Real-Time operating System

Outline

- Real-Time System
- Task categories
- Why we need scheduling
- Real time scheduling taxonomies
- Scheduling Periodic Tasks
- Real-Time Scheduling Algorithms
  - Static scheduling algorithms
  - Dynamic scheduling algorithms
  - Hybrid algorithm
- references

Real-Time System

- Real-time systems have been defined as: "*those systems in which the correctness of the system depends not only on the logical result of the computation, but also on the time at which the results are produced*"
- Correct function at correct time
- Usually embedded
- Deadlines
  - Hard real-time systems
  - Soft real-time systems
  - Firm real-time systems

Real Time Tasks

- Task in which performance is judged on the basis of time.
- The result of task are good if has produced within specified time constraints otherwise system fail or reduced value for the quality of service.

- Process control in industrial plants
- Robotics
- Air Traffic control
- Telecommunications
- Weapon guidance system e.g. Guided missiles
- Medical diagnostic and life support system
- Automatic engine control system
- Real time data base

Real time speech or video processing

- A speech or moving picture sample of 1 second, if processed in 1 second or in less, makes it real time processing
- if it takes more than 1 sec , it is no longer real time processing

RTOS

Question: Is real time systems and embedded systems are same?

- Embedded systems are designed for specific set of applications.
- When such a system requires ‘time constrained’ operation, it becomes real-time embedded system
- All embedded system are not real-time system.
- E.g.
  - printer is **not real-time system**
  - A robot which counts objects passing through conveyor belt is real-time system

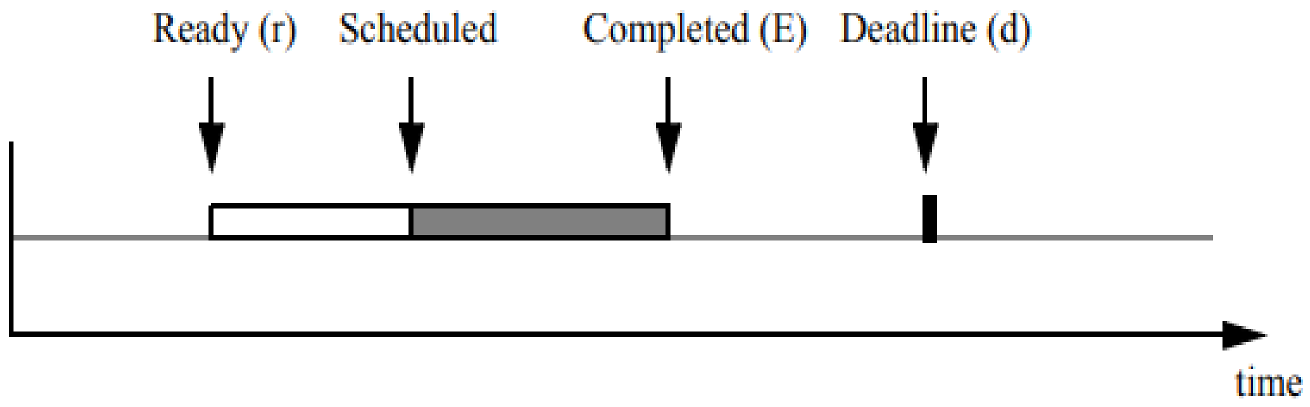
Terms and definition

- **Release time (or ready time):** This is the time instant at which a task(process) is ready or eligible for execution
- **Schedule Time:** This is the time instant when a task gets its chance to execute
- **Completion time:** This is the time instant when task completes its execution
- **Deadline:** This is the instant of time by which the execution of task should get completed.
- **Runtime:** The time taken without interruption to complete the task, after the task is released

Terms and definition

- **Tardiness:** amount of time by which task misses its deadline
- **Laxity** : deadline minus the remaining computation time. Max time task can wait and still meet its deadline.

Schematic of a Time Constrained Computation



Types of Real Time Systems

- **Hard real time systems**
  - Must always meet all deadlines
  - System fails if deadline window is missed
- Hard Real time task:
  - Task must complete before or at deadline.
  - Value of completing the task after deadline is zero.

  E.g. automobile braking system
    - Air traffic control
    - Vehicle subsystems control
    - Nuclear power plant control

Soft Real Time System

- **Soft real time systems**
  - Must try to meet all deadlines
  - System does not fail if a few deadlines are missed
- Software Real Time
  - Missing deadline is penalty
  - Penalty increases as tardiness increase

E.g. Weather stations have many sensors for reading temperature, humidity, wind speed, etc. The readings should be taken and transmitted at regular intervals, however the sensors are not synchronized. Even though a sensor reading may be early or late compared with the others it can still be relevant as long as it is close enough.

Soft Real Time System

- Example:
- Imagine a game in which scores of the players are displayed continuously.
- If scores are displayed late , no catastrophe occurs, but decisions have to be based on scores, if scores are outputted beyond deadline, we can say that performance of scoring system is bad.

Firm real time system

- **Firm real time systems**
  - Result has no use outside deadline window
  - Tasks that fail are discarded

Firm real time task

- Its value reduces to zero if deadline is not met
- Output of task is discarded if deadline is not met.
- Output of delayed execution is dropped.
- Deadlines are allowed to be missed once in a while , but not too frequently

E.g. decoding of video frame may occasionally get delayed by reasons like unexpected interrupts and the like. when such frames are delivered late, the playback does not look good. Skipping such frames and making sure that not too much such frames are delayed will be better. Because effect will be less noticeable to user than playing back the delayed frames.

Real time tasks

- **Periodic**
  - Tasks which are activated regularly at fixed rate
  - Periodic tasks must execute once per period
- Each task is repeated at a regular interval
- Max execution time is the same each period
- Arrival time is usually the start of the period
- Deadline is usually the end

Real time tasks

- **Aperiodic**

➢ Stream of jobs arriving at irregular intervals

  ➢ Each task can arrive at any time

- Inter-arrival period between two such tasks can be zero.
- Aperiodic tasks implied to have soft deadlines and aim of scheduling is to provide fast response time.

Real time tasks

- Sporadic is a an aperiodic task with hard deadline and a minimum inter-arrival time.
- Without minimum inter-arrival time restriction, it is impossible to guarantee that a deadline of sporadic task would always be met.
- E.g. such tasks are emergency conditions like fire

Preemptive and non-Preemptive tasks

- Preemptive task:

Task which can be preempted if another task of higher priority becomes ready.

- Non- preemptive:

execution of non-preemptive task should be without interruption, once started

Preemptive task

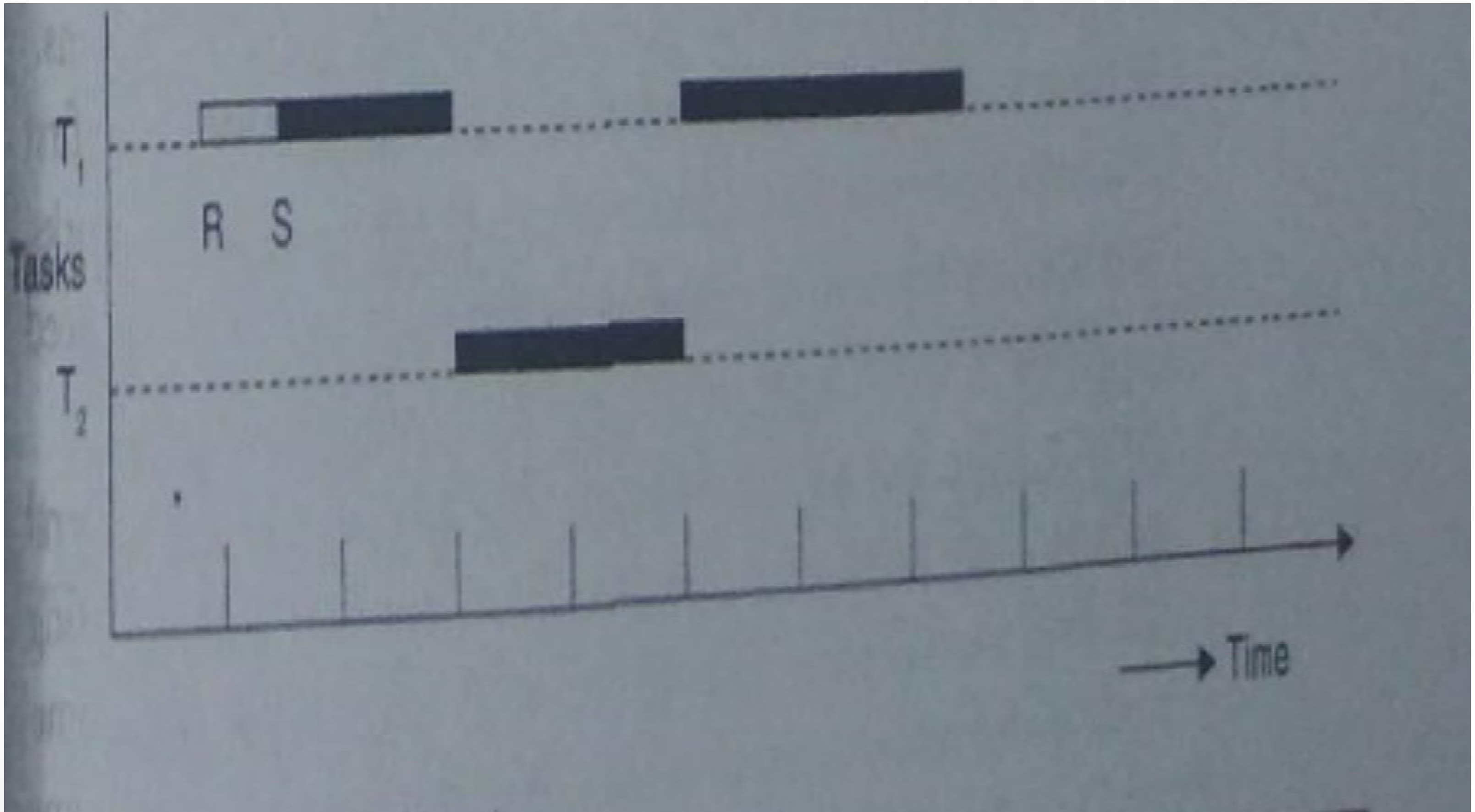


Figure 8.6 | A preemptible task

Non-preemptive task

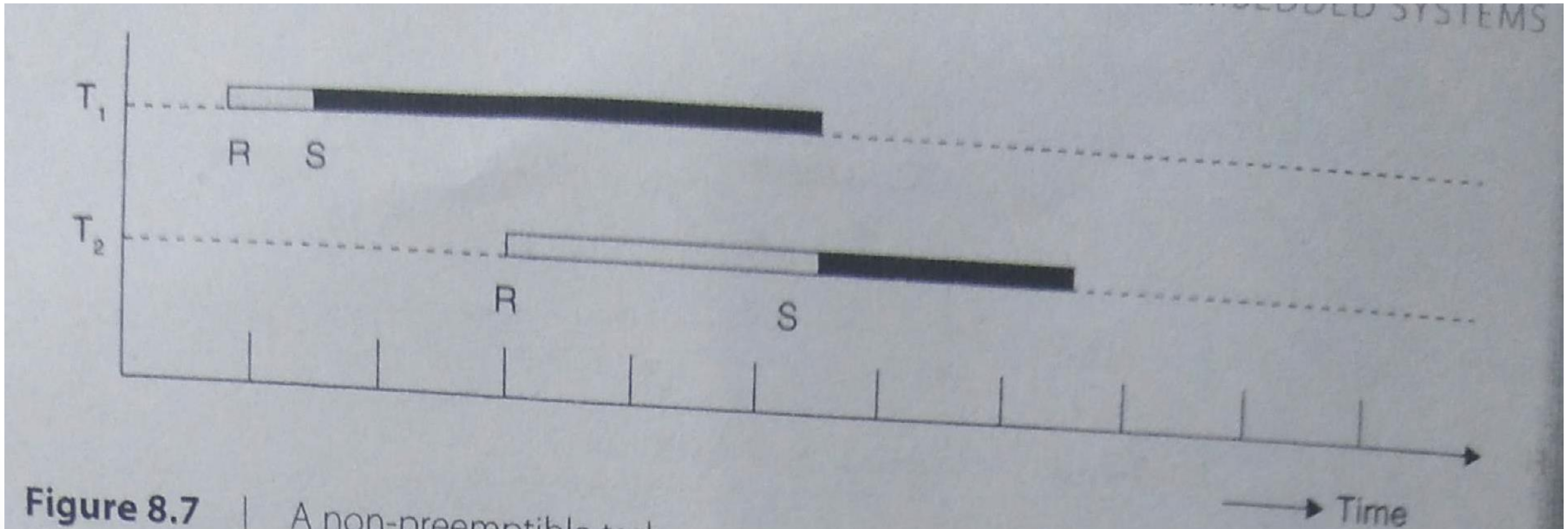


Figure 8.7 | A non-preemptible task

Real Time Operating System

- Some embedded applications require only dedicated hardware and firmware e.g. mp3 player, printer and scanner etc.
- Such system are called superloop system
- Whole code for systems is written in one loop which executes continuously.
- When external event come, interrupts are generated to alert the processor and system respond appropriately.
- There can be no. of inputs and corresponding actuators too.
- code for all these is in flash memory.

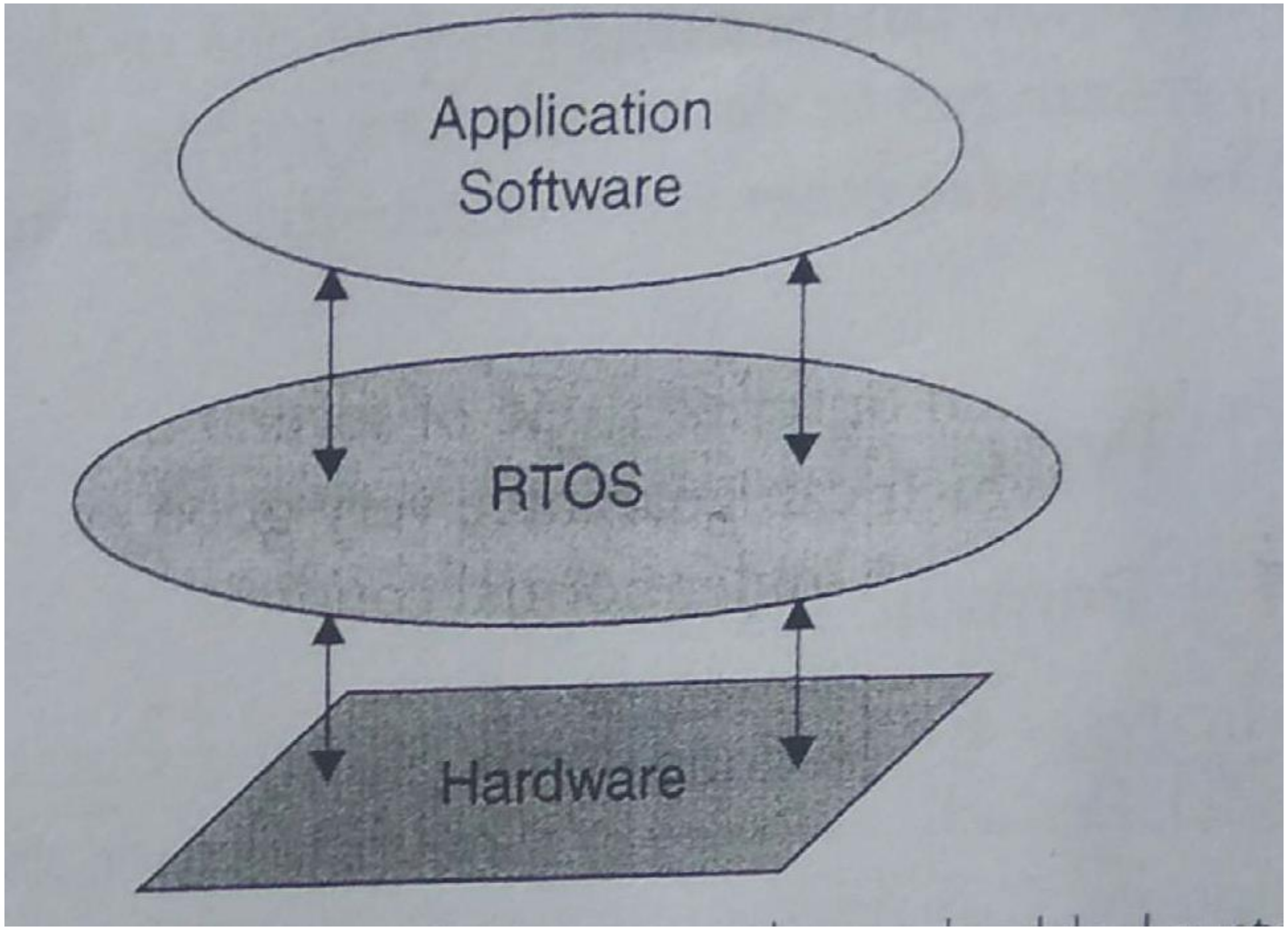
Real Time Operating System

- Some embedded system like mobile phone need a manager
- all embedded system need not real –time operating system. Only where time constraint is a factor real-time OS is required

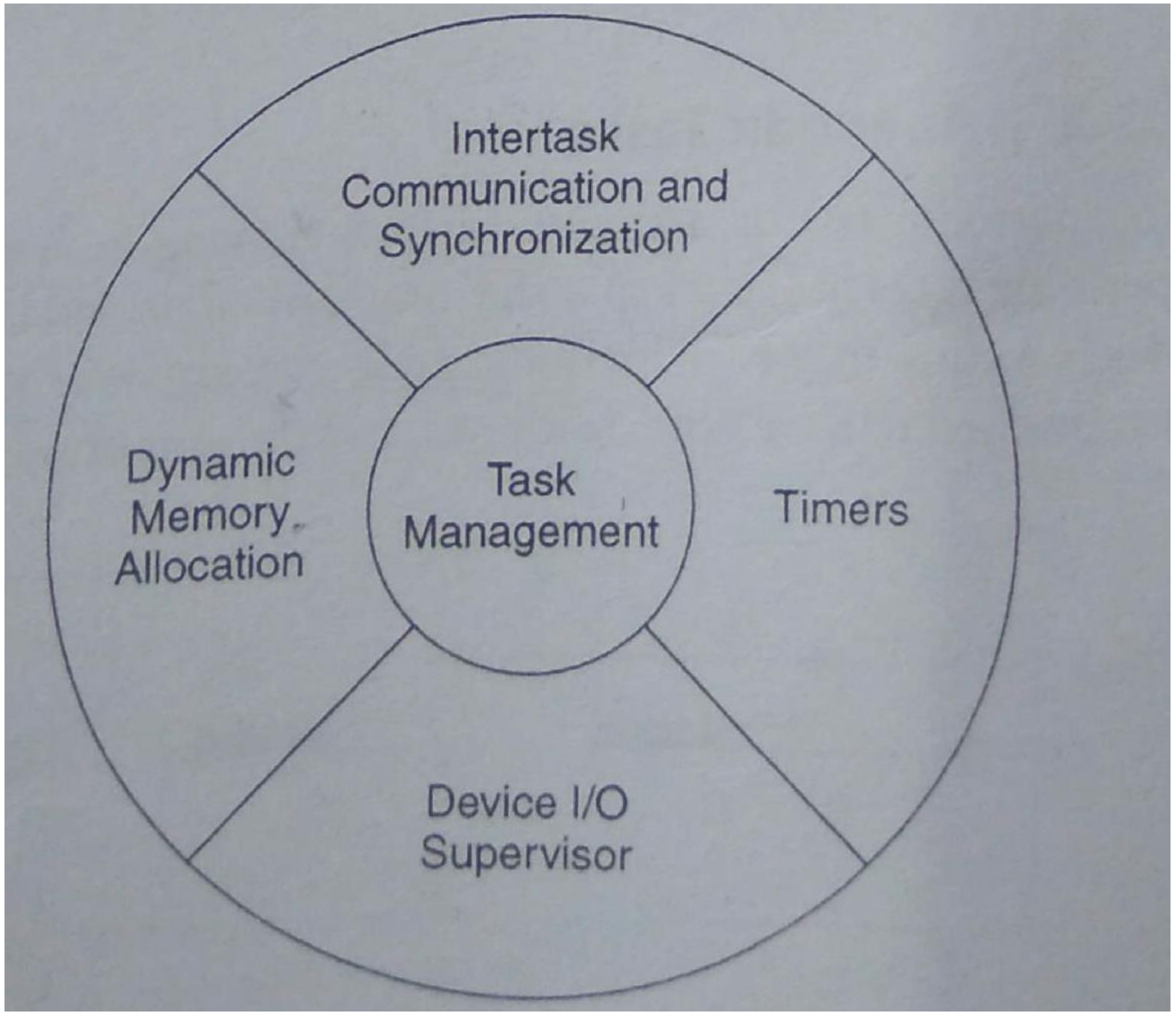
What does RTOS does?

- RTOS provides abstraction layer between embedded hardware and application software
- RTOS manages interaction between hardware and applications.
- RTOS ensures that the multiple tasks that comes in are managed and done on time.
- OS has a kernel which form core of OS.

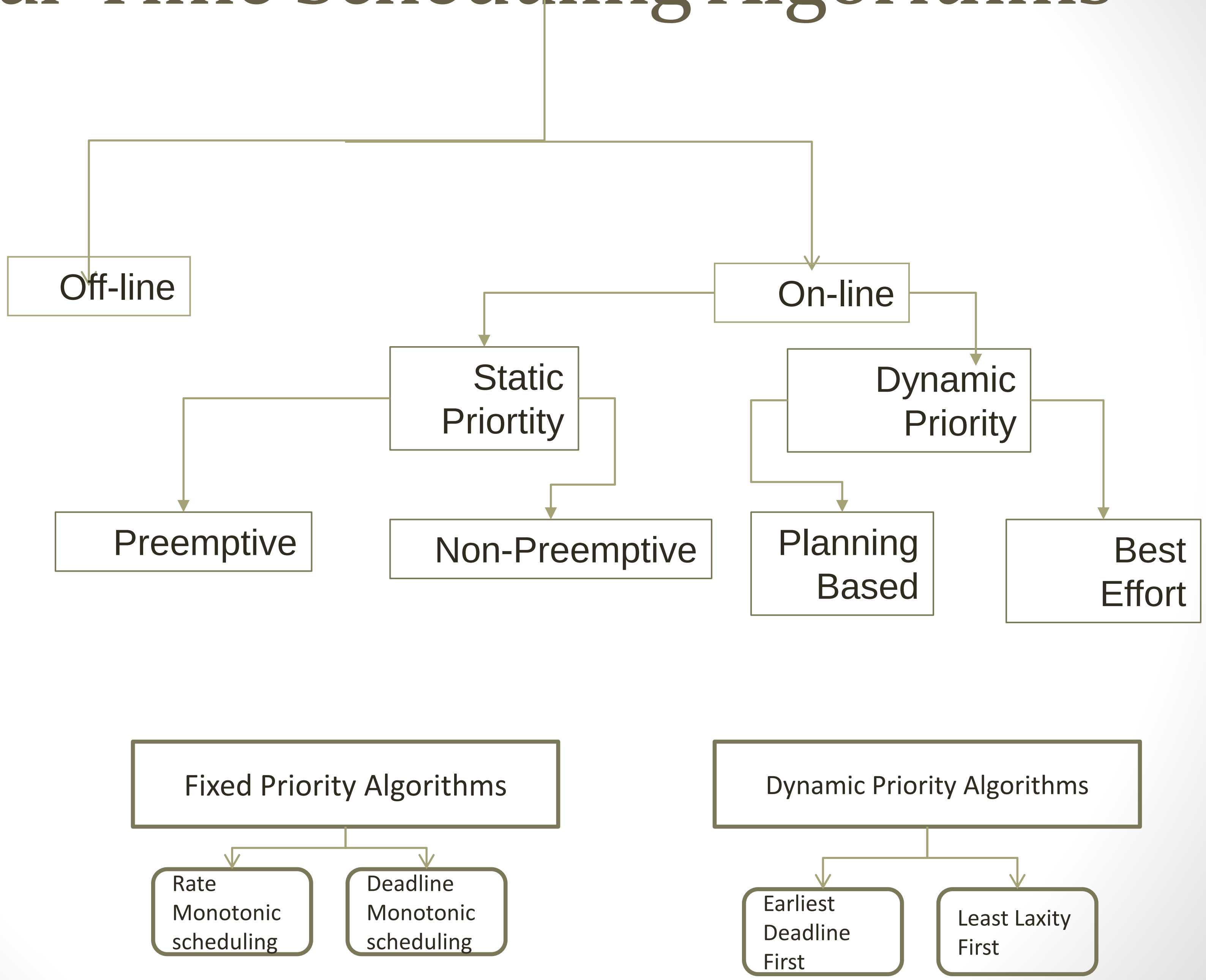
Hardware- Software hierarchy in complex embedded system



Kernel services of RTOS


Intertask Communication and Synchronization
Dynamic Memory Allocation
Task Management
Timers
Device I/O Supervisor

Real-Time Scheduling Algorithms



Off Line Scheduling (Pre Run time scheduling)

- They generate scheduling information prior to system execution (Deterministic System Model)
- This scheduling is based on :
  - Release time
  - Deadlines
  - Execution

  Scheduling algo can use precise schedule which optimizes several different measures and optimal algo can be used which guarantee very good system performance.

  E.g. Fixed factory jobs where nothing changes under normal conditions can use this approach
- Disadvantage: Inflexibility, If any parameter changes, the policy will have to be redone

On-Line Scheduling

- Number and types of tasks, associated parameters are not known in advance.
- Scheduling must accommodate
- dynamic changes in user demands
- Availability of resources

- Online Scheduling are of two types:
  - Static Priority
  - Dynamic Priority

Static Priority

- Tasks with highest priority gets the chance to execute first

- This can be preemptive or Non-preemptive
- **Non preemptive**: task with highest priority runs till it completes.
- **Preemptive**: The execution of task can be pre-empted when higher priority task appears in ready queue.

Dynamic Priority

- Priority can be allowed to change at run time
- Scheduling needs more computation
- Pre-emption may or may not be used
- Flexibility is quite high in such system
- Two subsets:

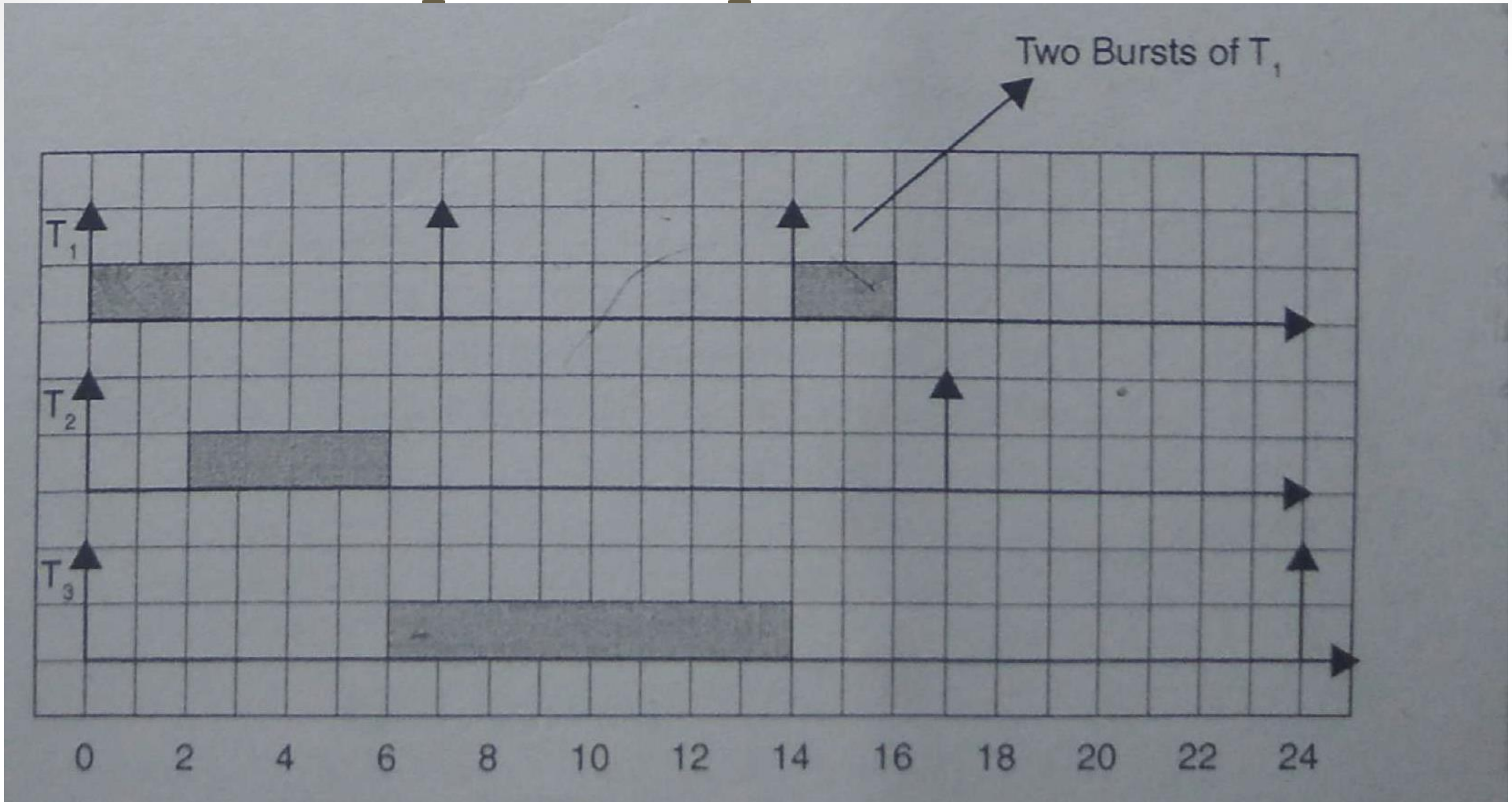
1. **Planning Based**:
    - Guarantees deadline for all accepted task.
2. **Best effort algorithm** does its best to maximize performance.
    - Guarantees meeting deadline for hard time task.
    - Optimizes the performance of soft time task

Problem: Static Priority

Tasks	Priority	Period	CPU Burst
T1	1	7	2
T2	2	17	4
T3	3	24	8

- Draw the Gantt-chart for scheduling these tasks.
- Assume the all jobs have same release time
- Schedule the process to meet their deadline
  - Without Pre-emption
  - With Pre-emption
- Note: Deadline of task is the time when its next burst arrive

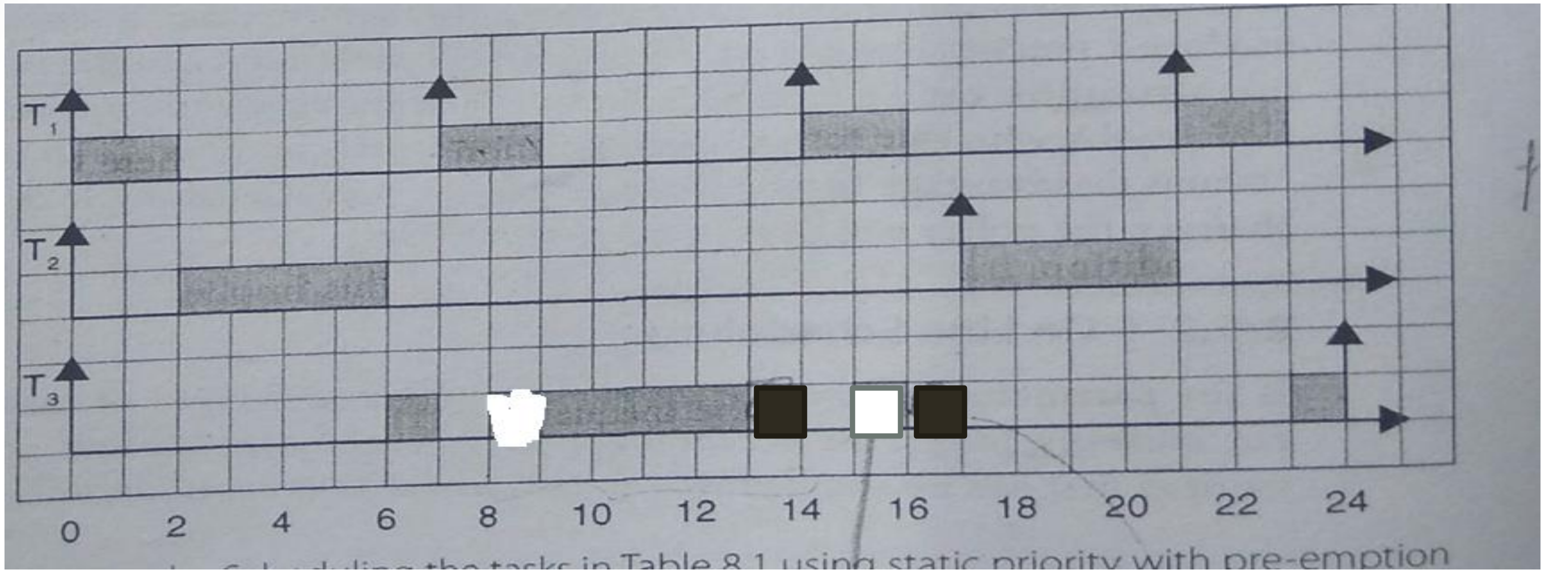
Without preemption



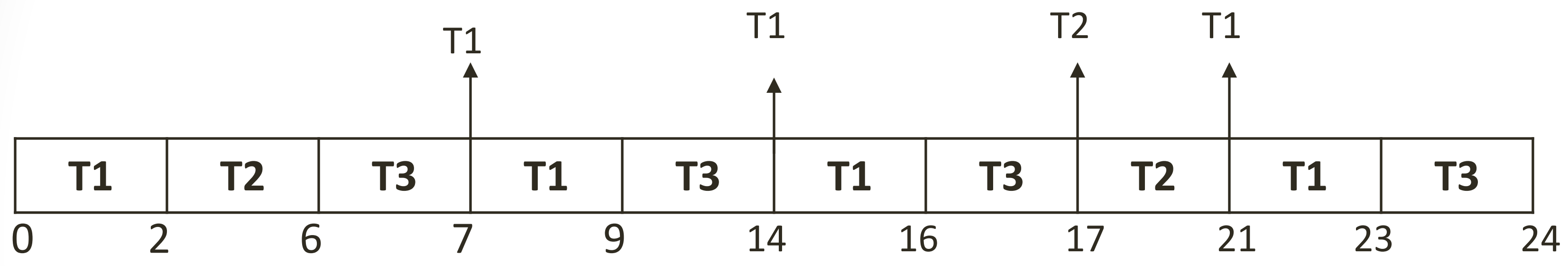
8.11 | Scheduling the tasks in Table 8.1 using static priority without pre-emption

Tasks	Priority	Period	CPU Burst
T1	1	7	2
T2	2	17	4
T3	3	24	8

With Preemption



Tasks	Priority	Period	CPU Burst
T1	1	7	2
T2	2	17	4
T3	3	24	8



Tasks	Priority	Period	CPU Burst
T1	1	7	2
T2	2	17	4
T3	3	24	8

Rate Monotonic Algorithm

- Introduced by Liu and Layland in 1973
- Assigning priorities as a monotonic function of the rate of a (periodic) process.
- (Monotonic means either increasing or decreasing)
- Priorities are assigned according to increased period of a process.
- **As period increases the priority decreases**.
- The process with lowest period will have highest priority
- A set of tasks is said to be schedulable if all of the tasks can meet their deadlines.

RM Algorithm

- RM provides simple inequality, to verify the sufficient condition for RM algorithm

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

- LHS of inequality is total CPU utilization for n tasks
- Where C is CPU Burst , P or T is Period of task
- If the condition satisfied, RM will schedule tasks within their respective deadline

- This is sufficient, but not necessary condition i.e. Set of task which satisfy this condition will definitely be schedulable, but there can be set of tasks which do not satisfy this condition, and still are schedulable.

RHS of expression

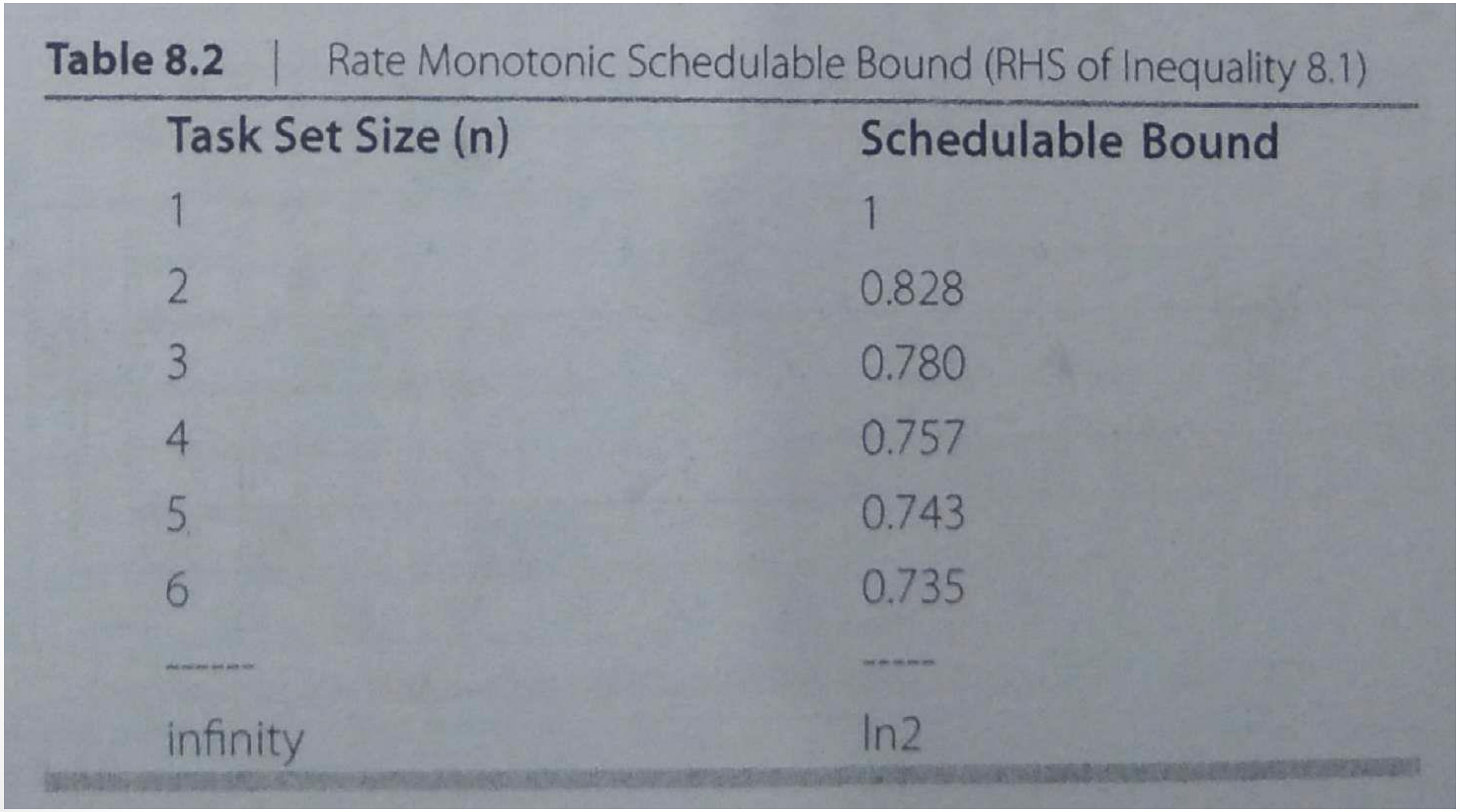


Table 8.2 | Rate Monotonic Schedulable Bound (RHS of Inequality 8.1)

Task Set Size (n)	Schedulable Bound
1	1
2	0.828
3	0.780
4	0.757
5	0.743
6	0.735
------	-----
infinity	In2

RMA algo

- If a task set passes the Liu and Layland test, then it is guaranteed to be RMA schedulable. On the other hand, even if a task set fails the Liu and Layland test, it may still be RMA schedulable.

Example 1

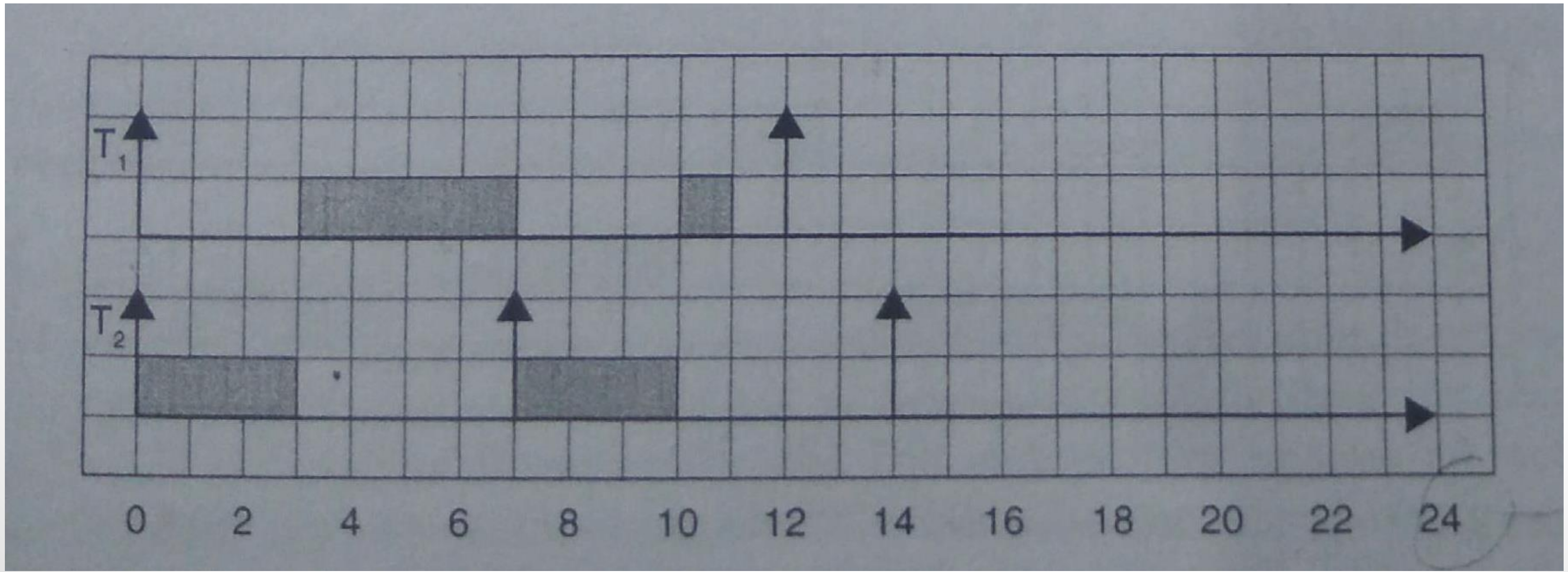
Tasks	Priority	Period (T)	CPU Burst(C)
T1	1	7	2
T2	2	17	4
T3	3	24	8

- CPU utilization for the set of task:
- LHS : 2/7 + 4/17 + 8/24 = 0.812
- RHS : = 3 X ($2^{1/3}$ -1)= 3 x .26 = .78
- LHS> RHS   :RM does not get satisfied
- **But the task still might get schedulable by RM technique**
- In last example Static priority with pre-emption was implementation of RM algorithm because priorities are ordered in decreasing order of their priorities

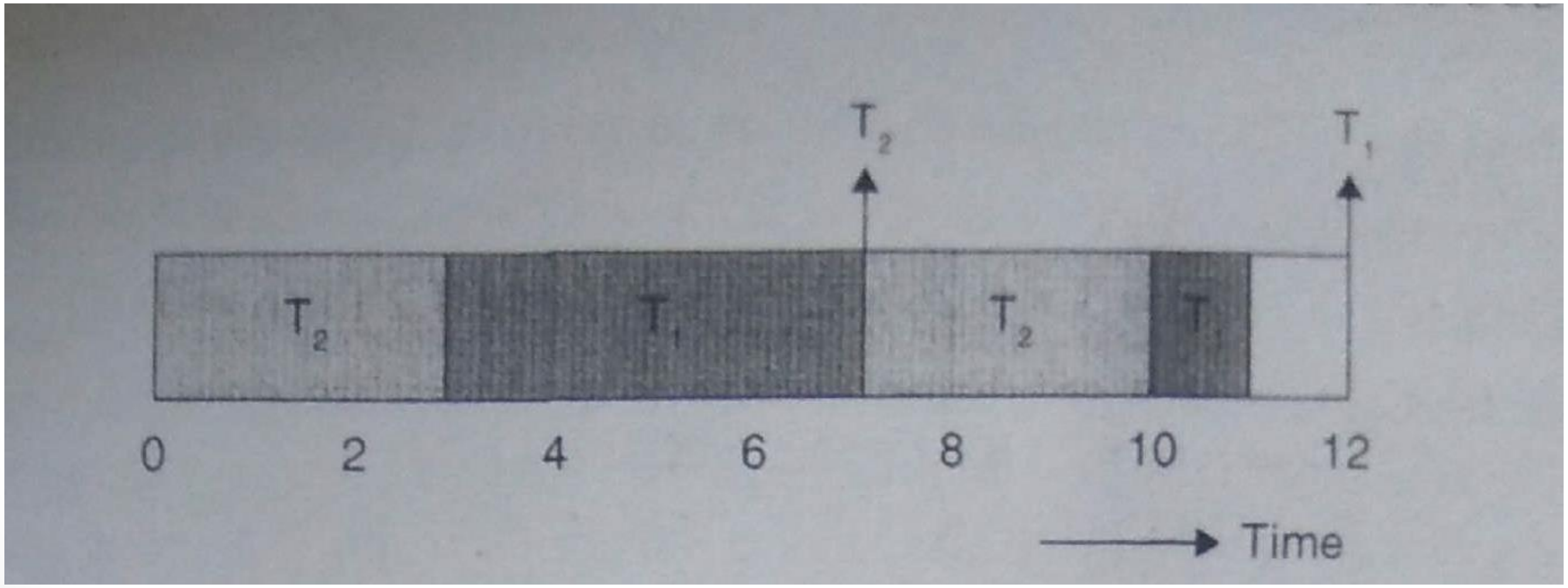
Example2

Tasks	Period	CPU Burst
T1	12	5
T2	7	3

- CPU utilization= 5/12+3/7=0.845
- For two tasks RHS of inequality is 0.828
- Since LHS>RHS sufficient condition for schedulability is not satisfied.



Example2



Example2

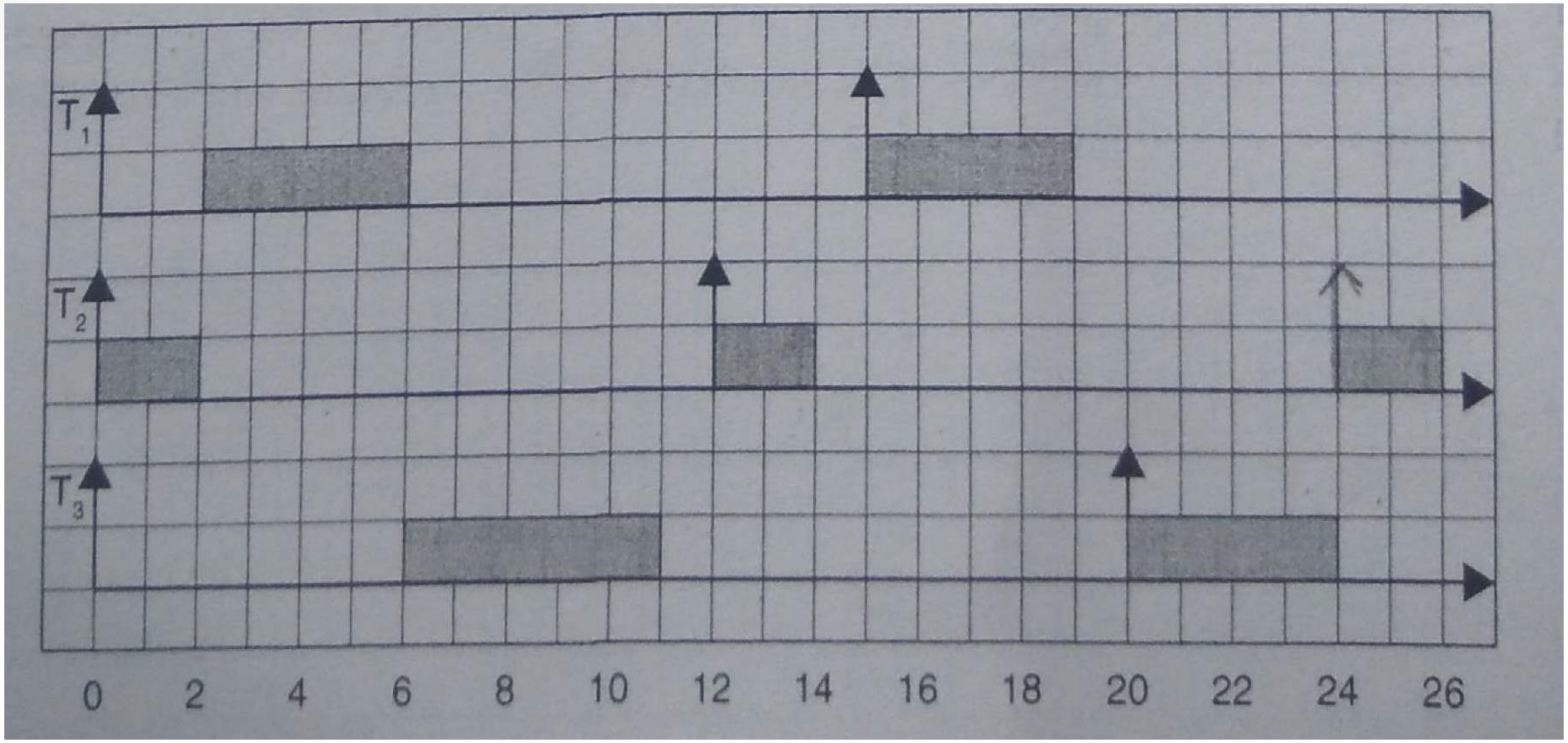
- Both tasks able to finish before their deadline, but processor is idle for certain periods

Example3

Tasks	Period (P or T)	CPU Burst(C)
T1	15	4
T2	12	2
T3	20	5

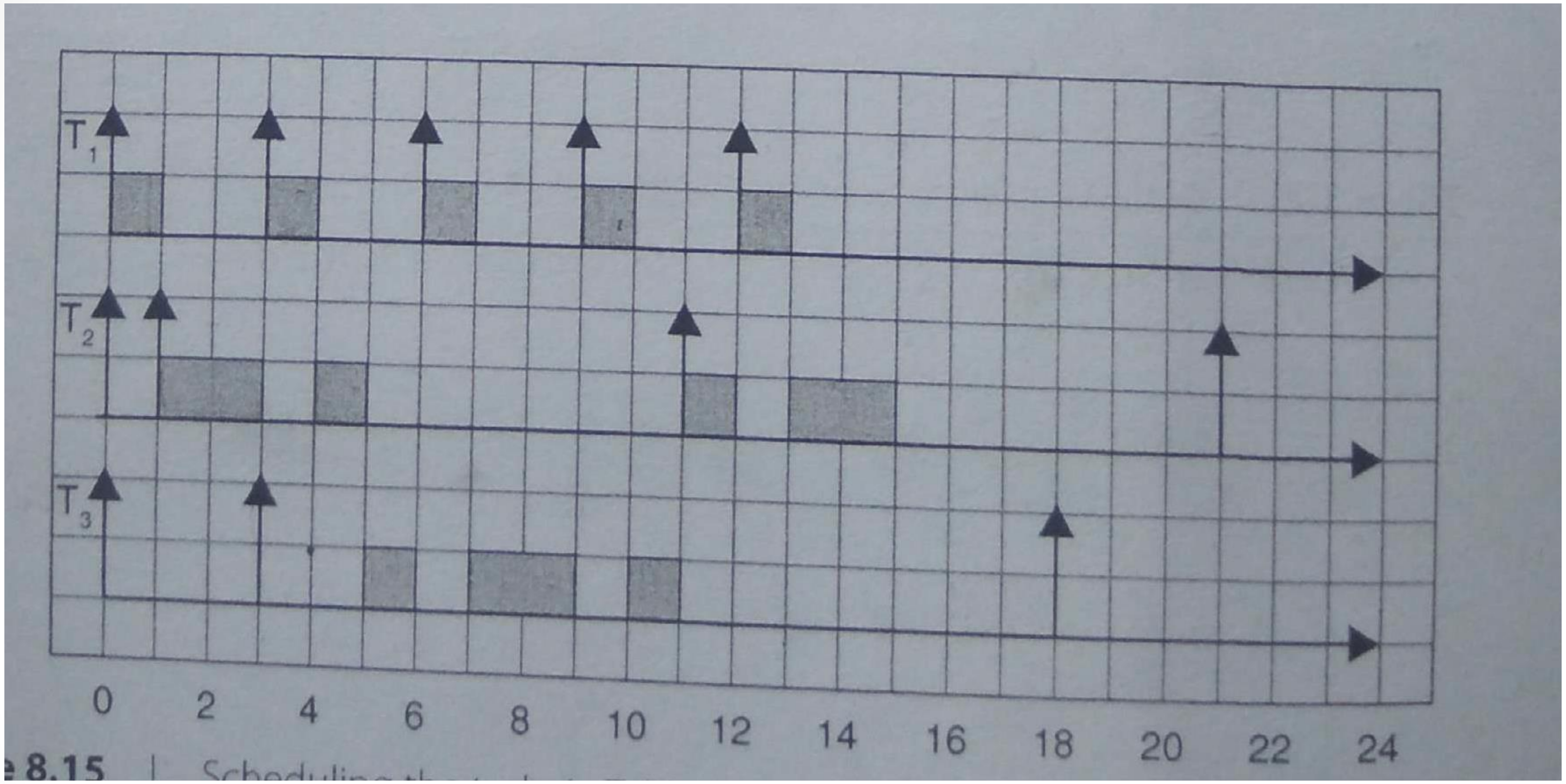
- CPU utilization =4/15+2/12+5/20=0.684
- Bound for scheduling for three tasks is 0.782
- Since LHS <RHS
- Sufficient condition is satisfied, this task set is definitely schedulable using RM algorithm

Example 3



Tasks	Period (P or T)	CPU Burst(C)
T1	15	4
T2	12	2
T3	20	5

Example 4



Tasks	Period	CPU Burst	Release Time
T1	3	1	0
T2	10	3	1
T3	15	4	3

Earliest deadline First

- Belong to class of dynamic priority allocation method
- Priority of task changes at run time
- At any instant, priority task is one that has the closest deadline.
- Tasks that cannot be scheduled by RM can be scheduled by this method.

Necessary condition for Schedulability:

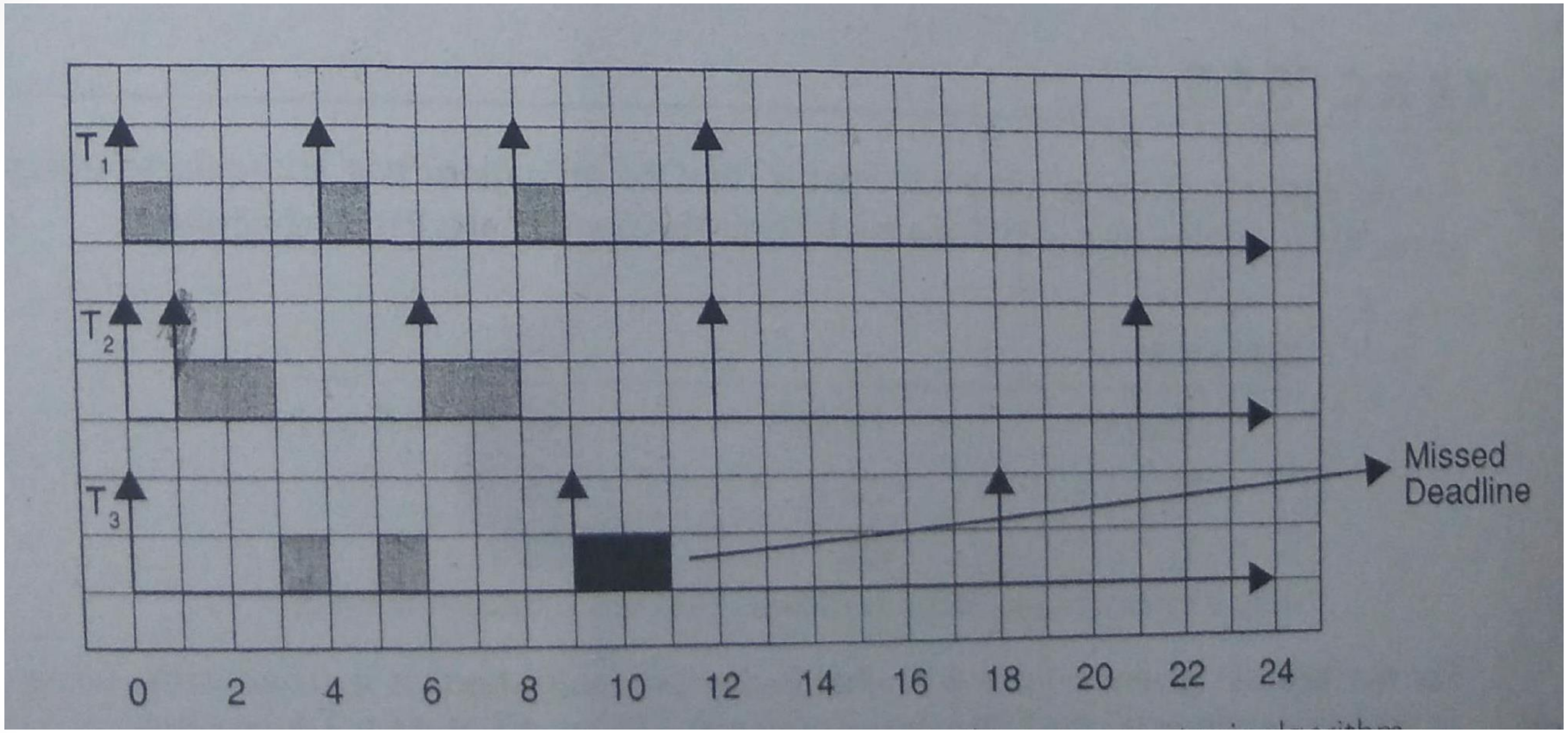
CPU utilization must be less than 1.

example

Tasks	Period	CPU Burst
T1	4	1
T2	6	2
T3	9	4

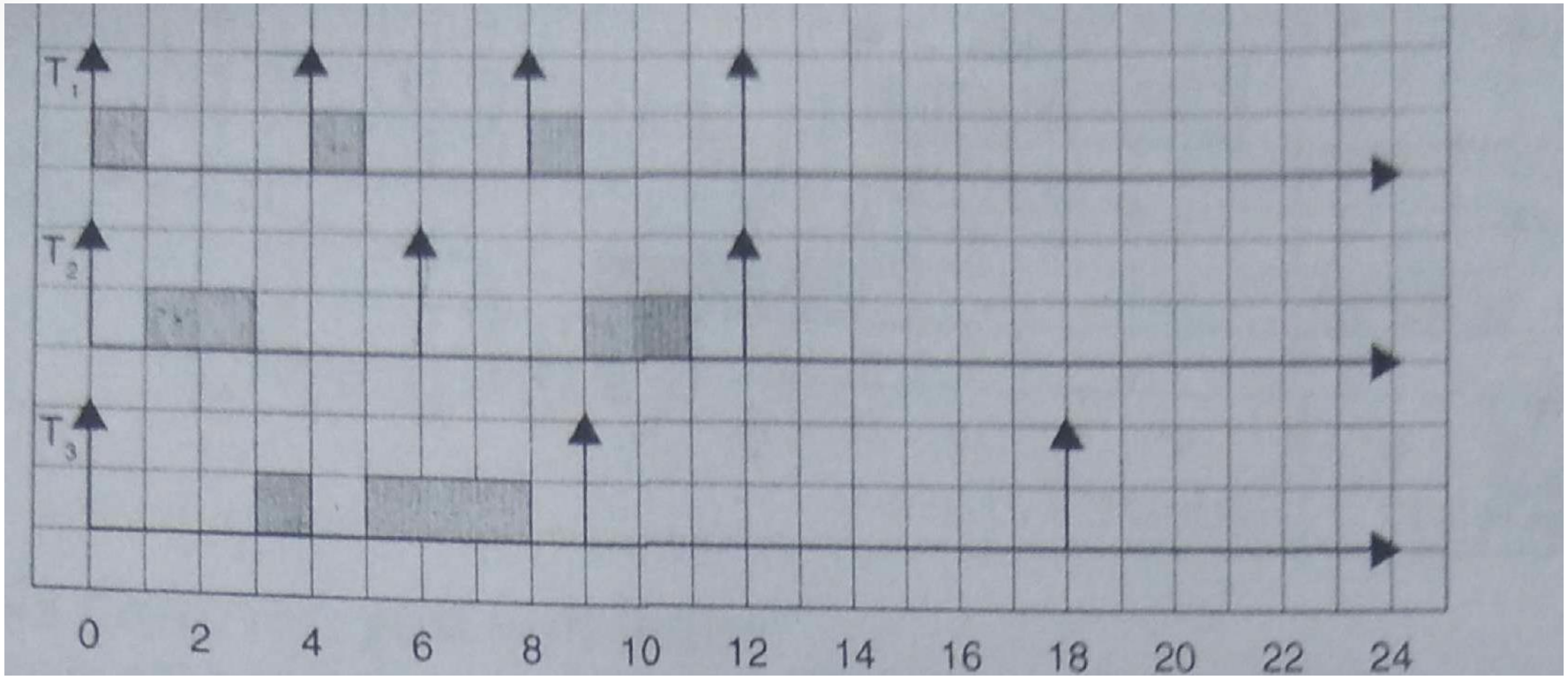
- CPU utilization=1/4+2/6+4/9=1.027

Using RM technique



Tasks	Period	CPU Burst
T1	4	1
T2	6	2
T3	9	4

Using EDF



Tasks	Period	CPU Burst
T1	8-12	1
T2	12	2
T3	9-18	4

Example 2

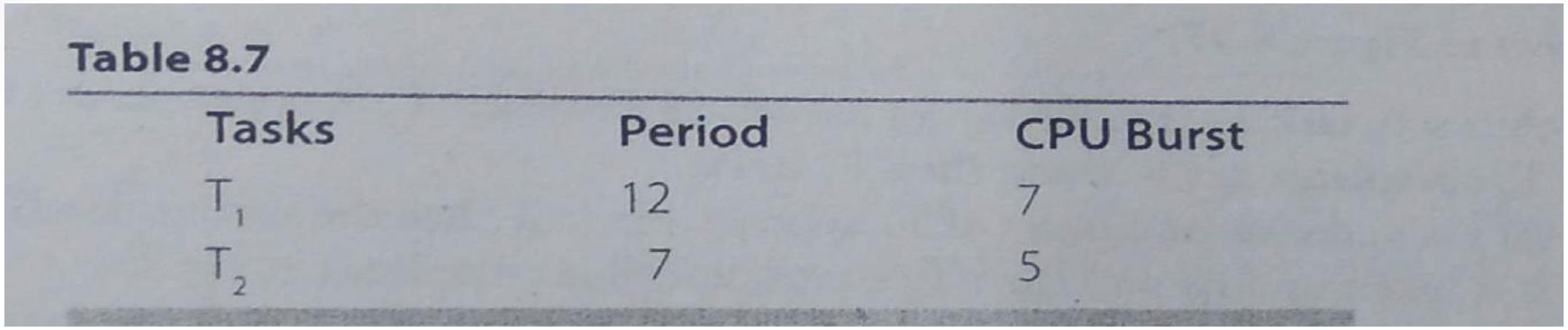
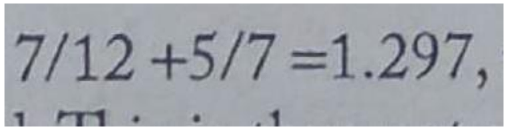


Table 8.7

Tasks	Period	CPU Burst
T_1	12	7
T_2	7	5



$7/12 + 5/7 = 1.297,$

EMBEDDED SYSTEMS

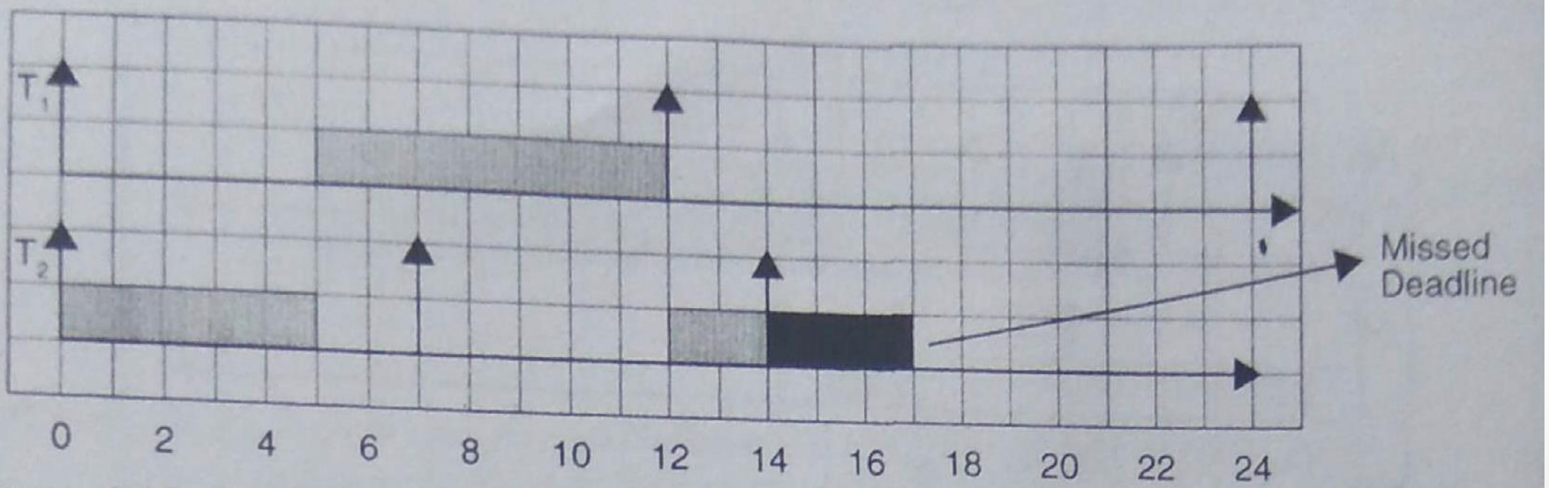


Figure 8.18 |

Laxity

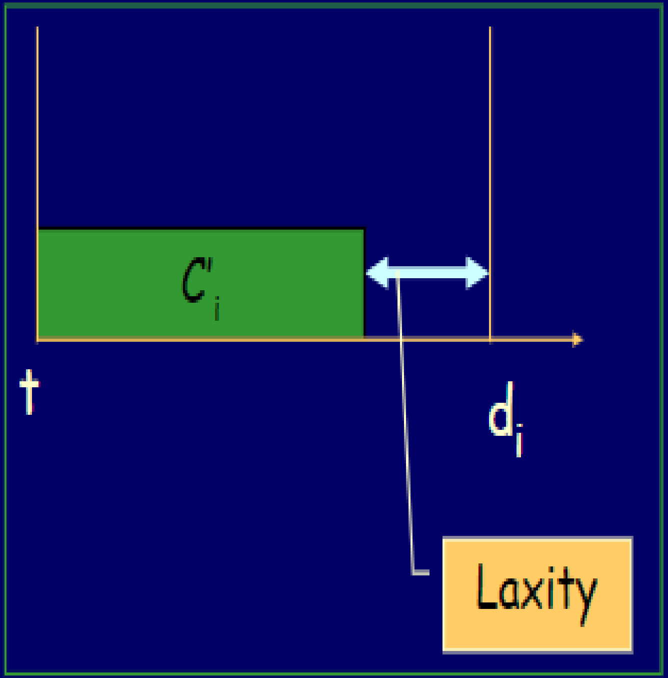
- Tasks are preemptable

- Laxity of a Task
- $Ti = d_i - (t + c_i')$

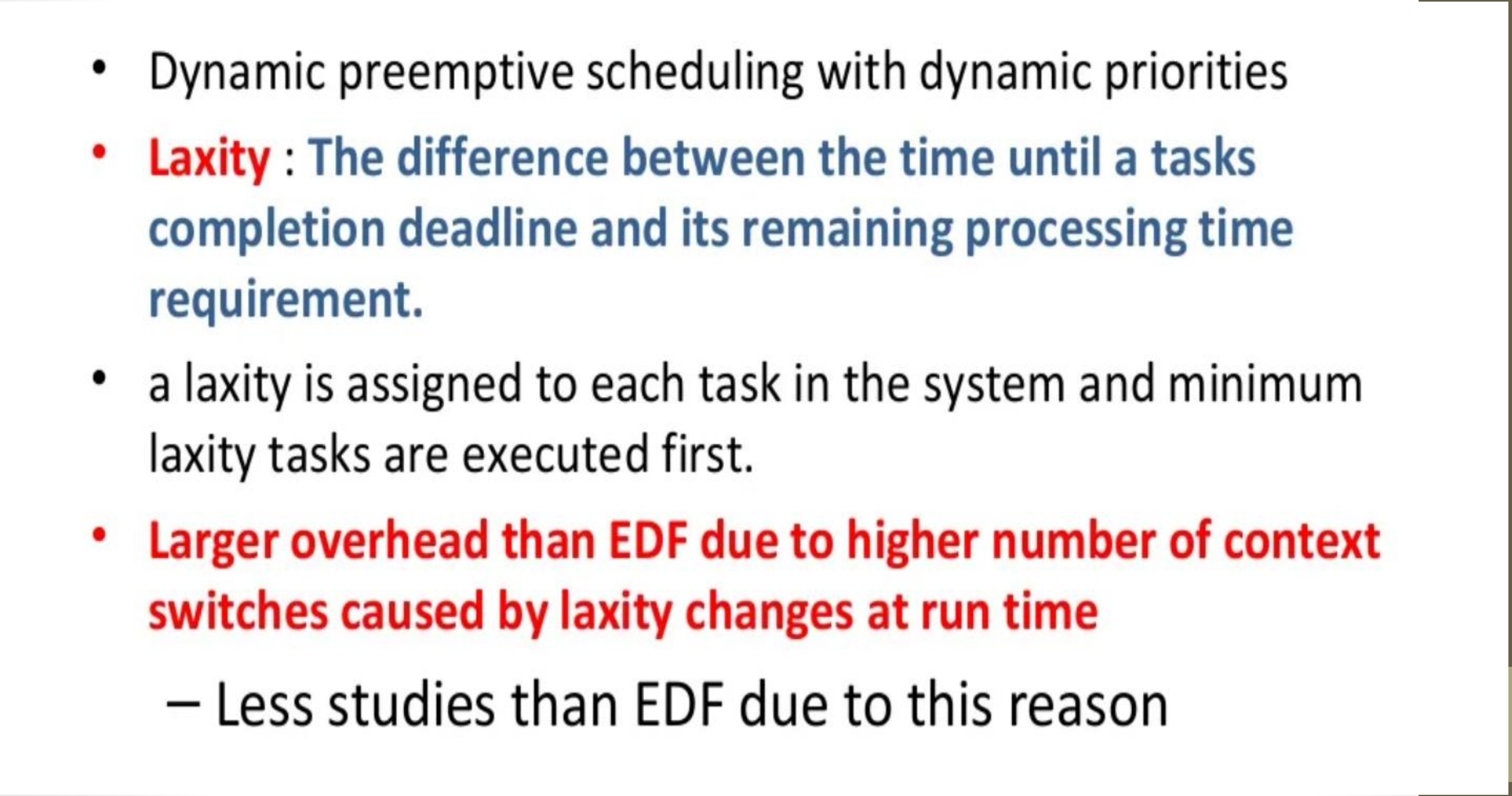
  where di: deadline;

  t : current time;

  c_i' : remaining computation time.



Least Laxity First Scheduling



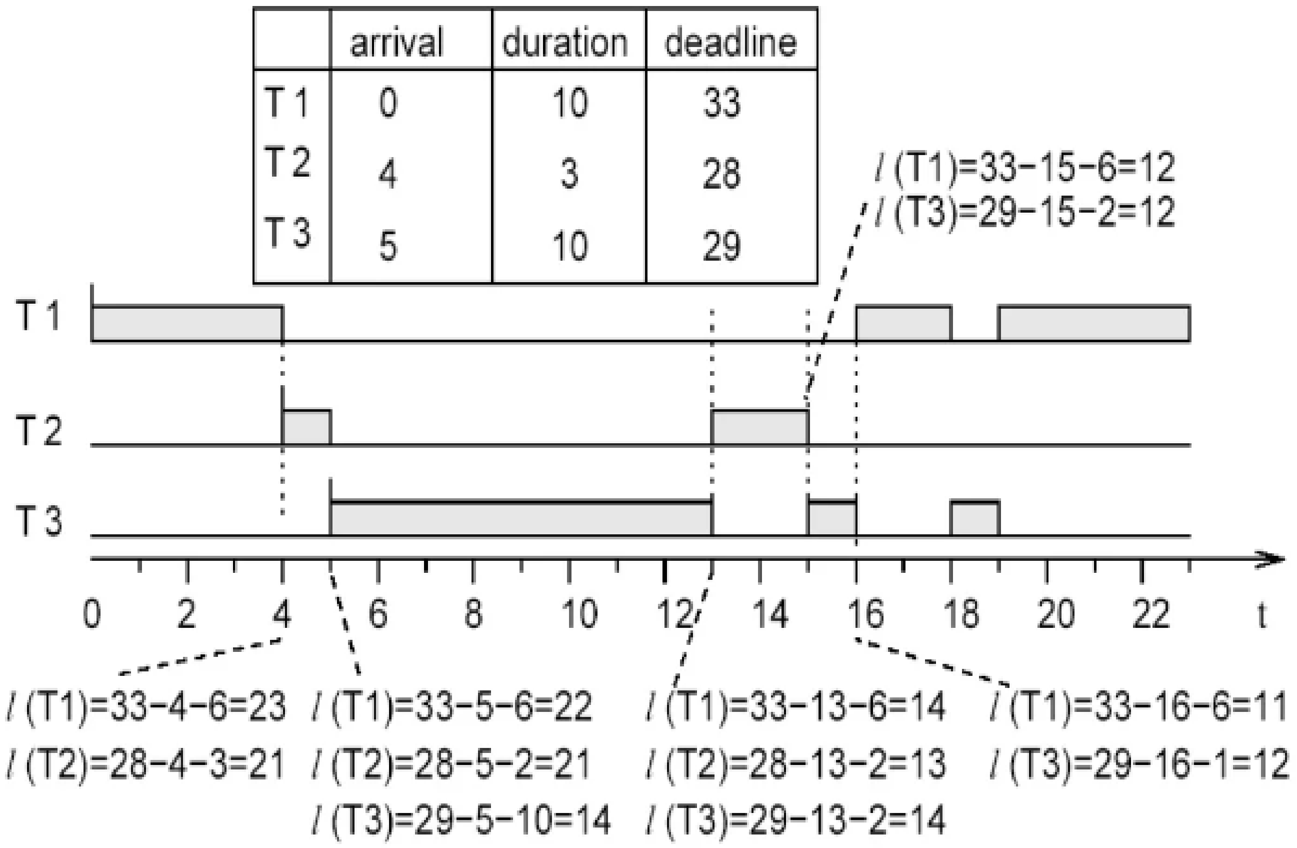
- Dynamic preemptive scheduling with dynamic priorities
- **Laxity** : **The difference between the time until a tasks completion deadline and its remaining processing time requirement.**
- a laxity is assigned to each task in the system and minimum laxity tasks are executed first.
- **Larger overhead than EDF due to higher number of context switches caused by laxity changes at run time**
  - Less studies than EDF due to this reason

- LLF considers the execution time of a task, which EDF does not.
- LLF assigns higher priority to a task with the least laxity.
- A task with zero laxity must be scheduled right away and executed without preemption or it will fail to meet its deadline.
- **The negative laxity indicates that the task will miss the deadline**, no matter when it is picked up for execution.

The **definition** of **laxity** is the quality or condition of being loose. Being lose and relaxed in the enforcement of a procedure is an example of **laxity**.

Example

Least Laxity First Example



	arrival	duration	deadline
T 1	0	10	33
T 2	4	3	28
T 3	5	10	29

Disadvantages of EDF

- It is dynamic priority algorithm and therefore requires dynamic determination of priorities.
- It is not as controllable as static priority algorithms.
- More overheads are required to implement this
- Are not usually used in systems which require absolute predictability
- Slightly greater overheads than fixed priority
- Dynamic priorities must also be calculated at each decision point, where as static never change and never have to be recalculated.

Qualities of RTOS

- **Performance**: fast and high throughput
- **Reliability**: should not fail or mean time between failure very high
- **Compactness**: software and OS ported to memory , size of OS as small as possible
- **Scalability:** unnecessary services are removed, OS must be scale down. Modularity in design of OS, to allow to be scaled down or up , as necessary.