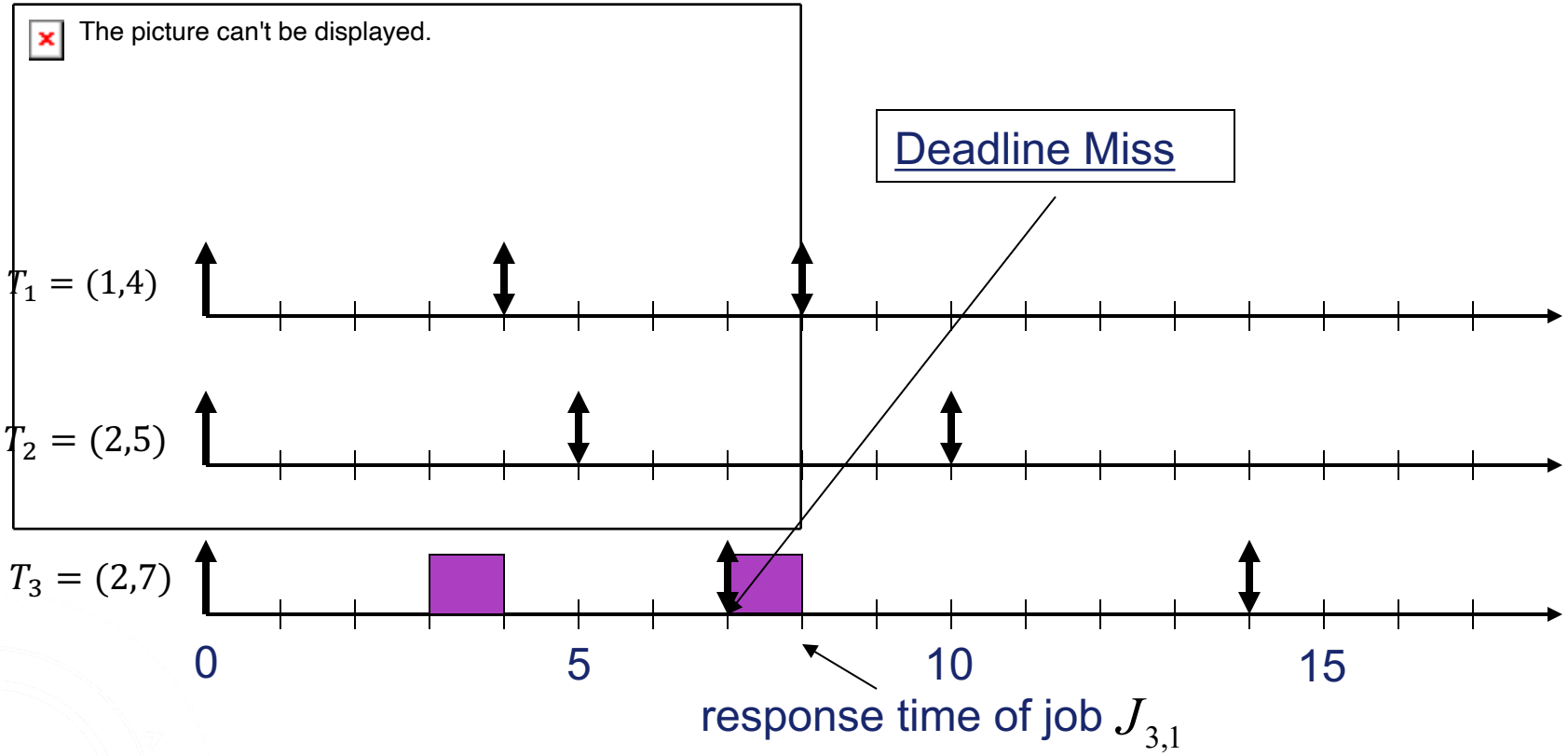
U=(1/4)+(2/5)+(2/7)
= 0.25 + 0.4+ 0.2857
= 0.935 ≤ 1

Necessary Test is passed hence given task set must be schedulable by EDF

# REAL TIME SCHEDULING

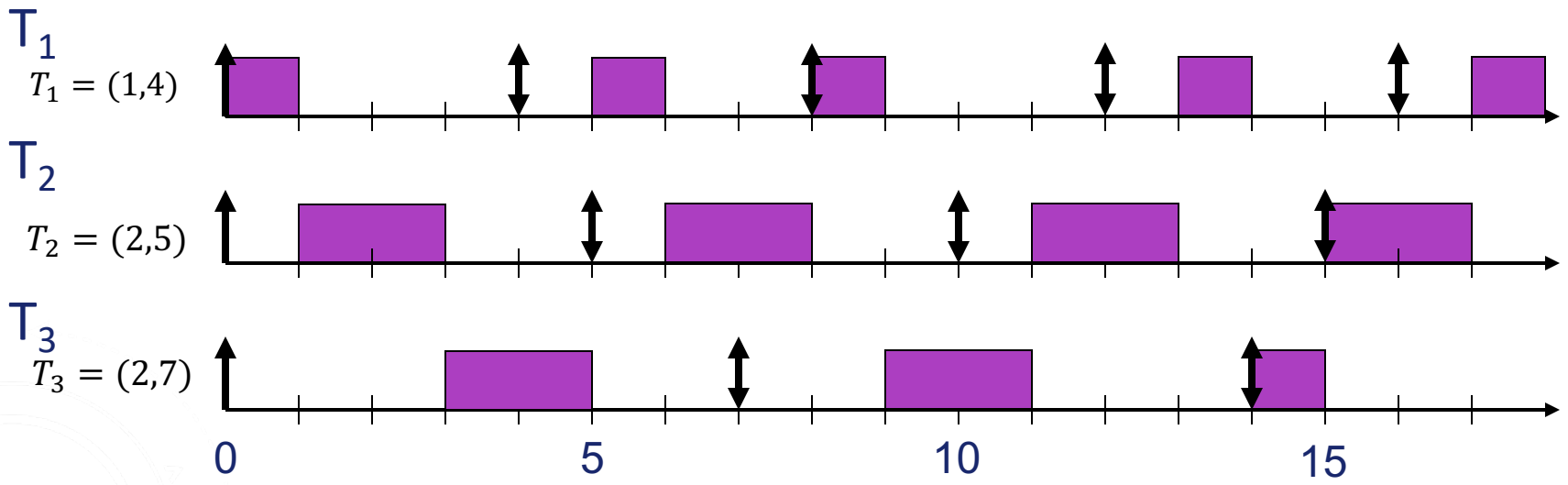
## EARLIEST DEADLINE FIRST (EDF) – EXAMPLE-1

Let's first try with RMS



# REAL TIME SCHEDULING

## EARLIEST DEADLINE FIRST (EDF) – EXAMPLE-1



# REAL TIME SCHEDULING

## EARLIEST DEADLINE FIRST (EDF) – EXAMPLE-2

- Consider set of four task (Ci, Pi) running on embedded system as follows, T1= (1, 3) , T2= (1, 5) , T3= (1, 6) , T4= (2, 10)

### Necessary Test

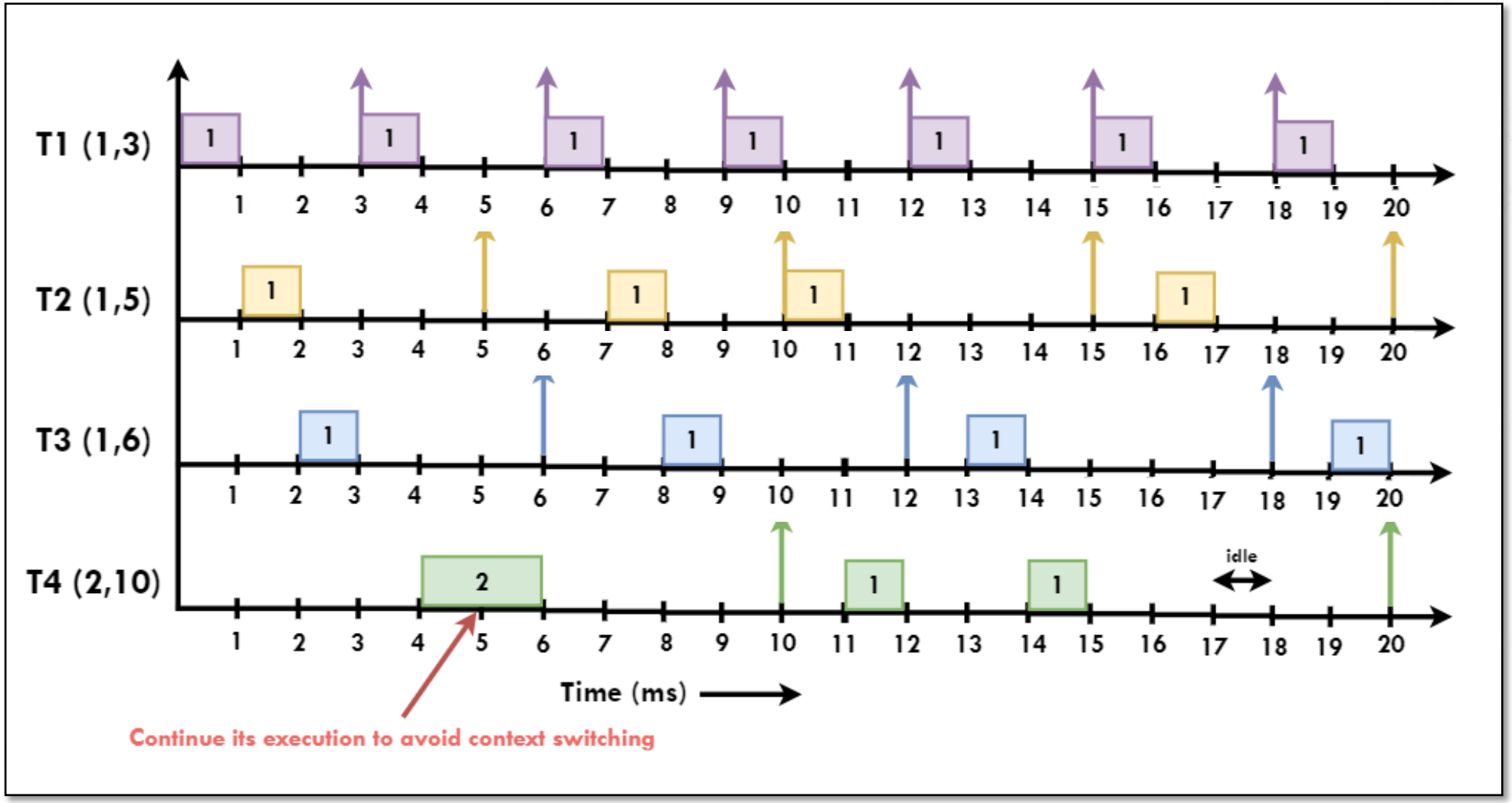
$$U = \sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq 1$$

$U=(1/3)+(1/5)+(1/6)+(2/10)=0.899 \leq 1$

Necessary Test is passed hence given task set must be schedulable by EDF

# REAL TIME SCHEDULING

## EARLIEST DEADLINE FIRST (EDF) – EXAMPLE-2



# REAL TIME SCHEDULING

## EARLIEST DEADLINE FIRST (EDF) - PROs & CONs

- PROs:
  - It works for all types of tasks: periodic or non periodic
  - Simple Schedulability test
  - Best CPU utilization
  - Optimal scheduling algorithm
- CONs:
  - Difficult to implement due to the dynamic priority-assignment.
  - Any task could get the highest priority even if the task is not important
  - Non stable because if any task fails to meet its deadline, the system is not predictable