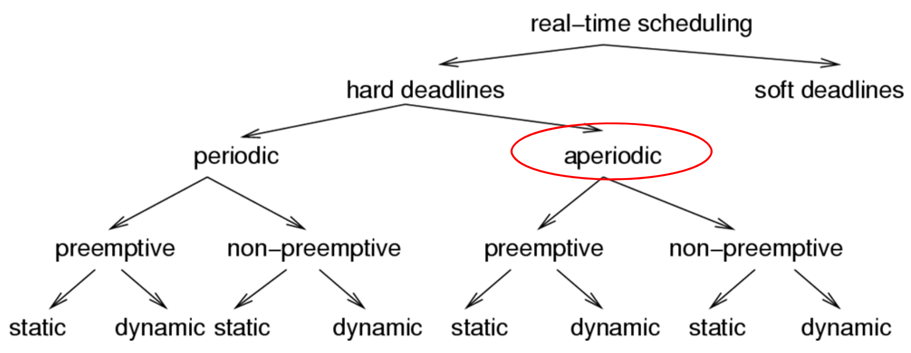
Selected Slides of Chapter 4, part 2

Embedded Operating Systems, Middleware, and Scheduling

Peter Marwedel
Informatik 12
Univ. Dortmund
Germany

2006/12/05

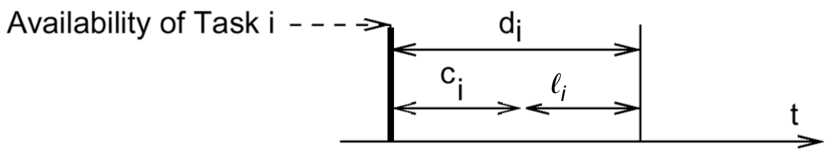
Classification of scheduling algorithms



Aperiodic scheduling - Scheduling with no precedence constraints -

Let $\{T_i\}$ be a set of tasks. Let:

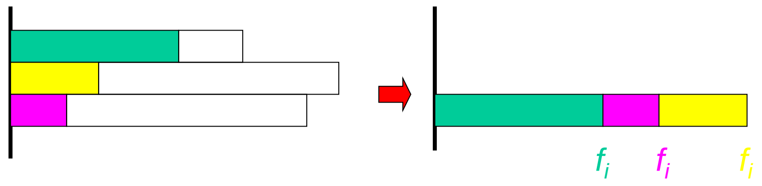
- c_i be the execution time of T_i,
- d_i be the **deadline interval**, that is, the time between T_i becoming available and the time until which T_i has to finish execution.
- ℓ_i be the **laxity** or **slack**, defined as $\ell_i = d_i - c_i$
- f_i be the finishing time.



Uniprocessor with equal arrival times

Preemption is useless.

Earliest Due Date (EDD): Execute task with earliest due date (deadline) first.



EDD requires all tasks to be sorted by their (absolute) deadlines. Hence, its complexity is $O(n \log(n))$.

Earliest Deadline First (EDF) - Horn's Theorem -

Different arrival times: Preemption potentially reduces lateness.

Theorem [Horn74]: Given a set of *n* independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks is optimal with respect to minimizing the maximum lateness.

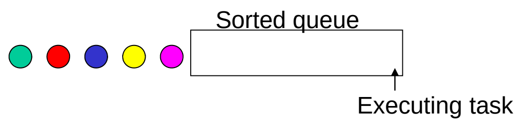
Earliest Deadline First (EDF) - Algorithm -

Earliest deadline first (EDF) algorithm:

Each time a new ready task arrives:

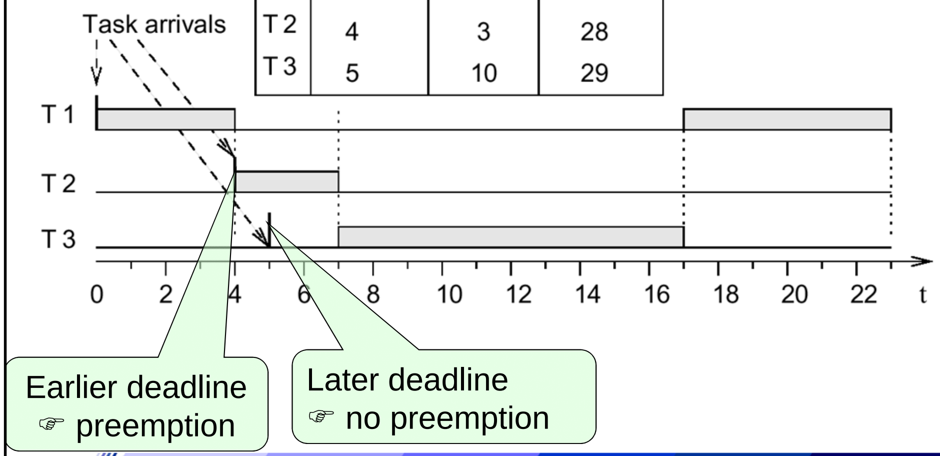
- It is inserted into a queue of ready tasks, sorted by their **absolute** deadlines. Task at head of queue is executed.
- If a newly arrived task is inserted at the head of the queue, the currently executing task is preempted.

Straightforward approach with sorted lists (full comparison with existing tasks for each arriving task) requires run-time $O(n^2)$; (less with binary search or bucket arrays).



Earliest Deadline First (EDF) - Example -

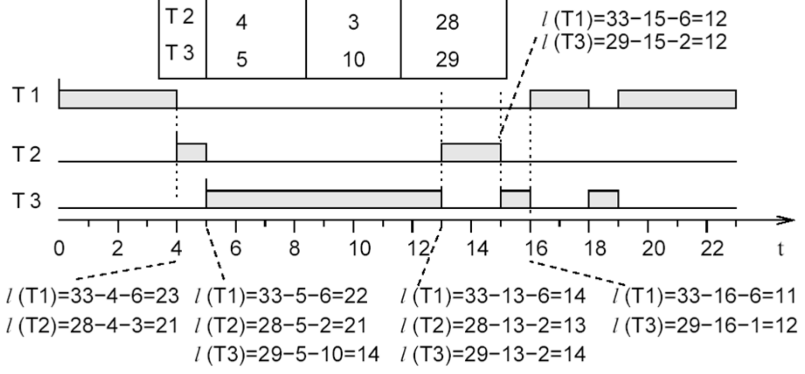
	arrival	duration	deadline
T 1	0	10	33
T 2	4	3	28
T 3	5	10	29


Task arrivals
T 1
T 2
T 3
0 2 4 6 8 10 12 14 16 18 20 22 t
Earlier deadline ☞ preemption
Later deadline ☞ no preemption

Least laxity (LL), Least Slack Time First (LST)

Priorities = decreasing function of the laxity (the less laxity, the higher the priority); dynamically changing priority; preemptive.

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



Properties

- Not sufficient to call scheduler & re-compute laxity just at task arrival times.
- Overhead for calls of the scheduler.
- Many context switches.
- Detects missed deadlines early.
- LL is also an optimal scheduling for mono-processor systems.
- Dynamic priorities ☞ cannot be used with a fixed prio OS.
- LL scheduling requires the knowledge of the execution time.

Scheduling without preemption

Lemma: If preemption is not allowed, optimal schedules may have to leave the processor idle at certain times.

Proof: Suppose: optimal schedulers never leave processor idle.

Scheduling without preemption (2)

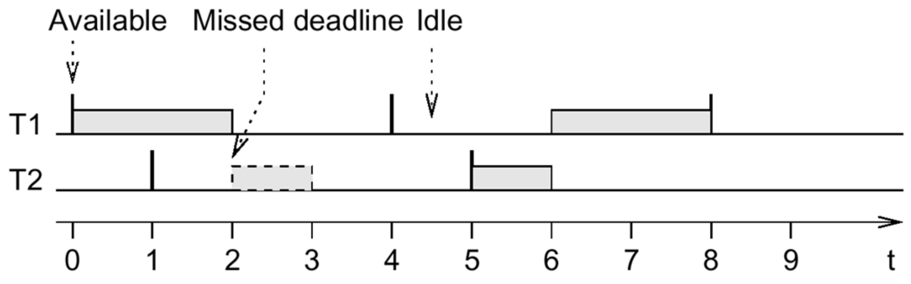
T1: periodic, $c_1 = 2$, $p_1 = 4$, $d_1 = 4$

T2: occasionally available at times $4*n+1$, $c_2= 1$, $d_2= 1$

T1 has to start at t=0

☞ deadline missed, but schedule is possible (start T2 first)

☞ scheduler is not optimal ☞ contradiction! q.e.d.



Scheduling without preemption

Preemption not allowed: ☞ optimal schedules may leave processor idle to finish tasks with early deadlines arriving late.

☞Knowledge about the future is needed for optimal scheduling algorithms

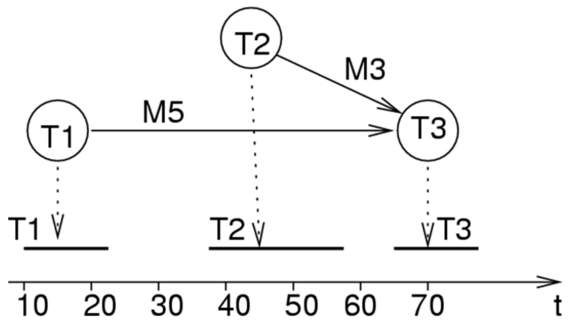
☞No online algorithm can decide whether or not to keep idle.

EDF is optimal among all scheduling algorithms not keeping the processor idle at certain times.

If arrival times are known a priori, the scheduling problem becomes NP-hard in general. B&B typically used.

Scheduling with precedence constraints

Task graph and possible schedule:



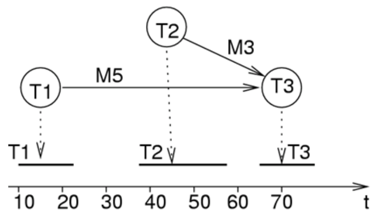
Schedule can be stored in table.

Simultaneous Arrival Times: The Latest Deadline First (LDF) Algorithm

LDF [Lawler, 1973]: reads the task graph and among the tasks with no successors inserts the one with the latest deadline into a queue. It then repeats this process, putting tasks whose successor have all been selected into the queue.

At run-time, the tasks are executed in the generated total order.

LDF is non-preemptive and is optimal for mono-processors.



If no local deadlines exist, LDF performs just a topological sort.

Asynchronous Arrival Times: Modified EDF Algorithm

This case can be handled with a modified EDF algorithm. The key idea is to transform the problem from a given set of dependent tasks into a set of independent tasks with different timing parameters [Chetto90].

This algorithm is optimal for mono-processor systems.

If preemption is not allowed, the heuristic algorithm developed by Stankovic and Ramamritham can be used.

Summary

Worst case execution times (WCET)
Definition of scheduling terms
- Hard vs. soft deadlines
- Static vs. dynamic ☞TT-OS
- Schedulability

Scheduling approaches
- – Aperiodic tasks
  - No precedences
    - – Simultaneous (☞EDD) & Asynchronous Arrival Times (☞EDF, LL)
  - Precedences
    - – Simultaneous Arrival Times (☞ LDF)
    - – Asynchronous Arrival Times (☞ mEDF)