Spring 2009 - Real-Time Systems

http://www.neu-rtes.org/courses/spring2009/

Chapter 4
Priority-Based Real-Time Scheduling

Real-Time Embedded Systems Laboratory
Northeastern University

Objectives

- In this chapter, you are supposed to learn:
  - The basic principle behind priority-based scheduling
  - What is Rate Monotonic (RM) scheduling, how it works, and its characteristics
  - Deadline Monotonic (DM) scheduling as an extension to RM
  - What is Earliest Deadline First (EDF) scheduling, how it works, and its characteristics

Contents

The Scheduling Problem

- Deciding the order and/or the execution time of a set of

Motivation for Scheduling

- In the old days, each control task runs on a dedicated CPU
  - No RTOS, bare metal
  - No need for scheduling
  - Just make sure that task execution time < deadline
- Now, multiple control tasks share one CPU
  - Multitasking RTOS
  - Need scheduling to make sure all tasks meet deadlines
- So resource sharing is the root for scheduling, both in GPOS and in RTOS

What Does Scheduling Mean to GPOS?

- In GPOS, the system is scheduled with the objective to …
  - User-Oriented objectives
    - Low Response Time
    - Low Turnaround Time
    - High Stability
  - System-Oriented
    - High Throughput
    - High Processor Utilization
    - Fairness
    - Balancing Resources

Do GPOS Scheduling suffice in RTS?

- GPOS Scheduling certainly does not fit to RTS, because
  - None of the takes deadlines into consideration
  - Scheduling objectives may be contradictory to real-time timing constraints
    - Preemptivity
    - Predictability
    - ……

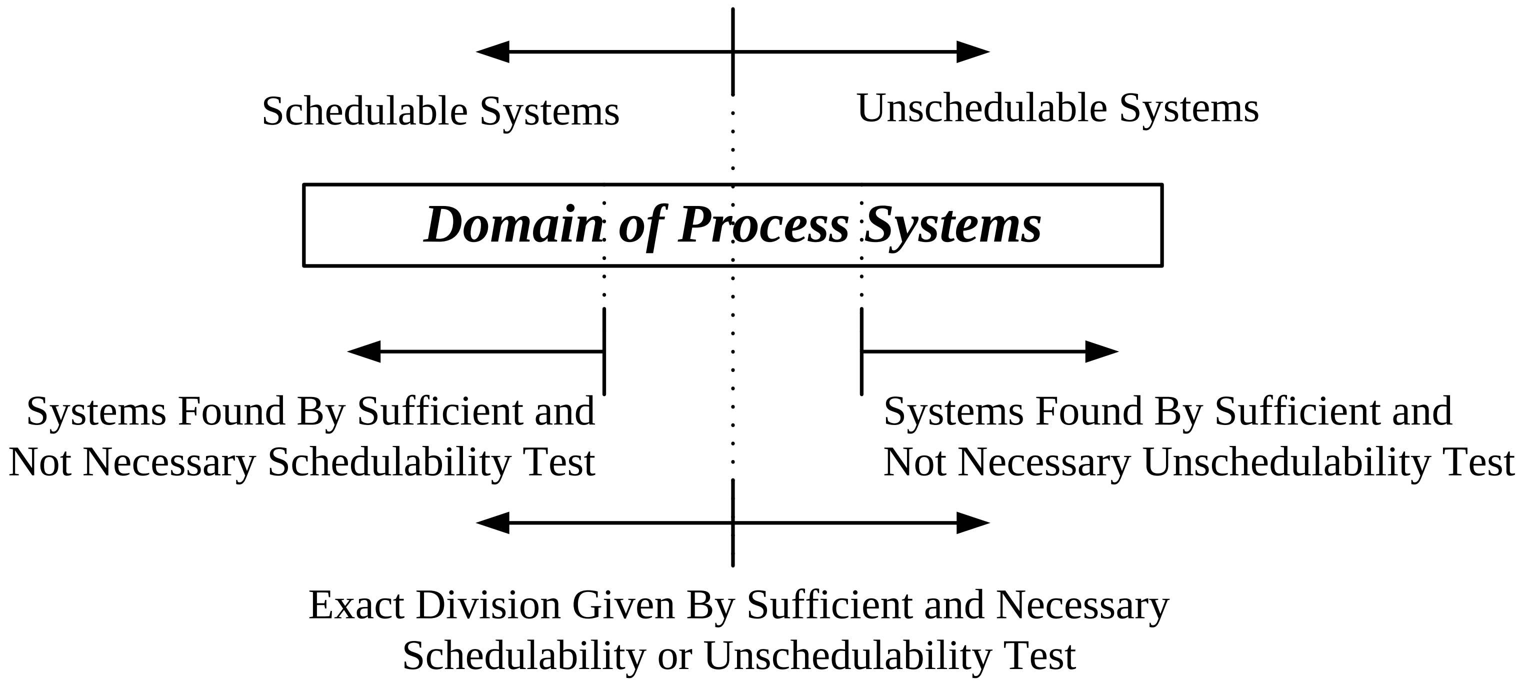
What Does Real-Time Scheduling Require?

- Satisfying deadline requirements of tasks, especially hard real-time tasks
- Average response time
- Total completion time
- Maximum lateness
- Predictability of an algorithm under transient overload
- Minimizing miss ratio
- ……

Schedule and Schedulability Test

- Feasible Schedule and Optimal Schedule
  - A schedule is a feasible schedule if all the jobs complete before their deadlines, with their execution sequence dictated by the schedule
  - A set of jobs is *schedulable* according to a scheduling algorithm if the scheduler that implements the algorithm always produce a feasible schedule
  - For any given set of tasks that is schedulable in reality, if a scheduling algorithm can always find a feasible schedule for the task set, then we say this algorithm is *optimal*
- Schedulability Test
  - An algorithm or a process that can validate whether a given task set is schedulable or not

Optimality of the Scheduling Algorithm


Schedulable Systems
Unschedulable Systems
Domain of Process Systems
Systems Found By Sufficient and Not Necessary Schedulability Test
Systems Found By Sufficient and Not Necessary Unschedulability Test
Exact Division Given By Sufficient and Necessary Schedulability or Unschedulability Test

Taxonomy of RT Scheduling Algorithms

- Centralized V.S. Distributed
  - This dimension specifies whether the hardware architecture on which tasks are executed is a centralized one or a distributed one
  - Problems of distributed scheduling: task division, distribution schemes, thread migration, inter-sub-task synchronization, communication among tasks, resource allocation and access control
  - In later chapters, we assume that all algorithms are centralized, if no special remarks are made

Taxonomy of RT Scheduling Algorithms

- Preemptive V.S. Non-Preemptive
  - Preemptability describes whether a task with lower priority can be preempted by a task with higher priority, assuming that there is no resource contending between the two tasks
  - Real-time scheduling often require preemptive schemes
- Deterministic V.S. Best-Effort
  - Some tasks require a 100% guarantee of their deadline, and some may have soft deadlines
  - The intrinsic characteristics of tasks lead to different scheduling algorithms

Taxonomy of RT Scheduling Algorithms

- On-line V.S. Off-line
  - Off-line Algorithms
    - The schedule is pre-calculated before any task starts execution
    - The scheduler must have a complete knowledge of the system and all the metrics of tasks
    - High predictability, but less flexibility, large maintenance overhead
  - On-line Algorithms
    - Scheduling can occur both prior to task execution and in the process of system running
    - Even if there are scheduled tasks running, the scheduler may also calculate or revise the schedule since new tasks may arrive
    - Flexible
    - But Large runtime overhead, generally

Taxonomy of RT Scheduling Algorithms

- Fixed-Priority V.S. Dynamic-Priority
  - Tasks are assigned priority according to their periods. The priority of each task is fixed during execution

  - In dynamic priority scheduling, the priorities of the tasks may change during execution
  - Overhead in calculating new priorities for each task

The Case for Priority-Based Scheduling

- In the last lecture, we introduced
  - Static scheduling, tick-driven scheduling, …
- Why priority-based scheduling?
  - In real-time systems, not all tasks are created equal!
  - Serve the tasks according to their urgency, and their criticality to the system, urgent or critical tasks must be satisfied first, even if this may sacrifice less urgent tasks
  - How do you assign priorities? What does an assignment imply? → different scheduling policies

Contents

Rate Monotonic Scheduling (RM)

- A fixed priority scheduling algorithm for periodic task sets
- Assumptions
  - A1: All tasks with hard deadlines are periodic
  - A2: The relative deadline of task equals to its period
  - A3: Tasks are independent of each other
  - A4: WCET of the tasks is known in advance
  - A5: All aperiodic tasks are not hard real-time ones

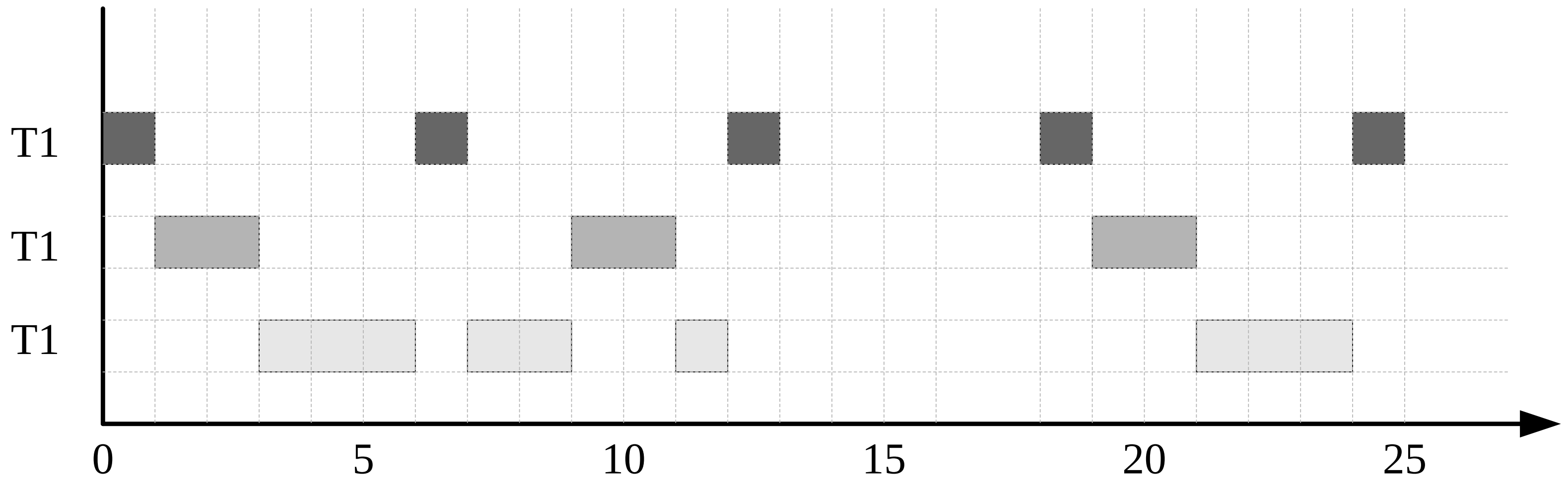
Basic Principles of RM

- Basic Idea
  - The more frequent a task is , the higher its priority!
  - The task with a shorter period is assigned a higher priority
  - This implies a more frequent task is more critical in the system, this is what the algorithm assumes, but not the real thing in real systems

- Properties of the Algorithm
  - Priority-driven
  - Preemptive
  - Static fixed priority

An Example of RM Scheduling

Task	Execution Time	Release Time	Period	Priority
T1	1	0	6	1
T2	2	0	9	2
T3	6	0	18	3



Schedulability Analysis (Test)

- A scheduling algorithm only specifies how to assign priorities among tasks, and in what order the tasks are scheduled
- But the algorithm itself cannot tell you whether a set of tasks is schedulable or not, so schedulability analysis (Test)

- Utilization Bound Test
  - Calculate total CPU utilization and compare it to a known bound
  - Simple, but pessimistic
- Exact Schedulability Analysis
  - Calculate Ri for each task i and compare it to its deadline Di
  - Accurate, with polynomial time complexity

Utilization-Based Schedulability Test

- A set of periodic tasks is schedulable with RM if

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

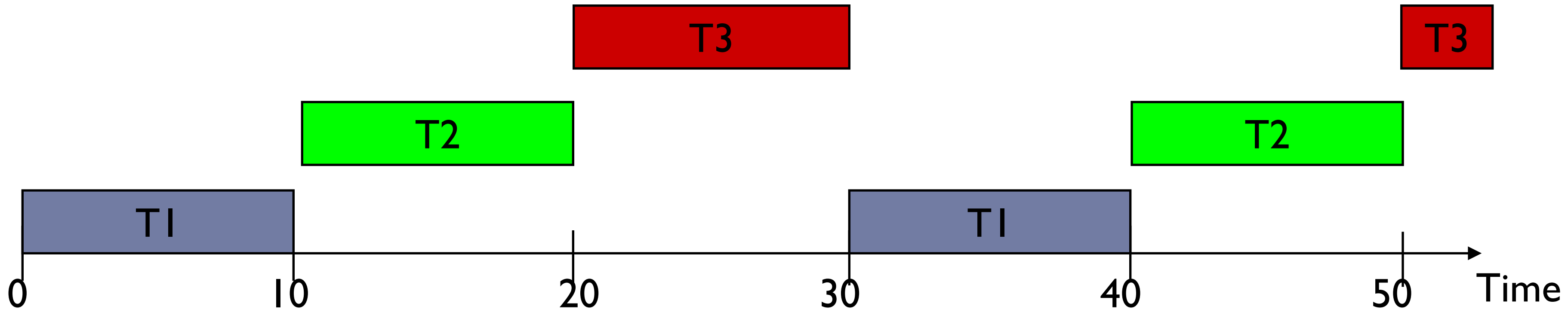
  - If n = 1, then U = 1; if n = 2, then U = 0.828; ….
  - If $n \to \infty$, then $U \to 0.69$

  - Guaranteed to be schedulable if test succeeds
  - May still be schedulable even if test fails
  - So the test is sufficient but not necessary

Pessimism of Utilization-Based Test

- Utilization = 12/52 + 10/40 + 10/30 = 81%
- Utilization bound (N = 3) = 78%
- Utilization bound test fails, but task set is actually schedulable!

Task	T	D	C	Prio
1	30	30	10	H
2	40	40	10	M
3	52	52	12	L



Optimality of RM

- Optimality: RM is optimal among all fixed-priority assignments in the sense that no other fixed-priority algorithm can schedule a task set that cannot be scheduled by RM
- Proof: the proof is done by considering several cases that may occur, the main idea is
  - A critical instant for any task occurs whenever the task is released simultaneously with all higher priority tasks. If all tasks are feasible at their critical instants, then the task set is schedulable in any other condition
  - Proof the above is true for a task set with two tasks
  - The extend the result to n tasks

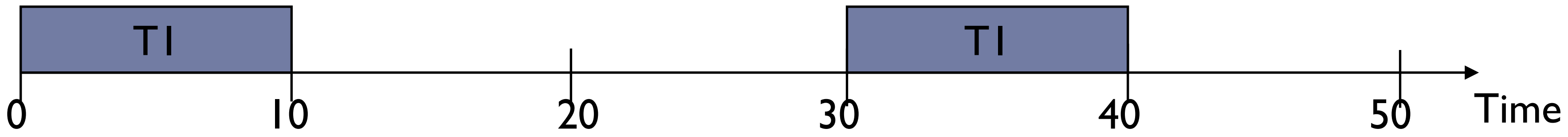
Response Time Based Analysis

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- For each task from the one with highest priority to the one with lowest priority, calculate the above formula recursively, until it converges, then check if the response time of task i is smaller than Di
- Here hp(i) is the set of tasks with priority higher than task i
- We will show this in an example

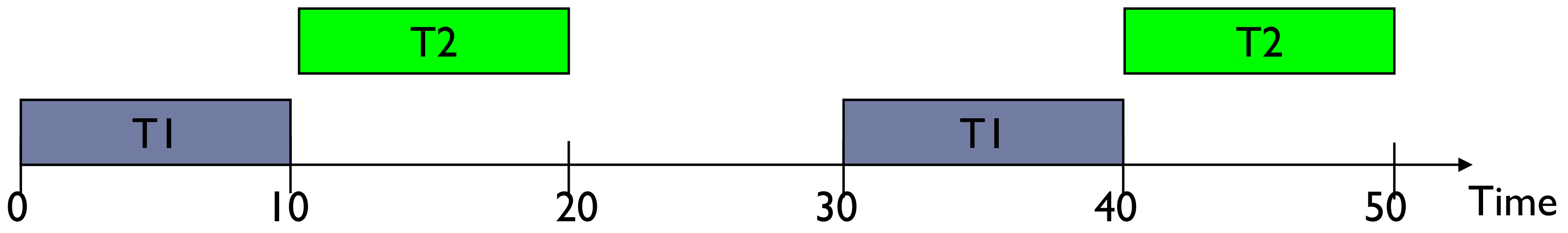
An Example

- Highest priority task, no interference from other tasks
- $R1 = C1 + 0 = 10$
- $R1 < D1$, so T1 is schedulable



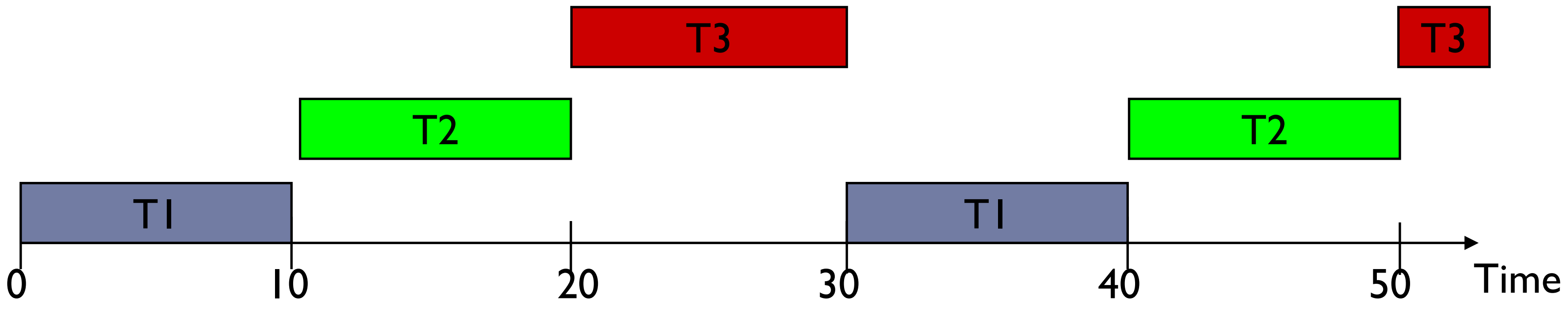
An Example

- Medium priority task T2
- R2 = C2 + ceil(R2/T1)*C1 = 10 + ceil(R2/30)*10
- Solve for R2 recursively, starting with R2 = C2 = 10:
  - R2 = 10 + ceil(10/30)*10 = 20
  - R2 = 10 + ceil(20/30)*20 = 20
  - We have converged, R2 = 20 < D2 = 40, so T2 is schedulable



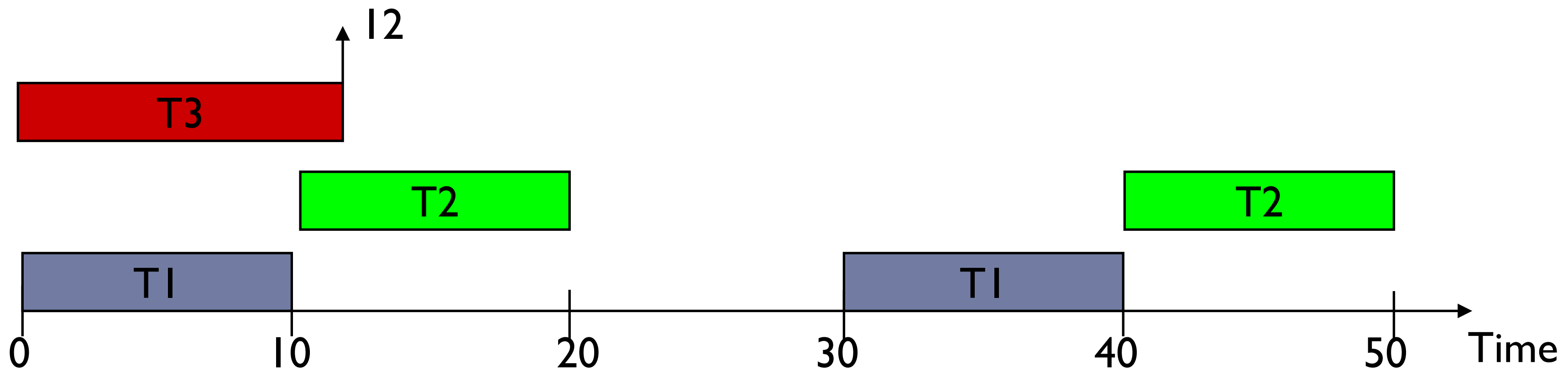
An Example

- Low priority task
- R3 = C3 + ceil(R3/T1)*C1 + ceil(R3/T2)*C2
- = 12 + ceil(R3/30)*10 + ceil(R3/40)*10
- Solve for R3 recursively, starting with R3 = C3 = 12:
  - R3 = 12 + ceil(12/30)*10 + ceil(12/40)*10 = 32
  - R3 = 12 + ceil(32/30)*10 + ceil(32/40)*10 = 42
  - R3 = 12 + ceil(42/30)*10 + ceil(42/40)*10 = 52
  - R3 = 12 + ceil(52/30)*10 + ceil(52/40)*10 = 52
  - We have converged, R3 = 52 = D3 = 52, so T3 is schedulable



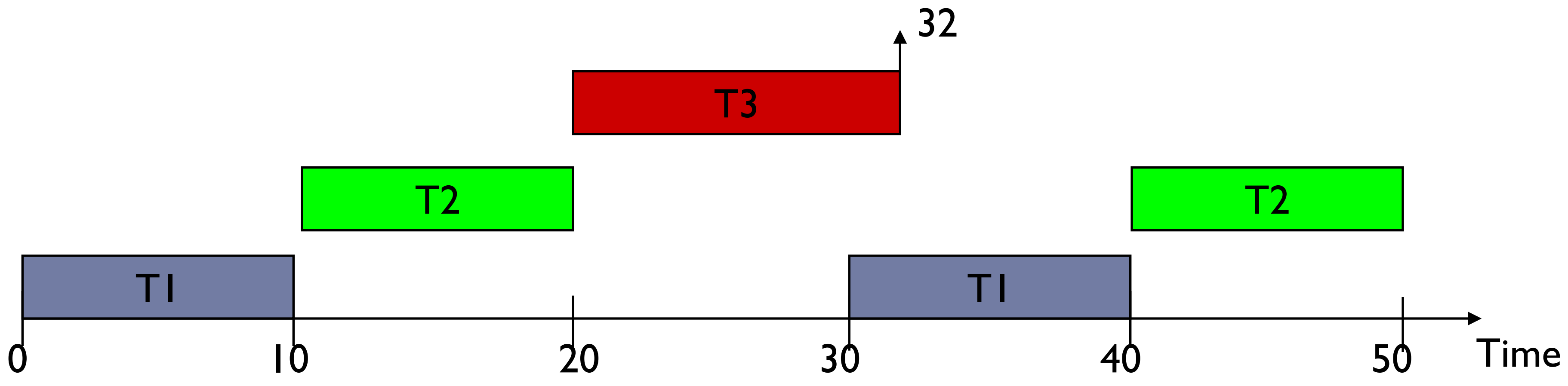
Some Intuition about the Recursive Analysis

- Initially R3 = 12
- We have not taken into account any preemption yet from T1 and T2 yet



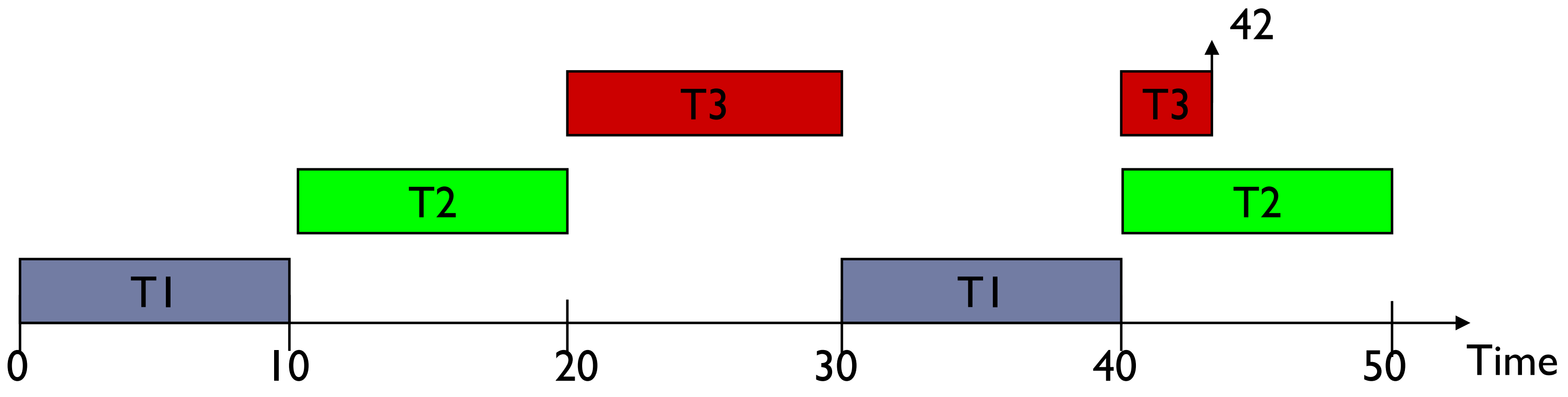
Some Intuition about the Recursive Analysis

- R3 = 12 + ceil(12/30)*10 + ceil(12/40)*10 = 32
- T1 preempts T3 once, and T2 preempts T3 once



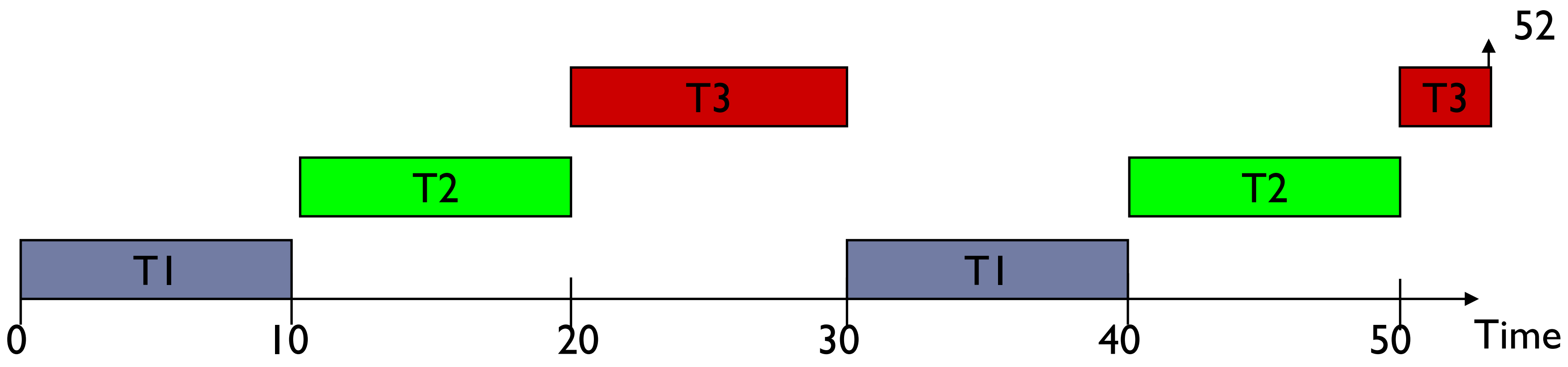
Some Intuition about the Recursive Analysis

- R3 = 12 + ceil(32/30)*10 + ceil(32/40)*10 = 42
- T1 preempts T3 twice, and T2 preempts T3 once



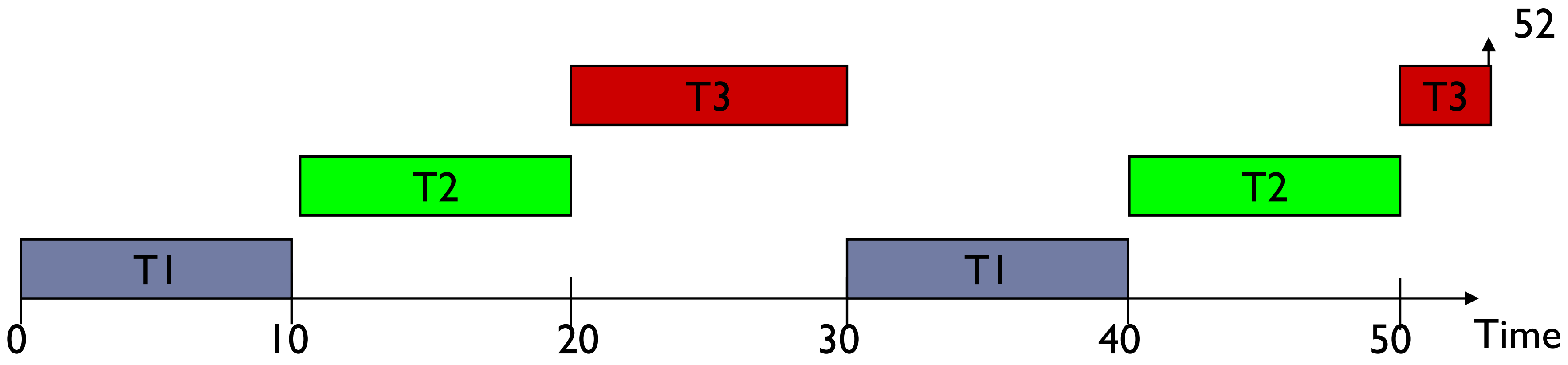
Some Intuition about the Recursive Analysis

- R3 = 12 + ceil(42/30)*10 + ceil(42/40)*10 = 52
- T1 preempts T3 twice, and T2 preempts T3 twice



Some Intuition about the Recursive Analysis

- R3 = 12 + ceil(52/30)*10 + ceil(52/40)*10 = 52
- T1 preempts T3 twice, and T2 preempts T3 twice
- Recursive equation has converged!
- This is the worst-case response time of T3



A Better Schedulability Test Condition

- A Sufficient and Necessary Test Condition for RM
  - Given periodic tasks $\tau_1, \tau_2, \ldots, \tau_n$

(1) τ_i can be scheduled for all task phasing using the RM algorithm iff

$$L_i = \min_{\{t \in S_i\}} W_i(t)/t \le 1$$

(2) The entire task set can be scheduled for all task phasing using the rate monotonic algorithm iff

$$L = \max_{\{1 \le i \le n\}} L_i \le 1$$

Where Si is the set of scheduling points, and

$$W_i(t) = \sum^{i} C_j \cdot \lceil t/T_j \rceil$$

Contents

Deadline Monotonic (DM) Scheduling

- Rate Monotonic Assumptions
  - A1: All tasks with hard deadlines are periodic
  - A2: The relative deadline of task equals to its period
  - A3: Tasks are independent of each other
  - A4: WCET of each task is known a prior
  - A5: All aperiodic tasks are not hard real-time ones
- Rate Monotonic Incapability
  - Cannot handle sporadic tasks
  - Cannot handle tasks of which D <> T
  - Inter-dependent tasks
  - Resource contention
  - System overhead

Basic Principles of DM

- Basic Idea
  - The task with shortest deadline is assigned highest priority
  - An optimal static priority assignment for one processor
- Some Remarks
  - Similar in concept with Rate Monotonic Scheduling
  - Deadline Monotonic can be viewed as a generalization of Rate Monotonic
  - Deadline Monotonic Algorithm can deal with a wider range of task sets since it relaxes the first two assumptions made by Rate Monotonic

Schedulability Test for DM – I

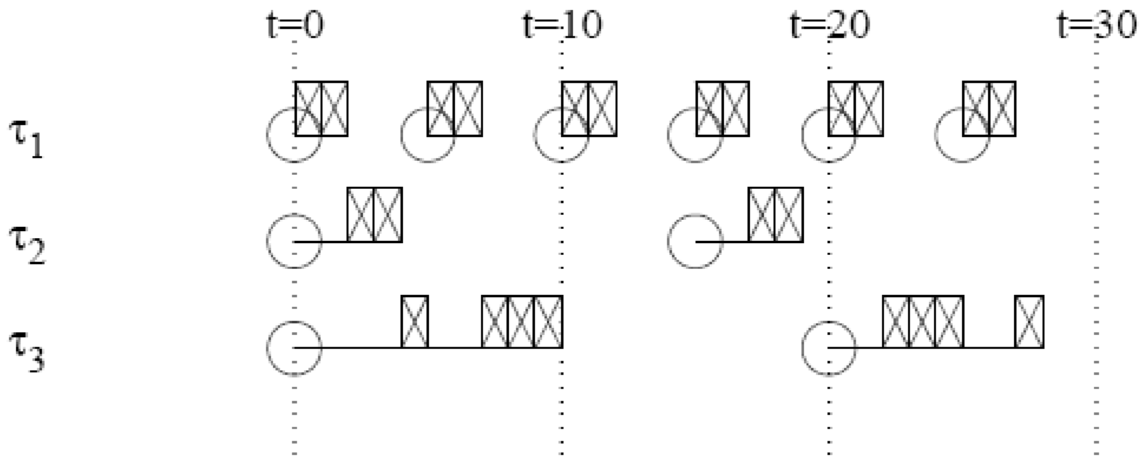
- A sufficient but not necessary test with O(n) complexity

$$\forall i : 1 \le i \le n : \frac{C_i}{D_i} + \frac{I_i}{D_i} \le 1 \qquad I_i = \sum_{j=1}^{i-1} \left\lceil \frac{D_i}{T_j} \right\rceil \cdot C_j$$

- An example

Task	C	D	T
τ_1	2	3	5
τ_2	2	6	15
τ_3	4	10	20

Schedulability Test for DM – I



τ_1 is schedulable since $2/3 < 1$

τ_2 is schedulable since $2/6 + 4/6 = 1$ with $I_2 = 4$

τ_3 is schedulable since $4/10 + 6/10 = 1$ with $I_3 = 6$

Schedulability Test for DM – II

- A sufficient but not necessary test with O(n^2) complexity

$$\forall i : 1 \le i \le n : \frac{C_i}{D_i} + \frac{I_i}{D_i} \le 1$$

$$I_i = \sum_{j=1}^{i-1} \left\{ \left[\left\lfloor \frac{D_i - D_j}{T_j} \right\rfloor + 1 \right] \cdot C_j + \left[\left\lceil \frac{D_i}{T_j} \right\rceil - \left[\left\lfloor \frac{D_i - D_j}{T_j} \right\rfloor + 1 \right] \right] \times \min \left[C_j , D_i - \left\lfloor \frac{D_i}{T_j} \right\rfloor \cdot T_j \right] \right\}$$

- Is This Schedulability Test Optimal?
  - Not optimal!
  - The calculation of incomplete execution is not exact in general situations

Schedulability Test for DM – III

- A sufficient and necessary test condition for DM
  - Assume that τ_i completes execution at t', it's apparent that all tasks with higher priorities have completed
  - So the schedulable condition for τ_i is:

$$\exists t_0 \le t \le D_i : \frac{I_i^t}{t} + \frac{C_i}{t} = 1 \qquad I_i^t = \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

  - We need not consider the whole time period of $[0, D_i]$.
  - We only consider the time spot at which τ_i is possibly completed

Schedulability Test for DM – III

```
foreach $\tau_i$ do
    $t = \sum_{j=1}^{i} C_j$
    continue = TRUE
    while (continue) do
        if $\left( \frac{I_i^t}{t} + \frac{C_i}{t} \le 1 \right)$
            continue = FALSE    /* $\tau_i$ is schedulable */
        else
            $t = I_i^t + C_i$
        endif
        if $\left( t > D_i \right)$
            exit        /* $\tau_i$ is unschedulable */
        endif
    endwhile
endfor
```

The sufficient and necessary schedulability test has data-dependent complexity!

Schedulability Test for DM – III

- An Example

Task	C	D	T
τ_1	4	6	10
τ_2	3	7	11
τ_3	5	13	20

$$t_0 = \sum_{j=1}^{3} C_i = 4+3+5 = 12$$

$$I_3^{12} = \sum_{j=1}^{2} \left\lceil \frac{12}{T_j} \right\rceil C_j = \left\lceil \frac{12}{10} \right\rceil 4 + \left\lceil \frac{12}{11} \right\rceil 3 = 14 \qquad \frac{14}{12} + \frac{5}{12} > 1$$

$$t_1 = I_3^{t_0} + C_3 = I_3^{12} + 5 = 19$$

Since we now have $t_1 > D_3$ we terminate with τ_3 unschedulable.

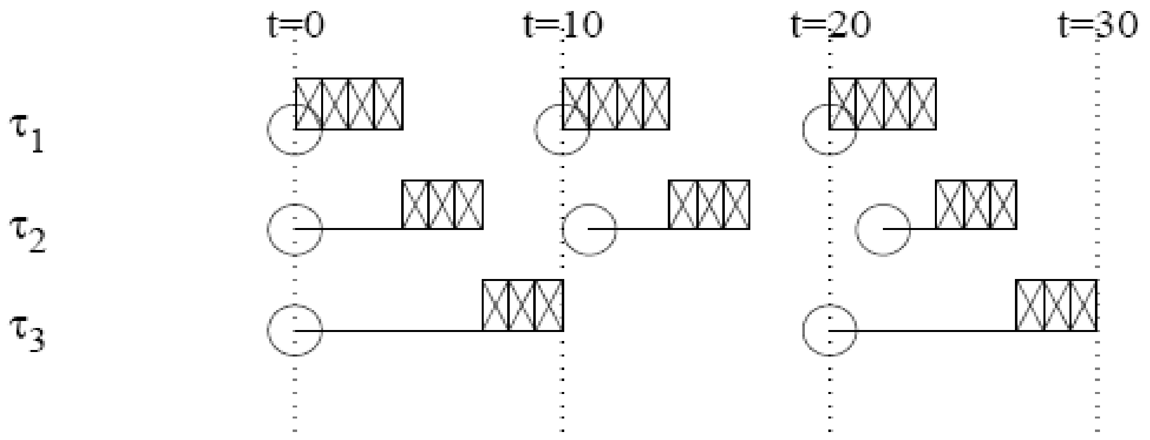
Schedulability Test for DM – III

Task	C	D	T
τ_1	4	6	10
τ_2	3	7	11
τ_3	3	13	20

$$t_0 = \sum_{j=1}^{3} C_j = 4+3+3 = 10$$

$$I_3^{10} = \sum_{j=1}^{2} \left\lceil \frac{1}{T_j} \right\rceil C_j = \left\lceil \frac{10}{10} \right\rceil 4 + \left\lceil \frac{10}{11} \right\rceil 3 = 7$$

$$\frac{7}{10} + \frac{3}{10} = 1$$



Contents

A Review of Fixed-Priority Scheduling

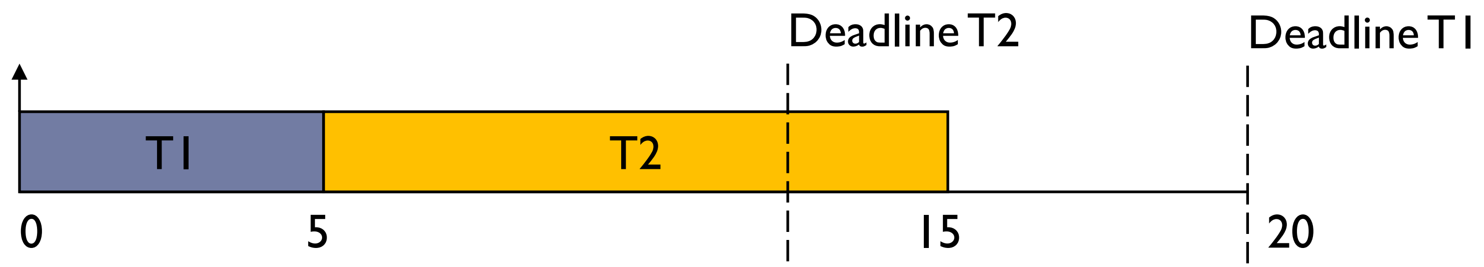
- In fixed-priority scheduling, each task is assigned a fixed priority for all its invocations
- Pros
  - Predictability
  - Low runtime overhead
  - Temporal isolation during overload
- Cons
  - Cannot achieve 100% utilization in general, except when task periods are harmonic
- Widely used in most commercial RTOSes and CAN bus

Basic Principles of EDF

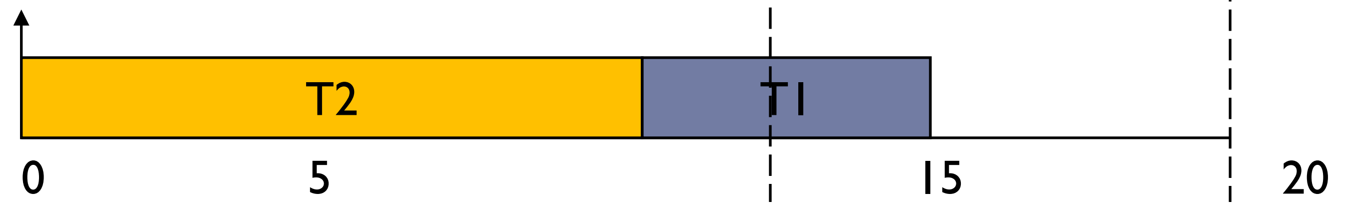
- Basic Idea
  - The priorities of tasks are assigned according to the absolute deadline of each task, the task with the deadline nearest to the current scheduling point is assigned highest priority
- Properties
  - For a specific task, the priority may change during runtime
  - The system can be either preemptive or non-preemptive
  - In preemptive systems, the EDF scheduling algorithm is optimal
  - The EDF algorithm can achieve full utilization of the processor
  - But, high runtime overhead, and poor temporal isolation during overload

A Simplest EDF Example

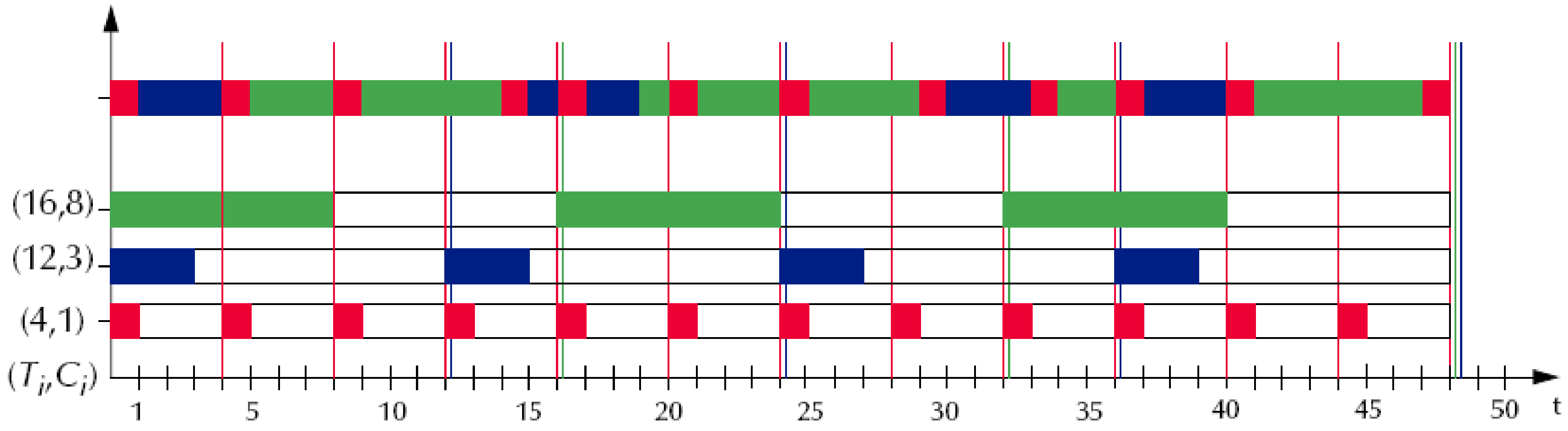
- Say you have two tasks to run:
  - T1 takes 5 ms, with a deadline of 20 ms
  - T2 takes 10 ms, with a deadline of 12 ms
- T1 before T2:



- T2 before T1:



A Bigger Example



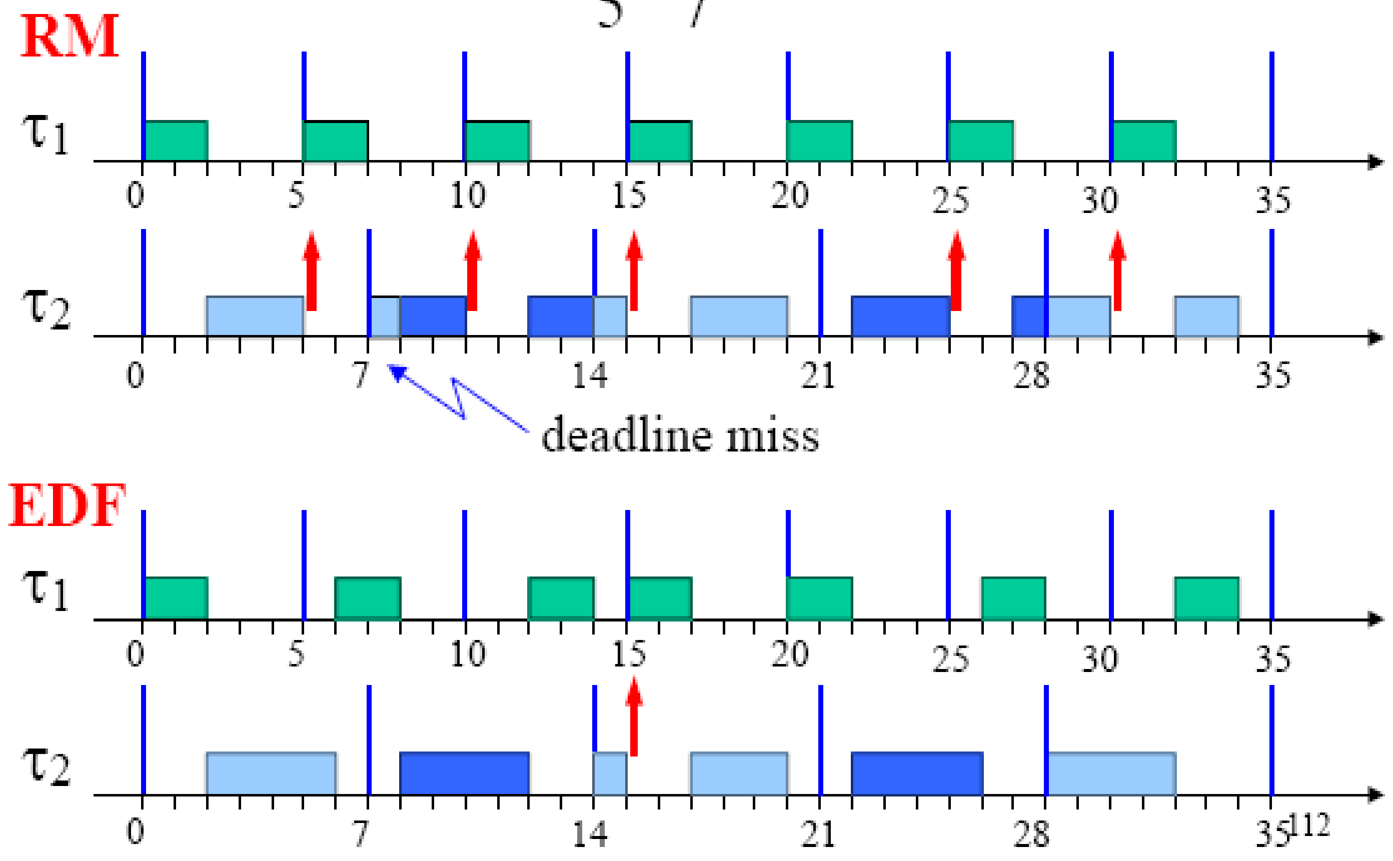
- Using EDF algorithm to schedule 3 periodic tasks
- Unlike RM algorithm, priorities of tasks may change in different activations (jobs)

RM vs EDF – Extended Discussion

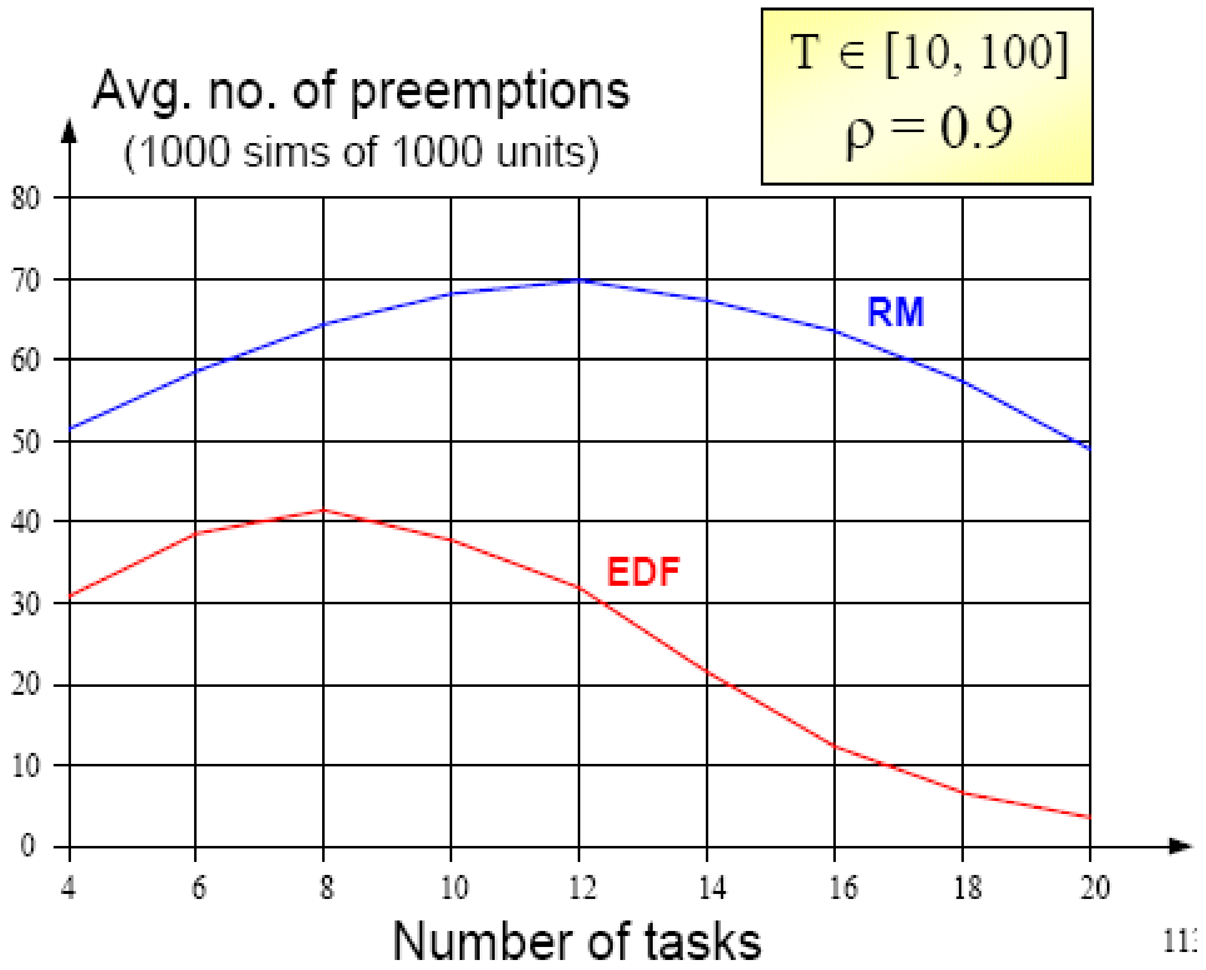
- If EDF is more efficient than RM, why commercial RT systems are still based on RM?
  - RM is simpler to implement on top of commercial (fixed priority) kernels.
  - EDF requires explicit kernel support for deadline scheduling, but gives other advantages, e.g., less overhead due to less preemptions
- Two different types of overhead are considered
  - (1) Overhead for job release
    -

An Example

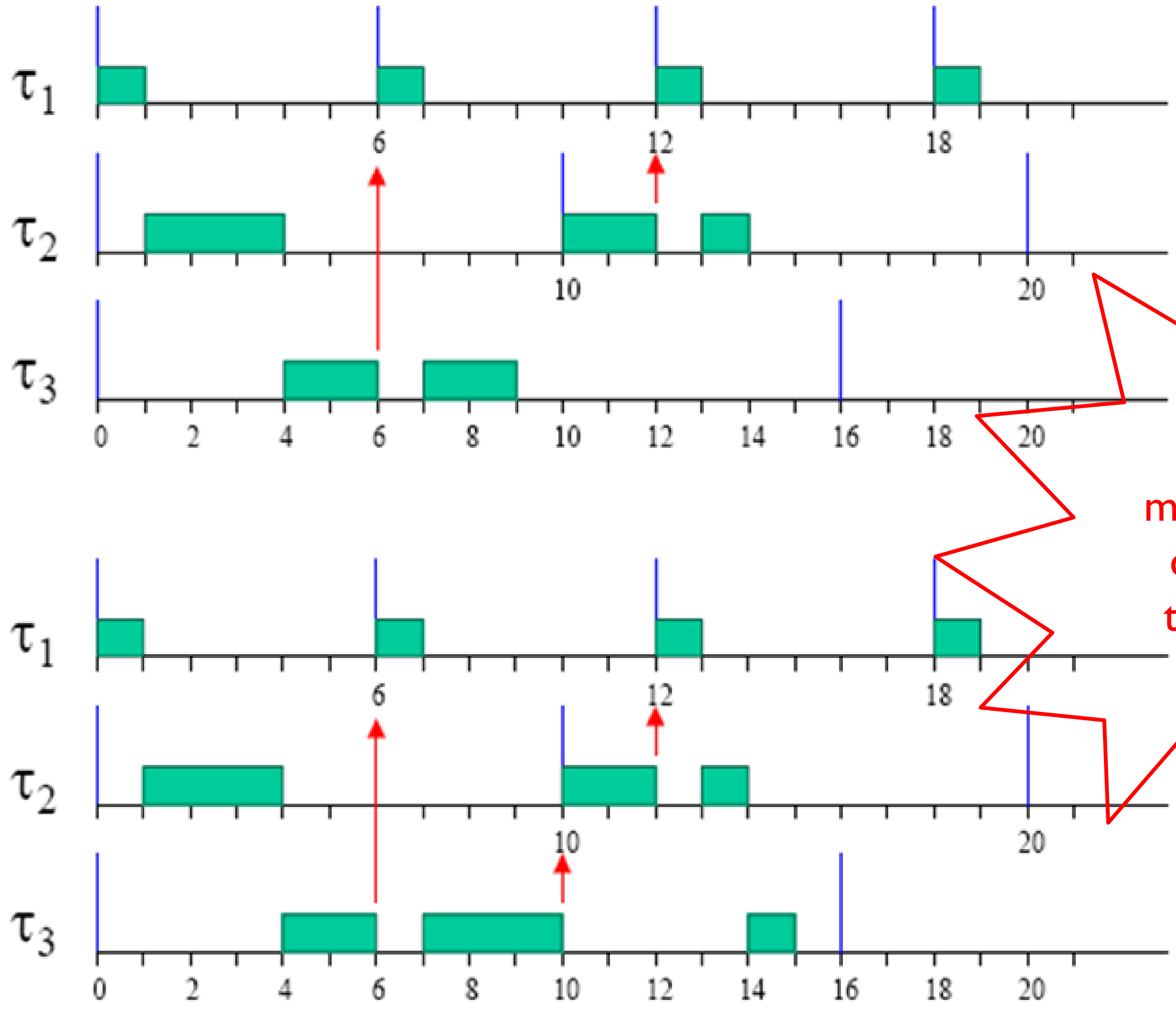
$$U = \frac{2}{5} + \frac{4}{7} \cong 0.97$$



The Overhead of Preemptions

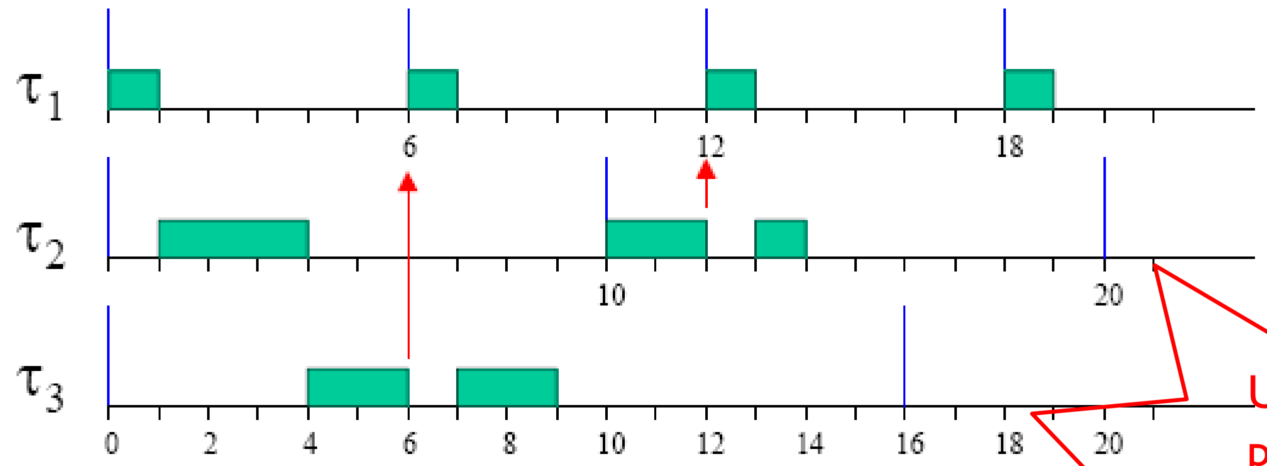


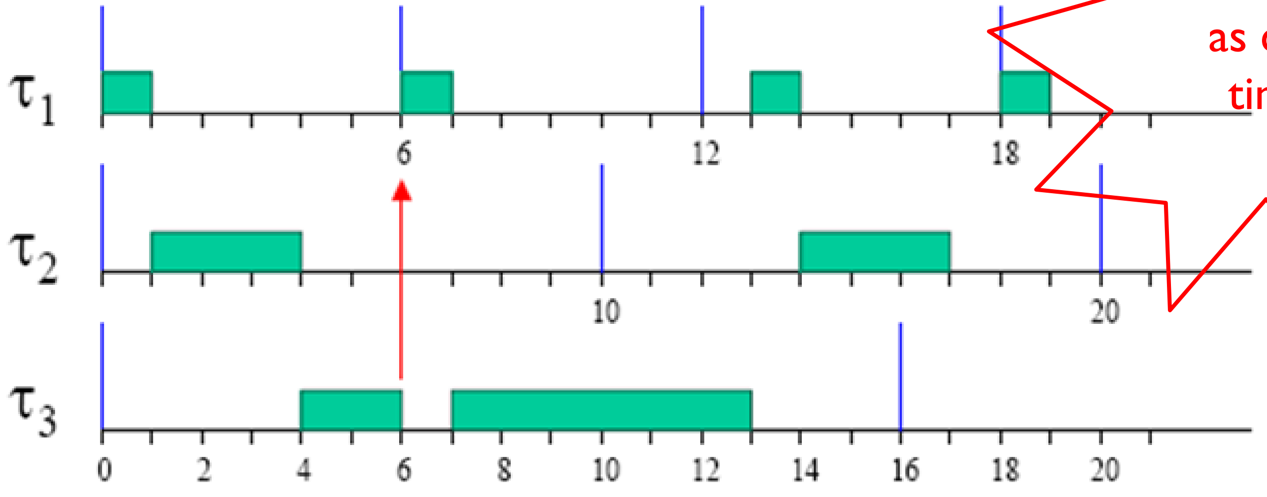
The Overhead of Preemptions



Under RM, preemtions may increase as computation time increase

The Overhead of Preemptions





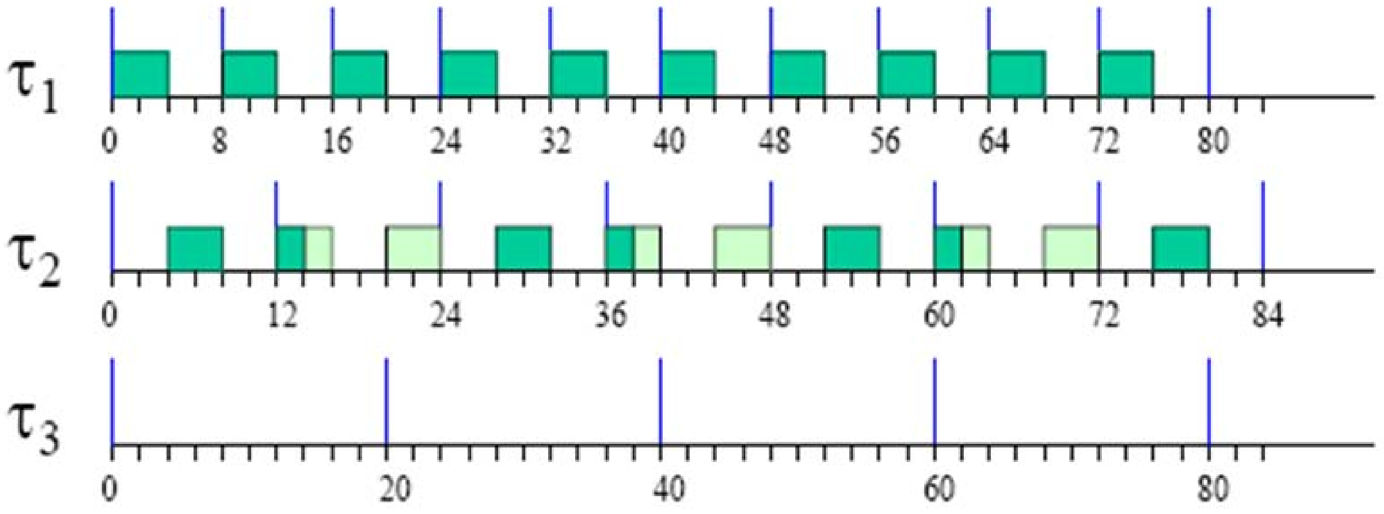
Under EDF, preemtions may decrease as computation time increase

Robustness under Overload

- There are two main kinds of overloads
  - Permanent overload
    - ⇒ This occurs when $U > 1$
  - Trnscient overload
    - ⇒ This occurs when some job executes more than expected

RM under Permanent Overload

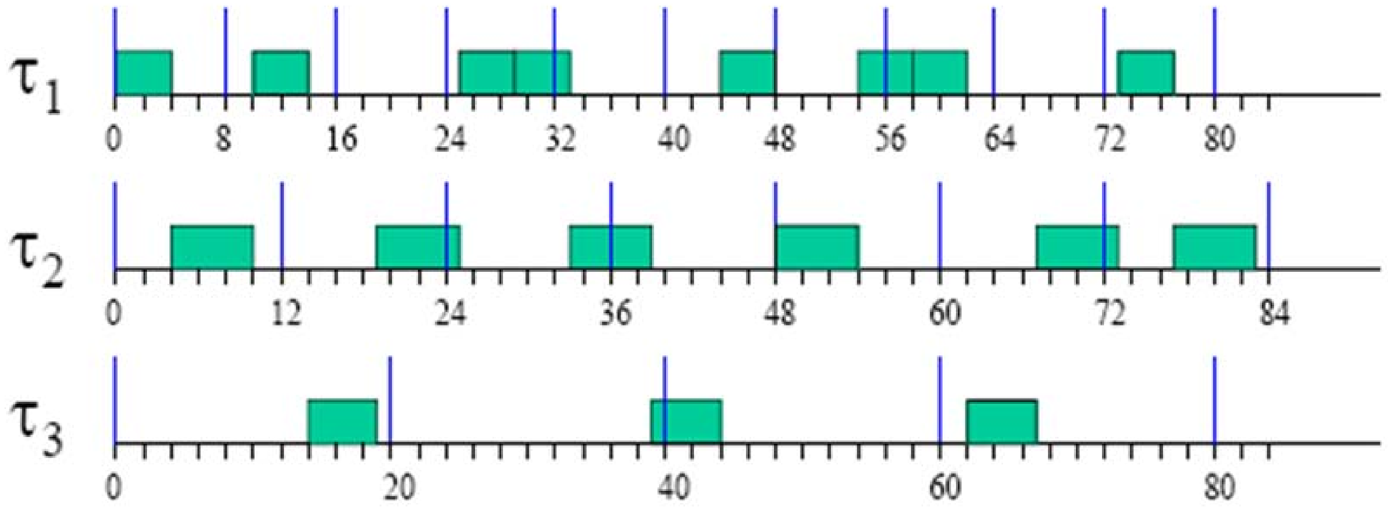
$$U = \frac{4}{8} + \frac{6}{12} + \frac{5}{20} = 1.25$$



- High priority tasks execute at the proper rate
- Low priority tasks are completely blocked

EDF under Permanent Overload

$$U = \frac{4}{8} + \frac{6}{12} + \frac{5}{20} = 1.25$$



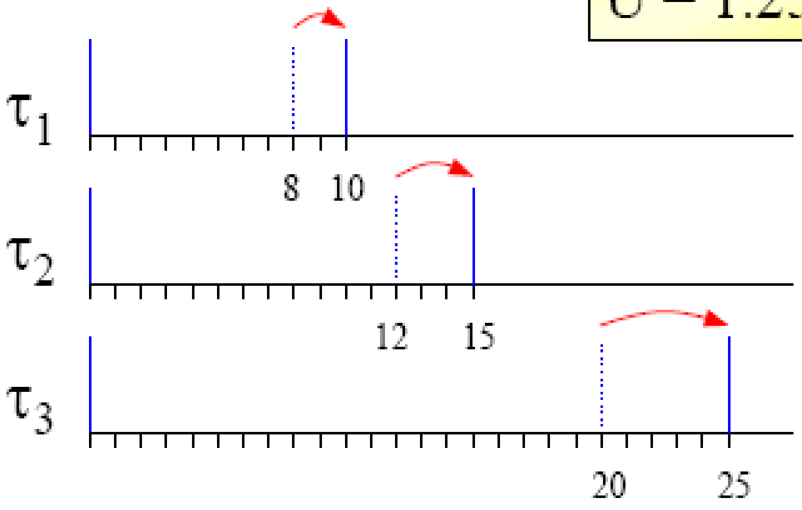
- All tasks execute at a slower rate
- No task is blocked

EDF under Permanent Overload

Theorem (Cervin '03)

If $U > 1$, EDF executes tasks with an average period $T'_i = T_i U$.

$U = 1.25$



	T_i	T'_i
τ_1	8	10
τ_2	12	15
τ_3	20	25

122

Misconceptions about Predictability

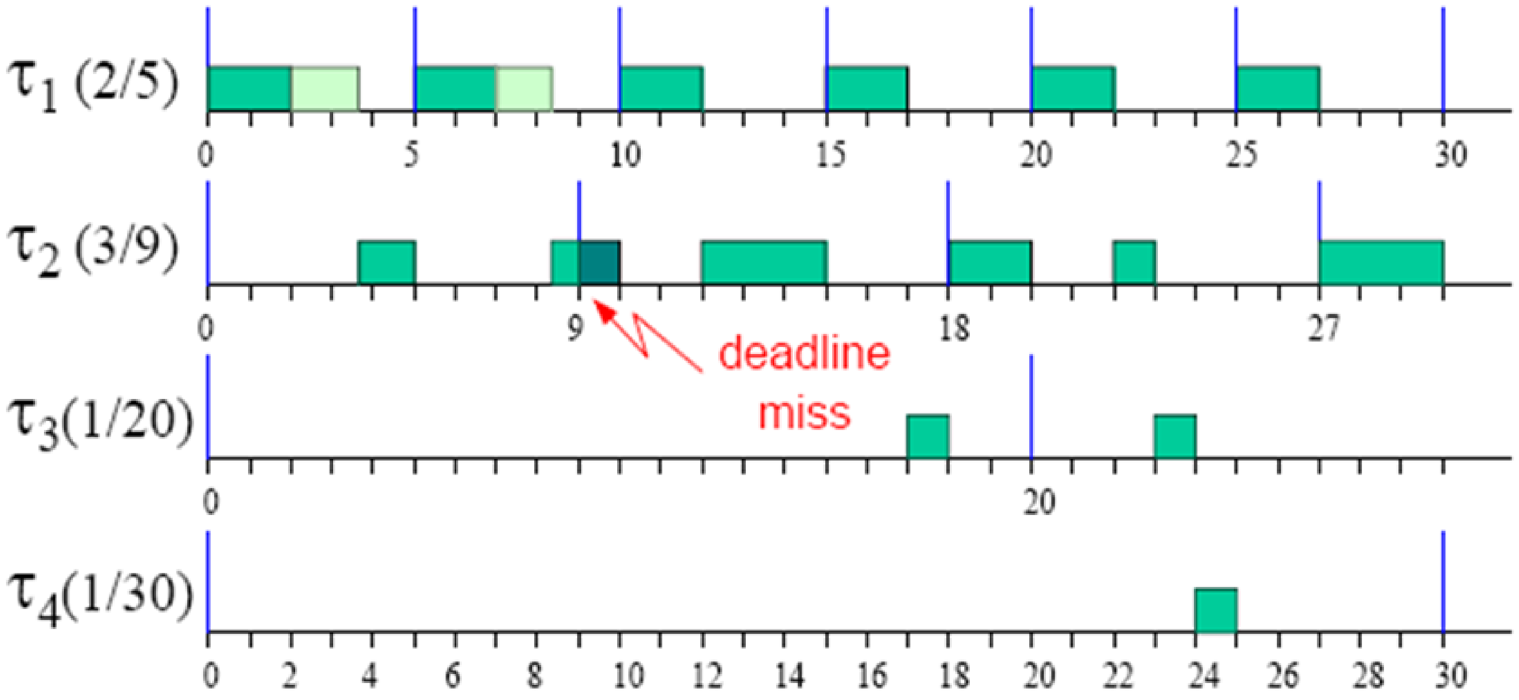
~~EDF is not predictable during overloads because we don't know which tasks are going to miss their deadlines~~

~~RM is predictable during overloads because the tasks that miss deadlines are low priority tasks~~

We now show that this is not true

RM During Transient Overrun

$U_{avg} = 0.817 \quad C_{1_{avg}} = 2, \; C_{1_{max}} = 4$

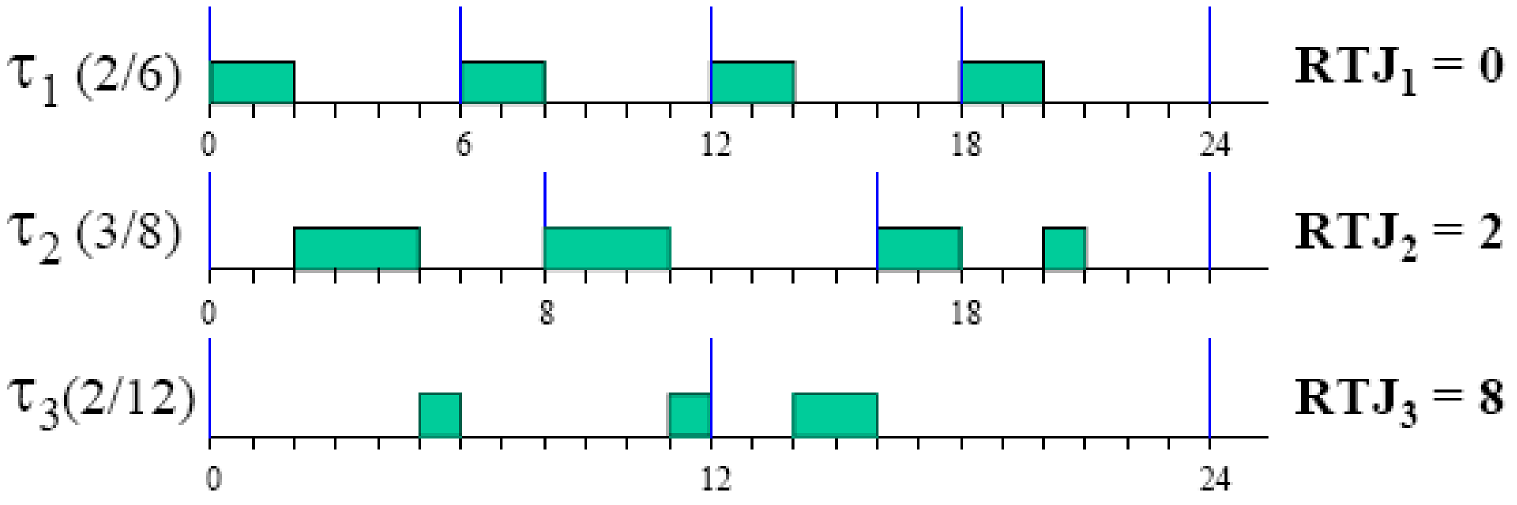


Who is missing its deadline is not the lowest priority task

Misconceptions about Jitters

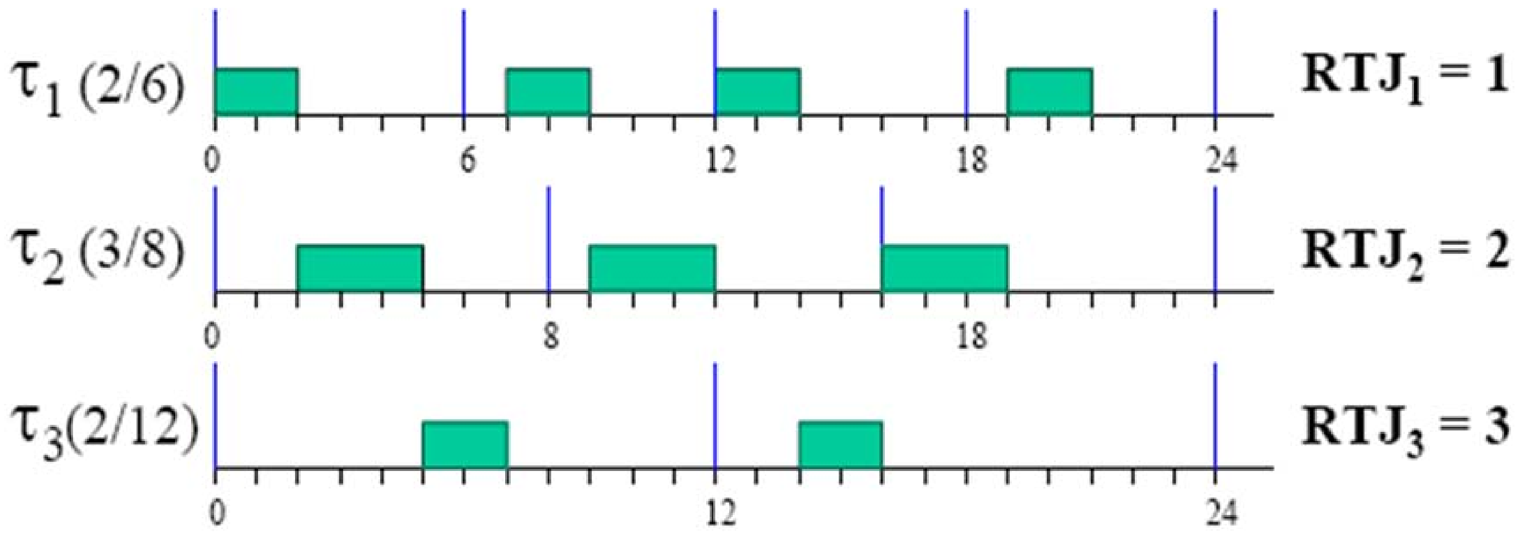
~~RM reduces jitter during task execution more than EDF~~

Task Jitter under RM



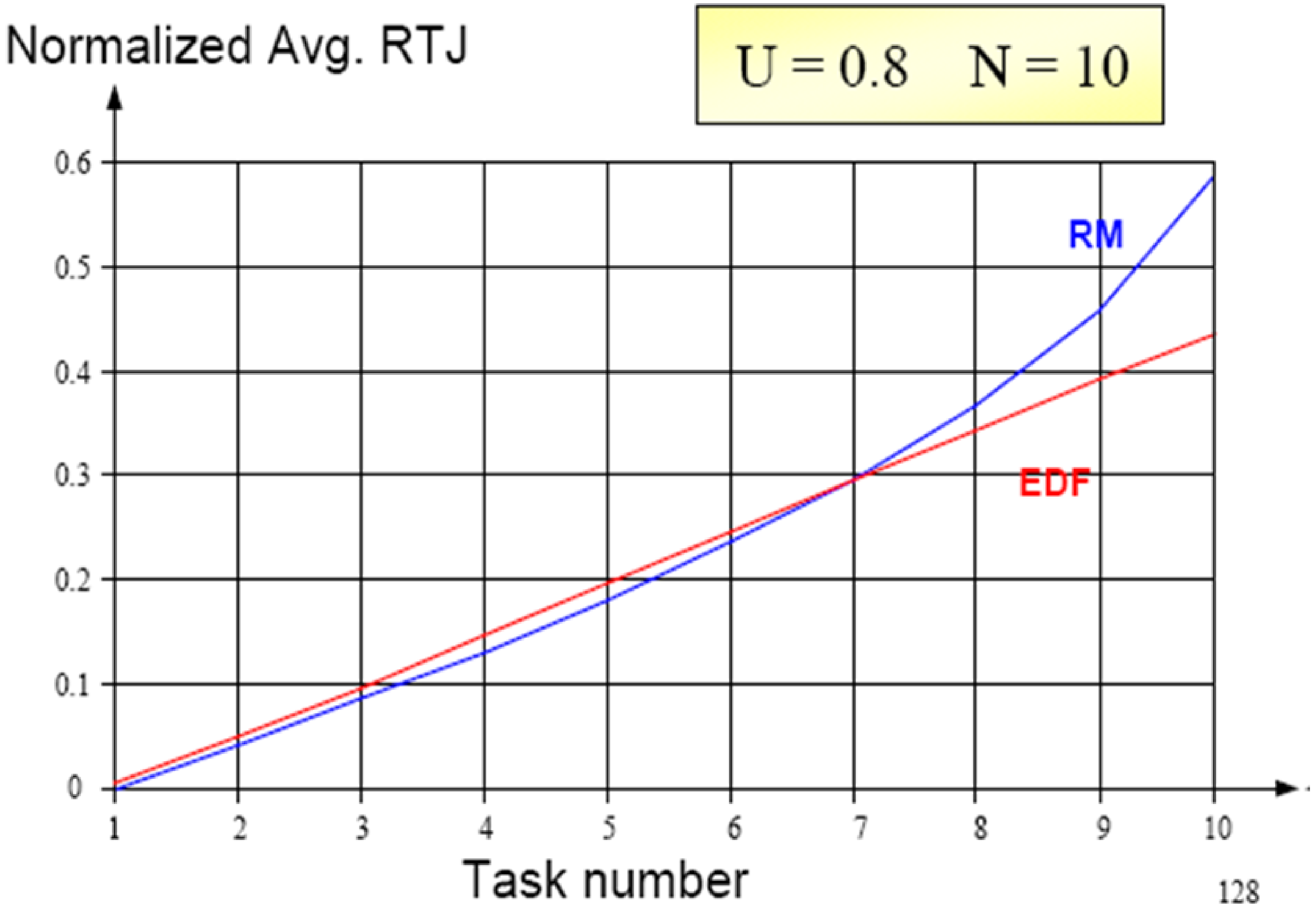
τ_3 experiences a very high jitter

Task Jitter under EDF

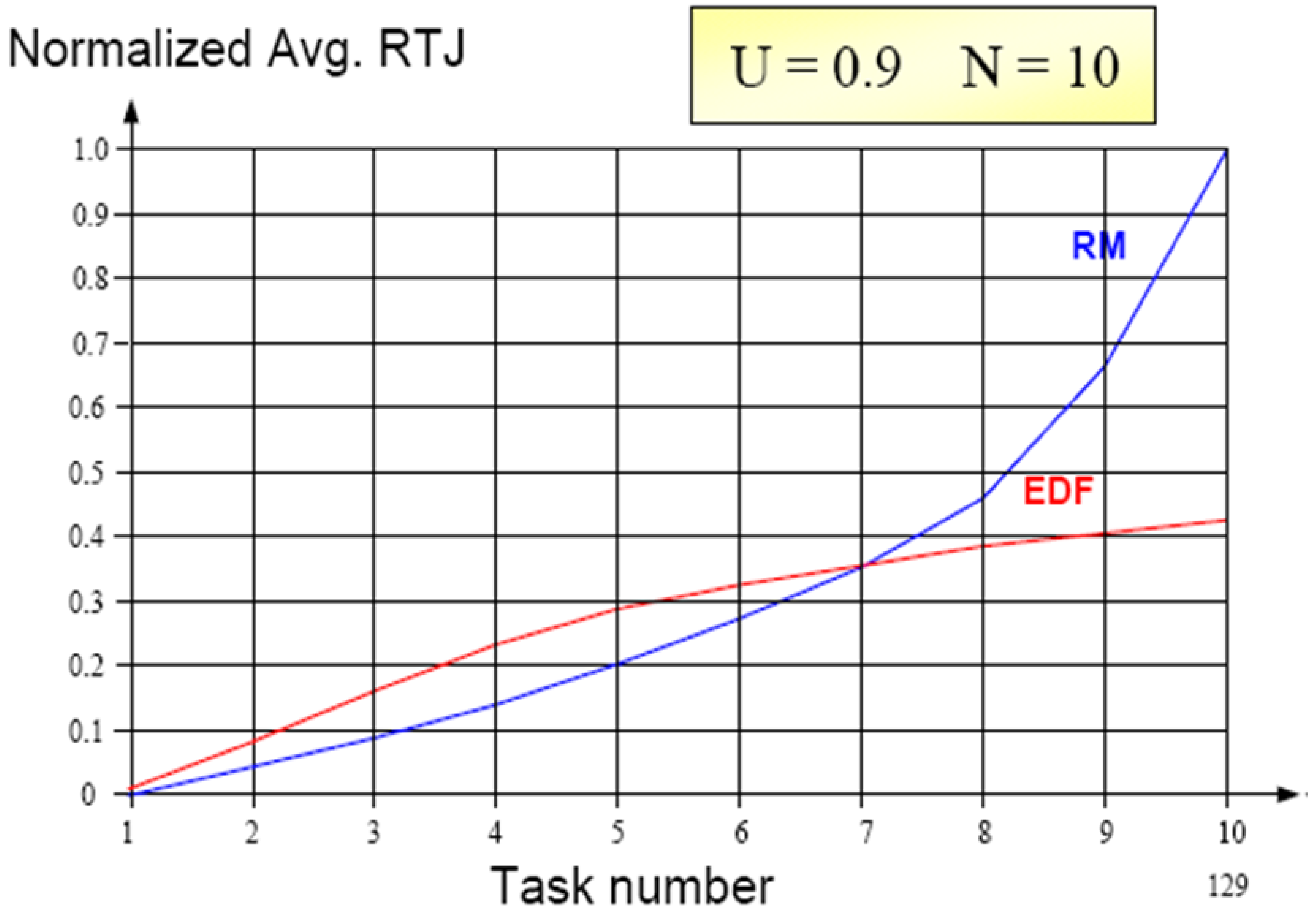


For a little increase of RTJ_1, RTJ_3 decreases a lot

Jitter Experiments



Jitter Experiments



Contents

Recommended Readings

- Real-Time Scheduling Overview

1. Jane W.S. Liu, *Real-Time Systems*, 2002.
2. William Stallings, *Operating Systems: Internals and Design Principles (3rd)*, tup, 1998.
3. Rita Voigt, *An overview of real-time scheduling*.
4. Krithi Ramamritham and John A. Stankovic, *Scheduling algorithms and operating systems support for real-time systems*.
5. Lui Sha, et al. *Real Time Scheduling Theory: A Historical Perspective*. 2004.

- Rate Monotonic Scheduling

1. Liu and Layland, *Scheduling algorithms for multiprogramming in a hard real-time environment.*
2. Liu Sha and R. Rajkumar, *Generalized Rate Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems*.
3. Liu Sha and M.H. Klein, *Rate Monotonic Analysis for Real-Time Systems*.
4. N. Audsley, A. Burns and M. Richardson, *Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling*.
5. Klara Nahrstedt, *Software design for a rate monotonic scheduler*.
6. J. Lehoczky and Liu sha, *The rate monotonic scheduling algorithm exact characterization and average case behavior*.
7. Janusz Zalewski, *What every engineer needs to know about rate monotonic scheduling a tutorial.*

Recommended Readings

- Deadline Monotonic Scheduling

1. Neil C. Audsley, *Deadline Monotonic Scheduling*.
2. Neil C. audsley and A. Burns, *Hard real-time scheduling the deadline monotonic approach*.
3. YOSHIFUMI MANABE and SHIGEMI AOYAGI, *A Feasibility Decision Algorithm for Rate Monotonic and Deadline Monotonic Scheduling*.

- Earliest Deadline First Scheduling

1. Jane W.S. Liu, Real-Time Systems, 2002.

- Acknowledgement
  - Lots of slides in this chapter are borrowed from Prof. Zonghua Gu's RTS course at HKUST, here we show our thankfulness to Prof. Gu ☺
  - http://www.cse.ust.hk/~zgu/comp680g/

Visit Our Website ☺

- The Website of Real-Time Embedded Systems Laboratory, Northeastern University
  - http://www.neu-rtes.org
  - http://www.neu-rtes.org/courses/spring2009/
- You can find
  - General information on the projects conducted in our lab
  - Research and publications
  - Research information and contacts of the members
  - Some useful research links
- Write me emails if you have questions in RTS
  - mingsong@research.neu.edu.cn