Real-Time Scheduling

Scheduling of Reactive Systems

Priority-Driven Scheduling

Current Assumptions

- Single processor
- Fixed number, n, of *independent periodic* tasks

  i.e. there is no dependency relation among jobs
  - Jobs can be preempted at any time and never suspend themselves
  - No aperiodic and sporadic jobs
  - No resource contentions

Moreover, unless otherwise stated, we assume that

- **Scheduling decisions take place precisely at**
  - release of a job
  - completion of a job

  (and nowhere else)
- Context switch overhead is negligibly small

  i.e. assumed to be zero
- There is an unlimited number of priority levels

Fixed-Priority vs Dynamic-Priority Algorithms

A priority-driven scheduler is on-line
i.e. it does not precompute a schedule of the tasks

- It assigns priorities to jobs after they are released and places the jobs in a ready job queue in the priority order

  with the highest priority jobs at the head of the queue

- At each scheduling decision time, the scheduler updates the ready job queue and then schedules and executes the job at the head of the queue

  i.e. one of the jobs with the highest priority

Fixed-priority = all jobs in a task are assigned the same priority

Dynamic-priority = jobs in a task may be assigned different priorities

Fixed-priority Algorithms – Rate Monotonic

Best known fixed-priority algorithm is *rate monotonic (RM)* scheduling that assigns priorities to tasks based on their periods

- The shorter the period, the higher the priority
- The *rate* is the inverse of the period, so jobs with higher rate have higher priority

RM is very widely studied and used

The *deadline monotonic (DM)* algorithm assigns priorities to tasks based on their *relative deadlines*

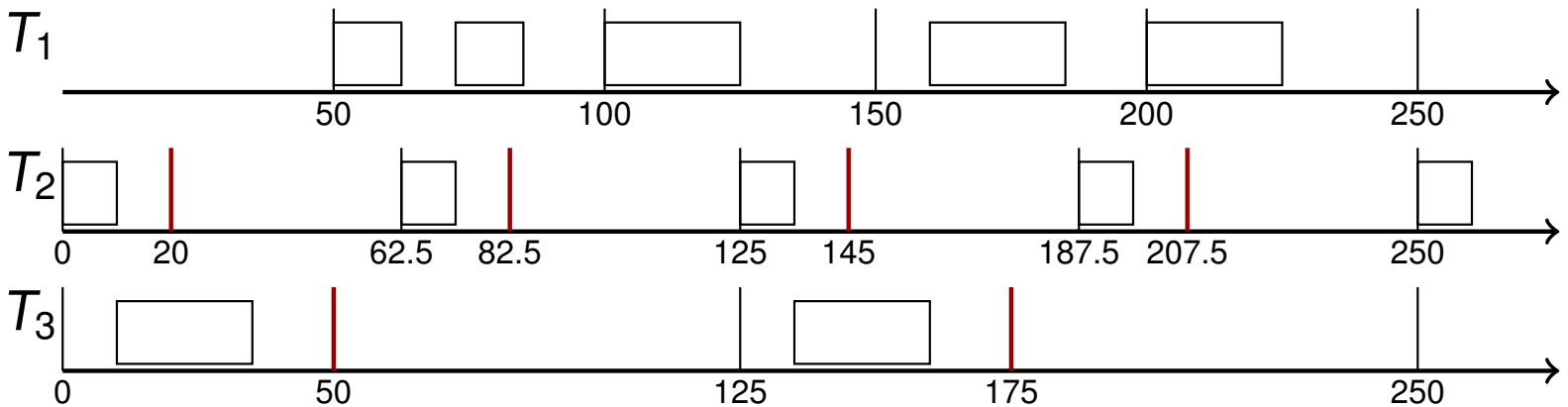
- the shorter the deadline, the higher the priority

Rate Monotonic vs Deadline Monotonic

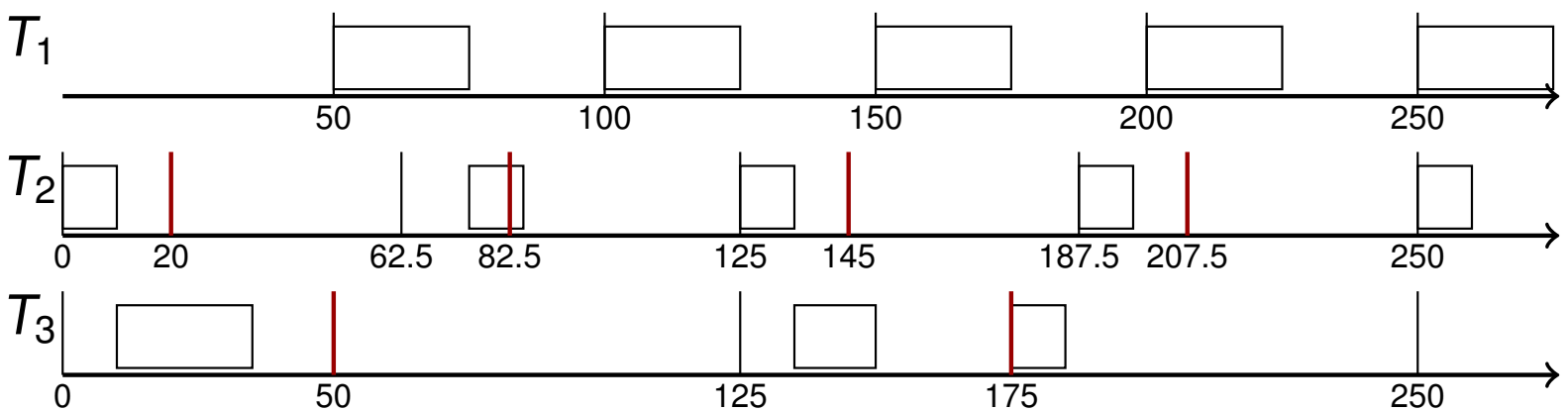
$T_1 = (50, 50, 25, 100)$, $T_2 = (0, 62.5, 10, 20)$, $T_3 = (0, 125, 25, 50)$

Priorities: $T_2 \succ T_3 \succ T_1$

DM is optimal:



RM is not optimal:



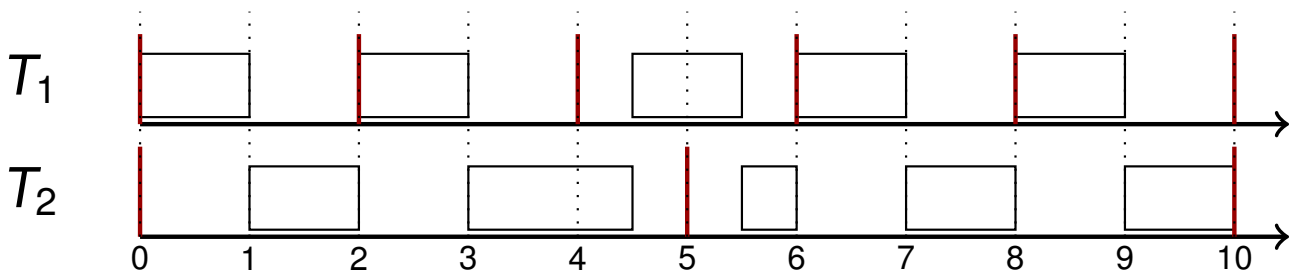
Dynamic-priority Algorithms

Best known is *earliest deadline first (EDF)* that assigns priorities based on *current* deadlines

- At the time of a scheduling decision, the job queue is ordered by earliest deadline

Example 1

$T_1 = (2, 1)$ and $T_2 = (5, 2.5)$



Summary of Algorithms

In what follows we consider

Dynamic-priority algorithms: EDF

Fixed-priority algorithms: RM and DM

We consider the following questions:

- Are the algorithms optimal?
- How to efficiently (or even online) test for schedulability?

To measure abilities of scheduling algorithms and to get fast online tests of schedulability we use a notion of *utilization*

Utilization

- *Utilization u_i of a periodic task T_i* with period p_i and execution time e_i is defined by $u_i := e_i/p_i$
  - The fraction of time a periodic task with period p_i and execution time e_i keeps a processor busy
- *Total utilization U^T of a set of tasks $T = \{T_1, \ldots, T_n\}$* is defined as the sum of utilizations of all tasks of T, i.e. by

$$U^T := \sum_{i=1}^{n} u_i$$

- U is a *schedulable utilization* of an algorithm ALG if all sets of tasks T satisfying $U^T \leq U$ are schedulable by ALG.

  Maximum schedulable utilization U_{ALG} of an algorithm ALG is the *supremum of schedulable utilizations of ALG*.

  That is
  - If $U^T < U_{ALG}$, then T is schedulable by ALG.
  - If $U > U_{ALG}$, then there is T with $U^T \leq U$ that is not schedulable by ALG.

Real-Time Scheduling

Priority-Driven Scheduling

Dynamic-Priority

Optimality of EDF

Theorem 2

Let T be a set of independent, preemptable periodic tasks with $D_i \geq p_i$. The following statements are equivalent:

1. *T can be feasibly scheduled on one processor*
2. *total utilization U^T of T is equal to or less than one*
3. *T is schedulable using EDF*

(i.e. $U_{EDF} = 1$)

Proof.

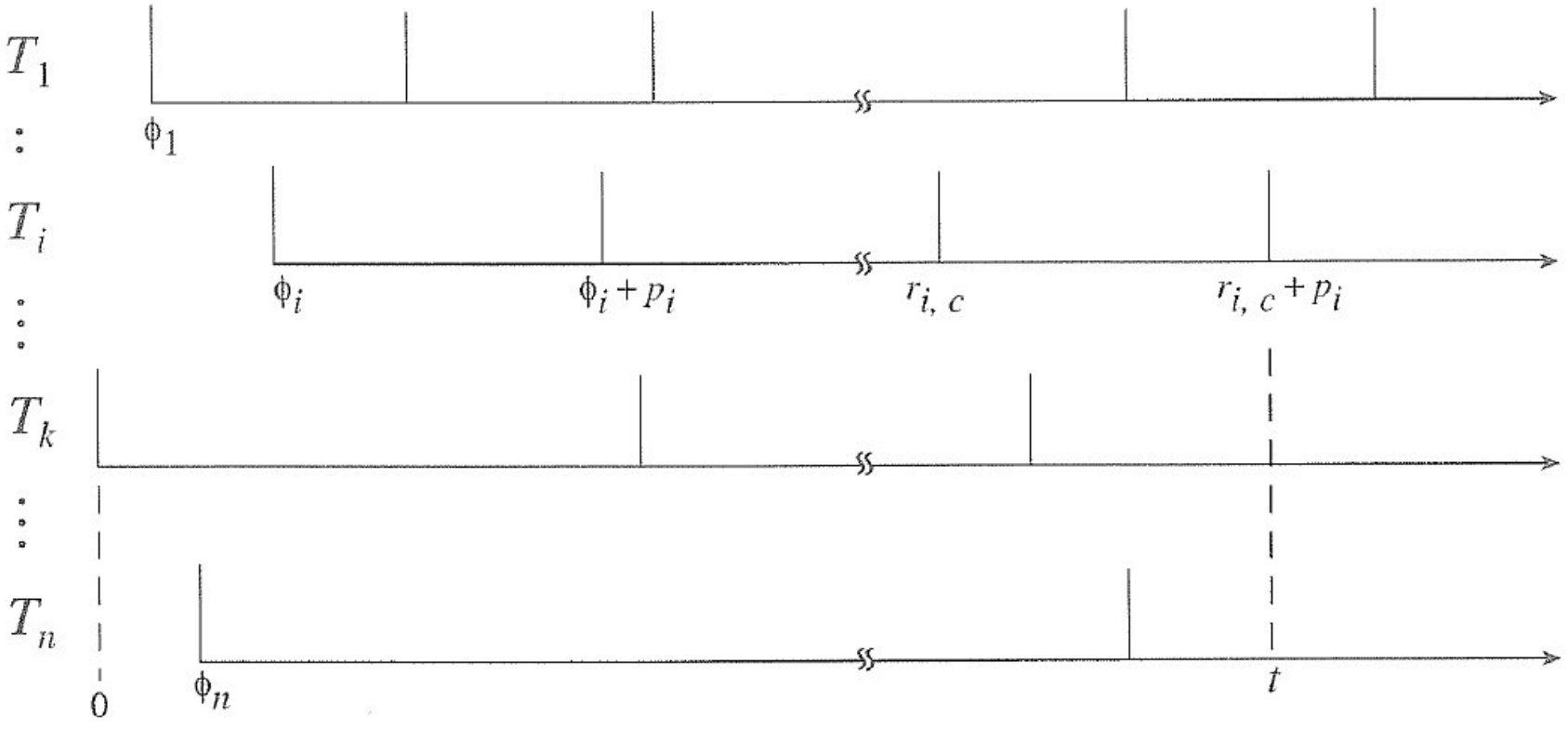
1.⇒2. We prove that $U^T > 1$ implies that T is not schedulable (whiteb.)

2.⇒3. Whiteboard ...

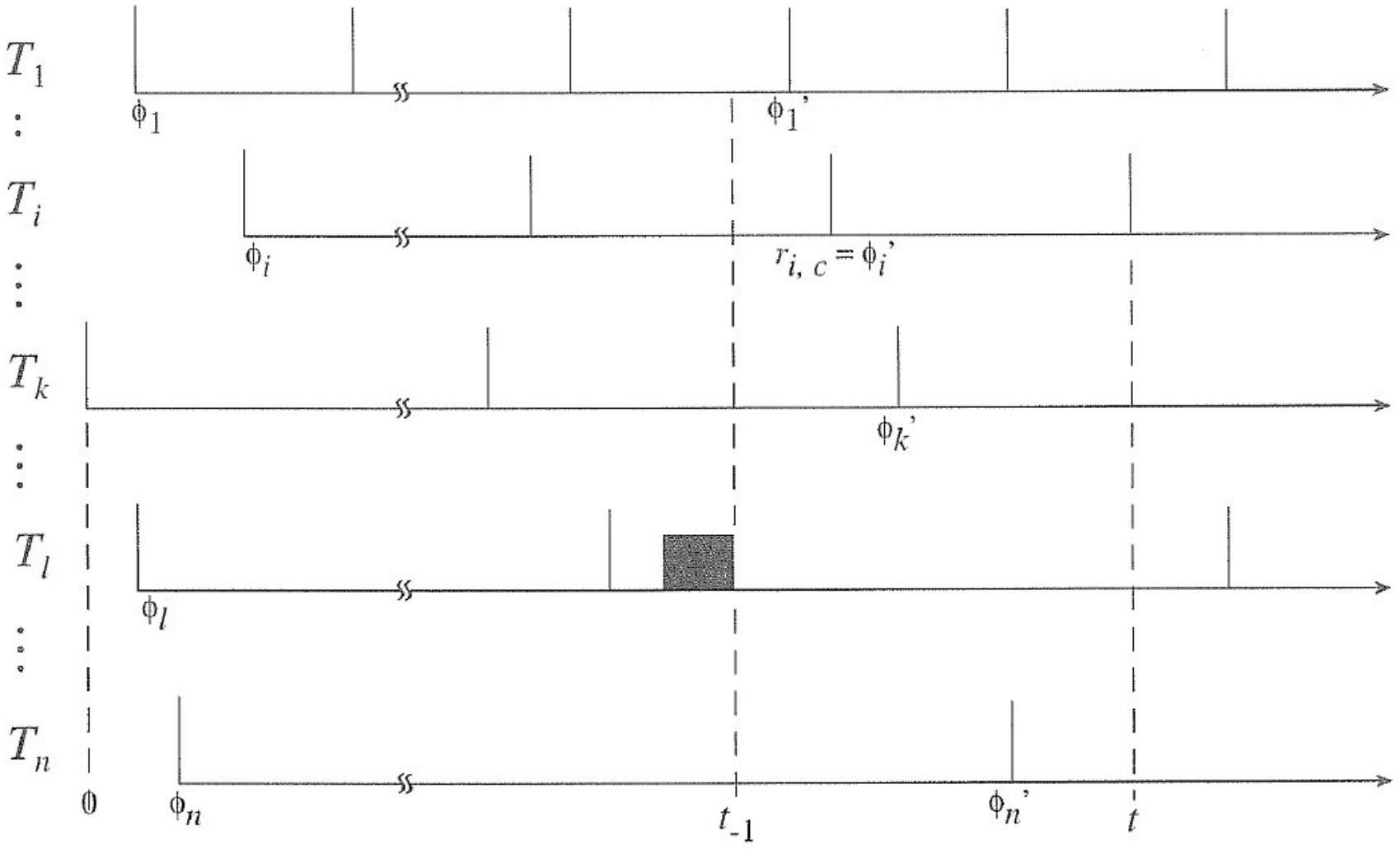
3.⇒1. Trivial

□

Optimality of EDF – proof – 2.⇒3.



Optimality of EDF – proof – 2.⇒3.



Density and EDF

What about tasks with $D_i < p_i$?

Density of a task T_i with period p_i, execution time e_i and relative deadline D_i is defined by

$$e_i / \min(D_i, p_i)$$

Total density Δ^T of a set of tasks T is the sum of densities of tasks in T

Note that if $D_i < p_i$ for some i, then $\Delta^T > U^T$

Theorem 3

A set T of independent, preemptable, periodic tasks can be feasibly scheduled on one processor if $\Delta^T \leq 1$.

Note that this is NOT a necessary condition!

Schedulability Test For EDF

The problem: Given a set of independent, preemptable, periodic tasks $T = \{T_1, \ldots, T_n\}$ where each T_i has a period p_i, execution time e_i, and relative deadline D_i, decide whether T is schedulable by EDF.

Solution using utilization and density:

If $p_i \leq D_i$ for each i, then it suffices to decide whether $U^T \leq 1$.

Otherwise, decide whether $\Delta^T \leq 1$:

- If yes, then T is schedulable with EDF
- If not, then T does not have to be schedulable

Note that

- Phases of tasks do not have to be specified
- Parameters may vary: increasing periods or deadlines, or decreasing execution times does not prevent schedulability

Schedulability Test for EDF – Example

Consider a digital robot controller

- A control-law computation
  - takes no more than 8 ms
  - the sampling rate: 100 Hz, i.e. computes every 10 ms

  Feasible? Trivially yes ....
- Add Built-In Self-Test (BIST)
  - maximum execution time 50 ms
  - want a minimal period that is feasible

  With 250 ms still feasible ....
- Add a telemetry task
  - maximum execution time 15 ms
  - want to minimize the deadline on telemetry
    period may be large

  Reducing BIST to once a second, deadline on telemetry may be set to 100 ms ....